

MATSDP

The materials simulation and data
processing toolkit

version 0.2.3
Apr. 20, 2021

Contents

1	Introduction	7
1.1	Functions	7
1.2	Requirements	8
1.3	Installation	11
1.4	Usage	11
1.4.1	Running with Python environment	11
1.4.2	Running Graphical User Interface (GUI) application	11
1.5	Notes	14
2	Subpackage: vasp	15
2.1	vasp_build module	15
2.1.1	vasp_build.substitution	15
2.1.2	vasp_build.selection_sphere	17
2.1.3	vasp_build.transform	19
2.1.4	vasp_build.create_supercell	20
2.1.5	vasp_build.build_vacuum_layer	21
2.1.6	vasp_build.exfoliation	22
2.2	vasp_plot module	23
2.2.1	vasp_plot.plot_poscar	23
2.2.2	vasp_plot.plot_poscar_for_workdir	30
2.2.3	vasp_plot.plot_dos	33
2.2.4	vasp_plot.plot_band	41
2.3	vasp_read module	48
2.3.1	vasp_read.read_doscar	48
2.4	vasp_analyze module	49
2.4.1	vasp_analyze.nn_map	49
2.4.2	vasp_analyze.simple_cna	50
2.4.3	vasp_analyze.estruct	51

2.4.4	vasp_analyze.overlap_peak_analyzer	53
2.4.5	vasp_analyze.job_status	54
2.4.6	Output	55
2.5	vasp_write module	55
2.5.1	vasp_write.write_poscar_with_force	55
3	Subpackage: apt	59
3.1	apt_read module	59
3.1.1	apt_read.read_proxigram_csv	59
3.2	apt_plot module	60
3.2.1	apt_plot.plot_proxigram_csv	60
4	Subpackage: dvm	63
4.1	dvm_build module	63
4.1.1	create_multiple_dvm_jobs	63
4.2	dvm_read module	65
4.2.1	dvm_read.read_input	65
4.2.2	dvm_read.read_ind	66
4.2.3	dvm_read.read_incar	66
4.2.4	dvm_read.read_otput	67
4.3	dvm_analyze module	67
4.3.1	dvm_analyze.ie_nn	67
4.3.2	dvm_analyze.job_status	69
4.3.3	Output	69
4.4	dvm_write module	69
4.4.1	dvm_write.write_input	69
4.4.2	dvm_write.write_ind	70
4.4.3	dvm_write.write_ie	71
5	Subpackage: pms	73
5.1	task_manager module	73
5.1.1	task_manager.write_task_summary	73
6	Other functions	75
6.1	funcs.py	75
6.1.1	cp()	75
6.1.2	write_file()	75
6.1.3	merge_files	75

<i>CONTENTS</i>	5
6.1.4 <code>dir_tree()</code>	75
7 Tests	77
A Other plotting settings	79
A.1 Named colors in the program	79

Chapter 1

Introduction

MATSDP is a materials simulation and data processing toolkit. The Vienna ab-initio simulation package (VASP) and the Three-dimensional atom probe tomography (APT) analyzing and data processing tools are included.

1.1 Functions

VASP analyzing and data processing tools:

- Build model by the following methods: atom substitution, atom selection, exfoliation (2D), make supercell, transformation (rotation + translation), reorientation, adding vacuum layer etc.
- Read and write VASP inputs/outputs.
- Visualization of model and results: Plot model based on the POSCAR/-CONTCAR file, also support color mapping of atom properties); Plot DOS (PDOS, LDOS, TDOS); Plot band structure (including fat band).
- Analyzing tools: Calculate the nearest neighbor information, perform simple common neighbor analysis, calculate structural energy, overlap peak analyzer of DOS, get band gap.
- VASP tools: Check VASP errors/warnings and give solutions; Check the job status of multiple jobs; Check lattice parameters of multiple VASP jobs; Conversion of coordinate systems (Fractional/Cartesian);

- Job management: Processing of multiple VASP jobs automatically; Write task summary/report of multiple VASP jobs.

APT postprocessing tools:

- Read the concentration profile *.csv file
- Plot the concentration profile

DVM tools:

- Read the *.input, *.incar, *.otput files
- Write the *.input, *.incar, IND.DAT files
- Write the interatomic energy (IE) files (including the IEs of the first nearest neighbor atoms)
- The *.incar file can also be prepared by atom selection from the vasp_build function in the vasp module

PMS tools:

- Job management: Processing of multiple VASP jobs automatically
- Write task summary (of VASP jobs)

Others tools:

- file format conversion
- fig2pdf (converting multiple images to a single .pdf file)

The matsdp package contains the vasp module, the apt module and the dvm module as shown in Figure 1.1. The structures of the vasp module, the apt module and the dvm module are shown in Figure 1.2, Figure 1.3, and Figure 1.4.

1.2 Requirements

- numpy
- scipy
- scikit-learn
- matplotlib

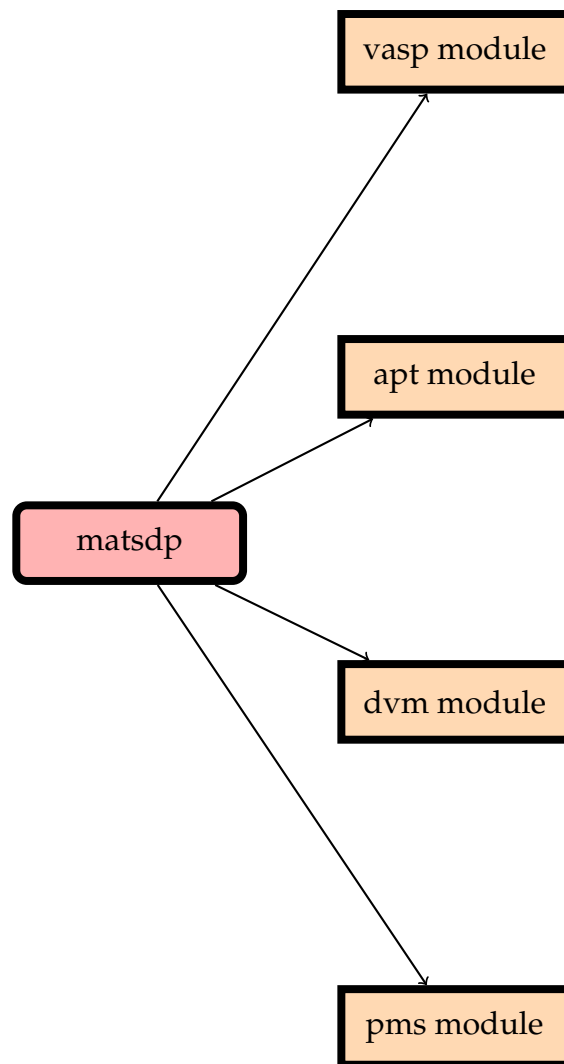


Figure 1.1: subpackages of the matsdp program.

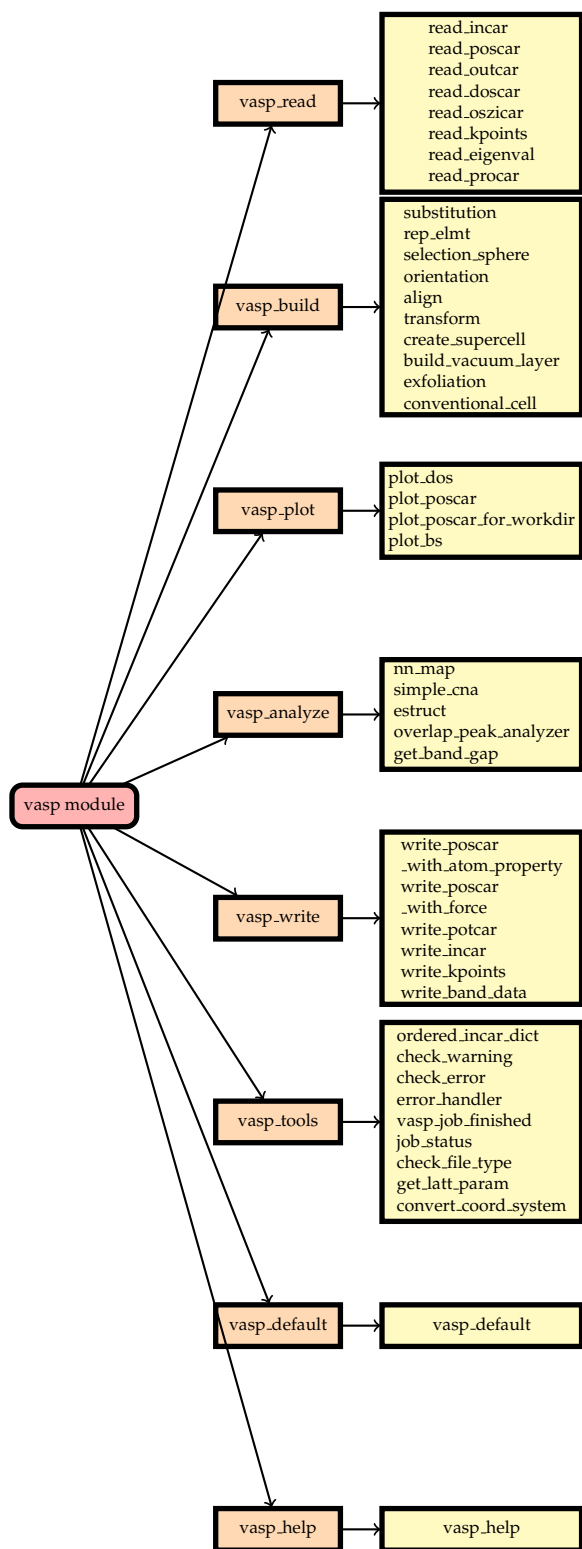


Figure 1.2: vasp module.

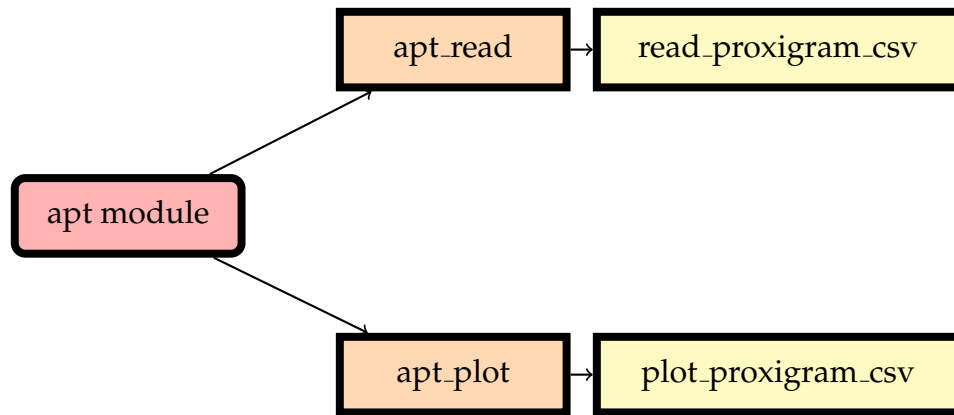


Figure 1.3: apt module.

1.3 Installation

For the Python users, the package can be retrieved by the following command.

```
pip install matsdp
```

For the GUI users, please run the matsdp_gui.exe directly (part of the functions are realized in the GUI version).

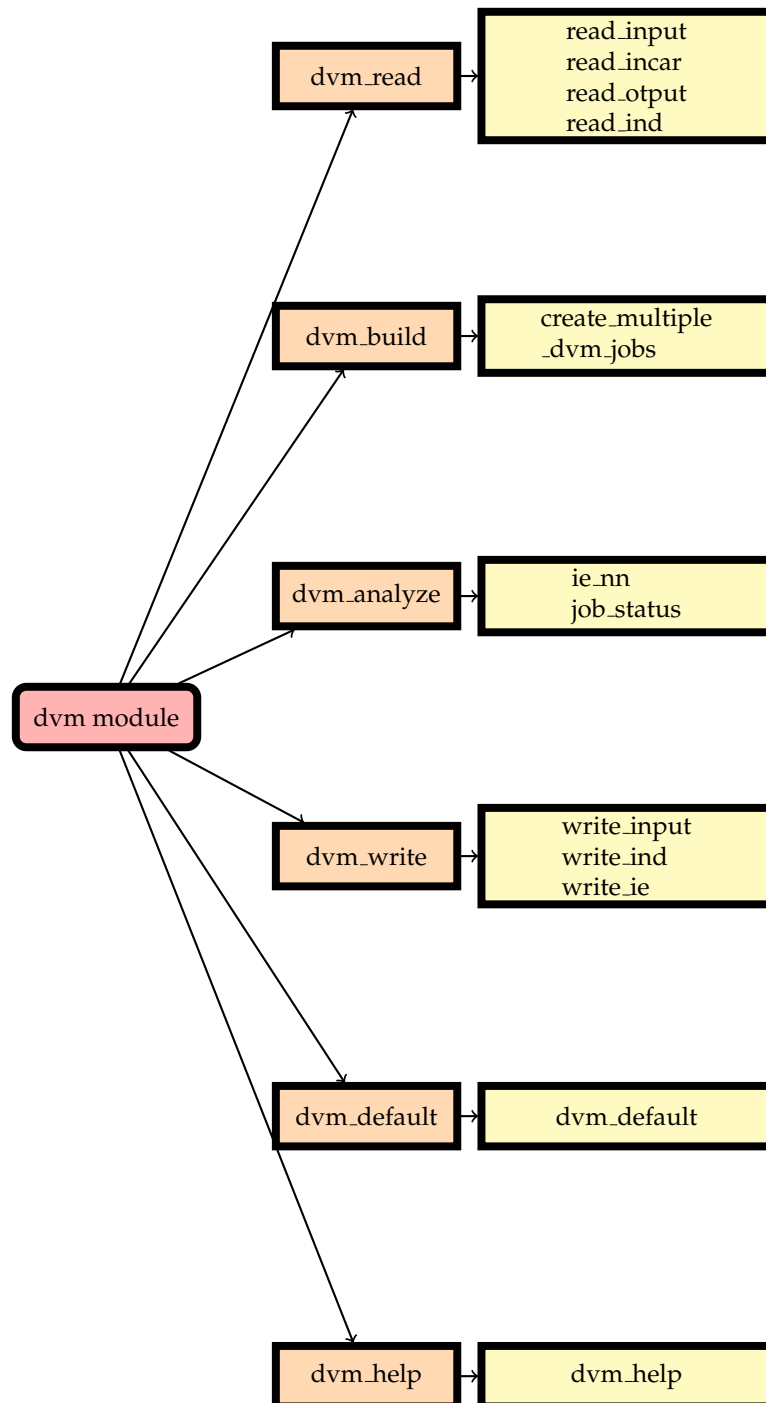
1.4 Usage

1.4.1 Running with Python environment

After installing the matsdp package, the program can be used by importing the modules and call the related functions.

