Alexandria Chemistry Toolkit

Version 0.99

David van der Spoel, Paul J. van Maaren and Mohammad M. Ghahremanpour

March 2, 2025

Contents

List of Code Listings

List of Figures

Preface

This manual is a work in progress, covering the physical background of the force field development in ACT, from potential energy functions to the molecular properties supported by version 1.0 and algorithms used for training.

For information on how to install and use the ACT we refer to the on-line documentation for now [58].

Throughout this document we highlight **alexandria commands** like this, and command line options like this. There are also hyperlinks in the document, both for the references and in the running text, often linking to Wikipedia.

The authors welcome suggestions for additions and improvements as well as corrections for factual errors. Please mail your comments to david.vanderspoel at icm.uu.se.

We kindly request that you cite the paper about ACT and other relevant works if you use the software for a scientific publication of your own.

Happy force field designing!

1 Introduction

The Alexandria Chemistry Toolkit (ACT) allows for global optimization of force fields for molecular simulation. That means that, in principle, a whole force field can be derived at once by tuning parameters to reproduce results from *ab initio* or density functional theory as well as experimental data when available.

The fundamental motivation for the development of the ACT are given in these reviews [53, 57, 34]. Stepping stones on the way to a new family of physics-based force fields are given below.

1. A long series of benchmarks of force fields [5, 14, 72, 73, 55, 46, 23].
2. Databases of quantum-chemical data produced as part of the Alexandria project: enthalpies of formation and thermochemistry [16] applied to force fields here [55].
3. The Alexandria Library with more quantum-chemistry data [19, 17].
4. A phase-transferable force field for alkali halides [66] with applications towards melting points and conductivities [62, 63, 64, 65].
5. New models for noble gases including accurate simulations of the melting point of solid noble gases [35].
6. Use of symmetry-adapted perturbation theory for a study of exchange around a water molecule [33].
7. An evaluation of potentials for chemical bonds in molecules [60].

The following sections present, somewhat rudimentary at the time of writing, the physical foundation of the ACT models, the algorithms for training force fields as well as the properties that can be produced by the ACT software.

The last few section describe installation and practical usage of the ACT.

2 Force field design

2.1 Physical Background

The Alexandria force-fields have a functional form consisting of van der Waals (*vdw*), electrostatics (*coul*), polarization (*pol*) and bonded terms including a radial (*b*), angular (*a*), out-of-plane dihedral (*i*) and torsion (*d*) terms.

$$E = V_{vdw}(r_{ij}) + V_{coul}(r_{ij}) + V_{pol}(r_{cs}) + V_b(r_{ij}) + V_a(\theta_{ijk}) + V_i(\phi_{ijkl}) + V_d(\phi_{ijkl}) \quad (1)$$

where r_{ij} refers to the distance between atoms i and j, r_{cs} to the distance between a core and a shell (Drude) particle, θ_{ijk} to the angle given by three atoms i, j and k, and ϕ_{ijkl} to a torsional angle between the planes given by atoms i,j,k and atoms j,k,l. For each of the terms, multiple functional forms are available, so that within the Alexandria framework, different force-fields, including previously published ones, can be reconstructed and compared to one another in a systematic manner [53]. In total there are seven “atom" parameter types and 7-14 “bond" parameter types. A variety of virtual sites can be used, like those used in water models [59, 36] or to model anisotropy due to σ-holes on halogen atoms or water [33].

2.2 Determining partial charges

The electrostatic potential (ESP, Section S2) has historically been used to determine partial charges [2] and the ACT supports training models to reproduce the ESP. However, in a very recent paper we have shown that fitting charges to reproduce the ESP in a limited volume around a compound is fundamentally flawed due to lack of information [24]. If the purpose is to build models that reproduce electrostatic interactions, this can be done directly by training models to reproduce SAPT energy components. Here, the split-charge equilibration (SQE) algorithm [61] is used to generate the effective partial charge on each atom in a molecule. SQE, in turn, is based on the electronegativity equalization method (EEM), as developed by Rappé and Goddard [44]. In brief, EEM uses the atomic hardness η and electronegativity χ to determine the atomic charges in a molecule from a Taylor expansion of the molecular energy in terms of charges. The SQE algorithm introduces a correction to the atomic electronegativities for bonded atoms $\Delta\chi$ as well as a bond hardness $\Delta\eta$. With this addition, charge can “flow" through bonds only, which overcomes issues with over-polarization in the EEM [41]. The

ACT code implements the possibility to generate charges for compounds in dimers or clusters where charge transfer between compounds is disallowed which is a reasonable approximation since charge transfer has been shown to have limited impact on the binding energy of non-covalent complexes [71]. For the SQE algorithm two atomic parameters (χ and η) as well as two bond parameter types ($\Delta\chi$ and $\Delta\eta$) need to be determined and the ACT can train SQE parameters to reproduce electrostatic and induction energies [24].

3 Energy Function

The ACT supports a range of different potentials for classical force field simulations. These potentials are described briefly here.

3.1 Non-bonded interactions

3.1.1 Coulomb Interaction

The Coulomb interaction can be described using Gaussian shielded charges

$$V_{coul}(r_{ij}) = \frac{q_i q_j \mathrm{erf}\left(\zeta_{ij} r_{ij}\right)}{4\pi\varepsilon_0\varepsilon_r r_{ij}}, \quad \zeta_{ij} = \frac{\zeta_i \zeta_j}{\sqrt{\zeta_i^2 + \zeta_j^2}}, \tag{2}$$

where ε_0 is the permittivity of vacuum, $q_{i,j}$ are the charges and $\zeta_{i,j}$ are the charge distribution widths (screening factors). Note that Eqn. 2 includes a relative dielectric constant ε_r that can be used to parameterize a non-polarizable force field using charge-scaling [31].

Alternatively, a standard Coulomb interaction can be used with point charges:

$$V_{coul}(r_{ij}) = \frac{q_i q_j}{4\pi\varepsilon_0\varepsilon_r r_{ij}} \tag{3}$$

and, in addition, Slater distributed charges can be used as decribed in ref. [18, 59, 66].

3.1.2 Long range Coulomb interactions

It is good practice [52] to use the particle-mesh Ewald [7, 13] method to treat long-range electrostatics interactions in the condensed phase. In short, the method splits the Coulomb potential, either Eqn. 3, Eqn. 2 or something similar into a short and a long-range part as follows

$$V_{coul}(r_{ij}) = V_{coul}\,\mathrm{erfc}(\alpha r_{ij}) + V_{coul}\,\mathrm{erf}(\alpha r_{ij}) \tag{4}$$

using the fact that the complementary error function erfc equal 1 - erf. The first term here is the short-range part, that is computed in real space, and the second term is the long-range part that is computed in Fourier (reciprocal) space by moving charges on a regular grid [13]. The constant α determines how quickly the short-range potential decays to zero, and it can be computed

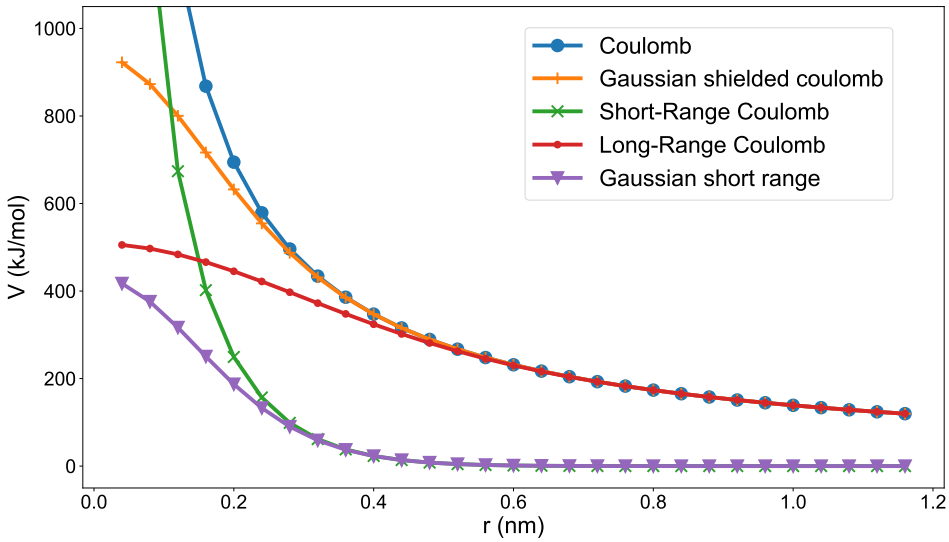

Figure 1: Total Coulomb energy V as a function of distance r for two like unit charges, Gaussian-shielded Coulomb and short and long-range parts of the energy function. Ewald-tolerance ϵ 0.0001, rc 0.9 nm, α 3.24269/nm and Gaussian screening ζ 5/nm.

from a user-specified relative error tolerance related to moving charges on a grid. Given the cut-off distance *rc* and an error tolerance ϵ, typically 10^{-4}, we have

$$\alpha = \frac{\sqrt{-\ln 2\epsilon}}{rc} \tag{5}$$

where ln is the natural logarithm. This shows that α has the dimension of 1 over distance in the units of rc. Fig. 1 shows how the division over short and long-range interactions works in practice, and how the long range contribution should be incorporated into simulations using a modified Coulomb function.

3.1.3 Polarization

Polarization can be treated explicitly as well in the ACT using the core-shell model [9, 29]

$$V_{pol}(r_{cs}) = \frac{1}{4\pi\varepsilon_0}\frac{q_s^2}{2\alpha_c}r_{cs}^2, \tag{6}$$

where q_s is the shell charge, α_c is the polarizability and r_{cs} the distance between core and shell particles.

3.1.4 Induction correction

A term to add extra attraction between atoms was proposed in ref. 39

$$V_{ic}(r_{ij}) = -A_{ij}e^{-b_{ij}r_{ij}} \tag{7}$$

with A_{ij} and b_{ij} constants to be trained. The geometric combination rule is used for A_{ij} and the arithmetic combination rule for b_{ij}. It is not certain whether the exponential functional form is optimal to describe this interaction and there is no theory behind this.

3.1.5 Pauli Repulsion and London Dispersion

The repulsion and dispersion interactions acting between atoms *i* and *j* can be described by a number of potentials. In a recent study on noble gases [35] we found that the 14-7 potential due to Halgren [21] was one of the most accurate ones:

$$V_{14-7}(r_{ij}) = \epsilon_{ij}\left(\frac{1+\delta_{ij}}{\frac{r_{ij}}{\sigma_{ij}}+\delta_{ij}}\right)^7\left(\frac{1+\gamma_{ij}}{(\frac{r_{ij}}{\sigma_{ij}})^7+\gamma_{ij}}-2\right) \tag{8}$$

where ϵ_{ij} is the well depth at minimum, γ_{ij} and δ_{ij} are dimensionless numbers that were originally shared for all elements. However, in our previous work [35], we treated γ and δ as free atom-specific parameters subject to optimization and combination rules.

Furthermore, the generalized 4-parameter Buckingham potential with an adjustable long-range attraction is implemented as[70]

$$V_{GBH}(r_{ij}) = \epsilon_{ij}\frac{\delta_{ij}+2\gamma_{ij}+6}{2\gamma_{ij}}\frac{1}{1+(\frac{r}{\sigma_{ij}})^6}\left[\frac{6+\delta_{ij}}{\delta_{ij}+2\gamma_{ij}+6}e^{\gamma_{ij}(1-\frac{r}{\sigma_{ij}})}-1\right]-\frac{\epsilon_{ij}}{1+(\frac{r_{ij}}{\sigma})^\delta} \tag{9}$$

where γ and δ again are dimensionless constants.

