lasMD: A hybrid molecular dynamics/two-temperature code for laser ablation simulations

Institute for Functional Matter and Quantum Technologies
(former ITAP)
University of Stuttgart
E. Eisfeld
Version 1

Abstract

The lasMD package is an extension of the IMD molecular dynamics package which was developed in the course of my PhD thesis. It combines classical molecular dynamics with the well known two-temperate model (TTM) to simulate the laser ablation of metals. In addition to the features of the original code it includes a wide-range model for the thermophysical, optical and transport properties of the electronic subsystem, a Helmholtz- and a Maxwell-solver for the laser-matter interaction, a heat-advection solver, non-reflecting pressure absorbing boundary conditions and many more.

The purpose of this manual is to describe the usage of the lasMD code. All the parameters of the original IMD package are also valid in this package. The documentation for the original package can be found at `http://imd.itap.physik.uni-stuttgart.de/`. The details regarding the theoretical backgrounds, the numerics and the implementation are described in my thesis.

Contents

1 Introduction

The lasMD code was developed in the course of my PhD thesis. The aim was to extend the existing hybrid approach, originally developed by Christian Markus Ulrich in 2007, as to include the effects related to the nascent plasma plume on the ablation. In a nutshell, the simulation domain of the molecular dynamics part is superimposed with a finite-difference (FD) grid where the electronic part of the problem is solved. This consits of the diffusion and advection of the heat of the electronic subsystem as well as the absorption of the laser energy itself. The two subsystems are then coupled by means of an additional damping force in the atomic equations of motion, thus providing a gradual exchange of energy (equilibration) between the subsystems. A schematic illustration of this hybrid approach is depicted in the figure below.

Based on an exemplary input file, providing the necessary parameters for a simulation, the following sections will describe the purpose of each parameter and point out some caveats.

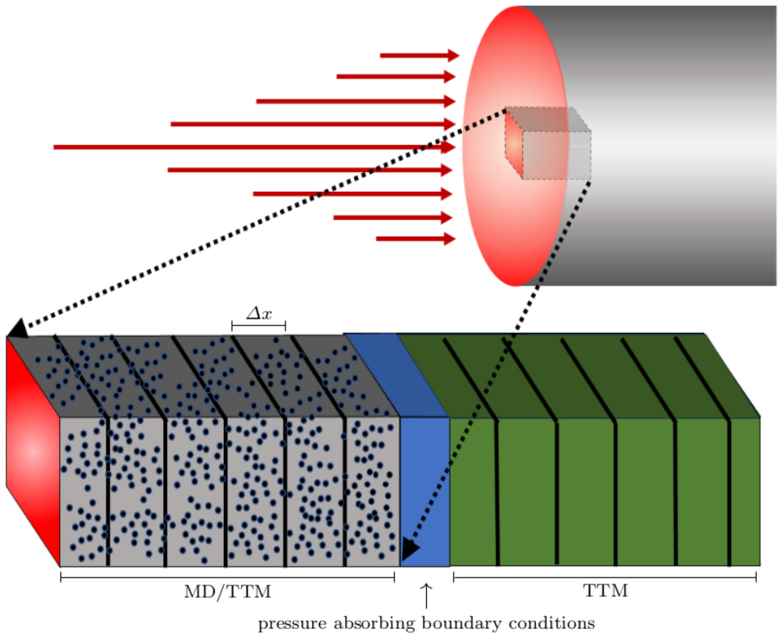

Figure 1: Schematic illustration of the hybrid MD/TTM approach. In a one-dimensional approximation the model simulates the laser-matter interaction for a small volume-element of the full sample located at the very center of the laser-beam. The MD/TTM-simulation domain is extended by a "'virtual lattice"', where the TTM is described on a continuum level only. Pressure absorbing boundary conditions between the hybrid model and the continuum model prevent laser induced shock-waves from being reflected at the rear side of the sample back toward the irradiated surface.

2 Building and running lasMD

The compilation and linking for the lasMD-package is straighforward. Inside the source directory simply issue a command like

```
make -j8 imd_mpi_eam_nve_nbl_nrb_stress_ttm_tmm_filter_lb
```

Don't forget to issue

```
make clean
```

before rebuilding the code. This command activates all the options necessary for a typical laser-ablation simulation. The exact order of the options `mpi`, `eam`, etc. doesn't matter. The binary file will be copied into `~/bin/`. Note that the file `imd_ttm.c` is a completely rewritten code compared to the original TTM-implementation of IMD. It is optimized for quasi-1D simulations, i.e. where the FD-lattice of the electronic subsystem is only allowed to be one-dimensional. If you want to perform 2D- or 3D-simulation you will need to replace this file with `imd_ttm_3D.c` and comment out the `-DTTM1D` flag in the Makefile in the line which says

```
PP_FLAGS += -DTTM -DTTM1D
```

In doing so the load-balancing module as well as the helmholtz-solver cannot be used anymore.

For the compilation you will need the MPI-library as well as an additionl third party library, called ”‘natural neighbors interpolation”’ which can be found in `https://github.com/sakov/nn-c`. Compile this library and place the static library file `libnn.a`, as well as the header files `delaunay.h` and `nn.h` into the subfolder `nn_interpol` before building lasMD.

