Documentation for TULIP 2

K. Henriksson

January 7, 2016

Abstract

This booklet gives an introduction to the TULIP, which is a program for fitting interatomic potentials to physical data of different phases and lattices.

Contents

1 Introduction

The program TULIP tries to fit a interatomic potential to *e.g.* given physical properties of compounds *etc*.. Physical properties are for instance lattice parameters, cohesive energy, bulk modulus and elastic constants.

Input is essentially an initial guess for potential parameters, a list of compounds and their desired properties (lattice parameters, cohesive energy, mixing energy, bulk modulus, elastic constants, ...), and technical specifications to guide the fitting. Using the parametrization, compounds are relaxed in a molecular dynamics (MD) simulation (MDS), and then the specified physical properties are calculated. The fitting routine tries to minimize the merit function. The fitting method can be selected.

2 Source code

0. Prerequisites: Install the `spglib` package from `http://spglib.sourceforge.net/` in a standard location and remember the location of installed files.

1. Local users only: Download the `libutils` source code into a folder:

```
git clone /home/phys-data/people/koehenri/repos/libutils.git
```

The `libutils` directory will be created. Descend into it and follow the instructions in the Readme file to make and install the library.

2. Local users only: Download the `tulip2` source code into a folder:

```
git clone /home/phys-data/people/koehenri/repos/tulip2.git
```

The `tulip2` directory will be created. Descend into it and follow the instructions in the Readme file to make and install the fitting code.

3. The static version `tulip_static` can be copied to other computer systems having the same architecture and be run there without any additional effort. The shared library version requires adding the `libutils` library to the library path.

⋆ ⋆ ⋆

To download updated versions of these codes descend into the root directories (e.g. `tulip2`) and execute

```
git pull
```

This will fetch and merge updated code with your local copy.

⋆ ⋆ ⋆

The `tulip2/examples` directory contains some ready-made examples, which can be run without additional editing to get a better feel for how the code works.

3 Program arguments

```
TULIP version 2 (c) Krister Henriksson 2013-
Purpose: Fit data to an interatomic potential.
Usage:
     tulip arguments [options]
Arguments:
     -pf file           Path to file containing potential information.
     -gf file           Path to file containing geometry information.

Options:
     -sf file           Path to file containing technical specifications about the calculations.
     -ro                Only calculate properties of reference compounds, then exit. Default: not used.
     -nof               Only calculate properties of reference and read-in compounds, then exit. Default: not used.
     -xyz               Use traditional XYZ format when writing XYZ files. Default: not used.
                        The extended XYZ format (http://jrkermode.co.uk/quippy/io.html#extendedxyz)
                        is used by default.

     -dfitpropn         Show information about fitting of properties. Here 'n' must be
                        an integer. Supported: 0-4. 0: debug fitting method. 1-4: debug deeper.
                        lying methods used by the fitting method. Default: not used
                        NOTE: 0 also shows some info about the initial Chi^2 object.
     -dfitpotn          Show information about fitting of potentials. Here 'n' have a similar
                        role as for fitting of the properties.
                        NOTE 1: 0 also shows some info about the initial Chi^2 object.
                        NOTE 2: 'fitpot0' is always set to true, others are false by default.

     -dforces           Debug the forces. Default: not used
     -dpressure         Debug the pressure. Default: not used
     -dmdsprop          Debug MDS runs of the structures. Default: not used
     -dall              Activate all debugging options (top level only). Default: not used

     -mif               Suggest an initial fit and exit. Default: not used
                        ABOP: Put D0=0.0 for the binary interaction you want to fit. Keep all other
                        parametrizations at their normal values.
```

Main arguments:

```
-pf potinfofile     This file contains info about the elements and interactions.
-gf geominfofile    This file contains the info about the compounds which are
                    to be fitted.
```

Recommended options to always use:

```
-sf specsinfofile   This file contains settings steering the calculation of
                    properties of read-in compounds and the fitting process.
```

Useful options:

```
-ro             Calculate properties of reference (single-species) compounds
                and quit.
-nof            Calculate properties of reference and read-in compounds and quit.
                This is useful when parametrization is finalized and high-accuracy
                values of properties are desired (use long MD relaxation times!).
-mif            Obtain an initial partial fit to start from.
-dmdsprop       Show progress of all MD relaxations of all compounds.
```

4 Potentials

In TULIP the following potentials are subject to fitting: (i) the ABOP (Analytical Bond Order Potential). The EAM (Embedded Atom Method) potential is understood by the program, but this type of interactions cannot be fitted.

4.1 ABOP potential

The Abell-Brenner-Tersoff [1, 4, 8] or ABOP [5] potential gives the total energy of a system of atoms as

$$V = \frac{1}{2}\sum_i \sum_j f_{c,ij}\left(V_{R,ij} - B_{ij}V_{A,ij}\right) = \sum_i \sum_{j>i} f_{c,ij}\left(V_{R,ij} - \overline{B}_{ij}V_{A,ij}\right), \tag{1}$$

Here

$$\overline{B}_{ij} = \frac{B_{ij} + B_{ji}}{2} \tag{2}$$

Note:

$$V_{ij} \equiv f_{c,ij}\left(V_{R,ij} - B_{ij}V_{A,ij}\right) \tag{3}$$

may not be equal to V_{ji}.

The repulsive (R) and attractive (A) parts are

$$V_{R,ij} = \frac{D_0}{S-1}\exp\left[-\beta\sqrt{2S}(r_{ij} - r_{0,ij})\right] \tag{4}$$

$$V_{A,ij} = \frac{SD_0}{S-1}\exp\left[-\beta\sqrt{2/S}(r_{ij} - r_{0,ij})\right] \tag{5}$$

with

$$r_{ij} = |\mathbf{r}_{ij}| = |\mathbf{r}_i - \mathbf{r}_j| \tag{6}$$

$\star \qquad \star \qquad \star$

For the cutoff function f_c there are two choices:

1. The Tersoff [8] cutoff function is

$$f_c(r) = \begin{cases} 1, & r \leq R - D, \\ \frac{1}{2}\left(1 - \sin\left(\frac{\pi}{2}\frac{r-R}{D}\right)\right), & R - D < r < R + D \\ 0, & r \geq R + D \end{cases} \tag{7}$$

Hence, full interaction is felt when $r < R - D$, and no interaction when $r > R + D$, making the cutoff distance $r_c = R + D$. In order to use the Tersoff cutoff function, specify e.g. for Y-Y

```
potpar(Y, Y):cutscr = tersoff
```

2. The Perriot [7] cutoff function is

$$f_c^{(P)}(r) = \begin{cases} 1, & r < r_{min} \\ 1 - q^3(6q^2 - 15q - 10), & r_{min} \leq r \leq r_{max} \\ 0, & r > r_{max} \end{cases} \tag{8}$$

$$q = \frac{r - r_{min}}{r_{max} - r_{min}} \tag{9}$$

Here r_{min}, r_{max} are parameters. In order to use the Perriot cutoff function, specify e.g. for Y-Y

```
potpar(Y, Y):cutscr = perriot cut
```

Note the two strings! The second one must be exactly 'cut', not e.g. 'cutoff'!

The default cutoff function is 'none', which causes the program to exit with an error message if a cutoff function is not specified.

All parameters relating to a cutoff function must be specified after the 'cutscr' setting! Otherwise these parameters are not set.

⋆ ⋆ ⋆

The bond-order parameter is defined as

$$B_{ij} = (1 + \chi_{ij})^{-p_{ij}} \tag{10}$$

The usual ABOP has $p_{ij} \equiv 1/2$, which is the default if nothing is provided for this parameter.

Here

$$\chi_{ij} = \sum_{k,k\neq i,k\neq j} f_{c,ik} g_{ijk}(\theta_{ijk}) \omega_{ijk} \exp\left[\alpha_{ijk}(r_{ij} - r_{ik})\right] \tag{11}$$

⋆ ⋆ ⋆

We have three different possibilities:

(V1) α_{ijk} and ω_{ijk} are used as parameters.