Another potential that is supported is the buffered (Wang) Buckingham potential [68]:

$$V_{WBH}(r_{ij}) = \left(\frac{2\epsilon_{ij}}{1-\frac{3}{\gamma_{ij}+3}}\right)\left(\frac{\sigma_{ij}^6}{\sigma_{ij}^6+r_{ij}^6}\right)\left[\frac{3}{\gamma_{ij}+3}e^{\gamma_{ij}\left(1-\frac{r_{ij}}{\sigma_{ij}}\right)}-1\right] \tag{10}$$

where ϵ_{ij} is the well depth at minimum, σ_{ij} is the Van der Waals radius and γ_{ij} determines the steepness of the potential and r_{ij} is the distance between particles i and j. The buffered Buckingham potential mentioned above have been used to develop an accurate phase-transferable model for alkali-halides [66, 62, 63, 64, 65]. and the original Buckingham potential [3] is implemented as

$$V_{BH}(r_{ij}) = A_{ij}\mathrm{e}^{-b_{ij}r_{ij}} - \frac{C_{ij}}{r_{ij}^6} \tag{11}$$

where A_{ij}, b_{ij} and C_{ij} are parameters to be trained. The well-known Tang-Toennies potential [48, 49] containing five parameters is implemented according to:

$$V_{TT}(x) = Ae^{-bx} - \sum_{n=3}^{5}\left[1 - e^{-bx}\sum_{k=0}^{2n}\frac{(bx)^k}{k!}\right]\frac{C_{2n}}{x^{2n}} \tag{12}$$

where repulsion and dispersion terms shared the parameter b. Here, all five parameters can be trained by the ACT. This potential was used in our recent work on noble gases published where we investigated combination rules [35].

Finally, the Lennard-Jones 12-6 potential [28] is available as well:

$$V_{LJ}(r_{ij}) = 4\varepsilon_{ij}\left(\frac{\sigma_{ij}^{12}}{r_{ij}^{12}} - \frac{\sigma_{ij}^{6}}{r_{ij}^{6}}\right). \tag{13}$$

It is worth noting that in all these potentials (except Buckingham, Eqn. 11) parameters describe both the exchange and dispersion interactions at the same time, which necessitates simultaneous training of these interaction. To account for anisotropy in exchange, a correction can be implemented using a virtual site particle on the atom that has a σ-hole interacting with other particles [33]:

$$V_{EXCH,Corr}(r_{ij}) = -A_{ij}\mathrm{e}^{-b_{ij}r_{ij}} \tag{14}$$

where A_{ij} and b_{ij} are parameters to be optimized. This correction term (Eqn. 14) does not need to be applied to all possible atom pairs, therefore a new combination rule, dubbed "Kronecker" is introduced

$$A_{ij} = (1 - \delta_{ij})\frac{A_i + A_j}{2} \tag{15}$$

where δ_{ij} is the Kronecker delta operating on particle types i and j. This means that only interactions between σ-holes (represented by a virtual site) and atoms are non-zero. To avoid parameter explosion, the atomic A_i are set

to zero and only the virtual site A_j is non-zero. This is reasonable since the virtual site represents a property of the σ-hole of the atom it is connected to. The arithmetic combination rule is used for b_{ij}.

Multiple different combination rules can be used for parameters involved in van der Waals interactions, both within ACT and through the interface to OpenMM. For details, see the paper by Kříž *et al.* [35].

3.1.6 Treatment of long range dispersion

The different Van der Waals potentials described above can be used with Lennard-Jones PME [7, 69] if treated carefully. LJ-PME assumes a simple dispersion interaction given by

$$V_{LJPME}(r_{ij}) = -\frac{C_{ij}}{r_{ij}^6} \tag{16}$$

for atoms *i* and *j*, which differs from the potentials mentioned above. For LJ-PME, it is beneficial to use the geometric combination rule, hence

$$V_{LJPME}(r_{ij}) = -\frac{C_i C_j}{r_{ij}^6} \tag{17}$$

where C_i, in contrast to Eqn. 16, have units of distance to the third power.

To minimize the difference between the potential used and the LJ dispersion over the whole volume outside the cut-off, and to specify correct inputs to OpenMM, starting from e.g. Eqn. 10 we have to solve the following equation:

$$0 = \int_{rc}^{\infty} 4\pi r_{ij}^2 \left[\left(-\frac{2\epsilon_{ij}}{1 - \frac{3}{\gamma_{ij}+3}} \right) \left(\frac{\sigma_{ij}^6}{\sigma_{ij}^6 + r_{ij}^6} \right) + \frac{C_i C_j}{r_{ij}^6} \right] \mathrm{d}r_{ij} \tag{18}$$

where we integrate from the cut-off *rc* to infinity. In this notation, we have to insert the combination rules for ϵ, σ and γ. For the special case that *i* = *j*, the constant C_{ii} can be derived using Mathematica™ as

$$C_{ii} = \epsilon_i \frac{3 + \gamma_i}{\gamma_i} rc^3 \sigma_i^3 \left(\pi - 2\arctan\left[\left(\frac{rc}{\sigma_i} \right)^3 \right] \right) \tag{19}$$

which we can then convert back to a σ' compatible with standard Lennard-Jones by

$$\sigma_i' = \left(\frac{C_{ii}}{4\epsilon_i} \right)^{1/6}. \tag{20}$$

Note however, that we need to get effective σ' for all atoms, which means the problem turns into a minimization problem where

$$\varepsilon^2 = \sum_{ij} \left[\int_{rc}^{\infty} 4\pi r_{ij}^2 \left[\left(-\frac{2\epsilon_{ij}}{1 - \frac{3}{\gamma_{ij}+3}} \right) \left(\frac{\sigma_{ij}^6}{\sigma_{ij}^6 + r_{ij}^6} \right) + \frac{C_i C_j}{r_{ij}^6} \right] \mathrm{d}r_{ij} \right]^2 \tag{21}$$

has to be minimized with respect to C_i independently for each set of combination rules. In the Alexandria alkali-halide model we used the Wang-Buckingham potential (Eqn. 10) in conjunction with the combination rules due to Kong [32] in which σ_{ij} depends on all the σ, ϵ and γ. This makes it cumbersome to analytically derive integrals like Eqn. 21 for many combinations of potentials and combination rules.

Another problem is, that the parameters C_i that we need to compute the long-range dispersion interaction now depend on the cut-off, which means they should preferably not be computed beforehand. In summary, the C_i should be derived numerically given the potential and combination rules at the start of a simulation.

3.1.7 Combination Rules

3.2 Bonded interactions

3.2.1 Harmonic potential

Bond vibrations can be described using a harmonic term based on the bond length r_{ij}

$$V_b(r_{ij}) = \frac{k_{ij}^b}{2} \left(r_{ij} - r_{ij}^0 \right)^2, \tag{22}$$

where k_{ij}^b is the force constant, and r_{ij}^0 is the equilibrium bond length.

3.2.2 Morse potential

A Morse potential [40] can be used, with one addition:

$$V_M(r_{ij}) = D_{ij}^e \left[\mathrm{e}^{-2\beta_{ij}(r_{ij} - r_{ij}^0)} - 2\mathrm{e}^{-\beta_{ij}(r_{ij} - r_{ij}^0)} \right] + D_{ij}^0 \tag{23}$$

The term D_{ij}^e roughly corresponds to a dissociation energy, however since there are Coulomb and/or Buckingham interactions between the atoms as well, the total "bond" potential is given by the sum of three terms and a correction term D_{ij}^0 is needed to get the correct energy minimum.

3.2.3 Wei-Hua potential

In a very recent study we found the potential due to Hua [25, 26] to be the best compromise between accuracy of the vibrational frequencies and computational cost [60]. It is given by

$$U(r) \;=\; D_e \left(\left[\frac{1 - e^{-b(r-r_e)}}{1 - ce^{-b(r-r_e)}} \right]^2 - 1 \right) \tag{24}$$

where D_e is the well-depth, r_e the equilibrium bond length, and b and c are constants with $\|c\| < 1$.

3.2.4 Angle potential

Angle vibrations are described using a harmonic term based on the angle θ_{ijk}

$$V_a(\theta_{ijk}) \;=\; \frac{k^{\theta}_{ijk}}{2} \left(\theta_{ijk} - \theta^0_{ijk}\right)^2, \tag{25}$$

where k^{θ}_{ijk} is the force constant, and θ^0_{ijk} is the equilibrium angle.

3.2.5 Angle potential for linear compounds

The reference position, corresponding to a minimum energy structure, $\mathbf{x}^0_j$ for a central atom j in a linear triplet of atoms i, j, k is given by

$$\mathbf{x}^0_j \;=\; \mathrm{a}\,\mathbf{x}_i + (1 - \mathrm{a})\,\mathbf{x}_k \tag{26}$$

where a is a constant defined by the bond-lengths $i - j$ and $j - k$. In a group with bonds $i - j$ and $j - k$ with lengths b_{ij} and b_{jk} respectively, the constant is

$$a \;=\; \frac{b_{jk}}{b_{ij} + b_{jk}}. \tag{27}$$

If the order of atoms is flipped a will change to $1 - a$. The potential V_{lin} is then given by

$$V_{lin} \;=\; \frac{\mathrm{k}_{lin}}{2} \left(\mathbf{x}_j - \mathbf{x}^0_j\right)^2 \tag{28}$$

with k_{lin} the force constant [56].

3.2.6 Out-of-plane vibrations

Finally, out-of-plane vibrations are treated by another harmonic potential

$$V_i(\phi_{ijkl}) = \frac{k^{\phi}_{ijkl}}{2}\phi^2_{ijkl}, \tag{29}$$

where k^{ϕ}_{ijkl} is the force constant and ϕ_{ijkl} is defined by the angle between the two planes i,j,k and j,k,l.

3.2.7 Torsion potential

A torsion potential is implemented using a Fourier series:

$$V_d(\phi_{ijkl}) = \sum_{n=0}^{5} c_n \cos^n(\pi + \phi_{ijkl}) \tag{30}$$

where c_n are constants and the torsion angle is defined as above. The constant π is added to be compatible with the Ryckaert-Bellemans potential [45] that is implemented in simulation codes like GROMACS [54] and OpenMM [11].

3.3 Virtual Sites

A virtual site is an extra point, located at a defined position in a molecule. A variety of virtual sites option is currently implemented within the ACT framework:

1. a virtual sites along the bond for the description of anisotropic charge distribution and exchange [33] such as encountered in σ holes
2. a virtual site on the bisector of a angle, like in the TIP4P water model [30]
3. off-plane virtual sites for modeling lone-pairs in sp^3 hybridized compounds, such as water (symmetric) or asymmetric for compounds like alcohols [38, 33].

3.4 Total energy

The total energy E of a compound then follows from

$$E = V_{vdw}(r_{ij}) + V_{coul}(r_{ij}) + V_{pol}(r_{cs}) + V_b(r_{ij}) + V_a(\theta_{ijk}) + V_i(\phi_{ijkl}) + V_d(\phi_{ijkl}). \tag{31}$$

Finally, it should be noted that the number of excluded neighbors is user-configurable. That means that atoms that are covalently bonded can interact both through the Buckingham (or Lennard-Jones) and Coulomb potentials, and through the bonded potentials. The main reason for this is that the short-range Coulomb interactions yield polarization anisotropy that is difficult to reproduce by a non-interacting model. To make sure that the forces on the atoms in a molecule are zero in the reference minimum-energy structure from quantum chemistry, both bond lengths r_{ij}^0 and angles θ_{ijk}^0 can be treated as free parameters, that may differ substantially from the reference geometry. The number of exclusions can be selected separately for Coulomb and Van der Waals forces.

In total there are up to seven "atom" parameter types (ϵ, σ, γ, δ, ζ, q_s, and α) and 7-14 "bond" parameter types (k^b, D^e, r^0, r^{max}, k^θ, θ^0, k^{lin} and c_n) where n is the dihedral term index running from 0 to 6 to determine. All the atomic parameters are taken to be hybridization state dependent (corresponding to, for instance, sp^1, sp^2 and sp^3 carbon atoms).

