LCC

1.0.0.

Chapter 1

about

title: Building LCC documentation

The folder (`src/docs`{.sourceCode}:) contains all the documentation relevant to both users and developpers.

Prerequisites

- [pdflatex] Latex GNU compiler. pdfTeX is an extension of TeX which can produce PDF directly from TeX source, as well as original DVI files. pdfTeX incorporates the e-TeX extensions.
- [doxygen] Doxygen is a documentation system for C++, C, Java, Objective-C, IDL (Corba and Microsoft flavors) and to some extent PHP, C#, and D.
- [sphinx] Sphinx is a documentation generator or a tool that translates a set of plain text source files into various output formats, automatically producing cross-references, indices, etc. That is, if you have a directory containing a bunch of reStructuredText or Markdown documents, Sphinx can generate a series of HTML files, a PDF file (via LaTeX), man pages and much more.
- Any pdf viewer.
- Any web browser.

These programs can be installed as follows:

```
sudo apt-get install pdflatex
sudo apt-get install doxygen
sudo apt-get install dot2tex
sudo apt-get install python3-sphinx
pip3 install PSphinxTheme
pip3 install recommonmark
```

Build the full documentation

This will build all three types of docs (Sphinx, Doxygen, and latex):

```
make
```

The documentation that is build with Sphinx can be tested as follows:

```
firefox lcc.html
```

The file can be explored using any web browser.

One can also build any of the documentations separatly. For example, to build the Sphinx documentation, we can do:

```
make sphinx
```

Documenting

In order to add a documentation using Sphinx follow these steps:

: 1) make a file with a proper name under `./sphinx-src/source/`{.sourceCode}. For example: `MYPA`↩ `GE.md`{.sourceCode}. 2) Add the documentation inside the file using "markdown" syntax. 3) Modify the file in `./sphinx-src/source/index.txt`{.sourceCode} to include the documentation.

After modyfing this file, recompile Sphinx by typing `make sphinx`{.sourceCode}.

Chapter 2

Todo List

Subprogram [lcc_build_mod::lcc_bravais_growth](#) (nCycles, dTol, dTo, tCoordination, seed_file, r_inout)

Optimize the routine.

Subprogram [lcc_lattice_mod::lcc_triclinic](#) (Nx1, Nx2, Ny1, Ny2, Nz1, Nz2, lattice_vectors, supra_lattice_↩ vectors, r_sy, verbose)

A angles_to_vectors transformation will be available.

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 lcc_allocation_mod Module Reference

Module for allocation operations.

Functions/Subroutines

- subroutine, public lcc_reallocate_realvect (vect, ndim)

  To reallocate a real vector.
- subroutine, public lcc_reallocate_realmat (mat, mdim, ndim)

  To reallocate a real mxn matrix.
- subroutine, public lcc_reallocate_intvect (vect, ndim)

  To reallocate a real vector.
- subroutine, public lcc_reallocate_intmat (mat, mdim, ndim)

  To reallocate an integer mxn matrix.
- subroutine, public lcc_reallocate_char2vect (vect, ndim)

  To reallocate a character vector.
- subroutine, public lcc_reallocate_char3vect (vect, ndim)

  To reallocate a character vector.

5.1.1 Detailed Description

Module for allocation operations.

5.1.2 Function/Subroutine Documentation

5.1.2.1 lcc_reallocate_char2vect()

```
subroutine, public lcc_allocation_mod::lcc_reallocate_char2vect (
            character(2), dimension(:), intent(inout), allocatable vect,
            integer, intent(in) ndim )
```

To reallocate a character vector.

This will reallocate a character len=2 vector If it is already allocated, a deallocation will first happen.

Parameters

vect	Character(2) 1D array.
ndim	Dimension to reallocate the vector to.

5.1.2.2 lcc_reallocate_char3vect()

```
subroutine, public lcc_allocation_mod::lcc_reallocate_char3vect (
            character(3), dimension(:), intent(inout), allocatable vect,
            integer, intent(in) ndim )
```

To reallocate a character vector.

This will reallocate a character len=3 vector. If it is already allocated, a deallocation will first happen.

Parameters

vect	Character(3) 1D array.
ndim	Dimension to reallocate the vector to.

5.1.2.3 lcc_reallocate_intmat()

```
subroutine, public lcc_allocation_mod::lcc_reallocate_intmat (
            integer, dimension(:,:), intent(inout), allocatable mat,
            integer, intent(in) mdim,
            integer, intent(in) ndim )
```

To reallocate an integer mxn matrix.

This will reallocate a matrix. If it is already allocated, a deallocation will first happen.

Parameters

mat	Integer 2D array.
mnim	First dimension to realocate the matrix to.
ndim	Second dimension to realocate the matrix to.

5.1.2.4 lcc_reallocate_intvect()

```
subroutine, public lcc_allocation_mod::lcc_reallocate_intvect (
            integer, dimension(:), intent(inout), allocatable vect,
            integer, intent(in) ndim )
```

To reallocate a real vector.

This will reallocate a vector If it is already allocated, a deallocation will first happen.

Parameters

vect	Integer 1D array.
ndim	Dimension to reallocate the vector to.

5.1.2.5 lcc_reallocate_realmat()

```
subroutine, public lcc_allocation_mod::lcc_reallocate_realmat (
            real(dp), dimension(:,:), intent(inout), allocatable mat,
            integer, intent(in) mdim,
            integer, intent(in) ndim )
```

To reallocate a real mxn matrix.

This will reallocate a matrix If it is already allocated, a deallocation will first happen.

