MD Tian Xia

Manual

Prof. Dr. Daniel J. Auerbach, Svenja M. Janke,
Marvin Kammler and Dr. Alexander Kandratsenka

Institute of Physical Chemistry
Georg-August Universität Göttingen
Max Planck Institute for Biophysical Chemistry, Göttingen.

Contents

1 Usage of mdtian

This chapter has been copied directly from Svenja M. Janke's PhD-Thesis [1].

MD_tian is a program for molecular dynamics calculations that Prof. Dr. Dan Auerbach, Dr. Alexander Kandratsenka, Svenja M. Janke and Marvin Kammler wrote in FORTRAN to deal with our molecular dynamics simulations and fitting the EMT to DFT input data. The program's name derives from the Chinese expression 天下 (Tiānxià), meaning 'under one sky', thereby underscoring our aim to produce a program which includes all subprograms needed for our EMT-related purposes. ' 天' and ' 下' (symbolized by the underscore) are deliberately wrong way round in the program's name, because symbols on temples in China were commonly written from right to left. At the moment, the MDtianxia program calculates forces for the MD-simulations using the effective medium theory, but in future, the program shall be expanded to include also Lennard-Jones Potentials or other potential forms. The positions of the all species in the system are subjected to periodic boundary conditions.

The program takes three types of input files: a file that determines the geometry that should be used for the calculations, a file which includes the EMT parameters for each species used during the MD calculations and a control file. The control file, invoked by different flags, determines whether an MD-simulation or a fit is run and with which properties, the number of species and what kind of input data is used.

1.1 The EMT-parameter file

Apart for the suffix '.nml' the name of a parameter file can be chosen arbitrarily. Each chemical element has its separate parameter file. So far, we have chosen the following convention for naming the files: First, the potential to which the parameters refer, followed by the number of the fit from which the parameters stem and then, separated by

```
Slab EMT Parameters              # Comment line
Name= Au                         # name of the species
  eta2=             3.29722801   # List of parameters
    n0=             0.04184152
    E0=            -3.80000000
lambda=             4.18153000
    V0=             0.34752943
 kappa=             3.24943176
    s0=             1.64174000
```

Figure 1.1: The parameter file for MD_tian.

an underscore, the element's name. So 'emt515_Au.nml' contains the parameters for the EMT potential for gold and stems from the fit No. 515. The file begins with a short comment line (see Fig.1.1). In the following lines, the name of the element is given followed by the list of the parameters. The order of lines in the file is fixed and must not be changed, otherwise the parameters will be wrongly assigned within the program.

1.2 POSCAR

The POSCAR-file is the same POSCAR-file as used in VASP [2, 3, 4, 5, 6, 7]. It contains the cell-geometry, the number of atoms and atomic species and their position for the problem under consideration. The first line (see Fig. 1.2) is reserved for a comment to give information about the calculation the POSCAR file is used in. The second line contains the lattice constant and the next three lines contain the cell matrix that contains the cell vectors in x- (line 3), y- (line 4) and z-direction (line 5). The lattice constant is always multiplied onto the cell matrix, so for cells in which the lattice constant is just a factor, it should be written in the second line. Due to the periodic boundary conditions VASP employs when one moves from calculating bulk-properties to surface properties one needs to separate the periodic images from one another in one direction to create a surface, usually done in z-direction. The amount of space by which the slabs are separated is called the vacuum distance.

```
fcc Au          # Comment line to describe the file
 4.0            # lattice constant
 0.5 0.5 0.0    # cell-vector in x-direction
 0.0 0.5 0.5    # cell-vector in y-direction
 0.5 0.0 0.5    # cell-vector in z-direction
   1            # number of atoms in the cell
cartesian       # coordinate system
0 0 0           # position of the atoms
```

Figure 1.2: The POSCAR file. Example for a $1 \times 1 \times 1$ cell for a bulk-calculation.

Line 6 contains the number of atoms in the cell. If there is more than one species in the cell, the first number denotes the number of atoms of the atomic species that comes first in the POTCAR-file, followed by the number of atoms of the second species and so on. The following line (line 7) describes in which coordinate system the atomic positions will be given. The choice here is between Cartesian coordinates and direct coordinates. Direct coordinates give the atomic positions between 0.0 and 1.0 and can be obtained by multiplying the Cartesian coordinates with the inverse of the cell matrix. From line 8 on, the positions of all atoms are given. Here, the atomic species must follow in the same order as specified in line 6. This means if one writes a POSCAR-file containing one Au atom and two H atoms and writes in line 6 '1 2', then one has to write the position of the Au atom first, followed by the positions of the two hydrogen atoms.

The POSCAR-file also determines the cell dimensions for a cell containing a surface. They are denoted by $n_x \times n_y \times n_z$ where n_x is the number of atoms in x-direction and n_y and n_z the number of atoms in y- and z-direction, respectively. So, in the example in Fig. 1.2, the cell dimensions are those of a $1 \times 1 \times 1$ cell and a $2 \times 2 \times 4$ cell would have four atoms in at its surface and four atoms in z-direction, that is four layers.

1.3 The Configuration files

There are two types of configuration files the program can read in. Which type of file should be read in is specified with the *conf*-flag in the control file. The *conf* flag can

be employed as shown in Tab. 1.1. *address* signifies the absolute path to the file of the name 'file'; 'POSCAR' and 'fit' both need a POSCAR-file as input (see Sec. 1.2). For the options 'geo' and 'conf', the program's own binary output mxt_conf*n*.bin-files (where n can be any number between 99999999 and 00000000) are read in. They contain positions, velocities, accelerations and densities for all atoms. These files make it possible to restart an interrupted calculation, e.g. if one wants to start MD-simulations from an already equilibrated slab. Species that are already treated in the mxt_conf*n*.bin-file need not be specified in the control file; another species may however be added. That means that if one has equilibrated a metal surface to a certain temperature, one can read in the information of this equilibration by means of an mxt_conf*n*.bin-file and add a particle to it by setting the *particle*-flag in the control file to start MD-simulations. In the latter case, the *pip*-flag needs to be specified in the control-file to give information on the initial positions of the particle. Likewise, the temperature only needs to be specified if one wants to continue simulations at a temperature that deviates from that of the mxt_conf*n*.bin-file.

If the option 'geo' is used, a specific mxt_conf*n*.bin-file of the number n (*which configuration*, Tab. 1.1) is read in. For 'conf', an mxt_conf*n*.bin-file is chosen randomly from a batch of 'number of configurations' mxt_conf*n*.bin-files which makes it possible to start MD trajectory calculations from a number of different configurations for the equilibrated surfaces.

With the option 'fit', a fit can be run, which reads in the relaxed atom positions form a POSCAR-file and further needs the number of the AIMD-trajectory to which the fit is to be done, followed by the number of the fit.

Table 1.1: Selection options for configuration files, to be inputted into the control file as shown. *address* signifies the absolute path to the file of the name 'file'.

conf	fit	'address/file'	AIMD trajectory No.	fit number
conf	POSCAR	'address/file'		
conf	conf	'address/file'	number of configurations	
conf	geo	'address/file'	which configuration	

```
start 1
ntrajs 1
Tsurf 230
step 1
nsteps 50001
wstep 50000 1
lattice   Au 196.96657 7 emt515_Au.nml ver -3
pes emt
celldim 2 2 6 none
rep 1 1
conf POSCAR au111_2x2x6.POSCAR
```

```
start 1
ntrajs 1
step 1
Tsurf 300
nsteps 200001
wstep 1 200
lattice  Au 196.96657 7 emt515_Au.nml ver -3
pes emt
conf mxt "/home/theory/mxt/thermalisation/p515_ver_6x6x6_T300/conf" 1
```

Figure 1.3: The control files used in MD_tian to create 1000 configurations of an equilibrated Au slab at 300 K. Top: Control file to create a single file for an equilibrated configuration, bottom: control file to create 1000 further equilibrated configurations.

1.4 The Control File for MD simulations

Upon execution, the *mdtianxia* program demands an input file which is the control file. It is usually named 'md_tian.inp' but its precise name has no influence on the program and it contains all the flags that are necessary to execute the program.

MD-simulations

Figure 1.3 shows the control files used to equilibrate a slab and Fig. 1.4 shows the control file that uses the configurations from the slab-equilibration as input to start an MD-simulation. The construction of equilibrated slab configurations that can be used as starting configurations works by first creating a single equilibrated configuration using the control file in Fig. 1.3, top, and then, starting from that configuration, creating 1000 configurations in total which can be randomly sampled at the start of a trajectory (Fig. 1.3, bottom). The flags are:

azimuth
(Real)
Default: azimuth 0
Sets the azimuth incidence angle ϕ_{in} of the projectile. $\phi_{\text{in}} = 0^\circ$ corresponds to scattering along the $[1\bar{1}0]$-direction.

celldim

```
start 1
ntrajs 1
step 0.1
nsteps 10000
wstep -1 1
Einc 3.33
inclination 45
azimuth 60
pes emt
projectile H   1.0079  7 emt515_H.nml lan  1
conf mxt "/home/theory/mxt/thermalisation/rec_p515_ver_6x6x4_T300/conf" 1000
pip 0 6.0
```

Figure 1.4: The control file used in MD_tian for a typical MD-simulation.

(Integer, Integer, Integer, Character(,Integers))
Default: celldim 2 2 4 none
Gives the dimensions of the slab. It is only necessary when *conf* is specified with 'fit' or 'POSCAR'. First, the number of atoms in x- and y-direction is given and then the number of layers in z-direction. This option also allows to include ad-atoms or steps, for which the flag 'none' in line 10 of Fig. 1.3, top, would have to be replaced by 'atlayer' followed by the number of atoms for each layer, starting from the surface (see Fig. 1.6, line 9).
Example:
A $2 \times 2 \times 4$ slab with one ad-atom on the surface would be specified in the control file in the following manner: *celldim 2 2 5 atlayer 1 4 4 4 4*.

conf offers the choice of the input file as described in the previous section (Sec. 1.3 and Tab. 1.1).

Einc
(Real)
Default: Einc 5
Defines the incidence energy of the projectile in eV.

inclination
(Real)
Default: inclination 0
Sets the polar incidence angle θ_{in} of the projectile. $\theta_{\text{in}} = 0°$ corresponds to normal incidence, $\theta_{\text{in}} = 90°$ to parallel incidence.

lattice and **projectile**
(Character, Real, Integer, Character, Character, Integer)
Default: lattice Elyrium 1.0 0 'empty' none -1 Specifies the species of the slab and particle, respectively. At the moment, the EMT-implementation only offers two species in interaction with one another. Both flags are followed by the element abbreviation of the

species, its atomic mass in atomic mass units, the number of the parameters, the name of the parameter file and the propagation algorithm (ver = Verlet, lan = Langevin, pef = Verlet, but saving hypothetical energy loss to ehp during the calculation, bee = Beeman). For 'lattice', the last integer determines how many atoms are to be kept fixed to their positions during the propagation. A negative number denotes the number of layers, while a positive number affixes that number of atoms, starting from the first atom given in the configuration file for the slab. For 'projectile', the last integer determines the number of particles in one periodic image.

nsteps
(Integer)
Default: nsteps 100
Are the number of steps made during the trajectory. So, *nsteps·step* gives the total length of the trajectory in fs.

ntrajs
(Integer)
Default: ntrajs 10
Defines the number of trajectories that are to be calculated.

pes
(Character)
Default: pes emt
This flag will in future offer the choice between different methods to calculate the forces and energies. At the moment, only EMT can be used.

pip
(Integer, (Real), Character) Default: pip -1 (6.0 top) Describes how the particle should start the calculation (see also Tab. 1.2). *pip* = 0 determines that only one particle is to be considered and that its position is to be chosen randomly. It is followed by the starting distance to the surface. The flag also allows to read in a specific starting position

Table 1.2: Options for *pip*.

Option	Explanation
-1	Readin from configuration file. Default.
0	Only one projectile will be considered. A random position is chosen.
1	One Projectile. The coordinates of impact at 0 Å are chosen via *keyword*.
2	Positions for projectile are read in from extra file. Set *keyword* to filename
3	One Projectile. The coordinates of start are chosen via *keyword*.

from the configuration file (*pip*=−1), the specific impact site (*pip*=1 that can be chosen with a keyword, see Tab. 1.3), extra file to read in starting positions (*pip*=2) or starting positions determined by site-name (*pip*=3). For *pip*=2) the structure of the file to read in the particle coordinates is as follows:
number of projectiles
position of projectiles
Be careful to keep to the number of projectiles you specified under the flag *projectile* in the input file. If your number of projectiles in the projectile-file is too small, the program will abort. If it is too large, the program will only read in the positions of the first projectiles.

rep

Table 1.3: Options for the keyword when *pip*=1 or 3.

Option	Explanation
top	Top site, i.e. *xy*-position of surface atom in middle of slab
fcc	Face-centred cubic hollow site.
hcp	Hexagonal closed packed hollow site.
bri	Bridge site, i.e. *xy*-position of the middle of the connecting line between two adjacent surface atoms.

(Integer, Integer)
Default: rep 2 2
Gives the number of times the slab is to be repeated into x- and y-direction. Although md_tian treats each problem within periodic boundary conditions, the EMT calculations need a supercell that includes up to the next-next-nearest neighbor atoms. This makes it necessary to increase the size of the input cell from usually 2×2 in xy-direction specified in the POSCAR file to 6×6. To put it in other words: this option allows the user to input a POSCAR-file of minimal extension in xy-direction (2×2), but use a much larger slab for the actual MD-simulation. A larger slab should circumvent the problem of artificial phonon-modes impressed upon the surface by small cells and periodic boundary conditions for longer time-scales. To use this flag, *celldim* and *conf* = POSCAR have to be set. Please bear in mind that *rep* results in repeating the slab in xy-direction *rep* times. That means that, if you start out with a $2 \times 2 \times 4$ slab and set *rep* to 1, your resulting cell will contain a $6 \times 6 \times 4$ slab. Likewise, *rep* 2 will result in a $10 \times 10 \times 4$ slab, &c.

start
(Integer)
Default: start 1
Is the number of the first trajectory in the simulation.

step
(Real)
Default: step -1.0 Is the step in femtoseconds with which the trajectory is propagated.

Tsurf
(Real)
Default: Tsurf 300 Gives the surface temperature that the slab should have. It overwrites the temperature given in the configuration file.

wstep
(integer, integer)

Default: wstep −1 1 Determines the type of the output file and when it is to be written. It will be explained in more detail in the following section (Sec. 1.4.3).