4 Molecular Properties

The ACT can be used to perform simulations of clusters in the gas-phase using the **alexandria simulate** command. This module includes the possibility to perform energy minimizations with the -minimize. For simulations employing periodic boundaries the OpenMM package[12] should be used instead.

Below we describe some of the properties that can be computed using the ACT.

4.1 Dimeric interaction energy

Computing interaction energies and components of interaction energies is a crucial part of force field development. In ACT we have hitherto used data from symmetry-adapted perturbation theory (SAPT) calculations [42]. It is not entirely trivial to match the energy components from SAPT to force field terms, however.

The component of the interaction energy of a dimer can be computed from the difference between dimer and monomers A and B [6]:

$$E_x^{inter}(AB) = E_x^{total}(AB) - (E_x(A) + E_x(B)) \tag{32}$$

where x is exchange or dispersion and *total* indicates that the energy includes both the intra- and intermolecular interactions. To compute the electrostatics or induction energy of a dimer in the gas phase, the ACT first computes the relaxed energy of the two monomers A and B, that is, the energy of the shell particles is minimized with respect to their positions, yielding $E_x(A)$ and $E_x(B)$. The rationale for this is that SAPT computes electrostatic energies between unperturbed monomers based on the response/relaxation of monomer Hartree-Fock (HF) orbitals in the electric field of the interacting partner. Then, the energies of the dimer AB are computed in three steps:

1. electrostatics is computed with shells located in the relaxed monomer positions, yielding $E_{elec}^{total}(AB)$,
2. the shells of compound A are allowed to relax (further) in the electric field from compound B, while shells of compound B remain at their monomer positions, and vice versa, yielding the second order relaxation $E_{induc}^{inter(2)}$ (see ref. 39 for details),

3. the shells are allowed to relax completely, yielding the total E_{induc} from which the higher order terms, named $E_{induc}^{inter(3)}$ here for convenience, can be derived by subtracting $E_{induc}^{inter(2)}$. According to ref. 39, parameters of models corresponding to the higher other terms, including, potentially, charge transfer, can be trained to the δHF contribution of the SAPT induction energy. Here, we have added the exponential term proposed by McDaniel and Schmidt for this purpose (section 3.1.4).

The terms below can be compared directly to SAPT:

$$E_{elec}^{inter}(AB) = E_{elec}^{total}(AB) - (E_{ei}(A) + E_{ei}(B)) \quad (33)$$

$$E_{induc}^{inter(2)}(AB) = \left(E_{ei}^{total}||_A + E_{ei}^{total}||_B\right) - 2E_{elec}^{total}(AB) \quad (34)$$

$$E_{induc}^{inter(3)}(AB) = E_{ei}^{total}(AB) - E_{induc}^{(2)}(AB) - E_{elec}^{total}(AB) \quad (35)$$

where *ei* is short for *elec* + *induc* and the notation $||_X$ indicates that the shells of compound *X* are kept fixed in the relaxed monomer conformation. If we sum Eqns. 33- 35 we recover Eqn. 32 where *x* equals *ei*. Eqn. 33 corresponds to the electrostatics in SAPT, Eqn. 35 to the δHF + δMP2 term, and Eqn. 34 to the induction term minus the delta terms.

4.2 Molecular Electrostatic Potential

The charge distribution ρ of a molecule is determined by the nuclear position of the atoms $\mathbf{x}$ and the electron density $n(\mathbf{r})$. The molecular electrostatic potential (MEP) at a point in space r_0 is thus given by

$$\Phi(\mathbf{r}_0) = \frac{1}{4\pi\varepsilon_0}\left[\sum_{i=1}^{N}\frac{z_i}{||\mathbf{x}_i - \mathbf{r}_0||} - \int_V \frac{n(\mathbf{r})}{||\mathbf{r} - \mathbf{r}_0||}\mathrm{d}V\right] \quad (36)$$

where *N* is the number of atoms, z_i are the nuclear charges, ε_0 is the permittivity of vacuum and integration is over the entire space *V*. The minus sign before the integral is due to the negative charge of electrons.

Accurate knowledge of the MEP contributes to, for example, the understanding of interactions and function of biological macromolecules in solution [37]. For a molecule in the gas phase, Eqn. 36 can be evaluated using density functional theory and wave function quantum chemistry, albeit at a significant computational cost. Databases of such calculations for small molecules are available to facilitate reuse [34]. For large condensed-phase systems,

however, it is common to apply classical force fields, where electrons are not taken into account explicitly. Instead, effective partial charges on atoms are used. The electronic degrees of freedom, charge polarization, is sometimes taken into account through induced point dipoles and higher electrostatic moments, or by using a core-shell model [9, 29, 59]. For additional background we refer to some excellent reviews [8, 20, 27].

The MEP can be used as a target in model development in the ACT, however we recommend against that for both fundamental and practical reasons [24].

4.3 Vibrational Frequencies

Vibrational frequencies are required to compute the IR spectra and thermochemistry of molecules. The normal modes of molecular vibrations can be obtained by eigenvalue decomposition of the Hessian matrix, whose elements are the second derivatives of the energy with respect to the atomic coordinates q.

$$H_{ij} = \frac{\partial^2 E}{\partial q_i \partial q_j} \tag{37}$$

where i and j run from 0 to $N - 1$, where N is the number of atoms in the molecule. If virtual sites v are used, for example, to model the σ-hole for halogen atoms, the energy, E, depends on the positions of both atoms and virtual sites; that is, $E = E(q_0, ..., q_{N-1}, v_0, ..., v_{M-1})$, where the positions of the M virtual sites, v, in the compound are a function of the atomic coordinates q.

The Hessian is computed numerically—the N atoms are moved independently in all three spatial dimensions, and the forces are computed. From these forces, the second derivative of the energy is then evaluated numerically. Note that the positions of the virtual sites are updated before each force calculation, which means that their influence on the Hessian is taken into account explicitly when computing H.

4.4 IR Spectra

Please note that the text in this section is taken `verbatim` from the paper by Henschel *et al.* (2020) [22] (with permission). For the calculation of a full IR

spectrum, in addition to the vibrational frequencies, the intensities and the line shapes are required.

In case of the quantum chemical calculations both the eigenfrequencies and the corresponding IR intensities are produced by default when a frequency calculation is requested in the Gaussian software [15], and were thus readily available from the Alexandria library. Details of the quantum chemical calculations from which the frequencies were obtained have been presented previously (refs. 16, 19).

For the force field calculations, the intensities I_n were derived from the transition dipole derivatives:

$$I_n = \sum_{k=1}^{3} \left(\frac{\partial p_k}{\partial Q_n} \right)^2 \tag{38}$$

where k iterates over cartesian dimensions, p is the dipole moment of the molecule, and Q_n the normal coordinate n. In order to take into account virtual sites v we note that

$$p_k = p_k(q_0, ..., q_{N-1}, v_0, ..., v_{M-1}) \tag{39}$$

and rewrite equation 38 as:

$$I_n = \sum_{k=1}^{3} \left(\frac{\partial p_k}{\partial q_s} \frac{\partial q_s}{\partial Q_n} \right)^2 \tag{40}$$

where s iterates over the N atomic coordinates. To make the calculation of intensities practical, the numerical derivative of the dipole moment with respect to the atomic coordinates $\frac{\partial p_k}{\partial q_s}$ is stored in a text file during the normal mode analysis and finally we note that the term $\frac{\partial q_s}{\partial Q_n}$ corresponds to one over component s of eigenvector n.

4.5 Thermochemistry

The canonical ensemble $Q(N, V, T)$ can be used to compute the molecular internal energy, the standard entropy, and the heat capacity at constant volume.

$$E = RT^2 \left(\frac{\partial \ln Q}{\partial T} \right)_{N,V} \tag{41}$$

$$C_v = 2RT\left(\frac{\partial \ln Q}{\partial T}\right)_{N,V} + RT^2\left(\frac{\partial^2 \ln Q}{\partial T^2}\right)_{N,V} \tag{42}$$

$$S^o = R\ln Q + RT\left(\frac{\partial \ln Q}{\partial T}\right)_{N,V} \tag{43}$$

where R is the ideal gas constant and T the absolute temperature. For an ideal gas, $Q(N, V, T)$ can be decomposed into partition function of different degrees of freedom: electronic (el), translational (tr), rotational (rot) and vibrational (vib) motions. Therefore, for a molecular ideal gas, $Q(N, V, T)$ can be expressed as:

$$Q(N, V, T) = \frac{(q_{\text{el}} q_{\text{tr}} q_{\text{rot}} q_{\text{vib}})^N}{N!} \tag{44}$$

The rigid rotator and the quantum harmonic oscillator can be used to approximate the contribution of the rotational and vibrational motions. The partition function of a rigid rotator is defined as

$$q_{\text{rot}} = \frac{T}{\sigma \Theta_{\text{rot}}} \tag{45}$$

where σ is the symmetry number, Θ_{rot} the rotational temperature defined as

$$\Theta_{\text{rot}} = \frac{h^2}{8\pi^2 I k_\beta} \tag{46}$$

where I is the moment of inertia, k_β the Boltzmann constant, and h the Planck constant. The partition function of a quantum harmonic oscillator is defined as

$$q_{\text{vib}} = \frac{e^{\frac{-\beta h\nu}{2}}}{1 - e^{-\beta h\nu}} \tag{47}$$

where ν is the vibrational frequency of the oscillator. If we define the vibrational temperature as $\Theta_{\text{vib}} = \frac{h\nu}{k_\beta}$, then we will have

$$q_{\text{vib}} = \frac{e^{\frac{\Theta_{\text{vib}}}{2T}}}{1 - e^{\frac{\Theta_{\text{vib}}}{T}}} \tag{48}$$

Applying Eqn.44 to Eqn.41 followed by the multiplication rule in logarithm yields:

$$E = E_{tr} + E_{rot} + E_{vib} \tag{49}$$

and similarly,

$$C_v = C_{tr} + C_{rot} + C_{vib} \tag{50}$$

$$S^o = S_{tr} + S_{rot} + S_{vib} \tag{51}$$

Thermochemical properties are computed automatically by the **alexandria nma** command. For more details, please see Van der Spoel *et al* [55].

4.6 Second Virial Coefficient

The second virial coefficient is the second term in the virial expansion that describes the deviation from the ideal gas law for real gases:

$$\frac{P}{RT\rho} = A + B_2(T)\rho + C_3(T)\rho^2 + ... \tag{52}$$

with P the pressure, R the gas constant, T the temperature and ρ the density.

