`profilerTools`
User Manual

Corresponding to Version 1.0

MSSM Group
Maintained by Yan M. H. Gonçalves

Citation Information

If the `profilerTools` suite was helpful in achieving your results, we kindly ask you to cite it in your work as

- Y. M. H. Gonçalves, S. Kashefolgheta, M. P. Oliveira, P. H. Hünenberger, and B. A. C. Horta. Simultaneous parametrization of torsional and third-neighbor interaction terms in force-field development: the LLS-SC algorithm. *Submitted*, 2021

License

Contact

For bug reports or other software-related questions, send an e-mail to `yanmarques@gmail.com`. More information about our research group can be found on the webpage `https://labmmol.iq.ufrj.br/mssm`.

Contents

Citation Information . . . i
License . . . i
Contact . . . i

Contents **i**

1 Introduction **1**

2 Molecular Mechanics and Energy Minimization **2**
2.1 Force-field Functional Forms . . . 2
2.2 Calculation of Forces . . . 6
2.3 Energy Minimization . . . 6
2.4 Limitations . . . 7

3 Parameter Optimization **8**
3.1 Torsional-scan Energy Profiles . . . 8
3.2 Parameter Types . . . 9
3.3 From Trial Parameters to Force-field Terms . . . 9
3.4 Reference Data and Weights . . . 9
3.5 Evolutionary Algorithms . . . 9
3.6 Self-consistent Linear Least-Squares Optimization . . . 11

4 Input and Output File Formats **12**
4.1 File Formats . . . 12

5 Tutorial **21**
5.1 Obtaining an STP File . . . 21
5.2 Generating Torsional-Scan Trajectories . . . 22
5.3 Optimizing the Parameters . . . 24

Bibliography **29**

A MIT License **30**

Chapter 1

Introduction

Dihedral-angle torsion parameters are typically optimized using quantum-mechanical relative conformational energies as target data. However, confomational energies are also affected by the 1,4 nonbonded interactions. This suggests that these two types of force-field terms should ideally be parameterized simultaneously. With this in mind, we developed `profilerOpt`, a Python program that allows this parametrization based on several molecules and using one of several different strategies: (*i*) a genetic algorithm (GA); (*ii*) a covariance matrix adaptation evolution strategy (CMA-ES); (*iii*) a linear least-squares self-consistent (LLS-SC) strategy. Together with `profilerGen`, an auxiliary Python program to execute multidimensional molecular torsional scans, these two form the `profilerTools` suite. The problem of simultaneously parameterizing the torsional and 1,4 terms is discussed in details in the `profilerTools` paper,[3] specially the LLS-SC strategy.

More succinctly, `profilerTools` is composed of two programs:

- `profilerGen`, for obtaining torsional-scan trajectories and their energy profiles;
- `profilerOpt`, for the parameterization of torsional terms and the related 1,4 Lennard-Jones terms;

The two evolutionary algorithms (GA and CMA-ES) are based on the DEAP library.[2] The LLS-SC implementation is based on the `numpy` and `scikit-learn` libraries, as well as on the GNU Scientific Library. `profilerTools` does not rely on external software for carrying out molecular-mechanics calculations and energy minimizations. They are carried out based on `numpy` and on a C extension to `numpy` called `geometrypy`, specifically implemented for `profilerTools`. This extension accelerates the calculation of the values of bond lengths, bond angles, dihedral angles, etc., by transferring the loops over the involved atoms to the underlying `C` implementation, instead of carrying out these loops in Python (which is significantly slower). The mathematical formulae and the supported physical models are described in Chapter 2.

Chapter 2

Molecular Mechanics and Energy Minimization

In `profilerTools`, molecular-mechanics energies and forces are calculated using custom code which depends only on standard Python libraries and `numpy`. In this chapter, we give details about the computation of all quantities which directly involve these calculations. First, we report all force-field functional forms available in the suite. After that, we give the expressions for the calculation of forces. We then describe the energy minimization algorithms implemented in the programs. Finally, we define the central object of the molecular-mechanics calculations of `profilerTools`, namely, the torsional scan and its corresponding energy profile.

Considering a system of N particles, the following notation and definitions will be used for mathematical and physical quantities whenever convenient:

— $\mathbf{u} \cdot \mathbf{v}$
Dot product of vectors $\mathbf{u}$ and $\mathbf{v}$.

— $\mathbf{u} \times \mathbf{v}$
Cross product of vectors $\mathbf{u}$ and $\mathbf{v}$.

— $\mathbf{r}_i$
Position of particle i.

— $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$
Displacement from particle j to particle i.

— $r_{ij} = \sqrt{\mathbf{r}_{ij} \cdot \mathbf{r}_{ij}}$
Distance between particle i and particle j.

— $\theta_{ijk} = \arccos \frac{\mathbf{r}_{ij} \cdot \mathbf{r}_{kj}}{r_{ij} r_{kj}}$
Angle involving particles i, j and k.

— $\phi_{ijkl} = \text{sign}\,(\mathbf{r}_{ij} \cdot \mathbf{r}_{nk}) \arccos\left(\frac{\mathbf{r}_{mj} \cdot \mathbf{r}_{nk}}{r_{mj} r_{nk}}\right)$, with $\mathbf{r}_{mj} = \mathbf{r}_{ij} \times \mathbf{r}_{kj}$ and $\mathbf{r}_{nk} = \mathbf{r}_{kj} \times \mathbf{r}_{kl}$
Dihedral angle involving particles i, j, k and l.

— $\mathbf{r}(t_n) = (\mathbf{r}_1(t_n), \mathbf{r}_2(t_n) \ldots \mathbf{r}_N(t_n))$
Configuration at the n-th step of the energy-minimization algorithm.

— $\mathbf{f}(t_n) = (\mathbf{f}_1(t_n), \mathbf{f}_2(t_n) \ldots \mathbf{f}_N(t_n))$
Configuration of the forces at the n-th step of the energy-minimization algorithm.

2.1 Force-field Functional Forms

2.1.1 Bond stretching

The following functional forms are supported for the calculation of the energies and forces associated with a bond-stretching degree of freedom:[1,4]

— *Harmonic*
Potential energy form:

$$V(r_{ij}; k_b^h, b_0) = \frac{1}{2}\, k_b^h (r_{ij} - b_0)^2 \quad , \tag{2.1.1}$$
$$\frac{\partial V(r_{ij}; k_b^h, b_0)}{\partial r_{ij}} = \; k_b^h (r_{ij} - b_0) \quad ,$$

— *Quartic*

Potential energy form:

$$V(r_{ij}; k_b^q, b_0) = \frac{1}{4}\ k_b^q (r_{ij}^2 - b_0^2)^2 \quad , \tag{2.1.2}$$
$$\frac{\partial V(r_{ij}; k_b^q, b_0)}{\partial r_{ij}} = r_{ij} k_b^q (r_{ij}^2 - b_0^2) \quad ,$$

where $V(r_{ij})$ is the potential energy term for the bond between particles i and j.

2.1.2 Bond-angle stretching

The following functional forms are supported for the calculation of the energies and forces associated with a bond-angle stretching degree of freedom:[1,4]

— *Harmonic*

Potential energy form:

$$V(\theta_{ijk}; k_a^h, \theta_0) = \frac{1}{2}\ k_a^h (\theta_{ijk} - \theta_0)^2 \quad , \tag{2.1.3}$$
$$\frac{\partial V(\theta_{ijk}; k_a^h, \theta_0)}{\theta_{ijk}} = k_a^h (\theta_{ijk} - \theta_0) \quad ,$$

— *Cosine-harmonic*

Potential energy form:

$$V(\theta_{ijk}; k_a^c, \theta_0) = \frac{1}{2}\ k_a^c (\cos\theta_{ijk} - \cos\theta_0)^2 \quad , \tag{2.1.4}$$
$$\frac{\partial V(\theta_{ijk}; k_a^c, \theta_0)}{\theta_{ijk}} = -k_a^c \sin\theta_{ijk} (\cos\theta_{ijk} - \cos\theta_0) \quad ,$$

— *Urey-Bradley*

Potential energy form:

$$V(\theta_{ijk}, r_{ik}; k_a^{ub}, \theta_0, k_{13}^{ub}, b_{13}^{ub}) = \frac{1}{2}\ k_a^{ub} (\theta_{ijk} - \theta_0)^2 + \frac{1}{2}\ k_{13}^{ub} (r_{ik} - b_{13}^{ub})^2 \quad , \tag{2.1.5}$$
$$\frac{\partial V(\theta_{ijk}, r_{ik}; k_a^{ub}, \theta_0, k_{13}^{ub}, b_{13}^{ub})}{\partial \theta_{ijk}} = k_a^{ub} (\theta_{ijk} - \theta_0) \quad ,$$
$$\frac{\partial V(\theta_{ijk}, r_{ik}; k_a^{ub}, \theta_0, k_{13}^{ub}, b_{13}^{ub})}{\partial r_{ik}} = k_{13}^{ub} (r_{ik} - b_{13}^{ub}) \quad ,$$

where $V(\theta_{ijk}, [r_{ik}])$ is the potential energy term for the angle involving particles i, j and k. Note that the Urey-Bradley form introduces dependence on an additional degree of freedom, the distance r_{ik}, which is treated as a harmonic bond between particles i and k.

2.1.3 Proper dihedral-angle torsion

The following functional forms are supported for the calculation of the energies and forces associated with a proper dihedral-angle torsion degree of freedom:[1,4]

— *Standard periodic*

Potential energy form:

$$V(\phi_{ijkl}; \{k_{d,m}^s, \phi_{0,m}\}_{m=1..6}) = \sum_{m=1}^{6} k_{d,m}^s\ (1 + \cos(m\phi_{ijkl} - \phi_{0,m})) \quad , \tag{2.1.6}$$
$$\frac{\partial V(\phi_{ijkl}; \{k_{d,m}^s, \phi_{0,m}\}_{m=1..6})}{\partial \phi_{ijkl}} = -\sum_{m=1}^{6} m k_{d,m}^s \sin(m\phi_{ijkl} - \phi_{0,m}) \quad ,$$

with $\phi_{0,m} \in (-\pi, \pi]$.

— *Fourier*

Potential energy form:

$$V(\phi_{ijkl};\{f_m\}_{m=1..4}) = \frac{1}{2}\sum_{m=1}^{4} f_m\left[1+(-1)^{m+1}\cos m\phi_{ijkl}\right] \quad , \tag{2.1.7}$$
$$\frac{\partial V(\phi_{ijkl};\{f_m\}_{m=1..4})}{\partial \phi_{ijkl}} = \frac{1}{2}\sum_{m=1}^{4} m f_m (-1)^m \sin m\phi_{ijkl} \quad ,$$

— *Harmonic restraint*

Potential energy form:

$$V(\phi_{ijkl};k_d^r,\phi_0) = \frac{1}{2}k_d^r(\phi_{ijkl}-\phi_0)^2 \quad , \tag{2.1.8}$$
$$\frac{\partial V(\phi_{ijkl};k_d^r,\phi_0)}{\partial \phi_{ijkl}} = k_d^r(\phi_{ijkl}-\phi_0) \quad ,$$

where $V(\phi_{ijkl})$ is the potential energy term for the proper dihedral angle involving particles i, j, k and l.