1.4.1 Fitting Procedure

The fitting procedure employed a Levenberg-Marquardt [8, 9] damped least squares procedure which minimizes the rms deviation of the energy values given by DFT and the EMT PES. Since I have used the fitting routine, the Levenberg-Marquardt damped least square procedure has been replaced with an Trusted Region Nonlinear Least Squares with Linear Bound Constrains procedure based on the Levenberg-Marquardt procedure from the Intel MKL libraries which allows fitting with constrains. Since I have used the fitting procedure, it has been improved using a genetic algorithm [10, 11] whose description, since not relevant for the present thesis, will be omitted. A control file for the fit using this routine can be seen in Fig. 1.5, also to be used with MD_tian. The flags in the control file that have not been mentioned in Section 1.3 here are:

3Dgrid
(Real, Real, Integer, Integers)
Gives options for the 3D-grid. The first two numbers determine the lowest and highest energy value of configurations allowed in the fit. The third number determines the number of sites that are to be used in the fit, followed by their identification number.

aimd
(Real, Real, Real, Real)
Default: aimd -20.0 20.0 0.0 7.0
The first two numbers determine the lowest and highest energy value in electron volts belonging to configurations that are to be used in the fit. The third number gives the minimal distance of the particle to a surface atom and the last one excludes distances from the surface above this number. In the example in Fig. 1.5 this means that only energy values between −20 and 20 eV are included into the fit, but no values where the particle is closer to the surface than 0.1 Åor further away than 3.0 Å

conf

(Character, Character, Integer, Integer)

The last two numbers determine which AIMD trajectory is to be used for the fit (in Fig. 1.5 that would be trajectory No. 817) and the number of the fit (2000)

evasp

(Real)

Default: evasp $-24.995869d0$ This is the reference energy calculated with the quantum mechanical code used to construct the input data set. Usually, this is the energy of a particle 6 Å above the surface used in the calculations.

fitconst

(Integer, Integers)

Determines which parameters are kept fixed to the values given in the parameter input files. The first number after the flag gives the number of parameters that are kept fixed (see Fig. 1.5, line 12). The following numbers correspond to the identification numbers of the parameters. Parameters 1–7 are the parameters of the particle and 8–14 those of the slab according to the order of parameters in the EMT parameter file ($1, 8 = \eta$, $2, 9 = n_0$, $3, 10 = E_0$, $4, 11 = \lambda$, $5, 12 = V_0$, $6, 13 = \kappa$,$7, 14 = s_0$).

fitmix

(Integer, Integer)

Determines the number of points to be taken from the 3D-grid for the fit (Fig. 1.5, line 8, 700) and from the selected AIMD trajectory (200).

maxit

(Integer)

Is the maximal number of iterations.

trajname

(Integer, Characters)

```
1 projectile H 1.0079 7 'emt_stroem_H.nml' ver 0
2 lattice Au 196.96657 7 'emt_stroem_Au.nml' ver 0
3 pes emt
4 celldim 2 2 4 x
5 rep 1 1
6 conf fit au111_2x2x4.POSCAR 817 2000
7 trajname 13 005 010 801 814 817 818 820 821 825 831 832 833 858
8 fitmix 700 200
9 evasp -24.995689d0  ! A value for Au 2x2
10 3Dgrid -20.0 20.0 4 7 3 1 10
11 aimd -20.0 20.0 -0.1 3.0
12 fitconst 14 1 2 3 4 5 6 7 8 9 10 11 12 13 14
13 maxit 1
```

Figure 1.5: An exemplary control file for fitting with MD_tian.

This one determines the total number of AIMD trajectories that are available and is followed by the number of all these trajectories. This allows to calculate the rms-error to a flexible number of AIMD configurations.

Strictly speaking, a routine is implemented into the program that allows fitting the electron densities obtained from the VASP calculations. However, because the electron densities obtained from EMT and VASP are not the same entities, this routine has been commented out and is at present not serviceable.

A number of input-filesgo along with the fit:

input fit

1.4.2 Surface Annealing

To test and describe the stability of the reconstructed surface, I performed simulations with surface annealing with the MD_tian program. An exemplary control file I used to steer the simulated annealing is shown in Fig. 1.6. The flags that were not described above are those that control the annealing procedure.

anneal

(Real, Real)

Default: none

Controls the annealing procedure; the first number is the maximal temperature $T_{\max}$ that should be reached during annealing (in case of Fig. 1.6 700 K, see line 11) and the second number is the number of steps t_{step} for which a temperature should be simulated. The annealing starts at $\frac{2\,t_{\text{step}}}{nstep} \cdot T_{\max}$, then, the surface is heated up in $nstep/(2\,t_{\text{step}})$ intervals to $T_{\max}$ and then goes down again to *Tsurf*. Each temperature interval is simulated for t_{step} steps. The simulated annealing is repeated in *ntrajs* cycles.

To simulate annealing, the Langevin Dynamics are used as a thermostat. A higher friction coefficient of $\eta \approx 3{\cdot}10^{-3}\,\text{fs}^{-1}$ was assumed. This makes the annealing simulations more effective and decreases the simulation time. I chose the friction coefficient for the simulated annealing to assume roughly the magnitude of friction an H atom would experience inside the gold surface. By probing higher and lower friction coefficient, I checked that the friction coefficient used for the simulated annealing gives the same results for structure and energy values after an annealing simulation.

1.4.3 Output files

MD-simulations

The *mdtianxia*-program offers seven different output options which can be controlled with the flag *wstep* in the input file. The first slot for *wstep* determines which kind of output file is printed and the last, where it makes sense, in which step.

wstep(1) = m saves all the information that is necessary to continue the simulation into a binary file of the form mxt_conf*n*.bin (where n can be any number between 99999999 and 00000000) after the m^{th} step and then every wstep(2)$^{\text{th}}$ step. This includes saving the step, the surface temperature, all input parameters for the species in the system such as number of atoms, fixed atoms, current velocities, positions, accelerations and densities. Furthermore, the step and the potential energy is saved.

wstep(1) = 0 produces files of the form mxt_trj*n*.dat. They contain the time step the energies for species along the trajectory and the density, the position and the velocities of the projectile. If only 'lattice' is specified, it can be used to determine the temperature

to which the slab equilibrates.
wstep(1) $= -1$ produces files of the form mxt_fin$n$.dat which contain the initial conditions and energies and those at the end of the trajectory as well as the number of bounces, the first five bounce sites and the lowest position of the projectile. This option should be used if MD-simulations are performed where only the result of a trajectory is interesting.
wstep(1) $= -2$ produces files of the form mxt_conf$n$.xyz which can be read in with visualization programs and contain only the positions of the slab and particle atoms.
wstep(1) $= -3$ produces a POSCAR-file at the end of the trajectory with the name mxt_anneal$n$.POSCAR. It also contains information about the energies at the end of the trajectory.
wstep(1) $= -4$ produces files of the form mxt_rv$n$.dat which contain the initial conditions and energies and those at the saving-time of the trajectory. Here, the positions, velocities and densities are saved in every wstep(2)$^{\text{th}}$ step of the trajectory.
wstep(1) $= -5$ produces files of the form mxt_conf$n$.pdb. These files have the pdb-

```
1 start 1
2 Tsurf 0
3 step 1
4 nsetps 100000
5 wstep 0 1
6 lattice Au 196.96657 7 emt515_Au.nml lan -3
7 pes emt
8 rep 0 1
9 celldim 22 2 6 atlayer 46 44 44 44 44 44
10 conf POSCAR 'rec_au111_22x2x6.POSCAR'
11 anneal 700 500
12 ntrajs 20
```

Figure 1.6: An exemplary control file for annealing simulations with MD_tian for a slab with a reconstructed surface that has one extra atom per discommensuration line.

format and can be used e.g. with the Visual Molecular Dynamics (VMD) package [12].

Output files of the fitting routine

The fitting routine automatically creates four types of output-files for the fit n: the parameter file(s) that contain the new parameters resulting from the fit, the dev2eqdft$n$.dat file and the densEqdft$n$.dat file, the traj$n$.dat. It furthermore writes information on the fit into the c44rms.dat-file.

The dev2Eqdft$n$.dat file contains information on the 3D-grid: The number of the site and its x, y-position in steps of 0.1 Å, staring from 6.0 Å above the surface to ∼9.0 below the surface. For this, the Eq_points_dft.dat-file is read in which contains all the coordinates for this purpose. In the dev2Eqdft$n$.dat-file, the coordinates are followed by the potential energy calculated with the parameters resulting from the fit. This file can be used to show a fit's reproduction of the 3D-grid input data set.

The densEqdft$n$.dat file works in the same manner as the dev2Eqdft$n$.dat file, only instead of the potential energy at each position, it contains the local background electron density.

The traj$n$_1.dat-file is related to the AIMD-trajectories that are used in the input-data set and control data set for the fit. It contains the step of each trajectory followed by the potential energy from the DFT-simulations and then by the potential energy calculated with the fit parameters from the positions corresponding to the step.

1.5 The Propagation

1.5.1 Propagation Algorithms

Propagation is done using either the Langevin or the Verlet algorithm. The program also offers the Refson-Beemann algorithm [13]. The velocity Verlet algorithm was implemented in the following form [14] where δt is the time step, $\mathbf{r}$ is the position, $\mathbf{v}$ the velocity and $\mathbf{a}$ the acceleration.

$$\mathbf{v}(t + \tfrac{1}{2}\,\delta t) = \mathbf{v}(t) + \tfrac{1}{2}\,\delta t\,\mathbf{a}(t) \tag{1.1}$$

$$\mathbf{r}(t + \delta t) = \mathbf{r}(t) + \delta t\,\mathbf{v}(t + \tfrac{1}{2}\,\delta t) \tag{1.2}$$

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t+\tfrac{1}{2}\,\delta t) + \tfrac{1}{2}\delta\, t\,\mathbf{a}(t+\delta t) \tag{1.3}$$

For the Langevin-equation the extended Verlet algorithm was implemented in the following manner according to Allen and Tildesly [14]:

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + c_1\,\delta t\,\mathbf{v}(t) + c_2\,\delta t^2\,\mathbf{a}(t) + \delta\mathbf{r}^{\mathrm{G}} \tag{1.4}$$

$$\mathbf{v}(t+\delta t) = c_0\,\mathbf{v}(t) + c_1\,\delta t\,\mathbf{a}(t) + \delta\mathbf{v}^{\mathrm{G}} \tag{1.5}$$

For $\eta > 0.01\,\text{Å}^{-3}$ and $T < 0.0001\,\mathrm{K}$, the parameters can be calculated from the exact expression. Otherwise, the Taylor series was used.

$$c_0 = \mathrm{e}^{-\eta\,\delta t} \approx 1 - \eta\,\delta t + \frac{1}{2}(\eta\,\delta t)^2 \tag{1.6}$$

$$c_1 = (\eta\,\delta t)^{-1}(1-c_0) \approx 1 - \frac{1}{2}\eta\,\delta t + \frac{1}{6}(\eta\,\delta t)^2 \tag{1.7}$$

$$c_2 = (\eta\,\delta t)^{-1}(1-c_1) \approx \frac{1}{2} - \frac{1}{6}\eta\,\delta t + \frac{1}{24}(\eta\,\delta t)^2 \tag{1.8}$$

and the stochastic integrals with m being the mass of the particle subjected to the Langevin equations.

$$\delta\mathbf{r}^G = \int_t^{t+\delta t} \frac{dt'}{m\eta}(1-\mathrm{e}^{-\eta(t+\delta t-t')})\mathbf{F}^{\mathrm{st}}(t') \tag{1.9}$$

$$\delta\mathbf{v}^G = \int_t^{\delta t+t} dt' \frac{\mathrm{e}^{-\eta(t+\delta t-t')}}{m}\mathbf{F}^{\mathrm{st}}(t') \tag{1.10}$$

The stochastic force $\mathbf{F}^{\mathrm{st}}$ is sampled from a bivariate Gaussian distribution.

1.5.2 Disintegration Temperature

I estimated the temperature at which the slab becomes unstable by equilibrating the slab at successively higher temperatures in steps of 50 K for 20 ps using a $6 \times 6 \times 6$ cell, and took the highest temperature at which the slab did not disintegrate as T_{stable}. Whether or not the slab had disintegrated, I first at all check visually by looking at the structure of the slab throughout the trajectory and especially in the last step: if one of the atoms left its surface site, either by going into the gasphase or moving on top of the other surface atoms and thereby leaving a vacancy, I considered the slab as above its temperature of

stability. I have not been able to link the melting temperature to other surface properties nor did I find a relation to the parameters of the fit. Since discerning the disintegration temperature visually is not very precise, I tested the disintegration temperature in large steps of 50 K and, if in doubt, always chose the lower temperature. This means that the disintegration temperatures T_{stable} given in this thesis are a lower limit: the surface will definitely be stable up to this temperature, but it may also still be stable for $\approx 50\,\text{K}$ above it.

2 How to do a fit

Overview

The procedure to procure a H@Metal(111)-GGA-PES with VASP is to do 1) Bulk-calculations: calculate the lattice constant for the chosen functional, 2) Slab-calculations: calculate the interlayer distance to obtain a fully relaxed slab, 3) Optimise the H@Metal-system, 4) Map out an energy grid, and 5) Fit to Effective Medium Theory.
1) to 4) involve VASP-calculations. The POTCAR-file stays the same for 1) and 2) and is joined with the POTCAR-file for H for 3) and 4). The other input-files needed are the POSCAR-file, which contains the position of the atoms; the INCAR-file, which contains options for VASP; and the KPOINTS-file which determines the sampling of the brillouin-zone.
All files, most of all the INCAR-file, should be kept minimal to minimise mistakes.
Information summed up here for the VASP-calculation is usually taken from the VASP-manual (http://cms.mpi.univie.ac.at/vasp/vasp/vasp.html, as of 15.01.15, 10:30.).

2.1 Bulk-Calculations

Purpose:
The Bulk-calculations determine the lattice constant corresponding to the chosen GGA-functional, because GGA-functionals do not reproduce the experimental lattice constant but have their own, internal lattice constant.
Optimisation:
Optimisation needs to be done to get the best result with the lowest computational effort. For that, several input parameters for VASP should be checked:

ENCUT: The plane-wave energy cut-off. The smaller this energy, the lower the computational effort. If it is too small, though, necessary contributions are neglected. Vary in steps of 50 eV. One already needs to think about future calculations at this point. ENCUT needs to be at least as large as the largest ENMAX of all POTCAR files which will be used. Do not compare calculations of different ENCUT. The ENMAX-tag can be found in the first lines of every POTCAR file.
K-points: The k-points are the mesh with which the brillouin-zone is sampled in reciprocal space. The larger the number of k-points, the more accurate the sampling. But a large number of k-points also increases the computational effort. Do calculations with 8 to ∼35 k-points
ISMEAR: This parameter determines the smearing. For metal bulk and slab do ISMEAR=1 or ISMEAR=2 for Methfessel-Paxton. You can also check other forms of smearing, but you do not need to.
SIGMA: This parameter is related to the entropy in the system. It needs to be checked and adjusted at the end of the calculations. For the beginning, set SIGMA = 0.2. After the lattice constant is determined, vary SIGMA between 0.01 and 0.5 and check the entropy as explained below. The entropy should be less than 1 meV/atom. Too large smearing values of SIGMA might lead to a wrong energy, but too low values require a larger kpoint-mesh to accomplish convergence. Adjust so that the entropy per atom is slightly below 1 meV. If you use a 1x1 cell (which you should for bulk-calculations), the entropy is already displayed with respect to one atom.
IBRION: The algorithm with which the ionic positions are updated and moved. 1 and 2 are both reasonable choices for bulk-calculations and should give the same result.

