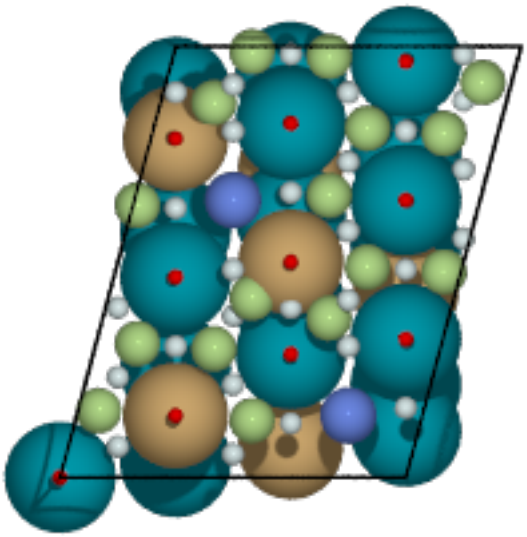
CatKit

Release 0.5.0

Apr 18, 2019

Contents

Welcome to CatKit! A staging ground for computational tools which are generally useful for catalysis research. The goal of the project is to provide a communal location for those interested in hosting such tools under a common banner. In doing so, we hope to provide the infrastructure to produce more advanced functionality based on modular components of individual contributors.



CHAPTER 1

Contents

1.1 Modules

1.1.1 Subpackages

catkit.gen package

Subpackages

catkit.gen.analysis package

Submodules

catkit.gen.analysis.classifier module

class catkit.gen.analysis.classifier.**Classifier**(*atoms*)

Class for classification of various aspects of an an atomic unit cell.

Currently, a tool for classification of adsorbates on surface environments and the active sites they rest on.

id_adsorbate_atoms(*classifier='trivial'*, *tag=False*)

Identify adsorbed atoms in a given atoms object.

Parameters

- **classifier** (*str*) – Classification technique to identify adsorbate atoms.

  'trivial': Adsorbate atoms assumed to have atomic number != 13 or < 21.

- **tag** (*bool*) – Return adsorbate atoms with tags of -2.

Returns **ads_atoms** – Index of adsorbate atoms found.

Return type ndarray (n,)

id_adsorbates(*classifier='radial'*, *return_atoms=False*)
Return a list of Gratoms objects for each adsorbate classified on a surface. Requires classification of adsorbate atoms.

Parameters

- **classifier** (*str*) – Classification technique to identify individual adsorbates.

  'radial': Use standard cutoff distances to identify neighboring atoms.

- **return_atoms** (*bool*) – Return Gratoms objects instead of adsorbate indices.

Returns **adsorbates** – Adsorbate indices of adsorbates in unit cell.

Return type list (n,)

id_slab_atoms(*classifier='trivial'*, *tag=False*, *rtol=0.001*)
Return the indices of the slab atoms using select characterization techniques.

Parameters

- **classifier** (*str*) – Classification technique to identify slab atoms.

  'trivial': Slab atoms assumed to have atomic number == 13 or >= 21.

- **tag** (*bool*) – Return adsorbate atoms with tags of 2.
- **rtol** (*float*) – Relative cutoff distance for tagging layers.

Returns **slab_atoms** – Index of slab atoms found.

Return type ndarray (n,)

id_surface_atoms(*classifier='voronoi_sweep'*)
Identify surface atoms of an atoms object. This will require that adsorbate atoms have already been identified.

Parameters **classifier** (*str*) – Classification technique to identify surface atoms.

'voronoi_sweep': Create a sweep of proxy atoms above surface. Surface atoms are those which are most frequent neighbors of the sweep.

Returns **surface_atoms** – Index of the surface atoms in the object.

Return type ndarray (n,)

catkit.gen.analysis.matching module

catkit.gen.analysis.matching.**reactant_indices**(*R1*, *R2*, *P*, *broken_bond*)

Match the indices of a pair of reactants from a product and broken bond.

Parameters

- **R1** (*networkx MultiGraph*) – Graph representing reactant 1
- **R2** (*networkx MultiGraph*) – Graph representing reactant 2
- **P** (*networkx MultiGraph*) – Graph representing the product
- **broken_bond** (*list (2,)*) – Indices representing the edge of the product to be removed.

Returns **pindex** – Indices of the product graph sorted by the order of the reactants indices.

Return type ndarrays (n,)

catkit.gen.analysis.matching.**slab_indices**(*slab0*, *slab1*, *mask=None*)
Match the indices of similar atoms between two slabs.

catkit.gen.analysis.reconfiguration module

Module contents

class catkit.gen.analysis.**Classifier**(*atoms*)
Class for classification of various aspects of an an atomic unit cell.