If you wish to use the ”‘collisional-radiative”’-module, you will need to link against the `gsl` library, the `blas`-library, the `OpenMP`-library and the `cvode` library, included in the `sundials`-package of version 4.1.0. However, since this module is still in a verly early alpha-state, the corresponding directives are not included in the Makefile. Still you can make it manually, for instance by issuing something like

```
make clean
CFLAGS="-Wno-unused-variable -I/user/path/to/sundials-4.1.0/instdir/include \
  -I/usr/local/include/gsl"
LFLAGS="-L/user/path/to/sundials-4.1.0/instdir/lib64 -lsundials_cvode \
  -lsundials_nvecserial -lsundials_sunlinsollapackdense \
  -L/usr/local/lib -lgsl -L/usr/local/lib -lgslcblas"

OPTFLAGS="-funroll-loops -march=corei7-avx
         -mtune=corei7-avx -mavx2 -ftree-vectorize -m64 -ffast-math"

mpicc -O2 $CFLAGS -DMPI  -DNBL -DEAM2 -DTTM -DTTM1D -DTMM -DCOLRAD -DLOADBALANCE \
   -c -Wno-unused-variable -Wno-unused-result -fopenmp \
   imd_maxwell.c imd_misc.c imd_param.c imd_alloc.c \
   imd_io.c imd_io_3d.c imd_potential.c imd_time.c \
   imd_generate.c imd_distrib.c imd_main_3d.c \
   imd_geom_3d.c imd_pictures_3d.c \
   imd_geom_mpi_3d.c imd_comm_force_3d.c \
   imd_fix_cells_3d.c imd_mpiio.c imd_mpi_util.c\
   imd.c imd_ttm.c imd_interpol.c fminbnd3.c \
   imd_tmm.c imd_colrad.c imd_forces_nbl.c imd_integrate.c \
   imd_loadBalance.c imd_loadBalance_direct.c

mpicc -O2 -o imd_mpi_eam_ttm_tmm_nbl_colrad_loadbalance \
   imd_maxwell.o imd_integrate.o imd_misc.o imd_param.o \
   imd_alloc.o imd_io.o imd_io_3d.o imd_loadBalance.o \
   imd_loadBalance_direct.o imd_potential.o imd_time.o \
   imd_generate.o imd_distrib.o imd_main_3d.o imd_geom_3d.o \
   imd_pictures_3d.o imd_geom_mpi_3d.o imd_comm_force_3d.o \
   imd_fix_cells_3d.o imd_mpiio.o imd_mpi_util.o imd.o imd_ttm.o \
   imd_interpol.o fminbnd3.o imd_tmm.o imd_colrad.o \
imd_forces_nbl.o ./nn_interpol/libnn.a -lm  $LFLAGS -fopenmp
mv imd_mpi_eam_ttm_tmm_nbl_colrad_loadbalance ~/bin/
```

After the code is compiled and linked you can run lasMD by simply issuing e.g.

```
mpirun -n 8 imd_mpi_eam_nve_nbl_nrb_stress_ttm_tmm_filter_lb
    -p laser.inp
```

The `-p` option is mandatory and requires the path to the parameter file for the simulation run. The structure of this file is discussed in section 4. Of course, the `-n` option needs to be compliant with the number of available cores and the domain-decomposition which is also dictated by the parameter file.

3 Required files/tables

Before running the binary however, a few files are needed, which are required by the TTM for a wide-range descirption of the thermophysical and optical properties of the electronic subsystem. The paths and names of these files are currently hardcoded into lasMD and need to be placed in the parent directory of the directory, where the simulation is executed from. The following list describes the purpose and structure of these files

`EOS_cve_from_r_te.txt` : This file contains a table which is needed by the TTM to calculate the electronic specific heat (heat capicity) from the electron temperature T_e and the material density ρ. The table provided by this package in the directory `EOS` is calculated based on the Thomas-Fermi model for aluminum. The structure of this file looks as follows:

```
2000 550
30 4000 3.000000000e+01 2.931711663e+06
3.000000000e+01 3.000000000e+01 1.223414444e+01
3.000000000e+01 3.141385644e+01 1.280942976e+01
3.000000000e+01 3.289434588e+01 1.341163386e+01
...
```

The first line represents the number of density- and temperature-coordinates provided by this table. The second line contains the values for the minimum and maximum density (30 and 4000 kg/m^3), as well as the minimum and maximum temperature (30, 2.0e6 K). The following 2000×550 lines contain scattered data, giving the specific heat as a function of density and temperature in units of J/(K $\cdot$ kg). During the simulation, this table is linearly interpolated by means of the natural-neighbor library mentioned in the previous section.

`EOS_ee_from_r_tesqrt.txt` : This table gives the specific internal energy of the electrons e_e as a function of density and the square-root of the electronic temperature in units of J/kg. The structure is exactly the same as for the heat capacity. The decision to use the square root of the temperature rather than the temperature itself is motivated by the relation $e_e \underset{\sim}{\propto} \sqrt{T_e}$. This way, the errors introduced by the linear interpolation can be minimized.