(V2) If α_{ijk} is used but ω_{ijk} is not, then the Brenner form is used for the latter:

$$\omega_{ijk} = \exp\left[-\alpha_{ijk}(r_{0,ij} - r_{0,ik})\right] \tag{12}$$

and ω_{ijk} is **not** a separate parameter.

(V3) If α_{ijk} is not used and $2\mu_{ik}$ is used, then the whole factor

$$\omega_{ijk} \exp\left[\alpha_{ijk}(r_{ij} - r_{ik})\right] \tag{13}$$

is replaced in its entirety by

$$\exp\left[2\mu_{ik}(r_{ij} - r_{ik})\right] \tag{14}$$

⋆ ⋆ ⋆

The function $g_{ijk}(\theta_{ijk})$ is given by the expression

$$g_{ijk}(\theta_{ijk}) = \gamma_{ik}\left(1 + \frac{c_{ik}^2}{d_{ik}^2} - \frac{c_{ik}^2}{d_{ik}^2 + (h_{ik} + \cos\theta_{ijk})^2}\right) \tag{15}$$

where θ_{ijk} is the angle between the bonds ij and ik.

⋆ ⋆ ⋆

The Perriot [7] screening method can be used instead of a cutoff function. In the Perriot method [7] the total potential energy is given by

$$V = \frac{1}{2}\sum_i \sum_j K_{ij}\left(V_{R,ij} - B_{ij}V_{A,ij}\right) \tag{16}$$

The bond-order is

$$\chi_{ij} = \sum_{k,k\neq i,k\neq j} f_{c,ik}^{(P)} K_{ik} g_{ijk}(\theta_{ijk}) \omega_{ijk} \exp\left[\alpha_{ijk}(r_{ij} - r_{ik})\right] \tag{17}$$

The Perriot method contains the following functions:

$$K_{ij} = \exp\left[-\sum_{s,s\neq i,s\neq j} T_{ijs}^{n_{ij}}\right] \tag{18}$$

$$T_{ijs} = \begin{cases} -\frac{1}{2} + \frac{r_{ij}}{X_{is}+X_{js}-r_{ij}}, & X_{is}+X_{js} < 3r_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

$$X_{is} = \frac{r_{is}}{1-(r_{is}/r_{c,is})^{m_{is}}} \tag{20}$$

$$X_{js} = \frac{r_{js}}{1-(r_{js}/r_{c,js})^{m_{js}}} \tag{21}$$

$$f_c^{(P)}(r) = \begin{cases} 1, & r < r_{min} \\ 1-q^3(6q^2-15q-10), & r_{min} \leq r \leq r_{max} \\ 0, & r > r_{max} \end{cases} \tag{22}$$

$$q = \frac{r-r_{min}}{r_{max}-r_{min}} \tag{23}$$

Here $n, m, r_c, r_{min}, r_{max}$ are parameters. In order to use the Perriot screening method, specify e.g. for Y-Y

```
potpar(Y, Y):cutscr = perriot scr
```

Note the two strings! The second one must be exactly 'scr'.

All parameters relating to a screening method must be specified after the 'cutscr' setting! Otherwise these parameters are not set.

⋆ ⋆ ⋆

The ABOP potential is inadequate for small interatomic distances. To ensure a more correct description a repulsive potential $V_{\text{rep}}(r)$ — e.g. the ZBL potential — describing interactions at small distances should be used. The original potential presented above is then modified to

$$V(r) = (1-F(r))V_{\text{rep}}(r) + F(r)V_{\text{orig}}(r) \tag{24}$$

where $F(r)$ is the Fermi function

$$F(r) = 1/(1+e^{-b_f(r-r_f)}) \tag{25}$$

and b_f, r_f are parameters that need to be supplied.

⋆ ⋆ ⋆

All parameters that can be used with the ABOP potential are:

Parameter	Notes
Group 1	Basic ABOP parameters
D_0	
r_0	
β	
S	
γ	
c	
d	
h	
p	Defaults to $1/2$.
b_f	
r_f	
Group 2.1	Tersoff cutoff parameters
D	
R	Cutoff radius is $r_c = R + D$.
Group 2.2	Perriot cutoff parameters
n	
m	
r_c	Cutoff radius for screening method.
r_{min}	
r_{max}	Cutoff radius for cutoff function.
Group 3	
(V1) α_{ijk} and ω_{ijk}	
(V2) α_{ijk}	ω_{ijk} given by Eq. (12).
(V3) $2\mu_{ik}$	

Default value for each parameter in groups 1 and 2 is zero, except for p.

4.2 EAM potentials

EAM potentials can as of now not be fitted, but only be used as read-in potentials.

The total energy of a solid is in the EAM formalism

$$V = \frac{1}{2}\sum_i \sum_{j,j\neq i} V_2(r_{ij}) + \sum_i a_s F_s(\rho^a_{s,i}) + \sum_i a_p F_p(\rho^a_{p,i}) + \sum_i a_d F_d(\rho^a_{d,i}). \tag{26}$$

Here a_i is 1 if the band i is included, else it is 0. Also,

$$\rho^a_i = \sum_{j=1,j\neq i} \rho(r_{ij}), \tag{27}$$

where $\rho(r)$ is the atomic electron density at distance r from an atom. The program recognized the EAM versions displayed in table 1.

⋆ ⋆ ⋆

The EAM potentials are read in from files. The format of these is shown in tables 2-4.

⋆ ⋆ ⋆

Note: The `Nr` points (r, V_2) in the EAM files are read in as $r = 0, \ldots, (\texttt{Nr}-1)\texttt{dr}$. To avoid any problems, we must have `Nr` $\times$ `dr` $>$ `rcut`. For instance, use `Nr` $\times$ `dr` $=$ `rcut` $+10\times$ `dr`.

Table 1: Recognized EAM flavors.

`EAM-s`	EAM with only s embedding energy
`EAM-p`	EAM with only p embedding energy
`EAM-d`	EAM with only d embedding energy
`EAM-sp`	EAM with s and p embedding energies
`EAM-sd`	EAM with s and d embedding energies
`EAM-spd`	EAM with s, p, and d embedding energies

Table 2: EAM file format when a single embedding energy is used. Note: First and second lines are ignored.

```
Comment line
Z1 Z2 mass1 mass2 latpar1 latpar2 latname1 latname2
Nrho drho Nr dr rcut
```
(Nrhod points of F_s or F_p or F_d)
(Nr points of V_2)
(Nr points of ρ_s or ρ_p or ρ_d)

Table 3: EAM file format when two embedding energies are used. Example is for s and d embedding energies. Note: First and second lines are ignored.

```
Comment line
Z1 Z2 mass1 mass2 latpar1 latpar2 latname1 latname2
Nrhod drhod Nr dr rcut Nrhos drhos
```
(Nrhod points of F_d)
(Nr points of V_2)
(Nr points of ρ_d)
(Nrhos points of F_s)
(Nr points of ρ_s)

5 Read-in files

All specifications about the fitting procedure, physical properties, and calculations in general are given in read-in files: (i) file containing settings for the potentials, (ii) file containing

Table 4: EAM file format when three embedding energies are used. Note: First and second lines are ignored.

```
Comment line
Z1 Z2 mass1 mass2 latpar1 latpar2 latname1 latname2
Nrhod drhod Nr dr rcut Nrhos drhos Nrhop drhop
(Nrhod points of F_d)
(Nr points of V_2)
(Nr points of ρ_d)
(Nrhos points of F_s)
(Nr points of ρ_s)
(Nrhop points of F_p)
(Nr points of ρ_p)
```

physical properties, and (iii) file containing technical specifications about the calculations. The last file is optional.

The delimiters separating tokens in the read-in files are: tabulator, space, and the following characters: `:,()[]=`. The point is not a delimiter. Character strings which are not part of the input value for an option are simply ignored.

5.1 File: Elements and interactions information

An overview of which potentials that can be read in from tabulated data in (x, y) format and those which can be specified via parameters only is shown in Table 5.

Table 5: R = Can be read in from data file. A = Can be specified by parameters.