$B_2(T)$ is a useful property gauging interactions in the gas phase because experimental values are available for close to 2000 compounds as a function of temperature. It is computed from an integral weighting the interaction between two molecules over three-dimensional space.

$$B_2^{cl}(T) = -\frac{1}{2}\int_0^\infty \left\langle e^{-\beta u_{12}(\mathbf{r})} - 1 \right\rangle d\mathbf{r} \tag{53}$$

where $u_{12}(r)$ is the interaction energy between two compounds (*See* 4.1), $\beta = 1/k_B T$ and the integral is over all space and relative orientations of the compounds. If we sample these adequately (including at close, repulsive, distance) we can simplify the integral to a one-dimensional one:

$$B_2^{cl}(T) = -2\pi\int_0^\infty r^2 \left\langle e^{-\beta u_{12}(r)} - 1 \right\rangle dr \tag{54}$$

The above equation is entirely classical and quantum corrections have to be added according to:

$$B_2^F(T) = \frac{\hbar^2}{24(k_B T)^3} \sum_{j=1}^{2} \left[\frac{\left\langle \mathbf{F}_j^2 \right\rangle}{m_j} \right] \tag{55}$$

for the force on the compounds and

$$B_2^\tau(T) = \frac{\hbar^2}{24(k_B T)^3} \sum_{j=1}^{2} \left[\sum_{\alpha=x,y,z} \frac{\left\langle \tau_{j,\alpha}^2 \right\rangle}{I_{j,\alpha}} \right] \tag{56}$$

for the torque on the compounds, where m_j is the mass of the molecules j and $\mathbf{F}^2$ is the averaged square force on one molecule given by

$$\left\langle \mathbf{F}^2 \right\rangle = k_B T \int_0^\infty \left\langle e^{-\beta u_{12}(\mathbf{r})} \left[\nabla u_{12}(\mathbf{r}) \right]^2 \right\rangle d\mathbf{r} \tag{57}$$

and where I is the moment of inertia of the molecule and τ^2 is the average square torque on one molecule defined by

$$\left\langle \tau_{j,\alpha}^2 \right\rangle = k_B T \int_0^\infty \left\langle e^{-\beta u_{12}(\mathbf{r})} \left[\nabla_\omega u_{12}(\mathbf{r}) \right]_{j,\alpha}^2 \right\rangle d\mathbf{r}. \tag{58}$$

The change in $B_2(T)$ as a function of temperature can be used to scrutinize the repulsive and attractive components of the potential energy. $B_2(T)$ is negative at low temperatures due to attraction forces, while it becomes positive at higher temperatures as repulsion forces start to dominate, and passes through a maximum and eventually decreases at very high temperatures where repulsion force are fully dominant [1].

Code to compute the second virial coefficient is available in the **alexandria b2** command.

5 Force Field Training Algorithms

Within the **alexandria train_ff** module of the Alexandria Chemistry Toolkit you can choose among three algorithms to optimize force field parameters:

- Markov Chain Monte Carlo (-optimizer MCMC)
- Genetic Algorithm (-optimizer GA)
- Hybrid GA/MCMC (-optimizer HYBRID)

Let us see what they do behind the scenes and how to control them.

5.1 MCMC

Assume we want to set the values for five parameters (nParam = 5) by performing ten MCMC iterations (-maxiter 10). Then, the code "reads"

```
for (int i = 0; i < 10; i++)
  for (int j = 0; j < 5; j++)
    stepMCMC();
```

That is, we do 10 $\times$ 5 = 50 MCMC steps. What is a MCMC step though? In essence,

1. prevDev: previous deviation from data
2. Choose a parameter and alter it
3. newDev: new deviation from data
4. If newDev $<$ prevDev, then we accept the parameter change and continue into the next step. Else, apply Metropolis Criterion. That is, with some probability we accept the change. Otherwise, we restore the parameter to its previous value and proceed into the next step.

Now, let us add some more detail.

First, we already have the deviation from data of the previous step in ‘prevDev‘. Second, we arbitrarily choose a parameter out of the ‘param‘ vector, say ‘param[i]‘, which is bounded by its maximum ‘pmax[i]‘ and its minimum ‘pmin[i]‘, yielding the range ‘prange[i] = pmax[i] - pmin[i]‘. Here is where the ‘-step [0, 1]‘ flag comes into play by specifying a fraction.
Third, we draw a value of ‘delta‘, which is uniformly distributed in ‘[- step *

prange[i], + step * prange[i]]‘ and add it to the parameter value ‘param[i] += delta‘. If the parameter has gone beyond its maximum, we set its value to the maximum. Same goes for the minimum.
 Fourth, we compute the deviation ‘newDev‘ from data of the modified parameter vector. If ‘newDev < prevDev‘, we accept the change and finish the MCMC step. Otherwise, we apply the Metropolis Criterion and finish the step.

5.1.1 Metropolis Criterion

If Mathematics is your thing and you *really* want to know the nitty-gritty stuff, you have thorough explanations of this method on Wikipedia and Shuyi Qin’s Master thesis. Here, we shall say that the Metropolis Criterion allows us to take steps that do not get us closer to a minimum, giving the opportunity to explore the parameter space and avoid local minima. How exactly?

The probability of accepting a "bad" parameter change is controlled by the -temp T flag and follows the equation

$$prob = exp(-deltaDev/T) \tag{59}$$

where deltaDev = newDev - prevDev. Note that the unit of temperature is the same as that of the deviation.

Given deltaDev, a higher temperature gives a higher probability of acceptance, as ‘-deltaDev/T‘ tends to ‘0‘ and ‘exp(0) = 1‘. On the other hand, a lower ‘T‘ gives less chances of acceptance, as ‘-deltaDev/T‘ tends to $-\infty$ and $exp(-\infty)$ = 0.

ACT gives us the possibility to lower the temperature (simulated annealing) during the MCMC optimization with the -anneal [0, 1] flag. If we use -anneal 0.5, the temperature will be -temp T until 50% of the iterations have been completed. Then, it will be linearly decreased until it reaches ‘0‘ on the last iteration.

There are two remarks to be made here:

- The temperature is kept constant during the ‘nParam‘ MCMC steps that take place for a given iteration.
- Since division by ‘0‘ is not defined, we set ‘T = 1e-6‘ on the last iteration.

Fig. 2 shows a schematic example of the temperature over time when we use actflagmaxiter 10 -temp 5 -anneal 0.5.

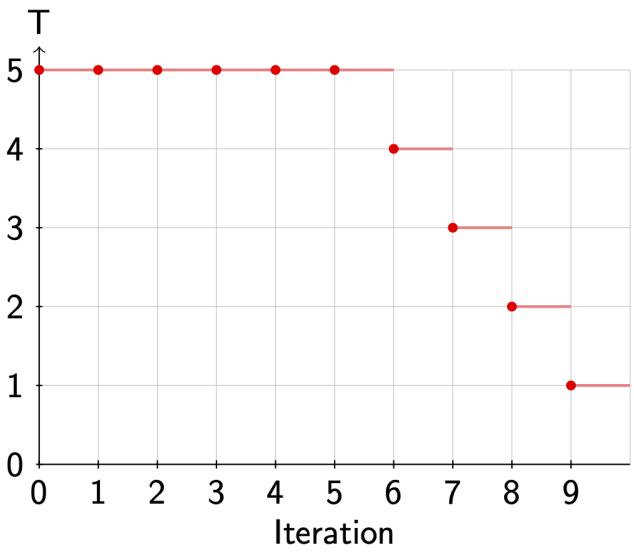

Figure 2: Annealing during a MCMC run.

5.1.2 Multiple MCMC runs

We can optimize several candidate solutions in parallel using the -pop_size flag. If -random_init is used (default), each candidate solution will be initialized arbitrarily and respecting parameter ranges. If -norandom_init is employed, the candidate solution(s) will be initialized as specified by the force field file(s) provided by the user.

5.2 GA

Quoting Wikipedia,

> In computer science and operations research, a genetic algorithm is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection.

In this section, we describe our implementation of a genetic algorithm for force field parameterization. If this is the first time you hear about genetic algorithms and want to acquaint yourself, we recommend you to read Chapter 2 of Steven F. van Dijk's PhD thesis and/or head over YouTube.

```
class Genome
{
    double params[];
    double deviation;
    double probability;
};

// Returns a sample of a random variable that is
// uniformly distributed in [0, 1]
double random();
```

Listing 1: Class definition.

Our implementation is based upon a class definition and an external function for computing random numbers (Listing 1) then the genome evolution boils down to the code in Listing 2

popSize and nElites are assumed to be even.

Let us explore the different stages of the process.

5.2.1 Initialization

This stage fills the ‘params‘ field in the ‘Genome‘ class, generating popSize (-pop_size) genomes.

If we are using -norandom_init, the genomes will be initialized as specified by the force field file(s) provided. If we are employing -random_init, each genome will be initialized by setting a random value for each parameter, uniformly distributed over the allowed range.

5.2.2 Deviation from data

Here we fill the ‘deviation‘ field in for each ‘Genome‘ in the population by computing the deviation from the dataset.

5.2.3 Sorting

Sorting is not a mandatory step but may be required depending on the GA components selected by the user. We sort the population in ascending order of ‘deviation‘. Whether we sort or not is controlled by the -sort flag.

```
void evolve(int popSize, int nElites, int prCross, int prMut)
{
    Genome oldPop[popSize] = initialize(popSize); // Initialize
    Genome newPop[];
    computeDeviation(oldPop); // Deviation from data
    int generation = 0;
    do
    {
        sort(oldPop); // Sorting
        if (penalize(oldPop, generation)) // Penalty?
        {
            // Recompute deviation and sort again
            computeDeviation(oldPop);
            sort(oldPop);
        }
        generation++;
        computeProbabilities(oldPop); // Selection probabilities
        // Elitism
        for (int i = 0; i < nElites; i++)
            newPop.add(oldPop[i]);
        // Rest of population
        for (int i = nElites; i < popSize; i += 2)
        {
            Genome parent1, parent2 = select(oldPop); // Selection
            Genome child1, child2;
            if (random() <= prCross) // Crossover?
                child1, child2 = crossover(parent1, parent2);
            else
                child1, child2 = parent1, parent2;
            // Mutation
            for (Genome child : {child1, child2})
            {
                mutate(child, prMut);
                newPop.add(child); // Add to new population
            }
        }
        computeDeviation(newPop); // Deviation from data
        oldPop = newPop; // Swap populations
        newPop.clear();  // Erase all genomes in the population
    }
    // Termination
    while (!terminate(oldPop, generation));
}
```


Listing 2: Schematic of the evolution algorithm

5.2.4 Penalties

At this stage, we may alter the population if certain conditions are met, with the main goal of preventing premature convergence and enforcing solution diversity.

> Covering such a small portion of the space you are. Broaden your search, you should. - Yoda

To that end, we have a function ‘penalize()‘ which returns ‘true‘ if the population was penalized and ‘false‘ otherwise.

For now, there are two components in this function:

1. **Volume.** This option enables -sort. If the volume spanned by the population divided by the total volume of the parameter space is smaller than -vfp_vol_frac_limit [0, 1]|, the *worst* fraction -vfp_pop_frac [0, 1] of genomes in the population will be randomly reinitialized. If -log_volume is used, volumes will be computed in logarithmic scale. *But wait, then the volume could be negative! Yes, we have to fix that!*

   > Death comes equally to us all, and makes us all equal when it comes. - John Donne

2. **Catastrophe.** Each -cp_gen_interval generations, a fraction -cp_pop_frac [0, 1] of the genomes in the population will be randomly reinitialized. Genomes to reinitialize are arbitrarily chosen.

5.2.5 Selection probabilities

We provide three options for computing selection probabilities:

1. Rank (-prob_computer RANK)
2. Fitness (-prob_computer FITNESS)
3. Boltzmann (-prob_computer BOLTZMANN)

The sum of the selection probabilities of the genomes in the population, is 1.

5.2.6 Rank

This option enables -sort. The selection probability depends exclusively on the index (rank) of genome in the population (Listing 3).

```
for (int i = 0; i < popSize; i++)
    oldPop[i].probability = (popSize - i) / (popSize * (popSize + 1) / 2);
```

Listing 3: Calculation of the probability from the order of probabilities.

That is, the lower the index, the higher the probability of being selected. The independence of the 'deviation' avoids the possible phenomena of a genome with a very high selection probability dominating the population.

5.2.7 Fitness

The selection probability is inversely proportional to the 'deviation' (Listing 4).

```
double total = 0;
double inverses[popSize];
double epsilon = 1e-4;
for (int i = 0; i < popSize; i++)
    inverses[i] = 1 / (epsilon + oldPop[i].deviation);
    total += inverses[i];
for (int i = 0; i < popSize; i++)
    oldPop[i].probability = inverses[i] / total;
```

Listing 4: Calculation of the probability from the deviations from data.

