Interatomic potentials

Numerical Simulation and Scientific Computing II

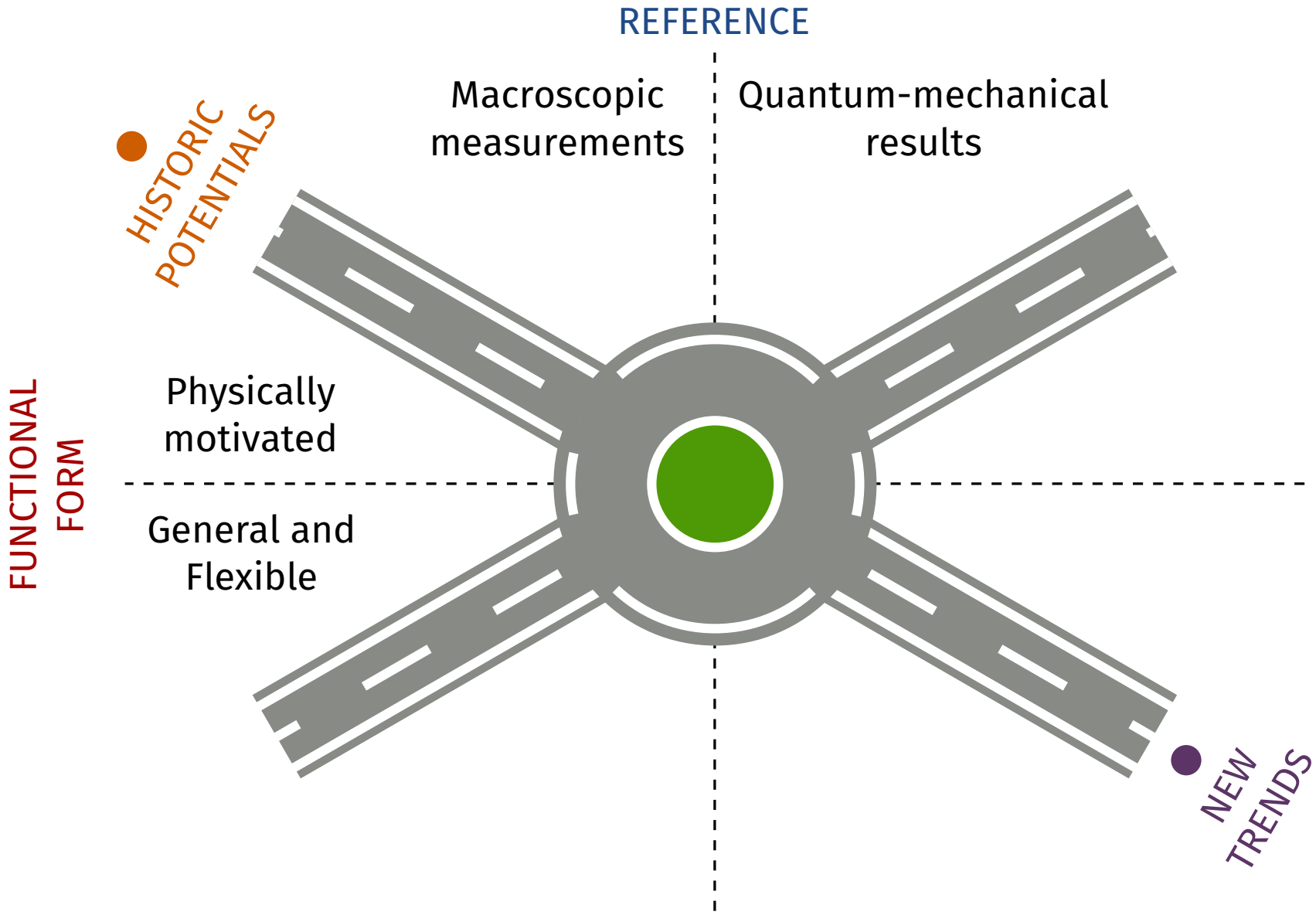
Jesús Carrete Montaña
jesus.carrete.montana@tuwien.ac.at

Lecture 6

TECHNISCHE UNIVERSITÄT WIEN

Introduction

Different approaches to E_{pot}



Physical constraints

Any reasonable potential must reflect the fundamental symmetries of mechanics:

- Translation: $E_{\text{pot}}(\{\boldsymbol{x}_I + \boldsymbol{A}\}) = E_{\text{pot}}(\{\boldsymbol{x}_I\}) \forall \boldsymbol{A} \in \mathbb{R}^3$
- Rotation: $E_{\text{pot}}(\{\mathcal{R}\boldsymbol{x}_I\}) = E_{\text{pot}}(\{\boldsymbol{x}_I\}) \forall \mathcal{R} \in \mathrm{SO}(3)$

...in the absence of external fields.

Not valid potentials:

- $E_{\text{pot}}(\boldsymbol{x}) = -\frac{K}{|\boldsymbol{x}|}$
- $E_{\text{pot}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = K(\boldsymbol{x}_1 + \boldsymbol{x}_2)$
- $E_{\text{pot}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = K(\boldsymbol{x}_1 \cdot \boldsymbol{x}_2)$
- $E_{\text{pot}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = K(y_1 - y_2)$

Furthermore:

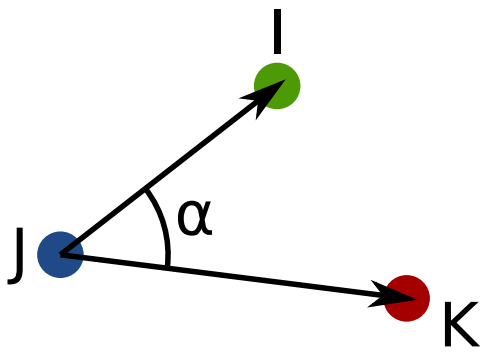
- No arbitrary labeling of identical particles
- No spurious singularities

n-body potentials

A classic, systematic approach

$$E_{\text{pot}}(\{\boldsymbol{x}_I\}) = V_0 + \sum_I V_1(\boldsymbol{x}_I) + \sum_{I,J\neq I} V_2(\boldsymbol{x}_I, \boldsymbol{x}_J) + \sum_{\substack{I,J\neq I \\ K\notin\{I,J\}}} V_3(\boldsymbol{x}_I, \boldsymbol{x}_J, \boldsymbol{x}_K) + \sum_{\substack{I,J\neq I \\ K\notin\{I,J\} \\ L\notin\{I,J,K\}}} V_4(\boldsymbol{x}_I, \boldsymbol{x}_J, \boldsymbol{x}_K, \boldsymbol{x}_L) + \ldots$$

- Rarely extended beyond V_4
- Usually based on explicit scalars like:



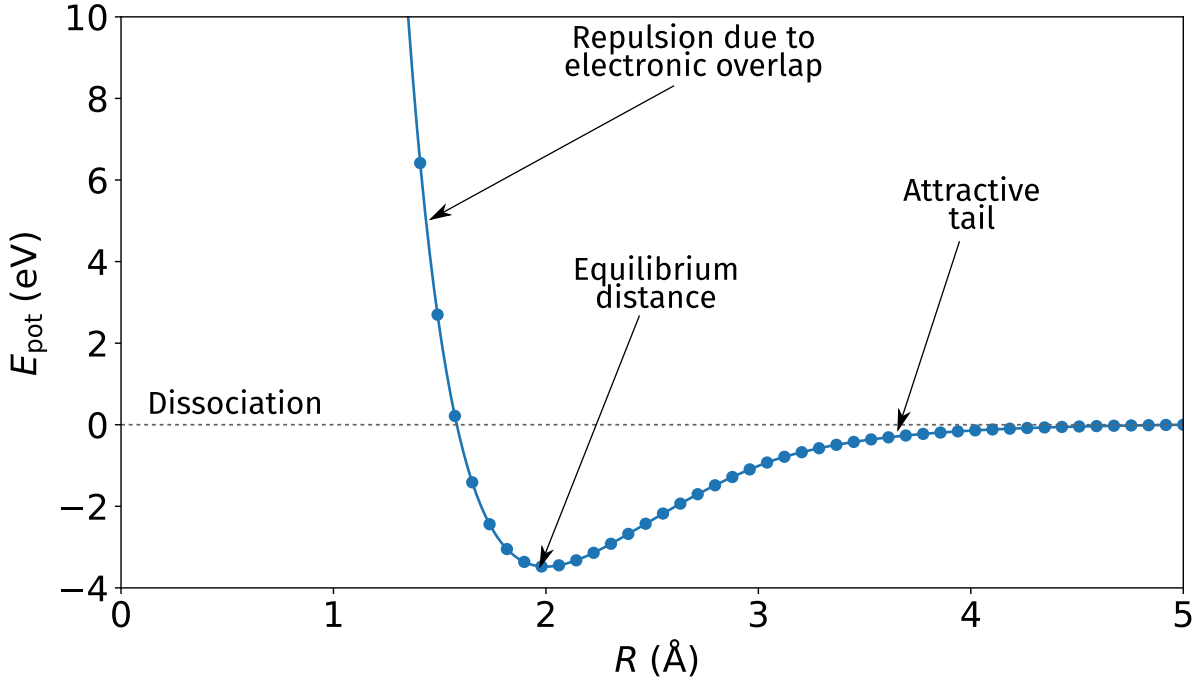
$$R_{IJ} = |\boldsymbol{x}_I - \boldsymbol{x}_J|$$

$$\cos\alpha_{IJK} = \frac{(\boldsymbol{x}_I - \boldsymbol{x}_J)\cdot(\boldsymbol{x}_K - \boldsymbol{x}_J)}{R_{IJ}R_{JK}}$$