1.4.2 Running Graphical User Interface (GUI) application

The program provides a graphical user interface (matsdp_gui.exe). The GUI is shown in the Figure 1.5:

Figure 1.4: `dvm` module.

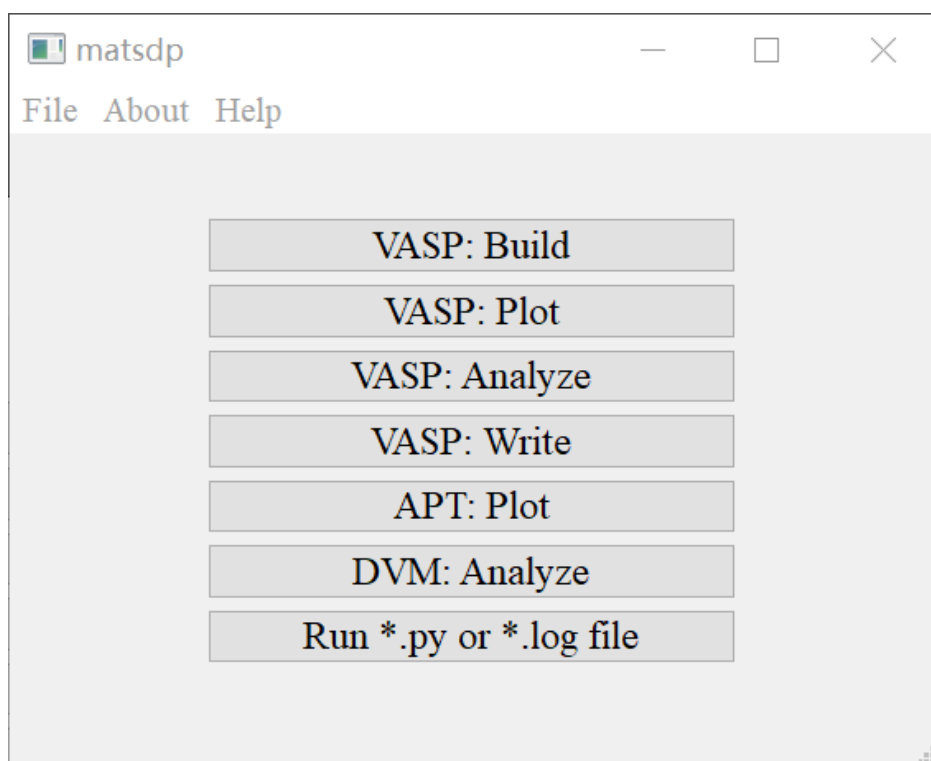


Figure 1.5: GUI for the main program

1.5 Notes

Note that for the module that requires POSCAR/CONTCAR, OUTCAR and DOSCAR files, these files need to be in the same folder.

The following sections will introduce the settings of the parameters in the GUI.

Chapter 2

Subpackage: vasp

Modules that may be imported before using the vasp package

- `from matsdp.vasp import vasp_read`
- `from matsdp.vasp import vasp_build`
- `from matsdp.vasp import vasp_plot`
- `from matsdp.vasp import vasp_analyze`
- `from matsdp.vasp import vasp_write`
- `from matsdp.vasp import vasp_tools`
- `from matsdp.vasp import vasp_default`
- `from matsdp.vasp import vasp_help`

2.1 vasp_build module

2.1.1 vasp_build.substitution

Descriptions

Building models by substitution of atoms

Syntax

```
from matsdp.vasp import vasp_build
vasp_build.substitution(
    substitution_list_file = './example/vasp/example/
    vasp.subst',
    poscar_file_path = './example/vasp/POSCAR.NoDope',
)
```

Arguments

- `substitution_list_file`: String format. It specifies the directory of the `.subst` file (substitution list file)
- `poscar_file_path`: String format. The directory of the POSCAR file which is to be substituted. It can either be full path or relative path.

`.subst` file

Descriptions

- The `.subst` file (substitution list file) is required and should consists of system entries.
- A system corresponds to a specific model configuration.
- System entries specifies how atoms are substituted in different systems.
- A system entry is a block of successive lines without line breaks.
- Each system entries must be separated by blank lines.

File formats. A typical system entry has the following format:

```
n_subst
element_name_to_be_substituted new_element_name
element_name_to_be_substituted new_element_name
...
(n_subst lines of element_name_to_be_substituted
element_subindx new_element_name)
```


where, `element_name_to_be_substituted` is the name of the element which is to be substituted. `new_element_name` is the name of the new element which take the place of the substituted atom. If `new_element_name = Va`, then a vacancy is added. As shown above, each system should start with a line which specifies a number: `n_subst`. `n_subst` is the number of atoms to be substituted in the system. Then each of the following `n_subst` lines specifies the element(s) to be substituted and the element(s) which take its/their place(s).

A specific example `.subst` file is as follow:

```
1
Ni244 W

2
Ni244 Re
Al12 Re

...
```

GUI

Outputs

Outputs: The final system name is `L(line_number)_composition_D(duplicate)`

2.1.2 vasp_build.selection_sphere

Descriptions

Building models by selection of atoms. The atoms within a sphere will be selected.

Syntax

```
from matsdp.vasp import vasp_build
vasp_build.selection_sphere(
    poscar_file_path = './tests/vasp/CONTCAR',
    origin_atom_name = 'Re1',
```

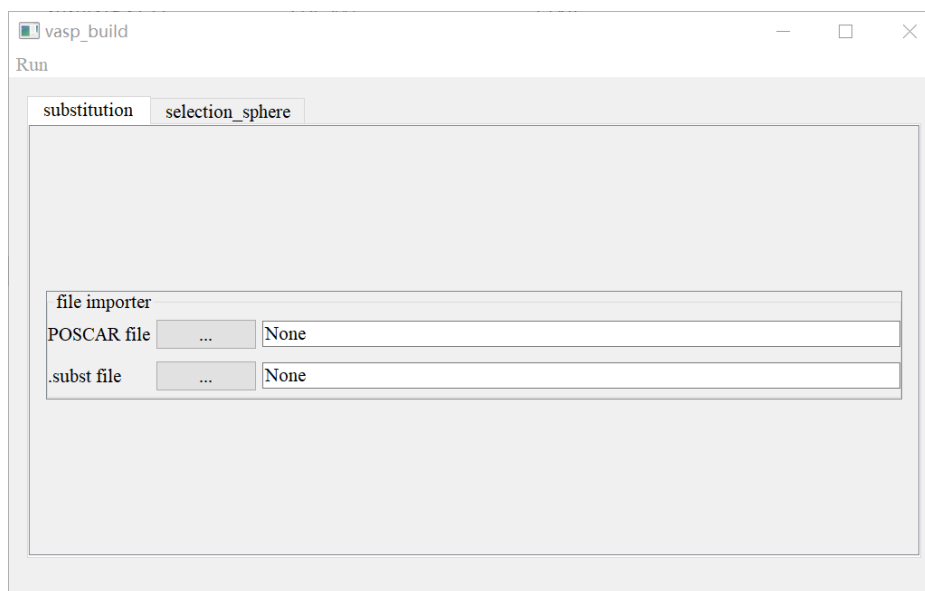


Figure 2.1: GUI for Substitution

```
radius = 7,  
include_mirror_atoms = False,  
output_file_name = 'example'  
)
```

Arguments

- `poscar_file_path`: String format. The directory of the POSCAR file. It can either be full path or relative path.
- `origin_atom_name`: String type. It defines the origin atom of the sphere
- `radius`: Float type. The atoms within a distance “radius” from the original atom are selected (units in Angstroms);
- `include_mirror_atoms`: Logical value. Whether to include the mirror atoms or not;
- `output_file_name`: user-defined output file name.

GUI

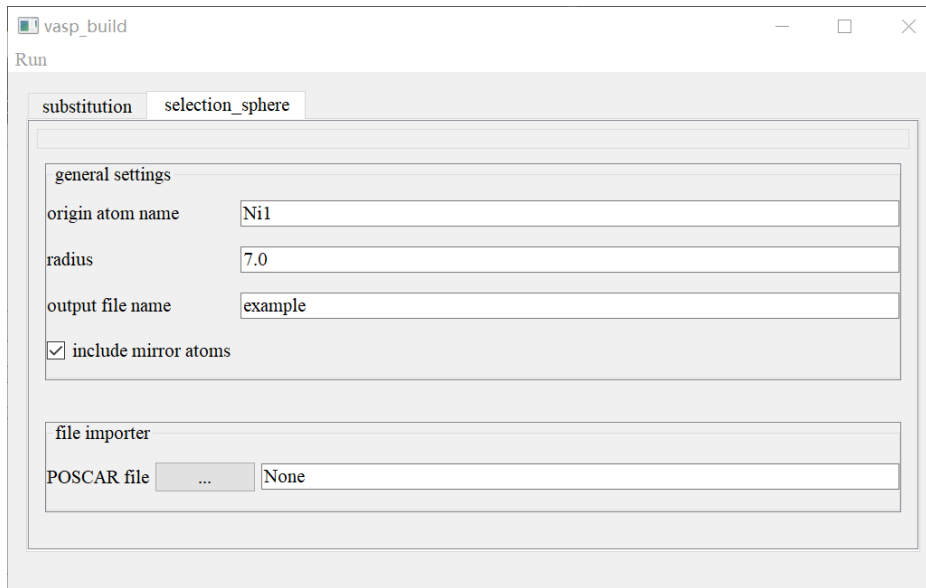


Figure 2.2: GUI for selection_sphere

Outputs

Outputs: *.vasp, *.xyz, and *.incarc files. The *.incarc file can be used as the input file for the DVM program.

2.1.3 vasp_build.transform

Descriptions

Transform by rotation and transformation operations.

Syntax

```
from matsdp.vasp import vasp_build
vasp_build.transform(
    poscar_file_path = './tests/vasp/POSCAR',
```

```
r_matrix_arr = None,  
t_matrix_arr = None,  
apply_to = 'atom',  
output_poscar_file_path = None  
)
```

Arguments

- `r_matrix_arr`: Rotation matrix
- `t_matrix_arr`: translation matrix
- `apply_to`: 'box' or 'atom'

Outputs

Outputs: POSCAR file after transformation.

2.1.4 `vasp_build.create_supercell`

Descriptions

Create supercell by expanding the cell in the direction of a, b, and c basis vectors.

Syntax

```
from matsdp.vasp import vasp_build  
vasp_build.create_supercell(  
    poscar_file_path = './tests/vasp/POSCAR',  
    num_xyz_list = [1,1,1],  
    output_poscar_file_path = None,  
)
```

Arguments

- `num_xyz_list` = [`n_x`, `n_y`, `n_z`], $n_x \geq 1$, $n_y \geq 1$, $n_z \geq 1$

Outputs

Outputs: POSCAR file after creating the supercell.

2.1.5 vasp_build.build_vacuum_layer

Descriptions

Build vacuum layer. The vacuum_layer_width is defined as the difference between the maximum and minimum projected(onto x, y, or z axis) atom coordinates.

Syntax

```
from matsdp.vasp import vasp_build
vasp_build.build\_vacuum\_layer(
    poscar_file_path = './tests/vasp/POSCAR',
    vacuum_layer_direction = 'z',
    vacuum_layer_width = 20,
    threshold_vacuum_layer_width = 5.98,
    output_poscar_file_path = None
)
```

Arguments

- vacuum_layer_direction: Designate the direction to add the vacuum layer
- threshold_vacuum_layer_width: This is the threshold distance for the vacuum layer separation, if layer separation is larger than this value, then it is denoted as vacuum layer. Otherwise, it is not a vacuum layer. Default value of threshold_vacuum_layer_width is $(1.15 * 2 * (\text{CSD_covalent_radius of Fr})) = 1.15 * 2 * 2.60 = 5.98$ Angstrom

Outputs

Outputs: POSCAR file after specifying the vacuum layer.