5.2.8 Boltzmann

The 'temperature' parameter is specified by the -boltz_temp flag and controls the smoothing of the selection probabilities (Listing 5). A higher value will avoid polarization in the probability values and vice versa. The -boltz_anneal flag allows us to decrease the temperature over time and has the same logic as -anneal, except that it targets the Boltzmann selection temperature and operates based on the maximum amount of generations.

5.2.9 Elitism

In order to avoid losing the best candidate solutions found so far, the GA will move the top nElites -n_elites genomes, *unchanged*, into the new population. That means, the genome will not undergo crossover nor mutation.

When -n_elites $>$ 0, -sort will be enabled.

```
double total = 0;
double exponentials[popSize];
double epsilon = 1e-4;
for (int i = 0; i < popSize; i++)
    exponentials[i] = exp(1 / (epsilon + oldPop[i].deviation) / temperature);
    total += exponentials[i];
for (int i = 0; i < popSize; i++)
    oldPop[i].probability = exponentials[i] / total;
```

Listing 5: Use of Boltzmann-weighting when calculating the probability

5.2.10 Selection

Once the selection probabilities are computed, the population becomes a probability density function from which we can sample genomes based on their ‘probability‘.

As of now, only a vanilla selector is available. It only looks at the probability and can select the same genome to be ‘parent1‘ and ‘parent2‘.

5.2.11 Crossover

With certain probability ‘prCross‘, controlled by the -pr_cross flag, two parents will combine their parameters to form two children. Right now, only an *N*-point crossover is available, where *N* is defined by the -n_crossovers flag.

If ‘N = 1‘, we arbitrarily select on crossover point (‘v‘) and

```
    v                           v
|X|X|X|X|X|X|X|X|  --> |X|X|Y|Y|Y|Y|Y|Y|
|Y|Y|Y|Y|Y|Y|Y|Y|  --> |Y|Y|X|X|X|X|X|X|
```

If ‘N = 2‘, we arbitrarily select two crossover points (‘v‘) and

```
  v     v                     v     v
|X|X|X|X|X|X|X|X|  --> |X|Y|Y|Y|X|X|X|X|
|Y|Y|Y|Y|Y|Y|Y|Y|  --> |Y|X|X|X|Y|Y|Y|Y|
```

If ‘N = 3‘, we arbitrarily select three crossover points (‘v‘) and

```
      v     v     v             v     v     v
|X|X|X|X|X|X|X|X|  --> |X|X|Y|Y|Y|X|X|X|Y|
|Y|Y|Y|Y|Y|Y|Y|Y|  --> |Y|Y|X|X|X|Y|Y|Y|X|
```

If ‘N = 4‘... you get the idea (hopefully).

5.2.12 Mutation

Given the mutation probability ‘prMut‘, controlled by -pr_mut, we iterate through the parameters. If the probability is met, we alter the parameter, otherwise, we leave it unchanged.

```
for (int i = 0; i < nParams; i++)
    if (random() <= prMut)
        changeParam(params, i);
```

The parameter is changed in the same way as in MCMC, by a fraction of its allowed range and not allowing values outside of it. The fraction in this case is controlled by the ‘-percentage‘ flag, which has the same meaning as ‘-step‘, but applies to GA instead of MCMC.

5.2.13 Termination

This stage decides whether the GA evolution should continue or halt. We allow the user to tweak the termination criteria with several flags:

- -max_generations: evolution will halt after so many generations.
- -max_test_generations (disabled by default): evolution will halt if in the last -max_test_generations the best ‘deviation‘ of the test set found so far has not improved.

5.3 HYBRID

Even though this optimizer has a very fancy name, it is just a GA with MCMC as its mutator engine. When MCMC acts as a mutator, it will always alter the genomes independently of -pr_mut. Also, it is important to note that the simulated annealing by default is applied independently in each MCMC run. For instance, in case we would use -max_generations 2 -maxiter 10 -temp 5 -anneal 0.5, Fig. 3 shows the temperature during the MCMC part.

However, when using the flag -anneal_globally, the starting temperature of the annealing will be decreased in steps at the beginning of each generation.

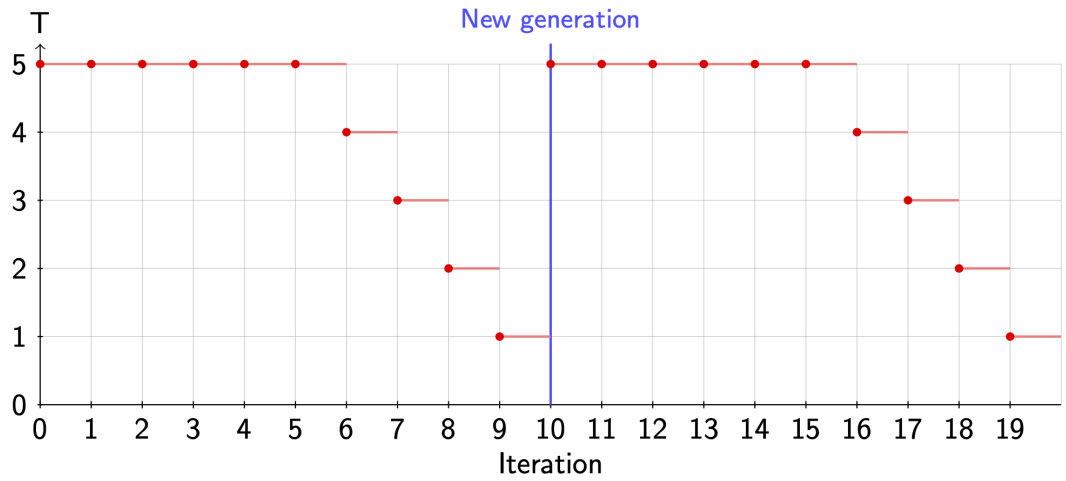

Figure 3: Annealing in the hybrid algorithm

6 Training Data

6.1 Using existing data

A recent review from the ACT developers [34] discusses the different available quantum chemistry data sets. Some of these can be used in the ACT, for instance the coupled-cluster dimer data set due to Donchev *et al* [10], the Non-covalent interaction atlas from the Czech group led by Řezáč [51] and some more. The SAPT dataset on protein side-chain analogs and backbone analogs by Burns [4] can used in the ACT as well.

In principle, the ANI-1 dataset [47] can be used to provide off-equilibrium energies of small compounds, but in the first version it was limited to compound containing the elements C, H, N, O only.

6.2 Generating SAPT data

A recent review described the available data sets available for machine learning or force field training [34]. There is however a lack of certain data, or data sets are incomplete. For this reason there is a git repository SaptACT that provides script to run SAPT calculations using the Psi4 software [50] including tools to convert the output to ACT compatible inputs (see below). The steps needed to prepare data for training in ACT are as follows:

1. Clone the SaptACT repository using `git`
2. Prepare selection of compounds, e.g. water, methanol, ethanol, and make sure that monomer structures for these compounds are present in the `xyz/monomers` catalog.
3. Generate a dimer selection file using the `gen_dimers.py` script in the SaptACT repository

   ```
   ./gen_dimers.py -sel water methanol ethanol
           -o alcohol.dat
   ```

   which will generate a file containing

   ```
   water#water
   water#methanol
   water#ethanol
   ```

```
methanol#methanol
methanol#ethanol
ethanol#ethanol
```

where the # symbol separates the monomers. Note that the order on the command line determines the order in the selection file.

4. Perform SAPT calculations using Psi4 by generating randomly oriented dimers at a series of distances defined by scaling the shortest distance between any pair of atoms in the generated orientations. In this case, six distances varying from 0.9 to 1.4 times the sum of the van der Waals radii of the nearest atoms will be generated.

```
./run_calcs.py -dimers alchol.dat -ndist 6
        -mindist 0.9 -maxdist 1.4 -norient 10
```

Note, that in the above command 6×10 calculations are started for each dimer in `alcohol.dat`, that is 360 calculations in total. You will obviously need a compute cluster to do these calculations, in particular if you select a more accurate SAPT level of theory [42] than the default (sapt2+/aug-cc-pvdz).

6.3 Generating single molecule data

Single molecules off-equilibrium energies and forces are needed to parameterize the intramolecular potential functions. Scripts are available that will take a structure of a monomer, perform a 50 ps MD simulation in the gas phase at elevated temperature using the GAFF force field [67] and the GROMACS software [43]. Then, conformations are extracted from the simulation trajectory and these are subjected to quantum chemistry calculations using Psi4 [50].

6.4 Conversion to ACT molprop files

The ACT uses the eXtensible Markup Language to store both data for training and force field files. Once your SAPT calculations are finished you need to perform the following steps to generate the ACT input:

1. Convert the Psi4 outputs to compact and human-readable json files using another script in SaptACT

   ```
   ./generate_json
   ```

   which will traverse the output directories, find Psi4 outputs, and if there is not already a `results.json` file, will generate it. Please familiarize yourself with the content of these files.

2. When the json files have been generated, you are ready to convert them to a molprop file

   ```
   ./write_molprop.py -o molprop.xml
   ```

   please inspect the output file from this script to verify that your calculations are present.

Both these scripts have more flags that can be useful, please investigate those using the `-h` option.

7 Installation of the ACT

The Alexandria Chemistry Toolkit (ACT) relies on a number of libraries. Even though we tried to keep it to a minimum, some more or less standard libraries are needed. ACT should compile fine on any UNIX (including MacOs) or Linux machine. Most of the libraries can be installed using Anaconda or even Miniconda which has the advantage of running in user-space entirely, that is you do not super-user access to install it. High-performance computer centers typically provide these libraries using some kind of module system.

7.1 Quick Installation

To install ACT relatively easy, we recommend that you install miniconda on your computer, download the ACT conda environment file ACT.yml and create and activate a new conda environment.

```
conda env create -f ACT.yml
conda activate ACT
```

This should install the libraries mentioned below (note: it will take some time!). You still need a C++ compiler and a MPI library as mentioned below. If you are installing ACT in a cluster we recommend to use the cluster-provided compilers and in particular the MPI library since it may be tuned to make optimal use of the communication hardware. If you are not on a cluster use the following command:

```
conda install openmpi
```

before proceeding. With that in place, you can skip the next section.

7.2 Long Installation

A patched version of OpenBabel is needed that can be found here. The official OpenBabel can be found here. The install_act script will automatically fetch and build the correct version for you! If you have another version of OpenBabel installed it may lead to unpredicted and incorrect behavior. We strongly recommend to install just the ACT version of OpenBabel.

1. Some version of a library that supports the message passing interface (MPI) for parallel programming. A popular version is Open MPI.

FFTW (version 3 or higher) for Discrete Fourier Transform (DFT)

2. A C++ compiler supporting C++17 at least. On Linux a GNU c++ version newer than 11.0 is needed.
3. The cmake tools (at least version 3.13.0) are needed for compiling the code.
4. For linear algebra operations we use the Eigen library.
5. The LibXml2 is needed for processing the XML data files used by the ACT. The SWIG library is needed to make Python bindings for OpenBabel.

Optionally, the following libraries may be useful for developing.

1. The Class Library for Numbers is used in an optional part of the code and can be omitted.
2. The SQLite database engine is needed to process experimental data as well as quantum chemistry data from the Alexandria Library, available from DOI.
3. the doxygen package can be used for generating documentation. In that case you need the Graphviz package as well.
4. Python, version 3, and a number of Python libraries. Namely, NumPy, Matplotlib, and PubChemPy.

7.3 Installing ACT on your computer

The easiest way to get going is to fetch the install_act script to your working directory of choice. To do so, once you have clicked on the link, click on Raw, then right-click anywhere on the page, and select Save Page As...