2.1.4 Improper dihedral-angle bending

The following functional forms are supported for the calculation of the energies and forces associated with an improper dihedral-angle bending degree of freedom:[1,4]

— *Harmonic*

Potential energy form:

$$V(\phi_{ijkl};k_i^h,\phi_0) = \frac{1}{2}k_i^h(\phi_{ijkl}-\phi_0)^2 \quad , \tag{2.1.9}$$
$$\frac{\partial V(\phi_{ijkl};k_i^h,\phi_0)}{\phi_{ijkl}} = k_i^h(\phi_{ijkl}-\phi_0) \quad ,$$

— *Periodic* Potential energy form:

$$V(\phi_{ijkl};\{k_i^p,\phi_{0,m}\}_{m=1..6}) = \sum_{m=1}^{6} k_i^s\ (1+\cos(m\phi_{ijkl}-\phi_{0,m})) \quad , \tag{2.1.10}$$
$$\frac{\partial V(\phi_{ijkl};\{k_i^s,\phi_{0,m}\}_{m=1..6})}{\partial \phi_{ijkl}} = -\sum_{m=1}^{6} m k_i^s \sin(m\phi_{ijkl}-\phi_{0,m}) \quad ,$$

with $\phi_{0,m} \in (-\pi,\pi]$.

where $V(\phi_{ijkl})$ is the potential energy term for the improper dihedral angle involving particles i, j, k and l.

2.1.5 Nonbonded interactions

Standard interactions

The nonbonded interactions are calculated for every pair of non-excluded particles, without applying any type of cutoff-based truncation. The electrostatic and van der Waals interactions are calculated based on the following functional forms, respectively:[1,4]

— *Coulomb*

$$V(r_{ij};q_i,q_j) = \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \quad , \tag{2.1.11}$$
$$\frac{\partial V(r_{ij};q_i,q_j)}{\partial r_{ij}} = -\frac{q_i q_j}{4\pi\epsilon_0 r_{ij}^2} \quad ,$$

— *Lennard-Jones*

$$V(r_{ij};C_{6,ij},C_{12,ij}) = \frac{C_{12,ij}}{r_{ij}^{12}} - \frac{C_{6,ij}}{r_{ij}^{6}} \quad , \tag{2.1.12}$$
$$\frac{\partial V(r_{ij};C_{6,ij},C_{12,ij})}{r_{ij}} = -\frac{6}{r_{ij}^{8}}\left(\frac{2C_{12,ij}}{r_{ij}^{6}} - C_{6,ij}\right) \quad ,$$

where the interaction between particles i and j is considered.

When the pair-specific $C_{12,ij}$ and $C_{6,ij}$ Lennard-Jones parameters are not explicitly defined in the topology they are obtained from atomic counterparts, using one of the following mixing rules:

— *Geometric*

$$C_{6,ij} = \sqrt{C_{6,i}C_{6,j}} \quad , \quad C_{12,ij} = \sqrt{C_{12,i}C_{12,j}} \tag{2.1.13}$$

— *Lorentz-Berthelot*

$$C_{6,ij} = \frac{C_{6,i}C_{6,j}}{2^6\sqrt{C_{12,i}C_{12,j}}}\left[\left(\frac{C_{12,i}}{C_{6,i}}\right)^{1/6} + \left(\frac{C_{12,j}}{C_{6,j}}\right)^{1/6}\right]^6 \tag{2.1.14}$$

$$C_{12,ij} = \frac{C_{6,i}C_{6,j}}{2^{12}\sqrt{C_{12,i}C_{12,j}}}\left[\left(\frac{C_{12,i}}{C_{6,i}}\right)^{1/6} + \left(\frac{C_{12,j}}{C_{6,j}}\right)^{1/6}\right]^{12} \quad .$$

Some force fields are specified in terms of the σ-ϵ representation of the Lennard-Jones potential:

— *Lennard-Jones* (σ-ϵ)

$$V(r_{ij};\sigma_{ij},\epsilon_{ij}) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^6 - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12}\right] \ , \ \sigma_{ij} = (C_{12,ij}/C_{6,ij})^{1/6} \ , \ \epsilon_{ij} = C_{6,ij}^2/4C_{12,ij} \tag{2.1.15}$$

with corresponding mixing rules

— *Geometric*

$$\sigma_{ij} = \sqrt{\sigma_i\sigma_j} \quad , \quad \epsilon_{ij} = \sqrt{\epsilon_i\epsilon_j} \tag{2.1.16}$$

— *Lorentz-Berthelot*

$$\sigma_{ij} = \frac{1}{2}(\sigma_i + \sigma_j) \quad , \quad \epsilon_{ij} = \sqrt{\epsilon_i\epsilon_j} \quad . \tag{2.1.17}$$

When this is the case, the nonbonded parameters are internally converted to the C_6-C_{12} representation, with which all molecular-mechanics calculations are performed.

1–4 interactions

The interactions of pairs of particles that are identified as sharing a third-neighbor relationship (1–4 interactions) are treated in a different manner. First, all 1–4 Coulomb interactions are scaled down from their standard values by a common factor f_{QQ} as

$$V(r_{ij};q_i,q_j) = f_{QQ}\ \frac{q_iq_j}{4\pi\epsilon_0 r_{ij}} \quad , \quad 0 \le f_{QQ} \le 1 \ . \tag{2.1.18}$$

For the 1–4 Lennard-Jones interactions, the same Lennard-Jones potential form is used, but with a special set of parameters $CS_{6,ij}, CS_{12,ij}$:

$$V(r_{ij};CS_{6,ij},CS_{12,ij}) = \frac{CS_{12,ij}}{r_{ij}^{12}} - \frac{CS_{6,ij}}{r_{ij}^6} \quad . \tag{2.1.19}$$

For the σ-ϵ representation, the same observations as the standard interactions apply.

When the 1–4 pair-specific nonbonded parameters $CS_{6,ij}, CS_{12,ij}$ (or their $\sigma_{S,ij}, \epsilon_{S,ij}$ equivalents) are not explicitly defined in the topology they are obtained from atomic parameters in one of two ways. The first is more common and establishes a simple mathematical rule for computing them. The second is more uncommon (but is used *e.g.* in GROMOS force fields) and relies on defining additional atomic Lennard-Jones parameters to use exclusively with 1–4 interactions.

— First strategy (*fudge*): Scale down the standard atomic parameters $C_{6,\alpha}, C_{12,\alpha}$ ($\alpha = i,j$) by a common factor f_{LJ}, obtaining 1–4 atomic parameters $CS_{6,\alpha} = f_{LJ}C_{6,\alpha}$ and $CS_{12,\alpha} = f_{LJ}C_{12,\alpha}$ ($\alpha = i,j$). After that, proceed as usual, obtaining the 1–4 pair parameters $CS_{12,ij}$ and $CS_{6,ij}$ *via* mixing rule from $CS_{6,\alpha}, CS_{12,\alpha}$ ($\alpha = i,j$) and computing the scaled-down potential energy:

$$V(r_{ij};C_{6,ij},C_{12,ij}) = \frac{CS_{12,ij}}{r_{ij}^{12}} - \frac{CS_{6,ij}}{r_{ij}^6} = f_{LJ}\left(\frac{C_{12,ij}}{r_{ij}^{12}} - \frac{C_{6,ij}}{r_{ij}^6}\right) \quad , \quad 0 \le f_{LJ} \le 1 \tag{2.1.20}$$

— Second strategy (*atomic*): Instead of using a fixed factor f_{LJ} to modify the atomic parameters, specify an independent additional set of atomic parameters $CS_{6,\alpha}$ and $CS_{12,\alpha}$ ($\alpha = i,j$) which are exclusive to 1–4 interactions. The 1–4 pair parameters $CS_{12,ij}$ and $CS_{6,ij}$ are then obtained *via* mixing rule from $CS_{6,\alpha}, CS_{12,\alpha}$ ($\alpha = i,j$), resulting in a pair-specific scaled-down potential energy.

2.1.6 Choice of Functional Forms

The functional forms used for each interaction term are selected in the input STP file via type codes. Likewise, the Lennard-Jones potential representation, mixing rule and 1–4 details are specified in this same file. For more information, read about the STP file format in Section 4.1.1.

2.2 Calculation of Forces

The atomic forces due to the potential energy terms reported above are obtained by computing the analytical derivatives of the corresponding expressions with respect to the atomic positions involved in the degree of freedom. They are:

— *Bond terms and nonbonded terms*

$$\mathbf{f}_\alpha = -\frac{\partial V(r_{ij})}{\partial r_{ij}} \nabla_{\mathbf{r}_\alpha} r_{ij} \quad , \quad \alpha = i, j \quad , \tag{2.2.1}$$

— *Angle terms*

$$\mathbf{f}_\alpha = -\frac{\partial V(\theta_{ijk})}{\partial \theta_{ijk}} \nabla_{\mathbf{r}_\alpha} \theta_{ijk} \quad , \quad \alpha = i, j, k \quad , \tag{2.2.2}$$

— *Dihedral terms*

$$\mathbf{f}_\alpha = -\frac{\partial V(\phi_{ijkl})}{\partial \phi_{ijkl}} \nabla_{\mathbf{r}_\alpha} \phi_{ijkl} \quad , \quad \alpha = i, j, k, l \quad , \tag{2.2.3}$$

where $\mathbf{f}_\alpha$ is the force acting on particle α and $\nabla_{\mathbf{r}_\alpha}$ denotes the gradient with respect to the position of particle α. Each expression above is the product of a straightforward uni-dimensional derivative involving the potential expression with a more complicated "geometrical" derivative. The former terms are reported in the Sections above for each functional form. The latter terms are listed below:[4]

— *Bond terms and nonbonded terms*

$$\begin{aligned} \nabla_{\mathbf{r}_i} r_{ij} &= +\frac{\mathbf{r}_{ij}}{r_{ij}} \\ \nabla_{\mathbf{r}_j} r_{ij} &= -\frac{\mathbf{r}_{ij}}{r_{ij}} \end{aligned} \tag{2.2.4}$$

— *Angle terms*

$$\begin{aligned} \nabla_{\mathbf{r}_i} \theta_{ijk} &= +\frac{1}{r_{ij}} \left(\frac{\mathbf{r}_{kj}}{r_{kj}} - \frac{\mathbf{r}_{ij}}{r_{ij}} \cos\theta_{ijk} \right) \\ \nabla_{\mathbf{r}_j} \theta_{ijk} &= -\nabla_{\mathbf{r}_i} \theta_{ijk} - \nabla_{\mathbf{r}_k} \theta_{ijk} \\ \nabla_{\mathbf{r}_k} \theta_{ijk} &= +\frac{1}{r_{kj}} \left(\frac{\mathbf{r}_{ij}}{r_{ij}} - \frac{\mathbf{r}_{kj}}{r_{kj}} \cos\theta_{ijk} \right) \end{aligned} \tag{2.2.5}$$