2.1.6 vasp_build.exfoliation

Descriptions

Exfoliate 2d material. Useful reference for determining the atom bonding status is the CSD covalent radius[1].

Syntax

```
from matsdp.vasp import vasp_build
vasp_build.exfoliation(
    poscar_file_path = './tests/vasp/POSCAR',
    num_layers = 1,
    vacuum_layer_width = 20,
    exfoliation_direction = 'z',
    layer_dist_tolerance = 0.664,
    criteria_list = ['bonding', 'periodicity'],
    align_center = False,
    align_upright = True,
    output_poscar_file_path = None
)
```

Arguments

- num_layers: the number of layers left after exfoliation.
- layer_dist_tolerance: This is the spacial resolution for the layer distance, for example $z=0.24$ and $z=0.25$ are considered to be in the same position if `layer_dist_tolerance=0.01`. Default value of `layer_dist_tolerance` is $(1.15 * 2 * (\text{CSD_covalent_radius of He})) + \text{arbitrary_delta} = 1.15 * 2 * 0.28 + 0.02 = 0.664$ Angstrom
- align_center: Align the atoms to the center of the model along the exfoliation direction
- align_upright: Align the exfoliation direction axis as an upright one (i.e. not a tilted one)

- `criteria_list`: ['bonding'], ['periodicity'], or ['bonding', 'periodicity'], the criteria in the list will be adopted. For ['bonding', 'periodicity'], this means a combination of bonding and periodicity (first check bonding, then check periodicity) will be adopted. If bonding criterion worked, then periodicity criterion will be skipped; If bonding criterion failed, then periodicity criterion will be adopted.

Outputs

Outputs: exfoliated POSCAR file.

2.2 vasp_plot module

2.2.1 vasp_plot.plot_poscar

Descriptions

- Visualization of POSCAR model. Euler angles are used to rotate the view of the model.
- Viewer direction is in x direction. The original orientation: x direction is perpendicular to the paper, z direction is in the paper and point to upper direction
- Reference for Eulerian angles: Herbert Goldstein, Charles P. Poole Jr. and John L. Safko, Classical Mechanics (3rd Edition). Goldstein Poole & Safko, 2001.
- This module can also show the atom properties by color mapping. The POSCAR file with additional data columns used to save the data of the atom properties.

Syntax

```
from matsdp.vasp import vasp_plot
vasp_plot.plot_poscar(
    poscar_file_path = './example/vasp/POSCAR',
    euler_angle_type = 'zyz',
```

```

phi = -3,
theta = 4,
psi = 0,
elmt_color = {'Ni': 'red', 'Re': 'blue'},
draw_mirror_atom = True,
box_on = True,
axis_indicator = True,
plot_cell_basis_vector_label = True,
plot_atom_label = None,
fig_format = 'png',
fig_dpi = 100,
draw_colormap = False,
colormap_column_indx = 1,
colormap_vmin = None,
colormap_vmax = None,
vmin_color = 'blue',
vmax_color = 'red',
colorbar_alignment = 'vertical'
)

```

Arguments

- `poscar_file_path`: String format. Directory of the POSCAR file which you want to plot
- `euler_angle_type`: string of length 3. It specify the type of rotations based on Eulerian angles. Choices are 'zyz', 'zxz', 'zyx', etc.. Usually the 'zyz' type is used.

'zyz' : proper Euler angle, y-convention. Perform consecutive rotations at axes counter-clockwisely. z-y-z rotation. First rotate the z axes of atoms by an angle phi, then rotate the intermediate y axis of atoms by an angle theta, finally rotate the final z axis of atoms by an angle psi

'zxz' : proper Euler angle, x-convention. Perform consecutive rotations at axes counter-clockwisely. z-x-z rotation. First rotate the z axes of atoms by an angle phi, then rotate the intermediate x axis of

atoms by an angle theta, finally rotate the final z axis of atoms by an angle psi

'zyx' : Tait-Bryan angles. z-y-x rotation. Perform consecutive rotations at axes counter-clockwisely. z-y-x rotation. First rotate the z axes of atoms by an angle phi, then rotate the intermediate y axis of atoms by an angle theta, finally rotate the final x axis of atoms by an angle psi

- phi, theta, psi: float formats. The first, second, and third rotation Eulerian angles, units in degrees.
- elmt_color: dictionary formats. this dictionary specifies the color for each element. For example `elmt_color = {'Ni':'black','Al':'magenta'}`
- draw_mirror_atom: Logical value. Whether to plot the mirror atoms at the periodic boundary
- box_on: Logical value. Whether to plot the box or not
- axis_indicator: Logic value. Whether to plot the axis indicator
- plot_cell_basis_vector_label: Logical value. Whether to plot the cell basis vector labels(i.e., to label the three basis vectors of the cell as a, b, and c)
- plot_atom_label: String value. values: "atom_name", "atom_index", "atom_species", "fix_info", "position_direct", "position_cartesian", "added_atom_data" or None/"None". It plots atom label to each atom
- fig_format: String format. fig_format is a string that defines output figure format. Supported fig_format: 'png', 'eps', 'pdf', 'tif', 'tiff', 'jpg', 'jpeg', 'svg', 'svgz', 'pgf', 'ps', 'raw', 'rgba'
- fig_dpi: float format. The DPI for non-vector graphics.
- draw_colormap: Logical value. If true, the color mapping of atom properties will be Performed. Default: False.
- colormap_column_idx: Integer value. Define which column of the atom property columns will be color mapped. Default: 1.

- `colormap_vmin`: Float value. Define the minimum value of the color map. If `colormap_vmin=None`, the minimum value of the original data will be used. Default: `None`.
- `colormap_vmax`: Float value. Define the maximum value of the color map. If `colormap_vmax=None`, the maximum value of the original data will be used. Default: `None`.
- `vmin_color = 'blue'`: String type. Define the color for the smallest value of the atom properties in the color map. Default: `'blue'`.
- `vmax_color = 'red'`: String type. Define the color for the largest value of the atom properties in the color map. Default: `'red'`.
- `colorbar_alignment`: String type. Defines the alignment of the color bar in the figure of the color map. The value can be either `'vertical'` or `'horizontal'`. Default: `'vertical'`.

GUI

The GUI of the `plot_poscar` module is shown in the Figure 2.3

Outputs

Figures of POSCAR models.

Examples

The examples are shown in the Figure 2.4, Figure 2.5, Figure 2.6 and Figure 2.7.

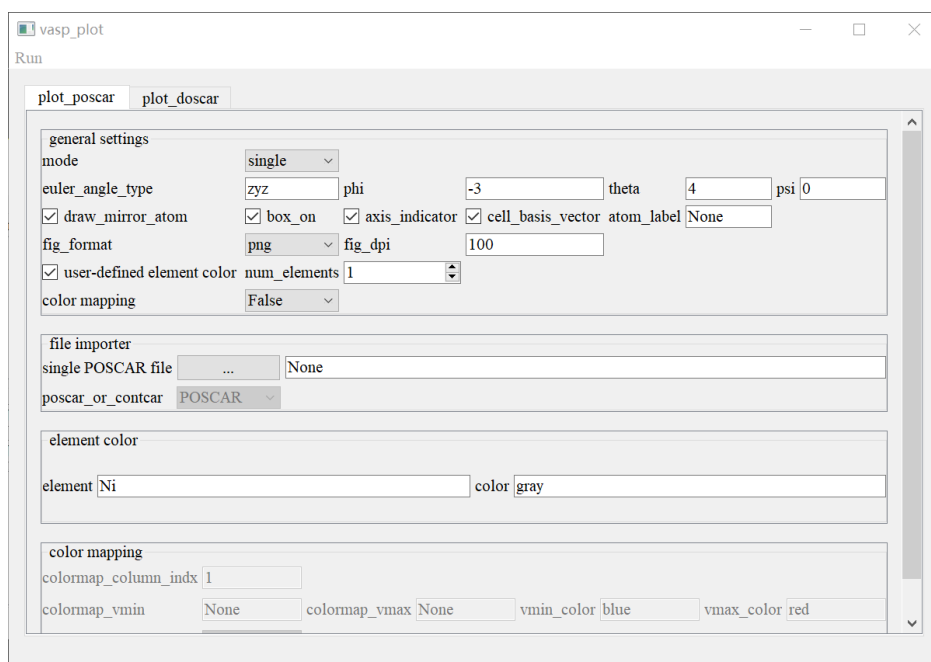


Figure 2.3: GUI for matsdp.vasp.vasp_plot.plot_poscar

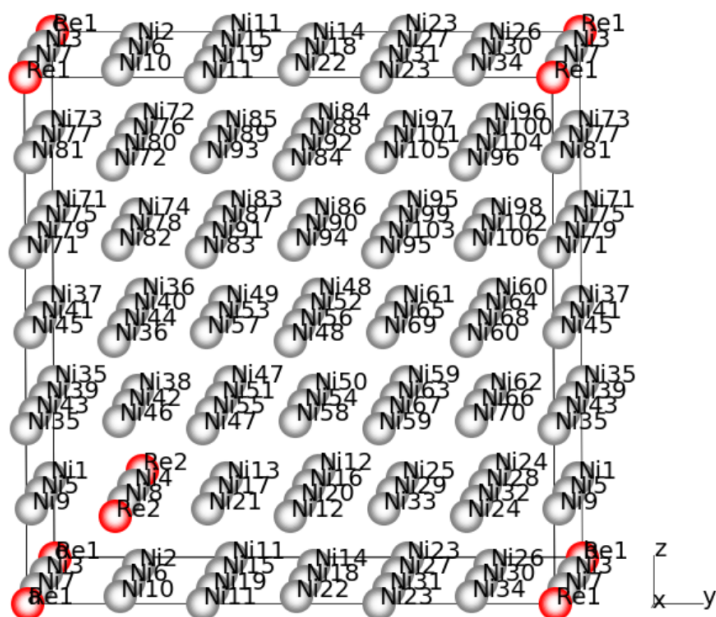


Figure 2.4: Result of the plot_poscar module. The atom label is added.

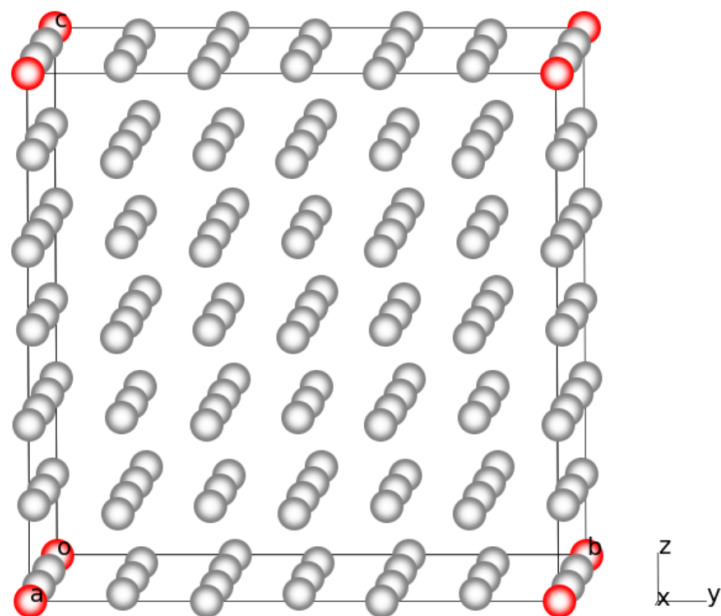


Figure 2.5: Result of the plot_poscar module. The atom label is removed.

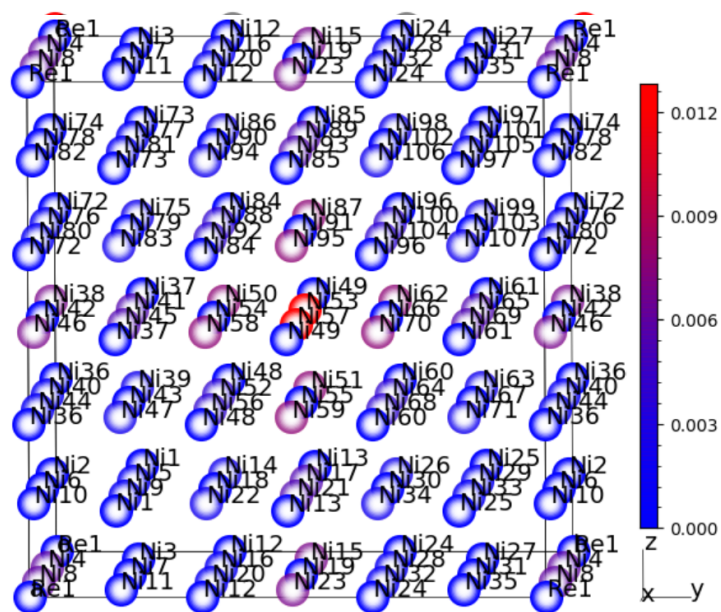


Figure 2.6: Result of the plot_poscar module: color mapping of atom properties. The color bar is vertically aligned.