Potential	R	A
EAM-*	Yes	No
ABOP	No	Yes

⋆ ⋆ ⋆

Complete example:

```
# MUST BE FIRST:
# Number of elements:
nelem = 3

# MUST BE SECOND:
# Element names:
elem(1) = Fe
elem(2) = Cr
elem(3) = C
```

```
# Optional:
atomtype(Fe) = 1
atomtype(Cr) = 2
atomtype(C)  = 3


# Masses
mass(Fe) = 55.8470
mass(Cr) = 51.9961
mass(C ) = 12.0110


# Reference lattices:
lat(Cr, Cr) = skip   BCC  Calc. for this will be skipped, we will have Ecoh(Cr)=0.0.
lat(Fe, Fe) = BCC
lat(C,  C ) = GRA

# Supported ref. lattices:
# DIM1 (homomer), DIM2 (heteromer), SC, BCC, BCC-P, FCC, FCC-P,
# DIA, HCP, GRA, GRP (graphene)
# The ...-P versions refer to alternate structures with
# non-Cartesian primitive vectors.

a(Cr, Cr)  = 2.87
a(Fe, Fe)  = 2.87
##########################################################################
# Use accurate values for C, otherwise the graphite might explode due
# to massive pressure!!! Also, use small enough time step!!!
##########################################################################
a(C,  C )   = 1.46   rNN for graphite (GRA)      DIA: 3.55647765821
c(C,  C )   = 6.689
# Also possible:
# bpa(C, C) = ...
# cpa(C, C) = ...


# Interaction types:
iac(Fe, Fe) = ABOP
iac(Cr, Cr) = ABOP
iac(C,  C ) = ABOP
iac(Fe, Cr) = ABOP   symmetric
iac(Fe, C ) = ABOP   symmetric
iac(Cr, C ) = ABOP   symmetric


# Fit this interaction? The interaction must be analytical.
fit(Fe, Cr) = yes    symmetric


# Repulsive potentials (ZBL):

use_rep_core( Fe, Fe ) = yes
use_rep_core( Cr, Cr ) = yes
use_rep_core( C,  C  ) = yes
use_rep_core( Fe, Cr ) = yes
```

```
use_rep_core( Fe, C  ) = yes
use_rep_core( Cr, C  ) = yes

# Parameters for fixed interactions

potpar( Fe, Fe ):D0 = 1.5
potpar( Fe, Fe ):r0 = 2.29
potpar( Fe, Fe ):beta = 1.4
potpar( Fe, Fe ):S = 2.0693109
potpar( Fe, Fe ):gamma = 0.0115751
potpar( Fe, Fe ):c = 1.2898716
potpar( Fe, Fe ):d = 0.3413219
potpar( Fe, Fe ):h = -0.26
potpar( Fe, Fe ):bfermi = 10
potpar( Fe, Fe ):rfermi = 1

potpar( Fe, Fe ):cutscr = perriot scr    Must be before Perriot parameters (below)
potpar( Fe, Fe ):n = 5
potpar( Fe, Fe ):m = 48
potpar( Fe, Fe ):rcut = 3.3
potpar( Fe, Fe ):rmin = 3.0
potpar( Fe, Fe ):rmax = 3.3




potpar(Cr, Cr ):D0 = 4.04222081
potpar(Cr, Cr ):r0 = 2.13018547
potpar(Cr, Cr ):beta = 1.62158721
potpar(Cr, Cr ):S = 3.36793914
potpar(Cr, Cr ):gamma = 0.02388562
potpar(Cr, Cr ):c = 1.03288255
potpar(Cr, Cr ):d = 0.13813230
potpar(Cr, Cr ):h = -0.28569237
potpar(Cr, Cr ):cutscr = tersoff    Must be before Tersoff parameters (below)
potpar(Cr, Cr ):R = 3.2
potpar(Cr, Cr ):D = 0.20
potpar(Cr, Cr ):bfermi = 12
potpar(Cr, Cr ):rfermi = 1.7

potpar(C, C ):D0 = 6.0
potpar(C, C ):r0 = 1.39
potpar(C, C ):beta = 2.1
potpar(C, C ):S = 1.22
potpar(C, C ):gamma = 2.0813e-4
potpar(C, C ):c = 330.0
potpar(C, C ):d = 3.5
potpar(C, C ):h = 1.0
potpar(C, C ):cutscr = tersoff    Must be before Tersoff parameters (below)
potpar(C, C ):R = 1.85
potpar(C, C ):D = 0.15
potpar(C, C ):bfermi = 8
potpar(C, C ):rfermi = 0.6

potpar(Fe, C ):D0 = 3.95000634
potpar(Fe, C ):r0 = 1.53426579
```

```
potpar(Fe, C ):beta = 1.82109816
potpar(Fe, C ):S = 1.43035110
potpar(Fe, C ):gamma = 0.07485571
potpar(Fe, C ):c = 1.11674155
potpar(Fe, C ):d = 0.94663188
potpar(Fe, C ):h = -0.18665305
potpar(Fe, C ):cutscr = tersoff    Must be before Tersoff parameters (below)
potpar(Fe, C ):R = 2.6
potpar(Fe, C ):D = 0.20
potpar(Fe, C ):bfermi = 10
potpar(Fe, C ):rfermi = 1

potpar( Cr, C ):D0        =  2.77620074
potpar( Cr, C ):r0        =  1.81289285
potpar( Cr, C ):beta      =  2.00816371
potpar( Cr, C ):S         =  2.04637644
potpar( Cr, C ):gamma     =  0.00068830
potpar( Cr, C ):c         =  3.93353757
potpar( Cr, C ):d         =  0.17497204
potpar( Cr, C ):h         =  -0.17850001
potpar( Cr, C ):cutscr = tersoff    Must be before Tersoff parameters (below)
potpar( Cr, C ):R         =  2.95
potpar( Cr, C ):D         =  0.1
potpar( Cr, C  ):bfermi = 8
potpar( Cr, C  ):rfermi = 1.2


# Parameters for fittable interactions

     potpar( Fe, Cr ):D0 = 3.48049488
min: potpar( Fe, Cr ):D0 = 0.1
max: potpar( Fe, Cr ):D0 = 10.0

     potpar( Fe, Cr ):r0 = 2.16998952
min: potpar( Fe, Cr ):r0 = 1.0
max: potpar( Fe, Cr ):r0 = 5.0

     potpar( Fe, Cr ):beta = 1.75467567
min: potpar( Fe, Cr ):beta = 1.0
max: potpar( Fe, Cr ):beta = 5.0

     potpar( Fe, Cr ):S  = 2.28661503
min: potpar( Fe, Cr ):S  = 1.1
max: potpar( Fe, Cr ):S  = 5.0

     potpar( Fe, Cr ):gamma = 0.15766130
min: potpar( Fe, Cr ):gamma = 1e-5
max: potpar( Fe, Cr ):gamma = 1e5

     potpar( Fe, Cr ):c = 0.48531613
min: potpar( Fe, Cr ):c = -1e5
max: potpar( Fe, Cr ):c =  1e5

     potpar( Fe, Cr ):d  = 0.31427413
min: potpar( Fe, Cr ):d  =  -1e5
max: potpar( Fe, Cr ):d  =   1e5
```

```
     potpar( Fe, Cr ):h  = -0.69
min: potpar( Fe, Cr ):h  =  1.0
max: potpar( Fe, Cr ):h  =  1.0

     potpar( Fe, Cr ):cutscr = tersoff    Must be before Tersoff parameters (below)

     potpar( Fe, Cr ):R  = 3.10
min: potpar( Fe, Cr ):R  = 2.0
max: potpar( Fe, Cr ):R  = 2.0

     potpar( Fe, Cr ):D         = 0.15
min: potpar( Fe, Cr ):D  = 1.0
max: potpar( Fe, Cr ):D  = 1.0

     potpar( Fe, Cr ):bfermi = 10
min: potpar( Fe, Cr ):bfermi =  5
max: potpar( Fe, Cr ):bfermi = 15

     potpar( Fe, Cr ):rfermi = 1
min: potpar( Fe, Cr ):rfermi = 0.1
max: potpar( Fe, Cr ):rfermi = 5



# ABOP alpha and omega parameters

#################################################################
##
##  NOTE: There are NO ABOP alpha, omega, 2mu parameters used by default.
##        Only the specified ones are used.
##        Defaults: alpha=2mu=0.0 and omega=1.0 as constants.
##
##        If an omega parameter is specified it is taken as an
##        independent parameter (non-Brenner form), otherwise
##        it is constructed from alpha parameters --- if they are
##        specified --- as
##
##          "omega_ijk" = exp( alpha_ijk*(r0_ij - r0_ik) )
##
#################################################################


# Fixed:

abop_alpha( Fe, Fe, Fe ) = 0.0
abop_omega( Fe, Fe, Fe ) = 1.0

abop_alpha( Cr, Cr, Cr ) = 1.39662066
abop_omega( Cr, Cr, Cr ) = 1.0

abop_alpha( C, C, C ) = 0.0
abop_omega( C, C, C ) = 1.0

abop_alpha( Cr, Cr, C  )        = 0.8640643600
abop_alpha( Cr, C , Cr )        = -1.7520448300
```

```
abop_alpha( C, Cr, Cr )         = 0.6122158900
abop_alpha( C, C , Cr )         = 0
abop_alpha( C, Cr, C )         = 0
abop_alpha( Cr, C, C  )         = 0

abop_omega( Cr, Cr, C  )         = 1.6402877600
abop_omega( Cr, C , Cr )         = 0.2939996300
abop_omega( C, Cr, Cr )         = 0.4190507900
abop_omega( C, C , Cr )         = 1
abop_omega( C, Cr, C )         = 1
abop_omega( Cr, C, C  )         = 1



# Fittable:

     abop_alpha( Fe, Fe, Cr ) = 1.0
min: abop_alpha( Fe, Fe, Cr ) =  100.0
max: abop_alpha( Fe, Fe, Cr ) =  100.0

     abop_alpha( Fe, Cr, Fe ) = 1.0
min: abop_alpha( Fe, Cr, Fe ) =  100.0
max: abop_alpha( Fe, Cr, Fe ) =  100.0

     abop_alpha( Cr, Fe, Fe ) = 1.0
min: abop_alpha( Cr, Fe, Fe ) =  100.0
max: abop_alpha( Cr, Fe, Fe ) =  100.0

     abop_alpha( Cr, Fe, Cr ) = 1.0
min: abop_alpha( Cr, Fe, Cr ) =  100.0
max: abop_alpha( Cr, Fe, Cr ) =  100.0

     abop_alpha( Fe, Cr, Cr ) = 1.0
min: abop_alpha( Fe, Cr, Cr ) =  100.0
max: abop_alpha( Fe, Cr, Cr ) =  100.0

     abop_alpha( Cr, Cr, Fe ) = 1.0
min: abop_alpha( Cr, Cr, Fe ) =  100.0
max: abop_alpha( Cr, Cr, Fe ) =  100.0




     abop_omega( Fe, Fe, Cr ) = 1.0
min: abop_omega( Fe, Fe, Cr ) =  100.0
max: abop_omega( Fe, Fe, Cr ) =  100.0

     abop_omega( Fe, Cr, Fe ) = 1.0
min: abop_omega( Fe, Cr, Fe ) =  100.0
max: abop_omega( Fe, Cr, Fe ) =  100.0

     abop_omega( Cr, Fe, Fe ) = 1.0
min: abop_omega( Cr, Fe, Fe ) =  100.0
max: abop_omega( Cr, Fe, Fe ) =  100.0

     abop_omega( Cr, Fe, Cr ) = 1.0
```

```
min: abop_omega( Cr, Fe, Cr ) =  100.0
max: abop_omega( Cr, Fe, Cr ) =  100.0

     abop_omega( Fe, Cr, Cr ) = 1.0
min: abop_omega( Fe, Cr, Cr ) =  100.0
max: abop_omega( Fe, Cr, Cr ) =  100.0

     abop_omega( Cr, Cr, Fe ) = 1.0
min: abop_omega( Cr, Cr, Fe ) =  100.0
max: abop_omega( Cr, Cr, Fe ) =  100.0


# Not used:
#     abop_2mu( Fe, Cr ) = 1.0
#     abop_2mu( Cr, Fe ) = 1.0
```