Parameters

mat	Real 2D array.
mnim	First dimension to realocate the matrix to.
ndim	Second dimension to reallocate the matrix to.

5.1.2.6 lcc_reallocate_realvect()

```
subroutine, public lcc_allocation_mod::lcc_reallocate_realvect (
            real(dp), dimension(:), intent(inout), allocatable vect,
            integer, intent(in) ndim )
```

To reallocate a real vector.

This will reallocate a vector If it is already allocated, a deallocation will first happen.

Parameters

vect	Real 1D array.
ndim	Dimension to reallocate the vector to.

5.2 lcc_aux_mod Module Reference

Module for auxiliary operations routines.

Functions/Subroutines

- subroutine, public [lcc_vectors_to_parameters](#) (lattice_vector, abc_angles, verbose)

  Transforms the lattice vectors into lattice parameters.
- subroutine, public [lcc_parameters_to_vectors](#) (abc_angles, lattice_vector, verbose)

  Transforms the lattice parameters into lattice vectors.
- subroutine, public [lcc_get_coordination](#) (r_at, r_env, thresh, cnum)

  Get the coordination of an atom.
- subroutine, public [lcc_canonical_basis](#) (lattice_vectors, r_inout, verbose)

  To "canonical base" transformation.
- subroutine, public [lcc_center_at_box](#) (lattice_vectors, r_inout, verbose)

  Cetering the system inside the lattice box.
- subroutine, public [lcc_center_at_origin](#) (r_inout, verbose)

  Cetering the system at the origin.
- real(dp) function, dimension(:,:), allocatable [inv](#) (A)

  Computes the inverse of a matrix using an LU decomposition.
- subroutine, public [lcc_get_reticular_density](#) (lattice_vectors, hkl_in, density)

  Get the reticular density of a particular hkl face: This soubroutine computes:
- real(dp) function, dimension(:), allocatable **crossprod** (r1, r2)
- subroutine, public **lcc_get_geometriccenter** (r_inout, geomCent)

5.2.1 Detailed Description

Module for auxiliary operations routines.

5.2.2 Function/Subroutine Documentation

5.2.2.1 inv()

```
real(dp) function, dimension(:,:), allocatable lcc_aux_mod::inv (
          real(dp), dimension(:,:), intent(in) A )
```

Computes the inverse of a matrix using an LU decomposition.

Parameters

A	nxn Matrix to be inverted.
Ainv	Inverse of matrix A

5.2.2.2 lcc_canonical_basis()

```
subroutine, public lcc_aux_mod::lcc_canonical_basis (
          real(dp), dimension(:,:), intent(inout), allocatable lattice_vectors,
```

```
          real(dp), dimension(:,:), intent(inout), allocatable r_inout,
          integer, intent(in) verbose )
```

To "canonical base" transformation.

This will reorient the shape/slab so that the first translation vector is alligned with x.

Parameters

lattice_vectors	Translation vectors for the shape/slab.
r_inout	Coordinates to be transformed.
verbose	Verbosity level.

5.2.2.3 lcc_center_at_box()

```
subroutine, public lcc_aux_mod::lcc_center_at_box (
          real(dp), dimension(:,:), intent(in), allocatable lattice_vectors,
          real(dp), dimension(:,:), intent(inout), allocatable r_inout,
          integer, intent(in) verbose )
```

Cetering the system inside the lattice box.

This will move the coordinates so that the geometric center of the system is at the center of the box.

Parameters

lattice_vectors	Translation vectors for the shape/slab.
r_inout	Coordinates to be transform.
verbose	Verbosity level.

5.2.2.4 lcc_center_at_origin()

```
subroutine, public lcc_aux_mod::lcc_center_at_origin (
          real(dp), dimension(:,:), allocatable r_inout,
          integer, intent(in) verbose )
```

Cetering the system at the origin.

This will move the coordinates so that the geometric center of the system is at (0,0,0).

Parameters

r_inout	Coordinates to be transform.
verbose	Verbosity level.

5.2.2.5 lcc_get_coordination()

```
subroutine, public lcc_aux_mod::lcc_get_coordination (
            real(dp), dimension(3), intent(in) r_at,
            real(dp), dimension(:,:), intent(in), allocatable r_env,
            real(dp), intent(in) thresh,
            integer, intent(inout) cnum )
```

Get the coordination of an atom.

Will count how many atoms are around a particular atom (coordination number) given a set radius.

Parameters

r_at	Coodinates of the atom for which we need the coordination.
r_env	Coordinated of the environment sorounding atom at r_at.
thres	Threshod distance to find coordinations.
cnum	Coordination number (output).

5.2.2.6 lcc_get_reticular_density()

```
subroutine, public lcc_aux_mod::lcc_get_reticular_density (
            real(dp), dimension(:,:), intent(in) lattice_vectors,
            real(dp), dimension(:), intent(in) hkl_in,
            real(dp), intent(out) density )
```

Get the reticular density of a particular hkl face: This soubroutine computes:

Parameters

lattice_vectors	Lattice vectors for the system.
hkl_in	Vector containing h, k, and l.
density	Reticular density.

5.2.2.7 lcc_parameters_to_vectors()

```
subroutine, public lcc_aux_mod::lcc_parameters_to_vectors (
            real(dp), dimension(2,3), intent(in) abc_angles,
            real(dp), dimension(3,3), intent(out) lattice_vector,
            integer, intent(in) verbose )
```

Transforms the lattice parameters into lattice vectors.