Once you have downloaded the script, start by executing

```
./install_act -h
```

Let's say you have a four core machine and you want to install first time around, then this command should do the trick:

```
./install_act -clone_OB -clone_ACT -ob -ncores 4
```

In the best of worlds, the script will have created a directory under your home catalog, with the name tools. In order to start using the software, run the

following command:

```
source $HOME/tools/bin/ACTRC
```

or add it to your .bash_profile (or equivalent, for remote machines) or .bashrc (or equivalent, for local machines), and restart the shell or log in again.

The you can run the alexandria executable using

```
alexandria -h
```

or OpenBabel commands such as

```
obabel -H
```

To make sure you do have the correct commands in your path, please try the command

```
which alexandria obabel
```

which should give you something like

```
~/tools/bin/alexandria
~/tools/bin/obabel
```

7.4 Setting up the custom OpenBabel Python library

Note: this should already be done by

```
source $HOME/tools/bin/ACTRC
```

Since we have built and installed custom Python libraries (namely, bindings for ACT and for a patched version of OpenBabel), we must tell our Python interpreter where to find them. We can do this by appending the location of the bindings to the PYTHONPATH environment variable

```
export PYTHONPATH=${PYTHONPATH}:<path1>:<path2>
```

If such a variable does not exist in your computer, you must create it yourself. For example, if you are using a UNIX or Linux machine, you have installed ACT and OpenBabel under /tools, you are running Python 3.9, and PYTHONPATH does not exist in your system, you should add

```
export PYTHONPATH=$HOME/tools/lib/site-packages/act:
$HOME/tools/lib/python3.9/site-packages/openbabel
```

to your /.bash_profile, /.zshrc, or equivalent. Please verify that these paths are correct.

7.5 Testing the ACT

To start testing, you first want to familiarize yourself with the test set. If the ACT is in your home directory, you can

```
cd ACT/build_Release_DOUBLE
```

Then you can build the test set using

```
make tests
```

and run it using

```
make test
```

which should give the following output:

```
Running tests...
Test project /Users/spoel/GG/ACT/build_Release_DOUBLE
      Start  1: TestUtilsUnitTests
 1/17 Test  #1: TestUtilsUnitTests ..............  Passed   3.10 sec
      Start  2: WangBuckinghamTests
 2/17 Test  #2: WangBuckinghamTests .............  Passed   0.33 sec
      Start 16: AlexandriaTests
```

(more tests)

```
16/17 Test #16: AlexandriaTests .................  Passed   8.80 sec
      Start 17: SobolTests
17/17 Test #17: SobolTests ......................  Passed   0.16 sec

100% tests passed, 0 tests failed out of 17
```

You can also run an individual test, like this `bin/sobol-test` which should give this output:

```
[==========] Running 2 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 2 tests from SobolTest
[ RUN      ] SobolTest.Test08
[       OK ] SobolTest.Test08 (0 ms)
```

```
[ RUN      ] SobolTest.Test09
[       OK ] SobolTest.Test09 (0 ms)
[----------] 2 tests from SobolTest (0 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran. (0 ms total)
[  PASSED  ] 2 tests.
```

Note that these tests are run every time a change in the ACT source code is uploaded to github, to prevent errors in the code from being introduced.

8 Using the ACT

To use the ACT, you first need to install the ACT. Once you have finished that, please try the command

```
alexandria -h
```

and

```
alexandria help commands
```

the output of which is given in the table b

Note that one can get detailed help for the alexandria modules using the -h flag, e.g.:

```
alexandria gentop -h
```

8.1 Creating a new force field file from scratch

Start by copying the examples directory from the source catalog, and change directory to the new directory.

Now, let's see what **alexandria gen_ff** has to offer:

```
alexandria gen_ff -h
```

(to see the output check the help text).

As we see there are many options, but for Alexandria force fields most can remain default. Make a fresh directory and try for instance:

```
alexandria gen_ff -f $ACTDATA/atomtypes.csv -o myff.xml
```

which gives output similar to

```
Read 18 rows from /Users/spoel/tools/share/act/forcefield/ffnames.csv
chargetype  = Gaussian
vanderwaals = BHAM
Generated myff.xml
```

The script prints that it has read and processes a number of files from the ACT installation. These data files can be copied and modified by the user. The files should be relatively simple to understand. Installed files are in the share/act directory. Please note that the force field files use the eXtensible

Table 1: Commands available in the ACT

Command	Description
analyze	Analyze molecular- or force field properties from a database and generate publication quality tables in LaTeX.
b2	Compute second virial coefficient as a function of temperature.
edit_ff	Manipulate and compare force field files in various ways and test whether reading and writing works.
edit_mp	Utility to merge a number of molecular property files and a SQLite database. Can also test reading and writing the molecular property file. It can also check the molecular property file for missing hydrogens and for whether it is possible to generate topologies for all compounds. Finally it can generate charges for all compounds.
gen_ff	Generate a force field file from a user specification.
gentop	Generate a molecular topology and coordinates based on structure files or quantum chemistry output from Gaussian. Only inputs for OpenMM can be generated at this point in time.
geometry_ff	Deduce bond/angle/dihedral distributions from a set of strucures and add those to a force field file.
help	Print help information
merge_ff	Utility to merge a number of force field files and write a new file with average parameters. Can also write a LaTeX table.
min_complex	Generate inputs for an energy scan.
nma	Perform normal mode analysis and compute thermochemistry properties.
simulate	Perform a MD simulation and generate a trajectory.
train_ff	Train a force field to reproduce reference data.

Markup Language that provides a structured way of storing data. Do not edit manually if at all possible.

This force field file gives us something to start with, but it is not complete yet. First, we have to add the possible bond lengths, bond angles etc., and those come from the Alexandria Library (see below). For now, we will use the provided file in XML/alcohol.xml. Thus, we need to run:

```
alexandria geometry_ff -ff myff.xml -mp XML/alcohol.xml
   -sel SELECTIONS/alcohol.dat -o myff2.xml

     Welcome to the Alexandria Chemistry Toolkit
<snip>
There are 14 molecules in the selection file SELECTIONS/alcohol.dat.
There are 15 molprops. Will now sort them.
There were 0 double entries, leaving 15 after merging.
There were 15 total molecules before merging, 15 after.

Thanks for using the Alexandria Chemistry Toolkit.
```

This generates a ready-to-optimize force field file, myff2.xml.

8.2 Optimizing your first force field parameters

Head over to the TRAIN_FF catalog and run the example optimization using:

```
./run_alcohol.py
```

and the adventure has begun! Inspect the different .xvg output files using a graphing program (see Figs. 4 and 5 below), and the train_ff.log file using a text viewer.

Note that these graphs were made using the viewxvg script that is shipped with ACT and that is based on matplotlib and tkinter. Please check it out using

```
viewxvg -h
```

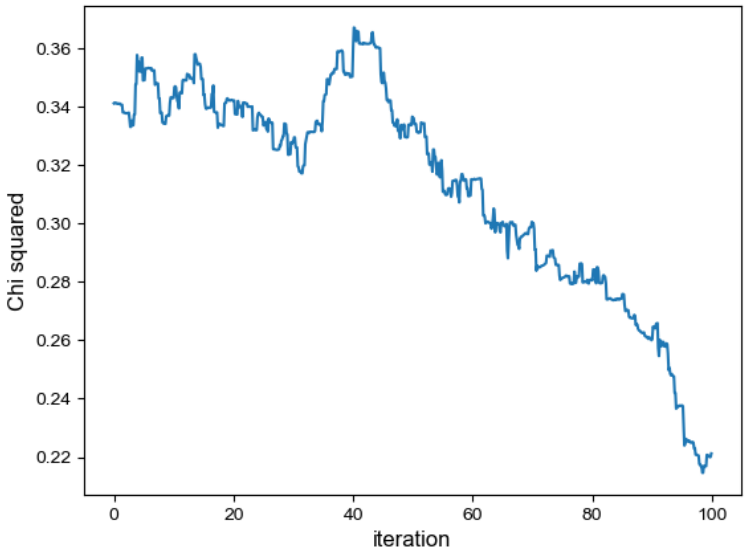

Figure 4: Sample convergence of the χ^2 fitness value.

8.3 Is my training 'hitting a wall'?

To make optimization efficient, it is good to limit the search space. In some cases we can use chemical intuition to estimate the search space. For instance, we know that polarizability should be a positive number and that it is highly element dependent. For hydrogen, a likely range is $0.1 < \alpha < 0.5$ Å^3. Such boundaries are implemented in the force field file like this:

```
<parameterlist identifier="h_s">
  <parameter type="alpha" unit="Angstrom3" value="0.46"
  uncertainty="0.008" minimum="0.1" maximum="0.5"
  ntrain="300" mutability="Bounded" nonnegative="yes"/>
</parameterlist>
```

this example is after an optimization where the optimal value ended up being 0.46, based on a dataset comprising 300 hydrogen atoms. Note the nonnegative="yes" which will prevent alexandria tools from making this parameter negative ever.

For many cases, we do not have an *a-priori* intuition of what the values should be and we make a guess. Then, after a series of optimizations, we can check whether the bounds are OK, using an alexandria tool:

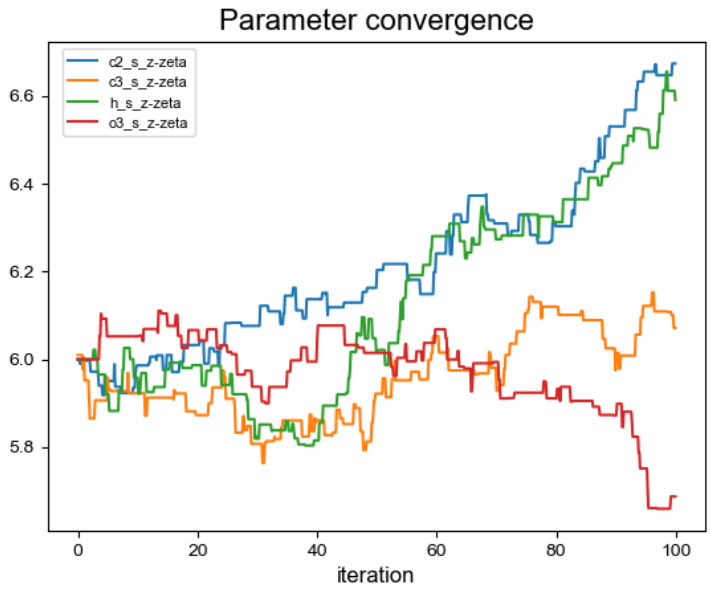

Figure 5: Convergence of the Gaussian distribution widths ζ.

```
alexandria edit_ff -ff 3/train_ff_13.xml -ana EEM
<snip>
POLARIZATION n2_s alpha at maximum 1.5
POLARIZATION n4_s alpha at minimum 0.6
BONDCORRECTIONS c2_~c3_z hardness at minimum -0.4
BONDCORRECTIONS c2_z:n2_z hardness at minimum 0
BONDCORRECTIONS c3_z~c3_z hardness at minimum -0.4
```

Some of the parameters are indeed hitting the wall. To adjust the parameters we use the same tool again

```
   alexandria edit_ff -ff 3/train_ff_13.xml -o new.xml -stretch -p alpha
<snip>
Thanks for using the Alexandria Chemistry Toolkit.
```

and then we check the resulting force field file in the same manner as before

```
    alexandria edit_ff -ff new.xml -ana EEM
<snip>
BONDCORRECTIONS c2_z~c3_z hardness at minimum -0.4
BONDCORRECTIONS c2_z:n2_z hardness at minimum 0
BONDCORRECTIONS c3_z~c3_z hardness at minimum -0.4
```

Now the polarizability minimum (for n4_s) and maximum (for n2_s) have been stretched. The same can now be done for hardness.