5.2 File: Compounds information

In order to specify a physical property to be fitted, the keywords in example below must be used.

Keywords are grouped into sets than begin with the string LAT on a separate line. Properties in each set refer to the same structure/lattice/geometry.

All properties that are specified (=set) are activated as fitting targets.

Complete listing of options for a lattice: Note that `...` means that numerical values (single or severeal ones) are expected, and `***` means that a string is needed.

```
LAT                   <= starts readin of new lattice info

name          = ...   string
csystem       = ...   OPTIONAL, crystal system
file          = ...   LAT file, string
elements      = ...   element names (e.g. W H), strings
Ndesired      = ...   number of cells in direction a, three integers
Neven_desired = ...   0 for false, 1 for true, three integers
Nodd_desired  = ...   0 for false, 1 for true, three integers

# Lattice parameters
a             = ...
w_a           = ...      weight
b             = ...
u_b           = ...      uncertainty
c             = ...      default: weight: 1.0

# weight for any property: w_***
# uncertainty for any property: u_***

# Lattice parameter relationships:
```

```
bpa             = ...
cpa             = ...

# Dimer bond distance (only for dimers):
# r0            = ...

# Angles (radians):
angle_ab        = ...
angle_ac        = ...
angle_bc        = ...

# Atomic volume (cubic Angstroms):
Vatom           = ...

# This compound should be considered ground/reference state when calculating
# cohesive energies:
Ecoh_delta_refcomp = true
# If no cohesive energies used, then skip this setting.

# Change in cohesive energy relative to reference:
Ecoh_delta      = ...  (>0.0, i.e. more unstable than reference)

# Formation energy Ef in 'mixing energy' Emix = Ef/natoms form:
Emix            = ...

# Bulk modulus B (GPa):
B               = ...

# Pressure derivative of B:
Bp              = ...

# Elastic constants (all):
C11             = ...
C12             = ...
C13             = ...
C14             = ...
C15             = ...
C16             = ...
C22             = ...
C23             = ...
C24             = ...
C25             = ...
C26             = ...
C33             = ...
C34             = ...
C35             = ...
C36             = ...
C44             = ...
C45             = ...
C46             = ...
C55             = ...
C56             = ...
C66             = ...
# A SPGLIB call will determine space group and which Cij are correct.
# Wrong ones will be turned off.
```

```
# Options:
option: no_heating          Do not use temperatures > 0 K.
option: fixed_geometry      Do not allow atom positions in compounds to change.
option: quench_always       Always scale velocities to 0.


# Force handling:
frc_file      = ***  string
# => use atomic forces, get them from specified file
frc_use       = ***  boolean
# => use atomic forces
frc_use_w     = ***  boolean, use weights for force components (default)
# frc_use_u   = ***  boolean, use uncertainties for force components
#
# Format of forces file: Lines of
#   fx  fy  fz  wufx  wufy  wufz
# in eV/fs, where wufx, wufy, wufz is interpreted as weights/uncertainties,
# depending on which of frc_use_w/frc_use_u is set.
#

# Options that still exist (?) but are deprecated:
# Fmax          = ...    try to achieve this largest atomic force
#                        in relaxed compound
# Pmax          = ...    try to achieve this largest Cartesian pressure
#                        in relaxed compound
# displmax      = ...    try to achieve this largest atomic displacement
#                        in relaxed compound
# Rationale: Prefer parametrizations that minimize forces, pressures, and
# displacements.
# Note: Usually the MD relaxation achieves these goals already, and specifying
# these options may interfere with the fitting (see the merit function discussion).


# Compound-specific MD options, overrides any others specifed elsewhere:
# Complete list (example values):

mds_skint     = 1.0       # Angstrom
mds_seed      = 12345
mds_ndump     = 10        # dump info every ndump steps (if -dmdsprop option)

mds_tstart    = 0.0
mds_tend      = 2000.0
mds_dt        = 3.0
mds_max_dt    = 3.0

mds_Tstart    =    0.5  starting temperature T (K)

# Negative values or missing settings turn off control:
mds_btc_tau   =   10.0  Berendsen time constant for T control (fs)
mds_btc_T0    =    0.0  desired T (K)

# Negative values or missing settings turn off control:
mds_bpc_tau   = 80.0   Berendsen time constant for P control (fs)
mds_bpc_P0    =  0.0   desired P (GPa)
mds_bpc_scale = 50.0   scaling constant, usually on the order of bulk modulus (GPa)
```

```
# mds_quench_tstart = 2000
# mds_quench_rate   = 1.0   # quenching rate (K/fs), negative value => heating

# Some default values are always used, even if no settings here:
mds_error_T_gt      = 1e6   Fatal error if T gets over this limit (K).
mds_error_dt_lt     = 0.01  Fatal error if dt gets under this limit (fs).
mds_error_boxlen_gt = 1e4   Fatal error if any boxlen gets over this limit (Angstrom).
```