— *Dihedral terms*

$$\begin{aligned} \nabla_{\mathbf{r}_i} \phi_{ijkl} &= +\frac{r_{kj}}{r_{mj}^2} \mathbf{r}_{mj} \\ \nabla_{\mathbf{r}_j} \phi_{ijkl} &= +\left[\frac{\mathbf{r}_{ij} \cdot \mathbf{r}_{kj}}{r_{kj}^2} - 1 \right] \nabla_{\mathbf{r}_i} \phi_{ijkl} - \frac{\mathbf{r}_{kl} \cdot \mathbf{r}_{kj}}{r_{kj}^2} \nabla_{\mathbf{r}_l} \phi_{ijkl} \\ \nabla_{\mathbf{r}_k} \phi_{ijkl} &= -\nabla_{\mathbf{r}_i} \phi_{ijkl} - \nabla_{\mathbf{r}_j} \phi_{ijkl} - \nabla_{\mathbf{r}_l} \phi_{ijkl} \\ \nabla_{\mathbf{r}_l} \phi_{ijkl} &= +\frac{r_{kj}}{r_{nk}^2} \mathbf{r}_{nk} \quad , \end{aligned} \tag{2.2.6}$$

where

$$\mathbf{r}_{mj} = \mathbf{r}_{ij} \times \mathbf{r}_{kj} \ , \ \mathbf{r}_{nk} = \mathbf{r}_{kj} \times \mathbf{r}_{kl} \quad .$$

2.3 Energy Minimization

In this section, we briefly describe the energy minimization algorithm implemented in `profilerTools`, namely *steepest-descents*. The implementation is based on the one described in the GROMOS user manual.[4]

2.3.1 Steepest-descents

In the steepest-descents energy minimization algorithm, one successively updates the configuration by moving along the direction of the gradient of the potential energy over the configurational space. The n-th step of the steepest-descents energy minimization algorithm is as follows:

1. Calculate $\mathbf{f}(t_n)$ from the configuration $\mathbf{r}(t_n)$.
2. Compute the next configuration from

$$\mathbf{r}(t_{n+1}) = \mathbf{r}(t_n) + \frac{\Delta x(t_n)}{\sqrt{\sum_{i=1}^{N} \mathbf{f}_i(t_n) \cdot \mathbf{f}_i(t_n)}}\, \mathbf{f}(t_n) \quad . \tag{2.3.1}$$

The initial value of the step size $\Delta x(t_n)$ is specified in the input variable DX0. If the energy decreases in a given step n, $\Delta x(t_{n+1}) = \min\{1.2 \cdot \Delta x(t_n),\ \text{DXM}\}$, where DXM is an input variable. If the energy increases, $\Delta x(t_{n+1}) = 0.5 \cdot \Delta x(t_n)$. The algorithm is terminated if the number of steps reaches the value specified in the input variable NMAX or if the (absolute) energy difference between successive steps is less than the one specified in the input variable DELE.

2.4 Limitations

— Periodic boundary conditions are currently not taken into account. Please make sure your systems fit entirely within the central simulation box to avoid any problems.

— The nonbonded interactions are computed explicitly regardless of the distance between the interacting particles. No other scheme for computing the nonbonded interactions is currently supported.

— For a given system, only one functional form is accepted for each type of bonded interaction. For example, if one bond-stretching degree of freedom of the system is described by the harmonic form, then all of them also must be. The only exception is for proper dihedral restraints, which can be defined along with any other form of proper dihedral.

Chapter 3

Parameter Optimization

In this chapter, we describe the structure of the parameter optimization, which can be performed based on a genetic algorithm (GA), a covariance matrix adaptation evolution strategy (CMA-ES) or the linear least-squares self-consistent method (LLS-SC) described in Ref.[3]

3.1 Torsional-scan Energy Profiles

The central objects of all molecular-mechanics calculations of `profilerTools` are the (multidimensional) torsional scans and the corresponding energy profiles. We define a *torsional scan* as a sequence of concatenated molecular configurations (*i.e.* a trajectory) for a system s where one or more dihedral angles [the *reference dihedrals* (RDs)] are successively restrained to pre-defined values (the *torsional-scan angles*), while all other internal degrees of freedom are relaxed. The number of RDs, $N_D^{(s)}$, is the *dimension of the torsional scan*. The number of configurations (the number of frames in the trajectory), $N_K^{(s)}$, is the *length of the torsional scan*. In the most general case, the torsional-scan angles can be specified *via* a $N_K^{(s)} \times N_D^{(s)}$ real matrix,

$$\Phi_s = \begin{bmatrix} \phi_{11}^{(s)} & \dots & \phi_{1N_D}^{(s)} \\ \vdots & \ddots & \vdots \\ \phi_{N_K1}^{(s)} & \dots & \phi_{N_KN_D}^{(s)} \end{bmatrix} \quad , \tag{3.1.1}$$

where the row vector $\boldsymbol{\phi}_k^{(s)} = (\phi_{k1}^{(s)}, \dots, \phi_{kN_D}^{(s)})$ describes the values of the RDs for the k-th torsional-scan configuration.

The RDs can be partitioned into N_R disjoint and possibly empty groups $\mathcal{R}_1^{(s)} \cup \mathcal{R}_2^{(s)} \cup \ldots \cup \mathcal{R}_{N_R}^{(s)}$ with similar dihedral-restraint settings (*e.g.*, each $\mathcal{R}_i^{(s)}$ may be associated with a particular harmonic force constant). We use the notation $\rho_i^{(s)}$ to refer to the reference-dihedral group of the i-th RD. In practice, this partitioning is performed in the STP file by listing the RDs of each group in separate `[ refdihedrals ]` blocks (see Section 4.1.1). The number of reference-dihedral groups N_R *must be the same* for all systems considered in a particular `profilerOpt` run—this is why empty groups are accepted. This is because the dihedral-restraint attributes of each group are configured in a single file that takes into account all systems simultaneously (the INP file; see Section 4.1.2).

The sequence of the unrestrained potential energies of the configurations of the torsional scan is the *torsional-scan energy profile* (or simply *energy profile*). All energy profiles in `profilerTools` are shifted (*i.e.* each energy of the sequence is subtracted by the same value) so that the average value is equal to 0. In practice, the torsional-scan energy profile $\mathcal{V}_s$ for a given system s is obtained with the following scheme:

1. For each torsional-scan configuration k:
   a. For each RD i,
      i. Apply a dihedral restraint with a reference angle of $\phi_{ki}^{(s)}$ and a force constant defined by the group to which the RD belongs;
   b. Perform a restrained energy minimization;
   c. Compute the value of the total potential energy subtracted by the dihedral-restraint energy (*i.e.* the unrestrained potential energy), referred to as $\mathcal{V}_{sk}$;
2. Subtract the unrestrained potential energies $\mathcal{V}_{sk}$ by their average over the torsional scan:

$$\mathcal{V}_s = \left(\mathcal{V}_{sk} - \langle \mathcal{V}_{sk} \rangle_k, \; k = 1 \dots N_K^{(s)} \right) \quad . \tag{3.1.2}$$

The torsional-scan matrix for each system s, Φ_s, can be set in three different ways:

1. From a systematic scan: In this case, each i-th reference-dihedral group $\mathcal{R}_i^{(s)}$ corresponds to a system-independent vector of evenly-spaced dihedral-angle values from RFRST[I] to RLST[I] in steps of RSTEP[I]:

$$\mathcal{S}(\mathcal{R}_i^{(s)}) = (\text{RFRST[I]}, \text{RFRST[I]} + \text{RSTEP[I]}, \ldots, \text{RLST[I]}) \quad , \quad s = 1 \ldots N_S$$

   The k-th row vector of Φ_s, $\boldsymbol{\phi}_k^{(s)}$, is the k-th component of $\mathcal{S}(\rho_1^{(s)}) \times \mathcal{S}(\rho_2^{(s)}) \times \ldots \times \mathcal{S}(\rho_{N_D}^{(s)})$, where "$\times$" denotes the cartesian product. For `profilerOpt`, the torsional-scan angles along each dimension are configured in the INP file (see Section 4.1.2). For `profilerGen`, they are configured in the command-line options.

2. From the input trajectory: In this case, the matrix Φ_s is filled with the values of the RDs calculated from the input trajectory for the system.

3. From a dihedral-angle specification file: In this case, the user supplies a file with the matrix Φ_s (the SPEC file; see Section 4.1.3).

3.2 Parameter Types

`profilerTools` relies on the definition of multiple *parameter types*. Each parameter type corresponds either to one torsional term with all its summands (standard periodic or Fourier) or to one Lennard-Jones term with both attractive and repulsive components. `profilerTools` admits a fairly flexible setup of boundary conditions to optimize these parameters — most components of a parameter type can be individually included or ignored in the optimization. For example, it is possible to carry out a `profilerOpt` run where only CS_{12} and odd-multiplicity terms of a standard proper dihedral form are taken into account. Currently, only the optimization of torsional phase-shift terms cannot be toggled independently — they are either optimized together with the corresponding force constants or not optimized at all.

The definition of the parameter types is performed in the INP input file, as described in Section 4.1.2.

3.3 From Trial Parameters to Force-field Terms

In general, each molecular system contains several torsional and Lennard-Jones degrees of freedom, and (i) not all of them might be modelled by the optimized parameters; (ii) different degrees of freedom in optimization might be modelled by different *parameter types*. In fact, for each system and each parameter type there is a specific list of *Optimized Dihedrals* (ODs) or *Optimized Pairs* (OPs) that convey the information of the dihedral-angle quadruples or 1–4 nonbonded pairs relevant to the parameters in optimization.

The specification of the OPs and ODs for each parameter type is performed in the STP file of each system, as described in Section 4.1.1.

3.4 Reference Data and Weights

The reference data used in the optimization (usually from QM torsional scans) will be referred to by the notation r_{sk}, where s identifies the system and k identifies the step of the torsional scan. They are supplied in single-column files, one for each system considered.

The relative weights w_{sk} attributed to each reference data point r_{sk} in the optimization can be customized by the user. These weights affect the optimization in the manner shown in Section 3.5.2. Full control of these values can be achieved by supplying files containing the desired weights. If these files are not supplied, the values of the weights are derived from the settings in the INP file, as described in Section 4.1.2. There are two options: uniform weights ($w_{sk} = 1$ for all s, k) or Boltzmann weights

$$w_{sk} = \exp\left(-r_{sk}/RT\right) \quad ,$$

where R is the universal gas constant and T is a customizable temperature. Note that it is implicitly assumed that the minimum of the reference data lies at zero for each system, *i.e.* $\min_k r_{sk} = 0$ for all s. To guarantee this condition, the reference data for each molecule is always shifted by $\min_k r_{sk}$ before the evaluation of the fitness function.

3.5 Evolutionary Algorithms

In this section, we describe the main aspects of the implementation of the evolutionary algorithms used in the parameter optimization. In particular, we discuss the random initialization of parameters (Section 3.5.1), the calculation of the fitness function (Section 3.5.2) and the (hard-coded) genetic operators (Section 3.5.3).

3.5.1 Parameter Randomization

The obtention of random parameters depends on the particular evolutionary algorithm selected for the optimization.

Genetic Algorithm

When GA is selected as the evolutionary algorithm, whenever random parameters for the individuals are requested, they are obtained from three individually customizable random distributions: one for the torsional force constants, one for CS_6 and one for CS_{12}. Torsional phase-differences are an exception: they are always obtained from a uniform distribution in the interval $[-180°, 180°)$. The types of distributions, the ranges of acceptable values and their mean and standard deviation are configured in the INP file, as described in Section 4.1.2. Four types of distributions are available: Uniform, LogNormal, Normal and LogUniform. In the case of the torsional force constants, after a sample within the acceptable range of values is drawn from the distribution, its sign can be reversed with a fixed probability specified in the INP file, as described in Section 4.1.2.