`EOS_phase_from_r_ti.txt` : This table gives the phase state of the material as a function of density and lattice/ion temperature T_i. The structure of this file is the same as for the previous tables. This is used by a subroutine which calculates the wide-range permittivity of the material. The integer numbers 3,4 and 5 represent thermodynamically stable liquid or gaseous states while -3,-4 and -5 represent metastable liquid or gaseous states. Any other number is interpreted as a solid state. The interpolated value is rounded to the nearest integer and if the modulus of the resuting number equals 3,4 or 5 the material is considered as either liquid or gaseous. In this case the interband-contribution to the permittivity is omitted.

`EOS_pe_from_r_te.txt` : This table gives the electronic pressure as a function of density and electron temperature in units of Pa. The structure of this file is the same as for the previous tables. This file will only be loaded in case the user activates a hardcoded parameter `ELECPRESS` in the file `imd_ttm.c`. This feature is discussed in section 5.

`alu_eps_bb.dat` : This table gives the real and imaginary parts of the interband-contribution to the relative permittivity of aluminum as a function of laser-wavelength. Further details can be found in my thesis. This table is headerless.

`K12.dat` : This table represents a term consisting of two precalculated integrals. It is used to calculate the relative permittivity in the limit of a hot plasma-state. Further details can be found in my thesis. This table is headerless.

4 Full parameter file

The following file, I'm just calling *laser.inp* demonstrates the paramters needed for a typical simulation run. All the text behind the #-symbol is interpreted as a comment by IMD.

```
##############
# SETUP
##############
ntypes 1
core_potential_file ../phi.zhakov.pt
embedding_energy_file ../f.zhakov.pt
atomic_e-density_file ../rho.zhakov.pt
coordname          ../shift.chkpt
cpu_dim            60 2 2 #=240
outfiles LASER/laser
maxsteps 500000
simulation 1
checkpt_int 5000
eng_int   1000
ensemble ttm
timestep 0.1
pbc_dirs 0 1 1
box_from_header 1

##################
#          LB
##################
lb_frequency        1000
lb_writeStatus      1
lb_balancingType    2
lb_preRuns          1

##############
# TTM
##############
fd_ext              2 4 4
ttm_int             100
atomic_weight       26.981538
atomic_charge       13
fd_n_timesteps      250
fd_min_atoms        100
ttmdimx             960         # 960 / 240 = 4
vlatdim             60
vlatbuffer          35
```

```
40
41 ###########
42 #DISTR
43 ############
44 #X        1.55e4  0.00e0  0.00e0
45 #Y        0.00e0  1.37e2  0.00e0
46 #Z        0.00e0  0.00e0  1.37e0
47 ##PBC 0 1 1
48
49
50  dist_int           100
51  dist_dim           1000 1   1
52  dist_ll            0 0 0
53  dist_ur            1.55e4  1.37e2  1.377e2
54
55  dist_dens_flag     3
56  dist_press_flag    3
57  dist_mdtemp_flag   1
58
59
60 ###############
61 # HELMHOLTZ
62 ###############
63  I0                 2.70e17
64  lambda             800e-9
65  laser_t_0          3.0394e-13
66  laser_sigma_t      7.22e-14
67  laser_t_1          1e9 #some huge number
68  laser_sigma_t1     42.5e-15
69  tmm_threshold      20.0
70 ############
71 # NBL
72 ############
73  nbl_size           1.2
74
75 ###########
76 # FILTER
77 ##########
78  filter_min_x       100
79  filter_int         2500
80
81 #######
82 # NRB #
83 #######
84  nrb_alat           4.05
85  nrb_eps            0.2
86  nrb_k              0.9
87  nrb_infile         ../shift.nrb
88 #nrb_overwrite      1
```

The following descirption presents a breakdown of the individual parameter blocks of the listing above.

lines 4-18: SETUP : This block contains the mandatory instructions every simulation needs, irrespective of whether the laser ablation module is to be used or not.

1. `ntypes`: an integer representing the number of atom types, the configuration file consists of. Regarding the two-temperature model, only atoms of type 0 are included into the model. Atoms belongig to other types will be regarded as vacuum by the TTM, while still being present in the MD-part of the model. This can be used to include e.g. an ambient gas or a transparent overlayer material.

2. `core_potential_file, embedding_energy_file, atomic_e-density_file`: Depending on the type of inter-

atomic interaction model, IMD requires the relative paths for the tables to be interpolated in order to calculate the interatomic forces. For more details the reader is referred to the original IMD documentation.

3. `coordname`: Requires the relative path to the atomic configuration file in IMD-format. Again, more details can be found in the original IMD documentation. Note: Currently, only rectangular simulation boxes are supported. Make sure that the simulation box vectors originate at $(0, 0, 0)^T$. Further it is suggested that the whole sample is shifted along the $+x$-direction by a value somehting between 500 nm and 1000 nm, depending on the problem you want to simulate. In doing so, sufficient room is left for the ablation plume to expand into vacuum, because the laser only irradiates from the $-x$-direction.

4. `cpu_dim`: Requires a 3D integer-vector, giving the domain decomposition for the MPI-tasks.

5. `outfiles`: Path for the output files. In this example all files will be written to the directory LASER. They will have the prefix *laser*, followed by a suffix, indicating the file type. For example **.ttm* and **.chkpt* for outputs related to the TTM (see next section) and the atomic configuration files, respectively.

6. `maxsteps`: Integer number giving the total MD-integration steps to be performed.

