Table of Contents

[Optimization scheme involving molecular dynamics 1](#_Toc345334936)

[Simulation server module 2](#_Toc345334937)

[The molmec package 3](#_Toc345334938)

[Force-Field template creator 3](#_Toc345334939)

[Dakota input file generator 4](#_Toc345334940)

[Force-field update script 4](#_Toc345334941)

[The simhost package 5](#_Toc345334942)

# Optimization scheme involving molecular dynamics

Optimization of force-fields employed in molecular dynamics (MD) simulations *via* fit to experimental dynamic neutron scattering structure factors requires optimization of two fundamentally different set of parameters. First, one seeks to optimize those parameters of the force-field that will be subsequently used in production MD runs (partial charges, Lennard-Jones potentials, and bonded force-constants). Second, one must also optimize extra parameters pertaining to the specifics of the neutron scattering experiment (background, beam intensity, and resolution function). Search of the optimal values for the first parameter set require new MD simulations runs every time new values are tried, simulations that usually complete in a time-scale of hours to days. On the other hand, search of the optimal values for the second parameter set require no MD simulation and the optimization completes under a second. Thus, the problem leads itself into an optimization strategy where the two parameter sets are optimized at different stages but whithin a global optimization loop.

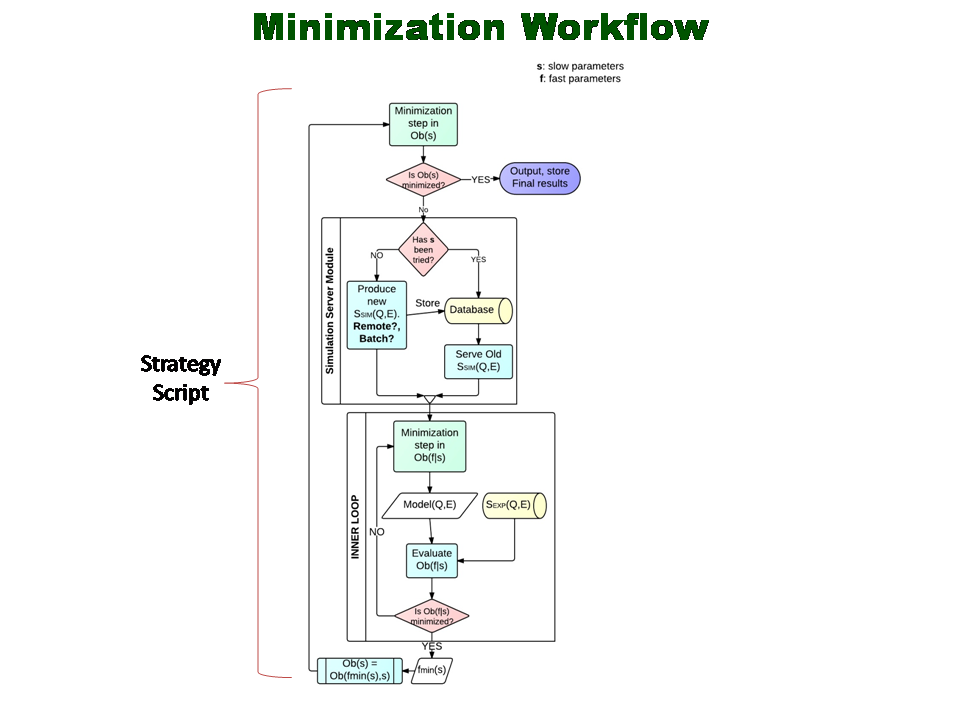


Figure 1: Optimization loop. Optimization of the objective function *Obs(f,s)*, is carried out in two stages. An inner loop optimizes the function under constant force-field parameters *s*, and then the optimal experimental parameters found *f=fmin(s)* are use to yield an objective function that depends only on the force-field parameters.

# Simulation server module

Every time the force-field parameters are changed to new values, a computation of the simulated structure factor must be realized. This computation involves a chain of jobs involving MD simulation(s) and the subsequent computation of the intermediate and final dynamic structure factors. First the MD must be called, which can be a single or a compound run, depending on the system to be simulated and the available computer resources. Next, calculation of the intemediate dynamic structure factor *I(Q,t)* must be called, which in itself is time consuming. Finally, Fourier transform of *I(Q,t)* yields the final dynamic structure factor *S(Q,E)*. A Database server will store the data generated in this chain of jobs. Conversely, the server will provide the data if the force-field parameter values have been already tried.