CMA-ES

When CMA-ES is selected as the evolutionary algorithm, random parameters are chosen from a multivariate normal distribution that evolves during the execution of the algorithm. For this reason, the mean $\boldsymbol{\mu}_i$ and the covariance matrix $\sigma_i \boldsymbol{C}_i$ of this distribution are specific to each generation i.

The initial value of the mean, $\boldsymbol{\mu}_0$, is obtained from a random choice of parameter values in the same manner as random parameters are obtained for the GA (see previous section). The initial covariance matrix is a diagonal matrix where the leading diagonal is 0.2 times the initial mean, $\sigma_0 C_0 = \mathrm{diag}(0.2\boldsymbol{\mu}_0)$.

The evolution of the distribution is dictated by the `deap.cma.Strategy` strategy. Internally, the evolution algorithm works in a normalized version of the parameter space. The conversion between the real and normalized parameter spaces is mediated by a `sklearn.preprocessing.MaxAbsScaler` scaler fit on the initial mean $\boldsymbol{\mu}_0$. In the normalized space, the initial parameters of the distribution, indicated by the prime symbols, are then

$$\begin{aligned} \boldsymbol{\mu}'_0 &= \mathbf{1} \\ \sigma'_0 &= 0.2 \\ \boldsymbol{C}'_i &= \mathrm{diag}(\mathbf{1}) \end{aligned}$$

3.5.2 Fitness

The *fitness* is calculated by considering the differences between the torsional-scan energy profiles of the individual and the reference-data profiles, encoding them into a real number $\mathcal{R}$ *via* a suitable metric and finally converting this distance-like value into a fitness $\mathcal{F}$.

First, each individual $\mathbf{X}_i$ of the population is evaluated in terms of the weighted root-mean-square deviation

$$\mathcal{R}\left(\mathbf{X}_i\right) = \sqrt{\frac{\sum_{s=1}^{N_S}\sum_{k=1}^{N_K} w_{sk}(\mathcal{V}_{sk}(\mathbf{X}_i) - r_{sk})^2}{N_S \sum_{k=1}^{N_R} w_{sk}}} \quad , \tag{3.5.1}$$

where the w_{sk}, r_{sk} are the weights and reference data described in Section 3.4 and each $\mathcal{V}_{sk}(\mathbf{X}_i)$ is the k-th step of the energy profile $\mathcal{V}_s(\mathbf{X}_i)$ described in Section 3.1. This distance-like quantity is converted into a fitness value *via*

$$\mathcal{F}(\mathbf{X}_i) = \frac{\left[\mathcal{R}(\mathbf{X}_i)\right]^{-1}}{\sum_{j=1}^{N_P}\left[\mathcal{R}(\mathbf{X}_j)\right]^{-1}} \quad . \tag{3.5.2}$$

The $\mathcal{R}$ and $\mathcal{F}$ quantities differ in terms of normalization, ordering and interpretability. The latter is normalized over the population and should be maximized in the optimization, but has no clear physical interpretation. The former is (at least *a priori*) not normalized and should be minimized, but has a very clear physical interpretation. For these reasons, $\mathcal{F}$ is used internally in the GA as the metric of fitness, while $\mathcal{R}$ is reported in the output files.

3.5.3 Genetic Algorithm Operators

The GA evolves according to the `deap.algorithms.eaSimple` strategy.

The evolutionary operators are:

- One-point Crossover (`deap.tools.cxOnePoint`)
- Roulette Selection (`deap.tools.selRoulette`)
- Gaussian Mutation (`deap.tools.mutGaussian`)
  - $\boldsymbol{\mu} = \mathbf{0}$
  - $\boldsymbol{\sigma} = 0.1\mathbf{P}$, where $\mathbf{P}$ is the vector of parameters of the individual
  - indpb $= 0.26$

The probability of applying the operators are:

- Crossover: 0.23
- Mutation: 0.39

3.6 Self-consistent Linear Least-Squares Optimization

3.6.1 Basic Aspects

The self-consistent linear least-squares optimization (LLS-SC) is based on two main ideas:

1. For fixed torsional-scan trajectories, the parametrization of torsional and third-neighbor Lennard-Jones terms can be recast as a linear least-squares regression problem, which is solved using the linear regression model in the `scikit-learn` library.
2. In practice, the torsional-scan trajectories may depend implicitly on the parameters being calibrated. For this reason, a self-consistent scheme is used, where each set of newly estimated parameters is used to reobtain the torsional-scan trajectories. These trajectories are then used to obtain new parameters, and so on, until convergence.

The LLS-SC method is explained in details in the `profilerTools` paper.[3] The main difference is that here the square of the weighted residual function in Equation 3.5.1, $\mathcal{R}^2$, is used as the actual residual function.

3.6.2 Regularized Optimization

This section complements the description of the LLS-SC method and relies on notation defined both in the `profilerTools` paper[3] and in this document.

The solution of the LLS problem by means of standard matrix techniques does not impose any conditions on the values of parameter estimates. Consequently, it may yield unphysical results, where the third-neighbor parameters have negative values. To avoid this outcome, a type of regularization term can be included in the residual function, designed to prevent the minimum of the residual function to stray too far away from some reference parameter vector $\mathbf{X}^0 \equiv (X^0_{\Omega_1,\alpha_1}, \ldots, X^0_{\Omega_K,\alpha_K})$.

Using $\mathcal{R}$ from Equation 3.5.1, the regularized residual function is

$$\chi^2_{\mathrm{R}}(\mathbf{X}) = \mathcal{R}^2(\mathbf{X}) + \lambda^2 \sum_{k=1}^{K} \left[l_k^{-1} (X_{\Omega_k,\alpha_k} - X^0_{\Omega_k,\alpha_k}) \right]^2 , \tag{3.6.1}$$

where l_k, with $t = 1, \ldots, K$, controls the relative weights attributed to the deviations of each parameter with respect to its reference value, and the regularization parameter λ quantifies the balance between minimizing the original residual function or the regularization term. A reasonable strategy to choose the value of l_t for one parameter is to set it to the expected distance between its value in the solution and in the reference parameter vector, in orders of magnitude. For example, one may set the l_t factors to 1 kJ mol^{-1} for torsional parameters, 10^3 kJ mol^{-1} nm^6 for attractive Lennard-Jones parameters and 10^6 kJ mol^{-1} nm^{12} for repulsive Lennard-Jones parameters, based on typical values for these parameters. Regarding λ, there are established methods for estimating a balanced value, such as picking the corner of the L-curve.

We note that the regularization strategy proposed above is slightly different from the usual kind of regularization adopted in LLS problems. Commonly, the residual function is used to control the *norm* $\|\mathbf{X}\|_2^2$ of the solution, and not the *distance* $\left\|\mathbf{X} - \mathbf{X}^0\right\|_2^2$ between the solution and a reference parameter vector $\mathbf{X}^0$. This distance-based regularization, however, is easily achieved by first carrying out a change of variables where the origin is shifted to $\mathbf{X}^0$, and then solving the usual regularized regression. With this caveat in mind, the optimization of the regularized residual function χ^2_{R} can be carried out by relying essentially on the same methods and tools as the optimization of the original residual function $\mathcal{R}^2$. In particular, the regularized minimization is still a LLS problem, amenable to be solved by standard matrix techniques, such as SVD, implemented in commonly used scientific libraries (*e.g.*, `GSL` for C/C++, and `numpy` for Python).

In `profilerTools`, regularization is carried out based on the `GSL` library. This is achieved by implementing a C extension to call the appropriate `GSL` functions to solve the regression problem within Python.

Chapter 4

Input and Output File Formats

The usage of `profilerOpt` and `profilerGen` depends on several input variables and input data supplied either as command-line options or as input files with specified formats. The output data is also written to files with specified formats. In this chapter, we describe describe the configuration of these input variables and the format of these input and output files.

4.1 File Formats

4.1.1 Special Topology File (STP)

The special topology (STP) file combines the topology with the specification of the RDs (Section 3.1), the ODs and the OPs (Section 3.3). Comments can be inserted after a ";" character. This file is organized into blocks, each starting with a line consisting of `[ blockname ]` (where `blockname` is the name of the block) and ending with an empty line.

The structure of the STP file is based on three parts, composed by the following blocks:

1. Topology and interaction parameters:
   `[ defaults ]`; `[ atoms ]`; `[ nbpairs ]`; `[ bonds ]`; `[ angles ]`; `[ dihedrals ]`.
2. Reference dihedrals (RDs):
   `[ refdihedral ]`.
3. Optimization targets (ODs and OPs):
   `[ optdihedrals ]`; `[ optpairs ]` or `[ optatoms ]`.

Topology and Interaction Parameters

Users who are familiar with the GROMACS package* will notice that the format of the STP file looks very similar to that of the GROMACS ITP file, specially in the nomenclature of the blocks of the topology part. However, the correspondence between these two file formats is not perfect. In the following, we discuss in details each of the topology blocks.

`[ defaults ]` The format of this block is similar to its correspondent in the ITP file format†. The following variables are specified in order:

— `nbfunc`: **(Integer)** the non-bonded function type. Only the value 1 (Lennard-Jones) is supported;

— `comb-rule`: **(Integer)** the number of the combination/mixing rule and Lennard-Jones potential representation (see Table 4.1). Note that all molecular-mechanics calculations are ultimately performed in the C_6-C_{12} representation, regardless of the representation expected in the STP file.

— `gen-pairs`: **(String)** the strategy of automatic obtention of 1–4 Lennard-Jones pair parameters based on the mixing rule. The allowed values are:

  — `no`: do not generate pair parameters. This requires explicitly setting the interaction parameters for all 1–4 nonbonded pairs in the `[ nbpairs ]` block;

  — `fudge`: compute the pair parameters by applying the mixing rule on the standard atomic parameters (C_6, C_{12}) and then scaling the results by `fudgeLJ` (f_{LJ});

*See `http://www.gromacs.org`

†Strictly speaking, this block is in most cases imported into the ITP file from the force-field file, so it is not directly a part of the ITP file.

Table 4.1: Representation of the Lennard-Jones parameters expected in the STP file along with the corresponding mixing rule for each allowed value of `comb-rule` in the `[ defaults ]` block. See Section 2.1.5 for more details.

`comb-rule`	Representation	Mixing Rule
1	C_6-C_{12}	Geometric
2	σ-ϵ	Lorentz-Berthelot
3	σ-ϵ	Geometric

  - `atomic`: compute the pair parameters by applying the mixing rule on the 1–4 atomic parameters (CS_6, CS_{12}). This requires setting CS_6 and CS_{12} for all atoms in the `[ atoms ]` block.

— `fudgeLJ`: **(Float)** scaling factor f_{LJ} used to obtain 1–4 Lennard-Jones pair parameters from the standard atomic parameters when `gen-pairs` is `fudge`.

— `fudgeQQ`: **(Float)** scaling factor f_{QQ} used to obtain 1–4 electrostatic pair parameters from the atomic partial charges.

For more details about the calculation of the pair parameters using the mixing rule and about the scaling factors f_{LJ} and f_{QQ}, see Section 2.1.5.

`[ atoms ]` The format of this block depends on the `gen-pairs` strategy and on `comb-rule`. If `gen-pairs` is `atomic`, each line must contain, in sequence and separated by spaces:

— Atom type **(String)**

— Lennard-Jones parameter C_6 or σ **(Float)**; see Table 4.1.