The POSCAR-file

The POSCAR-file (cf Fig. 2.1) has the following structure: The first line is a comment-line. In the second line (Fig. 2.1), the lattice constant a_0 is given. Then, from line 3 to 5, the cell matrix needs to be specified. In line 6, the number of atoms per cell is given, followed in line 7 by the coordinate system in which the coordinates of the atoms are given from line 8 downwards. Due to the periodic boundary conditions, only one atom

```
fcc Au
 a0
 0.5 0.5 0.0
 0.0 0.5 0.5
 0.5 0.0 0.5
   1
cartesian
0 0 0
```

Figure 2.1: An exemplary POSCAR-file for the Bulk-Calculations. “a0" indicates where the lattice constant needs to be placed.

per cell is needed for the bulk calculations. This results in a $1 \times 1 \times 1$ cell.

KPOINTS-file

```
K-Points
 0
Gammacentered
 8 8 8
 0  0  0
```

Figure 2.2: An exemplary KPOINTS-file for the Bulk-Calculations.

The structure for the KPOINTS-file is as follows (cf Fig. 2.2): The first line is a comment-line, followed by the number of k-points. If that number is set to zero, the k-point mesh is set automatically. It is best to leave it at that. The third line expresses where the mesh should be centred (in the example on the Gamma-point of the Brillouin zone), the number of k-points for each coordinate x, y and z are given in line four and in the last line, the shift of the kpoints can but need not be expressed.

2.1.1 General Procedure

To calculate the lattice constant, one starts with initial guesses for the lattice constant in the POSCAR-file (Fig. 2.1, "a_0"). Example: The experimental lattice constant for Au is $a_0 = 4.08\,\text{Å}$. To get the GGA-lattice constant, a_0 is varied from 3.8 to 4.4 Å in steps of 0.05 Å.
The lattice constant can be calculated in two different ways which differ in the input of the INCAR-file. The KPOINTS- (Fig. 2.2) and POSCAR-file (Fig. 2.1) are the same for both options.

Lattice Constant by changing the cell volume

VASP gives the option to relax the cell volume (ISIF = 7 in the INCAR-file, Fig. 2.3). That means that for each POSCAR with their different lattice constants, VASP will try to minimise the stress imposed by the specified lattice constant a_0 and end up with new cell-vectors after the calculation that represent a less strained cell. To evaluate the result, look at the end of the OUTCAR-file for the line "VOLUME and BASIS-vectors are now :" and read the new cell-matrix printed below the line 'direct lattice vectors' (cf Fig. 2.4). In the POSCAR-file used above, you can see that the components of the cell-matrix are either calculated by multiplying the lattice constant by 0.5 or 0.0. To get the new lattice constant, you can now, for example, take the first element of the first lattice vector. Since it is given as 0.5 in your input-POSCAR-file, to calculate the new lattice constant, you now need to multiply the value from the OUTCAR-file by 2. Example (Fig. 2.4): The first element of the cell matrix in the OUTCAR-file is 2.055159357 Å. In the POSCAR-file (Fig. 2.1), the first element of the first lattice vector is 0.5. To get the new lattice constant, you need to calculate $2.055159357/0.5 = 4.110\,\text{Å}$. Your new lattice constant is therefore $a_0 = 4.110\,\text{Å}$. But, if the initial guess is bad, VASP cannot relax the cell enough to get the perfect lattice constant. Therefore, it is necessary to try out several guesses and choose that lattice constant which is closest to its initial guess. It will also be the lattice constant to which most of the other initial guesses converge to.

```
System=Bulk Au
ISTART=0 ! WAVECAR isn't read in
ICHARG=2 ! CHGCAR or CHG aren't read in

PREC=Accurate
EDIFF=1E-05 ! 1st accuracy parameter
EDIFFG=-1.0E-03 ! 2nd accuracy paramter

ISMEAR=1 ! smearing
SIGMA=0.2 ! entropy-related
ALGO=Fast

NSW=100 !number of ionic steps during relaxation
ISIF=7
IBRION=2
POTIM=0.6

GGA=RE ! RPBE-GGA-functional
```

Figure 2.3: An exemplary INCAR-file for calculation of lattice constant via cell volume

Lattice Constant by energy

Another possibility is to evaluate the optimal lattice constant via the system energy as opposed to reading it directly from the OUTCAR file. To use this method, one needs to vary the lattice constant in the vicinity of the experimental one, as well, just as in the calculation of the lattice constant via the cell volume. This time, the cell volume is kept constant by specifying NSW = 1, meaning that no ionic relaxation is done. Just the energy of the given configuration is calculated. To evaluate the result, look at the end of the OUTCAR-file for the line " energy(sigma→0) = " and read in the energy. The guess with the lowest energy will be very close to the correct lattice-constant. If you fit a parabola through your energy-points, the minimum of the parabola will give the lattice

```
 VOLUME and BASIS-vectors are now :
 -----------------------------------------------------------------------------
  energy-cutoff  :      229.94
  volume of cell :       17.36
      direct lattice vectors                 reciprocal lattice vectors
     2.055159357  2.055159357  0.000000000     0.243290136  0.243290136 -0.243290136
     0.000000000  2.055159357  2.055159357    -0.243290136  0.243290136  0.243290136
     2.055159357  0.000000000  2.055159357     0.243290136 -0.243290136  0.243290136

  length of vectors
     2.906434236  2.906434236  2.906434236     0.421390877  0.421390877  0.421390877
```

Figure 2.4: An exemplary OUTCAR-file for calculation of lattice constant via cell volume

constant (in case of Fig. 2.5 that would be $a_0 \approx 3.96$). The goodness of the fit can be rated by eye inspection of a plot. A second order polynomial might not be a sufficient fitting function. In this case, use a fourth order polynomial. One needs to remember that such a polynomial has five fitting parameters, so one needs to make sure to use a much higher number of energy points to avoid overfitting.

After the calculations for different ENCUTs and k-points (and whichever parameter was varied) are finished, the true lattice constant and the best set of parameters is chosen in the following manner:
Plot the lattice constant for different k-points (cf Fig. 2.7). For large k-points, the lattice constant should converge to a certain value ($a_0 = 3.9662\,\text{Å}$ in case of Fig. 2.7). This value should be regarded as the optimal value.
Lastly, the SIGMA-parameter needs to be checked. For that, vary the parameter SIGMA between 0.01 and 0.5. Once the calculations are done, the entropy needs to be determined. This can be done in two different way. You can check in the OUTCAR-file for the lowest occurrence of the line "entropy T*S EENTRO =" (cf. Fig. 2.8), or you can calculate the difference between the free energy and the energy without entropy. Both should be

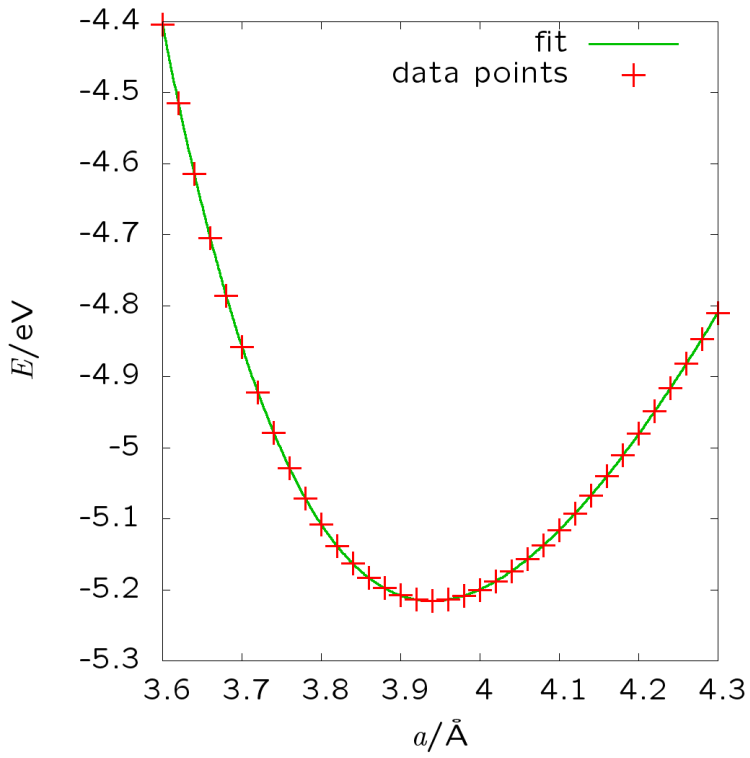

Figure 2.5: An example of fitting different lattice constants with a polynomial expression. The optimal lattice constant is at the fit's minimum.

less than 1 meV/atom. SIGMA should be as large as possible without compromising the energy of the system, so choose the largest SIGMA whose energy still corresponds to the energy of the calculations with lower SIGMA.

2.2 Slab-Calculations

Purpose:
The metal-slab terminates at the end of the surface to the vacuum. For the atoms of at least the surface layer, neighbours are missing. This can lead to a contraction or expansion of the interlayer distance of the first few layers of the slab. In theory, it can

```
SYSTEM = Au: fcc
ENCUT = 300
PREC = Normal
LREAL = .False.
SIGMA = 0.2
ISMEAR = 1
GGA = 91        ! PW91

NSW = 1     ! No ionic relaxation

IBRION = 1  ! These parameters are
ISIF = 2    !   of no importance if no
POTIM = 0.5 !   ionic relaxation is performed.
```

Figure 2.6: An exemplary INCAR-file for calculation of lattice constant via energy

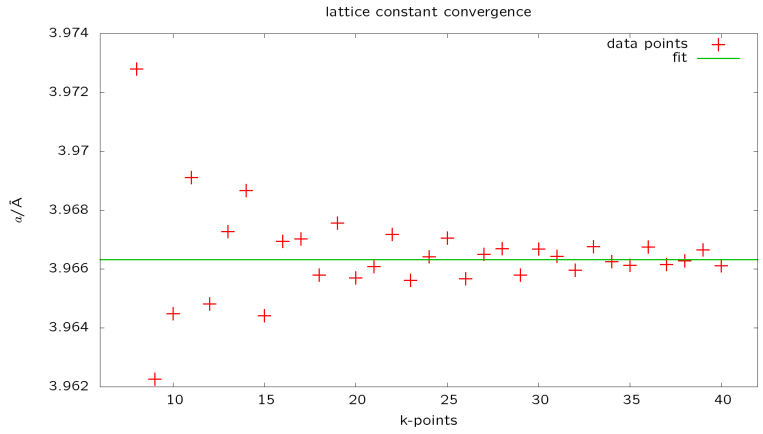

Figure 2.7: Lattice constant convergence for different k-points.

also lead to a reconstruction of the surface's geometry (cf e.g. herringbone reconstruction of Au(111)). Since the reconstruction usually takes place in larger unit cells than we can consider here, we omit reconstruction in x and y-direction and only consider the interlayer-relaxation in z-direction.

```
 Free energy of the ion-electron system (eV)
  ---------------------------------------------------
  alpha Z        PSCENC =       116.98128788
  Ewald energy   TEWEN  =      -950.74055391
  -1/2 Hartree   DENC   =      -103.57651771
  -V(xc)+E(xc)   XCENC  =      -424.26980779
  PAW double counting   =          .00000000         .00000000
  entropy T*S    EENTRO =         -.00002139
  eigenvalues    EBANDS =         -.66210617
  atomic energy  EATOM  =      1360.63085531
  ---------------------------------------------------
  free energy    TOTEN  =        -1.63686377 eV

  energy without entropy =       -1.63684238  energy(sigma->0) =       -1.63685664
```

Figure 2.8: Finding the entropy in the OUTCAR-file.

For this, we fix the x and y-coordinates of all atoms in the system and let the z-coordinate relax for all layers but the lowest two layers. The z-coordinates of the lowest two layers need to be fixed, too, to prevent the slab from drifting through space in z-direction and to give the above layers the impression that they are sitting on a metal-bulk.

2.2.1 KPOINTS

In the KPOINTS-file, a surface needs to be considered now by setting the last k-point to 1 (Fig. 2.9).

2.2.2 POSCAR

In the POSCAR-file, we now need to construct a slab. The first step for this is forcing periodic boundary conditions to see a slab of metal and not a bulk as before. We do that by enlarging the cell in z-direction and leaving part of the space without any atoms, that is, we create a vacuum of a certain thickness *vac* above our slab. Because the

```
K-Points
0
Gammacentered
17 17 1
0 0 0
```

Figure 2.9: Exemplary KPOINTS-file for the slab-relaxation.

```
  Au(111)  1 x 1  vac= 13.0000000000000000  A nlayer=  4
  1.000000
   2.970555537    .000000000    .000000000
  -1.485277768   2.572576696    .000000000
    .000000000    .000000000  20.276200000
   4
Selective dynamics
Cartesian
   .000000000   .000000000   .000000000  F  F  T
  1.485277813   .857525591 -2.425400000  F  F  T
   .000000000  1.715051182 -4.850800000  F  F  F
   .000000000   .000000000 -7.276200000  F  F  F
```

Figure 2.10: Exemplary POSCAR-file for the slab-relaxation. Please note that the lattice constant has been multiplied into the cell-matrix and the coordinates and is therefore set to 1.0 in line 2.

periodic boundary conditions consequently do not replicate our single atom from the bulk-calculations in z-direction to form a continuous bulk, there is only one single layer of metal if only one atom is placed in the unit cell. If more layers are to be considered (which is advisable, since we want to model a slab and not only monolayer of the metal), more atoms need to be added into the unit cell, one for each layer (in xy-direction, the periodic boundary conditions still work perfectly fine). Another problem encountered is that, because we want to study a specific surface with its unique geometry, our cell matrix changes compared to the bulk (cf Fig. 2.1 vs Fig. 2.10).