Some two-body potentials

Traditional structure

Functional form: $V_2(R)$. Typical shape (DFT energies, Cl_2 dimer):



Some handy Lego blocks:

- Harmonic
- Coulomb
- Dipole-dipole
- Screened Coulomb
- Correlation between dipoles (London)

The Lennard-Jones potential (1924)

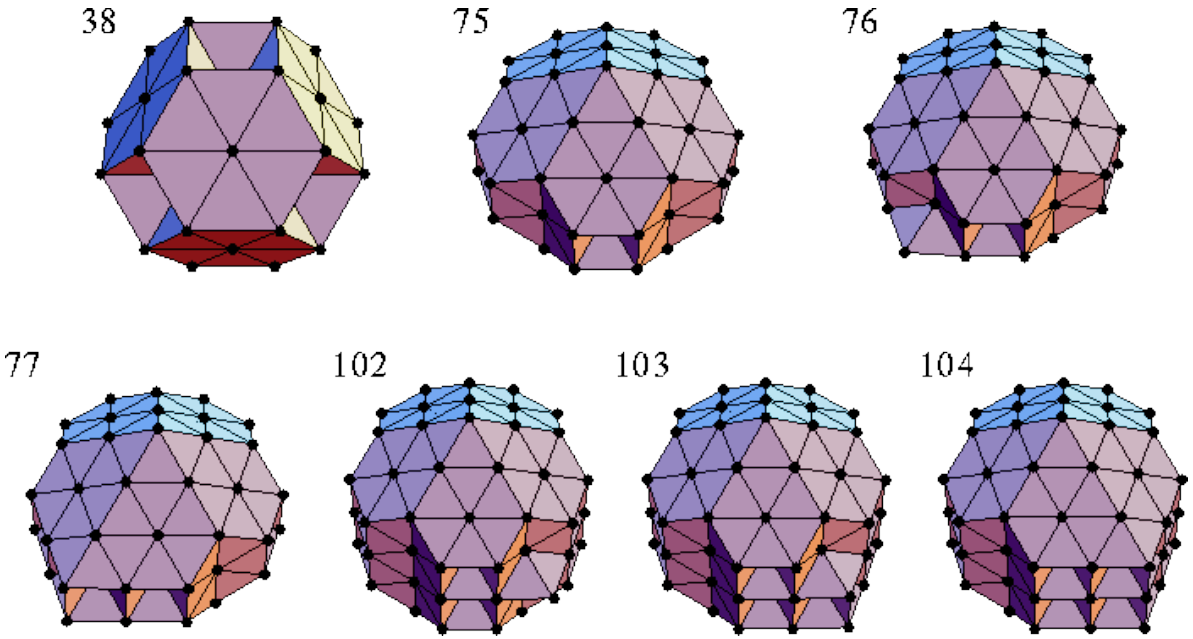
A classic that lives on

$$V_2(R) = 4\epsilon\left[\left(\frac{\sigma}{R}\right)^{12} - \left(\frac{\sigma}{R}\right)^{6}\right]$$

- Exchange interaction + dipole correlation
- Developed for noble gases
- Parameters: ϵ and $\sigma \Rightarrow$ attraction and repulsion interlinked

Rich behaviour from a simple model

- Bulk Lennard-Jones systems adopt close-packing structures
- Finding the ground states of LJ clusters took decades and intense algorithmic development!
- Cambridge energy landscape database: `http://www-wales.ch.cam.ac.uk/CCD.html`



The Buckingham potential (1938)

A much more expensive proposal (at the time)

$$V_2(R) = Ae^{-BR} - \frac{C}{R^6}$$

- Three parameters $\Rightarrow$ more flexible
- Often extended with a Coulomb term

The Morse potential (1929)

A very familiar expression

$$V_2(R) = D_e \left\{ \left[1 - e^{-\alpha(R-R_c)}\right]^2 - 1 \right\}$$

- Quite standard in chemistry (works for many diatomic systems)
- Suitable for analytical treatments of *e.g.* anharmonicity
- D_e: minimum potential energy ($\neq$ dissociation energy!)
- R_c: position of the minimum

The Morse/long-range potential (2009)

A very different story for modern times

$$V_2(R) = D_e \left\{ \left[1 - \frac{u(R)}{u(R_c)} e^{-\beta(R) y(r)} \right]^2 - 1 \right\}$$

where

$$y(R) = \frac{R^p - R_c^p}{R^p + R_c^p}, \text{ with } p \in \mathbb{Z}^+ - \{1\}$$

$$\beta(R) = y(R)\beta_\infty + [1 - y(R)] \sum_{i=0}^{N_{\text{short}} \text{ or } N_{\text{long}}} \beta_i y(R)^i$$

$$u(R) = \sum_i \frac{C_{m_i}}{R^{m_i}}, \text{ with } \{m_i\} \text{ dictated by symmetries}$$

- Standard in modern spectroscopic applications
- *Ab-initio*-like accuracy
- Directly fitted to spectroscopic data: J. Chem. Phys. 131 (2009) 204309

Combination rules for multicomponent systems

When there are several types of atoms $a \in \{1, 2 \dots c\}$:

$$E_{\text{pot}}(\{\boldsymbol{x}_I\}) = \sum_{a,b} \sum_{\substack{I \in a \\ J \in b}} V_2^{(ab)}(R_{IJ})$$

There are $\frac{c(c+1)}{2}$ functions to parameterize $\Rightarrow$ very inconvenient.

Combination rules based on "physical interpretations" of each parameter. For Lennard-Jones:

Lorentz-Berthelot

$$\sigma^{(ab)} = \frac{\sigma^{(a)} + \sigma^{(b)}}{2}$$

$$\epsilon^{(ab)} = \sqrt{\epsilon^{(a)}\epsilon^{(b)}}$$

Waldman-Hagler

$$\sigma^{(ab)} = \left\{ \frac{\left[\left(\sigma^{(a)}\right)^6 + \left(\sigma^{(b)}\right)^6\right]}{2} \right\}^{1/6}$$

$$\epsilon^{(ab)} = \sqrt{\epsilon^{(a)}\epsilon^{(b)}}$$

GROMACS

$$\sigma^{(ab)} = \sqrt{\sigma^{(a)}\sigma^{(b)}}$$

$$\epsilon^{(ab)} = \sqrt{\epsilon^{(a)}\epsilon^{(b)}}$$

Kong

$$\epsilon^{(ab)}\left(\sigma^{(ab)}\right)^6 = \sqrt{\epsilon^{(a)}\left(\sigma^{(a)}\right)^6 \epsilon^{(b)}\left(\sigma^{(b)}\right)^6}$$

$$\epsilon^{(ab)}\left(\sigma^{(ab)}\right)^{12} = \left\{ \frac{\sqrt[13]{\epsilon^{(a)}\left(\sigma^{(a)}\right)^{12}} + \sqrt[13]{\epsilon^{(b)}\left(\sigma^{(b)}\right)^{12}}}{2} \right\}^{13}$$

Uncontrolled approximation: See *e.g.* J. Phys. Chem. B 119 (2015) 14563

Fitting a pair potential

Please see Fitting_a_pair_potential.ipynb

The moral of the story: Transferability

Potentials parameterized for

- Structure
- Energetics
- Vibrations
- Anharmonicity
- Nanostructures
- Surfaces
- Bulk systems
- Solid phases
- Liquids

Often fail to describe

- Structure
- Energetics
- Vibrations
- Anharmonicity
- Nanostructures
- Surfaces
- Bulk systems
- Solid phases
- Liquids