7. `simulation`: Integer number specifying the number of the current simulation. IMD allows you to encapsulate several simulations into a single parameter file.

8. `checkpt_int`: Integer number giving the interval for the output atomic configuration files in units of MD-integration steps.

9. `eng_int`: Integer number giving the frequency of output writes into a file with the extension *.eng. Depending on the ensemble used, different outputs can be expected. In any case IMD writes every `eng_int` steps a single line containing the total potential and kinetic energy in units of eV per atom as well as the total pressure in units of $\mathrm{eV}/\mathrm{Å}^3$ into this file.

10. `ensemble`: Expects a string defining the MD-ensemble to be used. All possible options are descibed in the original IMD documentation.

11. `timestep`: A floating point number defining the size of the integration step in units of 10.18 fs.

12. `pbc_dirs`: An integer 3D-vector telling IMD whether or not to apply periodic boundary conditions along the corresponding catersian axis of the simulation box.

13. `box_from_header`: This option tells IMD wheter or not to read the simulation box parameters from the header of the provided `coordname`-file. 1 represent "‘yes"’, while 0 means "‘no"’.

lines 23-26: LB : This block contains paramters regarding the load-balancing of the MD-part of the model.

1. `lb_frequency`: Integer number defining the steps between recalculating and readjusting the computational load on every

MPI-task.

2. `lb_writeStatus`: If set to 1, IMD writes a log-file with additional information regarding LB.
3. `lb_balancingType`: An integer number specifying the LB-scheme. Unfortunately this information is not documented in the orignal IMD manual. The interested reader has to examine the code itself.
4. `lb_preRuns`: An integer number specifiying the number of short test-simulation in order to optimize the LB-scheme before the actual production run.

lines 31-39: TTM : In this block the user provides the parameters concerning the TTM.

1. `fd_ext`: An integer 3D-vector specifying the number of MD cells along the three cartesion directions to be combined into a single node of the finite-difference grid for the TTM. The number of MD cells is automatically determined by IMD using the simulation box and the cutoff radius derived from the potential files. The size of an MD cell is written to STDOUT at the very beginning of every simulation, irrespective of the ensemble to be used.
2. `ttm_int`: An integer number specifying how often the *.ttm-files have to be written.
3. `atomic_charge`: An integer number specifying the atomic number (nucleus charge) of the material to be simulated.
4. `atomic_weight`: A floating point number representing the atomic weight in a.u. of the material to be simulated.
5. `fd_n_timesteps`: An integer number which defines the minimum number of substeps of the electronic part (TTM) of the model. This is used for the integration of the heat-diffusion equation. The actual integration step of the TTM may be higher, as it is dynamically computed from the Courant-Friedrich-Lewy criterion.
6. `fd_min_atoms`: This integer number determines a lower threshold value for the number inside a single FD-cell to be activated. FD-cells containting less atoms will be interpreted as vacuum by the TTM because the lattice temperature derived from a minor amount of atoms may result in statistically insignificant average temperature values.
7. `ttmdimx`: This option is only allowed for 1D-TTM simulations, i.e. where the FD-grid is one-dimensional. This parameter defines the number of FD-cells along the x-direction of the simulation box and thus overwrites whatever is computed from the parameter `fd_ext`. Note, hower, that this number must currently be divisible by the total number of MPI-tasks.
8. `vlatdim`: This option is only allowed for 1D-TTM simulations. The corresponding integer number specifies the number of "‘virtul"’ FD-cells which are appended to the end of the MD-simulation box. In this virtual lattice, there is no interaction of the TTM with the actual atoms. Both, the electrons as well as the lattice are described solely on a continuum level of the TTM.

9. `vlatbuffer`: This integer number can be used to shift the cell of the virtual lattice to the left, i.e. into the "‘real"’ TTM-lattice. This can be used to skip a region of the atomic lattice, which is affected by the cooling effect of the non-reflecting pressure absorbing boundary conditions.

lines 50-57: DISTR : This block determines how the MD-part of the simulation-domain is sampled during the simulation. The resulting distributions are written to files and provide tables containing the profiles for the temperature, density and pressure. By combining these files, the temperoal evolution of these profiles can be visualized using surface- or contour-plots.

1. `dist_int`: This integer number specifies the number of steps between the writouts of the distribution files.
2. `dist_dim`: An integer 3D-vector determining the number of slices, the cartesion simulation-box is subdivided into, along each dimension. In this example the simulation-box is divided into 1000 slices along the x-direction and a single slice along the y- and z-direction. Within each of these slices an average quantitiy can be computed from the number of atoms within this slice as well as their momenta.
3. `dist_ll`: A floating point 3D-vector representing the coordinate of the lower left corner of the cuboid sampling domain.
4. `dist_ur`: A floating point 3D-vector representing the coordinate of the upper right corner of the cuboid sampling domain.
5. `dist_dens_flag, dist_press_flag` : An integer flag specifying the output format of the distribution files. The density is written out in units of atoms per Å^3, the hydrostatic "‘pressure"’ in eV. Thus, the latter one needs to be divided by the "‘volume"’ of a single atom. To a good approximation, this can be achieved by multiplying this value by the atomic density of this slice. More information is provided by the original IMD documentation.
6. `dist_mdtemp_flag`: This parameter only accepts the number 1. The distribution file is written in ASCII-format.

lines 63-69: HELMHOLTZ : This block determines the parameters for the description of the laser-matter interaction. The Helmholtz-solver is based on the Transfermatrix-method. You can also choose to simulate the laser-matter interaction by means of the Beer-Lambert law. For this purpose you need to replace the `_tmm_` option by the `_laser_` option when building the code. Details regarding the parameters needed for the Beer-Lambert-description can be found in the original IMD documentation.