⋆ ⋆ ⋆

Ignoring elastic constants: The space group is determined by a call to the `spglib` library. The SPG information is used to determine which elastic constants C_{ij} that can be calculated. If elastic constants are not used, and especially if the compound *e.g.* contains defects that complicate the determination of the space group, use

```
csystem = any
```

This setting makes the program skip the determination of the space group for that compound. This skipping is also performed if the compound is non-periodic in any dimension.

⋆ ⋆ ⋆

If any setting relating to temperature and pressure control is mentioned it activates the corresponding control. The default values are negative. After readin the values of the time constants are checked; if any time constant value is negative the corresponding control is turned off.

⋆ ⋆ ⋆

Providing logical true or false: If a boolean value of logical true is to be input, use `yes`, `Yes`, `true`, `True`, `set`, or `Set`. Only the first letter is significant. The string `on` or a mixed version of lower- and upper-case also translates to logical true. In addition, the digit `1` also means logical true. Any other string or character evaluates to logical false.

⋆ ⋆ ⋆

Default behavior for weights and uncertainties: If neither weight nor uncertainty is specified for any property a default weight of 1 is used. The weights w_i are normalized: $\sum_i w_i = 1$, where the sum goes over properties with weights only, properties with uncertainties are not taken into account.

⋆ ⋆ ⋆

Example:

```
LAT                          <= triggers readin of new compound data

name     = dimer
file     = in/dimer.lat
elements = W H

r0 = 1.5                     Desired bond distance in dimer
w_r0 = 0.0001                weight

Ecoh     = -0.123
u_Ecoh   = 0.0068            uncertainty (w and u can be freely mixed
                             for different properties)

option: no_heating           Do not use temperatures > 0 K.



LAT                          <= triggers readin of new compound data

name     = bcc
file     = in/bcc.lat
elements = W
a        = 2.9
w_a      = 1.0
Ecoh_delta_refcomp = true
C11                = 80.1605326176
w_C11              = 20
C12                = 20.4233027208
w_C12              = 20
C44                = 25.072362326
w_C44              = 20



LAT                          <= triggers readin of new compound data

name     = octa
file     = in/octa.lat
elements = W H
Ndesired      = 2 2 2
Neven_desired = 1 1 1
Nodd_desired  = 0 0 0
a        = 2.9
Emix = 0.001

mds_seed  = 123
mds_tend    = 5000.0
mds_dt      = 3.0
mds_max_dt  = 3.0

mds_Tstart  =    0.5  starting temperature T (K)
mds_btc_tau =   10.0  Berendsen time constant for T control (fs)
mds_btc_T0  =    0.0  desired T (K)
```

```
mds_bpc_tau   = 80.0  Berendsen time constant for P control (fs)
mds_bpc_P0    =  0.0  desired P (GPa)
mds_bpc_scale = 50.0  scaling constant, usually on the order of bulk modulus (GPa)
```

5.3 LAT files

5.3.1 Basic structure

Format of LAT file:

```
Comment                       Arbitrary comment.
  S                           Overall scaling constant.
a1 a2 a3  optional-string     Components of the first primitive vector U1.
b1 b2 b3  optional-string     Components of the second primitive vector U2.
c1 c2 c3  optional-string     Components of the third primitive vector U3.
  format                      internal or direct
  Nbasis                      number of basis vectors
E1  B11  B12  B13 constr1     Element name and vector components of first basis atom.
E2  B21  B22  B23 constr2     Element name and vector components of second basis atom.
...
```

The real primitive vectors are

$$\mathbf{v}_i = S\mathbf{u}_i \tag{28}$$

The strings E_i — e.g. Cr, H, W — specify the element/species of the basis atom.

If `optional-string` is present and is `pbc`, then the corresponding direction is considered periodic, i.e. it is a true primitive vector in an infinite lattice. Other strings are ignored.

If the `format` keyword is `scaled` or `Scaled` or letter `s` or `S` then *e.g.* the j :th basis vector is

$$\mathbf{b}_j = SB_{j1}\mathbf{e}_x + SB_{j2}\mathbf{e}_y + SB_{j3}\mathbf{e}_z = \sum_i SB_{ji}\mathbf{e}_i \tag{29}$$

This is equivalent to **Cartesian** in VASP.

If instead the `format` keyword is `internal` or `Internal` or letter `i` or `I` then the basis vector is

$$\mathbf{b}_j = SB_{j1}\mathbf{u}_x + SB_{j2}\mathbf{u}_y + SB_{j3}\mathbf{u}_z = \sum_i SB_{ji}\mathbf{u}_i \tag{30}$$

This is equivalent to **Direct** in VASP.

5.3.2 Constraints

The `constr1`, `constr2`, *etc.* are atomic constraint strings (each atom can be given constraints). There are three possible constraint types:

1. Fix atom: `fix`
2. Constrain atom to move in one direction only: `freedir u1 u2 u3`, where the `u`i are coordinates. Example: `freedir 0 0 1` allows motion in z direction only.
3. Constrain atom to move in a plane only: `freeplane u1 u2 u3`, or `freeplanevecs v1 v2 v3 w1 w2 w3`, where the `u`i are coordinates of the plane's normal vector, and the `v`i, `w`i are coordinates of two vectors lying in the plane.

All direction vectors are normalized automatically. Only one option is valid for any atom.

5.3.3 Specifying the origin

An explicit origin can be specified on the line after the last basis atom. The first string has to start with lower- or upper-case O, then three coordinates must be given. This origin will be used when the simulation cell is created.

5.3.4 Examples

Example 1: Fe-Y dimer in vacuum, non-periodic boundaries:

```
Fe-Y dimer
  10.0   <= scaling constant
1.0  0.0  0.0    primitive vector U1
0.0  1.0  0.0    primitive vector U2
0.0  0.0  1.0    primitive vector U3
  Scaled
  2
Y   0.0000000000   0.0000000000   0.0000000000
Fe  0.2000000000   0.0000000000   0.0000000000 freedir 1 0 0
origin: 0.0 0.0 0.0
```

Example 2: CrC in the NaCl crystal form:

```
#
  4.0741512214
0.0  0.5  0.5   pbc    primitive vector U1
0.5  0.0  0.5   pbc    primitive vector U2
0.5  0.5  0.0   pbc    primitive vector U3
  Internal
  2
C   0.0  0.0  0.0
Cr  0.5  0.5  0.5
```

Scripts are provided to convert between POSCAR, XYZ and LAT formats.

⋆ ⋆ ⋆

A simulation box with box lenghts $L_i = N_i|\mathbf{v}_i|$ is constructed from a compound with primitive vectors $\mathbf{v}_i = S\mathbf{u}_i$, so that all box lenghts are more than twice the largest cutoff radius for the elements occuring in the compound:

$$L_i = N_i v_i \geq 2r_c \tag{31}$$

Here N_i is initially the corresponding value in the `Ndesired` setting, if specified.