You can also do the reverse. When you are confident that the optimization is close to the final result, but want to randomize again without starting from scratch, you can set new minimum and maximum values for the parameters using, e.g.:

```
alexandria edit_ff -ff new.xml -o new.xml -p bondlength
-limits 0.98
```

please check the on-line help (**alexandria edit_ff -h**) for more information.

8.4 Running training using parallel processing

The optimization process is heavy in computer time. Therefore, it has been parallellized using the MPI library. Since this is a prerequisite for compiling the ACT, you likely have it installed if you got this far. To enable it, just add mpirun -np N before the alexandria command in the example script, where N is the number of cores you have available. Please consult with your clsuter manager if in doubt. As far as we have tested, the code is quite efficient down to 4-5 molecules per core. In the above example there are 10 compounds in the training set, such that it is not worthwhile to use more than 2-3 cores, but do experiment with the number of cores.

9 Simulations using alexandria force fields

9.1 The ACT-OpenMM interface

The OpenMM [12] software in its native form is controlled from Python scripts. This allows users great control over the simulations. OpenMM allows to specify user-defined energy functions, which we use to implement both Gaussian-distributed charges [18, 66] and many Van der Waals potentials [35] (see Section 3). To facilitate this, a special python code was implemented that makes it relatively easy for the user to run simulations and minimization. The first step is to convert an ACT force field file (`actff.xml`) to one compatible with OpenMM (`openmmff.xml`):

```
alexandria gentop -ff actff.xml -openmm openmmff.xml
-charges mp2.xml -db "water methanol"
```

with two additional arguments First, -charges mp2.xml indicates a molprop file (database) with monomeric (optimized) structures used for generating charges and, second, the -db "water methanol" instruct the code to produce an OpenMM topology for those compounds. If one or more of the compounds is not present in the database, a warning will be issued.

9.2 Molecular dynamics simulations

It is easy to perform simulations based on an Alexandria force field. For this, we rely on the OpenMM software that you need to install separately. Then you can use the ACT python interface to OpenMM, as in this simple script, that we will call run.py:

```
#!/usr/bin/env python3

from act_openmm import ActOpenMMSim

sim = ActOpenMMSim(pdbfile="file.pdb",
                   datfile="dimer.dat",
                   xmlfile="ff.xml",
                   txtfile="output2.txt",
                   verbose=True)
sim.run()
sim.log_to_xvg("energy.xvg", [ "Potential Energy (kJ/mole)"])
```

```
sim.log_to_xvg("temperature.xvg", [ "Temperature (K)" ])
sim.log_to_xvg("density.xvg", [ "Density (g/mL)" ])
```

Here, we first instantiate an ActOpenMMSim object, and pass it a structure file `file.pdb`, a simulation parameter file `dimer.dat` and a force field file `ff.xml`. We also instruct the code that output should be written to `output2.txt` and that we want a lot of information in that file. Then we can run it, that's all! The last three lines are for convenience of plotting the results. We can run this script using

```
python ./run.py
```

or submitted to a cluster, preferably with GPUs available.

9.3 Minimization using OpenMM

A minimization of the input structure can be done using this script:

```
    #!/usr/bin/env python3

from act_openmm import ActOpenMMSim

sim = ActOpenMMSim(pdbfile="file.pdb",
                   datfile="dimer.dat",
                   xmlfile="ff.xml",
                   txtfile="output2.txt",
                   verbose=True)
sim.setup()
sim.minimize()
sim.write_coordinates("final.pdb")
```

which is run here using the same flags as the simulation. Note that details about simulations and minimization can be specified in the "dimer.dat" file.

References

[1] I. Amdur and E. A. Mason. Properties of gases at very high temperatures. *Phys. Fluids*, 1:370–383, 09 1958. ISSN 0031-9171. doi:10.1063/1.1724353. URL https://doi.org/10.1063/1.1724353.

[2] Brent H Besler, Kenneth M Merz Jr., and Peter A Kollman. Atomic charges derived from semiempirical methods. *J. Comput. Chem.*, 11:431–439, 1990. doi:10.1002/jcc.540110404.

[3] R A Buckingham. The Classical Equation of State of Gaseous Helium, Neon and Argon. *Proc. R. Soc. London Ser. A*, 168:264–283, 1938.

[4] Lori A. Burns, John C. Faver, Zheng Zheng, Michael S. Marshall, Daniel G. A. Smith, Kenno Vanommeslaeghe, Alexander D. MacKerell, Kenneth M. Merz, and C. David Sherrill. The BioFragment Database (BFDb): An open-data platform for computational chemistry analysis of noncovalent interactions. *J. Chem. Phys.*, 147(16):161727, October 2017. ISSN 0021-9606. doi:10.1063/1.5001028.

[5] C Caleman, P J van Maaren, M Hong, J S Hub, L T Costa, and D van der Spoel. Force field benchmark of organic liquids: Density, enthalpy of vaporization, heat capacities, surface tension, compressibility, expansion coefficient and dielectric constant. *J. Chem. Theory Comput.*, 8:61–74, 2012. doi:10.1021/ct200731v.

[6] Grzegorz Chałasiński and Małgorzata M Szcześniak. State of the art and challenges of the ab initio theory of intermolecular interactions. *Chem. Rev.*, 100(11):4227–4252, 2000. doi:10.1021/cr990048z.

[7] T Darden, D York, and L Pedersen. Particle mesh Ewald: An N-log(N) method for Ewald sums in large systems. *J. Chem. Phys.*, 98:10089–10092, 1993.

[8] Pnina Dauber-Osguthorpe and A T Hagler. Biomolecular force fields: where have we been, where are we now, where do we need to go and how do we get there? *J. Comput. Aid. Mol. Des.*, 33(2):133–203, 2019. ISSN 1573-4951. doi:10.1007/s10822-018-0111-4.

[9] B G Dick and A W Overhauser. Theory of the dielectric constants of alkali halide crystals. *Phys. Rev.*, 112:90–103, 1958.

[10] Alexander G. Donchev, Andrew G. Taube, Elizabeth Decolvenaere, Cory

Hargus, Robert T. McGibbon, Ka-Hei Law, Brent A. Gregersen, Je-Luen Li, Kim Palmo, Karthik Siva, Michael Bergdorf, John L. Klepeis, and David E. Shaw. Quantum chemical benchmark databases of gold-standard dimer interaction energies. *Sci. Data*, 8(1):55, 2021. ISSN 2052-4463. doi:10.1038/s41597-021-00833-x.

[11] Peter Eastman and Vijay S Pande. Efficient Nonbonded Interactions for Molecular Dynamics on a Graphics Processing Unit. *J. Comput. Chem.*, 31:1268–1272, 2010. ISSN 0192-8651. doi:10.1002/jcc.21413.

[12] Peter Eastman, Raimondas Galvelis, Raúl P. Peláez, Charlles R. A. Abreu, Stephen E. Farr, Emilio Gallicchio, Anton Gorenko, Michael M. Henry, Frank Hu, Jing Huang, Andreas Krämer, Julien Michel, Joshua A. Mitchell, Vijay S. Pande, João PGLM Rodrigues, Jaime Rodriguez-Guerra, Andrew C. Simmonett, Sukrit Singh, Jason Swails, Philip Turner, Yuanqing Wang, Ivy Zhang, John D. Chodera, Gianni De Fabritiis, and Thomas E. Markland. Openmm 8: Molecular dynamics simulation with machine learning potentials. *J. Phys. Chem. B.*, 128:109–116, 2024. doi:10.1021/acs.jpcb.3c06662.

[13] U Essmann, L Perera, M L Berkowitz, T Darden, H Lee, and L G Pedersen. A smooth particle mesh Ewald method. *J. Chem. Phys.*, 103:8577–8592, 1995.

[14] Nina M. Fischer, Paul J. van Maaren, Jonas C. Ditz, Ahmet Yildirim, and David van der Spoel. Properties of liquids in molecular dynamics simulations with explicit long-range Lennard-Jones interactions. *J. Chem. Theory Comput.*, 11:2938–2944, 2015. doi:10.1021/acs.jctc.5b00190.

[15] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P.

Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox. Gaussian 16 Revision A.03, 2016. Gaussian Inc. Wallingford CT.

[16] Mohammad Mehdi Ghahremanpour, Paul J. van Maaren, Jonas Ditz, Roland Lindh, and David van der Spoel. Large-scale calculations of gas phase thermochemistry: enthalpy of formation, standard entropy and heat capacity. *J. Chem. Phys.*, 145:114305, 2016. doi:10.1063/1.4962627.

[17] Mohammad Mehdi Ghahremanpour, Paul J. van Maaren, and David van der Spoel. Alexandria library [data set]. Zenodo, 2017. https://doi.org/10.5281/zenodo.1004711.

[18] Mohammad Mehdi Ghahremanpour, Paul J. van Maaren, Carl Caleman, Geoffrey R. Hutchison, and David van der Spoel. Polarizable drude model with s-type gaussian or slater charge density for general molecular mechanics force fields. *J. Chem. Theory Comput.*, 14:5553–5566, 2018. doi:10.1021/acs.jctc.8b00430.

[19] Mohammad Mehdi Ghahremanpour, Paul J. van Maaren, and David van der Spoel. The Alexandria library: a quantum chemical database of molecular properties for force field development. *Sci. Data*, 5:180062, 2018. doi:10.1038/sdata.2018.62.

[20] A.T. Hagler. Force field development phase II: Relaxation of physics-based criteria... or inclusion of more rigorous physics into the representation of molecular energetics. *J. Comput. Aided Mol. Des.*, 33:205–264, 2019. doi:10.1007/s10822-018-0134-x.

[21] Thomas A Halgren. The representation of van der Waals (vdW) interactions in molecular mechanics force fields: potential form, combination rules, and vdW parameters. *J. Amer. Chem. Soc.*, 114:7827–7843, 1992. doi:10.1021/ja00046a032.

[22] Henning Henschel, Alfred T. Andersson, Willem Jespers, Mohammad Mehdi Ghahremanpour, and David van der Spoel. Theoretical infrared spectra: Quantitative similarity measures and force fields. *J. Chem. Theory Comput.*, 16(5):3307–3315, 2020. doi:10.1021/acs.jctc.0c00126.

[23] A. Najla Hosseini and David van der Spoel. Simulations of amyloid-

forming peptides in the crystal state. *Prot. J.*, 42:192–204, 2023. doi:10.1007/s10930-023-10119-3.

[24] A. Najla Hosseini, Kristian Kříž, and David van der Spoel. Accurate electrostatics for physics-based force fields though machine learning. In preparation, 2025.

[25] W Hua. 4-parameter exactly solvable potential for diatomic-molecules. *Phys. Rev. A*, 42(5):2524–2529, SEP 1 1990. ISSN 1050-2947. doi:10.1103/PhysRevA.42.2524.

[26] Wei Hua. Four-parameter potential and its bound-state matrix elements. *J. Phys. B: Atom., Molec. Opt. Phys.*, 23:2521, aug 1990. doi:10.1088/0953-4075/23/15/019.

[27] Zhifeng Jing, Chengwen Liu, Sara Y. Cheng, Rui Qi, Brandon D. Walker, Jean-Philip Piquemal, and Pengyu Ren. Polarizable force fields for biomolecular simulations: Recent advances and applications. *Ann. Rev. Biophys.*, 48(1):371–394, 2019. doi:10.1146/annurev-biophys-070317-033349.

[28] J E Jones. On the determination of molecular fields. -II. From the equation of state of a gas. *Proc. Royal Soc. Lond. Ser. A*, 106:463–477, 1924. doi:10.1098/rspa.1924.0082.

[29] P C Jordan, P J van Maaren, J Mavri, D van der Spoel, and H J C Berendsen. Towards Phase Transferable Potential Functions: Methodology and Application to Nitrogen. *J. Chem. Phys.*, 103:2272–2285, 1995. doi:10.1063/1.469703.