The cell-matrix consists of three lattice vectors written horizontally. Accordingly, line 3 of the POSCAR-file contains the lattice-vector in x-direction, line 4 the one in y-direction and line 5 the one in z-direction. For a (111)-surface, they can be calculated as follows:

$$c_x = a_0 \left(\frac{1}{\sqrt{2}},\ 0.0,\ 0.0 \right) \tag{2.1}$$

$$c_y = a_0 \left(-\frac{1}{2\sqrt{2}},\ \sqrt{\frac{3}{8}},\ 0.0 \right) \tag{2.2}$$

$$c_z = \left(0.0,\ 0.0,\ \frac{(n_{lay} - 1)a_0}{\sqrt{3}} + vac \right) \tag{2.3}$$

Here, a_0 is the lattice constant from the bulk-calculations, n_{lay} is the number of layers in the slab and vac is the vacuum distance. It becomes evident why the lattice constant was multiplied into the cell matrix in the POSCAR-file in Fig. 2.10; the z-coordinate of c_z is not divisible by a_0 any longer. The complete cell matrix can looks as follows:

$$\begin{pmatrix} a_0 \frac{1}{\sqrt{2}} & 0.0 & 0.0 \\ -a_0 \frac{1}{2\sqrt{2}} & a_0 \sqrt{\frac{3}{8}} & 0.0 \\ 0.0 & 0.0 & \frac{(n_{lay}-1)a_0}{\sqrt{3}} + vac \end{pmatrix} \tag{2.4}$$

Furthermore, because we do not want to relax the entire cell now but only parts of the cell, we need to specify this in our POSCAR-file. This can be done by putting the command 'selective dynamics' above the specification of the coordinate system. To tell VASP how to relax the coordinates, either T for 'true' (relax) or F for 'false' (keep fixed) need to be placed behind the coordinates of each atom, for each coordinate. Example: In line 9 of the POSCAR-file (cf Fig. 2.10), the coordinates of the first atom are given. Because relaxation should only be done in z-direction, the x and y-coordinates are fixed by writing F F at the end of the line. The z-coordinate is relaxed by placing T behind the two Fs.

A good first guess for vac is $\approx 13\,\text{Å}$. This should avoid interaction between the slabs created by the periodic boundary conditions.

In the exemplary POSCAR file in Fig. 2.10, the z-coordinate $= 0.0\,\text{Å}$ is used to denote where the surface of the slab is.

2.2.3 Convergence Tests

First, the most advantageous parameters for the relaxation should be determined. For that, the following flags can (e.g.) be tested:
ENCUT: This flag does not necessarily need to be tested. The value obtained from the Bulk-calculations or the default should be fine.
K-points: Check k-points from 4x4x1 to about 20x20x1 and look at the surface energy and surface relaxation. Compare the values from small k-point sets to larger k-point sets and note which small k-points set produce results similar to the high k-point limit. You might also want to check the literature to see how well your functional of choice performs in general.
ISMEAR: Dfferent smearing methods can but need not be tested. If not tested, ISMEAR = 1 is a good choice.
Number of layers: It might be useful to do the procedure for several surface layers while one is at it already.
vac: The thickness of the vacuum layer should be checked to avoid interactions between the slabs created by the periodic boundary conditions. (2-15 Å). The smaller the vacuum, the lower the calculation effort.
EDIFF: Different convergence criteria can be checked.
SIGMA: At the end of the convergence tests, SIGMA should be checked again. As mentioned in section 2.1, too large smearing values of SIGMA might lead to a wrong energy, but too low values require a larger kpoint-mesh. SIGMA should be as large as possible while keeping the difference between the free energy and the total energy minimal.

In the first step, the k-points need to be tested. For this, set IBRION = 1 or 2 and ISIF = 2 in the INCAR-file (cf Fig. 2.11) and test the k-points from 4 to ∼20. To avoid interaction of the slabs created through the periodical boundary conditions, set *vac* to 13 Åin the POSCAR-file. For each calculation, extract the 'energy without entropy' from the OUTCAR-file after the calculations have finished and select the lowest k-points that reproduces the energy to which the larger k-points converge to. You can perform this procedure for several ISMEAR or check the behaviour of ISMEAR with the optimal k-point you get from the k-point-calculations here. In the next step, evaluate the behaviour

```
System=Slab Au

PREC=Accurate
EDIFF=1E-05
EDIFFG=-1.0E-03

ISMEAR=1
SIGMA=0.1
ALGO=Fast

NSW=100
ISIF=2
IBRION=1
POTIM=0.6

GGA=RE
```

Figure 2.11: Exemplary INCAR-file for the relaxation of the slab and the optimisation of the K-points and ISMEAR.

of the SIGMA. For this, set IBRION = -1 in the INCAR-file and remove the 'Selective Dynamics' line and the corresponding 'F' and 'T' declarations form the POSCAR-file, or simply set NSW = 1, invalidating IBRION by keeping the ions on fixed positions. Test SIGMA from 0.01 to ~ 0.5. After the calculations, extract the value of the entropy from the OUTCAR-file either by looking for the bottommost occurrence of the line 'entropy T*S EENTRO =' or by calculating the difference between the free energy and the energy without entropy (cf Fig. 2.8). Choose SIGMA such that the value is maximal without compromising the energy, but less than 1 meV/atom.
Once SIGMA is chosen, you can perform further tests for the thickness of the vacuum or the accuracy of the calculations (EDIFF). Make sure that no ionic relaxation is performed (I'm sure it'll also work if you have IBRION=1, ISIF=2 and NSW > 1 and use the POSCAR from Fig. 2.10, but it might take a little longer.)

2.2.4 Relaxation of the slab

For the relaxation of the slab, use the optimised results from the above section and put them into the INCAR and POSCAR-files given in Fig. 2.11 and Fig. 2.10. In the POSCAR-file, no matter how many layers you are considering, keep the lowest two layers fixed to give an impression of the bulk-interlayer distance.
Once the calculations are over, check whether the calculations have converged and if they have, extract the new interlayer distances from the CONTCAR-file. The calculations are converged if e.g. there is a line at the end of the OUTCAR-file saying 'reached required accuracy - stopping structural energy minimisation' (Fig. 2.12). If the calculation did not converge, consider raising the number of ionic (NSW) or electronic (NELM) steps or check for errors. The electronic energy needs to be converged to obtain accurate forces for the ionic relaxation.
In case you run into an error saying "ZBRENT: fatal error in bracketing please rerun with smaller EDIFF, or copy CONTCAR to POSCAR and continue", check if your EDIFF is reasonable, copy CONTCAR to POSCAR, set IBRION=1, ICHARG=1 and ISTART=1 in the INCAR file. Then you (re)start your calculation just as the one that produced the error. ICHARG and ISTART determine the type of calculation which is done. Setting them equal to one means to continue a previous calculation using the previously produces and nearly converged CHG, CHGCAR and WAVECAR files.
The CONTCAR file is a POSCAR-file that contains the result of the relaxation (cf Fig. 2.13). The positions are given in direct coordinates. You can transform them into Cartesian coordinates via applying the cell-matrix in a matrix transformation to the position and multiplying the resulting matrix by the lattice constant given in line 2. Either use the CONTCAR-file as a POSCAR-file for the next calculations or extract the interlayer distance and form a new POSCAR-file to continue.

2.3 Convergence tests on Surface + H calculations

For the convergence test, the difference in energy between an H-atom 6.0 Å above the slab's surface and on the slab's surface (1.1 Å) is tested.
The **KPOINTS-file** stays the same as for the slab-calculations, with the set of k-points

```
 FORCES: max atom, RMS      .000666     .000369
 FORCE total and by dimension     .000739     .000666


----------------------------------------------------------------------------------------------


reached required accuracy - stopping structural energy minimisation
writing wavefunctions
    LOOP+:  VPU time    1.07: CPU time    1.07


General timing and accounting informations for this job:
========================================================

                 Total CPU time used (sec):       25.321
                           User time (sec):       24.972
                         System time (sec):         .349
                        Elapsed time (sec):       30.153

                  Maximum memory used (kb):      250432.
                  Average memory used (kb):           0.

                         Minor page faults:        4040
                         Major page faults:           2
                Voluntary context switches:        2902
```

Figure 2.12: Did the calculations converge?

deemed optimal during the convergence test for the slab-calculations. The **POTCAR-**

```
 1 x 1  vac= 13.000000000
 1.00000000000000000
     2.9705555370000001     .0000000000000000     .0000000000000000
    -1.4852777680000000    2.5725766960000001     .0000000000000000
      .0000000000000000     .0000000000000000   20.2761999999999993
   4
Selective dynamics
Direct
   .0000000000000000   .0000000000000000  -.0009317218716555   F   F   T
   .6666666865794397   .3333333433103576   .8787660531382623   F   F   T
   .3333333431981487   .6666666866207223   .7607638512147261   F   F   F
   .0000000000000000   .0000000000000000   .6411457768220856   F   F   F

   .00000000E+00   .00000000E+00   .00000000E+00
   .00000000E+00   .00000000E+00   .00000000E+00
   .00000000E+00   .00000000E+00   .00000000E+00
   .00000000E+00   .00000000E+00   .00000000E+00
```

Figure 2.13: The CONTCAR-file contains the new positions after the relaxation, but in direct coordinates.

file for the metal needs to be concatenated with a POTCAR-file for H.

2.3.1 The POSCAR-file

Use the new POSCAR-file with the relaxed interlayer distance from Section 2.2. Remove everything from the POSCAR that refers to relaxations, that is, the line 'selective dynamics' and the Fs and Ts at the end of the single Au-atom coordinates.
New about the POSCAR-file in this step of the calculations is that a hydrogen atom is added to the calculation. That changes two things. For one, the cell-size needs to be increased to avoid interaction between the hydrogen atom with its images created from the periodic boundary conditions. So far, the POSCAR-file had a $1 \times 1 \times n_{lay}$-cell where

n_{lay} is the number of layers in the cell. Now, the cell-size is increased by one atom in each x- and y-direction and a $2 \times 2 \times n_{lay}$ is formed (cf Fig. 2.14). Larger cell-sizes are imaginable, but for the creation of the potential energy surface (PES), AIMD trajectories will later need to be produced which are computationally very demanding, and larger cell-sizes will add a considerable number of atoms which will increase the computational effort tremendously.
Furthermore, the H-atom needs to be added to the POSCAR-file. This is done by adding the number of H-atoms in the cell (1) to line 6. Because the H-atom is an atom of a different species than the Au-atoms, its number is not added to the number of metal-atoms (24, cf Fig. 2.14) but written behind the number of metal atoms. The ordering here needs to be the same as in the POTCAR-file. That is, if the metal-POTCAR-file comes first and the H-POTCAR-file was added behind it, in the POSCAR-file the number and positions of the metal-atoms must come first and the H-number and position after that. Therefore, in our case, the position of the H-atom is written below the positions of the metal atoms.

```
  H + Au(111)  2 x 2  vac= 13.0000000000000000  A nlayer=  6
  1.000000
   5.941111074    .000000000    .000000000
  -2.970555537   5.145153392    .000000000
    .000000000    .000000000  25.107500000
  24   1
Cartesian
    .000000000    .000000000    .000000000
   2.970555537    .000000000    .000000000
  -1.485277768   2.572576696    .000000000
   1.485277768   2.572576696    .000000000
   1.485277813    .857525591  -2.451700000
   4.455833350    .857525591  -2.451700000
    .000000044   3.430102287  -2.451700000
   2.970555581   3.430102287  -2.451700000
    .000000000   1.715051182  -4.857500000
   2.970555537   1.715051182  -4.857500000
  -1.485277768   4.287627878  -4.857500000
   1.485277768   4.287627878  -4.857500000
    .000000000    .000000000  -7.254500000
   2.970555537    .000000000  -7.254500000
  -1.485277768   2.572576696  -7.254500000
   1.485277768   2.572576696  -7.254500000
   1.485277813    .857525591  -9.682100000
   4.455833350    .857525591  -9.682100000
    .000000044   3.430102287  -9.682100000
   2.970555581   3.430102287  -9.682100000
    .000000000   1.715051182 -12.107500000
   2.970555537   1.715051182 -12.107500000
  -1.485277768   4.287627878 -12.107500000
   1.485277768   4.287627878 -12.107500000
    .000000000   1.715051182   6.000000000
```


Figure 2.14: Exemplary POSCAR-file. H-atom was placed above the fcc-hollow site.

Two POSCAR-files need to be produced for this step of the calculations. One where the H-atom is 6.0 Å above the surface and one where the H-atom is close to the surface ($\sim$ 1.1 Å). For the x- and y-coordinate of the system, one of the recognisable surface sites is advisable like top, bridge or one of the two hollow sites.

2.3.2 The INCAR-file

```
ALGO=F
NELM = 120
GGA = RP
ISYM = 1
ISPIN = 2 ! Calculate the spin
LWAVE = .FALSE. ! Don't print the WAVECAR, they waste memory space here
LCHARG = .FALSE. ! Don't print the CHG-files, they waste memory space here
AMIX = 0.3 ! These settings are good.
BMIX = 0.0001 !
AMIX_MAG = 0.3 !
BMIX_MAG = 0.0001 !
MAXMIX = 20 !

PREC = Accurate
SIGMA = 0.1
ISMEAR = -1
EDIFF = 1.0E-4

MAGMOM = 25*1.0
ENMAX  = 350 ! Another way to specify ENCUT
```

Figure 2.15: Exemplary INCAR-file. Note the magnetic moment (MAGMOM) and the spin (ISPIN) that is now specified.

The H-atom has a free electron, therefore, it has a spin and a magnetic moment. This needs to be accounted for. At the end of the optimisation, you can check if it makes a difference to specify it or not.
I have been told that ISMEAR = -1 is the best setting for ISMEAR you can choose for H+metal. You are welcome to check it. But remember, you must not continue any calculations with ISMEAR = -1 that have been performed earlier with another value of ISMEAR. That is, if you did your slab relaxation with ISMEAR = 1, you must not select ISMEAR = -1 for your H@Metal-system.

2.3.3 Convergence tests

Again, the number of **K-points** should be checked (4-20).
ENCUT: Start with the highest of the specified ENMAX-tag of both concatenated POTCAR files. Increase it in steps of 50 eV until convergence is accomplished. If no convergence can be observed, give it the value of the highest ENMAX.
Number of layers: Either choose the number of layers you would like to have to fit your PES or check which number of layers with which number of k-points, ENCUT and SIGMA calculates fastest while still giving the same result as the most expensive calculations.
SIGMA: For sigma, proceed as described in the sections above. Check how thick your **Vacuum-layer** needs to be. Again, 13 Åseems a good guess to begin with.
Again, start out with checking which number of k-points you need. For that, set ENCUT to its default-value, SIGMA to the value from the slab-calculations and the vacuum distance to 13 Å.
Of course, check also whatever other parameter you want.
If you want to be very thorough, test the dependence of each of these parameters from each other. That is: calculate the k-point- and ENCUT- and SIGMA-dependence for every change in the vacuum distance you consider and vice versa.
In general, choose your parameters in such a way that they still reproduce the energy-difference for the most accurate calculations (that is: those calculations that have the highest number of k-points, highest ENCUT and largest vacuum layer). For the energy-

difference, get the 'energy sigma → 0: ' from the OUTCAR-file (cf Fig. 2.4) for both positions of the hydrogen atom and subtract the energy of H at 6.0 Å from the energy of H at 1.1 Å. By the end of this optimisation, you should have your perfect parameters to start building up your PES.