Example: Transferability of water models

Calculated physical properties of the water models

Model	Dipole moment e	Dielectric constant	self-diffusion, 10^{-5} cm^2/s	Average configurational energy, kJ × mol^{-1}	Density maximum, °C	Expansion coefficient, 10^{-4} °C^{-1}
SSD	2.35 [511]	72 [511]	2.13 [511]	-40.2 [511]	-13 [511]	-
SPC	2.27 [181]	65 [185]	3.85 [182]	-41.0 [185]	-45 [983]	7.3 [704] **
SPC/E	2.35 [3]	71 [3]	2.49 [182]	-41.5 [3]	-38 [183]	5.14 [994]
SPC/Fw	2.39 [994]	79.63 [994]	2.32 [994]	-	-	4.98 [994]
PPC	2.52 [3]	77 [3]	2.6 [3]	-43.2 [3]	+4 [184]	-
TIP3P	2.35 [180]	82 [3]	5.19 [182]	-41.1 [180]	-91 [983]	9.2 [180]
TIP3P/Fw	2.57 [994]	193 [994]	3.53 [994]	-	-	7.81 [994]
iAMOEBA	2.78 [2031]	80.7 [2031]	2.54 [2031]	-	4 [2031]	2.5 [2031]
uAMOEBA	2.80 [2401]	76.3 [2401]	2.41 [2401]	-	-	3.38 [2401]
QCT **	1.85 [1251]	-	1.5 [1251]	-42.7 [1251]	+10 [1251]	3.5 [1251]
TIP4P	2.18 [3,180]	53 a [3]	3.29 [182]	-41.8 [180]	-25 [180]	4.4 [180]
TIP4P-Ew	2.32 [649]	62.9 [649]	2.4 [649]	-46.5 [649]	+1[649]	3.1[649]
TIP4P-FQ	2.64[197]	79 [197]	1.93 [197]	-41.4 [201]	+7 [197]	-
TIP4P/2005	2.305 [984]	60 [984]	2.08 [984]	-	+5 [984]	2.8 [984]
TIP4P/2005f	2.319 [1765]	55.3 [1765]	1.93 [1765]	-	+7 [1765]	-
TIP4P/ε	2.4345 [2444]	78.3 [2444]	2.10 [2444]	-	+4 [2444]	-
OPC	2.48 [2168]	78.4 [2168]	2.3 [2168]	-	-1 [2168]	2.7 [2168]
OPC3	2.43 [2722]	78.4 [2722]	2.3 [2722]	-	-13 [2722]	4.3 [2722]
SWFLEX-AI	2.69 [201]	116 [201]	3.66 [201]	-41.7 [201]	-	-
COS/G3 **	2.57 [704]	88 [704]	2.6 [704]	-41.1 [704]	-78[1939]	7.0 [704]
COS/D2	2.55 [1617]	78.9 [1617]	2.2 [1617]	-41.8 [1617]	-	4.9 [1617]
GCPM	2.723 [859]	84.3 [859]	2.26 [859]	-44.8 [859]	-13 [859]	-
SWM4-NDP	2.461 [933]	79 [933]	2.33 [933]	-41.5 [933]	<-53 [1999]	-
BK3	2.644 [2080]	79 [2080]	2.28 [2080]	-43.32 [2080]	+4 [2080]	3.01 [2080]
SWM6	2.431 [1999]	78.1 [1999]	2.14 [1999]	-41.5 [1999]	-48 [1999]	-
TIP5P	2.29 [180]	81.5 [180]	2.62 [182]	-41.3 [180]	+4 [180]	6.3 [180]
TIP5P-Ew	2.29 [619]	92 [619]	2.8 [619]	-	+8 [619]	4.9[619]
TTM2-F	2.67 [1027]	67.2 [1027]	1.4 [1027]	-45.1 [1027]	-	-
POL5/TZ	2.712 [256]	98 [256]	1.81 [256]	-41.5 [256]	+25 [256]	-
Six-site *	1.89 [491]	33 [491]	-	-	+14 [491]	2.4 [491]
MB-pol [3133]	-	68.4	2.3	-	-10	3.5
Experimental	2.95	78.4	2.30	-41.5 [180]	+3.984	2.53

All the data is at 25 °C and 1 atm, except * at 20 °C and ** at 27 °C.

Source: `http://www1.lsbu.ac.uk/water/water_models.html`

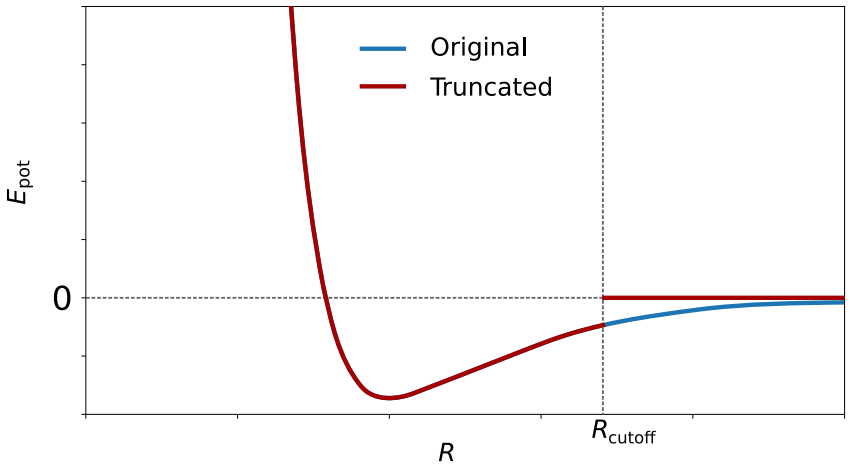
Cutoffs: The problem

Most pair potentials:

- Decay quickly with R (short sighted), but
- Have non-zero value for any finite r (non-bounded support)

$\implies$ number of atoms pairs scales quadratically

If one could ignore atom pairs with $R > R_{\text{cutoff}}$, scaling would be linear. Naive approach:

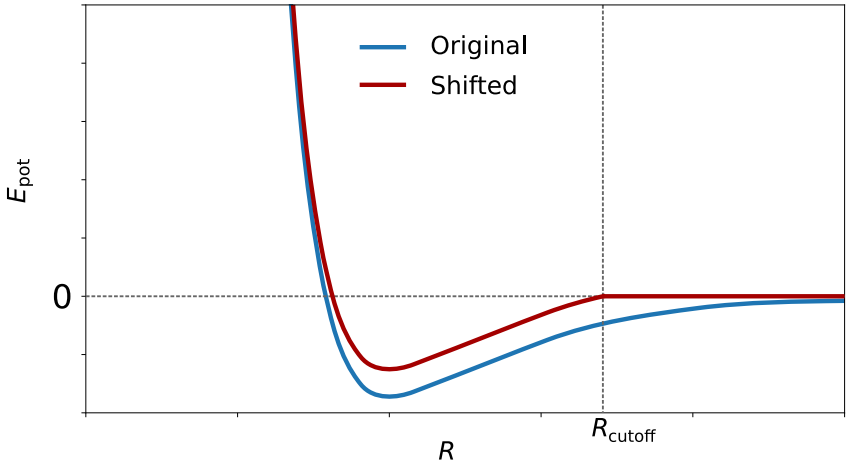


Unphysical consequences:

- Discontinuity of potential energy
- What happens with forces?

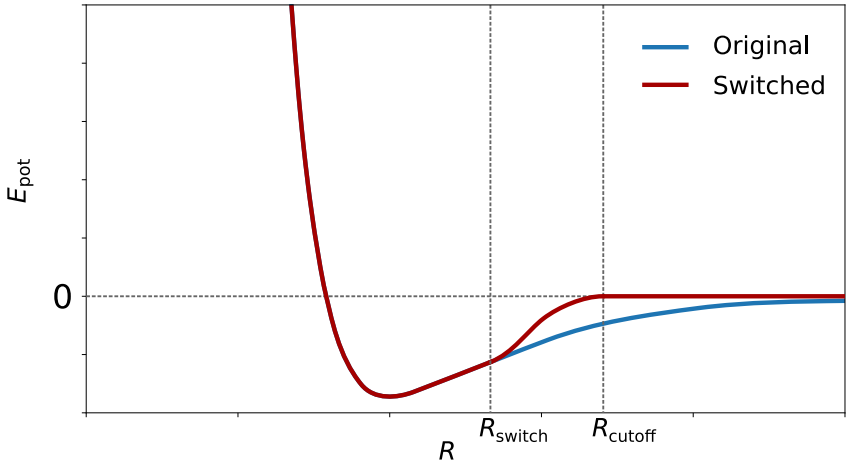
Cutoffs: Smarter approaches

Shifting:



- Rigid displacement
- Affects the whole range
- Forces still undefined

Switching:



- Multiplication by smooth function (1 at R_{switch}, 0 at R_{cutoff})
- Number of continuous derivatives controllable
- Does not affect other regions

A C^∞ bump function

Recipe for a nice cutoff function:

1. Start with the C^∞ function $f(t) = \begin{cases} e^{-1/t} & \text{, if } t > 0 \\ 0 & \text{, otherwise} \end{cases}$
2. Use it to build a smooth step function $g(t) = \frac{f(t)}{f(t)+f(1-t)}$
3. Make it symmetric and ajust the range:

$$\boxed{\rho(x) = 1 - g\left(\frac{x^2 - R_{\text{switch}}^2}{R_{\text{cutoff}}^2 - R_{\text{switch}}^2}\right)}$$