1. I_0: This floating point number determines the peak intensity of the Gaussian laser pulse in units of W/m^2. For a Gaussian pulse, the relation between the peak intensity and the incident laser fluence (energy per area) $F_{\text{inc.}}$ is given as

$$I_0 = \frac{F_{\text{inc.}}}{\tau_{\text{FWHM}}\sqrt{\pi/\log(16)}}\,, \tag{1}$$

where τ_{FWHM} is the pulse duration of full widht at half maximum. See below.

2. `lambda`: This number specifys the wavelength of the laser in units of meters.

3. `laser_t0`: The time of the laser peak intensity of the first laser pulse in units of seconds.

4. `laser_sigma_t`: This scalar represents the pulse duration in terms of the standard deviation σ_t of the first Gaussian pulse in units of seconds. Note that the relation between the full width at half maximum τ_{FWHM} and σ_t is simply:

$$\tau_{\mathrm{FWHM}} = 2\sqrt{2\log 2}\sigma_t \approx 2.355\sigma_t \tag{2}$$

5. `laser_t1,laser_sigma_t1`: Same as above, but for an optional 2nd pulse.

6. `tmm_threshold`: This optinoal floating point number defines a threshold value for the Transfermatrix method. This method becomes exponentially unstable beyond a point, where the magnitude of the incident electric field decays below a value $E(x)/E(x_0) < \exp(-\mathrm{tmm\_threshold})$. Depending on the permittivity of the material this value may vary. By default it is initialized to 20.0. Note that for very short samples, this value may need to be lowered.

line 73: NBL : In this section an optinal parameter `nbl_size` can be provided. It dermines the size of the neighbor-list which can significantly speed up the MD-part of the simulation. More details are provided in the original IMD-manual.

lines 78-79: FILTER : This block provides the parameters for the filter-module of lasMD. This is used to remove atoms from the simulation domain as soon as they cross a specific x-coordinate. This can considerably increase the performance of the simulation. Note that this feature only deletes an atom if it's not within the cutoff-distance of another atom, which itself is within the cutoff-distance of another atom,...etc., which is finally not flagged as "‘filter-atom"’ (it hasn't crossed the threshold x-coordinate. This way the deletion of "‘filter"’-atoms does not influence the trajectories of other atoms, whose x-coordinate has not yet crossed the user-defined threshold coordinate. Simple deletion of filter-atoms without considering their neighbors may result in an accumulation of material at the boundary of the simulation domain, since the attractive forces of the deleted atoms is missing.

1. `filter_min_x`: All atoms with a coordinate $x <$ `filter_min_x` are tagged as filter-atoms. If the algorithm approves it, these atoms are deleted in the next integration step.

2. `filter_min_int`: Since checking all the neighboring atoms of a tagged atom, as well as their neighbors and so forth is an computationally expensive task, this parameter determines the number of steps between the checks.

lines 84-88: NRB : This block determines the parameters of the non-reflective, pressure absorbing boundary conditions. Using this module, a large fraction of the laser-inducued shockwave can be absorbed at the rear side of the sample. Note: Currently, only fcc-lattices with the crystallographic $\{1, 0, 0\}$-axis oriented along the x-direction are supported.

1. nrb_alat: This floating scalar represents the lattice constant of the material in units of Angstrom.
2. nrb_eps: This parameter is only needed once to compute the topology of the boundary layer at the rear side of the sample. It represents a tolerance in units of Angstrom which is used by the algorithm to determine the nearest neighbors of the boundary layer atoms. The neighbors are detected by checking all atoms surrounding the boundary atoms for their distance along specific directions from the boundadry atoms.
3. nrb_k: This floating scalar specifies the spring-constant for the spring-sphere model, the NRB-module is based on. It is given in units of eV/$\mathring{A}^2$.
4. nrb_infile: This optional parameter can be used to continue a simulation, as this module writes out a topology-file havng the extension *.nrb into the output-directory every checkpt_int steps. If this parameter is specified, the costly detection of the topology of the boundary atoms is skipped and read from this file. It is strongly recommended to use it for the start of a production run as well. For this purpose a short pre-simulation (maybe only using the NVE-ensemble) needs to be carried out (without specifying nrb_infile). Ideally this pre-simulation uses the non-equilibrated sample having a perfect lattice structure in order to facilitate a correct detection of the nearest neighbors of the boundary layer atoms. The boundary layer atoms are automatically detected as those atoms having the most positive x-coordinate (in the 1D-case). Make sure to choose a small value for checkpt_int and maxsteps for this auxiliary simulation. A value of 1 is sufficient. The resulting *.nrb-file can then be used as an input-file for the production run of the equilibrated sample.
5. nrb_overwrite: This parameter is only needed if you use the *.nrb-file from a previous pre-simulation using the non-equilibrated perfect lattice as described before. Setting it to 1 will overwrite the equilibrium positions of the boundary atoms with their current positions. If you forget to overwrite these positions, the boundary atoms will very likely be pulled heavily towards the equilibrium positions of the perfect lattice (where probably no shift was applied). This will result in an immediate crash of the simulation.