— Lennard-Jones parameter C_{12} or ϵ **(Float)**; see Table 4.1.

— 1–4 Lennard-Jones parameter CS_6 or σ_S **(Float)**; see Table 4.1.

— 1–4 Lennard-Jones parameter CS_{12} or ϵ_S **(Float)**; see Table 4.1.

— Atomic partial charge q **(Float)**

For other values of `gen-pairs`, CS_6 and CS_{12} are not used and may be omitted. If they are supplied, their values are skipped when parsing the file.

`[ nbpairs ]` In this block, each line describes a nonbonded pair a_i—a_j and must contain, in sequence and separated by spaces, the indexes a_i **(Integer)** and a_j **(Integer)**, the corresponding interaction type code **(Integer)** and, in some cases, the values of the Lennard-Jones interaction parameters listed in a specific order (see Table 4.2). The interaction parameters are only required when they cannot be inferred from the atomic parameters, *i.e.* for 1–4 nonbonded pairs when `gen-pairs` is `no`. When they are not required but nonetheless supplied, they *override* the default values derived using the mixing rule. In divergence with the ITP file format, note that *all* nonbonded pairs must be listed in this block, regardless if the atoms involved are third-neighbors or not.

`[ bonds ]` In this block, each line describes a bond a_i—a_j and must contain, in sequence and separated by spaces, the indexes a_i **(Integer)** and a_j **(Integer)**, the corresponding interaction type code **(Integer)** and the values of the interaction parameters listed in a specific order (see Table 4.2). Note that the format of this block is the same as its correspondent in the ITP file format, with the caveat that the interaction parameters must always be listed explicitly.

`[ angles ]` In this block, each line describes an angle a_i—a_j—a_k and must contain, in sequence and separated by spaces, the indexes a_i **(Integer)**, a_j **(Integer)** and a_k **(Integer)**, the corresponding interaction type code **(Integer)** and the values of the interaction parameters listed in a specific order (see Table 4.2). Note that the format of this block is the same as its correspondent in the ITP file format, with the caveat that the interaction parameters must always be listed explicitly.

Table 4.2: Interaction types, interaction type codes and list of corresponding parameters for use in the topology part of the STP file.

Interaction type	Block	Code	List of parameters	Cross-reference
Nonbonded				
Standard pair	`nbpairs`	1	C_6 (kJ mol^{-1}nm^{-6}); C_{12} (kJ mol^{-1}nm^{-12})$^{(a)}$ σ (nm); ϵ (kJ mol^{-1})$^{(b)}$	Equation 2.1.12
1–4 pair	`nbpairs`	2	CS_6 (kJ mol^{-1}nm^{-6}); CS_{12} (kJ mol^{-1}nm^{-12})$^{(a)}$ σ_S (nm); ϵ_S (kJ mol^{-1})$^{(b)}$	Equation 2.1.19
Bonds				
Harmonic	`bonds`	1	b_0 (nm); k_b^h (kJ mol^{-1}nm^{-2})	Equation 2.1.1
Quartic	`bonds`	2	b_0 (nm); k_b^q (kJ mol^{-1}nm^{-4})	Equation 2.1.2
Angles				
Harmonic	`angles`	1	θ_0 (deg); k_a^h (kJ mol^{-1}rad^{-2})	Equation 2.1.3
Cosine-harmonic	`angles`	2	θ_0 (deg); k_a^c (kJ mol^{-1})	Equation 2.1.4
Urey-Bradley	`angles`	5	θ_0 (deg); k_a^{ub} (kJ mol^{-1}rad^{-2}); b_{13}^{ub} (nm); k_{13}^{ub} (kJ mol^{-1}nm^{-2})	Equation 2.1.5
Proper dihedrals				
Standard periodic$^{(c)}$	`dihedrals`	1	$\phi_{0,m}$ (deg); $k_{d,m}^s$ (kJ mol^{-1}); m	Equation 2.1.6
Fourier	`dihedrals`	5	f_1; f_2; f_3; f_4 (kJ mol^{-1})	Equation 2.1.7
Standard periodic$^{(d)}$	`dihedrals`	9	$\phi_{0,m}$ (deg); $k_{d,m}^s$ (kJ mol^{-1}); m	Equation 2.1.6
Harmonic restraint	`dihedrals`	−1	ϕ_0 (deg); k_d^r (kJ mol^{-1}rad^{-2})	Equation 2.1.8
Improper dihedrals				
Harmonic	`dihedrals`	2	ϕ_0 (deg); k_i^h (kJ mol^{-1}rad^{-2})	Equation 2.1.9
Periodic	`dihedrals`	4	ϕ_0 (deg); k_i^p (kJ mol^{-1}); m	Equation 2.1.10

$^{(a)}$: For a C_6-C_{12} representation; see Table 4.1.
$^{(b)}$: For a σ-ϵ representation; see Table 4.1.
$^{(c)}$: Using only one summand of Equation 2.1.6.
$^{(d)}$: Using one or more summands of Equation 2.1.6.

`[ dihedrals ]` In this block, each line describes a dihedral angle a_i—a_j—a_k—a_l and must contain, in sequence and separated by spaces, the indexes a_i **(Integer)**, a_j **(Integer)**, a_k **(Integer)** and a_l **(Integer)**, the corresponding interaction type code **(Integer)** and the values of the interaction parameters listed in a specific order (see Table 4.2). Note that the format of this block is the same as its correspondent in the ITP file format, except that: (*i*) the interaction parameters must always be listed explicitly and (*ii*) this block also accommodates dihedral restraints. However, bear in mind that the dihedral restraints specified in this manner have no connection with the dihedral restraints used to perform the torsional scans in `profilerGen` and `profilerOpt` (except for the common functional form).

Reference Dihedrals

The *i*-th `[ refdihedrals ]` block appearing in the STP file contains the specification of the dihedral angles to which the settings of the *i*-th torsional-scan type are applied. For example, a 3D systematic scan $\mathcal{S}_1 \times \mathcal{S}_2 \times \mathcal{S}_3$ [‡] may sample the dihedral-angle values $\mathcal{S}_1 = \{0°, 10°, \ldots, 360°\}$ along one dimension and $\mathcal{S}_2 = \mathcal{S}_3 = \{0°, 30°, \ldots, 360°\}$ along the remaining dimensions. In this case, 2 torsional-scan settings are required for 3 RDs; one setting for the first RD and another setting for the 2 remaining RDs. In the STP file, this is indicated by defining the first RD in one `[ refdihedrals ]` block, and the other two RDs in another `[ refdihedrals ]` block. For `profilerOpt`, the corresponding torsional-scan settings are defined in the `[ torsional_scan ]` block of the INP file (see Section 4.1.2). For `profilerGen`, they are defined in the command-line options.

Each `[ refdihedrals ]` block is formed by zero or more lines containing the indexes of the four atoms of the desired dihedral angle separated by spaces. For consistency, it is required that each reference dihedral match one of the dihedral angles listed in a `[ dihedrals ]` block. For `profilerOpt` jobs involving many molecules, empty `[ refdihedrals ]` blocks can be used when there are torsional-scan settings that do not apply to all molecules (see Chapter 5 for an example).

Optimization Targets (exclusive to `profilerOpt`)

Dihedrals The dihedral-angle terms affected by the optimization are specified in `[ optdihedrals ]` blocks. The *i*-th block appearing in the STP file contains the specification of the dihedral angles to which the settings

[‡] $A \times B$ is the cartesian product of the sets A and B.

of the i-th torsional type defined in the ITP file (see Section 4.1.2) apply. In these blocks, each line describes a dihedral angle and contains the indexes of its four atoms separated by spaces. For consistency, every dihedral angle listed in these blocks must match one of the dihedral angles listed in a `[ dihedrals ]` block. For `profilerOpt` jobs involving many molecules, empty blocks can be used when there are torsional types under optimzation that do not apply to all molecules.

1–4 Pairs The third-neighbor nonbonded pairs affected by the optimization can be specified in two mutually exclusive ways: either *via* `[ optpairs ]` blocks or *via* `[ optatoms ]` blocks. In both cases, the i-th block appearing in the STP file contains the specification of the pairs/atoms to which the settings of the i-th Lennard-Jones type defined in the INP file (see Section 4.1.2) apply.

When optimization of specific nonbonded pairs is intended, the most effective strategy is to use the `[ optpairs ]` blocks, wherein each line describes a 1–4 nonbonded pair and contains the indexes of its two atoms separated by spaces. When the molecular-mechanics calculations are performed, the estimated parameters override the pair parameters of all pairs listed in these blocks, respecting the correspondence between the blocks and the Lennard-Jones types in the INP file. For consistency, it is required that every listed pair must match one of the 1–4 pairs in the `[ nbpairs ]` block.

When optimization of the 1–4 Lennard-Jones parameters of *atomtypes* is intended instead (which only makes sense if `gen-pairs` is `atomic`, which is not the case for most force fields), `[ optatoms ]` blocks can be used. In this case, the indexes of the atoms of the desired atomtype are listed, separated by spaces or line breaks. When the molecular-mechanics calculations are performed, the estimated parameters override the atomic parameters of all atoms listed in these blocks, respecting the correspondence between the blocks and the Lennard-Jones types in the INP file. This in turn affects (*via* mixing rule) *all* 1–4 nonbonded pair interactions involving at least one of these atoms.

4.1.2 Input Parameters File (INP)

The input parameters (INP) file contains the definition of the majority of the input variables of `profilerOpt`. Like the STP file, the INP file is organized into blocks, wherein input variables of similar nature are defined according to their position. Successive positions in a block are separated by spaces or by a line break, with no syntactic distinction between the two. However, the use of line breaks is recommended to separate clusters of closely related variables within each block. In the following, we go through each block in succession, giving the position, description and allowed values of the input variables.

`[ parameter_optimization ]`

Positions of the variables:

NTORS	NLJ	FTORS	
TORSNT[1][1]		...	TORSNT[1][6]
		⋮	
TORSNT[NTORS][1]		...	TORSNT[NTORS][6]
LJNT[1][1]	LJNT[1][2]		
		⋮	
LJNT[NLJ][1]	LJNT[NLJ][2]		
WTEMP			

Details of the variables:

NTORS : **(Integer)** Number of torsional types.
Allowed values:
≥ 0 : Number of torsional types.

NLJ : **(Integer)** Number of Lennard-Jones types.
Allowed values:
≥ 0 : Number of Lennard-Jones types.

FTORS : **(Integer)** Controls functional form of optimized torsional terms.
Allowed values:
1 : Standard periodic dihedral
2 : Fourier dihedral

TORSNT[I][J] : **(Matrix of Integers)** Fine control of the functional form of torsional terms.
Allowed values:
0 : set (I,J)-th term to zero
1 : optimize force constant of (I,J)-th term; the corresponding phase shifts are either set to zero (for standard periodic dihedral; FTORS = 1) or to (I mod 2)π (for Fourier dihedrals; FTORS = 2)
2 : optimize force constant and phase shift of (I,J)-th term
Matrix index targets:
I : I-th torsional type
J : J-th summand of Equation 2.1.6 or Equation 2.1.7

LJNT[I][J] : **(Matrix of Integers)** Controls which Lennard-Jones components are optimized.
Allowed values:
0 : use (I,J)-th value derived from STP file
1 : optimize (I,J)-th term
Array index targets:
I : I-th Lennard-Jones type
$J = 1$: CS_6
$J = 2$: CS_{12}

WTEMP : **(Float)** Controls weights given to reference data points when weight files are not supplied.
Allowed values:
0 : uniform weights
> 0 : Boltzmann weights with temperature WTEMP (see Section 3.4)

An entry of TORSNT that is 0 sets the corresponding force constant to 0.0 kJ/mol. On the other hand, an entry of LJNT that is 0 sets the corresponding Lennard-Jones coefficient to the value specified in the topology.