[30] W L Jorgensen, J Chandrasekhar, J D Madura, R W Impey, and M L Klein. Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.*, 79:926–935, 1983. doi:10.1063/1.445869.

[31] Brian J. Kirby and Pavel Jungwirth. Charge scaling manifesto: A way of reconciling the inherently macroscopic and microscopic natures of molecular simulations. *J. Phys. Chem. Lett.*, 10:7531–7536, 2019. doi:10.1021/acs.jpclett.9b02652.

[32] Chang Lyoul Kong and Manoj R Chakrabarty. Combining rules for intermolecular potential parameters. iii. application to the exp 6 potential. *J. Phys. Chem.*, 77(22):2668–2670, 1973.

[33] Kristian Kříž and David van der Spoel. Quantification of anisotropy in exchange and dispersion interactions: A simple model for physics-based force fields. *J. Phys. Chem. Lett.*, 15:9974–9978, 2024. doi:10.1021/acs.jpclett.4c02034.

[34] Kristian Kříž, Lisa Schmidt, Alfred T. Andersson, Marie-Madeleine Walz, and David van der Spoel. An imbalance in the force: The need for standardised benchmarks for molecular simulation. *J. Chem. Inf. Model.*, 63:412–431, 2023. doi:10.1021/acs.jcim.2c01127.

[35] Kristian Kříž, Paul J. van Maaren, and David van der Spoel. Impact of combination rules, level of theory and potential function on the modelling of gas and condensed phase properties of noble gases. *J. Chem. Theory Comput.*, 20:2362–2376, 2024. doi:10.1021/acs.jctc.3c01257.

[36] G Lamoureux, A D MacKerell, and B Roux. A simple polarizable model of water based on classical Drude oscillators. *J. Chem. Phys.*, 119:5185–5197, 2003. doi:10.1063/1.1598191.

[37] Daniel S D Larsson and David van der Spoel. Screening for the location of RNA using the chloride ion distribution in simulations of virus capsids. *J. Chem. Theory Comput.*, 8:2474–2483, 2012. doi:10.1021/ct3002128.

[38] M W Mahoney and W L Jorgensen. A five-site model for liquid water and the reproduction of the density anomaly by rigid, nonpolarizable potential functions. *J. Chem. Phys.*, 112:8910–8922, 2000.

[39] Jesse G. McDaniel and J.R. Schmidt. Physically-motivated force fields from symmetry-adapted perturbation theory. *J. Phys. Chem. A.*, 117(10): 2053–2066, 2013. doi:10.1021/jp3108182.

[40] P M Morse. Diatomic molecules according to the wave mechanics. II. Vibrational levels. *Phys. Rev.*, 34:57–64, 1929. doi:10.1103/PhysRev.34.57.

[41] Razvan A. Nistor, Jeliazko G. Polihronov, Martin H. Müser, and Nicholas J. Mosey. A generalization of the charge equilibration method for nonmetallic materials. *J. Chem. Phys.*, 125(9):094108, 2006. doi:10.1063/1.2346671.

[42] Trent M. Parker, Lori A. Burns, Robert M. Parrish, Alden G. Ryno, and C. David Sherrill. Levels of symmetry adapted perturbation theory (sapt). I. efficiency and performance for interaction energies. *J. Chem. Phys.*, 140(9):094106, 2014. doi:10.1063/1.4867135.

[43] Sander Pronk, Szilárd Páll, Roland Schulz, Per Larsson, Pär Bjelkmar, Rossen Apostolov, Michael R Shirts, Jeremy C Smith, Peter M Kasson, David van der Spoel, Berk Hess, and Erik Lindahl. GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, 29(7):845–854, April 2013. ISSN 1367-4811. doi:10.1093/bioinformatics/btt055. URL http://www.ncbi.nlm.nih.gov/pubmed/23407358.

[44] A K Rappé and W A Goddard III. Charge Equillibration for Molecular Dynamics Simulations. *J. Phys. Chem.*, 95:3358–3363, 1991. doi:10.1021/j100161a070.

[45] J P Ryckaert and A Bellemans. Molecular Dynamics of Liquid n-Butane near its Boiling Point. *Chem. Phys. Lett.*, 30:123–125, 1975.

[46] Lisa Schmidt, David van der Spoel, and Marie-Madeleine Walz. Probing phase transitions in organic crystals using atomistic md simulations. *ACS Phys Chem Au*, 3:84–93, 2023. doi:10.1021/acsphyschemau.2c00045.

[47] Justin S. Smith, Olexandr Isayev, and Adrian E. Roitberg. ANI-1, a data set of 20 million calculated off-equilibrium conformations for organic molecules. *Sci. Data*, 4:170193, 2017.

[48] K T Tang and J P Toennies. An improved simple-model for the vanderwaals potential based on universal damping functions for the dispersion coefficients. *J. Chem. Phys.*, 80:3726–3741, 1984. doi:10.1063/1.447150.

[49] K. T. Tang and J. P. Toennies. The van der Waals potentials between all the rare gas atoms from He to Rn. *J. Chem. Phys.*, 118(11):4976–4983, 03 2003. ISSN 0021-9606. doi:10.1063/1.1543944. URL https://doi.org/10.1063/1.1543944.

[50] Justin M. Turney, Andrew C. Simmonett, Robert M. Parrish, Edward G. Hohenstein, Francesco A. Evangelista, Justin T. Fermann, Benjamin J. Mintz, Lori A. Burns, Jeremiah J. Wilke, Micah L. Abrams, Nicholas J. Russ, Matthew L. Leininger, Curtis L. Janssen, Edward T. Seidl, Wesley D. Allen, Henry F. Schaefer, Rollin A. King, Edward F. Valeev, C. David Sherrill, and T. Daniel Crawford. Psi4: an open-source ab initio electronic structure program. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, 2:556–565, 2011. doi:10.1002/wcms.93.

[51] Jan Řezáč. Non-covalent interactions atlas benchmark data sets: Hydrogen bonding. *J. Chem. Theory Comput.*, 16:2355–2368, 2020.

[52] D van der Spoel and P J van Maaren. The origin of layer structure artifacts in simulations of liquid water. *J. Chem. Theory Comput.*, 2:1–11, 2006. doi:10.1021/ct0502256.

[53] David van der Spoel. Systematic design of biomolecular force fields. *Curr. Opin. Struct. Biol.*, 67:18–24, 2021. doi:10.1016/j.sbi.2020.08.006.

[54] David van der Spoel, Erik Lindahl, Berk Hess, Gerrit Groenhof, Alan E Mark, and Herman J C Berendsen. GROMACS: Fast, Flexible and Free. *J. Comput. Chem.*, 26:1701–1718, 2005. doi:10.1002/jcc.20291.

[55] David van der Spoel, Mohammad Mehdi Ghahremanpour, and Justin Lemkul. Small molecule thermochemistry: A tool for empirical force field development. *J. Phys. Chem. A*, 122:8982–8988, 2018. doi:10.1021/acs.jpca.8b09867.

[56] David van der Spoel, Henning Henschel, Paul J. van Maaren, Mohammad M. Ghahremanpour, and Luciano T. Costa. A potential for molecular simulation of compounds with linear moieties. *J. Chem. Phys.*, 153(8): 084503, 2020. doi:10.1063/5.0015184.

[57] David van der Spoel, Jin Zhang, and Haiyang Zhang. Quantitative predictions from molecular simulations using explicit or implicit interactions. *Wiley Interdiscip. Rev.-Comput. Mol. Sci.*, 12:e1560, 2022. ISSN 1759-0876. doi:10.1002/wcms.1560.

[58] David van der Spoel, Paul J. van Maaren, and Mohammad Mehdi Ghahremanpour. Alexandria chemistry toolkit. https://github.com/AlexandriaChemistry/, 2025. Date accessed: 2025-02-12.

[59] P J van Maaren and D van der Spoel. Molecular dynamics simulations of water with a novel shell-model potential. *J. Phys. Chem. B.*, 105: 2618–2626, 2001. doi:10.1021/jp003843l.

[60] Paul J. van Maaren and David van der Spoel. Quantitative evaluation of anharmonic bond potentials for molecular simulations. *Digit. Discov.*, pages –, 2025. doi:10.1039/D4DD00344F.

[61] Toon Verstraelen, Veronique Van Speybroeck, and Michel Waroquier.

The electronegativity equalization method and the split charge equilibration applied to organic systems: Parametrization, validation, and comparison. *J. Chem. Phys.*, 131:044127, 2009. ISSN 0021-9606. doi:10.1063/1.3187034.

[62] Marie-Madeleine Walz and David van der Spoel. Molten alkali halides - temperature dependence of structure, dynamics and thermodynamics. *Phys. Chem. Chem. Phys.*, 21:8516–18524, 2019. doi:10.1039/C9CP03603B.

[63] Marie-Madeleine Walz and David van der Spoel. Systematically improved melting point prediction: a detailed physical simulation model is required. *Chem. Comm.*, 55:12044–12047, 2019. doi:10.1039/C9CC06177K.

[64] Marie-Madeleine Walz and David van der Spoel. Direct link between structure, dynamics and thermodynamics in molten salts. *J. Phys. Chem. C.*, 123:25596–25602, 2019. doi:10.1021/acs.jpcc.9b07756.

[65] Marie-Madeleine Walz and David van der Spoel. Microscopic origin of conductivity in molten salts unraveled by computer simulations. *Commun. Chem.*, 4(9):1–10, 2021. doi:10.1038/s42004-020-00446-2.

[66] Marie Madeleine Walz, Mohammad M. Ghahremanpour, Paul J. van Maaren, and David van der Spoel. Phase-transferable force field for alkali halides. *J. Chem. Theory Comput.*, 14:5933–5948, 2018. doi:10.1021/acs.jctc.8b00507.

[67] J Wang, R M Wolf, J W Caldwell, P A Kollman, and D A Case. Development and testing of a general AMBER force field. *J. Comput. Chem.*, 25:1157–1174, 2004. doi:10.1002/jcc.20035.

[68] Lee-Ping Wang, Jiahao Chen, and Troy Van Voorhis. Systematic Parametrization of Polarizable Force Fields from Quantum Chemistry Data. *J. Chem. Theory Comput.*, 9(1):452–460, 2013. doi:10.1021/ct300826t.

[69] Christian L. Wennberg, Teemu Murtola, Berk Hess, and Erik Lindahl. Lennard-Jones lattice summation in bilayer simulations has critical effects on surface tension and lipid properties. *J. Chem. Theory Comput.*, 9:3527–3537, 2013.

[70] Jasper C Werhahn, Evangelos Miliordos, and Sotiris S Xantheas. A new variation of the buckingham exponential-6 potential with a tunable,

singularity-free short-range repulsion and an adjustable long-range attraction. *Chem. Phys. Lett.*, 619:133–138, 2015.

[71] Chunhua Yin, Ziheng Cui, Yang Jiang, David van der Spoel, and Haiyang Zhang. Role of host-guest charge transfer in cyclodextrin complexation: A computational study. *J. Phys. Chem. C.*, 123:17745–17756, 2019. doi:10.1021/acs.jpcc.9b05399.

[72] Jin Zhang, Badamkhatan Tuguldur, and David van der Spoel. Force field benchmark II: Gibbs energy of solvation of organic molecules in organic liquids. *J. Chem. Inf. Model.*, 55:1192–1201, 2015. doi:10.1021/acs.jcim.5b00106.

[73] Jin Zhang, Badamkhatan Tuguldur, and David van der Spoel. Correction to force field benchmark II: Gibbs energy of solvation of organic molecules in organic liquids. *J. Chem. Inf. Model.*, 56:819–820, 2016. doi:10.1021/acs.jcim.6b00081.