⋆ ⋆ ⋆

Predicted lattice parameter $\widetilde{a}$ is calculated as

$$\widetilde{a} = (\widetilde{L}_1/L_1)a, \tag{32}$$

where a is the read-in lattice parameter in the file listing the compounds, L_1 is the initial length of the simulation box in the $\mathbf{v}_1$ direction, and $\widetilde{L}_1$ is the relaxed length of the simulation box in the same direction.

Q: Why are read-in values for a, b, c not equal to v_1, v_2, v_3 ? Why are predicted values for a, b, c not equal to relaxed values of v_1, v_2, v_3 ?

A: Primitive vectors for compounds may be given in many different ways. The current scheme avoids keeping track of them all, lightening the work burden for the user and minimizes additional coding.

The tradeoff is that for e.g. cubic phases $a = b = c$, but now a is taken from the change in v_1, changes in v_2, v_3 are not taken into account.

⋆ ⋆ ⋆

Note: There must always be at least one basis vector. The following example for bcc Fe should be helpful:

```
Comment: BCC Fe with primitive vectors.
-0.25   0.25    0.25    pbc
 0.25  -0.25    0.25    pbc
 0.25   0.25   -0.25    pbc
  Internal
  1
Fe   0.0   0.0   0.0
```

Using a conventional cubic cell, the lattice is

```
Comment: BCC Fe.
 1.0  0.0  0.0   pbc
 0.0  1.0  0.0   pbc
 0.0  0.0  1.0   pbc
  Internal
  2
Fe   0.0   0.0   0.0
Fe   0.25  0.25  0.25
```

5.4 Optional file: Technical specifications

```
# ------------------------------------
# Calculating properties of compounds
# ------------------------------------

# Bulk modulus
prop:BM_rel_sys = yes      relax a strained frame
prop:BM_fmin    = -0.01
prop:BM_fmax    =  0.01
prop:BM_Nf      =  10      number of points to use for (V,E) curve
prop:BM_ef      =  1e-10   error ratio for each energy valye

# Elastic constants
prop:C_rel_sys  = yes
prop:C_fmin     = -0.01
prop:C_fmax     =  0.01
prop:C_Nf       =  10
prop:C_ef       =  1e-10


# ------------------------------------
# Fitting properties of compounds
# ------------------------------------
prop:fitmet       = LM    options: CG, PM, GN, LM, DL, SM, DE, PS, BC, GS, or SA
prop:nitermin     = 5     do at least this many iterations
prop:nitermax     = 100   do a maximum of this many iterations
prop:niterrestart = 20    restart every 20th iteration

#### Negative values means that it will not be used when testing for convergence:
prop:functolabs =  1e-5   convergence when absolute merit function value is less
                          than this
prop:functolrel = -1e-5   ... change in merit function value ...
prop:gradtolabs =  1e-5   ... absolute gradient value ...
prop:steptolabs =  1e-5   ... absolute step size ...
prop:steptolrel = -1e-5   ... change in step size ...

prop:dogleg_radius    = 0.2    initial trust region radius
prop:dogleg_minradius = 1e-5   exit when radius gets this low
prop:barrier_scale    = 0.0    scaling constant for barrier penalty function,
                               to keep parameter values inside min/max interval
prop:simann_delta_rel = 0.2    initial displacements in coordinate directions
```

```
prop:use_data_scales  = false  use/do not use scaled values in the merit function


# -----------------------------------
# General MDS settings
# -----------------------------------
prop:mds_skint = 1.0       Angstrom
prop:mds_seed  = 12345
prop:mds_ndump = 10        dump info every ndump steps (if -dmdsprop option)

prop:mds_tstart  = 0.0
prop:mds_tend    = 2000.0
prop:mds_dt      = 0.5
prop:mds_max_dt  = 1.0

prop:mds_Tstart  = 300.0   starting temperature T (K)

# Negative values or missing settings turn off control:
prop:mds_btc_tau =  20     Berendsen time constant for T control (fs)
prop:mds_btc_T0  =  0      desired T (K)

# Negative values or missing settings turn off control:
prop:mds_bpc_tau   = 100   Berendsen time constant for P control (fs)
prop:mds_bpc_P0    = 0     desired P (GPa)
prop:mds_bpc_scale = 100   scaling constant, usually on the order of bulk modulus
                           (GPa)

prop:mds_quench_tstart = 100
prop:mds_quench_rate   = 1.0   quenching rate (K/fs), negative value => heating


# -----------------------------------
# MDS settings for reference compounds
# -----------------------------------

# Use this to copy MDS settings for reference compounds from the general settings
# specified above:
# prop:ref:mds = prop:mds

prop:ref:mds_skint = 1.0       Angstrom
prop:ref:mds_seed  = 12345
prop:ref:mds_ndump = 10        dump info every ndump steps (if -dmdsprop option)

prop:ref:mds_tstart  = 0.0
prop:ref:mds_tend    = 3000.0
prop:ref:mds_dt      = 5.0
prop:ref:mds_max_dt  = 5.0

prop:ref:mds_Tstart  = 1.0   starting temperature T (K)

# Negative values or missing settings turn off control:
prop:ref:mds_btc_tau =  20   Berendsen time constant for T control (fs)
prop:ref:mds_btc_T0  =  0    desired T (K)

# Negative values or missing settings turn off control:
prop:ref:mds_bpc_tau   = 80  Berendsen time constant for P control (fs)
```

```
prop:ref:mds_bpc_P0    = 0   desired P (GPa)
prop:ref:mds_bpc_scale = 80  scaling constant, usually on the order of bulk modulus
                             (GPa)

prop:ref:mds_quench_tstart = 100
prop:ref:mds_quench_rate   = 1.0   # quenching rate (K/fs), negative value => heating



# ##############################################################
# Fitting potential(s)
# ##############################################################

pot:fitmet       = DL    options: CG, PM, GN, LM, DL, SM, DE, PS, BC, GS, or SA
pot:nitermin     = 5
pot:nitermax     = 100
pot:niterrestart = 20    restart every 10th iteration

#### Negative values means that it will not be used when testing for convergence:
pot:functolabs =  1e-5
pot:functolrel = -1e-5
pot:gradtolabs =  1e-5
pot:steptolabs =  1e-5
pot:steptolrel = -1e-5

pot:dogleg_radius    = 0.2
pot:dogleg_minradius = 1e-5
pot:barrier_scale    = 0.0     barrier value
pot:simann_delta_rel = 0.2
pot:use_data_scales  = false   scaling option


  INFO:

# Gradient-based fitting methods:
# -------------------------------
# CG = conjugate gradients (ls)
# PM = Powell's method (ls)
# GN = Gauss-Newton (mi)
# LM = Levenberg-Marquardt (mi)
# DL = Powell's dog-leg method (mi) (usually most robust)
# SA = simulated annealing

#
# Population-based fitting methods:
# ---------------------------------
# SM = simplex method
# DE = differential evolution
# PS = particle swarm method
# BC = bee colony method
# GS = gravitational search method

# mi: uses matrix inversion
# ls: uses line-search, usually implies slow fitting
```

The keyword `prop` is relevant to the calculations in which lattice parameter, cohesive energy, bulk modulus and other properties are calculated and/or fitted for the read-in structures using a given potential parameter set.

On the other hand, the keyword `pot` is relevant to the potential fitting itself, *i.e.* the evolution of the potential parameters.

6 Calculation of physical properties

Temperature T is obtained from the equipartition theorem:

$$\frac{3}{2}NkT = \sum_{i=1}^{N}\frac{1}{2}m_i v_i^2 \tag{33}$$

The pressure is obtained from the general stress tensor σ_{ab},

$$\sigma_{ab} = \frac{1}{2}\sum_i\sum_j F_{ij,a}r_{ij,b} \tag{34}$$

where $\mathbf{F}_{ij} \propto \mathbf{r}_{ij}$ and $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$. The Cartesian pressure components are $P_x = \sigma_{xx}$, ..., and $P = (P_x + P_y + P_z)/3$.

Temperature and pressure is controlled according to the method of Berendsen et al. [2].