[parameter_randomization]

Positions of the variables:

PINV				
DIST[1]	MIN[1]	MAX[1]	MEAN[1]	STDDEV[1]
DIST[2]	MIN[2]	MAX[2]	MEAN[2]	STDDEV[2]
DIST[3]	MIN[3]	MAX[3]	MEAN[3]	STDDEV[3]

Details of the variables:

PINV : **(Integer)** Probability of inverting sign of random torsional parameters.
Allowed values:
0 . . . 100 : set probability to PINV/100

DIST[I] : **(Array of Integers)** Controls types of probability distributions of random parameters.
Allowed values:
1 : uniform distribution
2 : LogNormal distribution
3 : normal distribution
4 : LogUniform distribution
Array index targets:
1 : torsional terms
2 : CS_6
3 : CS_{12}

MIN[I] : **(Array of Float)** Minimum values of random parameters.
Allowed values:
$\in \mathbb{R}$: only accept random values larger than MIN[I]
Array index targets:
1 : torsional terms
2 : CS_6
3 : CS_{12}

MAX[I] : **(Array of Float)** Maximum values of random parameters.
Allowed values:
$\in \mathbb{R}$: only accept random values smaller than MAX[I]
Array index targets:
1 : torsional terms
2 : CS_6
3 : CS_{12}

MEAN[I] : **(Array of Float)** Mean values of the probability distributions.
Allowed values:
$\in \mathbb{R}$: set mean of DIST[I] to MEAN[I]; ignored for uniform and LogUniform distributions
Array index targets:
1 : torsional terms
2 : CS_6
3 : CS_{12}

STDDEV[I] : **(Array of Float)** Std. dev. of the probability distributions.
Allowed values:
$\in \mathbb{R}$: set std. dev. of DIST[I] to STDDEV[I]; ignored for uniform and LogUniform distributions
Array index targets:
1 : torsional terms
2 : CS_6
3 : CS_{12}

Note that the uniform and LogUniform distributions are well defined by the values of MIN and MAX. For this reason, the MEAN and STDDEV values are ignored for these types of distributions.

`[ seed ]`

Positions of the variables:

SEED

Details of the variables:

SEED : **(Integer)** Random number generator seed.
Allowed values:
-1 : set random number generator seed to Python default
> 0 : set random number generator seed to SEED

`[ evolutionary_strat ]`

Positions of the variables:

STRAT
POPSIZE NGENS

Details of the variables:

STRAT : **(Integer)** Evolutionary strategy.
Allowed values:
1 : set strategy to CMA-ES
2 : set strategy to GA

POPSIZE : **(Integer)** Size of the population.
Allowed values:
> 0 : set size of the population to POPSIZE

NGENS : **(Integer)** Number of generations.
Allowed values:
> 0 : set number of generations to NGENS

`[ lls_sc ]`

Positions of the variables:

ITER DPAR
SREG
CREG[1]
⋮
CREG[N_P]
LREG

Details of the variables:

ITER : **(Integer)** Maximum number of self-consistent iterations.
Allowed values:
> 0 : set maximum number of iterations to ITER

DPAR : **(Float)** Tolerance for the relative change in the values of the parameters.
Allowed values:
> 0 : set tolerance to DPAR

SREG : **(Integer)** Controls the use of regularization in the linear regression.
Allowed values:
0 : do not use regularization
1 : use regularization

CREG[I] : **(Array of Float)** Reference values for the regularization distance.
Allowed values:
$\in \mathbb{R}$: set reference value of the I-th parameter to CREG[I]
Array index targets:
I : I-th parameter

The enumeration of the parameters respects the following order. First, the reference values corresponding to each Lennard-Jones type are listed. These types should follow the same order they are defined in the `[ parameter_optimization ]` block, with the reference value for CS_6 appearing just before the reference value for the corresponding CS_{12}. After that, the reference values for the force constants of each torsional type are listed, also following the order the types are defined in the `[ parameter_optimization ]` block. After *all* force constants, the reference values for the phase-shift parameters are listed. This only takes into account the parameters that are actually optimized; those that are not should be omitted from the enumeration.

LREG : **(Float)** Controls the value of λ in regularization.
Allowed values:
≤ 0 : set λ automatically *via* corner of the L-curve
> 0 : set λ to LREG

`[ torsional_scan ]`

Positions of the variables:

NTSCAN
NRESTR
RFRST[1] RSTEP[1] RLST[1] RFCT[1]
⋮
RFRST[NRESTR] RSTEP[NRESTR] RLST[NRESTR] RFCT[NRESTR]

Details of the variables:

NTSCAN : **(Integer)** Controls the origin of the reference-dihedral values.
Allowed values:
1 : perform systematic torsional scan
2 : read reference values from input configurations
3 : read reference values from SPEC file

NRESTR : **(Integer)** Number of torsional-scan types.
Allowed values:
> 0 : set number of torsional-scan types to NRESTR.

RFRST[I] : **(Array of Float)** First angle of I-th torsional-scan type.
Allowed values:
$\in \mathbb{R}$: set first angle to RFRST[I] degrees
Array index targets:
1 . . . NRESTR : index of torsional-scan type

RSTEP[I] : **(Array of Float)** Step angle of I-th torsional-scan type.
Allowed values:
$\in \mathbb{R}$: set step angle to RSTEP[I] degrees
Array index targets:
1 . . . NRESTR : index of torsional-scan type

RLST[I] : **(Array of Float)** Last angle of I-th torsional-scan type (inclusive).
Allowed values:
$\in \mathbb{R}$: set last angle to RLST[I] degrees
Array index targets:
1 . . . NRESTR : index of torsional-scan type

RFCT[I] : **(Array of Float)** Force constant of I-th torsional-scan type.
Allowed values:
> 0 : set force constant of the harmonic dihedral restraint to RFCT[I] (kJ $\mathrm{mol}^{-1}\mathrm{rad}^{-2}$)
Array index targets:
1 . . . NRESTR : index of torsional-scan type

Note that, for NTSCAN $\neq 1$, only the values of NRESTR and of the force constants RFCT[I] matter; the other values are read but not used to generate the reference-dihedral reference values.

`[ minimization ]`

Positions of the variables:

MALG[1]	DX0[1]	DXM[1]	NMAX[1]	DELE[1]
MALG[2]	DX0[2]	DXM[2]	NMAX[2]	DELE[2]

Details of the variables:

MALG[I] : **(Array of Integers)** Controls types of the minimization algorithms.
Allowed values:
0 : do not perform energy minimization
1 : steepest-descents (more robust, less prone to errors)
Array index targets:
1 : minimization settings for all individuals/all generations/all LLS-SC iterations
2 : enhanced minimization settings for optimal parameters at the end of the run

DX0[I] : **(Array of Floats)** Initial step sizes DX0 in the minimization algorithms.
Allowed values:
> 0 : set DX0 of MALG[I] to DX0[I]
Array index targets:
1 : minimization settings for all individuals/all generations/all LLS-SC iterations
2 : enhanced minimization settings for optimal individual at the end of the run

DXM[I] : **(Array of Floats)** Maximum step sizes DXM in the minimization algorithms.
Allowed values:
> 0 : set DXM of MALG[I] to DX0[I]
Array index targets:
1 : minimization settings for all individuals/all generations/all LLS-SC iterations
2 : enhanced minimization settings for optimal individual at the end of the run

NMAX[I] : **(Array of Integers)** Maximum number of steps in the minimization algorithms.
Allowed values:
> 0 : set the maximum number of steps of MALG[I] to NMAX[I]
Array index targets:
1 : minimization settings for all individuals/all generations/all LLS-SC iterations
2 : enhanced minimization settings for optimal individual at the end of the run

DELE[I] : **(Array of Floats)** Energy-convergence criteria of the minimization algorithms.
Allowed values:
> 0 : set DELE of MALG[I] to DELE[I]
Array index targets:
1 : minimization settings for all individuals/all generations/all LLS-SC iterations
2 : enhanced minimization settings for optimal individual at the end of the run

For more details about the algorithmic meaning of these variables, see Section 2.3.

4.1.3 Dihedral-angle Specification File (SPEC)

The dihedral-angle specification file is an optional file for fine control of the conformations of the reference dihedrals. In this file, each line corresponds to a conformation of the torsional-scan, while each column (separated by spaces) lists the values of the reference dihedrals, in order of appearance in the corresponding STP file.

4.1.4 Interaction Function Parameters File (IFP)

The interaction function parameters file (IFP) contains the final values (*i.e.* those of the best individual of the population) of the optimized parameters for each type, along with the corresponding value of the weighted RMSD. In this file, the types are identified by an integer that reflects the order of appearance in the INP file, except that all the Lennard-Jones types are written before the torsional types.

Chapter 5

Tutorial

In this chapter, the use of `profilerTools` is illustrated in a didactic way in the form of a tutorial. The tutorial has two parts:

1. In Section 5.2, `profilerGen` is used to obtain a 2D torsional scan of a united-atoms pentane molecule. The torsional-scan angles are specified in two alternative ways:

   a) explicitly, by running the program with an input file containing the list of the values of the two reference dihedrals for each configuration of the torsional scan:

   ```
   -180        -180
   -180        -160
   -180        -140
         ...
   180         140
   180         160
   180         180
   ```

   b) indirectly, by defining the pair of axes of a bidimensional grid of angles of which the grid points (the elements of the cartesian product of the two axes) are the torsional-scan angles:

   $$\underbrace{(-180^\circ, -160^\circ, \ldots, 180^\circ)}_{\text{Axis 1}} \times \underbrace{(-180^\circ, -160^\circ, \ldots, 180^\circ)}_{\text{Axis 2}}$$

   Note that the coordinates that result from the cartesian product above are the values specified in the file mentioned in the previous item.

2. In Section 5.3.2, `profilerOpt` is used to optimize the torsional (force constant and phase-difference) and 1–4 Lennard-Jones parameters modelling the aliphatic chains of a united-atoms butane and a united-atoms pentane molecule. This is done by adjusting the atomic CS_6 and CS_{12} parameters of the CH2 and CH3 atom types using CMA-ES and LLS-SC.

We strongly suggest reading the following sections before starting the tutorial: Section 3.1, where we explain how the concept of a multidimensional torsional-scan is interpreted in `profilerTools`; and Section 4.1.1, where we describe the syntax of the special-topology (STP) file, central in the usage of `profilerTools`.

Throughout this chapter, we assume the reader has access to a Linux terminal emulator and is familiarized with some type of shell language. In particular, the command-line snippets in this chapter are written in `bash`. If this is not your case, please adapt the commands to match your terminal configuration.

5.1 Obtaining an STP File

The biggest barrier to use `profilerTools` is creating an STP file for the systems of interest. With that difficulty in mind, `profilerTools` is shipped together with two utility programs that may be very useful to the user:

- `gromacs_itp_to_stp`, that can be used to convert GROMACS ITP files into STP files.
- `gromos_top_to_stp`, that can be used to convert GROMOS TOP files into STP files.