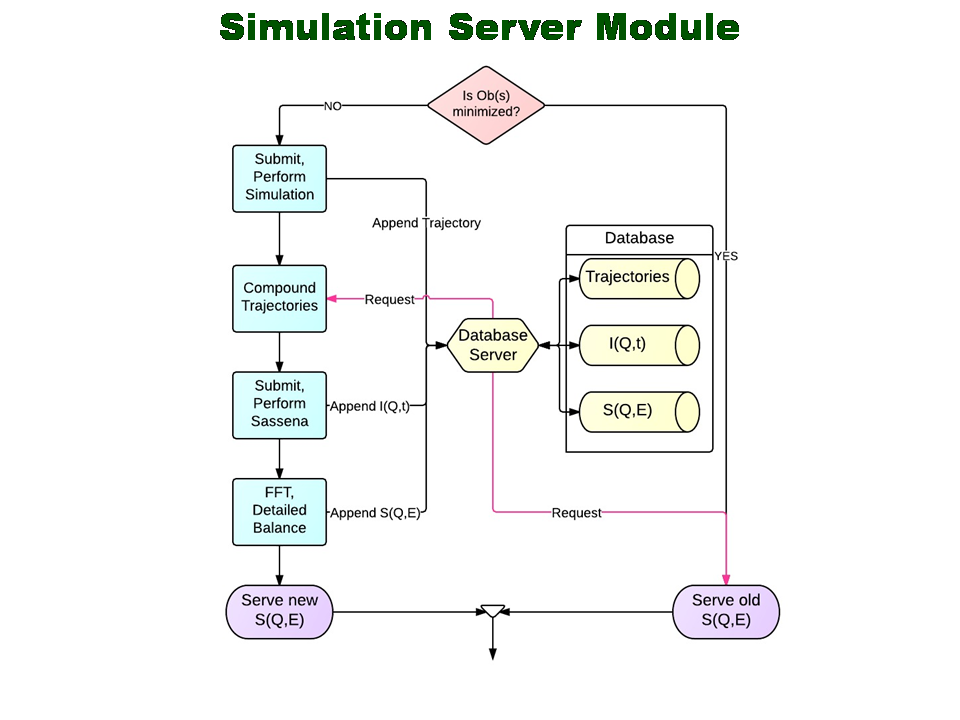


Figure 2. Chain of jobs. A decision is made whether to execute new MD and Sassena simulations to generate a new dynamics structure factor *S(Q,E)* or whether to serve and old *S(Q,E)*.

# The molmec package

We are developing a new open-source python package, molmec, for the purpose of managing all the necessary steps revolving around the molecular mecanics calculations within the overall optimization scheme of force-field parameters. These steps involve, among others: (*i*) preparation and update of force-field files in popular formats (CHARMM, LAMPS, AMBER, GROMACS); (*ii*) preparation of input and output files for the Dakota minimization package; (*iii*) preparation and management of the hierarchy of subdirectories housing the MD and Sassena runs; (*iv*) preparation and update of the input files for each MD run; and (*v*) file format adaptation to the requirements of Sassena.

Configuration of these different steps will be first implemented through simple configuration files, although we plan to extend these with graphical user interfaces as the package gains in stability.

## Force-Field template creator

The current version of the molmec.fftpl module contains the algorithms necessary to select any partial charges in a CHARMM/XPLOR topology file as force-field parameters to be optimized. The module generates a template topology file that can be updated every time the optimization scheme requires trial of new values for the parameter set.