Parameters

abc_angles	2x3 array containing the lattice parameters. abc_angles(1,1) = a, abc_angles(1,2) = b, and abc_angles(1,3) = c abc_angles(2,1) = α , abc_angles(2,2) = β and abc_angles(2,3) = γ
lattice_vector	3x3 array containing the lattice vectors. lattice_vector(1,:) = $\overrightarrow{a}$
verbose	Verbosity level.

5.2.2.8 lcc_vectors_to_parameters()

```
subroutine, public lcc_aux_mod::lcc_vectors_to_parameters (
            real(dp), dimension(3,3), intent(in) lattice_vector,
            real(dp), dimension(2,3), intent(out) abc_angles,
            integer, intent(in) verbose )
```

Transforms the lattice vectors into lattice parameters.

Parameters

lattice_vector	3x3 array containing the lattice vectors. lattice_vector(1,:) = $\overrightarrow{a}$
abc_angles	2x3 array containing the lattice parameters. abc_angles(1,1) = a, abc_angles(1,2) = b and abc_angles(1,3) = c abc_angles(2,1) = α, abc_angles(2,2) = β, and abc_angles(2,3) = γ.
verbose	Verbosity level.

5.3 lcc_build_mod Module Reference

Module for generating the shapes after lattice is constructed.

Functions/Subroutines

- subroutine, public [lcc_bravais_growth](#) (nCycles, dTol, dTo, tCoordination, seed_file, r_inout)

  For "growing" a crystal shape using Bravias type of growth teory.
- subroutine, public [lcc_plane_cut](#) (planes, ploads, interPlanarDistances, lattice_vectors, cluster_lattice_↩ vectors, resindex, r_inout, verbose)

  Cutting a shape based on Miller planes.
- subroutine [lcc_build_slab](#) (slab, sloads, lattice_vectors, cluster_lattice_vectors, resindex, r_inout, verbose)

  Cutting a shape based on PBC vectors.
- subroutine, public [lcc_add_randomness_to_coordinates](#) (r_inout, seed, rcoeff)

  Will add randomness to the system.

5.3.1 Detailed Description

Module for generating the shapes after lattice is constructed.

5.3.2 Function/Subroutine Documentation

5.3.2.1 lcc_add_randomness_to_coordinates()

```
subroutine, public lcc_build_mod::lcc_add_randomness_to_coordinates (
            real(dp), dimension(:,:), intent(inout), allocatable r_inout,
            integer, intent(in) seed,
            real(dp), intent(in) rcoeff )
```

Will add randomness to the system.

Parameters

r_inout	System coordinates.
lattice_vectors	Lattice vectors.
seed	Random seed. rcoeff Coefficient for randomness.

5.3.2.2 lcc_bravais_growth()

```
subroutine, public lcc_build_mod::lcc_bravais_growth (
            integer, intent(in) nCycles,
            real(dp), intent(in) dTol,
            real(dp), intent(in) dTo,
            integer, intent(in) tCoordination,
            character(len=*), intent(in) seed_file,
            real(dp), dimension(:,:), intent(inout), allocatable r_inout )
```

For "growing" a crystal shape using Bravias type of growth teory.

Parameters

nCycles	Number of shells to add.
dTol	Tolerance for distinguising the coordinates from the seed to the coodinates from the bulk.
dTo	Parameter to determine the coordination the incoming atom.
tCoordination	Target coordination. If coodination is larger than the target, the atom will be picked.
seed_file	Name of the file containing the seed.
r_inout	Input: Bulk lattice, Output: Crystal shape.

[Todo](#) Optimize the routine.

5.3.2.3 lcc_build_slab()

```
subroutine lcc_build_mod::lcc_build_slab (
            real(dp), dimension(:,:), intent(in), allocatable slab,
            real(dp), dimension(:), intent(in), allocatable sloads,
            real(dp), dimension(:,:), intent(inout), allocatable lattice_vectors,
            real(dp), dimension(:,:), intent(inout), allocatable cluster_lattice_vectors,
            integer, dimension(:), allocatable resindex,
            real(dp), dimension(:,:), intent(inout), allocatable r_inout,
            integer, intent(in) verbose )
```

Cutting a shape based on PBC vectors.

A set of PBC vectors and distances is provided.

Parameters

planes	List of planes to cut the shape with.
ploads	Distance from the origin to locate the plane.
interPlanarDistance	Use "interplanar distances" as measure for the cut.
lattice_vectors	Lattice vectors.
cluster_lattice_vectors	Lattice vectors of the shape. Note: this only makes sense if the planes make a parellelepiped.
r_inout	Coordinates in and out.
verbose	Verbosity level.

5.3.2.4 lcc_plane_cut()

```
subroutine, public lcc_build_mod::lcc_plane_cut (
            real(dp), dimension(:,:), intent(in), allocatable planes,
            real(dp), dimension(:), intent(in), allocatable ploads,
            logical, intent(in) interPlanarDistances,
            real(dp), dimension(:,:), intent(inout), allocatable lattice_vectors,
            real(dp), dimension(:,:), intent(inout), allocatable cluster_lattice_vectors,
            integer, dimension(:), allocatable resindex,
            real(dp), dimension(:,:), intent(inout), allocatable r_inout,
            integer, intent(in) verbose )
```

Cutting a shape based on Miller planes.

A set of panes and distances is provided.

Parameters

planes	List of planes to cut the shape with.
ploads	Distance from the origin to locate the plane.
interPlanarDistance	Use "interplanar distances" as measure for the cut.
lattice_vectors	Lattice vectors.
cluster_lattice_vectors	Lattice vectors of the shape. Note: this only makes sense if the planes make a parellelepiped.
r_inout	Coordinates in and out.
verbose	Verbosity level.