7 Calculation of fittable properties

Predicted value of a property with the read-in value P is written $\widetilde{P}$. In order to simplify notation in this section the tilde will be omitted, with the understanding that all values used are based on the relaxed compounds. If a read-in value is used, it is written P^{ri}.

Predicted lattice parameter $\widetilde{a}$ (Ångströms) is calculated as

$$\widetilde{a} = (\widetilde{L_1}/L_1) \times a, \tag{35}$$

where a is the read-in lattice parameter in the compounds file (the file listing all the compounds to be used), L_1 is the initial length of the simulation box in the $\mathbf{v}_1$ direction, and $\widetilde{L_1}$ is the relaxed length of the simulation box in the same direction.

L_i are determined from $\mathbf{v}_i$ as $L_i =$`Ndesired`$(i) \times v_i$, is the `Ndesired` option is used. Otherwise N_i is determined from the requirement $L_i = N_i v_i \geq 2max(r_c)$, where $max(r_c)$ is the largest cutoff radius between species present in the compound. In this calculations the

options `Neven_desired` and `Nodd_desired` are considered, if given in the compounds file.

Predicted lattice parameters $\widetilde{b}, \widetilde{c}$ are calculated in a similar way:

$$\widetilde{b} = (\widetilde{L_2}/L_2) \times b \quad (36)$$
$$\widetilde{c} = (\widetilde{L_3}/L_3) \times c \quad (37)$$
$$(38)$$

Ratios of lattice parameters:

$$\widetilde{b/a} = \frac{\widetilde{L_2}/L_2}{\widetilde{L_1}/L_1} \times b/a \quad (39)$$
$$\widetilde{c/a} = \frac{\widetilde{L_3}/L_3}{\widetilde{L_1}/L_1} \times c/a \quad (40)$$

Dimer bond length (Ångströms)::

$$\widetilde{r_o} = |\widetilde{\mathbf{r}_2} - \widetilde{\mathbf{r}_1}| \quad (41)$$

Angle $\widetilde{\gamma}$ (in degrees), i.e. angle between primitive vectors $\widetilde{\mathbf{a}} = \widetilde{\mathbf{v}_1}$ and $\widetilde{\mathbf{b}} = \widetilde{\mathbf{v}_2}$:

$$\widetilde{\gamma} = = \arccos\left(\frac{\widetilde{\mathbf{L}_1} \cdot \widetilde{\mathbf{L}_2}}{\widetilde{L_1}\widetilde{L_2}}\right) \cdot 1/(2\pi) \cdot 360 \quad (42)$$

Angle $\widetilde{\beta}$ (in degrees), i.e. angle between primitive vectors $\widetilde{\mathbf{a}} = \widetilde{\mathbf{v}_1}$ and $\widetilde{\mathbf{c}} = \widetilde{\mathbf{v}_3}$:

$$\widetilde{\beta} = = \arccos\left(\frac{\widetilde{\mathbf{L}_1} \cdot \widetilde{\mathbf{L}_3}}{\widetilde{L_1}\widetilde{L_3}}\right) \cdot 1/(2\pi) \cdot 360 \quad (43)$$

Angle $\widetilde{\alpha}$ (in degrees), i.e. angle between primitive vectors $\widetilde{\mathbf{b}} = \widetilde{\mathbf{v}_2}$ and $\widetilde{\mathbf{c}} = \widetilde{\mathbf{v}_3}$:

$$\widetilde{\alpha} = = \arccos\left(\frac{\widetilde{\mathbf{L}_2} \cdot \widetilde{\mathbf{L}_3}}{\widetilde{L_2}\widetilde{L_3}}\right) \cdot 1/(2\pi) \cdot 360 \quad (44)$$

Atomic volume (cubic Ångströms):

$$\widetilde{V_a} = \widetilde{V_{tot}}/N \tag{45}$$

where N is the number of atoms in the cell.

Cohesive energy (eVs):

$$\widetilde{E_{\rm coh}} = \widetilde{E_P}/N < 0 \tag{46}$$
$$\widetilde{\Delta E_{\rm coh}} = \widetilde{E_{\rm coh}} - \widetilde{E^{\rm ref}_{\rm coh}} \tag{47}$$

Here N is the number of atoms in the compound, $\widetilde{E_P}$ is the potential energy of the compound, and $\widetilde{E^{\rm ref}_{\rm coh}}$ is the cohesive energy of the reference compound, if used.

Formation energies are read in and calculated as "mixing energies", which is simply the formation energy normalized with the number of atoms in the cell. Hence the mixing energy (eVs) is:

$$\widetilde{E}_{\rm mix} = \widetilde{E_f}/N \tag{48}$$
$$\widetilde{E_f} = \widetilde{E_p} - \sum_s N(s)\widetilde{E_{\rm coh}(s)} \tag{49}$$

Here $N(s)$ is the number of atoms of species s in the compound, and $\widetilde{E_{\rm coh}(s)}$ is the cohesive energy of the ground state of species s. Parametrization for species s must be given. For instance, if s is C, then ground state might be GRA (graphite).

Bulk modulus B (GPa) and its pressure derivative $B'(P)$ is calculated from $(V(\varepsilon), E_p(\varepsilon))$ data using the Birch-Murnaghan equation of state [3, 6]

$$\begin{aligned} E &= E_0 + \frac{9}{16}V_0B_0\left(\left(\frac{V_0}{V}\right)^{2/3} - 1\right)^2 \\ &\quad \times \left[\left(\left(\frac{V_0}{V}\right)^{2/3} - 1\right))B_0' + 6 - 4\left(\frac{V_0}{V}\right)^{2/3}\right] \end{aligned} \tag{50}$$

The volume is $V(\varepsilon) = (1+\varepsilon)V_0 = (1+f)^3L_1(0)L_2(0)L_3(0)$, with $f \in [-f_{\max}, f_{\max}]$, usually $f_{\max} \sim 0.01$.

Each box — for a given value of f — may be allowed to relax (`prop:BM_rel_sys = yes`), or not (`prop:BM_rel_sys = no`).

Elastic constants C_{ij} (GPa) are calculated using volume-non-conserving strains ε_{ij}. Given a strain matrix

$$[\varepsilon_{ij}] = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{bmatrix} \tag{51}$$

the transformation matrix is

$$F = 1_3 + [\varepsilon_{ij}] = \begin{bmatrix} 1+\varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & 1+\varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & 1+\varepsilon_{33} \end{bmatrix} \tag{52}$$

Any ε_{ij} can take only three values: $-\varepsilon, 0, \varepsilon$. The exact structure of F depends on which elastic constants are being calculated.

The atomic positions are in essence transformed as

$$r' = Fr, \tag{53}$$

where r is a column vector containing the Cartesian coordinates x, y, z. From the calculations the data points $(\varepsilon, E_p(\varepsilon))$ are obtained. A second degree polynomial in strains is fitted to this data to obtain the elastic constants.

Each box — for a given value of ε — may be allowed to relax (`prop:C_rel_sys = yes`), or not (`prop:C_rel_sys = no`).

$\star \quad \star \quad \star$

The physical properties are not independent: e.g. a and V_a are dependent for a cubic lattice, since $V_a \propto a^3$. Specifying properties which are dependent on each other as fitting targets will most likely give rise to singular matrices when using a fitting method relying on matrix inversion, due to presence of zero-valued rows/columns.

8 Data fitting

Given compounds and their desired properties, as well as technical specifications on how to compute them, the program tries to obtain a best fit of potential parameters. The best fit is achieved by minimizing the merit/cost function

$$\chi^2 = \frac{1}{2}\sum_i f_i^2 \tag{54}$$

$$f_i = t_i \frac{\widetilde{Y}_i - Y_i}{s_i} \tag{55}$$

Here

- the $\widetilde{Y}_i$ is the predicted value of property i which has the read-in (desired) value of Y_i;
- the t_i is either a weight ($t_i = w_i$ if weight is used) or an inverse uncertainty ($t_i = 1/u_i$ if uncertainty is used); and
- the $s_i = 1$ if scales are not used (`use_data_scales = false` in the specifications file), and $s_i = Y_i$ if scales are used.