Reference: L. W. Tu, *An Introduction to Manifolds*, 2nd edition, Springer (2011). ISBN: 1441973990

Some potentials for solid-state simulations

The Stillinger-Weber potential (1985)

A paradigmatic three-body potential

$$V_2(R) = A\epsilon\left[B\left(\frac{\sigma}{R}\right)^p - \left(\frac{\sigma}{R}\right)^q\right]e^{\frac{\sigma}{R-a\sigma}}$$

$$V_3(R_{IJ}, R_{IK}, \theta_{IJK}) = \lambda\epsilon(\cos\theta_{IJK} - \cos\theta_0)^2 e^{\frac{\gamma\sigma}{R_{IJ}-a\sigma}} e^{\frac{\gamma\sigma}{R_{IK}-a\sigma}}$$

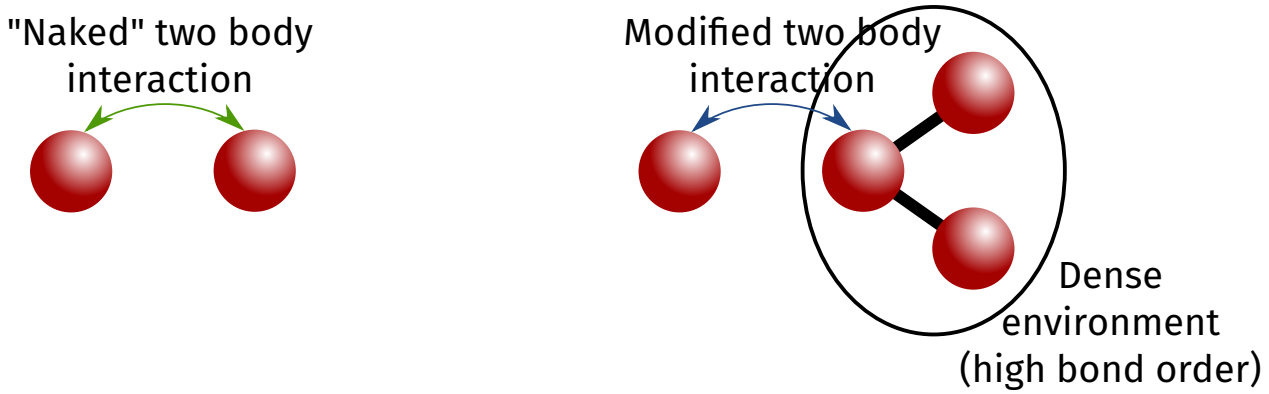
- Parameters: A, ϵ, B, σ, p, q, a, λ, θ_0, γ
- Original success: solid and liquid phases of Si, based on experimental data. $\cos\theta_0 = -1/3$ explains why
- Reparameterized time and time again: GaN, 2D materials, Si (redux)...

Bond-order potentials: Tersoff

Effective two-body term influenced by the environment

$$
\begin{aligned}
V_2(R_{IJ}) =& f_c(R_{IJ})[f_r(R_{IJ}) + b_{IJ} f_a(R_{IJ})] \\
f_r(R_{IJ}) =& A e^{-\lambda_1 R_{IJ}} \\
f_a(R_{IJ}) =& -B e^{-\lambda_2 R_{IJ}} \\
b_{IJ} =& (1 + \beta^n \xi_{IJ}^n)^{-\frac{1}{2n}} \\
\xi_{IJ} =& \sum_{K \notin \{I,J\}} f_c(R_{IK}) g(\theta_{IJK}) e^{\lambda_3^m (R_{IJ} - R_{IK})^m} \\
g(\theta) =& \gamma \left\{ 1 + \frac{c^2}{d^2} - \frac{c^2}{d^2 + (\cos\theta - \cos\theta_0)^2} \right\} \\
f_c(R) =& \begin{cases} 1, & R < C - D \\ \frac{1}{2} - \frac{1}{2}\sin\left(\frac{\pi}{2}\frac{R-C}{D}\right), & C - D < R < C + D \\ 0, & R > C + D \end{cases}
\end{aligned}
$$

The concept behind bond-order potentials



From a fundamental point of view, an atom with saturated bonds will not interact as strongly with others.

Transferability?

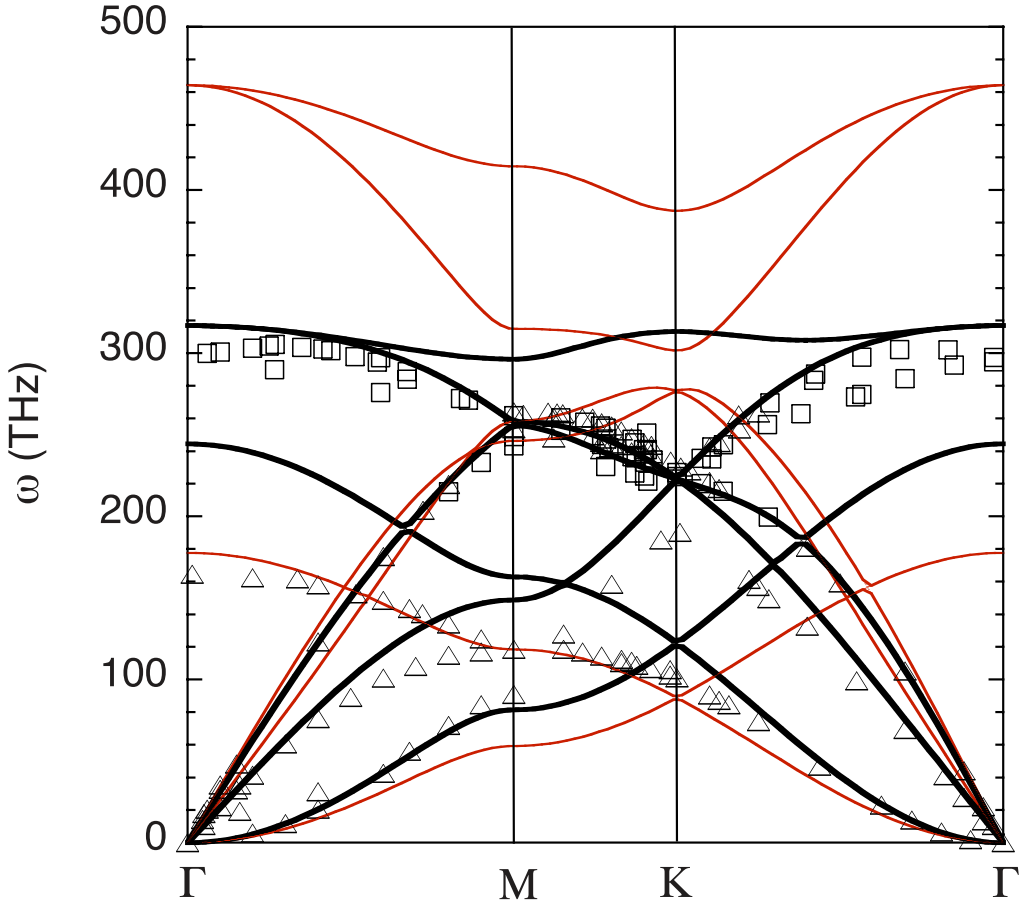


FIG. 1. (Color online) Phonon dispersion for graphene along high-symmetry directions obtained using the Tersoff EIP. Thick black lines correspond to the optimized parameter set (this work); thin red lines correspond to the original parameter set. Squares (triangles) are in-plane experimental data points for graphite from Ref. 43 (Ref. 44).

Source:
Phys. Rev. B 81 (2010) 205441

For a more depressing read, see *Effects of Cutoff Functions of Tersoff Potentials on Molecular Dynamics Simulations of Thermal Transport*, Modelling Simul. Mater. Sci. Eng. 19 (2011) 025004.

Machine-learning (ML) potentials

Mapping the energy landscape

$E_{\text{pot}}\left(\{\boldsymbol{x}_I\}_{i=1}^{M}\right)$ defines a manifold with complex features:

- Number of minima exponential on M
- Multiple transition paths
- Features at different length and energy scales

ML provides powerful general regression techniques to create controlled interpolations on a region of this surface.