2.3.4 Spin restricted or Spin unrestricted calculations?

VASP offers the possibility to calculate the energy of the hydrogen atom with and without involving the spin. As long as the hydrogen atom is close to the surface, the spin should not play any role, for the interaction with the Au's atoms should even out the spin. When the H-atom is far away from the surface, though, the spin becomes important.
The parameter in VASP that governs if the spin is calculated or not is **ISPIN**. For ISPIN=1, the calculations are performed non-spin-polarized and for ISPIN = 2, spin-polarized calculations are being performed. If the second option is chosen, the flag **MAGMOM** can also be specified. If you count the number of ions N you have in your POSCAR-file and set MAGMOM= $N * 1.0$, you should be fine.
Finally, do another calculation with the H-atom above and at the surface with and without magnetic moment and spin. If the energy is not different, neglect the magnetic moment for the ongoing calculations.
It is more likely that you will need to take the spin into account. If you are not restricted on calculation time, take the spin into account for your entire energy grid.

2.4 The Potential Energy Surface

2.4.1 The H-atom-grid

An exemplary INCAR-file for these calculations is shown in Fig. 2.15. The first step consists of building an energy-configuration grid in which metal-atoms are at their previously calculated relaxed surface layer positions and the H-atom is moved on a narrow grid to sample the configuration space. So far, we have taken a grid of ten sites on the metals surface that focuses on surface sites with distinct symmetrical properties (cf Fig. 2.16). The points 1 to 4 scan the region from the top site (1) to the bridge site (4).

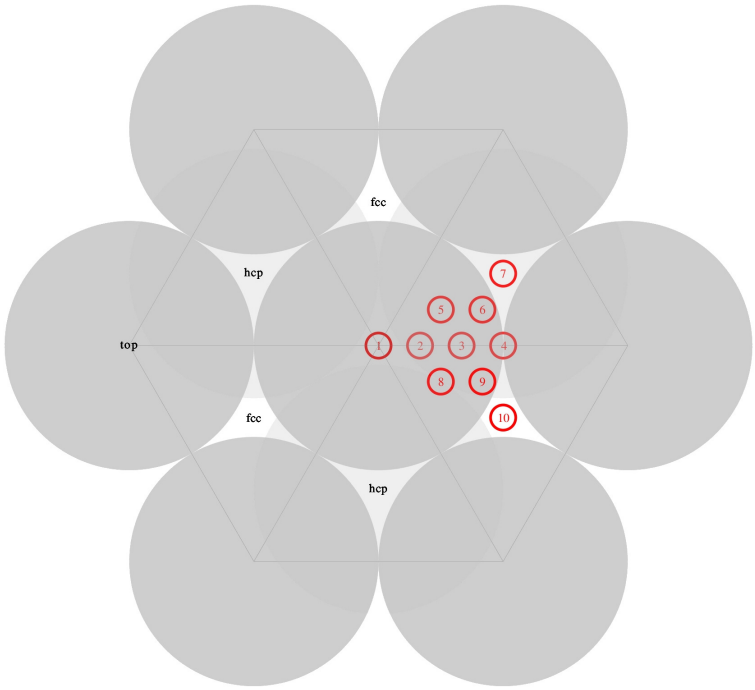

Figure 2.16: Ten Surface Sites for the PES.

(1)-(5)-(7) maps out the straight line from the top site to the hcp-hollow site (7) and (1)-(8)-(10) from the top site to the fcc-hollow site (10). (6) and (9) sample the site of the metal atom. The H-atom should be moved perpendicular (that is, keeping the x- and y-coordinates fixed) to each of these sites starting from 6 Å above the surface to the bottom layer of the slab in steps of 0.2 Å. The x- and y-coordinates of the points can be calculated as shown in Tab. 2.2.
Extract from the OUTCAR-files of each calculation the 'energy sigma $\rightarrow$ 0' -energy. The energy when the H-atom is above the fcc-site and 6.0 Å above the surface serves also as **reference energy**.

2.4.2 The AIMD-trajectories

Part of the fitting procedure also includes taking geometries at which the metal-atoms are not on their equilibrium positions and the H-atom is on other plausible positions

Table 2.1: How to calculate the x- and y-positions of the ten surface sites.

Site	x-coordinate	y-coordinate
1	0.0	0.0
2	$\frac{a_0}{6\sqrt{2}}$	0.0
3	$\frac{a_0}{3\sqrt{2}}$	0.0
4	$\frac{a_0}{\sqrt{8}}$	0.0
5	$\frac{a_0}{4\sqrt{2}}$	$\frac{\sqrt{3}}{12\sqrt{2}}a_0$
6	$\frac{5}{12\sqrt{2}}a_0$	$\frac{\sqrt{3}}{12\sqrt{2}}a_0$
7	$\frac{a_0}{\sqrt{8}}$	$\frac{\sqrt{3}}{6\sqrt{2}}a_0$
8	$\frac{a_0}{4\sqrt{2}}$	$-\frac{\sqrt{3}}{12\sqrt{2}}a_0$
9	$\frac{5}{12\sqrt{2}}a_0$	$-\frac{\sqrt{3}}{12\sqrt{2}}a_0$
10	$\frac{a_0}{\sqrt{8}}$	$-\frac{\sqrt{3}}{6\sqrt{2}}a_0$

that it might take up during a trajectory. To sample that configuration space, we take geometries from an AIMD-trajectory calculated for the very same system. Because we have not yet found out what kind of AIMD-geometries might give the best contribution to a fit, it is best to calculate roughly a dozen AIMD-trajectories and use geometries from each of these trajectories as input for the fit.

Before actually calculating AIMD trajectories, the slab needs to be equilibrated to the desired temperature. First, choose a k-point mesh between 4x4x1 and 10x10x1 that gave good results in terms of Metal-H adsorption energy, surface energy, lattice constant and interlayer relaxation distance. You need to use the same k-points mesh for the trajectories and the H-atom-grid. Then, find the corresponding fully relaxed slab CONTCAR file and use this as the geometry to start the slab equilibration. Make sure it contains the *selective dynamics* tag and allow movement for all atoms in all dimensions except for the two bottommost layers which you fix. The INCAR file for equilibration can be found in Fig. 2.17. Set the initial temperature to twice the value that you want to have for your trajectories. If you do not know why, read about the equipartition theorem. Then, let the slab equilibrate for 500 to 1000 ps (NSW · POTIM) and check the OSZICAR file if the temperature has settled somewhere around your desired temperature. If this is

not the case, there is most probably something wrong with your initial slab geometry in that you did not find the minimum energy configuration. If the temperature is correct, generate different slab geometries for your AIMD trajectories.

Referring again to Fig. 2.17, change ISTART and ICHARGE to 1, delete the TEBEG line and adjust NSW such that 100 fs are simulated. The resulting CONTCAR file is your first slab geometry for a trajectory. Generate about a dozen of those and remember to always keep the CONTCAR files. Next, you need to add an H-atom to them. In the sixth line, add a whitespace and "H" to the metal element abbreviation and in the seventh line, add a whitespace and "1". Below the metal atomic coordinate section, place the coordinates of the H atom. Its x- and y-values must be random in each trajectory and can be generated with your RNG of choice. Its z-value is the height above the slab at which no interaction with the slab is observed and defaults to 6 Å. Also add a line below the metal atom velocity section that describes the initial velocity vector of your H-atom in Å/fs. This one can be the same in each trajectory. Delete all lines below the H-atom initial velocity vector in each CONTCAR file. Also check whether the total energy remained somewhat constant during the equilibration. Plot the total energy (labeled "E=" in the OSZICAR file) against the ionic step and check if it stays constant within about 10 meV. If this is not the case, try decreasing POTIM, decreasing EDIFF or setting real space projection to false.

Now that you have about a dozen files describing system geometries and atom velocities, you can start the trajectories. A suitable INCAR file is shown in Fig. 2.18. Use the same k-points file as before and remember to concatenate the metal and H POTCAR files. After a trajectory has finished e.g. the H-atom was scattered back to its initial height, the H-atom has traversed the slab, or 120 fs have passed, you need to check several things to make sure the simulation is reasonable. First, check again if the total energy remained somewhat constant during the trajectory. Then, check the spin which is labeled "mag=" at each ionic step in the OSZICAR file. The spin should start decreasing when the H-atom is at a distance of around 2 Åand approaching the slab. When the spin reaches zero, it means the electron of the H-atom has affiliated itself with bulk electronic states. Of course, in reality the spin is always in integer but that's just how DFT rolls, do not worry about it too much. The crucial part is scattering the H-atom from the slab.

The spin is zero when the H-atom is close to or inside the slab. Now if it gets scattered back, it will have to recover its original spin state. From my experience, this is not always the case. I encountered trajectories in which the overall spin of the system remained zero when the H-atom reached its initial height of six angstroms above the surface. What was happening there was that the electron density around the H-atom consisted of half spin up and half spin down. In consequence, the potential energy of the system was about 1 eV too high since zero interaction energy is defined as the H-atom being in a doublet state six angstroms about the surface. What VASP does there is to take the electron density from the previous ionic step and use it as an initial guess for the next step so that it does not need to optimize much and speed up the simulation. In doing so, it gets stuck in a local minimum electronic configuration having an unreasonable potential energy. The only solution I have found so far is to always solve the electronic structure problem from scratch at each ionic step while the H-atom is regaining its spin. Then, the chance of the program ending up at the wrong energy is minimized but of course in exchange for a much longer run-time. Most importantly though, each ionic step is independent from both its predecessors and successors which facilitates the identification of convergence problems. A python script which automatically stops and restarts VASP to get the spin right can be found in Ch. 5.0.1.

2.5 Fitting the Potential Energy Surface with EMT

2.5.1 The parameters

The Effective Medium Theory developed by Nørskov *et al.* [15, 16] has seven parameters per species which are related to the physical properties of the species.
There are η_2, n_0, E_0, λ, V_0, κ and s_0. s_0 is the neutral sphere radius. It is calculated (Eq.) as follows from the lattice constant a_0 determined in the bulk-calculations:

$$s_0 = \frac{a_0}{\beta' \sqrt{2}} \tag{2.5}$$

Here, β' is a geometrical factor for the fcc-crystal:

$$\beta' = \frac{\sqrt[3]{16\pi/3}}{\sqrt{2}} \tag{2.6}$$

```
SYSTEM = Slab Equilibration
PREC = Accurate     ! use your setting here
ENMAX = 400         ! use your setting here
ALGO = Very Fast    ! fastest if close to optimum, use for MD
ISTART = 0
ICHARGE = 2
NELM = 300
EDIFF = 1.0E-05
EDIFFG = -1.0E-03
NELMIN = 4          ! ensures convergence
ISYM = 0
SMASS = -3          ! use microcanonical ensemble
TEBEG = 600         ! twice the desired temperature
LREAL = Auto
NSW = 1000          ! enough time even for the heaviest metals
ISIF = 0            ! only calculate what's necessary
IBRION = 0          ! a Verlet algorithm
POTIM = 0.5         ! time step in fs (works for 3d metals and heavier)
ISMEAR = 1          ! Methfessel-Paxton for metals
SIGMA = 0.1         ! use your smearing here
GGA = PE            ! your functional
```

Figure 2.17: Exemplary INCAR-file for nonmagnetic surfaces to perform an equilibration.

```
NPAR = 2            ! square root of # CPUs
PREC = Accurate     ! use your setting here
ENMAX = 400         ! use your setting here
ALGO = Very Fast    ! use this for MD
ISTART = 0
ICHARGE = 2
NELM = 300
EDIFF = 1.0E-05
EDIFFG = -1.0E-03
NELMIN = 4          ! ensures convergence
ISYM = 0
SMASS = -3
ISPIN = 2           ! turn on spin of H
MAGMOM = 24*0 1     !  and possibly also of the slab
AMIX = 0.3          ! use
BMIX = 0.0001       !  these
AMIX_MAG = 0.3      !  settings
BMIX_MAG = 0.0001   !  for
MAXMIX = 20         !  mixing
LREAL = Auto
NSW = 1200          ! 120 fs is enough
ISIF = 0
IBRION = 0
POTIM = 0.1         ! use 0.1 when an H-atom is involved
ISMEAR = 1          ! Methfessel-Paxton for metals
SIGMA = 0.1         ! use your smearing here
GGA = PE            ! your functional
```

Figure 2.18: Exemplary INCAR-file for an AIMD trajectory.

E_0 is the cohesive energy (sublimation energy) which can be found in literature. Together with λ, E_0 and s_0 are directly related to the bulk modulus:

$$B = -\frac{E_0\lambda^2}{12\pi s_0} \tag{2.7}$$

s_0 is furthermore related to the shear modulus, together with κ, η_2 and V_0:

$$C_{44} = \frac{3V_0\kappa\delta}{8\pi s_0} \tag{2.8}$$

As well as the other elastic constants:

$$C_{11} = \frac{3V_0\delta\kappa - E_0\lambda^2}{12\pi s_0} \tag{2.9}$$

$$C_{12} = \frac{3V_0(\kappa - \beta'\eta_2)\kappa - 2E_0\lambda^2}{24\pi s_0} \tag{2.10}$$

with

$$\delta = \beta'\eta_2 - \kappa \tag{2.11}$$

n_0 is a parameter relating to the individual density of the species and responsible for the coupling between species.

Recommendations for the parameters

Parameters for the metal (Me)

All the parameters apart for $E_{0,Me}$ should have positive values. $s_{0,Me}$ and $E_{0,Me}$ should be kept constant during the fit to reproduce the most important bulk properties. If you varied $s_{0,Me}$, you would also change the lattice constant of your metal and that does not make very much sense since you spent a lot of time actually finding the right one. Varying $E_{0,Me}$ will mostly destabilise the fit anyway, but it will also lead to PES which melt at very low temperatures (if $E_{0,Me}$ is too small) or not at all. So, it's not advisable to change this parameter. λ_{Me} is a parameter you can keep fixed during the fitting procedure so the experimental bulk-modulus is reproduced. Actually fitting it might improve the fit slightly, but then, you won't reproduce the bulk-modulus as accurately anymore. Furthermore, you should keep κ_{Me}, $\eta_{2,Me}$ and $V_{0,Me}$ such that they roughly reproduce the

shear-modulus. For Au, C_{44} (without s) should be $C_{44} \geq 10^{10}$ and best $C_{44} \geq 1.2 \cdot 10^{10}$. The melting temperature of the slab is in some obscure way related to C_{44}. I haven't figured out in which way, yet, but if your shear modulus gets too small (smaller than the numbers I recommend here), the surface will start disintegrating at unreasonably low temperatures.