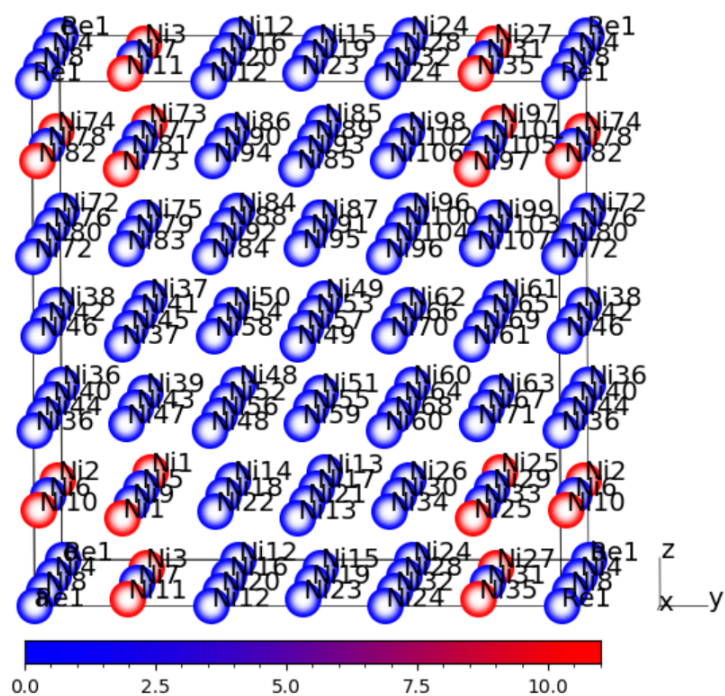


Figure 2.7: Result of the plot_poscar module: color mapping of atom properties. The color bar is horizontally aligned.

2.2.2 vasp_plot.plot_poscar_for_workdir

Descriptions

- Visualization of POSCARs.
- The mother folder needs to be specified which contains the folders with POSCARs
- Euler angles are used to rotate the view of the model
- This module can also show the atom properties by color mapping. The POSCAR file with additional data columns used to save the data of the atom properties.

Syntax

```
from matsdp.vasp import vasp_plot
vasp_plot.plot_poscar_for_workdir(
    workdir = './tests/vasp/',
    euler_angle_type = 'zyx',
    phi = -3,
    theta = 4,
    psi = 0,
    elmt_color = None,
    draw_mirror_atom = True,
    box_on = True,
    axis_indicator = True,
    plot_cell_basis_vector_label = True,
    plot_atom_label = None,
    poscar_or_contcar = 'POSCAR',
    fig_format = 'png',
    fig_dpi = 100,
    draw_colormap = False,
    colormap_column_indx = 1,
    colormap_vmin = None,
    colormap_vmax = None,
    vmin_color = 'blue',
    vmax_color = 'red',
```

```
colorbar_alignment = 'vertical'
)
```

Arguments

- **workdir**: String format. The mother folder which contains the folders with POSCARs
- **euler_angle_type**: string of length 3. It specifies the type of rotations based on Eulerian angles. Choices are 'zyz', 'zxz', 'zyx', etc.. Usually the 'zyz' type is used.
 'zyz' : proper Euler angle, y-convention. Perform consecutive rotations at axes counter-clockwisely. z-y-z rotation. First rotate the z axes of atoms by an angle phi, then rotate the intermediate y axis of atoms by an angle theta, finally rotate the final z axis of atoms by an angle psi
 'zxz' : proper Euler angle, x-convention. Perform consecutive rotations at axes counter-clockwisely. z-x-z rotation. First rotate the z axes of atoms by an angle phi, then rotate the intermediate x axis of atoms by an angle theta, finally rotate the final z axis of atoms by an angle psi
 'zyx' : Tait-Bryan angles. z-y-x rotation. Perform consecutive rotations at axes counter-clockwisely. z-y-x rotation. First rotate the z axes of atoms by an angle phi, then rotate the intermediate y axis of atoms by an angle theta, finally rotate the final x axis of atoms by an angle psi
- **phi, theta, psi**: float formats. The first, second, and third rotation Eulerian angles, units in degrees.
- **elmt_color**: dictionary formats. this dictionary specifies the color for each element. For example `elmt_color = {'Ni':'black','Al':'magenta'}`
- **draw_mirror_atom**: Logical value. Whether to plot the mirror atoms at the periodic boundary
- **box_on**: Logical value. Whether to plot the box or not

- `axis.indicator`: Logic value. Whether to plot the axis indicator
- `plot.cell.basis.vector.label`: Logical value. Whether to plot the cell basis vector labels(i.e., to label the three basis vectors of the cell as a, b, and c)
- `plot.atom.label`: Logical value. If true, then plot the atom name of each atom.
- `poscar_or_contcar`: String format. Determine whether to plot POSCAR or CONTCAR. Either 'POSCAR' or 'CONTCAR' can be used.
- `fig.format`: String format. `fig.format` is a string that defines output figure format. Supported `fig.format`: 'png', 'eps', 'pdf', 'tif', 'tiff', 'jpg', 'jpeg', 'svg', 'svgz', 'pgf', 'ps', 'raw', 'rgba'
- `fig.dpi`: float format. The DPI for non-vector graphics.
- `draw.colormap`: Logical value. If true, the color mapping of atom properties will be Performed. Default: False.
- `colormap.column.indx`: Integer value. Define which column of the atom property columns will be color mapped. Default: 1.
- `colormap.vmin`: Float value. Define the minimum value of the color map. If `colormap.vmin=None`, the minimum value of the original data will be used. Default: None.
- `colormap.vmax`: Float value. Define the maximum value of the color map. If `colormap.vmax=None`, the maximum value of the original data will be used. Default: None.
- `vmin.color = 'blue'`: String type. Define the color for the smallest value of the atom properties in the color map. Default: 'blue'.
- `vmax.color = 'red'`: String type. Define the color for the largest value of the atom properties in the color map. Default: 'red'.
- `colorbar.alignment`: String type. Defines the alignment of the color bar in the figure of the color map. The value can be either 'vertical' or 'horizontal'. Default: 'vertical'.

Outputs

Figures of POSCAR models.

2.2.3 vasp_plot.plot_dos

Descriptions

* Plot PDOS, LDOS, TDOS, now only available for LORBIT = 11. * There are three types of input arguments: atom related input arguments, subplot related input arguments, and others

Syntax

```
from matsdp.vasp import vasp_plot
dos1_file_path = './tests/vasp/DOSCAR'
vasp_plot.plot_dos(
    atom_doscar_file_path_list = [dos1_file_path],
    atom_sysname_list = ['C5'],
    atom_indx_list = ['Ni1'],
    atom_palette_list = ['black'],
    atom_subplot_arg_list = [111],
    subplot_arg_list = [111],
    subplot_xlo_list = [-6.5],
    subplot_xhi_list = [4.0],
    subplot_ylo_list = [None],
    subplot_yhi_list = [None],
    subplot_xtick_list = [True],
    subplot_ytick_list = [True],
    subplot_xlabel_list = [False],
    subplot_ylabel_list = [False],
    subplot_share_xy_list = [False, False],
    mainplot_axis_label_list = [True, True],
    xtick_direction = 'out',
    ytick_direction = 'out',
    dos_mode_dict = None,
    fermi_shift_zero = True,
    peak_analyzer = False,
```

```
    peak_analyzer_factor = 0.02,
    smoothing = False,
    smoothing_factor = 0.05,
    line_width = 2.0,
    font_size = 18,
    fig_format = 'png',
    fig_size = [13.0, 9.5],
    fig_dpi = 600,
)
vasp_plot.plot_dos(
    atom_doscar_file_path_list = [dos1_file_path,
    dos1_file_path],
    atom_sysname_list = ['C1', 'C1'],
    atom_indx_list = ['Ni1', 'Re1'],
    atom_palette_list = ['black', 'red'],
    atom_subplot_arg_list = [111, 111],
    subplot_arg_list = [111],
    subplot_xlo_list = [-6.5],
    subplot_xhi_list = [4.0],
    subplot_ylo_list = [None],
    subplot_yhi_list = [None],
    subplot_xtick_list = [True],
    subplot_ytick_list = [True],
    subplot_xlabel_list = [False],
    subplot_ylabel_list = [False],
    subplot_share_xy_list = [False, False],
    mainplot_axis_label_list = [True, True],
    xtick_direction = 'out',
    ytick_direction = 'out',
    dos_mode_dict = {'Ni': ['d'], 'Re': ['d']},
    fermi_shift_zero = True,
    peak_analyzer = False,
    peak_analyzer_factor = 0.02,
    smoothing = False,
    smoothing_factor = 0.05,
    line_width = 2.0,
    font_size = 18,
    fig_format = 'png',
```

```
fig_size = [11.0, 9.5],
fig_dpi = 600,
)
```

Arguments

Atom related Args

- `atom_doscar_file_path_list`: list format. Contains DOSCAR files for each atom. The directory of DOSCAR files can either be full path or relative path
- `atom_sysname_list`: system name for each atom, it corresponds to the atoms in the `atom_doscar_file_path_list`. This is for the purpose of labeling the DOS curves in the legend.

If `sysnameList = None`, then the label of system name will not shown in the legend

For example, `sysnameList = ['System1', 'System1', 'System2']`

- `atom_idx_list`: list format. Atom index list, it corresponding to the atoms in `atom_doscar_file_path_list`. If it is integer type then it denotes the atom index, if it is string type then it denotes the atom name

`atom_idx_list = [1,2,45]` denotes the 1st, 2nd, and the 45th atoms in the POSCAR

`atom_idx_list = ['Ni1','Al3','Re3']` denotes Ni1, Al3, and Re3 in the POSCAR

`atom_idx_list = ['TDOS']` and `atom_idx_list = [0]` denotes the total dos

- `atom_palette_list`: list format. Color for DOS curves of each atom.
- `atom_subplot_arg_list`: list format. Defines the DOS curves of the atom are in which subplot. For example, `atom_subplot_arg_list = [221, 222]` denotes that the DOS curves of the first and the second atoms are in the subplot(221) and subplot(222) subplots, respectively.

Subplot related Args

- `subplot_arg_list`: list format. The subplot argument list, for example `subplot_arg_list=[221,222]` corresponds to `subplot(221)` and `subplot(222)`
- `subplot_xlo_list`: list format. Low boundary of the x axis for each subplots. If None value is given, the low boundary of x axis in the data set will be chosen.
- `subplot_xhi_list`: list format. High boundary of the x axis for each subplots. If None value is given, the high boundary of x axis in the data set will be chosen.
- `subplot_ylo_list`: list format. Low boundary of the y axis for each subplots. If None value is given, the low boundary of y axis in the data set will be chosen.
- `subplot_yhi_list`: list format. High boundary of the y axis for each subplots. If None value is given, the high boundary of y axis in the data set will be chosen.
- `subplot_xtick_list`: list of logical values. If the list element is True (False), then the tick of the x axis will be shown (removed).
- `subplot_ytick_list`: list of logical values. If the list element is True (False), then the tick of the y axis will be shown (removed).
- `subplot_xlabel_list`: list of logical values. Defines whether the x-label of each subplots are shown, it won't work for `subplot=(111)` figure.
- `subplot_ylabel_list`: list of logical values. Defines whether the y-label of each subplots are shown, it won't work for `subplot=(111)` figure.
- `subplot_share_xy_list`: list of logical values of length two. Defines whether the x or y axis will be shared or not. [False, False] denotes both x and y axes will not be shared.

Other Args

- `mainplot_axis_label_list`: list of logical values of length two. Defines whether the x or y labels of the main figure will be shown or not. [False, False] denotes both x and y labels of the main figure will not be shown.

- `xtick_direction`: The direction of the x axis tick in the plot (pointing inward or pointing outward).
- `ytick_direction`: The direction of the y axis tick in the plot (pointing inward or pointing outward).
- `dos_mode_dict`: A dictionary that defines which partial DOS or whether LDOS is plotted for different element type. e.g.: `dos_mode_dict = {'Ni': ['s', 'p', 'd'], 'Al': ['s', 'p']}` or `dos_mode_dict = {'Ni': ['dxy', 'dx2']}`, or `dos_mode_dict = {'Ni': ['LDOS']}`.
- `fermi_shift_zero` is a logical value which determines whether to shift Fermi energy level to zero.
- `peak_analyzer`: logical value. Determines whether to analyze peaks in DOS. if True, the peaks will be labeled.
- `peak_analyzer_factor`: Float value. Determines the factor for peak analysis. The smaller this value, the more fine peaks can be found.
- `smoothing`: DOS curve smoothing (Lorentian broadening scheme)
- `smoothing_factor`: Float type. This defines the smoothing factor. In the case of the Lorentzian broadening scheme, the smoothing factor is the broadening width (units in eV).
- `line_width`: Float type. Line width. Recommended value 0.5 – 3.0 (from thin to fat)
- `font_size`: This value designate the font size for the axis label font size and the legend font size. Recommended value is 18
- `fig_format`: String format. Defines output figure format. Supported `fig_format`: 'png', 'eps', 'pdf', 'tif', 'tiff', 'jpg', 'jpeg', 'svg', 'svgz', 'pgf', 'ps', 'raw', 'rgba'
- `fig_size`: list of floats. Defines the size of the figure, e.g. `fig_size = (7.0, 6.0)`
- `fig_dpi`: float format. The DPI for non-vector graphics.