5.4 lcc_check_mod Module Reference

Module for checking operations routines.

Functions/Subroutines

- subroutine, public lcc_check_periodicity (r_in, lattice_vectors, r_ref, tol, verbose)
  Check the periodicity.

5.4.1 Detailed Description

Module for checking operations routines.

5.4.2 Function/Subroutine Documentation

5.4.2.1 lcc_check_periodicity()

```
subroutine, public lcc_check_mod::lcc_check_periodicity (
            real(dp), dimension(:,:), intent(in), allocatable r_in,
            real(dp), dimension(:,:), intent(in), allocatable lattice_vectors,
            real(dp), dimension(:,:), intent(in), allocatable r_ref,
            real(dp), intent(in) tol,
            integer, intent(in) verbose )
```

Check the periodicity.

Will use a "brute force" approach to check periodidity.

Parameters

r_in	Input coordinates.
lattice_vectors	Translation vectors for the slab.
r_ref	Reference or "bulk structure from where the shape was cut.
verbose	Verbosity level.

5.5 lcc_compute_mod Module Reference

Template module for contributing.

Functions/Subroutines

- subroutine, public lcc_compute_roughness (coords, lattice_vectors, isoval, rab, ni, nj, nk, verbose)
  Example subroutine.

5.5.1 Detailed Description

Template module for contributing.

5.5.2 Function/Subroutine Documentation

5.5.2.1 lcc_compute_roughness()

```
subroutine, public lcc_compute_mod::lcc_compute_roughness (
            real(dp), dimension(:,:), intent(in), allocatable coords,
            real(dp), dimension(:,:), intent(in), allocatable lattice_vectors,
            real(dp), intent(in) isoval,
            real(dp), intent(in) rab,
            integer, intent(in) ni,
            integer, intent(in) nj,
            integer, intent(in) nk,
            integer, intent(in) verbose )
```

Example subroutine.

Parameters

coords	Coordinates.
lattice_vectors	Lattice vectors.
isoval	Parameter value to compute isosurface.
rab	Radius of the spherical probe.
ni	Number of discrete points on the a1 axis.
nj	Number of discrete points on the a2 axis.
nk	Number of discrete points on the a3 axis.
verbose	Verbosity level.

5.6 lcc_constants_mod Module Reference

A module to handle the constants needed by the code.

Variables

- integer, parameter, public dp = kind(1.0d0)

  Precision used troughout the code.
- real(dp), parameter pi = 3.14159265358979323846264338327950_dp

  Pi number.

5.6.1 Detailed Description

A module to handle the constants needed by the code.

This module will be used to store the constants needed in the code

5.7 lcc_lattice_mod Module Reference

Module to hold routines for handling the lattice and lattice base.

Functions/Subroutines

- subroutine, public lcc_make_lattice (bld, ltt, check, sy)

  Make a lattice depending on the input parameter.
- subroutine lcc_read_base (bld, ltt, check, verbose)

  Reading the basis from an input file.
- subroutine lcc_check_basis (base_format, r_base, lattice_vectors, verbose)

  Routine to check for atom repetitions in basis \bnrief It will do all possible translations searching for atoms that could be repeated.
- subroutine lcc_add_base_to_cluster (ltt, sy, verbose)

  Add a basis to the lattice.
- subroutine lcc_sc (Nx1, Nx2, Ny1, Ny2, Nz1, Nz2, h_lattice_a, supra_lattice_vectors, r_sy)

  Simple cubic (SC) lattice construction.
- subroutine lcc_fcc (Nx1, Nx2, Ny1, Ny2, Nz1, Nz2, h_lattice_a, supra_lattice_vectors, r_sy, verbose)

  Face center cubic (FCC) lattice construction.
- subroutine lcc_triclinic (Nx1, Nx2, Ny1, Ny2, Nz1, Nz2, lattice_vectors, supra_lattice_vectors, r_sy, verbose)

  Triclinic lattice construction.
- subroutine, public lcc_set_atom_type (a_type, atom_symbol, atom_name, nats)

  Sets the atom type.
- subroutine lcc_add_randomness (r_inout, lattice_vectors, seed, rcoeff)

  Will add randomness to the system.
- subroutine lcc_minimize_from (xVar, i, ai, nats, trs, verbose)

  To get the best translation that minimizes the distance to any previous fragment.
- subroutine lcc_get_besttranslations (nop, nats, r_inout, verbose)

  To get the best translation that minimizes the distance to any previous fragment.

5.7.1 Detailed Description

Module to hold routines for handling the lattice and lattice base.

5.7.2 Function/Subroutine Documentation

5.7.2.1 lcc_add_base_to_cluster()

```
subroutine lcc_lattice_mod::lcc_add_base_to_cluster (
            type(lattice_type) ltt,
            type(system_type), intent(inout) sy,
            integer, intent(in) verbose )
```

Add a basis to the lattice.

This routine will add the basis to the system points previously cut from the lattice. This is the last step of the solid/shape/slab creation.

Parameters

ltt	lattice_type See lcc_structs_mod
sy	system_type See progress library

5.7.2.2 lcc_add_randomness()

```
subroutine lcc_lattice_mod::lcc_add_randomness (
            real(dp), dimension(:,:), intent(inout), allocatable r_inout,
            real(dp), dimension(:,:), intent(in), allocatable lattice_vectors,
            integer, intent(in) seed,
            real(dp), intent(in) rcoeff )
```

Will add randomness to the system.