Back to Cl_2

We have already parameterized a ML potential:

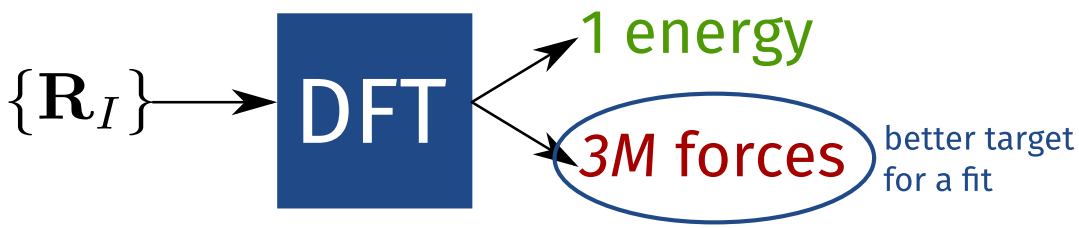
```
data = np.loadtxt("e_vs_l.txt")
r = data[:, 0]
E = data[:, 1] * ase.units.Hartree
newr = np.linspace(r.min(), r.max(), num=1000)
newE = sp.interpolate.interp1d(r, E, kind="cubic")(newr)
```

The interpolating spline:

- \+ Does not rely on “chemical insight”
- \+ Outperforms the Morse potential
- \+ Has clear limits for transferability: `E.min()`, `E.max()`
- \- Analytical calculations mostly out of the question
- \- Not easy to “interpret”

The ML workflow

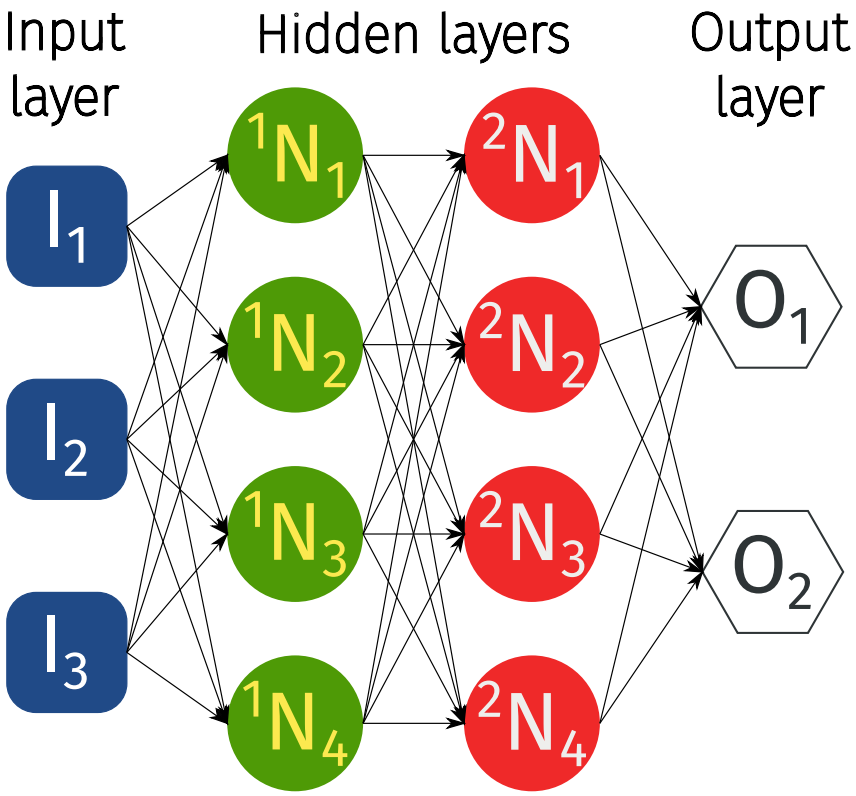
1. Data gathering



2. Choice of regression method
   - Descriptors (inputs)
   - Functional form and target
   - Regularization
3. Training: Parameter fitting
4. Validation: Assessment of transferability and accuracy
5. Application

Not fundamentally different from training a self-driving car, a spam filter or Google Translate

Neural networks



- Each "neuron"/perceptron transforms its inputs $\{J_i\}$:
  1. Linear combination + constant: $L = b + \sum_i a_i J_i$
  2. Nonlinear activation function: $N = f(L)$
- Activation function: $\tanh$, ELU, ReLU...
- Huge number of parameters, but can be adjusted systematically (*e.g.* backpropagation).
- This basic idea has been greatly expanded recently (*e.g.* deep learning).

Other popular approaches: Gaussian process regression, kernel methods

Methods for validation

Overfitting happens when a model is trained to reproduce meaningless minutia of the input data at the expense of transferability.

To assess and avoid overfitting, fits must be validated on out-of-sample data. Several strategies are possible:

Data splitting: Use a given percentage of the data for fitting the model, assess its accuracy using the rest.

Bootstrap: Extract hundreds or thousands of samples from the dataset (with replacement), fit the model each time, evaluate its performance with the remaining data.

k-fold cross-validation: Split the data in k equal parts, fit the model to each subset of $k-1$ in sequence, measuring its quality based on the remaining "fold".

Repeated k-fold cross-validation: Run several k-fold cross-validation steps with random splits.

Leave-one-out cross-validation: Fit the model to all data minus one point, repeatedly, and use that lone point to obtain validation statistics.

Useful software

- `http://scikit-learn.org` : a comprehensive, well documented ML kit for Python, covering the whole workflow
- `https://www.r-project.org` The language of statistics. Check out the Rstudio (`https://www.rstudio.com/`) environment and the caret (`http://caret.r-forge.r-project.org`) package.
- `https://www.tensorflow.org/` : Google's open source software library for numerical computation using data flow graphs.
- `https://pytorch.org/` : Facebook's competitor to TensorFlow

Descriptors

Big challenge for NN potentials. Must be invariant w.r.t.:

- Rotation
- Translation
- Permutation of equivalent atoms

Still a work in progress

Descriptors: Three examples

Atom-centered symmetry functions (ACSFs):

$$
\begin{aligned}
G_I^{\mathrm{rad}} &= \sum_J e^{-\eta(R_{IJ}-Rs)^2} f_{\mathrm{cut}}(R_{IJ}) \\
G_I^{\mathrm{ang}} &= 2^{1-\xi} \sum_{J,K\neq I} (1+\lambda\cos\theta_{IJK})^{\xi} e^{-\eta(R_{IJ}^2+R_{IK}^2+R_{JK}^2)} \times \\
&\quad f_{\mathrm{cut}}(R_{IJ}) f_{\mathrm{cut}}(R_{IK}) f_{\mathrm{cut}}(R_{JK})
\end{aligned}
$$

Smooth overlap of atomic positions:

$$
\rho_{\mathrm{SOAP}}(\boldsymbol{x}) = \sum_J e^{-\alpha|\boldsymbol{x}-\boldsymbol{x}_J|^2}
$$

Eigenvalues of the Coulomb matrix:

$$
M_{IJ} = \begin{cases} 0.5 Z_I^{2.4}, & \text{if } I = J \\ \frac{Z_I Z_J}{|\boldsymbol{x}_I - \boldsymbol{x}_j|}, & \text{if } I \neq J \end{cases}
$$

A Gaussian approximation potential (GAP) for graphene

Phys. Rev. B 97 (2018) 054303

- Sum of atomic contributions based on SOAP
- Training data: energies, forces and stresses from DFT
- Around 1000 configurations in the training set
- DFT-like accuracy
- Great transferability: phonons, finite-T anharmonicity, elasticity...