GUI

The GUI is shown in Figure 2.8. The panel can be divided into several control regions, which includes “DOSCAR” region, “Atom” region, “Subplot” region, “Element” region and “General settings” region. The settings for the plot_dos function is shown in Figure 2.9. The subplot layout is shown in Figure 2.10

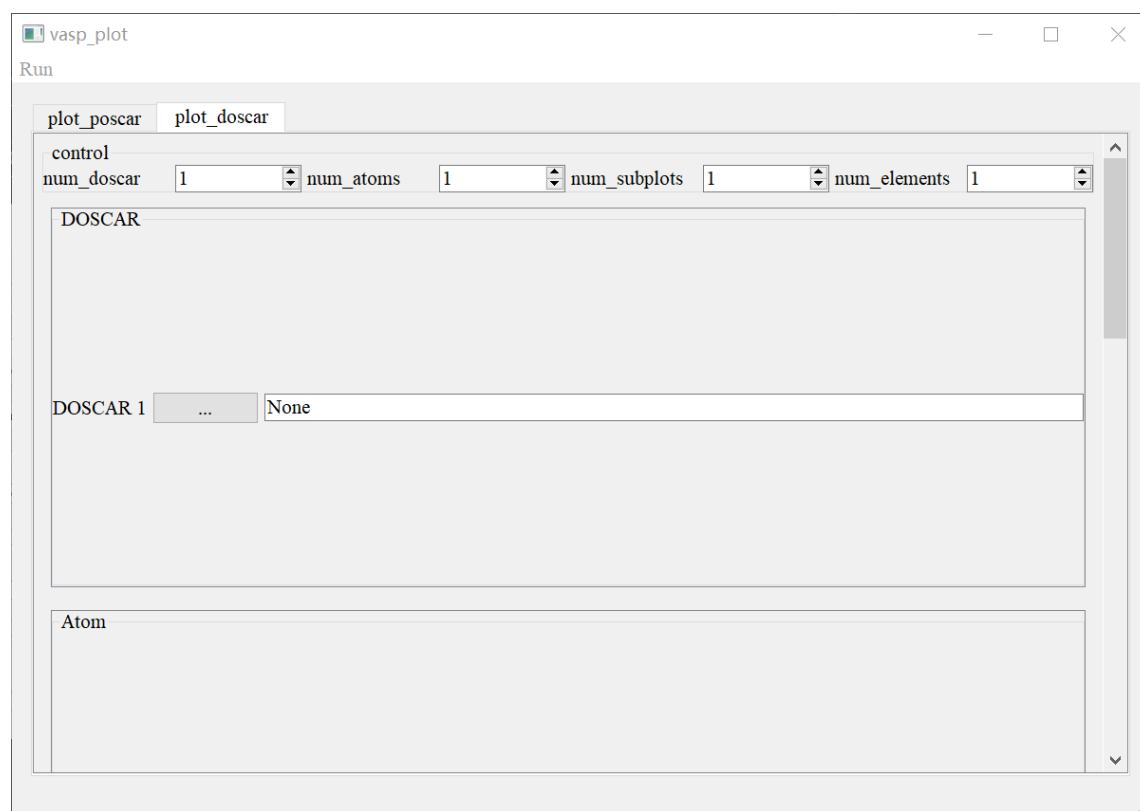
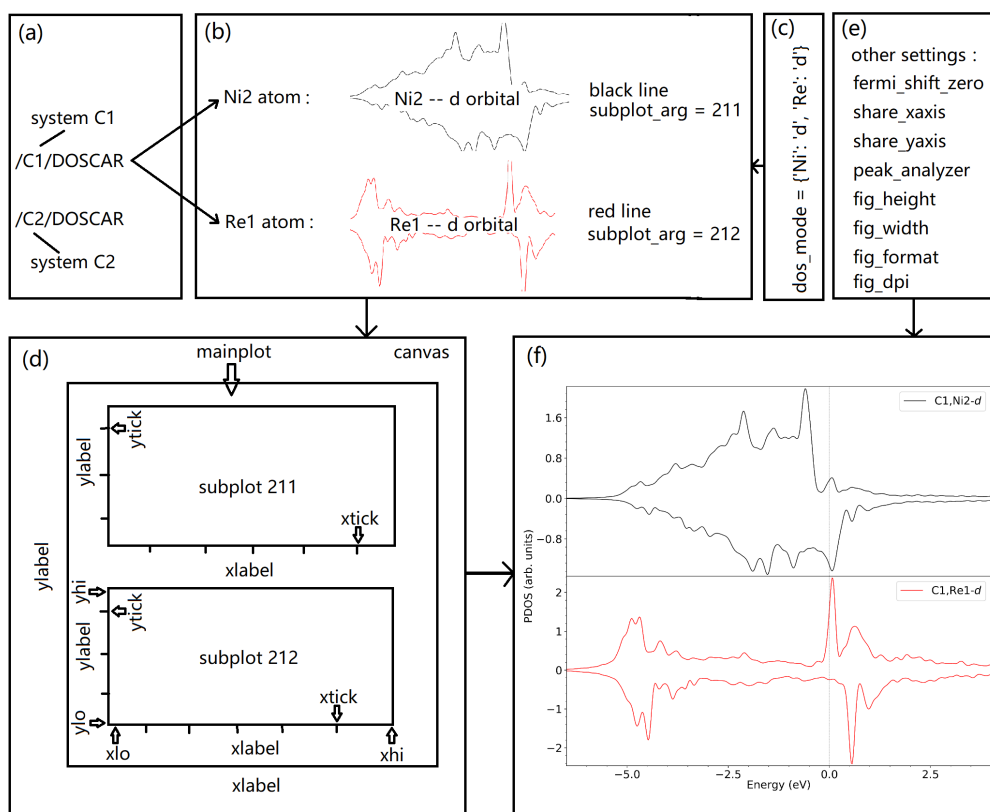


Figure 2.8: GUI for plot_dos

Some of the parameters are listed below:

- num_doscar: Number of DOSCAR files, this region can be used to import different DOSCAR
- num_atoms: Number of atoms for plotting the DOSs
- num_elements: Number of elements



(a) DOSCAR related settings; (b) atom related settings; (c) element related settings;
(d) subplot related settings; (e) other settings; (f) figure output

Figure 2.9: plot_dos settings

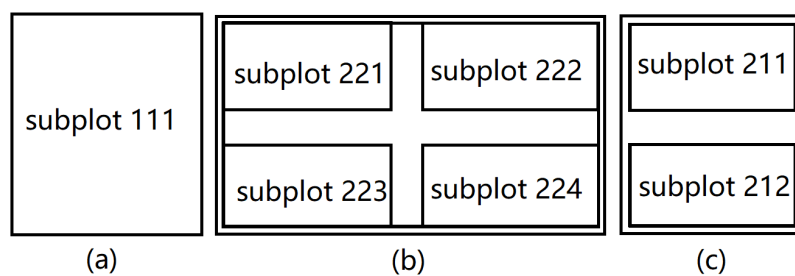


Figure 2.10: Subplot layout

- `num_subplots`: Number of subplots
- `subplot_arg`: The position of the subplot. The illustration of the subplot is shown in Fig. 2.10

If only one DOS curve will be plotted, then set `num_doscar=1` and `num_atom=1`. The value of `subplot_arg` then can be `subplot_arg=111`. For example, if the PDOSs of “Ni1” and “A2” are to be compared, the parameter `num_atom` should be taken as `num_atom=2`.

Output

Figures of DOS curves

Examples

The examples are shown in the Figure 2.11 and Figure 2.12.

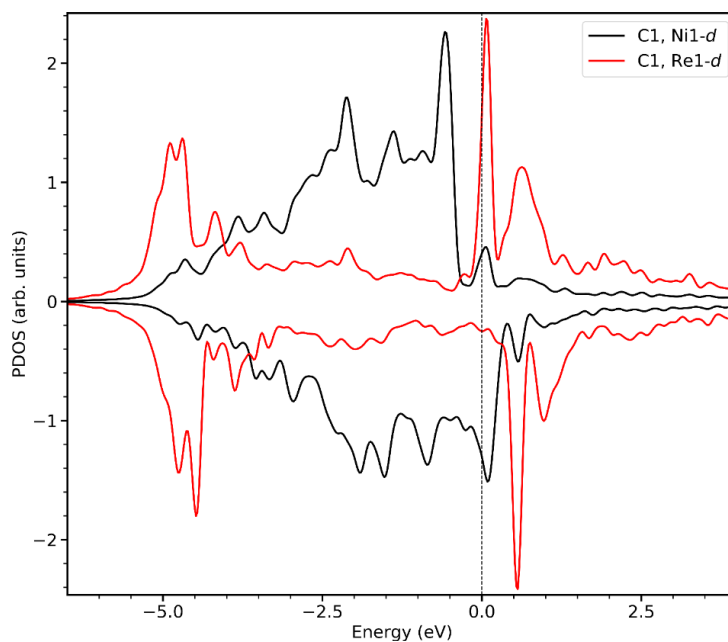


Figure 2.11: Result of the `plot_doscar` module.

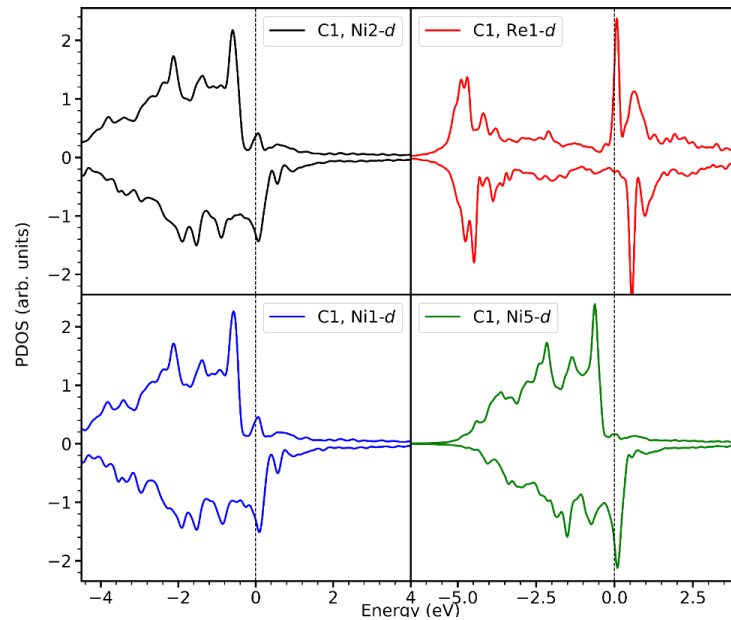


Figure 2.12: Result of the plot_doscar module.

2.2.4 vasp_plot.plot_band

Description

Plot band structure, including fat band.

Syntax

```
plot_bs(
    infile_path_list ,
    xlim = None,
    ylim = None,
    fermi_shift_zero = True,
    band_list = None,
    interp_on = True,
    show_band_data_point = False,
    band_gap_label = False,
    band_palette_dict = None,
    band_lty_dict = None,
```

```
system_color_list = None,  
system_lty_list = None,  
spd_and_site_projections_file_path_list = None,  
projections_point_size_factor = 1,  
legend_on = True,  
plot_fermi_level = False,  
xtick_direction = 'out',  
ytick_direction = 'out',  
line_width = 2.0,  
font_size = 23,  
fig_format = 'png',  
fig_size = [15,10],  
fig_dpi = 600)
```

Arguments

- `infile_path_list`: A list of input files. The input file can either be EIGENVAL or PROCAR.
- `xlim`: the limit of the x axis for the band structure plot.
- `ylim`: the limit of the y axis for the band structure plot.
- `fermi_shift_zero`: whether the energies are shifted to zero or not.
- `band_list`: Defines which bands are to be plotted. Band number starts from 1 (numbered from 1).
- `interp_on`: whether to interpolate the band data points or not.
- `show_band_data_point`: this is only valid when `interp_on = True`. if `show_band_data_point = True`, the raw data of the data points will be shown.
- `band_gap_label`: Logic value. If `band_gap_label = True`, then the band gap, CBM, CVM will be labeled.
- `band_palette_dict`: this is used to define the color of specific band. Dictionary key is the band index (numbered from 1).

- `band_lty_dict`: this is used to define the linestyle of specific band. Dictionary key is the band index (numbered from 1).
- `system_color_list`: if multiple infile are to be plotted. the label only label the system index. The line color of different systems, this is used when the band structure for different systems are to be compared.
- `system_lty_list`: if multiple infile are to be plotted. the band linestyles are designated for each infile. The line style of different systems, this is used when the band structure for different systems are to be compared.
- `spd_and_site_projections_file_path_list`: The parameter denotes the file which can be used to designate the spd- and site projected wave function character or each orbit. This parameter is only valid when the input file has the PROCAR file format. This file contains five columns: spin, mag, ion, orbit, color.