As far as I've seen it, there aren't any issues related to $n_{0,Me}$..

Parameters for Hydrogen

Table 2.2: Recommendations for the metal parameters.

Parameter	Relation to metal properties	Recommendation
s_0	$s_0 = \frac{a_0}{\beta'\sqrt{2}}$	fix
E_0	Cohesive/Sublimation Energy	fix
λ	$B = -\frac{E_0\lambda^2}{12\pi s_0}$	(fix)
η_2, κ, V_0	$C_{44} = \frac{s \cdot V_0 \cdot \kappa(\beta'\eta_2 - \kappa)}{8\pi s_0}$	restrain
n_0	density	none

There is no experimental data for metallic hydrogen that I know of. I therefore recommend to use the values suggested by Strömquist *et al* [17] as a first guess but try to vary all of them during the calculations.

In general, you should not fit all hydrogen parameters at once, because that will destabilise the fit; if you are fitting $s_{0,H}$, $E_{0,H}$ should not fitted simultaneously and vice versa. The values suggested by Strömquist *et al* for $s_{0,H}$ and $E_{0,H}$ result in good fits, so you do not necessarily need to fit them. If you decide to do so, for fitting $s_{0,H}$, I would recommend to set a guess-$s_{0,H}$ before you start the actual fitting. The estimated lattice constant for metallic hydrogen is around 0.88 Å resulting in $s_{0,H} = 0.49$ Å [18], you should take note, though, that lowering $s_{0,H}$ also reduces the density the H-atom feels. In the nonadiabatic calculations, this would decrease the friction coefficient.

If you are fitting $s_{0,H}$ or $E_{0,H}$, I would first fit them with some of the other parameters and after that perform another fit in which you take the new values for $s_{0,H}$ or $E_{0,H}$ from your previous fit, but optimise the other parameters separately.

```
   eta2=             5.4254d0
     n0=             0.182205d0
     E0=            -2.37100000
 lambda=             7.72709d0
     V0=             0.427d0
  kappa=             8.86282d0
     s0=             0.68041100
```

Figure 2.19: Exemplary file for the H-parameters. File contains the values suggested by Strömquist *et al.* Please don't change the order of the parameters.

```
projectile H 1.0079 7 'fitdata/emt573_H.nml' ver 0
lattice Au 196.96657 7 'fitdata/emt573_Au.nml' ver 0
pes emt
celldim 2 2 4 x
rep 1 1
conf fit au111_2x2x4.POSCAR 10 987
aimd 700 200 0.00
evasp -24.995689d0  ! A value for Au 2x2
fitconst 14 1 2 3 4 5 6 7 8 9 10 11 12 13 14
maxit 1
```

Figure 2.20: Exemplary input-file for the fit in the MD_tian Programm.

2.5.2 Performing the fit

The input-file (cf Fig. 2.20) for the fit begins with the specification of the two species fitted. 'projectile' classifies the projectile, in our case, the hydrogen atom. 'lattice' specifies the metal slab. The flag is followed by the name of the projectile, its mass, the number of parameters, the file-name of the file that contains the starting parameters, the algorithm and the number of layers that should be kept fixed for eventual MD-simulations. The last two variables can be safely ignored for fitting. Next, 'pes' declares for which theory the PES should be fitted. 'celldim' records the dimensions of your input-cell as described

Table 2.3: Number of parameter for fitting.

No.	Parameter	No.	Parameter
1	$\eta_{2,H}$	8	$\eta_{0,metal}$
2	$n_{0,H}$	9	$n_{0,metal}$
3	$E_{0,H}$	10	$E_{0,metal}$
4	λ_H	11	λ_{metal}
5	$V_{0,H}$	13	$V_{0,metal}$
6	κ_H	12	κ_{metal}
7	$s_{0,H}$	14	$s_{0,metal}$

for the optimisation of the H/Metal-system. 'rep' organises how many times your cell should be replicated to produce a cell that is fit for EMT-usage. 'rep 1 1' is a reasonable choice. 'conf' determines if the program is going to be doing MD-simulations or fitting. For fitting, 'conf' should be followed by 'fit', the name of the POSCAR-file that contains your equilibrium structure with the H-atoms 6.0 Å above the surface, the number of the AIMD-trajectory you would like to fit (in this case 10) and the number of the fit (in this case 987).

'aimd' controls the AIMD-contribution. The flag is followed by the number of Equilibrium-DFT-Points you would like to use, then by the number of AIMD-DFT-points and finally by de_aimd_max which determines how the AIMD-points should be chosen. If it is set to 0.0, then the points will be chosen in regular intervals from the trajectory. For larger values, a point is only then used in fitting if its energy-difference to the preceding point is larger than de_aimd_max.

'evasp' controls the reference energy. It is followed by the energy a hydrogen atom has at 6.0 Å above an equilibrated Au-surface (see previous section for determination of the reference energy).

'fitconst' determines which parameters should be kept fixed. Thus, the integer behind it is the number of fixed parameters in the fit, followed by the numbers corresponding to the fixed parameter. The numbers are assigned as follows (Tab. 2.3).

'maxit' Maximal number of iteration. Followed by the maximal number of iterations.

100 is a good choice here.

Recommendations for the fitting

Use a ratio of 5:2 between your Equilibrium- and AIMD-DFT Points. This appears to give good results. But we should do more detailed testing here.
Some AIMD-trajectories will work better for the fit than others. In my experience, trajectories in which the hydrogen atom spends a long time in the subsurface are not quite as suited as single- or double bounce trajectories. In our fitting procedure, you can decide between either taking regularly spaced points from the AIMD-trajectory (de_aimd_max = 0.0) or only such points that differ by a certain energy from the previous point (de_aimd_max = x). In my experience, taking equally spaced points works better. Also, you can decide points up to a certain energy should be taken into account (e_max). In the program, this value is set to 20 eV which I found to be suited best.

2.5.3 After the Fit

After the fit, the physically reasonable behaviour of the fit and its convergence needs to be determined. To check the convergence, calculate the rms to the input-data and the rms to the remaining AIMD-trajectories. If those values are both below 190 meV, the fit can be considered a candidate.

Checking the Physical Behaviour of the Fit

The first check is calculating C_{44} (cf Eq. 2.8). If the fit's $C_{44} \ll s \cdot 1.2 \cdot 10^{10}$ the metal-surface will probably disintegrate at a too low temperature. You can test this by running MD-trajectories for different temperatures which propagate the motion for the metal-surface for several pico-seconds and by checking if the metal-slab still retains its shape as a surface after the trajectories.
Furthermore, you should also check if the H-atom can abstract metal-atoms from the surface. For that, you calculate the energy an H-atom has at 6 Å above the surface. Then do the same calculations, only additionally to the H-atom at 6 Å, remove a metal-atom

Table 2.4: The arts of metal-H formation.

Metal	$\Delta_f H^\circ(Au_{s\to g})$(eV)	$\Delta_f H^\circ(Au-H)$(eV)	bond length (Å)	ζ (eV)
Au	3.8[19]	2.99 [20]	1.52 (CRC)	0.8
Cu		2.53 [20]		
Ag		2.19 [20]		
Ni		2.58 [20]		

from the surface and place it at the distance of the H-metal-bond above the H-atom. With that I mean: Form a vacancy in the surface and use the atom from this vacancy to create a metal-H-molecule that is also 6 Å above the surface.
Calculate from the difference the energy ζ that is needed to form an H-metal molecule from a surface atom and an H-atom. We should probably implement a subroutine into the fitting procedure that does that automatically at the end of the fit.

$$\zeta = \Delta_f H^\circ(Au_{s\to g}) - \Delta_f H^\circ(Au-H) \tag{2.12}$$

Compare the value to experimental values. ζ can be calculated by subtracting the metal-H-formation energy from the sublimation energy of the metal (cf Eq. 2.12). The value obtained for your fit should not be much below it.

2.5.4 Possible issues

If the fit has problems converging, you can first try fitting to the equilibrium-GGA-DFT-points and use the resulting parameters as input in a new fit where you then include both the equilibrium-DFT-points and the aimd-DFT-points. Last but not least: form a circle of salt around your computer before starting the fitting and sacrifice a few drops of water to Thoth.

3 Skycruiser

Skycruiser is Genetic Algorithm wrapper script to the md_tian program. It fully automatizes the fitting of a Potential Energy Surface. Without this script, the only part in the entire procedure that lacked a systematic approach is the actual tuning of the EMT parameter set itself. The problem is the following: on the one hand, md_tian uses a gradient descend method to minimize the root mean square error between EMT and DFT reference energies. On the other hand, there are seven parameters per atomic species in EMT which enable the PES to be applicable to a large variety of systems, but even the most sophisticated nonlinear least squares fitting routine does not perform well in such a high dimensional environment. In the best case, it simply does not find the global minimum, but most of the time, the routine yields a PES which is far from being physically reasonable. It might, for example, not fulfill the elastic stability criteria or let the metal disintegrate at a ridiculous temperature, among other exasperating problems.

Skycruiser a systematic approach to this multidimensional optimization problem. It extends the goal from reproducing DFT energies by requiring compliance to metal properties like the bulk and shear moduli and the cohesive energy alongside metal-hydrogen interaction characteristics like the vacuum bond dissociation energy. A Genetic Algorithm seems well-suited for this task since one of its inherent features is a fitness function that combines all of these measures of quality into a single number. Skycruiser can run md_tian in parallel with different initial EMT parameters. In this approach, a large share of parameter space can be explored in a productive fashion. The algorithm shoots off several hundred fits in each generation and those, which give physically meaningful PESs and also reproduce the interaction energies, are possible candidates for the successful simulation of atomic beam scattering experiments. In the algorithm, PESs are assigned a fitness value after each local optimization and mix their EMT parameters to form the subsequent generation. By doing so, new initial parameter sets are generated

with each individual and optimized sets are passed on, which reproducibly leads to fitness convergence after roughly one hundred generations.

In this optimization problem, the global minimum is not (necessarily) found because of two reasons. First, search space is simply to large to be explored exhaustively. This is why a heuristic method like a Genetic Algorithm is appropriate. Second, and much more general, the objective function to minimize is not really objective. It is an empirical expression that condenses all quality criteria into a single number. Different contributions might be weighted differently and since so many of them need to be taken into account, there are a lot of possible paths to take.

3.1 Folder structure

This section describes the files and folders that need to present to use the Skycruiser framework. You will need two files that contain the initial parameters (one file for the projectile, one file for the metal). They need to be in a format that is readable to the md_tian program. Also, you will need the "Eq_points_dft.dat" and "energy_eq.dat" files which are mandatory to do a fit in md_tian. Next, the relaxed slab geometry must be in a file in which the lattice constant (factor in 2nd line of VASP POSCAR file) has been multiplied into the cell vectors and atomic coordinates, so that it only contains plain cartesian coordinates. You will also need a folder which contains the AIMD trajectories. You can name it however you want but inside it, there must be folders of the format "traj%03d". Inside each of these folders, there must be two files. One called "analyse.dat" which contains the propagation time of the trajectory in fs in the first column and the "E0=" values of each ionic step from the OSZICAR file in the third column. The other file is called "XDATCAR.dat" and is the XDATCAR file produced by VASP (comb_XDATCAR if you are using Sestra). The only necessary modification you need to make is to replace all lines starting with "Direct configuration" with a blank line.

```
geometry_file = /home/mkammle/Calculations/Nickel/Fit2/ni111_2x2x6.POSCAR
fitdata_folder = /home/mkammle/Calculations/Nickel/Fit2/traj_folder
projectile_fitparams_file = /home/mkammle/Calculations/Nickel/Fit2/emt_stroem_H.nml
lattice_fitparams_file = /home/mkammle/Calculations/Nickel/Fit2/emt_stroem_Ni.nml
energy_eq = /home/mkammle/Calculations/Nickel/Fit2/energy_eq.dat
eq_points_dft = /home/mkammle/Calculations/Nickel/Fit2/Eq_points_dft.dat
md_tian_path = /home/theory/mxt/source/work_with_Skycruiser/md_tian_1021

projectile_element = H
projectile_mass = 1.0079
lattice_element =  Ni
lattice_mass = 58.693
celldim = 2 2 6 x
rep = 1 1
evasp = -127.3035325
lit_bulk_mod = 180
lit_E0_metal = 4.44
lit_C44 = 122
lit_V0H = 0.427
lit_zeta = 1.44
static_lattice_points = 749
trajectories = 001 002 003 004 005 006 007 008 009 010 011 012
always_include_sites = 1 2 3 4 5 6 7 8 9 10

nthreads = 64
population = 300
maxit = 30
last_spawned_creature = 60000
keep_fix = 10 11 14

param_constraint = 2 40 20
param_constraint = 7 15 20
param_constraint = 9 40 20
```

Figure 3.1: Exemplary input.sky file to use within the Skycruiser framework.