These programs are in the `utils` directory of the repository, and their usage can be consulted in the corresponding `README` files.

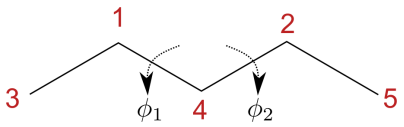

Figure 5.1: United-atoms pentane molecule.

5.2 Generating Torsional-Scan Trajectories

The files for this part of the tutorial are in `./doc/tutorial/profilerGen`. They are split into two subdirectories: `spec` and `systematic`. The `spec` subdirectory contains the files for a `profilerGen` run in which the torsional-scan angles are specified explicitly (this is done via the SPEC file, hence the name). The `systematic` subdirectory contains the files for a run where *the same torsional-scan angles* are specified using the command-line option for a systematic scan, `-dr`.

The system considered in this example is a united-atoms pentane molecule, with 2 reference dihedrals, ϕ_1, ϕ_2 (see Figure 5.1). The torsional-scan angles considered are $(-180°, -160°, \ldots, 180°) \times (-180°, -160°, \ldots, 180°)$, forming a $19^2 \times 2$ matrix of angles,

$$\Phi = \begin{bmatrix} -180° & -180° \\ -180° & -160° \\ & \vdots \\ -180° & 180° \\ -160° & -180° \\ & \vdots \\ 180° & 160° \\ 180° & 180° \end{bmatrix} . \tag{5.2.1}$$

The STP file for pentane (`pe.stp`) is the same for both examples, so there is no need to distinguish between them. The first few blocks contain the topology and the force-field parameters. They are pretty similar in concept and syntax to the contents of the ITP topology file of GROMACS, and users who have already done any type of force-field based molecular-mechanics calculation will probably very easily understand these blocks. Following the topology part, there is a single reference-dihedral group, where both dihedral angles (ϕ_1 and ϕ_2) are listed:

```
[ refdihedrals ]
;  ai   aj   ak   al
    3    1    4    2
    5    2    4    1
```

It is important to understand that, although there is a single reference-dihedral group, the torsional scan is *bidimensional*, and the two dihedral angles vary *independently*. They are listed in the same group because (by choice) they share the same dihedral-restraint attributes, namely the harmonic force constants and, for the case of the systematic scan, also the same angle values along each dimension of the scan. If different harmonic force constants were used for each dihedral angle, or, for the systematic scan, if different base angles were used along each dimension (for instance, $(-180°, -160°, \ldots, 180°) \times (0°, 90°, \ldots, 360°)$), each dihedral angle would have to be listed in its own `[ refdihedrals ]` block.

The input trajectory (`pe.xyz`) is also the same for both examples, and it contains a single configuration of pentane in vacuum:

```
5
dihedral scan frame
 C            36.2175140        2.8697201       13.5523708
 C            38.7301917        2.1817050       13.2514028
 C            36.7128127        3.7399489       14.7152958
 C            37.2739523        2.0533908       12.7853724
 C            39.6555292        1.3152352       12.3955582
```

5.2.1 Based on a SPEC file

The dihedral-angle specification file (`dih.spec`) for the torsional-scan angles described above is a file containing the matrix in Equation 5.2.1, with each row in a line and with the columns separated by blank characters.

To run the `profilerGen` job for pentane based on this SPEC file, enter the appropriate directory

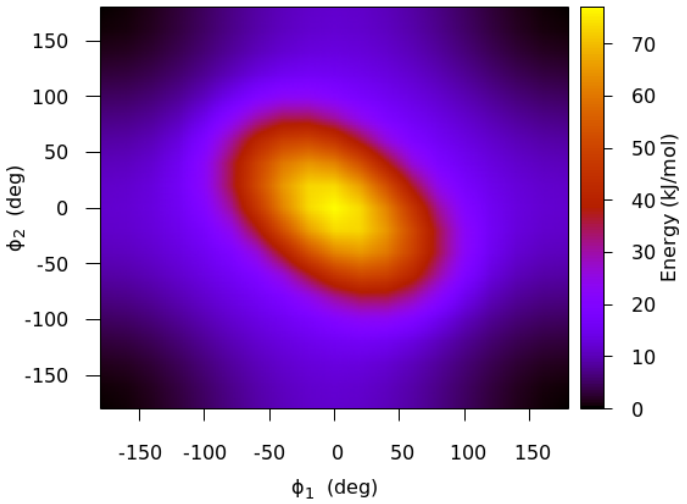

Figure 5.2: Torsional PES for pentane generated with `profilerGen` (see Figure 5.1 for the definition of ϕ_1 and ϕ_2).

```
$ cd doc/tutorial/profilerGen/spec
```

and run `profilerGen` using the appropriate STP, XYZ and SPEC files as input:

```
$ profilerGen -c pe.xyz -t pe.stp -s dih.spec -op pe_spec -dk 5000
```

Notice that we used a single restraint force constant of 5000 kJ mol^{-1} rad^{-2} for both dihedrals, which is compatible with listing both of them under a single `[ refdihedrals ]` block. Also, the string `pe_spec` is set as the prefix for all output files.

The output files of this run are the XYZ trajectory file `pe_spec.xyz` and the DAT file `pe_spec.dat` containing the energy profile of the torsional scan. The energies (and configurations of the trajectory) are listed in a "flattened" fashion, as a single column with 19^2 rows, one for each torsional-scan configuration. Also, for plotting convenience, the energy values are shifted so that their minima lies at zero (in the program, the calculations are performed with the *average* value lying at zero). When plotted as a 19×19 matrix using the corresponding values of ϕ_1 and ϕ_2 as axes labels, the result is a nice-looking torsional PES, as shown in Figure 5.2.

5.2.2 Based on a Systematic Scan

To run the `profilerGen` job for pentane based on a systematic scan that is equivalent to the SPEC file in the previous section, enter the appropriate directory

```
$ cd doc/tutorial/profilerGen/systematic
```

and run `profilerGen` using the appropriate STP and XYZ files and the appropriate command-line arguments as input (run `profilerGen -h` for an overview):

```
$ profilerGen -c pe.xyz -t pe.stp -op pe_systematic -dr -180 20 180 -dk 5000
```

Notice that we used a single set of restraint settings for both dihedrals: a force constant of 5000 kJ mol^{-1} rad^{-2} (option `-dk`) and torsional-scan angles (*along each dimension*) from $-180°$ to $180°$ with a step of $20°$ (option `-dr`). This is compatible with listing ϕ_1 and ϕ_2 under a single `[ refdihedrals ]` block.

The output files of this run are the XYZ trajectory file `pe_systematic.xyz` and the DAT file `pe_systematic.dat` containing the energy profile of the torsional scan. The energies are listed in a "flattened" fashion, as a single column with 19^2 rows, one for each torsional-scan configuration. Also, for plotting convenience, the energy values are shifted so that their minima lies at zero (in the program, they are stored with the *average* value lying at zero). When plotted as a 19×19 matrix using the corresponding values of ϕ_1 and ϕ_2 as axes labels, the results is a nice-looking torsional PES, as shown in Figure 5.2.

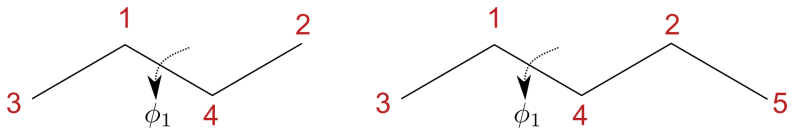

Figure 5.3: United-atoms butane and united-atoms pentane molecules.

5.3 Optimizing the Parameters

The files for this part of the tutorial are in `./doc/tutorial/profilerOpt`. The systems considered in this example are an united-atoms butane and an united-atoms pentane molecule, with one reference dihedral each (see Figure 5.3). The dihedral angle 5-2-4-1 of the pentane molecule was not chosen as a reference dihedral because the quantum-mechanical data for this molecule corresponds to a 1D scan along the 3-1-4-2 dihedral angle. The optimization of the Lennard-Jones parameters is done *via* the CH2-CH3 and CH3-CH3 pair types.

5.3.1 Understanding the Input-Parameters (INP) File

When compared to `profilerGen`, `profilerOpt` introduces a new input file (the INP file) where the settings of the parameter optimization are configured. A complete description of the syntax of the INP file is available in Section 4.1.2. In this section, we briefly go over the contents of the INP files used in this part of the tutorial.

CMA-ES

The INP file for the run based on CMA-ES is `tutorial/profilerOpt/run_cmaes.inp`. The first block of this file is

```
[ parameter_optimization ]
; NTORS   NLJ    FTORS
      1     2        1
; TORSNT
      0     0     2     0     0     0
; LJNT
      1   1
      1   1
; WTEMP
0
```

NTORS controls the number of optimized torsional types. NTORS = 1 means that a single torsional type is optimized. NLJ controls the number of optimized Lennard-Jones types. NLJ = 2 means that two Lennard-Jones types are optimized (one for each pair type). FTORS controls the functional form of the torsional type in optimization. FTORS = 1 means that the functional form is the standard periodic one (Equation 2.1.6). The matrix TORSNT, with NTORS rows and 6 columns, controls which summands of the torsional potential energy function are considered for each torsional type, and also whether to optimize only the force constant or also the phase. In this application, it is set such that only the term with multiplicity 3 (third column) is used in the torsional type, optimizing both the force constant and the phase (value is 2). The matrix LJNT, with NLJ rows and 2 columns, controls which components of the Lennard-Jones potential are optimized for each Lennard-Jones type. In this application, it is set such that both the attractive (first column) and repulsive (second column) components are optimized, for both types. Finally, WTEMP is the temperature used for Boltzmann-weighting of the reference data. WTEMP = 0 actually means an infinite temperature, which is equivalent to setting all weights to 1.

The second block is

```
[ parameter_randomization ]
; PINV
    50
; DIST[I]  MIN[I]    MAX[I]     MEAN[I]  STDDEV[I]
        2       0        30        20.0        8.0
        2       0         1        8e-3      24e-3
        2       0         1        8e-6      24e-6
```

and contains the details of the obtention of random parameters. The distributions (DIST) from which torsional, CS_6 and CS_{12} values are drawn, respectively, are specified along with their mean (MEAN), standard deviation (STDDEV) and minimum (MIN) and maximum allowed (MAX) values. With the values above, all of these distributions are LogNormal (DIST = 2). In the case of torsional parameters, after randomly obtained, they may have their sign reversed with a probability of 0.50 (PINV).

The third block is

```
[ evolutionary_strat ]
; STRAT
      1
; POPSIZE    NGENS
       20       20
```

and contains the specification of the evolutionary algorithm used to perform the optimization. STRAT = 1 sets the algorithm to CMA-ES. POPSIZE = 20 and NGENS = 20 set the size of the population and the number of generations, respectively, to 20. Note that it usually takes a larger number of generations, with a larger population, to achieve convergence in the results of a `profilerOpt` job. The values in this example are merely for the purpose of illustration.

The fourth block is

```
[ seed ]
    1485
```

and sets the RNG seed to 1485.