- spin: spin status of the projected contribution.
- mag: Magnetization_density. The total and local magnetizations, i.e. rho (orbital-projected contributions to the wavefunctions for each ion) and magnetization density (orbital-projected contributions to the magnetization) in the x(mx), y(my) and z(mz) directions. Please choose from the quadruplet of texts:

`'rho','mx','my','mz'`

or

`'tot','mx','my','mz'`

. The 'rho' represents 'tot'. The default is

`'rho'`

.

- ion: the ion name. for example, 'Ni1', 'Al3', 'Te2'.
- orbit: It contains one of the following orbits: 's', 'py', 'pz', 'px', 'dxy', 'dyz', 'dz2', 'dxz', 'dx2'(or 'dx2-y2'), 'tot'.
- color: the color for each plotted projected contribution.

One example of the `spd_and_site_projections_file` is as follows:

spin	,	mag	,	ions_list	,	orbit	,	color	,	legend
None	,	tot	,	['Li1',2]	,	s	,	black	,	None
None	,	tot	,	['tot']	,	px	,	red	,	Auto
None	,	tot	,	['Li']	,	py	,	pink	,	'Li'
None	,	tot	,	['Li3']	,	pz	,	blue	,	None
None	,	mx	,	[1,3]	,	py	,	green	,	Li1+Li3 \$p-{\$y}\$ \$m-{\$x}\$
...										

in which, the first line 'spin, mag, ion, orbit, color' is the header line. The parameter choices for the `spd_and_site_projections_file` is as follows (In "x / y", x is the parameter, y is the internal label of the parameters in the program.):

spin	mag	ions_list	orbit	color	legend
None / 0	None / 0	['tot'] / [4]	s / 0	black	None
up / 1	tot / 0	['Te2','Li5'] / [2, 7]	py / 1	red	Auto
dw / 2	mx / 1	['Se5'] / [4]	pz / 2	blue	'Li'
	my / 2	['Al3'] / [3]	px / 3	pink	'Li3_py_\$m_{x}\$'
	mz / 3	['Bi3'] / [12]	dxy / 4	green	
			dyz / 5		
			dz2 / 6		
			dxz / 7		
			dx2 / 8		
			tot / 9		

- ## Output

Examples

```
# -*- coding: utf-8 -*-
import os
import sys
package_path = '/user/specified/path/to/matsdp/'
sys.path.insert(0, os.path.abspath(package_path))

from matsdp.vasp import vasp_plot

vasp_plot.plot_bs(infile_path_list = ['./bs_soc/PROCAR'],
                  xlim = None,
                  ylim = [-4, 8.5],
                  interp_on = True,
                  spd_and_site_projections_file_path_list =
                      None,
                  legend_on = True,
                  plot_fermi_level = True,
                  fig_size = [13, 9],
                  band_gap_label = True,
                  )

vasp_plot.plot_bs(infile_path_list = ['./bs/PROCAR'],
                  xlim = None,
                  ylim = [-4, 8.5],
                  interp_on = False,
                  spd_and_site_projections_file_path_list =
                      ['./projections_li.txt'],
                  projections_point_size_factor = 0.7,
```

```

legend_on = True,
plot_fermi_level = True,
fig_size = [13, 9],
)

```

projections.li.txt:

spin	mag	ions_list	orbit	color	legend
up	tot	['Li']	tot	black	Auto
dw	tot	['Li']	tot	gray	Auto
up	tot	['Li']	px	red	Auto
dw	tot	['Li']	px	darkviolet	Auto
up	tot	['Li']	py	blue	Auto
dw	tot	['Li']	py	steelblue	Auto
up	tot	['Li']	pz	green	Auto
dw	tot	['Li']	pz	seagreen	Auto

The examples are shown in the Figure 2.13 and Figure 2.14.

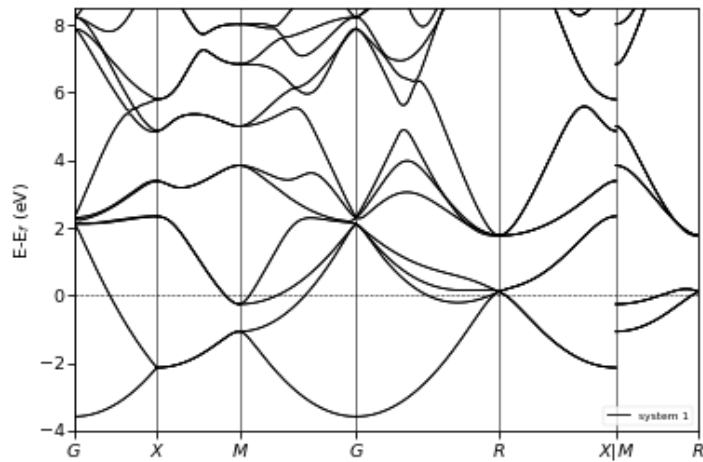


Figure 2.13: Result of the plot.bs module.

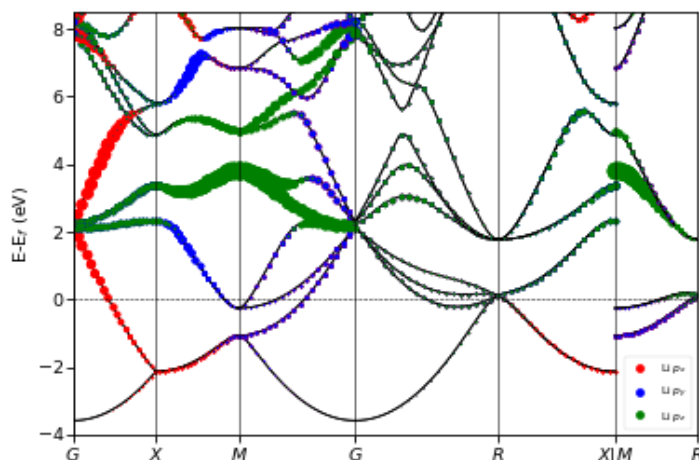


Figure 2.14: Result of the plot_bs module.

2.3 vasp_read module

2.3.1 vasp_read.read_doscar

Descriptions

Read DOSCAR and dump density of states information file into the folder where the DOSCAR lies

Syntax

```
from matsdp.vasp import vasp_read
vasp_read.read_doscar(
    doscar_file_path = './tests/vasp/DOSCAR',
    atom_indx = 2,
    save_dos_arr = True,
)
```


Arguments

- `doscar_file_path`: String format. The directory of the DOSCAR file. It can either be full path or relative path
- `atom_indx`: Integer format. The real atom index in the POSCAR. If there are N atoms then the atom indices are from 1 to N. Note that `atom_indx = 0` means to extract TDOS information
- `save_dos_arr`: logical format. If `save_dos_arr = True`, the density of states information will be saved to files. If `save_dos_arr = False`, the density of states information will not be saved to files

Outputs

DOS information file for the specified atom

2.4 vasp_analyze module

2.4.1 vasp_analyze.nn_map

Calculate the nearest neighbor (NN) map.

Descriptions

Calculate the nearest neighbor (NN) map.

Syntax

```
from matsdp.vasp import vasp_analyze
vasp_analyze.nn_map(
    poscar_file_path = './tests/vasp/POSCAR',
    a0 = 3.545,
    n_shell = 2,
)
```

Args

- `poscar_file_path`: String format. It specifies the directory of the POSCAR file
- `a0`: Float format. The lattice constant of the model. Unit in Angstrom
- `n_shell`: Integer format. It determines up to which crystallographic shell the nearest neighbour map calculates

GUI

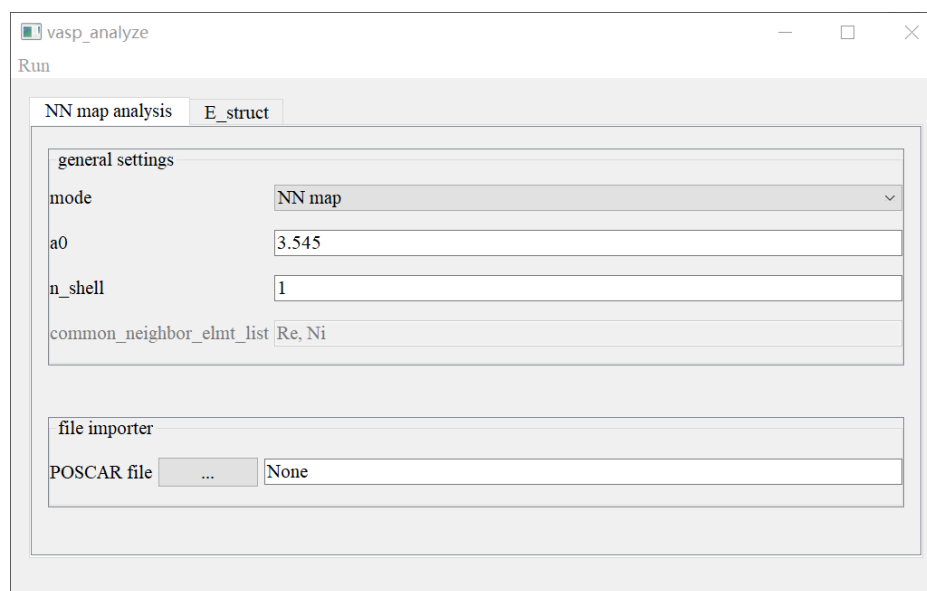


Figure 2.15: GUI for nn_map

2.4.2 vasp_analyze.simple_cna

Perform simple common neighbor analysis (CNA).

Descriptions

Perform simple common neighbor analysis (CNA). Atom A is the common neighbor of element E1 and E2, this module will count the times that A appeared as the common neighbor of E1 and E2.

Syntax

```
from matsdp.vasp import vasp_analyze
vasp_analyze.simple_cna(
    poscar_file_path = './tests/vasp/POSCAR',
    a0 = 3.545,
    common_neighbor_elmt_list = ['Re', 'W', 'Ta', 'Ni']
)
```

Args

- `poscar_file_path`: String format. It specifies the directory of the POSCAR file
- `a0`: Float format. The lattice constant of the model. Unit in Angstrom
- `common_neighbor_elmt_list`: List format. It determines what elements are taken into account in the common neighbor analysis. If `common_neighbor_elmt_list = ['Re', 'W', 'Ta']`, then the common neighbor to Re-Re, Re-W, Re-Ta, W-W, W-Ta, Ta-Ta will be counted and printed.

GUI

2.4.3 vasp_analyze.estruct

Descriptions

- Calculates the structural energies at each atomic site
- The definition of E_{struct} can be found in the literature of the author Chongyu Wang [3, 4]

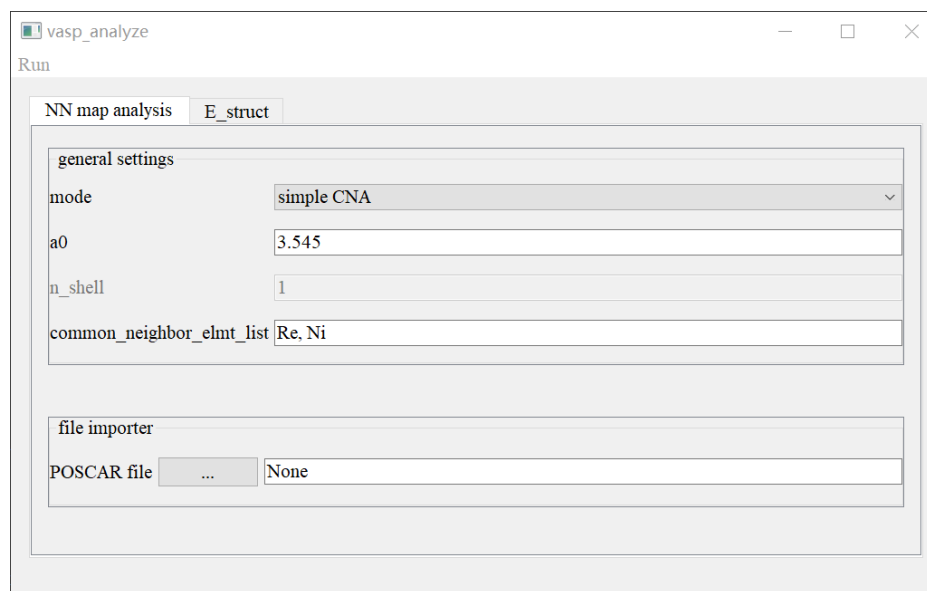


Figure 2.16: GUI for simple.cna

Syntax

```
from matsdp.vasp import vasp_analyze
vasp_analyze.estruct(
    doscar_file_path = './tests/vasp/DOSCAR',
    sysname = 'DOS1',
)
```

Arguments

- `dOScar_file_path`: String format. It specifies the directory of the DOSCAR
- `sysname`: String format. User defined system name

GUI

Output

The first column is the atom name, the second column is E_{struct} for each atom

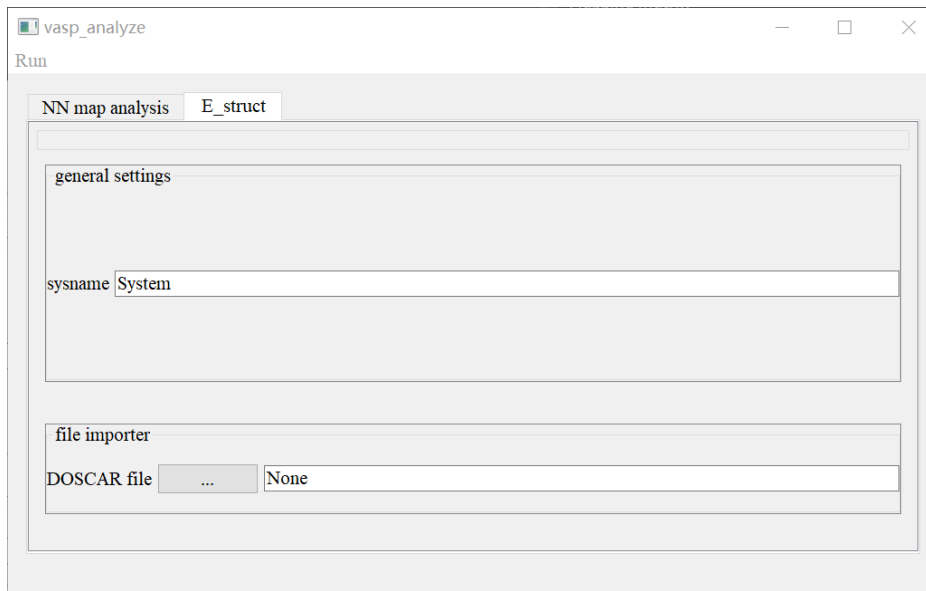


Figure 2.17: GUI for estruct

2.4.4 vasp_analyze.overlap_peak_analyzer

Descriptions

- Finding the overlapped orbitals of two neighboring atoms in the DOS analysis.
- DOS peak analyses for selected atoms with their neighboring atoms.
- Find the overlapped orbitals and their corresponding energy levels.