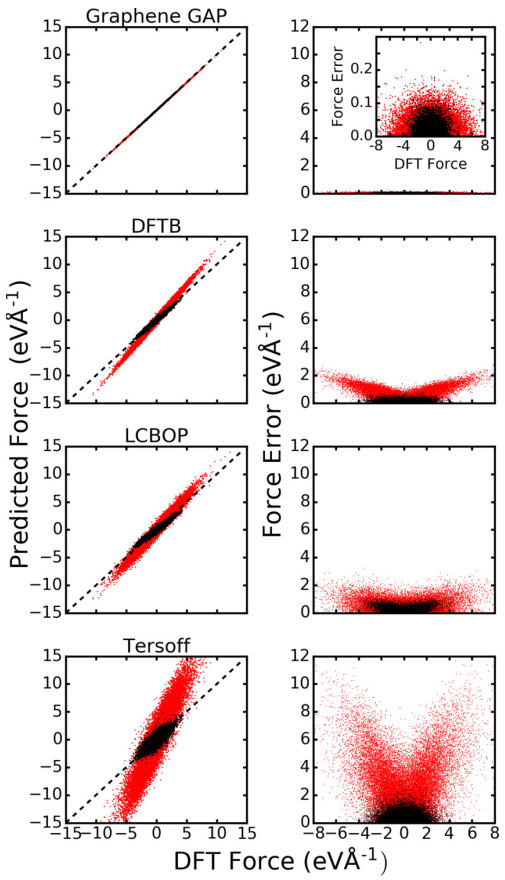
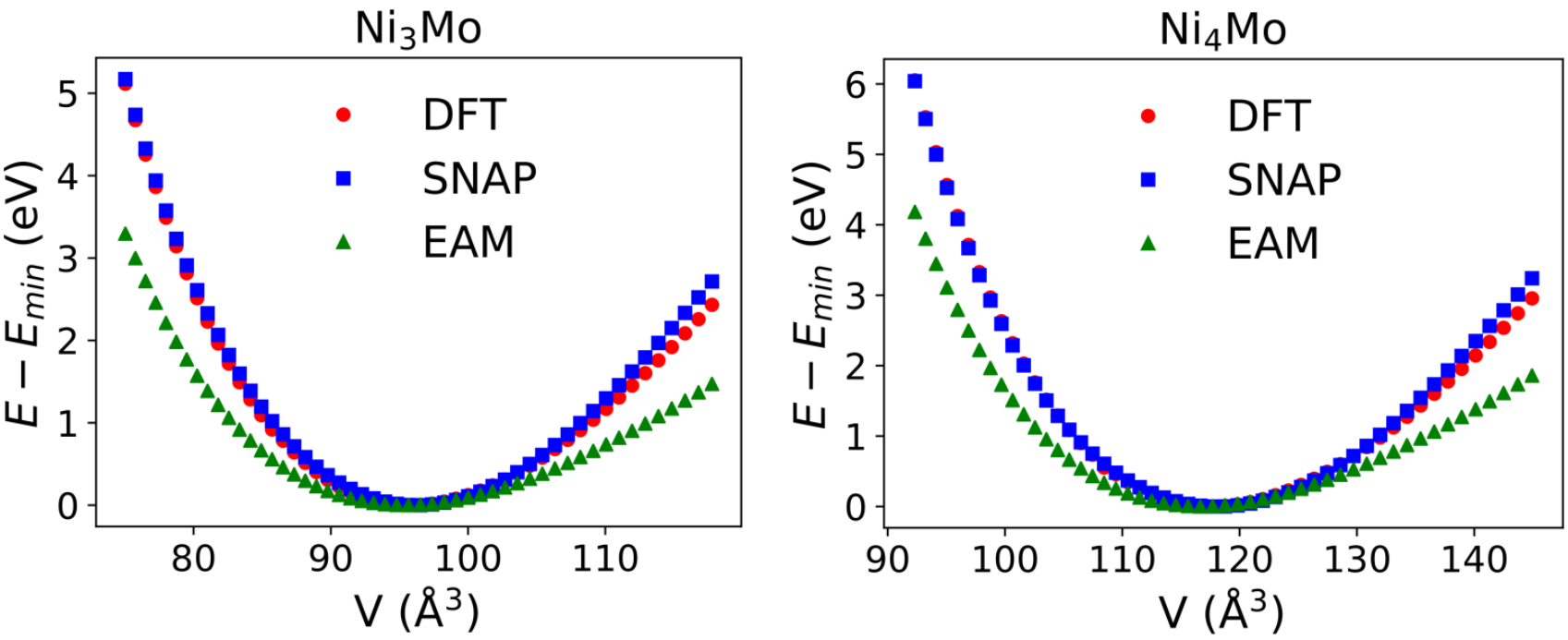


FIG. 1. Force correlations (left) and associated force errors (right) on an independent reference data set of conf gurations for the graphene GAP model, DFTB, LCBOP, and Tersoff potentials as compared to the reference DFT method, the plots for all methods considered can be found in the SM. Black points indicate forces perpendicular to the plane of the graphene sheet (out-of-plane) while red points indicate forces oriented in the plane. The inset in the graphene GAP plot has a different scale on the y axis to show more clearly the distribution of force errors, which are smallest for large forces with a Gaussian distribution.

SNAP (Spectral network analysis potential) vs classical

Equation of state for Ni-Mo alloy:



[EAN: Embedded atom method]

Source: X.-G. Li *et al.*,

`https://lammps.sandia.gov/workshops/Aug19/talk_li.pdf` (2019)

ANI-1: An extensible neural network potential

Chem. Sci. 8 (2017) 3192

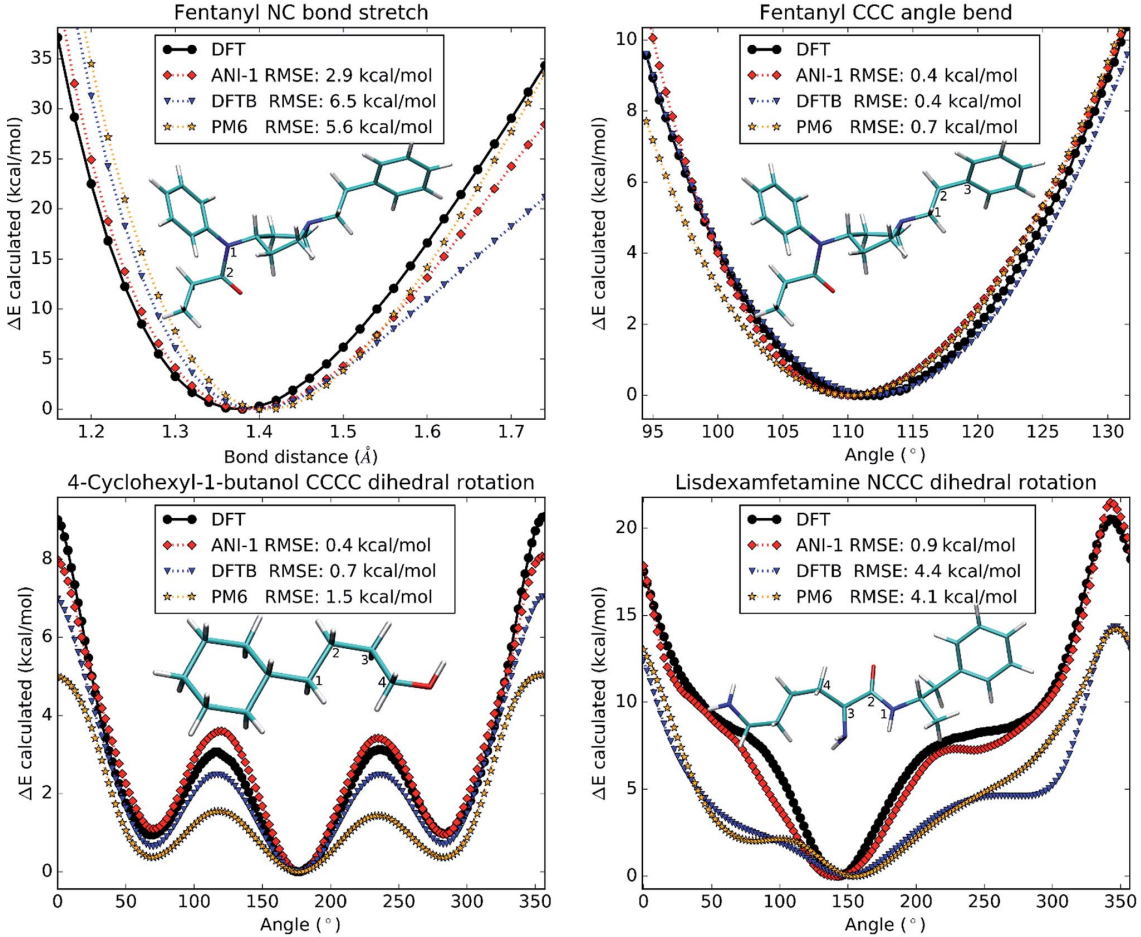


Fig. 7 Each subplot shows a one-dimensional potential surface scan generated from DFT, the ANI-1 potential, and two popular semi-empirical methods, DFTB and PM6. The atoms used to produce the scan coordinate are labeled in the images of the molecules in every sub-plot. Each figure also lists the RMSE, in the legend, for each method compared to the DFT potential surface.

- Sum of atomic contributions based on ACSFs
- Deep-learning NN
- 57951 molecules from the GDB-11 database
- Configuration sampling based on normal modes

What about the long-range part?

No commonly accepted approach yet. Some possibilities:

- Fixed atomic multipoles (charges, dipoles...– see next section)
- Machine-learning method for the charge density, potential energy computed from it [CENT Method, Goedecker and coworkers, Phys. Rev. B 92 (2015) 045131]
- Project long-range part onto local descriptors [A. Grisafi & M. Ceriotti M, J. Chem. Phys. 151 (2019) 204105]

Very relevant for systems with local or global charges

Evaluating the forces

First derivatives of the potential required for:

- Molecular dynamics
- Training of advanced potentials using forces

[And we may need higher orders too!]