5 Hardcoded parameters

Apart from the parameters discussed so far, there are also some useful parameters which have not been incorporated into the IMD-parser due to lack of time. In particular the parameters related to the thermophysical and transport-properties are optimized for aluminum. The following list summarizes the most important parameters.

RHOMIN : Just like fd_min_atom defines a threshold value for the number of atoms within a FD-cell, below which a cell is deactived, this parameter defines a threshold value for the corresponding density in units of kg/m^3. This needs to be larger or at least equal to the minimum density provided in the tables desribed in section 3. The parameter is defined in the files imd_ttm.c and imd_tmm.c.

ELEC_PRESS : If this compiler-flag is defined, the gradient of the electronic pressure (blast force) is accounted for in the equations of motion. The parameter is defined in the files `imd_ttm.c` and in `imd_integrate.c`.

BALLISTIC : Uncommenting this compiler flag in `imd_ttm.c` activates the effects of finite thermalization time as well as ballistic transport for the electronic subsystem. The correspinding relaxation time τ_b in units of seconds and the Fermi-velocity v_F in units of m/s are also hardcoded and can be found in the subroutine `do_BALLISTIC`.

```
    double tau_b=tau*10.18*1e-15;
    double vF=1.5e6;
```

TMM_t0_suggest : Uncommenting the definition of this compiler flag in `imd_tmm.c` results in an automatic adjustment of the time t_0 for the peak intensity I_0 of the Gaussian pulse in such a way, that the intensity at $I(t = t_0 - \sigma_t) = 10^{-5} I_0$ and $I(t = t_0 + \sigma_t) = 10^{-5} I_0$. Since the Helmholtz-solver is switched off for laser intensities below this thershold value, using this flag guarantees that the major part of energy, provided by the laser, is included in the simulation.

fd_k : This variable corresponds to the thermal conductivity of the electronic system. It is a function of the electronic temperature T_e, the ionic temperature T_i, the density ρ and the average ion charge $\langle Z \rangle$ The computation of this quantitiy is based on a wide-range interpolation model, the parameters of which are hardcoded in the subroutine `getKappa` in the file `imd_ttm.c`. The details of this interpolation model are described in my thesis.

fd_g : This quantity represents the electron-ion energy-coupling parameter of the TTM. Just like the thermal conductivity it is a function of T_e, T_i, ρ and $\langle Z \rangle$ computed from an interpolation model. The parameters are defined in the subroutine `getGamma` in the file `imd_ttm.c`.

tmm_eps_real_arr_global : This array contains the real part of the relative dielectric function for each non-empty cell within the FD-grid. It is computed inside the subroutine `tmm_get_epsilon` in the file `imd_tmm.c` from a wide-range interpolation model as a function of T_e, T_i, $\rho, \langle Z \rangle$ and the laser-wavelength λ. Just as in the case of the two previous quantities the corresponding interpolation-parameters are hardcoded inside this subroutine.

tmm_eps_imag_arr_global : Same as above, but for the imaginary part of the relative dielectric function.

6 The *.ttm-output file

The structure of the *.ttm-output file is very similar to the distribution files described in `http://imd.itap.physik.uni-stuttgart.de/userguide/output.html#distributions`. Basically, the header of the file tells you everything you need to know:

```
#x y z natoms temp md_temp U xi source dens vx vy vz fd_k fd_g Z proc Ce
```

The columns `x,y` and `z` represent the integer coordinates of a node of the FD-grid. The columns denoted by `temp` and `md_temp` are the electronic and the ionic temperatures of this node/cell, respectively. This is followed by `U`, giving the internal specific energy of the electrons, the damping term `xi` due to electron-ion coupling, which enters the MD equations of motion and the laser source term `source` representing the absorbed power density.

The next three columns contain the averaged components of the atomic velocities `vx,vy` and `vz`. The columns `fd_k`, `fd_g` and `Z` are the electronic thermal conductivity, the electron-ion energy coupling parameter and the mean charge, respectively. Finally, the column `proc` tells, you which MPI-task is responsible for the corresponding TTM-node and `Ce` is the electronic specific heat (heat capacity). All quantities are given in IMD-units rather tham SI-units. The relation between IMD-units and the corresponding value in term of SI-units is given in the table below.