Defaults:

- Weights are used, with $w_i = 1$ for all i. Specifying weight or uncertainty for any property i overrides any default.
- Scales are used. If a property value is positive and less than machine accuracy then it is replaced by the machine accuracy. By using scaled properties all are on an equal footing.

Note: The weights w_i are normalized: $\sum_i w_i = 1$, so that properties using uncertainties have zero weight in this normalization sum.

⋆ ⋆ ⋆

Assume the predicted value is $\widetilde{Y}_i = Y_i + \delta Y_i$, then $f_i = t_i(\widetilde{Y}_i - Y_i)/s_i = t_i \delta Y_i / Y_i$ if scales are used, and $f_i = t_i \delta Y_i$ otherwise.

Suppose the relative deviation $\delta Y_i/Y_i \approx \delta Y_j/Y_j \approx 10\%$ for two properties i, j, e.g. lattice parameter, formation energy. If they are equally important to the fitting then $t_i = t_j$ and $f_i \approx f_j$ and these properties give equal contributions to the merit function.

Now consider the general case that the read-in values are vastly different, e.g. $Y_i = 1$ and $Y_j = 10^{-3}$. Using the former choice of scaling there is no dependence on the absolute values of the read-in property values, and we have no problems. If scales are not used, then the contributions to the merit function are $f_i = t_i$ and $f_j = t_j \times 10^{-3} = t_i 10^{-3}$ and there is a considerable relative difference, although both properties were assumed to be equally important. In this case property i will have a larger impact on the fitting process (assuming $J_{ik} \equiv \partial f_i/\partial x_k \approx J_{jk}$).

⋆ ⋆ ⋆

Consider now the use of any of the deprecated options, e.g. `Fmax`. This option may be used to try to guide the fitting process to accept parametrizations giving small forces of relaxed compounds. Assume $t_i = 1$ and that scaling is used, so $s_i = Y_i$.

(1) Case $\widetilde{Y}_i \gg Y_i$: Now

$$f_i = \frac{\widetilde{Y}_i - Y_i}{Y_i} = \widetilde{Y}_i/Y_i - 1 \sim \widetilde{Y}_i/Y_i \gg 1 \tag{56}$$

E.g. `Fmax`$= 10^{-10}$ and the predicted value is e.g. 10^{-5}:

$$f_i \sim 10^{-5}/10^{-10} = 10^5 \tag{57}$$

i.e. probably much larger than for any of the conventional properties. In a gradient-based search the moves in parameter space will tend to focus on minimizing the forces, paying less attention to other properties, e.g. differences in cohesive energies.

(2) Case $\widetilde{Y}_i \ll Y_i$: Now

$$f_i = \frac{\widetilde{Y}_i - Y_i}{Y_i} = \widetilde{Y}_i/Y_i - 1 \sim 1 \tag{58}$$

E.g. `Fmax`$= 10^{-3}$ and the predicted value is e.g. 10^{-5}. Only in this case will the contribution be manageable.

In the general case the predicted forces could be very small. And in the general case we also want force values close to zero (read-in force values). Involving very small values and especially their ratios in the calculations is not desirable.

Conclusion: Do not use `Fmax`, `Pmax` or `displmax`. Rely on the MD relaxation instead, to provide zero forces and zero pressure. In fact, the MD run can be made to scale the pressure to any desired value (`mds_bpc_P0`), so it's better to use that one.

$\star \quad \star \quad \star$

The **gradient** of the merit function is

$$g_j \;=\; \frac{\partial \chi^2}{\partial x_j} = \sum_i f_i \frac{\partial f_i}{x_j} \equiv \sum_i f_i J_{ij} = \sum_i J^T_{ji} f_i = -h_j \tag{59}$$

where h_j are components of the **antigradient**, and

$$J_{ij} \;=\; \frac{\partial f_i}{\partial x_j} = \frac{t_i}{s_i}\frac{\partial \widetilde{Y}_i}{\partial x_j} \tag{60}$$

9 Convergence conditions

Convergence criteria in the technical specifications file are of the form (`prop`, for fitting compound properties):

```
prop:functolabs =  1e-5   convergence when absolute merit function value is less
                          than this
prop:functolrel = -1e-5   ... change in merit function value ...
prop:gradtolabs =  1e-5   ... absolute gradient value ...
prop:steptolabs =  1e-5   ... absolute step size ...
prop:steptolrel = -1e-5   ... change in step size ...
```

Note: A **negative** value means that criterion is **not used**.

- `functolabs` refers to $\min(|\chi^2|)$
- `functolrel` refers to $\min(|\Delta\chi^2|/|\chi^2|)$, where Δ signifies the change (in the merit function in this case) during the last step.
- `gradtolabs` refers to $\min(|\nabla_x\chi^2|)$, where $\nabla_x\chi^2$ is the gradient of the merit function with respect to the parameters.
- `steptolabs` refers to $\min(|\mathbf{h}|)$, where $\mathbf{h}$ is the last step taken in parameter space.
- `steptolrel` refers to $\min(|\Delta\mathbf{h}|/|\mathbf{h}|)$

10 Parameters

The parameters are $x_1, x_2, \ldots, x_M$.

Parameter limits $x_i^{\min} = x_i^{\max} = 0$ means that the parameter is **unconstrained** and can vary freely in the interval $(-\infty, \infty)$.

The limits $x_i^{\min} = x_i^{\max} \neq 0$ means that the parameter is **fixed**. The default for any parameter is $x_i^{\min} = x_i^{\max} = 1$.

The limits $x_i^{\min} < x_i^{\max}$ means that the parameter is **constrained** in the given interval.

11 Constraint method

In order to forcibly constrain parameter values a barrier penalty function can be used. The option `barrier_scale`$= \mu$ with a value > 0 switches on the penalty function U:

$$U = -\sum_i \mu \ln\left\{\frac{1}{s_i}\left(\left[\frac{1}{2}(x_i^{\min} - x_i^{\max})\right]^2 - \left[x_i - \frac{1}{2}(x_i^{\min} + x_i^{\max})\right]^2\right)\right\} < 0 \qquad (61)$$

The total cost function is then

$$\chi^2_{\text{tot}} = \chi^2 + U \qquad (62)$$

A complication with this scheme is that with small weights w_i one may get $\chi^2_{\text{tot}} = \chi^2 + U < 0$. This may interfere with the convergence criteria $\min(|\chi^2|)$, since $\chi^2_{\text{tot}} < 0 < \min(|\chi^2|)$ as soon as $|U|$ starts to dominate over $|\chi^2|$.

With $\mu = 0$ the program still tries to stop parameter values from going outside their limits, but a penalty function is not used.

References

[1] G. C. Abell. Empirical chemical pseudopotential theory of molecular and metallic bonding. *Phys. Rev. B*, 31(10):6184–6196, 1985.

[2] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. Di Nola, and J. R. Haak. Molecular dynamics with coupling to an external bath. *J. Chem. Phys.*, 81(8):3684–3690, 1984.

[3] F. Birch. Finite elastic strain of cubic crystals. *Phys. Rev.*, 71:809–824, 1947.

[4] Donald W. Brenner. Empirical potential for hydrocarbons for use in simulating the chemical vapor deposition of diamond films. *Phys. Rev. B*, 42(15):9458, 1990.

[5] N. Juslin, J. Nord, K. O. E. Henriksson, P. Traskelin, E. Salonen, K. Nordlund, P. Erhart, and K. Albe. Analytical interatomic potential for modelling non-equilibrium processes in the W-C-H system. *J. Appl. Phys.*, 98:123520, 2005.

[6] F. D. Murnaghan. The compressibility of media under extreme pressures. *PNAS*, 30(9):244–247, 1944.

[7] R. Perriot, X. Gu, Y. Lin, V. V. Zhakhovsky, and I. I. Oleynik. Screened environment-dependent reactive empirical bond-order potential for atomistic simulations of carbon materials. *Phys. Rev. B*, 88:064101, 2013.

[8] J. Tersoff. New empirical model for the structural properties of silicon. *Phys. Rev. Lett.*, 56:632, 1986.