Syntax

```
from matsdp.vasp import vasp_analyze
vasp_analyze.overlap_peak_analyzer(
    doscar_file_path = './tests/vasp/DOSCAR',
    sysname = 'DOS1',
    atom_indx_list = ['Ni1', 'Re1'],
    n_shell = 2,
    a0 = 3.52,
```

```
dos_mode_dict = {'Ni': ['d'], 'Re': ['d']},  
fermi_shift_zero = True,  
)
```

Arguments

- `doscar_file_path`: String format. The directory which contains the DOSCAR file, abstract path can be accepted
- `sysname`: String format. A string character which specifies the name of the system, this string will be used as part of the output file name
- `atom_indx_list`: List of strings. Specifies the list of selected atoms.
- `n_shell`: float format. Up to which crystallographic shell (up to which nearest neighbor) of the selected atom will be considered
- `a0`: float format. The approximate lattice constant of the crystal
- `dos_mode_dict`: dictionary format. Determines which orbital will be considered, f, d, p, s, dxy, dyz, ... can be used
- `fermi_shift_zero`: A logical value determining whether the energy levels of the DOS will be shifted to zero

Outputs

overlapped peak files

2.4.5 `vasp_analyze.job_status`

Check the job status for multiple jobs

Descriptions

Check the job status for multiple jobs

Syntax

```
from matsdp.vasp import vasp_analyze
vasp_analyze.job_status(
    job_parent_dir = './tests/vasp/'
)
```

Args

- `job_parent_dir`: This is the parent directory which contains the multiple VASP jobs.

2.4.6 Output

The `vasp_job_status.txt` file. This file tells us which job has been finished and which job has not been finished.

2.5 vasp_write module**2.5.1 vasp_write.write_poscar_with_force****Descriptions**

write atom force data into the POSCAR file.

Syntax

```
from matsdp.vasp import vasp_write
vasp_write.write_poscar_with_force(
    outcar_file_path = './tests/vasp/OUTCAR',
    ionic_step = 'last',
    output_poscar_file_name = None
)
```

Arguments

- `outcar_file_path`: String format. It specifies the directory of the OUTCAR
- `ionic_step`: String format or interger type. If string type value is taken, either `ionic_step='last'` or `ionic_step='first'` can be taken. If integer type value is taken, `ionic_step` defines the ionic step number. `ionic_step = 3` denotes that the force of each atom for the third ionic step will be written to the POSCAR file.
- `output_poscar_file_name`: String type or None. If string type is taken, this parameter lets the user define the POSCAR file name which contains the atom force information. If `output_poscar_file_name=None`, the program determines the name of the output POSCAR file.

GUI

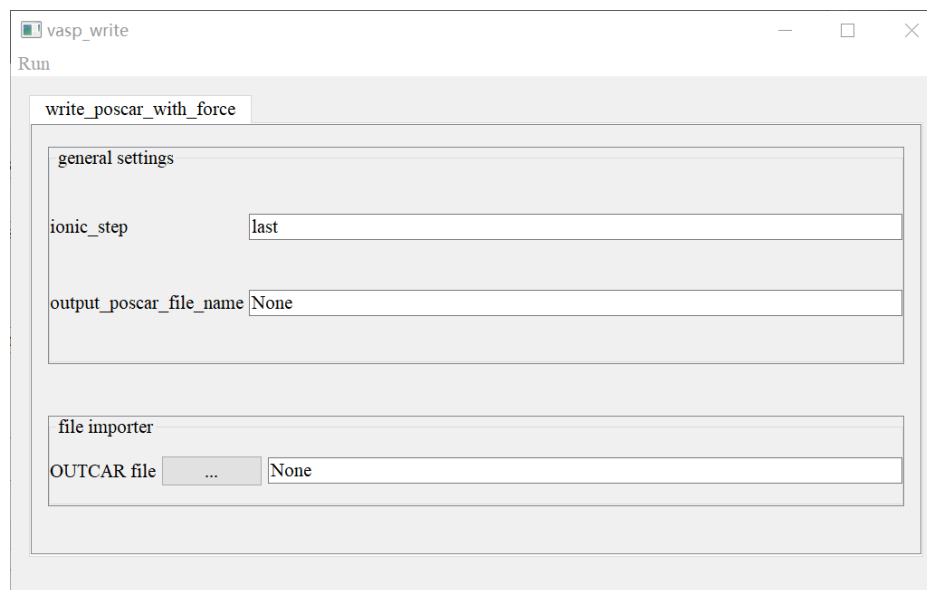


Figure 2.18: GUI for `write_poscar_with_force`

Output

The POSCAR file which contains the force on each atom.

Chapter 3

Subpackage: apt

Modules that may be imported before using the apt package

- `from matsdp.apt import apt_read`
- `from matsdp.apt import apt_plot`

3.1 apt_read module

3.1.1 apt_read.read_proxigram_csv

Descriptions

- Read the concentration profile file (*.csv file)

Syntax

```
from matsdp.apt import apt_read  
apt_read.read_proxigram_csv(proxigram_csv_file_path)
```

Arguments

- `proxigram_csv_file_path`: string type. The concentration profile file.

Outputs

- `data_set`: numpy array type. The original data of the concentration profile file
- `elmtname_list`: List type. The elements contained in the concentration profile file

3.2 apt_plot module

3.2.1 apt_plot.plot_proxigram_csv

Descriptions

- Plot the concentration profile based on the proxigram *.csv file

Syntax

```
from matsdp.apt import apt_plot
apt_plot.plot_proxigram_csv(
    proxigram_csv_file_path = './tests/apt/profile-
    interface0.csv',
    sysname = 'M2',
    visible_elmt_list = ['Ni', 'Al'],
    interpolation_on = False,
    fig_width = 6,
    fig_height = 5,
    fig_dpi = 600,
    fig_format = 'png',
)
```

Arguments

- `proxigram_csv_file_path`: string type. The concentration profile file.
- `sysname`: string type. The system name.

- `visible_elmt_list`: List type. The elements which are to be plotted. For example, ['Ni', 'Al'].
- `interpolation_on`: logical type. whether to interpolate the concentration profile or not.
- `fig_width`: float type. Figure width.
- `fig_height`: float type. Figure height.
- `fig_dpi`: float format. The DPI for non-vector graphics.
- `fig_format`: String format. `fig_format` is a string that defines output figure format. Supported `fig_format`: 'png', 'eps', 'pdf', 'tif', 'tiff', 'jpg', 'jpeg', 'svg', 'svgz', 'pgf', 'ps', 'raw', 'rgba'

GUI

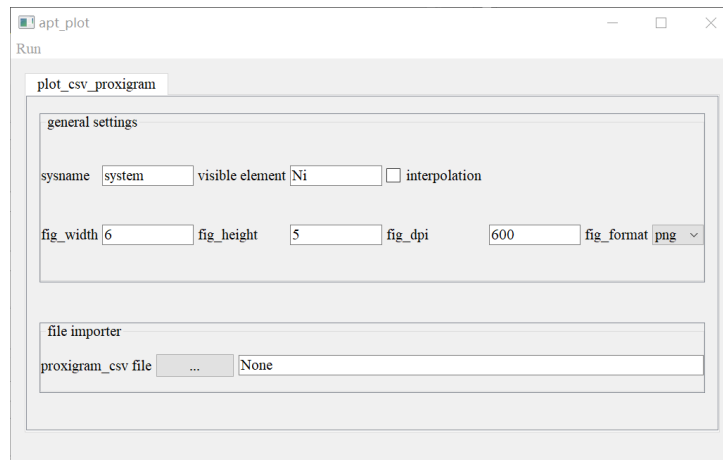


Figure 3.1: GUI for `plot_concentration_profile`

Examples

The example is shown in the Figure 2.11.

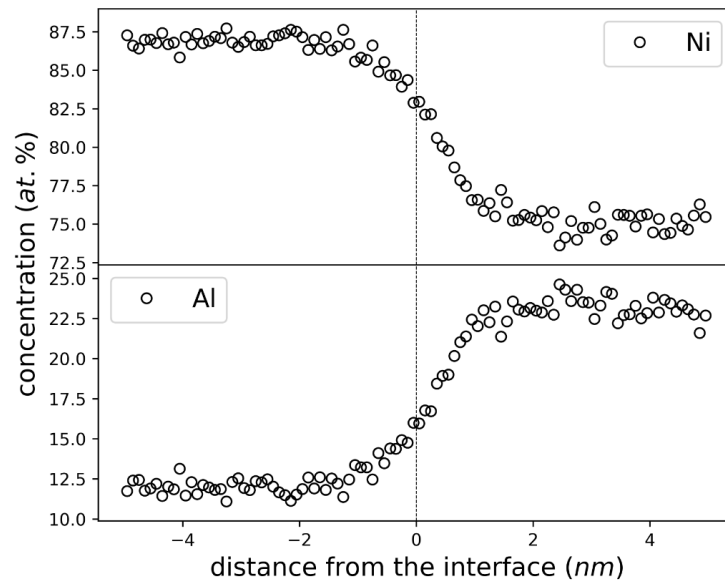


Figure 3.2: Result of the `plot_proxigram_csv` module.

Chapter 4

Subpackage: dvm

Modules that may be imported before using the dvm package

- `from matsdp.dvm import dvm_read`
- `from matsdp.dvm import dvm_build`
- `from matsdp.dvm import dvm_analyze`
- `from matsdp.dvm import dvm_write`
- `from matsdp.dvm import dvm_default`
- `from matsdp.dvm import dvm_help`

4.1 dvm_build module

4.1.1 create_multiple_dvm_jobs

Descriptions

create multiple DVM jobs (the *.incar, *.input, IND.DAT files will be created) based on atom selection (spherical) of the POSCAR files.

Syntax

```

from matsdp.dvm import dvm_build
poscar_file_path_dict = {}
origin_atom_name_list_dict = {}
poscar_file_path_dict['L0000926'] = './outputs/example/
L0000926-Ta7Re6Ni332Al47_D2/CONTCAR'
poscar_file_path_dict['L0000911'] = './outputs/example/
L0000911-Ta7Re6Ni332Al47_D1/CONTCAR'
poscar_file_path_dict['L0000941'] = './outputs/example/
L0000941-Ta7Re6Ni326Al53_D1/CONTCAR'
poscar_file_path_dict['L0000956'] = './outputs/
example/L0000956-Ta7Re6Ni326Al53_D2/CONTCAR'
origin_atom_name_list_dict['L0000926'] = ['Ta4', '
Ni124']
origin_atom_name_list_dict['L0000911'] = ['Re2', '
Ni124']
origin_atom_name_list_dict['L0000941'] = ['Re1', '
Ni124']
origin_atom_name_list_dict['L0000956'] = ['Ta1', '
Ni124']
elmt_ind_file_dir = './dvm_ind/'
radius = 11
dvm_build.create_multiple_dvm_jobs(
    poscar_file_path_dict = poscar_file_path_dict,
    origin_atom_name_list_dict =
    origin_atom_name_list_dict,
    elmt_ind_file_dir = elmt_ind_file_dir,
    radius = radius,
    include_mirror_atoms = True
)

```

Arguments

- `poscar_file_path_dict`: Dictionary type. A dictionary which contains the POSCAR file path, the key of the dictionary will be used as part of the DVM job name.
- `origin_atom_name_list_dict`: Dictionary type. A dictionary which con-

tains the list of origin atom names. The origin atoms in the center of the sphere in the atom selection (spherical) of the POSCAR. The key of the dictionary will be used as part of the DVM job name.