Quantity	**IMD-unit**	**SI-value**
Time	$\text{Å}\sqrt{\text{u/eV}}$	$10.18 \cdot 10^{-15}$ s
Velocity	$\sqrt{\text{eV/u}}$	9822.59 m/s
Temperature	eV	11604.5 K
Specific heat	$\text{eV}/(\text{K}^2\text{Å}^2)$	$1.381 \cdot 10^7$ J/(m^{3}K)
Heat conductivity	$\text{eV}^{3/2}/(\sqrt{u}K\text{Å})$	13.561 J/(sKm)
Specific heat	eV/u	$9.649 \cdot 10^7$ J/kg
Electron-ion coupling	$\text{eV}^{3/2}/(\sqrt{u}K\text{Å}^4)$	$1.356 \cdot 10^{21}$ J/(m^{3}Ks)
Power density	$\text{eV}^{3/2}/(\sqrt{(\text{u})}\text{Å}^4)$	$1.574 \cdot 10^{25}$ W/m^3

7 Large-scale 2D simulations (pre-alpha)

The one-dimensional approach proves to be a reliable model when one is interested in quantities such as the ablation depth, the melting depth, the intensity of laser induced-shockwaves as a result of a short single pulse. On the other hand, if the main focus is the expansion of the ablation plume, the interaction of a second, third, etc. pulse with the plume or maybe the shape of the ablation crater, a fully three-dimensional model is needed. However, the Rayleigh-limit for the minimum laser spot diameter $d \approx 1,22 \cdot \lambda \cdot \text{NA}$, with a typical numerical aperture $\text{NA} \approx 0,9$ poses a serious challange to a molecular dynamics approach. This requires spot sizes for visible light on the order of microns. Not only must the sample's dimensions accommodate for the spot size itself, but also for the even larger heat-affected zone, such that a natural heat flow is allowed. As a result, the size of the sample needs to be on the order of several microns at least, giving rise to tens of billions of atoms. Clearly, this three-dimensional approach is not an option nowadays due to the huge computational demand.

Somewhat more practical is a quasi-2D model, where only two of the sample's dimensions extend to the micron scale, while the third can be considerably smaller. In this situation the MD-simulation box is still 3D, but with only a 2D FD-grid.

`lasMD` provides a means for this kind of simulations. In this case, the laser-matter interaction is accounted for by a 2D Maxwell-solver. The curl equations are integrated by means of the Finite Difference Time Domain method (FDTD) for a single-pole Drude-Lorentz medium. The relative permittivity for such a medium is given as

$$\epsilon_r(\omega) = \epsilon_\infty - \frac{\omega_p^2}{\omega^2 + i\Gamma\omega} + \frac{\tilde{\omega}_p^2}{\omega_0^2 - \omega^2 - i\gamma\omega}\,, \tag{3}$$

In lasMD the user can decide wheter all the Drude-Lorentz parameters should be constants or if they should be interpolated from tables as functions of T_e, T_i and ρ.

7.1 Additional tables

In the latter case the user needs to provide 5 additional tables in the parent directory of the directory, from where the simulation is executed: DL1.txt for the dimensionless parameter ϵ_∞, DL2.txt for Γ, DL3.txt for $\tilde{\omega}_p$, DL4.txt for ω_0 and DL5.txt for γ, while ω_p^2 is calculated internally.

The structure of the tables is as follows:

```
80 140 100
100.000000 3500.000000 -1.587505 1.236404 -1.587505 0.935374
100.000000 -1.587505 -1.587505 12.453404
100.000000 -1.587505 -1.562022 12.453384
100.000000 -1.587505 -1.536538 12.453362
100.000000 -1.587505 -1.511054 12.453339
```

The first two lines represent the header. The first line defines the number of ρ-,$\log(T_e)$-,and $\log(T_i)$ coordinates in this table. Note that the temperature coordinates are given in terms of their logarithm to the base 10. The second line defines the boundaries of the corresponding coordinates, i.e. their minimum and maximum values. This is followed, in this example by $80 \times 140 \times 100$ entries of the independent variables in the columns 1,2,3. The dependent variable (excluding the dimensionless ϵ_∞) is given in the third column in units of eV, as a result of multiplying it by $\hbar$ [eV $\cdot$ s]. This transforms a frequency into an energy. This tables are then interpolated during the simulation using a tricubic interpolation scheme.

On the other hand, if the Drude-Lorentz parameters should be constants, simply uncomment the following lines in the file imd_ttm3D.c and the subroutine fitDL:

```
  /*
    node.DL[0]=2.73;
    node.DL[1]=1.1174e+15;
    node.DL[2]=7.6595e+15;
    node.DL[3]=2.4024e+15;
    node.DL[4]=4.5199e+14;
    node.DL[5]=2.2955e+16;
    return 0;
  */
```

At the same time comment out the lines, responsible for reading the tables in the same file

```
  read_tricub_interp(&Lop1i, "../DL1.txt");
  read_tricub_interp(&Lop2i, "../DL2.txt");
  read_tricub_interp(&Lop3i, "../DL3.txt");
  read_tricub_interp(&Lop4i, "../DL4.txt");
  read_tricub_interp(&Lop5i, "../DL5.txt");
```

These values corresponds to the Drude-Lorentz model of solid aluminum at room temperature.

7.2 Building

In order to build the quasi-2D code, first the Makefile needs to be adjusted. For this purpose, comment the -DTTM1D flag in the line

```
 PP_FLAGS += -DTTM -DTTM1D
```

Then, the `imd_ttm_3D.c` source file needs to be renamed into `imd_ttm.c`. Caution: Make a backup of the original `imd_ttm.c` as this is the optimzed code for the 1D-simulations. Alternatively, you could simply modify the `Makefile` to take care of an easy switching between the two files (as it is supposed to be done, but lack of time...). Another caveat to consider is, that `imd_ttm_3d.c` is not supported by the load-balancing, so keep that in mind.