The fifth block is

```
[ minimization ]
;   MALG     DXO  DXM  NMAX  DELE
       1    0.05  0.2  5000  1e-5
       1   0.001  0.1 10000  1e-9
```

and contains the settings of the minimization algorithms. They correspond to two steepest-descents algorithms (MALG = 1) of different precision (DELE) and duration (NMAX). The first one refers to the routine minimization performed for every individual; it runs for at most 5000 steps or until the energy difference between successive steps is less than 1×10^{-5} kJ mol^{-1}. The second one refers to the final minimization performed only for the optimal individual; it runs for at most 10000 steps or until the energy difference between successive steps is less than 10^{-9} kJ mol^{-1}.

The last block is

```
[ torsional_scan ]
;  NTSCAN
        1
;  NRESTR
        1
;   RFRST  RSTEP  RLST   RFCT
        0     10   360   5000
```

and specifies the settings of the torsional scan. NTSCAN = 1 means that the torsional-scan angles are generated by the program based on a multidimensional systematic scan. NRESTR = 1 means that a single set of dihedral-restraint settings is specified. Note that NRESTR *must be equal* to the number of reference-dihedral groups in *each and all* of the STP files provided to `profilerOpt`. The NRESTR uncommented lines following specify the systematic-scan parameters (RFRST, RSTEP and RLST) and the force constant (RFCT) that apply to all the dihedrals of each reference-dihedral group. RFRST, RSTEP and RLST are ignored if the torsional-scan angles originate from an external source (for example, from a SPEC file, which can be achieved with NTSCAN = 3). The molecules considered in this example have only one reference-dihedral group, with one dihedral angle that is restrained to the values $(0°, 10°, \ldots, 360°)$ using a force constant of 5000 kJ mol^{-1} rad^{-2}.

LLS-SC

The INP file for the run based on LLS-SC (`tutorial/profilerOpt/run_lls-sc.inp`) differs from the one for CMA-ES by:

- removal of the block `[ evolutionary_strat ]`, to switch off the use of evolutionary algorithms;
- removal of the block `[ parameter_randomization ]`, which is not needed to use LLS-SC;
- inclusion of the block `[ lls_sc ]`, to switch on the use of LLS-SC.

The included block reads

```
[ lls_sc ]
; ITER      DPAR
   40   1.0e-04
; SREG
    0
```

It sets a LLS-SC algorithm that runs for at most 40 iterations (ITER = 40) or until the relative changes in the optimized parameters is less than 1.0×10^{-4} in an iteration (DPAR = 1.0e-04). Regularization is not used (SREG = 0), so the other input variables of this block, such as CREG and LREG (see Section 4.1.2), are not needed.

5.3.2 Optimizing

To carry out the optimization, it is necessary to specify the atom indexes of the reference dihedrals, as before, and also those of the dihedral angles and pairs to which the optimized parameters will apply. This is done in the STP files `./doc/tutorial/profilerGen/bu.stp` and `./doc/tutorial/profilerGen/pe.stp`, together with the specification of the topology itself. In particular, the reference dihedrals for butane and pentane have the same atom indexes:

```
[ refdihedrals ]
; idx
3   1   4   2
```

The optimized dihedral angles are a bit more trickier to configure. For butane, it is straightforward:

```
[ optdihedrals ]
; idxs
3   1   4   2
```

However, for pentane, they are *both* 3-1-4-2 and 5-2-4-1, since the optimized parameters are applied to both of these dihedrals. They are listed in the same `[ optdihedrals ]` block, since they are modelled by the same torsional parameters:

```
[ optdihedrals ]
; idxs
    3    1    4    2
    5    2    4    1
```

It is important to understand the following difference: the dihedral angle 5-2-4-1 *is not* a reference dihedral—its torsion is relaxed (as opposed to restrained) at each torsional-scan configuration. However, it *is* an optimized dihedral—it is modelled by the optimized torsional parameters obtained by `profilerOpt`. This situation is, of course, not ideal, since, in general, we want each optimized dihedral to also be a reference dihedral, in order to better isolate its influence on the torsional-scan energy profile. However, this ideal scenario is limited, in practice, by the cost of obtaining reference quantum-mechanical data for a large number of reference dihedrals. In this particular case, we only have such data for the torsional scan along the 3-1-4-2 dihedral, and this is why this is the only reference dihedral angle considered in this example.

The optimized pairtypes are specified in the `[ optpairs ]` blocks. There will be two such blocks per molecule, one for the CH2-CH3 and another for the CH3-CH3 pairtype. In `bu.stp` (see Figure 5.3):

```
[ optpairs ]
; CH2-CH3 empty

[ optpairs ]
; CH3-CH3
2   3
```

In `pe.stp` (see Figure 5.3):

```
[ optpairs ]
; CH2-CH3
1   5
2   3

[ optpairs ]
; CH3-CH3 - empty
```

Contrary to `profilerGen`, `profilerOpt` requires complete torsional-scan trajectories as input. The trajectories `./doc/tutorial/profilerOpt/atoms/bu.xyz` (of butane) and `./doc/tutorial/profilerOpt/atoms/pe.xyz` (of pentane) each consists of 37 configurations where the reference torsional-scan dihedral angle assumes, in turn, the values $(0°, 10°, \ldots, 360°)$. These input trajectories can be obtained with `profilerGen`, as illustrated in Section 5.2.

The reference energy values for the optimization can be found in the files `bu.dat` (of butane) and `pe.dat` (of pentane) in the same directory. They are single-column files where the k-th row is the reference energy for the k-th torsional scan configuration.

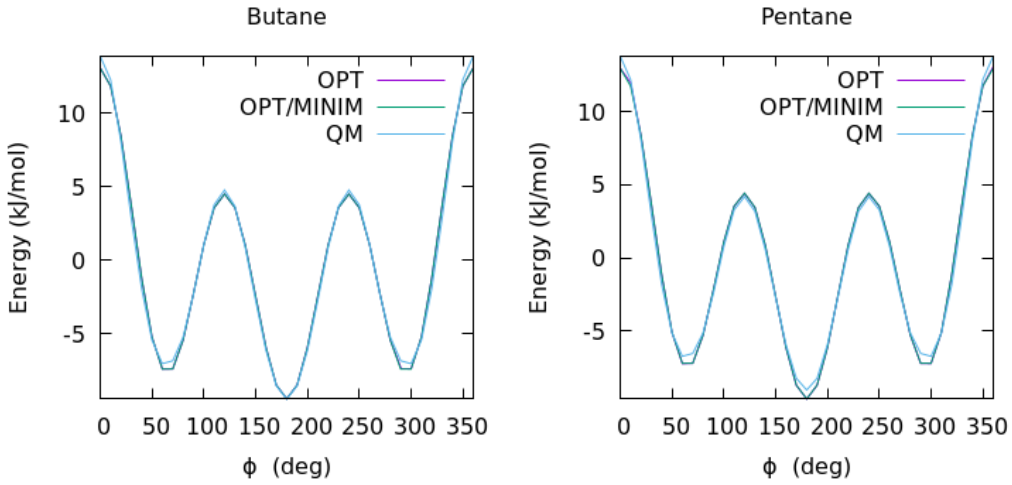

Figure 5.4: Optimization results for butane and pentane.

To run the `profilerOpt` job, enter the appropriate directory

```
$  cd doc/tutorial/profilerOpt
```

and run `profilerOpt` using the appropriate input files and command-line arguments (run `profilerOpt -h` for an overview). For the job based on CMA-ES:

```
$  profilerOpt -np 8 -r bu.dat pe.dat -c bu.xyz pe.xyz -t bu.stp pe.stp -i run_cmaes.inp -op cmaes
```

For the job based on LLS-SC:

```
$  profilerOpt -r bu.dat pe.dat -c bu.xyz pe.xyz -t bu.stp pe.stp -i run_lls-sc.inp -op lls-sc
```

The CMA-ES job is parallelized across 8 processors for the sake of speed. This number can be incremented or decremented depending on the machine used to follow this tutorial. The other differences between the two jobs above are in the input file (option `-i`), which sets the optimization algorithm, and in the prefix for the output files (option `-op`).

A successful run of `profilerOpt` will create several output files, and their formats are described in Section 4.1. We comment below each of these output files using the LLS-SC job as a basis.

The IFP file (`lls-sc.ifp`) reads

```
Type 0
     cs6 = 1.2678551e-03
    cs12 = 2.4554817e-06
Type 1
     cs6 = 2.3904714e-03
    cs12 = 2.8299851e-06
Type 2
       k_3 = 6.9972309e+00
     phi_3 = -5.0528647e-03
rmsd = 0.4215987974404587
```

and contains the main results of the parameter optimization. The last line contains the value of the weighted root-mean-square deviation (in $\mathrm{kJ\,mol^{-1}}$) for the best individual. The lines above list the optimized parameter values for each parameter type. The first parameters are those of the Lennard-Jones types, in the order they are defined in the STP and INP files. Therefore, in this case, type 0 applies to the CH2-CH3 pairtype, while type 1 applies to the CH3-CH3 pairtype. After that, the parameters of the torsional types are listed, also in the order they are defined in the STP and INP files. Therefore, in this case, type 2 is the single torsional type defined in this example. The subscripts in the torsional parameters indicate the summand of the torsional potential function—in this case, the optimized parameters are the force constant and the phase of the term with multiplicity 3.

The energy profiles for the best individual are listed for two different energy-minimization settings. The files `lls-sc_1.dat` and `lls-sc_2.dat` contain the energy profiles considered during the execution of the optimization algorithm. The order of the systems follows the order their files were written in the command-line arguments: first butane, then pentane. Each DAT file has a single-column and a number of rows equal to the length of the torsional scan—in this case, 37 data points. The value in the k-th row is the energy of the k-th configuration of the torsional scan (see Section 3.1). The files `lls-sc_minim_1.dat` and `lls-sc_minim_2.dat` follow the same structure, but

contain refined energy values obtained after re-running the torsional scan of the optimal individual with enhanced energy-minimization settings (read the explanation of the `[ minimization ]` block in Section 5.3.1). These energy profiles are plotted together with the reference data (`bu.dat` and `pe.dat`) in Figure 5.4. In this case, the results for the original and enhanced energy-minimization settings are almost indistinguishable. Note that, for comparison, the reference data was subtracted by the average value for each molecule.

The XYZ files `lls-sc_1.xyz`, `lls-sc_2.xyz`, `lls-sc_minim_1.xyz` and `lls-sc_minim_2.xyz` are the torsional-scan trajectories corresponding to the energy profiles described above. In these files, the k-th frame is the configuration of the k-th step of the torsional scan.

Bibliography

[1] M. J. Abraham, D. van der Spoel, E. Lindahl, B. Hess, and the GROMACS development team. *GROMACS User Manual version 2019*.

[2] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[3] Y. M. H. Gonçalves, S. Kashefolgheta, M. P. Oliveira, P. H. Hünenberger, and B. A. C. Horta. Simultaneous parametrization of torsional and third-neighbor interaction terms in force-field development: the LLS-SC algorithm. *Submitted*, 2021.

[4] W. F. van Gunsteren, S. R. Billeter, A. A. Eising, P. H. Hünenberger, P. Krüger, A. E. Mark, W. R. P. Scott, and I. G. Tironi. *Biomolecular Simulation: The GROMOS96 Manual and User Guide. Volume 2: Algorithms and Formulae for Modelling of Molecular Systems.* Vdf Hochschulverlag AG an der ETH Zürich, Zürich, Switzerland, 1996.

Appendix A

MIT License