By hand: 👍 for Lennard-Jones

😱🤣 for advanced potentials

Computer algebra systems + code generation:

🐌 (redundant calculations)

⚠️ (loss of precision)

Numerical differentiation:

🐌 (lots of evaluations)

⚠️ (fragile)

Automatic differentiation: 👌

Analytic differentiation + code generation (input)

- Let a computer algebra system do most of the work

Code

```python
#!/usr/bin/env python

import sympy
from sympy.utilities.codegen import codegen

x = sympy.Symbol("x")
y = sympy.Symbol("y")

r = sympy.sqrt(x**2 + y**2)
V = sympy.exp(r / 1.5) * sympy.cos((r - 1.) / 3.)
fx = -sympy.diff(V, x)
fy = -sympy.diff(V, y)
result = codegen([("fx", fx), ("fy", fy)], "C")

print(result[0][1])
```

Analytic differentiation + code generation (result)

Code generated with SymPy 1.5.1

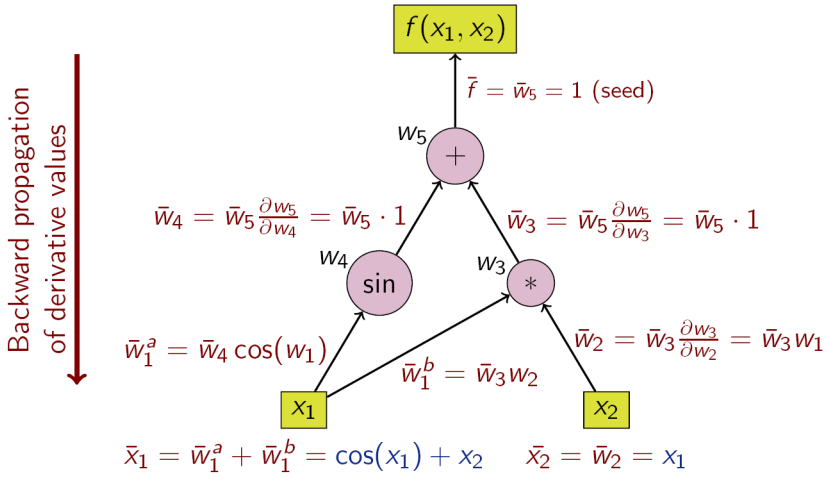
```
#include <math.h>

double fx(double x, double y)
{
    double fx_result;
    fx_result =
        0.33333333333333331 * x *
            exp(0.66666666666666663 * sqrt(pow(x, 2) + pow(y, 2))) *
            sin(0.33333333333333331 * sqrt(pow(x, 2) + pow(y, 2)) -
                0.33333333333333331) / sqrt(pow(x, 2) + pow(y, 2)) -
        0.66666666666666663 * x *
            exp(0.66666666666666663 * sqrt(pow(x, 2) + pow(y, 2))) *
            cos(0.33333333333333331 * sqrt(pow(x, 2) + pow(y, 2)) -
                0.33333333333333331) / sqrt(pow(x, 2) + pow(y, 2));
    return fx_result;
}

double fy(double x, double y)
{
    double fy_result;
    fy_result =
        0.33333333333333331 * y *
            exp(0.66666666666666663 * sqrt(pow(x, 2) + pow(y, 2))) *
            sin(0.33333333333333331 * sqrt(pow(x, 2) + pow(y, 2)) -
                0.33333333333333331) / sqrt(pow(x, 2) + pow(y, 2)) -
        0.66666666666666663 * y *
            exp(0.66666666666666663 * sqrt(pow(x, 2) + pow(y, 2))) *
            cos(0.33333333333333331 * sqrt(pow(x, 2) + pow(y, 2)) -
                0.33333333333333331) / sqrt(pow(x, 2) + pow(y, 2));
    return fy_result;
}
```

Automatic differentiation

- Usually, potentials are built from a limited number of functions (exponentials, trigonometric, sums, multiplications...)
- Recursive, automated, optimised application of the chain rule $\frac{\mathrm{d}f[g(x)]}{\mathrm{d}x} = f'[g(x)]g'(x) \implies$ the function and the derivative can be evaluated at the same time
- Numerical precision/accuracy comparable to function itself



(Source: Berland at the English language Wikipedia)

The JAX framework

Automatic differentiation and JIT compilation for Python and NumPy

- Google-sponsored research project
- Goal: high-performance machine-learning research (great for NNs)
- Targets XLA (Accelerated Linear Algebra), designed to accelerate TensorFlow
- Good performance on CPUs, GPUs and TPUs
- Foundation for new ecosystem of machine learning (Flax), optimization (Optax)...

Home page: `https://github.com/google/jax`

JAX in action

Lennard-Jones with derivatives and test

```python
#!/usr/bin/env python

import numpy.random as random
import jax
import jax.numpy as np

def lj_potential(positions):
    """Lennard-Jones potential in reduced units.

    'positions' must be an Mx3 array.
    """
    # Compute all relative positions between pairs without iterating.
    delta = positions[:, np.newaxis, :] - positions
    # Take only the combinations of two different atoms.
    indices = np.triu_indices(positions.shape[0], k=1)
    delta = delta[indices[0], indices[1], :]
    # Compute the squared distances and evaluate the potential energy.
    r2 = (delta * delta).sum(axis=1)
    rm2 = 1. / r2
    rm6 = rm2 * rm2 * rm2
    rm12 = rm6 * rm6
    return (rm12 - 2. * rm6).sum()

lj_gradient = jax.jit(jax.grad(lj_potential))  # Automatic gradient function
# Put 20 particles in a cube of side 5.
test_positions = 5. * random.random_sample((20, 3))
e_pot = lj_potential(test_positions)
forces = -lj_gradient(test_positions)
print("Potential energy:", e_pot, "Forces:", forces, sep="\n")
```

Potentials for molecular mechanics

Molecular mechanics

Specialized domain of atomistic simulation: molecular systems, with a bent towards organic molecules in fluid phases

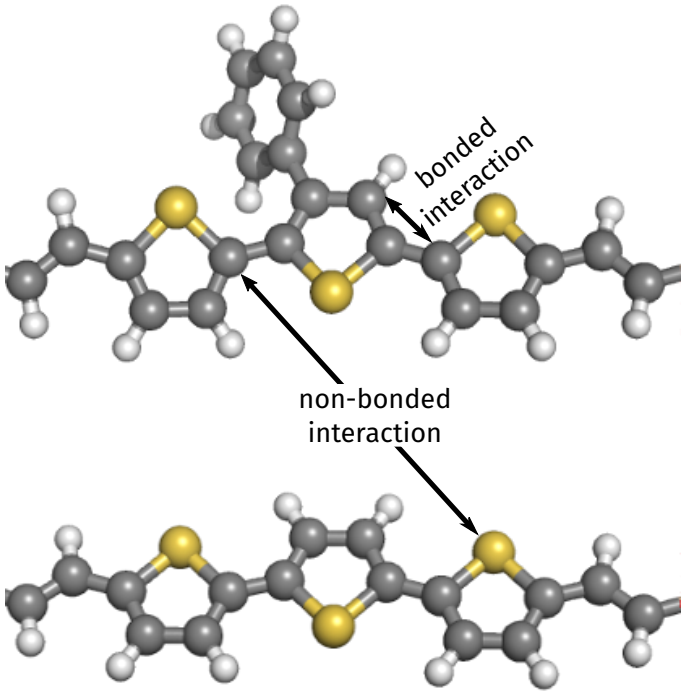
---

Features:

- Molecular geometry + topology
- Bonded vs. non-bonded interactions
- Atoms have vdW radii, partial charges and possibly polarizabilities

Variations:

- All-atom vs. united-atom
- Coarse graining
- Hybrid MM + *ab-initio*



OPLS-AA

Jorgensen *et al.*, J. Am. Chem. Soc. 118 (1996) 11225

A typical molecular-mechanics potential

$$V_{\text{non-bonded}}(a,b) = \sum_{\substack{I\in a\\ J\in b}} f_{IJ}\left\{4\epsilon_{IJ}\left[\left(\frac{\sigma_{IJ}}{R_{IJ}}\right)^{12} - \left(\frac{\sigma_{IJ}}{R_{IJ}}\right)^{6}\right] + \frac{1}{4\pi\varepsilon_0}\frac{q_i q_j}{R_{IJ}}\right\},$$

$$V_{\text{bonded}} = V_{\text{stretch}} + V_{\text{angle}} + V_{\text{dihedrals}}$$

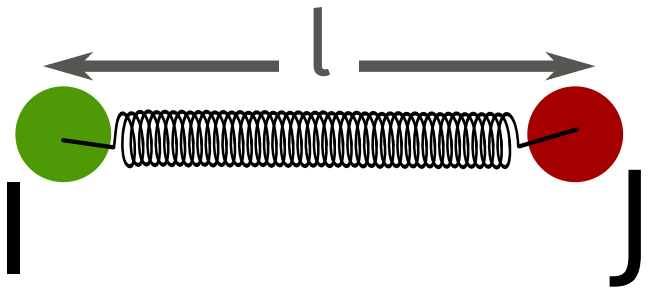
$$V_{\text{stretch}} = \sum_{(I,J)\text{ are bonded}} K_R(I,J)\left[R_{IJ} - R_{0,IJ}\right]^2$$

$$V_{\text{angle}} = \sum_{\substack{(I,J,K)\text{ form}\\ \text{an angle}}} K_\theta(I,J,K)\left[\theta_{IJK} - \theta_{0,IJK}\right]^2$$

$$V_{\text{dihedrals}} = \sum_{\text{dihedrals }\Phi_s} \left\{\frac{V_1(s)}{2}[1+\cos\Phi_s] + \frac{V_2(s)}{2}[1-\cos(2\Phi_s)] + \right.$$
$$\left. + \frac{V_3(s)}{2}[1+\cos(3\Phi_s)] + \frac{V_4(s)}{2}[1-\cos(4\Phi_s)]\right\}$$