Parameters

r_inout	System coordinates.
lattice_vectors	Lattice vectors.
seed	Random seed. rcoeff Coefficient for randomness.

5.7.2.3 lcc_check_basis()

```
subroutine lcc_lattice_mod::lcc_check_basis (
            character(len=*), intent(in) base_format,
            real(dp), dimension(:,:), intent(in), allocatable r_base,
            real(dp), dimension(:,:), intent(in), allocatable lattice_vectors,
            integer, intent(in) verbose )
```

Routine to check for atom repetitions in basis \bnrief It will do all possible translations searching for atoms that could be repeated.

Parameters

base_format	Basis format, if xyz of abc
r_base	Coordinates of the basis. r_base(1,7) means coordinate x of atom 7
lattice_vectors	Lattice vectors. WARNING, in this case lattice_vector(1,3) means the coordinate 3=z of vector 1.

5.7.2.4 lcc_fcc()

```
subroutine lcc_lattice_mod::lcc_fcc (
            integer, intent(in) Nx1,
```

```
                    integer, intent(in) Nx2,
                    integer, intent(in) Ny1,
                    integer, intent(in) Ny2,
                    integer, intent(in) Nz1,
                    integer, intent(in) Nz2,
                    real(dp), intent(in) h_lattice_a,
                    real(dp), dimension(:,:), intent(inout), allocatable supra_lattice_vectors,
                    real(dp), dimension(:,:), intent(inout), allocatable r_sy,
                    integer, intent(in) verbose )
```

Face center cubic (FCC) lattice construction.

Constructs a "bulk" of Face center cubic lattice.

Parameters

Nx1	Initial x lattice point.
Nx2	Final x lattice point.
Ny1	Initial y lattice point.
Ny2	Final y lattice point.
Nz1	Initial z lattice point.
Nz2	Final z lattice point.
h_lattice_a	Lattice parameter.
supra_lattice_vectors	Lattice unit vectors of the resulting slab.
r_sy	Output system coordinates.

5.7.2.5 lcc_get_besttranslations()

```
subroutine lcc_lattice_mod::lcc_get_besttranslations (
                    integer, intent(in) nop,
                    integer, intent(in) nats,
                    real(dp), dimension(:,:), intent(inout), allocatable r_inout,
                    integer, intent(in) verbose )
```

To get the best translation that minimizes the distance to any previous fragment.

Parameters

nop	Number of symmetry operations
nats	Number of atoms in each fragment
r_inout	Coordinates for the fragment
verbose	Verbosity level

5.7.2.6 lcc_make_lattice()

```
subroutine, public lcc_lattice_mod::lcc_make_lattice (
                    type(build_type), intent(inout) bld,
```

```
        type(lattice_type), intent(inout) ltt,
        logical, intent(in) check,
        type(system_type), intent(inout) sy )
```

Make a lattice depending on the input parameter.

This will make one of the following latices: SC: Simple cubic, FCC: Face center cubic, or Triclinic.

Parameters

bld	Building structure (see lcc_structures_mod)
ltt	Lattice structure (see lcc_scturctures_mod)
check	If we want to check the basis for atom repetition. Note that checks can be expensive.

5.7.2.7 lcc_minimize_from()

```
subroutine lcc_lattice_mod::lcc_minimize_from (
        real(dp), dimension(:,:), intent(in) xVar,
        integer, intent(in) i,
        integer, intent(in) ai,
        integer, intent(in) nats,
        real(dp), dimension(3), intent(inout) trs,
        integer, intent(in) verbose )
```

To get the best translation that minimizes the distance to any previous fragment.

Parameters

xVar	Coordinates of the full basis (including symmetry operations).
i	Fragmet being added at the "i" operation.
ai	Atom index to translate and get the optimal translation.
nats	Number of atoms in the fragment.
trs	Optimal translation.

5.7.2.8 lcc_read_base()

```
subroutine lcc_lattice_mod::lcc_read_base (
        type(build_type), intent(inout) bld,
        type(lattice_type), intent(inout) ltt,
        logical, intent(in) check,
        integer, intent(in) verbose )
```

Reading the basis from an input file.

This will read the coordinates for the basis from an input file If information about the lattice is contained, it will also be read.

Parameters

bld	Building structure (see lcc_structures_mod).
ltt	Lattice structure (see lcc_scturctures_mod).
check	If we want to check the basis for atom repetition.
verbose	Verbose level. Note that checks can be expensive.

5.7.2.9 lcc_sc()

```
subroutine lcc_lattice_mod::lcc_sc (
            integer, intent(in) Nx1,
            integer, intent(in) Nx2,
            integer, intent(in) Ny1,
            integer, intent(in) Ny2,
            integer, intent(in) Nz1,
            integer, intent(in) Nz2,
            real(dp), intent(in) h_lattice_a,
            real(dp), dimension(:,:), intent(inout), allocatable supra_lattice_vectors,
            real(dp), dimension(:,:), intent(inout), allocatable r_sy )
```

Simple cubic (SC) lattice construction.

Constructs a "bulk" of Simple Cubic lattice.

Parameters

Nx1	Initial x lattice point.
Nx2	Final x lattice point.
Ny1	Initial y lattice point.
Ny2	Final y lattice point.
Nz1	Initial z lattice point.
Nz2	Final z lattice point.
h_lattice_a	Lattice parameter.
supra_lattice_vectors	Lattice unit vectors of the resulting slab.
r_sy	Output system coordinates.