Finally, the code can be built by something like

```
make -j8 imd_mpi_eam_nve_nbl_nrb_stress_ttm_fdtd_filter_mpiio
```

Note that there are two new compiler flags `fdtd` and `mpiio`, which will be discussed in the next section.

7.3 Parameter file

A valid parameter input file for the 2D-simulation might look something like in the listing below.

```
##############
# SETUP
##############
ntypes 1
core_potential_file ../phi.ercolessi.al.pt
embedding_energy_file ../f.ercolessi.al.pt
atomic_e-density_file ../rho.ercolessi.al.pt

parallel_input  2
parallel_output 2

cpu_dim         80 120 1

simulation 1
ensemble ttm

outfiles LASER/laser
maxsteps 500000

eng_int   1000
ttm_int   100
checkpt_int 5000

timestep 0.1
coordname ../config.mpiio
box_from_header 1

pbc_dirs 0 0 1
##############
# TTM
############
fd_ext          4 4 8
atomic_weight   26.9815
atomic_charge   13
fd_n_timesteps  300 #lower limit
fd_min_atoms    100
#############
# FDTD
###############
pml     20      #thickness of pml
I0      1.0e16
lambda  800e-9
srcw    500e-9  # waist radius
srcx    500e-10  # x-pos. of soft-source in m
laser_t_0       200e-15
laser_sigma_t   42.5e-15   #in SIi = 100fs tfwhm
laser_t_1       500;       #-> never
laser_sigma_t1  42.5e-15
```

```
49 Sc 0.7          #corant number
50
51 ########
52 # NBL
53 #########
54 nbl_size 1.2
55
56 ###########
57 # FILTER
58 ##########
59 filter_min_x    500
60 filter_min_y    100
61 filter_max_y    35900
62 filter_int      5000 # use it rarely
63
64 #######
65 # NRB #
66 #######
67 nrb_alat         4.05   #300 K
68 nrb_eps          0.2
69 nrb_k            0.9
70 nrb_infile       init.nrb
71 nrb_overwrite    1
72
73 ############
74 #  SHIFT
75 #############
76 shiftx_front 10000
77 shiftx_rear  0
78 shifty_front 495
79 shifty_rear  18.82
80
81 ############
82 #DISTR
83 ############
84 dist_int         100
85 dist_dim         1000 2000  1
86 dist_ll          0 0 0
87 dist_ur          18427.0362 36004.6070 48.6171
88
89 dist_dens_flag  2
90 dist_press_flag 2
```

The following description focuses on the parameters, that have not been discussed yet in section 4.

lines 4-28: SETUP :

1. `parallel_input`: Specifying this parameter as 2 will invoke the mpiio-module of `lasMD`. This should be preferred in large scale simulations, because parallel reading is significantly faster and POSIX-reading/writing of huge files may even be forbidden on your cluster of choice. Further, the input configuration file, specified by `coordname` should be in binary format. The tool `bin_to_chkpt.c` in the source folder of `lasMD` can be used to convert a binary IMD-file into a *.chkpt-file in serial.
2. `parallel_output`: Just as reading in parallel using MPIIO, the output configuration files can be written in parallel as well. ASCII-output in parallel is not supported.
3. `cpu_dim`: Make sure to apply periodic boundaries only along the z-direction.

lines 40-49: FDTD :

1. `pml`: This parameter specifies the width of the perfectly matched layers (light absorbing boundaries) in units of FD-lattices nodes.
2. `srcw`: This scalar determines the waist radius of the spatially

Gaussian laser beam in units of meters. This corresponds to the distance from the coordinate of maximum electric field excitation, where the intesity has decayed by a factor of $1/e$ or the electric field by a factor of $1/e^2$.

3. `srcx`: Here, the x-coordinate of the soft source is defined. Make sure, that this coordinate is neither inside the sample nor the pml-region. Note, that `lasMD` excites both possible modes, the TEZ- as well as the TMZ-mode by the same amount. This means, each mode receives half of the total laser energy.

4. `Sc`: This optional parameter represents the Courant number and thus the time-step for the FDTD-method. If you don't specify it, it will be automatically computed from the CFL-criterion.

lines 59-62: FILTER In the 2D-case, in addition to `filter_min_x`, the two threshold coordinates along the y-direction `filter_min_y` and `filter_max_y` need to be given in units of Angstrom.

lines 76-79: SHIFT If you're working with huge samples it can be quite tedious to shift the sample in order to create a vacuum buffer in front of it by means of some script (e.g. awk or python). For this purpose, `lasMD` provides the optional parameters `shiftx_front`,`shifty_front` as well as `shiftx_rear` and `shifty_rear`. The latter two simply extend to x-and y-vectors of the simulation box by the scalars given. The former two, additionally shift the atomic coordinates by the defined values in units of Angstrom. As shifting is performed in parallel, it is way faster than a custom script could ever achieve. Don't forget to adjust the `dist_ur` parameters to account for the enlargement of the simulation box.

lines 84-90: DISTR Note that for 2D-simulatino it is suggested to specify the output format of the distributions files as 2. Compared to format 1, this will also output the coordinates of the slices/cells, which simplifies post processing and visualization.

8 Collisional Radiative Model (WIP)