- `elmt_ind_file_dir`: the top directory which contains the IND.DAT files of the elements
- `radius`: the radius of the sphere in the atom selection (spherical) of the POSCAR
- `include_mirror_atoms`: whether to include the mirror atoms

Outputs

Outputs: The *.input, *.incar, IND.DAT, *.vasp files

4.2 dvm_read module

4.2.1 dvm_read.read_input

Descriptions

read the *.input file of the DVM program

Syntax

```
from matsdp.dvm import dvm_read
read_input(input_file_path)
```

Arguments

- `input_file_path`: the *.input file path

Outputs

a dictionary with input parameters

4.2.2 `dvm_read.read_ind`

Descriptions

read the *.ind file of the DVM program

Syntax

```
from matsdp.dvm import dvm_read  
read_ind(ind_file_path)
```

Arguments

- `ind_file_path`: the file path of the IND.DAT file

Outputs

a dictionary with IND.DAT parameters

4.2.3 `dvm_read.read_incar`

Descriptions

read the *.incard file of the DVM program

Syntax

```
from matsdp.dvm import dvm_read  
read_incar(incard_file_path)
```

Arguments

- `incard_file_path`: the *.incard file path

Outputs

a dictionary with *.incard parameters

4.2.4 dvm_read.read_output

Descriptions

read the *.output file of the DVM program

Syntax

```
from matsdp.dvm import dvm_read  
read_output(output_file_path)
```

Arguments

- output_file_path: the *.output file path

Outputs

a dictionary with *.output parameters

4.3 dvm_analyze module

4.3.1 dvm_analyze.ie_nn

extract interatomic energy between the atoms and their nearest neighbor atoms. This module has been tested for the source_23oct05 version of the DVM program

Descriptions

extract interatomic energy between the atoms and their nearest neighbor atoms. This module has been tested for the source_23oct05 version of the DVM program

Syntax

```
from matsdp.dvm import dvm_analyze
dvm_analyze.ie_nn(
    dvm_output_file_path = dvm_output_file_path ,
    a0 = 3.54
)
```

Args

- `dvm_output_file_path`: the *.otput file of the DVM output
- `a0`: Float format. The lattice constant of the model. Unit in Angstrom

GUI

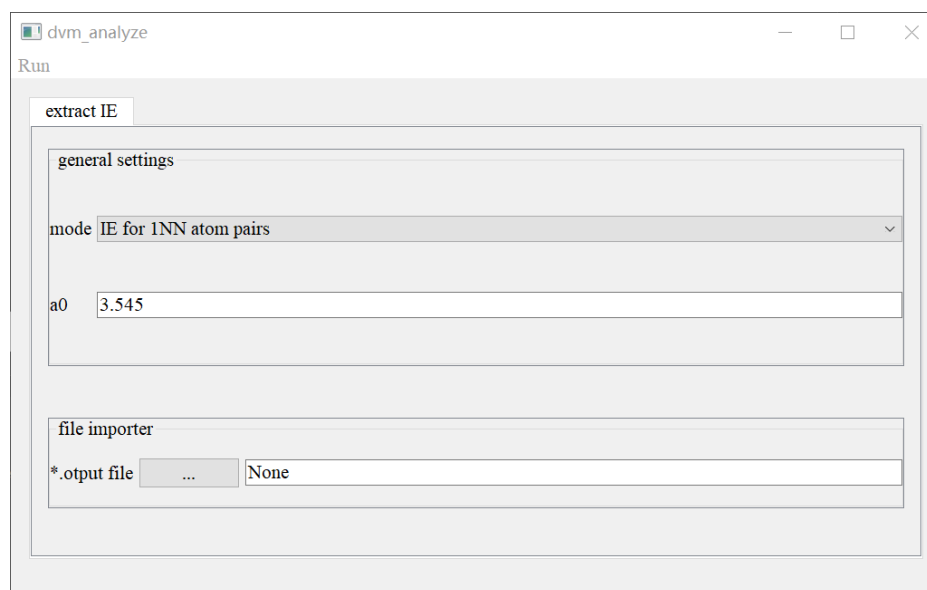


Figure 4.1: GUI for `ie_nn`

Output

The files which contains the information of interatomic energies between the first nearest neighbor atoms pairs.

4.3.2 dvm_analyze.job_status

Check the job status for multiple jobs

Descriptions

Check the job status for multiple jobs

Syntax

```
from matsdp.dvm import dvm_analyze
dvm_analyze.job_status(
    job_parent_dir = './tests/dvm/'
)
```

Args

- job_parent_dir: This is the parent directory which contains the multiple DVM jobs.

4.3.3 Output

The dvm_job_status.txt file. This file tells us which job has been finished and which job has not been finished.

4.4 dvm_write module

4.4.1 dvm_write.write_input

Descriptions

Write the *.input file for a DVM calculation based on the atom position file (*.incard or POSCAR format) Currently, for the pos_file_path, only the POSCAR format is supported. In the future the *.incard will also be supported and the automatic file format recognition should be used.

Syntax

```
from matsdp.dvm import dvm_write
dvm_write.write_input(pos_file_path , dvm_input_dict =
None)
```

Arguments

- `pos_file_path`: the file path for the file which contains the atom coordinates, now the POSCAR file is supported.
- `dvm_input_dict`: the dictionary which contains the input parameters of the *.input file. If `dvm_input_dict = None`, then the default value of the parameters in the *.input file will be used

Output

The *.input file.

4.4.2 dvm_write.write_ind

Descriptions

write the IND.DAT file for a DVM calculation

Syntax

```
from matsdp.dvm import dvm_write
dvm_write.write_ind(
    pos_file_path = pos_file_path ,
    elmt_ind_file_dir = './dvm\_ind/')
```

Arguments

- `pos_file_path`: the file path for the file which contains the atom coordinates, now the POSCAR file is supported.

- `elmt_ind_file_dir`: the directory which contains the element IND.DAT files. The element IND.DAT files if the file with IND.DAT information of each element

An example of the typical element IND.DAT file is shown below:

```
0 0 Al ATOM 13 1S2-2S2-2P6-3S2-3P1
300 5 35.00000000 30.00000000 13.00000000
0.00000000 0.000000
0.30000000 0.30000000 2500.00 0.00000001
0.00000001
4.00 -2.0 6.0 0.0
0 150 0 0 0 0
-0.70000000 0.00000000
1.0 0.0 0.0 0.00000 2.00000 0.00000
2.0 0.0 0.0 0.00000 2.00000 0.00000
2.0 1.0 0.0 0.00000 6.00000 0.00000
3.0 0.0 0.0 0.00000 2.00000 0.00000
3.0 1.0 0.0 0.00000 1.00000 0.00000
```

If the element IND.DAT file is named as `IND_Al.DAT` and its path is `./dvm_ind/IND_Al.DAT`, then `elmt_ind_file_dir = './dvm_ind/'`

Output

The IND.DAT file.

4.4.3 dvm_write.write_ie

Descriptions

extract interatomic energy between all the first nearest neighbor atom pairs. This module has been tested for the source_23oct05 version of the DVM program

Syntax

```
from matsdp.dvm import dvm_write
dvm_write.write_ie(dvm_output_file_path)
```

Arguments

- `dvm_output_file_path`: the *.otput file of the DVM output

GUI

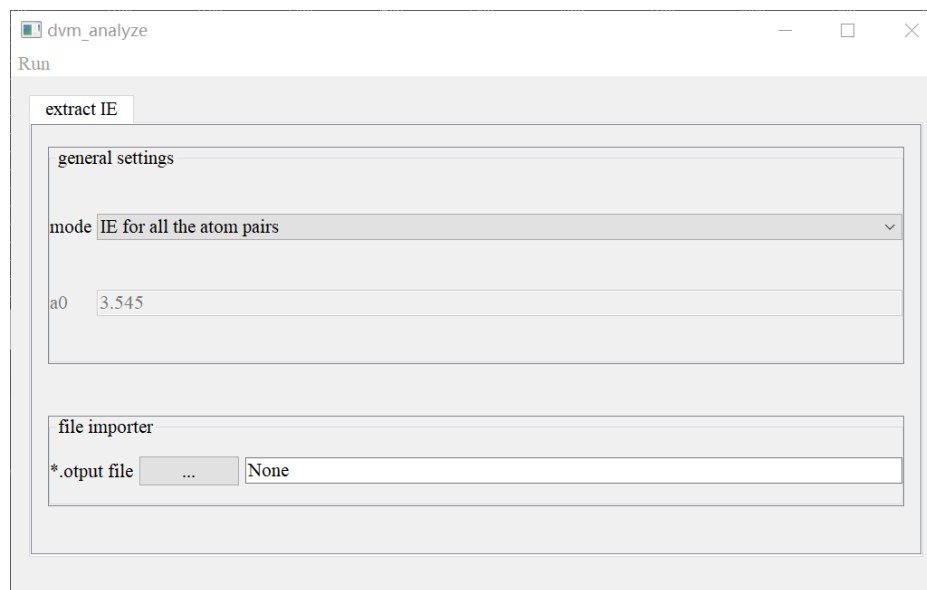


Figure 4.2: GUI for write_ie

Output

The files which contains the information of interatomic energies between the atoms pairs.

Chapter 5

Subpackage: pms

Modules that may be imported before using the pms package

- `from matsdp.pms import project_manager`
- `from matsdp.pms import task_manager`

5.1 task_manager module

5.1.1 task_manager.write_task_summary

Functions

Write task summary for the jobs in a specific directory

Syntax

```
write_task_summary(  
    task_dir ,  
    task_type = 'VASP' ,  
    band_gap_label = True ,  
    plot_fatband = False ,  
    band_struct_ylim = None ,  
    num_added_bands = 3)
```

Examples

```
from matsdp.pms import task_manager  
task_manager.write_task_summary( './ outputs/example/' )
```

Chapter 6

Other functions

6.1 funcs.py

6.1.1 cp()

```
from matsdp import funcs
funcs.cp(src_filedir, dst_filedir)
```

6.1.2 write_file()

```
from matsdp import funcs
funcs.write_file(input_str, dest_file_path)
```

6.1.3 merge_files

```
from matsdp import funcs
funcs.merge_files(file1, file2)
```

6.1.4 dir_tree()

```
from matsdp import funcs
funcs.dir_tree('./outputs/')
```


Chapter 7

Tests

The test files are in the “matsdp/tests/” folder. The runtests.py file can be used to automatically run multiple tests.

Appendix A

Other plotting settings

A.1 Named colors in the program

The named colors which can be used by the program is listed in Figure A.1¹.

¹https://matplotlib.org/3.1.0/gallery/color/named_colors.html

CSS Colors

black	bisque	forestgreen	slategrey
dimgray	darkorange	limegreen	lightsteelblue
dimgray	burlywood	darkgreen	cornflowerblue
gray	antiquewhite	green	royalblue
grey	tan	lime	ghostwhite
darkgray	navajowhite	seagreen	lavender
darkgrey	blanchedalmond	mediumseagreen	midnightblue
silver	papayawhip	springgreen	navy
lightgray	moccasin	mintcream	darkblue
lightgrey	orange	mediumspringgreen	mediumblue
gainsboro	wheat	mediumaquamarine	blue
whitesmoke	oldlace	aquamarine	slateblue
white	floralwhite	turquoise	darkslateblue
snow	darkgoldenrod	lightseagreen	mediumslateblue
rosybrown	goldenrod	mediumturquoise	mediumpurple
lightcoral	cornsilk	azure	rebeccapurple
indianred	gold	lightcyan	blueviolet
brown	lemonchiffon	paleturquoise	indigo
firebrick	khaki	darkslategray	darkorchid
maroon	palegoldenrod	darkslategrey	darkviolet
darkred	darkkhaki	teal	mediumorchid
red	ivory	darkcyan	thistle
mistyrose	beige	aqua	plum
salmon	lightyellow	cyan	violet
tomato	lightgoldenrodyellow	darkturquoise	purple
darksalmon	olive	cadetblue	darkmagenta
coral	yellow	powderblue	fuchsia
orangered	olivedrab	lightblue	magenta
lightsalmon	yellowgreen	deepskyblue	orchid
sienna	darkolivegreen	skyblue	mediumvioletred
seashell	greenyellow	lightskyblue	deeppink
chocolate	chartreuse	steelblue	hotpink
saddlebrown	lawngreen	aliceblue	lavenderblush
sandybrown	honeydew	dodgerblue	palevioletred
peachpuff	darkseagreen	lightslategray	crimson
peru	palegreen	lightslategrey	pink
linen	lightgreen	slategray	lightpink

Figure A.1: The named colors supported by the current program

Acknowledgements

Thanks to Yuancheng Lin, Jingyi Duan for helpful discussions and tests.

Bibliography

- [1] Beatriz Cordero et al. Covalent radii revisited. Dalton Transactions, 2008: 2832–2838.
- [2] Herbert Goldstein, Charles P. Poole Jr. and John L. Safko, Classical Mechanics (3rd Edition). Goldstein Poole & Safko, 2001.
- [3] Chongyu Wang and Feng An and Binglin Gu and Liu Fusui and Ying Chen, Electronic structure of the light-impurity (boron)–vacancy complex in iron. Physical Review B, 1988: 3905–3912.
- [4] Chong-yu Wang, Sen-ying Liu and Lin-guang Han, Electronic structure of impurity (oxygen)–stacking-fault complex in nickel. Physical Review B, 1990: 1359–1367.