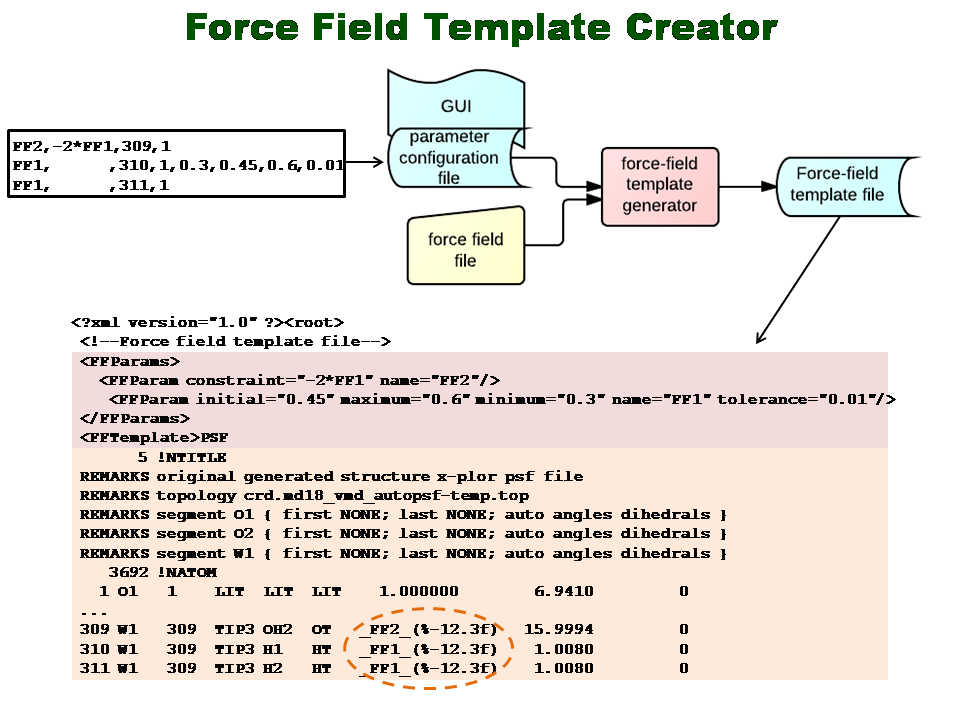


Figure 3. The force-field template creator reads-in a simple configuration file specifying the force-field parameters to be optimized and generates a template CHARMM topology file (in XML format).

## Dakota input file generator

The current version of the molmec.dakota module reads in a templated Dakota input file and inserts the force-field parameters to be optimized with the help of the template topology file. Due to the enormous variety of Dakota options, we plan to offer a limited selection of relevant templated Dakota input files, along with the option allowing experienced users to provide their own. The resulting file will serve as the starting input file for the whole optimization scheme.

## Force-field update script

The current version of the molmec.ffupdate module uses the CHARMM topology template file and the request generated by Dakota in order to produce an updated topology file, with new values for the force-field parameter set to be optimized. This file will be used in the subsequent MD simulations.

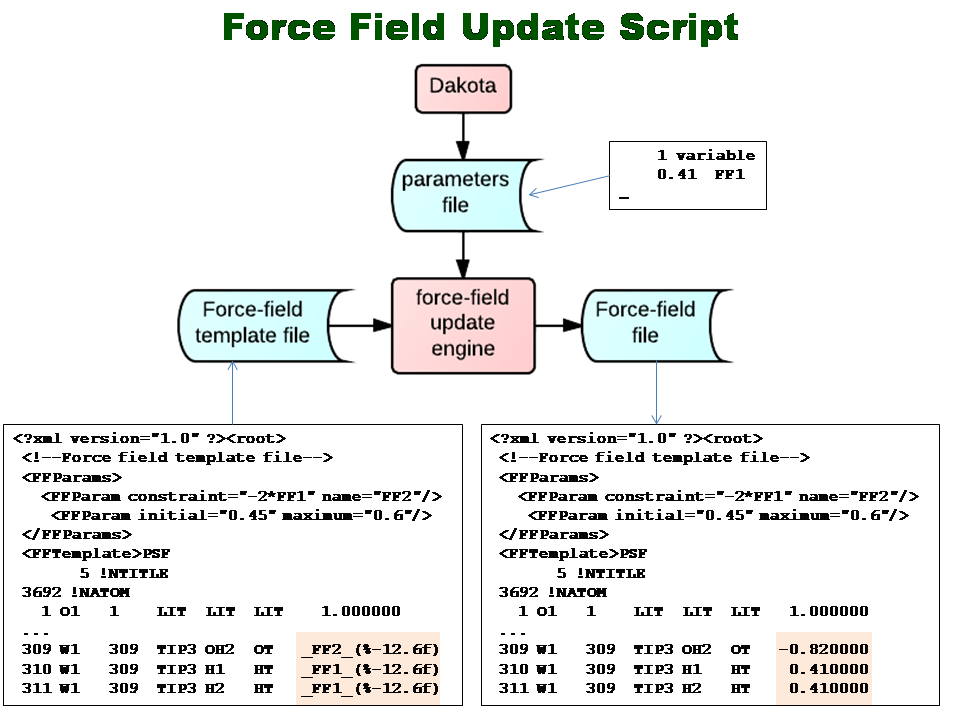


Figure 4. Update of the topology file with a single fit parameter. The templated topology file shows the partial charge of the hydrogen atom in the water molecule as the force-field parameter to be optimized (*FF1*). A second parameter (*FF2*) is required to ensure that the water molecule has a neutral charge. Parameter *FF2* is thus determined by the value of *FF1*. The Force-field update engine takes care of such constraints among force-field parameters. The correct format (%-12.6) is ensured in the template topology file.

# The simhost package

We are developing a new open-source python package, simhost, for the purpose of managing correct scheduling and execution of MD and Sassena simulations in high-performance clusters. The current version of the package contains host, server, and queue modules. Typical tasks to be carried out by the package are: (*i*) query of cluster resources; (*ii*) estimation of required computer time, depending on simulation-system size and cluster resources; (*ii*) creation of on-demand PBS submission scripts for MD and Sassena simulations; and (*iii*) ensure dependencies among the PBS scripts when necessary.