Bonds and angles

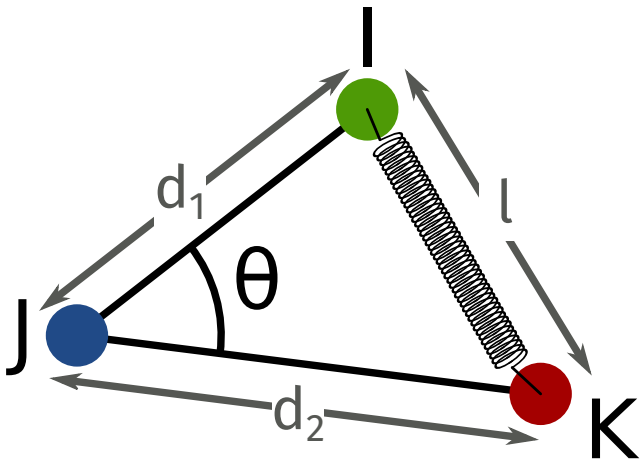
Bond term:



Harmonic potential:

$\Delta E_{\text{pot}} = \frac{1}{2}\tilde{K}(l - l_0)^2$

Angle term:



Approximate harmonic term:

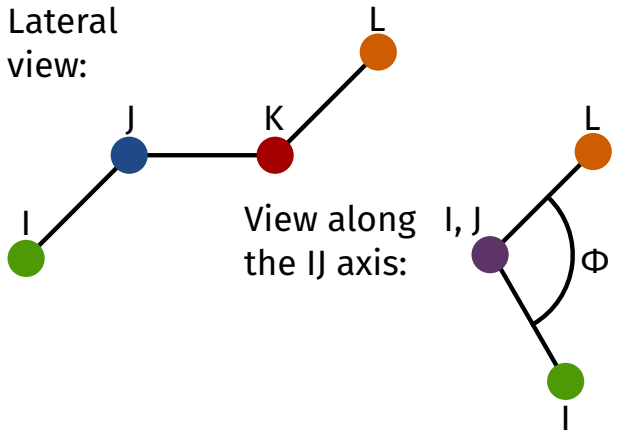
$$\Delta E_{\text{pot}} = \frac{1}{2}\tilde{K}(l - l_0)^2$$

$$l = \sqrt{d_1^2 + d_2^2 - d_1 d_2 \cos\theta}$$

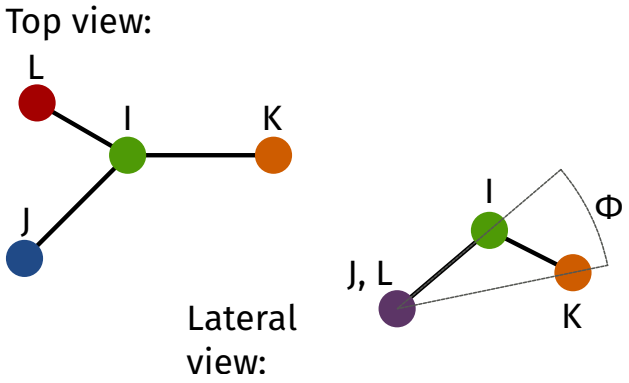
$$\Delta E_{\text{pot}} \simeq \frac{1}{2}\tilde{K}\left(\left.\frac{\mathrm{d}l}{\mathrm{d}\theta}\right|_{\theta=\theta_0}\right)^2 (\theta - \theta_0)^2$$

Dihedrals: A 4-body term

Proper dihedrals:



Improper dihedrals:



Ryckaert-Bellemans functional form:

$$\Delta E_{\text{pot}} = \frac{V_1}{2}[1+\cos\Phi] + \frac{V_2}{2}[1-\cos(2\Phi)] + \frac{V_3}{2}[1+\cos(3\Phi)] + \frac{V_4}{2}[1-\cos(4\Phi)]$$

$\Rightarrow$ different symmetries depending on which constants are zero.

The 1-4 reduction

"Fudge" factor for non-bonded interactions:

$$f_{IJ} = \begin{cases} 0.5, & \text{for atoms up to three bonds apart} \\ 1, & \text{for other pairs of atoms} \end{cases}$$

Goal: Avoid double counting of non-bonded interactions, whose perturbative expansions give rise to bond and angle terms.

Parameterizing proper dihedrals

Please see Parameterizing_dihedrals.ipynb

Schemes for obtaining the charges

No notion of "atomic charges" in quantum mechanics $\Rightarrow$ many different schemes to assign partial charges:

- Formal charges
- Restrained electrostatic potential (RESP)
- Mulliken charges
- Bader charges
- Born charge tensors
- ...(~ 30 schemes listed at `https://en.wikipedia.org/wiki/Partial_charge`)

Bader charges

Comput. Mater. Sci., 36 (2006) 35

- Based on charge density $n(\boldsymbol{x})$
- Space partitioned with zero-flux surfaces (inflection points of n)
- $Q_I =$ integral of n in the Bader volume of I
- Preserves total charge

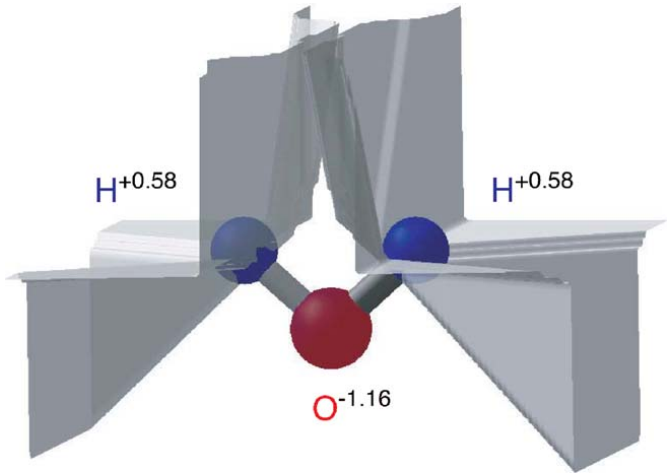


Fig. 2. Dividing surfaces separating the oxygen and hydrogen Bader regions in a water molecule. The integration over the Bader regions indicates that 0.58 e are transferred from each hydrogen atom to the oxygen atom.

RESP

J. Phys. Chem. 97 (1993) 10269

- Charges used as a fitting parameters to reproduce the electrostatic component of DFT results
- Some constraints imposed:
  - Equivalent charges for equivalent atoms
  - Fixed total charge of known charge groups
  - Hyperbolic penalty restrain on the charges avoids spurious values
- Prescriptive for many molecular-mechanics force fields
- Many similar methods exist (see J. Comput. Chem. 19 (1998) 377)

		Mulliken	NPA	AIM[a]	CHELP	CHELPG	RESP	MK	CHELP-BOW	CHELMO
$(CH_3)_2SO$	C	0.63	0.90	0.17	0.29	0.25	0.18	0.49	0.77	0.92
	H	0.17–0.20	0.24–0.26	0.04–0.06	0.05	0.11	0.09	0.18	0.26	0.30
	S	0.75	1.26	1.27	0.08	0.22	0.17	0.28	0.35	0.40
	O	0.62	0.96	1.25	0.39	0.40	0.39	0.39	0.38	0.38
H_2O	O	0.77	0.93	1.09	0.75	0.75	0.74	0.75	0.74	0.73
	H	0.39	0.47	0.55	0.38	0.37	0.37	0.37	0.37	0.37
$(H_2O)_2$	O_{donor}	0.83	0.97	1.15	0.79	0.81	0.80	0.81	0.81	0.81
	H	0.40	0.48	0.58	0.35	0.37	0.36	0.37	0.37	0.37
	$H_{H\text{-}bond}$	0.38	0.46	0.54	0.39	0.38	0.38	0.38	0.38	0.39
	$O_{acceptor}$	0.75	0.94	1.10	0.76	0.74	0.73	0.74	0.75	0.75
	H	0.41	0.49	0.57	0.41	0.40	0.40	0.40	0.40	0.40

Interatomic potentials

Numerical Simulation and Scientific Computing II

Jesús Carrete Montaña
jesus.carrete.montana@tuwien.ac.at

Lecture 6

TECHNISCHE UNIVERSITÄT WIEN