3.2 Input file

When this setup is complete, you need to tell Skycruiser where to find all these files and what run-time parameters to use. This is the purpose of a file called "input.sky". All tags have to be separated from their arguments by an equals sign. An exemplary input file is shown in Fig 3.1. Mandatory and optional tags are

geometry_file must be the absolute path to the relaxed slab geometry file.

fitdata_folder must be the absolute path to the folder which holds the different AIMD trajectories in folders with the names "traj%03d" of which the trailing integer part must be provided by the "trajectories" tag.

md_tian_path must be the absolute path to the md_tian executable.

energy_eq must be the absolute path to the "energy_eq.dat" file.

eq_points_dft must be the absolute path to the "Eq_points_dft.dat" file.

projectile_fitparams_file must be the absolute path to the file holding the initial parameter set of the projectile.

lattice_fitparams_file must be the absolute path to the file holding the initial parameter set of the surface.

projectile_element must be the chemical symbol of the projectile.

projectile_mass must be the mass of the projectile in atomic mass units.

lattice_element must be the chemical symbol of the lattice.

lattice_mass must be the mass of a lattice atom in atomic mass units.

celldim must be the dimensions of the provided slab geometry file in $<x>$ $<y>$ $<z>$ dimensions.

rep must be the repetition of the provided slab geometry file in x and y direction.

evasp must be the overall reference energy meaning the "E0=" value of the DFT grid in which the H-atom is six angstroms above the surface, the slab is fully relaxed and spin is taken into account.

trajectories must contain the "%03d" part of the folders containing the AIMD trajectory information. See Fig 3.1 for more details.

static_lattice_points must contain the number of configurations in your DFT energy grid.

nthreads can be the number of threads you want to use, defaults to one.

population can be the number of individuals per generation, defaults to one hundred.
maxit can be the maximum number of optimization steps in the least squares fitting routine, defaults to 50.
last_spawned_creature can be the number of the individual for which the last generation will be created, defaults to one million.
fuzzy_input can be the standard deviation of a normal distribution from which a number is drawn for each initial EMT parameter separately when the expectation value is the one provided in the projectile_fitparams_file or lattice_fitparams_file and the parameter is not being held constant. More precisely, the probability density is

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

with μ as the value given in the parameter file, and σ as the number given in this tag in percent of μ (defaults to zero).
keep_fix can contain the parameters in the format "%d" which Skycruiser must not change during the fit. 1-7 correspond to projectile parameters and 8-14 belong to the metal.
lit_bulk_mod can contain the literature (target) bulk modulus. Additionally, the penalty-free deviation from this value can be specified as a second argument in percent of the first argument and defaults to 20. The penalty increase for each percent additional deviation from the bound given in the second argument can be set via the third argument and defaults to 10. All penalties in the program are determined by these three numbers and have the same general shape. See Fig. 3.2 for more details.
lit_C44 has the same input structure as "lit_bulk_mod" but constrains the shear elastic constant.
lit_zeta has the same input structure as "lit_bulk_mod" but constrains ζ defined in Eq. 2.12.
lit_bondlength has the same input structure as "lit_bulk_mod" but constrains the *in vacuo* M-H bond length.
energy_min can contain the minimum interaction energy that shall be considered during the fit, defaults to -20.
energy_max can contain the minimum interaction energy that shall be considered during the fit, defaults to 20. This is to exclude absurd energies from the fit which result

from the H-atom being very close to a surface atom in the DFT grid.
always_include_sites can contain the symmetry sites of the DFT grid that will always be included in the fit in the format "%d". The sites not included here will be treated as part of the genome and will be subject to crossover and mutation.
min_dist_to_metal can contain a lower limit for the H-metal atom distance and defaults to 0.1. If the H-atom comes closer to a surface atom than given by this tag, it will not be used for the fit. This excludes very large interaction energies from the fit which might occur in the DFT grid.
max_dist_from_surface can contain a number that describes the maximum distance between the H-atom and the surface and defaults to 6. If the H-atom is further away, any configuration/energy pairs will not be considered in the fit.
aimd_weight is a number between zero and one, defaults to 0.5, and determines the contribution of the validation data set to the fitness of the individuals. The basis for the fitness F is the sum of the inverses of the RMSEs to the training and validation data set

$$F \propto \frac{\alpha}{\mathrm{RMSE_{vd}}} + \frac{(1-\alpha)}{\mathrm{RMSE_{td}}}.$$

As a default, both contributions are weighted equally. With this tag it is possible to unbalance the weights, more precisely, α can be directly set via this tag.
check_esc can be set to "False" to disable the verification of the elastic stability criteria which is activated by default

$$C_{11} > 0, \qquad C_{44} > 0, \qquad C_{11} > |C_{12}|, \qquad (C_{11} + 2C_{12}) > 0.$$

param_constraint can be set to directly apply the penalty function to EMT parameters. The first number (1-14) describes the parameter which shall be considered. The second and third number work the same way as described in the "lit_bulk_mod" tag paragraph. Every parameter needs to be constrained in a separate tag as shown in the last lines of Fig. 3.1.

3.3 Output files

To monitor how the fit performs during run-time and also to extract good fits at the end of a run, Skycruiser prints to STDOUT and to several files. Screen output includes

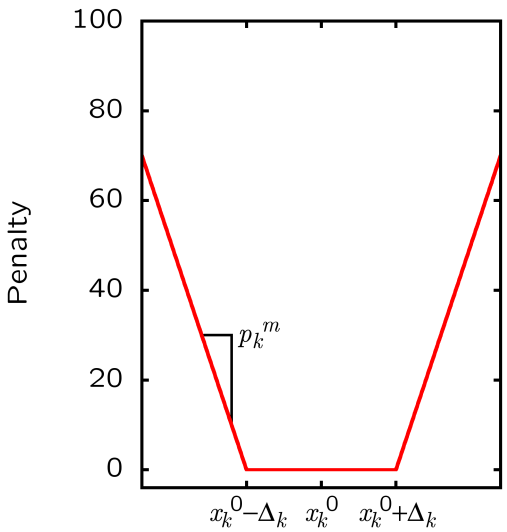

Figure 3.2: General shape of the penalty applied to a constrained value. x_k^0 is the target value, Δ_k is the penalty-free region always specified in percent of x_k^0, and p_k^m is the penalty per additional percent deviation from Δ_k.

information about the creation and evaluation of individuals. Much more important, though, are the files in the "Stats" folder. Note that these files are not synchronized with the progress of the program. Since it launches its threads asynchronously, an evaluation of a generation as a whole may only be performed once all threads have come to a halt.
aimd_contrib.txt keeps track of the number of non-equilibrium configurations used in the training data. Each column represents a 10% interval and one row gets created for each generation. Since Skycruiser handles the AIMD contribution to the training data set as a fraction of the total number of configurations present in the chosen trajectory, only relative values w.r.t this trajectory are displayed in this file. Absolute number are less interesting because trajectories may be based on different propagation times. For example, 37 in the second row, first column means, that 37 individuals in the second generation had 0-10% of the configuration/energy pairs of the assigned trajectory included in the training data.
creature_overview.txt is probably the most important file in the output directory. It allows to delete the Population folder after the program has finished without losing

essential information. This file contains all necessary information about all individuals including but not limited to their ID, fitness, RMSE and final EMT parameters.

energy_grid_sites.txt has a structure similar to "aimd_contrib.txt". Each row represents a generation and each column symbolizes one symmetry site. A 180 in the second row, third column means that 180 individuals in the second generation have symmetry site three included in their training data set.

held_const.txt has a structure similar to "aimd_contrib.txt". Each row represents a generation and each column symbolizes one EMT parameter. The first seven integers belong to the projectile and the last seven to the metal. Each number describes how many individuals have held this parameter constant during the fit in this generation.

stats.txt is the best file to monitor the convergence of the algorithm by plotting the generation against the highest creature fitness therein for example. A quick "cat stats.txt | sort -k4" displays the highest fitness together with the creature ID at the lowermost line. Also, this file shows the mean fitness of each generation and its standard deviation.

traj_fit.txt has a structure similar to "aimd_contrib.txt". Each row represents a generation and each column symbolizes one AIMD trajectory. The numbers tell you how many times non-equilibrium geometries were take from that trajectory in a generation. For example, a 22 in the third row in the second column means that 22 individuals in the third generation used configuration/energy pairs from the second AIMD trajectory in their training data set.

out.txt are files produced by Skycruiser which are not located in the "Stats" folder but in the individuals' folders in the "Population" folder. They contain the STDOUT output of md_tian and are not essential for the analysis of a Skycruiser run, but for the Skycruiser program itself. It reads this file to learn how the local optimization routine has performed and to extract numbers of interest.

sky.log is located in the working directory of Skycruiser. It contains all run-time parameters from "input.sky", AIMD trajectory information and errors encountered during execution.

4 Function of the single modules and subroutines

4.1 atom_class

This model contains all parameters that cannot be changed during the execution of the program as well as some default values. This includes commonly used irrational numbers and physical constants and unit-conversion factors. Furthermore, the object atoms and its allocation defined which holds information such as the positions, velocities, forces &c about the particle or the lattice.

Also, the program's basic units are defined. They are:
Length: Å
Time: fs
Energy:
The program-derived units are:
Mass: $\mathrm{eV \cdot fs^2/Å^2} = 1/103.6382\,\mathrm{amu}$
Angle: radian = 180°
bohr: 0.5291772 Å

4.2 dtrnlspbc

describe

4.3 fit4tian

This is the interface for the fitting routine. It contains the following subroutines: **fit** Here, the fit is prepared. The parameter files and the input into the fit are formed and information on the beginning and end of the fit are printed out. The non-linear least square algorithm with boundary constraints is prepared and called.

dev2eqdft

Generates the dev2Eqdt$n$.dat output file of the fit. This file relates to the 3D-grid and for a fixed surface samples the ten surface sites starting from 6 Å above the surface with whatever step and to whatever depth is defined in the Eq_points_dft.dat-file. Each position is followed by the potential energy calculated with the fit's parameter set at surface atoms frozen to their relaxed position for that position of the particle. **denseqdf** This routine is almost identical to the above, only instead of the potential energy, the denseqdft$n$.dat-file contains the background electron density at these positions. **dev2aimddft** This routine produces the traj$n$_1.dat file. This file is related to the AIMD-trajectories that are used in the input-data set and control data set for the fit. It contains the step of each trajectory followed by the potential energy from the DFT-simulations and then by the potential energy calculated with the fit parameters from the positions corresponding to the step.

Furthermore, information on the fit (number, shear modulus, the H-metal bond formation energy, rms-error to the testing set (it consisting of all the AIMD-trajectories) and the parameter values) are printed into the c44rms.dat-file and information on the fit (rms-error to the testing set, elastic moduli) are printed out.

readinaimd

This routine is called up by the dev2aimddft-routine (see Sec. 4.3) and reads in the positions and energies of the XDATCAR.dat and analyse.dat files, respectively, of the AIMD trajectories. It proceeds to produce a 6×6 surface cell from the data so the EMT-routine can be used to calculate the energy from the AIMD positions.

restricted_fit

restricted_fit

4.4 Force

This module deals with the calculation of energies, forces and analytic derivatives of the PES usable in the program as well as the routine that calculates the friction coefficient from the local density. In general, the subroutines fall into three classes. The ones with a '1' at the end of the name deal with the calculation of only one species. Those with '_e' at the end calculate only the energy without the forces and the ones with 'd' (e.g. de or ddens) also includes routines to calculate the derivatives with respect to the parameters (see Tab. 4.1).

In the LDFA subroutine, the friction coefficient is calculated from the local background electron density. The variable containing the local background electron density is then overwritten by the friction coefficient (in fs^{-1}). In case of simulated annealing a general friction coefficient of $3\,\mathrm{ps}^{-1}$ is assumed.

4.5 md_init

This module is a monster. Tab. 4.2 gives an overview over the subroutines in the module and their purpose. In general, this module serves as an initialization module for (almost) everything that goes into the program. Due to this reason, it starts with a large block of global variable that need to be used in other modules as well.

4.6 md_tian

This is the main program-file that calls up all other routines. It starts with calling the simbox_init-routine to initiate the program and then calculates the reference energy. According to *confname*, either fit or MD-routine are started. The first is started by calling up **fit4tian**, in the second case, the loop over the trajectories starts in the main program with initiating the species, followed by a loop over the time steps that calls up the propagators in **mdalgo** for the propagation of the trajectory. The routines for the output are also called here. Furthermore, the collectors for the number of bounces, the collector (pEfric) for the energy loss to phonons (when md_algo_p = 3 or 4 (Langevin dynamics)) or to calculate electronic friction post facto (pEF, when md_algo_p = 5

(Verlet with pEF)), the collector for the lowest particle position and the routine that stops a trajectory when the particle has moved through the slab or scattered further back than 6.1 Åfrom the surface are to be found here.

MD-simulations are either the calculation of trajectories or simulated annealing. The decision between the former and latter is controlled by the variable *sasteps* that also controls when the surface temperature during the simulated annealing is to be lowered or raised.

Table 4.1: Subroutines in the force-module and what they calculate (BED = background electron density).
(1) only next neighbors included.
(2) calculated numerically.
(3) calculates the friction coefficient from the BED.

name	No. species	energy	forces	BED	parameter derivatives	system
emt	2	×	×	×		EMT
emt_e	2	×		×		EMT
emt1	1	×	×	×		EMT
emt1_e	1	×		×		EMT
emt_e_fit	2	×				EMT
emt_de_fit	2	×		×	× (to energy)	EMT
emt_dens_fit	2	×		×		EMT
emt_ddens_fit	2	×		×	× (to density)	EMT
emt1nn	2	×	×	×		EMT(1)
num_emt	2	×	×(2)	×		EMT
num_emt1	1	×	×(2)	×		EMT
ldfa	1					LDFA(3)
lj_e	2	×				Lennard-Jones
lj	2	×	×			Lennard-Jones
lj1_e	1	×				Lennard-Jones
lj1	1	×	×			Lennard-Jones

The number of bounces is calculated by finding maxima in the background electron density. The reasoning behind this is that if a particle gets close to a surface atom, the electron density must rise; it will peak when the particle is closest to the surface atom. To avoid double-counting of maxima due to fluctuations in the BED, a bounce is only counted if the minimum between two maxima is larger than $0.025\,\text{Å}^{-3}$. This is not a perfect way to count the number of bounces, but the best we could come up with so far.

At the end, there are some lines commented out that make it possible to calculate the timing of the program if they are commented in again.

4.7 mdalgo

This routine contains the propagation algorithms divided into their first (propagator_1) and second step (propagator_2). Those algorithms are: the Verlet Algorithm [14], the Beeman-Refson Algorithm [13], the Langevin algorithm [14] with exact formulae and with approximate formulae (langevin and langevins). We have tried both exact and approximate algorithm, but our general recommendation is to use the exact Langevin algorithm. Lastly, the module contains a routine that calculates the accelerations from the forces.

Table 4.2: Subroutines in the md_init module and their purpse.

name	purpose
simbox_init	read input file, calls up dependent on *confname*
read_conf	
prep_slab	
prep_teil	
read_mxt	
read_fit	
traj_init	
particle_init	

4.8 model and nllsq.f

These modules is obsolete and should no longer exist; they are the first routines we used to calculate the non linear least square fitting routines. If you have them, you may keep them for reasons of nostalgia, but we would recommend not using them. Or if you have no Intel Fortran Compiler, but then, you will need to modify the code to include them (which you naturally may).

4.9 open_file

This module contains the subroutines responsible for reading, writing and appending to files. Those subroutines are meant to facilitate the opening of files and print corresponding error messages that make file-opening errors more comprehensible.

4.10 output

Here, the output-files are written. Please see also Sec. 1.4.3 and for the files that each subroutine gives rise to Tab. 4.3. The only routine in this module that does not lead to any output files is the *sartre*-function which checks if a trajectory has been calculate before. It skips all but the last already calculated trajectory and recalculates the last before it continues with the next.

Table 4.3: Subroutines in the output module and the output files they produce.

name	file name	file size
full_conf	mxt_conf$n$.bin	medium
out_short	mxt_fin$n$.dat	small
out_detail	mxt_trj$n$.dat	medium
out_all	mxt_confxyz.dat	large
out_poscar	mxt_anneal.POSCAR	small
out_posvel	mxt_rv$n$.dat	large
out_pdb	mxt_conf$n$.pdb	medium

4.11 useful_things

This module contains useful subroutines returning random numbers, a normal function, turning input into lower-case characters, calculating the normed distance between two vectors, calculating the kinetic energy, periodic boundary conditions and a routine that counts the lines in a file.