5.7.2.10 lcc_set_atom_type()

```
subroutine, public lcc_lattice_mod::lcc_set_atom_type (
            character(len=2), intent(in) a_type,
            character(len=2), dimension(:), intent(inout), allocatable atom_symbol,
            character(len=3), dimension(:), intent(inout), allocatable atom_name,
            integer, intent(in) nats )
```

Sets the atom type.

Sets the atom "symbol/type/name."

Parameters

a_type	Atom symbol character.
atom_symbol	Atom symbols.
atom_name	Atom name. Note: Atom name is a tag that can distinguish atoms with same symbol.

5.7.2.11 lcc_triclinic()

```
subroutine lcc_lattice_mod::lcc_triclinic (
            integer, intent(in) Nx1,
            integer, intent(in) Nx2,
            integer, intent(in) Ny1,
            integer, intent(in) Ny2,
            integer, intent(in) Nz1,
            integer, intent(in) Nz2,
            real(dp), dimension(:,:), intent(in), allocatable lattice_vectors,
            real(dp), dimension(:,:), intent(inout), allocatable supra_lattice_vectors,
            real(dp), dimension(:,:), intent(inout), allocatable r_sy,
            integer, intent(in) verbose )
```

Triclinic lattice construction.

Constructs a "bulk" of triclinic lattice.

Parameters

Nx1	Initial x lattice point.
Nx2	Final x lattice point.
Ny1	Initial y lattice point.
Ny2	Final y lattice point.
Nz1	Initial z lattice point.
Nz2	Final z lattice point.
lattice_vectors	Lattice vectors.
supra_lattice_vectors	Lattice unit vectors of the resulting slab.
r_sy	Output system coordinates. Note: Unit cell representation has to be transformed from edges and angles to vetors before calling this routine.

Todo A angles_to_vectors transformation will be available.

5.8 lcc_lib Module Reference

Library module.

Functions/Subroutines

- subroutine, public **lcc** (readInputFile, inputFileName, syOut, writeOut, clType, planeIn)

5.8.1 Detailed Description

Library module.

5.9 lcc_mc_mod Module Reference

Module for Monte Carlo related routines.

Functions/Subroutines

- subroutine lcc_check_system (r, iter, temp, cost, cost0)

  Maximize: This checks the acceptance.

5.9.1 Detailed Description

Module for Monte Carlo related routines.

5.10 lcc_message_mod Module Reference

Module for printing through the code.

Functions/Subroutines

- subroutine, public lcc_print_ussage ()

  For printing the instructions on how to execute the code.
- subroutine, public lcc_print_message (message, verbose)

  Print a simple message.
- subroutine, public lcc_print_warning (at, message, verbose)

  Print a Warning (will not stop execution).
- subroutine, public lcc_print_error (at, message)

  Print error (will stop execution).
- subroutine, public lcc_print_intval (name, value, units, verbose)

  Print integer magnitude.
- subroutine, public lcc_print_realval (name, value, units, verbose)

  Print real magnitude.
- subroutine, public lcc_print_realvect (name, vect, units, verbose)

  Print real vector.
- subroutine lcc_print_realmat (name, mat, units, verbose)

  Print real vector.
- subroutine **lcc_help** ()

5.10.1 Detailed Description

Module for printing through the code.

5.10.2 Function/Subroutine Documentation

5.10.2.1 lcc_print_error()

```
subroutine, public lcc_message_mod::lcc_print_error (
            character(len=*), intent(in) at,
            character(len=*), intent(in) message )
```

Print error (will stop execution).

Parameters

at	Name of the routine.
message	Message to print.

5.10.2.2 lcc_print_intval()

```
subroutine, public lcc_message_mod::lcc_print_intval (
            character(len=*), intent(in) name,
            integer, intent(in) value,
            character(len=*), intent(in) units,
            integer, intent(in) verbose )
```

Print integer magnitude.

Parameters

name	Name of the magnitude.
value	Value to print.
units	Units of the magnitude.

5.10.2.3 lcc_print_message()

```
subroutine, public lcc_message_mod::lcc_print_message (
            character(len=*), intent(in) message,
            integer, intent(in) verbose )
```

Print a simple message.

Parameters

message	Message to print.
verbose	Verbosity level.

5.10.2.4 lcc_print_realmat()

```
subroutine lcc_message_mod::lcc_print_realmat (
            character(len=*), intent(in) name,
            real(dp), dimension(:,:), intent(in), allocatable mat,
            character(len=*), intent(in) units,
            integer, intent(in) verbose )
```

Print real vector.

Parameters

name	Name of the quantities.
mat	Matrix to print.
units	Units of the quantities.
verbose	Verbosity level.

5.10.2.5 lcc_print_realval()

```
subroutine, public lcc_message_mod::lcc_print_realval (
            character(len=*), intent(in) name,
            real(dp), intent(in) value,
            character(len=*), intent(in) units,
            integer, intent(in) verbose )
```

Print real magnitude.

Parameters

name	Name of the magnitude.
value	Value to print.
units	Units of the magnitude.

5.10.2.6 lcc_print_realvect()

```
subroutine, public lcc_message_mod::lcc_print_realvect (
            character(len=*), intent(in) name,
            real(dp), dimension(:), intent(in), allocatable vect,
            character(len=*), intent(in) units,
            integer, intent(in) verbose )
```

Print real vector.

Parameters

name	Name of the quantities.
vect	Vector to print.
units	Units of the quantities.
verbose	Verbosity level.

5.10.2.7 lcc_print_warning()

```
subroutine, public lcc_message_mod::lcc_print_warning (
            character(len=*), intent(in) at,
            character(len=*), intent(in) message,
            integer, intent(in) verbose )
```

Print a Warning (will not stop execution).

Parameters

at	Name of the routine.
message	Message to print.
verbose	Verbosity level.

5.11 lcc_parser_mod Module Reference

This module controls the initialization of the variables.

Functions/Subroutines

- subroutine, public lcc_parse (filename, bld, ltt, cmp)

    Clustergen parser.
- subroutine, public lcc_make_sample_input ()

    Make a sample inputfile sample_input.in.
- subroutine, public lcc_write_coords (sy, bld, coordsout_file, verbose)

    Writes the coordinates to a file (coordsandbase.pdb)

5.11.1 Detailed Description

This module controls the initialization of the variables.

5.11.2 Function/Subroutine Documentation

5.11.2.1 lcc_parse()

```
subroutine, public lcc_parser_mod::lcc_parse (
            character(len=*), intent(in) filename,
            type(build_type), intent(inout) bld,
            type(lattice_type), intent(inout) ltt,
            type(compute_type), intent(inout) cmp )
```

Clustergen parser.

This module is used to parse all the input variables for this program. Adding a new input keyword to the parser:

- If the variable is real, we have to increase nkey_re.

- Add the keyword (character type) in the keyvector_re vector.

- Add a default value (real type) in the valvector_re.

- Define a new variable and pass the value through valvector_re(num) where num is the position of the new keyword in the vector.

Parameters

filename	File name for the input.
bld	Build type.
ltt	Lattice type.

5.11.2.2 lcc_write_coords()

```
subroutine, public lcc_parser_mod::lcc_write_coords (
            type(system_type) sy,
            type(build_type) bld,
            character(len=*) coordsout_file,
            integer, intent(in) verbose )
```

Writes the coordinates to a file (coordsandbase.pdb)

Parameters

sy	System type.
bld	Build type.
coordsout_file	File name to write the coordinates to.
verbose	Verbosity level.

5.12 lcc_regular_mod Module Reference

Module for generating regular shapes after lattice is constructed.

Functions/Subroutines

- subroutine, public lcc_spheroid (a_axis, b_axis, c_axis, r_inout)

    For building spheroidal shapes out of a bulk lattice.

5.12.1 Detailed Description

Module for generating regular shapes after lattice is constructed.

5.12.2 Function/Subroutine Documentation

5.12.2.1 lcc_spheroid()

```
subroutine, public lcc_regular_mod::lcc_spheroid (
            real(dp) a_axis,
            real(dp) b_axis,
            real(dp) c_axis,
            real(dp), dimension(:,:), allocatable r_inout )
```

For building spheroidal shapes out of a bulk lattice.

Parameters

a_axis	Lenght in the x direction.
b_axis	Lenght in the y direction.
c_axis	Lenght in the z direction.
r_inout	Input and output coordinates.

5.13 lcc_string_mod Module Reference

Module for manipulating strings.

Functions/Subroutines

- subroutine, public lcc_get_word (string, posh, post, word)

    Cut a word from string.
- subroutine, public lcc_split_string (string, delimit, head, tail)

    Split a string in two words uning a delimiter.

5.13.1 Detailed Description

Module for manipulating strings.

5.13.2 Function/Subroutine Documentation

5.13.2.1 lcc_get_word()

```
subroutine, public lcc_string_mod::lcc_get_word (
            character(len=*), intent(in) string,
            integer, intent(in) posh,
            integer, intent(in) post,
            character(20), intent(inout) word )
```

Cut a word from string.

Parameters

string	Full string.
posh	Cut from position.
post	Cut to position.
word	Extracted word.

5.13.2.2 lcc_split_string()

```
subroutine, public lcc_string_mod::lcc_split_string (
            character(len=*), intent(in) string,
            character(1), intent(in) delimit,
            character(20), intent(inout) head,
            character(20), intent(inout) tail )
```

Split a string in two words uning a delimiter.

Parameters

string	Full string.
delimit	Delimiter.
head	First word.
tail	Last word.

5.14 lcc_structs_mod Module Reference

A module to handle the structures needed by the code.

Data Types

- type build_type

Build type.

- type [compute_type](#)

  Compute type.

- type [lattice_type](#)

  Lattice type to be read and extended.

5.14.1 Detailed Description

A module to handle the structures needed by the code.

This module will be used to build and handle structures in the code.

5.15 lcc_template_mod Module Reference

Template module for contributing.

Functions/Subroutines

- subroutine, public [lcc_template_subroutine](#) (dummy, verbose)

  Example subroutine.

5.15.1 Detailed Description

Template module for contributing.

5.15.2 Function/Subroutine Documentation

5.15.2.1 lcc_template_subroutine()

```
subroutine, public lcc_template_mod::lcc_template_subroutine (
            character(*), intent(in) dummy,
            integer, intent(in) verbose )
```

Example subroutine.

Parameters

dummy	Example variable
verbose	Verbosity level.

Chapter 6

Class Documentation

6.1   lcc_structs_mod::build_type Type Reference

Build type.