5 Appendix

5.0.1 Sestra

```
### VERSION 0.11 ###

import sys, subprocess, time, shutil, re, math, time, os

### START THIS BY EXECUTING "./Sestra.py <nProcessors>" ###

### THESE NEED TO BE EVALUATED AND SET FOR EACH SYSTEM ###
SPIN_ZONE_UPPER_LIMIT = 3.0
SPIN_ZONE_LOWER_LIMIT = 1.6


def slog(s):
    sestra_log.write("[" + time.strftime("%Y-%m-%d %H:%M:%S") + "] " + str(s) +
        "\n")

def stop_VASP(phandle):
    print "VASP will be stopped after this iteration"; slog("VASP will be
        stopped after this iteration.")
    stopcar = open("STOPCAR", "w")
    stopcar.write("LSTOP = .TRUE.")
    stopcar.close()
    while phandle.poll() is None:
        time.sleep(2)
    print "VASP stopped"; slog("VASP stopped")

def start_VASP(np):
```

```
    print "VASP started with", np, "processors"; slog("VASP started with %02d
        processors" % int(np))
    phandle = subprocess.Popen(["mpirun", "-n", np, "vasp"])
    return phandle

def restart_VASP(curr_iter, np, OSZICAR_STEP):
    print "Restarting VASP"; slog("Restarting VASP")

    xdat = open("XDATCAR", "r")
    for line in xdat:
        pass
    old_proj_pos = float( line.strip("\n\r\t ").split()[2] ) * LATTICE_CONSTANT
        * CELL_IN_Z
    xdat.close()

    oszicar = open("OSZICAR", "r")
    curr_oszicar_step = 0
    req_elec_steps = 0
    for line in oszicar:
        if "T=" in line:
            curr_oszicar_step = int(line.strip("\n\t\r ").split()[0])
        elif "RMM" in line:
            req_elec_steps = int(line.strip("\n\t\r ").split()[1])
    oszicar.close()

    if req_elec_steps >= VASP_NELM:
        print "[WARNING] ionic step %04d did not converge" % (OSZICAR_STEP +
            curr_oszicar_step)
        slog("[WARNING] ionic step %04d did not converge. Use CONTCAR_%04d to
            rerun this step." \
            % ((OSZICAR_STEP + curr_oszicar_step - 1), (curr_iter - 1)) + "
                Also check next step(s)." )

    shutil.copy2("CONTCAR", "POSCAR")
    shutil.copy2("CONTCAR", "CONTCAR_%04d" % curr_iter)
```

```
    shutil.copy2("OSZICAR", "OSZICAR_%04d" % curr_iter)
    shutil.copy2("XDATCAR", "XDATCAR_%04d" % curr_iter)
    phandle = start_VASP(np)
    OSZICAR_STEP += curr_oszicar_step
    slog("Restarted at step %04d" % OSZICAR_STEP)
    return phandle, old_proj_pos, OSZICAR_STEP


# initialize globals
sestra_log = open("sestra.log", "w")
OSZICAR_STEP = 1  # one-based b/c VASP is
VASP_ITERATION = 0
VASP_NSW = None
VASP_NELM = None
LATTICE_CONSTANT = None
CELL_IN_Z = None
SPIN_RELEVANT = True

try:
    os.remove("XDATCAR")
except OSError:
    pass
try: os.remove("OSZICAR")
except OSError:
    pass

# initialize number of processors
NPROCESSORS = "1"
try:
    NPROCESSORS = sys.argv[1]
except IndexError:
    print "You did not specify the number of processors that VASP is allowed to
        use."
    print "USAGE: python SestraX.XX.py num_proc\nContinuing serially..."
    pass
```

```
# check if extra spin safety is required
RUN_EXTRA_SAFE = False
additional_arg = None
try:
    additional_arg = sys.argv[2]
except IndexError:
    pass
if additional_arg == "-extra_safe":
    RUN_EXTRA_SAFE = True

# save initial POSCAR
shutil.copy2("POSCAR", "POSCAR_INITIAL")

# find lattice constant from second line in POSCAR file
try:
    fh = open("POSCAR", "r")
except IOError:
    sys.exit("No POSCAR file found."); slog("No POSCAR file found.")

for line in fh:
    line = fh.next()
    LATTICE_CONSTANT = float( line.strip("\n\r\t ") )
    line = fh.next(); line = fh.next(); line = fh.next()
    CELL_IN_Z = float( line.strip("\n\r\t ").split()[2] )
    break
fh.close()

# find maximum number of ionic steps
incar = open("INCAR", "r")
for line in incar:
    if "NSW" in line and VASP_NSW == None:
        VASP_NSW = int(line.split("=")[-1].strip("\n\r\t "))
        slog("VASP NSW read as %04d" % VASP_NSW)
    elif "NELM" in line and VASP_NELM == None:
```

```
        VASP_NELM = int(line.split("=")[-1].strip("\n\r\t "))
        slog("VASP NELM read as %04d" % VASP_NELM)
incar.close()

curr_oszicar_step = 0
new_proj_pos = None
proj_pos = None
going_up = False
phandle = start_VASP(NPROCESSORS)

while phandle.poll() is None:
    # wait for some work to be done
    #print ""   #; slog(" ")
    time.sleep(30)
    #print ""   #; slog(" ")

    # look at last projectile position
    line = None
    try:
        xdatcar = open("XDATCAR", "r")
    except IOError:
        print "Couldn't open XDATCAR"; slog("Couldn't open XDATCAR")
        continue

    try:
        for line in xdatcar:
            pass
    except NameError:
        continue

    try:
        new_proj_pos = float( line.strip("\n\r\t ").split()[2] ) *
            LATTICE_CONSTANT * CELL_IN_Z
    except (IndexError, AttributeError):
        if proj_pos == None:
```

```
            print "Couldn't read projectile position. This is expected if this
                is your first ionic step."
            print "Else, there's something wrong with your XDATCAR file."
            slog("Couldn't read projectile position. This is expected if this
                is your first ionic step.")
            slog("Else, there's something wrong with your XDATCAR file.")
        continue

    # compare to last projectile position
    if new_proj_pos != None:
        if proj_pos != None:
            if proj_pos - new_proj_pos < 0:
                going_up = True
            elif proj_pos - new_proj_pos > 0:
                going_up = False
            else:
                # H atom position has stayed the same. Continue.
                continue
            print "Going up", going_up; slog("Going up " + str(going_up))

        proj_pos = new_proj_pos + 0.0
        new_proj_pos = None

    if proj_pos != None:
        print "H atom position read as", proj_pos; slog("H atom position read
            as %6.5f" % proj_pos)


    # extract current OSZICAR step
    curr_oszicar_step = 0
    oszicar = open("OSZICAR", "r")
    for line in oszicar:
        if "T=" in line:
            curr_oszicar_step = int(line.strip("\n\t\r ").split()[0])
    oszicar.close()
```

```
print "Current ionic step", (OSZICAR_STEP + curr_oszicar_step)
slog("Current ionic step %04d" % (OSZICAR_STEP + curr_oszicar_step))
slog("VASP ITERATION IS %04d" % VASP_ITERATION)


## decide what to do ##
# trajectory finished
if 6.1 < proj_pos <= 7:
    print "Projectile above starting position. Finish!"
    slog("Projectile above starting position. Finish!")
    stop_VASP(phandle)
    slog("Job's done."); break

if (OSZICAR_STEP + curr_oszicar_step) >= VASP_NSW:
    print "Maximum ionic reached. Finish!"
    slog("Maximum ionic reached. Finish!")
    stop_VASP(phandle)
    slog("Job's done."); break

# if projectile transitions from spin-relevant region into bulk: turn off
    spin, then restart
if SPIN_RELEVANT and proj_pos != None and proj_pos < SPIN_ZONE_LOWER_LIMIT
    and not going_up:
    print "Transition from spin-relevant region into Bulk."
    slog("#########################################################")
    slog("#  Transition from spin-relevant region into bulk #")
    slog("#########################################################\n")
    SPIN_RELEVANT = False
    slog("Spin relevant is set to " + str(SPIN_RELEVANT))
    print "Spin relevant is set to", SPIN_RELEVANT
    subprocess.call(["sed", "-i", "s/ISPIN *= *2/ISPIN = 2/", "INCAR"])
    stop_VASP(phandle)
```

```
        phandle, proj_pos, OSZICAR_STEP = restart_VASP(VASP_ITERATION,
            NPROCESSORS, OSZICAR_STEP)
        VASP_ITERATION += 1
        continue

    # if projectile transitions from bulk into spin-relevant region: turn on
        spin, then restart
    if not SPIN_RELEVANT and SPIN_ZONE_LOWER_LIMIT <= proj_pos < 6. and
        going_up:
        print "Transition from bulk into spin-relevant region."
        slog("############################################################")
        slog("#  Transition from bulk into spin-relevant region #")
        slog("############################################################\n")
        SPIN_RELEVANT = True
        slog("Spin relevant is set to " + str(SPIN_RELEVANT))
        print "Spin relevant is set to", SPIN_RELEVANT
        subprocess.call(["sed", "-i", "s/ISPIN *= *2/ISPIN = 2/", "INCAR"])
        stop_VASP(phandle)
        phandle, proj_pos, OSZICAR_STEP = restart_VASP(VASP_ITERATION,
            NPROCESSORS, OSZICAR_STEP)
        VASP_ITERATION += 1
        continue

    # in danger zone
    if going_up and (SPIN_ZONE_LOWER_LIMIT <= proj_pos < SPIN_ZONE_UPPER_LIMIT)
        and not RUN_EXTRA_SAFE:
        print "Projectile is fleeing the Bulk and in spin-critical region."
        slog("Projectile is fleeing the Bulk and in spin-critical region.")
        stop_VASP(phandle)
        phandle, proj_pos, OSZICAR_STEP = restart_VASP(VASP_ITERATION,
            NPROCESSORS, OSZICAR_STEP)
        VASP_ITERATION += 1
        continue
```

```
    if RUN_EXTRA_SAFE and (SPIN_ZONE_LOWER_LIMIT <= proj_pos <
        SPIN_ZONE_UPPER_LIMIT):
        if not going_up:
            print "[EXTRA SAFE]: Projectile is approaching the Bulk in
                spin-relevant region."
            slog("[EXTRA SAFE]: Projectile is approaching the Bulk in
                spin-relevant region.")
        else:
            print "[EXTRA SAFE]: Projectile is fleeing the Bulk and in
                spin-relevant region."
            slog("[EXTRA SAFE]: Projectile is fleeing the Bulk and in
                spin-relevant region.")
        stop_VASP(phandle)
        phandle, proj_pos, OSZICAR_STEP = restart_VASP(VASP_ITERATION,
            NPROCESSORS, OSZICAR_STEP)
        VASP_ITERATION += 1
        continue

print "Trajectory finished."
slog("Trajectory finished.")
shutil.copy2("OSZICAR", "OSZICAR_%04d" % VASP_ITERATION)
shutil.copy2("XDATCAR", "XDATCAR_%04d" % VASP_ITERATION)
shutil.copy2("CONTCAR", "CONTCAR_%04d" % VASP_ITERATION)
VASP_ITERATION += 1
sestra_log.close()

# Merge OSZICARs
comb_oszicar = open("comb_OSZICAR", "w")
oszi_step = 1
for i in range(VASP_ITERATION):
    seq_oszicar = open("OSZICAR_%04d" % i, "r")
    for line in seq_oszicar:
        if "T=" in line:
            orig_step = line.strip(" \n\r\t").split()[0]
            # adjust ionic step
```

```
            line = re.sub(r"[0-9]* T=", str(oszi_step) + " T=", line)
            # adjust spacing at beginning of line if
                log10(orig_step)!=log10(oszi_step)
            for i in range(len(str(oszi_step))-len(orig_step)):
                line = re.sub(r"^ ", "", line)
            oszi_step += 1
        comb_oszicar.write(line)
    seq_oszicar.close()
comb_oszicar.close()

# Merge XDATCARs
comb_xdatcar = open("comb_XDATCAR", "w")
xdat_step = 1
written_xdatcar_header = False
for i in range(VASP_ITERATION):
    seq_xdatcar = open("XDATCAR_%04d" % i, "r")
    if written_xdatcar_header:
        for j in range(7):
            line = seq_xdatcar.next()
    for line in seq_xdatcar:
        if "Direct configuration" in line:
            written_xdatcar_header = True
            line = re.sub(r"[0-9]{1,}", str(xdat_step), line)
            line = re.sub(r"= *", "=", line)
            for j in range(5-int(math.log10(xdat_step))):
                line = re.sub(r"=", "= ", line)
            xdat_step += 1
        comb_xdatcar.write(line)
    seq_xdatcar.close()
comb_xdatcar.close()
```

Bibliography

[1] S. M. Janke, Ph.D. thesis, Georg-August University Göttingen, Germany, **2016**.

[2] G. Kresse, J. Furthmüller, *Physical Review B* **1996**, *54*, 11169–11186.

[3] G. Kresse, J. Furthmüller, *Computational Materials Science* **1996**, *6*, 15–50.

[4] G. Kresse, J. Hafner, *Physical Review B* **1993**, *47*, 558.

[5] G. Kresse, J. Hafner, *Physical Review B* **1994**, *49*, 14251.

[6] G. Kresse, M. Marsman, J. Furthmüller, *VASP the GUIDE*, University of Vienna, **2012**.

[7] G. Kresse, M. Marsman, J. Furthmüller, *VASP the GUIDE*, http://cms.mpi.univie.ac.at/vasp/vasp/vasp.html, **2015**.

[8] K. Levenberg, *Quart. Appl. Math.* **1944**, *2*, 164–168.

[9] D. W. Marquardt, *Journal of the Society for Industrial and Applied Mathematics* **1963**, *11*, 431–441.

[10] M. Kammler, Master's thesis, Georg-August University Göttingen, Germany, **2015**.

[11] M. Kammler, **2016**, Private Communications.

[12] W. Humphrey, A. Dalke, K. Schulten, *Journal of Molecular Graphics* **1996**, *14*, 33–38.

[13] K. Refson, *Physica B+C* **1985**, *131*, 256 – 266.

[14] M. P. Allen, D. J. Tildesley, *Computer Simulation of Liquids*, of *Oxford science publications*, Oxford University Press, **1989**.

[15] K. W. Jacobsen, P. Stoltze, J. K. Nørskov, *Surface Science* **1996**, *366*, 394–402.

[16] K. W. Jacobsen, J. K. Nørskov, M. J. Puska, *Physical Review B* **1987**, *35*, 7423–7442.

[17] J. Strömquist, L. Bengtsson, M. Persson, B. Hammer, *Surface Science* **1998**, *397*, 382–394.

[18] E. Wigner, H. B. Huntington, *J. Chem. Phys* **1935**, *3*, 764.

[19] D. L. Hildenbrand, W. F. Hall, *Journal of Physical Chemistry* **1962**, *66*, 754–755.

[20] A. Kant, K. A. Moon, *High Temperature Science* **1979**, *11*, 55–62.