Public Attributes

- character(len=20) job_name

  Job name.
- character(len=20) output_file_name

  Output file name.
- character(len=60), public coordsout_file

  Output file name for coordinates.
- character(len=60), public latticebase_file

  Lattice base file name.
- character(len=1) cut_by_planes

  Cut lattice using planes.
- character(len=1) cut_with_base

  Cut lattice after base is added.
- character(len=1) read_lattice_from_file

  Read lattice from file.
- character(len=1) use_lattice_base

  Use lattice base.
- character(len=60) cl_type

  Cluster (or solid) shape to be constructed.
- character(len=60) planes_type

  Type of planes used for the cut.
- character(len=60) seed_file

  File name for the seed used to grow a cluster.
- integer n

  Number of atoms.
- integer nplanes

  Number of planes to use in the cut.
- integer nx1

  Number of lattice points in +-(x, y, and z) directions.

- integer **nx2**
- integer **ny1**
- integer **ny2**
- integer **nz1**
- integer **nz2**
- integer seed

    Random seed.
- integer cl_number

    Cluster number (if it is a solid with "magic" numbers)
- real(dp) a_axis

    Axis length if cluster is a spheroid.
- real(dp) **b_axis**
- real(dp) **c_axis**
- real(dp) rcoeff

    Coefficient used with random seed to create noise in coordinates.
- real(dp) r_cut

    Cutoff radius to build spheroids.
- real(dp) trunc

    Truncation for solids.
- character(2) a_type

    Atom type (if specified on the input file)
- real(dp), dimension(:,:), allocatable planes

    Planes for the cut.
- real(dp), dimension(:), allocatable ploads

    Plenes weight factors.
- type(system_type) syseed

    System seed to be grow on top.
- integer ncluster

    Number of atoms in cluster/slab.
- character(2), dimension(:), allocatable atom_in

    Atoms in the cluster/slab.
- character(2), dimension(:), allocatable **atomname_in**
- integer, dimension(:), allocatable **resindex_in**
- character(2), dimension(:), allocatable **resname_in**
- real(dp), dimension(:,:), allocatable r_cluster

    Coordinates of the resulting cluster/slab.
- integer maxcoordination

    Max coordination number.
- real(dp) rtol

    Distance tolerance for distinguishing coordinates.
- integer niter

    Number of iterations.
- integer verbose

    Verbose level.
- logical center

    Center at box.
- logical reorient

    Reorient first lattice vector toward x direction.
- logical writecml

    Reorient first lattice vector toward x direction.
- logical checkperiod

To check periodicity.

- character(5) rdfpair

    To compute RDFs.

- logical writelmp

    Write LAMMPS input coordinates.

- logical interplanardistances

    Use "number of interplanar distances" as unit of measurement for plane cut.

- real(dp), dimension(:,:), allocatable slab

    To build a slab out of regular vectors.

- real(dp), dimension(:), allocatable **sloads**

- logical randomcoordinates

    To add randomness to coordinates.

6.1.1 Detailed Description

Build type.

The documentation for this type was generated from the following file:

- /tmp/LCC/src/lcc_structs_mod.F90

6.2 lcc_structs_mod::compute_type Type Reference

Compute type.

Public Attributes

- logical computeroughness

    Compute surface roughness.

- real(dp) roughnessisoval

    Surface roughness parameters.

- real(dp) **roughnessrab**

- integer **roughnessni**

- integer **roughnessnj**

- integer **roughnessnk**

6.2.1 Detailed Description

Compute type.

The documentation for this type was generated from the following file:

- /tmp/LCC/src/lcc_structs_mod.F90

6.3 lcc_structs_mod::lattice_type Type Reference

Lattice type to be read and extended.

Public Attributes

- character(len=3) base_format

    Lattice basis.
- character(len=60) primitive_format

    The lattice primitive format (Angles of Vectors)
- character(len=60) type_of_lattice

    Type of lattice (sc, bcc, fcc, and triclinic)
- real(dp) angle_alpha

    Angles for triclinic lattice.
- real(dp) **angle_beta**
- real(dp) **angle_gamma**
- real(dp) h_lattice_a

    abc parameters for lattice
- real(dp) **h_lattice_b**
- real(dp) **h_lattice_c**
- real(dp), dimension(:,:), allocatable lattice_vectors

    Lattice vectors.
- real(dp) volr

    Volume of the cell.
- real(dp), dimension(:,:), allocatable recip_vectors

    Lattice reciprocal vectors.
- real(dp) volk

    Volume of the reciprocal cell.
- integer nbase

    Number of atoms in the basis.
- character(2), dimension(:), allocatable base_atom

    Basis atoms.
- real(dp), dimension(:,:), allocatable r_base

    Basis coordinates.
- type(system_type) sybase

    System for the basis.
- logical bsopl

    If there are symmetry operations to be performed.
- integer nop

    Number of Symmetry operations.
- real(dp), dimension(:,:), allocatable bstr

    Translations to be performed.
- real(dp), dimension(:), allocatable bsopload

    Scaling factos (load) for the translation.
- real(dp), dimension(:,:), allocatable bssym

    Symmetry operation (diagonal)
- integer, dimension(:), allocatable spindex

    Spicies index.
- real(dp), dimension(:), allocatable base_mass

    System basis masses.

- integer, dimension(:), allocatable resindex

    Residue index.
- real(dp), dimension(:,:), allocatable bulk

    To save the "bulk" positions.
- logical check

    Check lattice.
- logical getopttrs

    Get optimal translations at symmetry operations.
- logical randomlattice

    To add randomness to each lattice position.
- logical randrotations

    To add random orientations to each lattice molecules/basis.
- real(dp) setdensity

    To set a particular density in [gr/cc].

6.3.1 Detailed Description

Lattice type to be read and extended.

The type of lattice read from input.

The documentation for this type was generated from the following file:

- /tmp/LCC/src/lcc_structs_mod.F90

Index