Currently, a tool for classification of adsorbates on surface environments and the active sites they rest on.

id_adsorbate_atoms(*classifier='trivial'*, *tag=False*)
Identify adsorbed atoms in a given atoms object.

Parameters

- **classifier** (*str*) – Classification technique to identify adsorbate atoms.

  'trivial': Adsorbate atoms assumed to have atomic number != 13 or < 21.

- **tag** (*bool*) – Return adsorbate atoms with tags of -2.

Returns **ads_atoms** – Index of adsorbate atoms found.

Return type ndarray (n,)

id_adsorbates(*classifier='radial'*, *return_atoms=False*)
Return a list of Gratoms objects for each adsorbate classified on a surface. Requires classification of adsorbate atoms.

Parameters

- **classifier** (*str*) – Classification technique to identify individual adsorbates.

  'radial': Use standard cutoff distances to identify neighboring atoms.

- **return_atoms** (*bool*) – Return Gratoms objects instead of adsorbate indices.

Returns **adsorbates** – Adsorbate indices of adsorbates in unit cell.

Return type list (n,)

id_slab_atoms(*classifier='trivial'*, *tag=False*, *rtol=0.001*)
Return the indices of the slab atoms using select characterization techniques.

Parameters

- **classifier** (*str*) – Classification technique to identify slab atoms.

  'trivial': Slab atoms assumed to have atomic number == 13 or >= 21.

- **tag** (*bool*) – Return adsorbate atoms with tags of 2.

- **rtol** (*float*) – Relative cutoff distance for tagging layers.

Returns **slab_atoms** – Index of slab atoms found.

Return type ndarray (n,)

id_surface_atoms(*classifier='voronoi_sweep'*)
Identify surface atoms of an atoms object. This will require that adsorbate atoms have already been identified.

Parameters **classifier** (*str*) – Classification technique to identify surface atoms.

'voronoi_sweep': Create a sweep of proxy atoms above surface. Surface atoms are those which are most frequent neighbors of the sweep.

Returns **surface_atoms** – Index of the surface atoms in the object.

Return type ndarray (n,)

Submodules

catkit.gen.adsorption module

class catkit.gen.adsorption.**AdsorptionSites**(*slab*, *surface_atoms=None*, *tol=1e-05*)

Adsorption site object.

ex_sites(*index*, *select='inner'*, *cutoff=0*)

Get site indices inside or outside of a cutoff radii from a provided periodic site index. If two sites are provided, an option to return the mutually inclusive points is also available.

get_adsorption_edges(*symmetric=True*, *periodic=True*)

Return the edges of adsorption sties defined as all regions with adjacent vertices.

Parameters

- **symmetric** (*bool*) – Return only the symmetrically reduced edges.
- **periodic** (*bool*) – Return edges which are unique via periodicity.

Returns **edges** – All edges crossing ridge or vertices indexed by the expanded unit slab.

Return type ndarray (n, 2)

get_adsorption_vectors(*unique=True*, *screen=True*)

Returns the vectors representing the furthest distance from the neighboring atoms.

Returns **vectors** – Adsorption vectors for surface sites.

Return type ndarray (n, 3)

get_connectivity(*unique=True*)

Return the number of connections associated with each site.

get_coordinates(*unique=True*)

Return the 3D coordinates associated with each site.

get_periodic_sites(*screen=True*)

Return an index of the coordinates which are unique by periodic boundary conditions.

Parameters **screen** (*bool*) – Return only sites inside the unit cell.

Returns **periodic_match** – Indices of the coordinates which are identical by periodic boundary conditions.

Return type ndarray (n,)

get_symmetric_sites(*unique=True*, *screen=True*)

Determine the symmetrically unique adsorption sites from a list of fractional coordinates.

Parameters

- **unique** (*bool*) – Return only the unique symmetrically reduced sites.
- **screen** (*bool*) – Return only sites inside the unit cell.

Returns **sites** – Dictionary of sites containing index of site

Return type dict of lists

get_topology(*unique=True*)
Return the indices of adjacent surface atoms.

plot(*savefile=None*)
Create a visualization of the sites.

class catkit.gen.adsorption.**Builder**(*slab*, *surface_atoms=None*, *tol=1e-05*)
Bases: *catkit.gen.adsorption.AdsorptionSites*

Initial module for creation of 3D slab structures with attached adsorbates.

add_adsorbate(*adsorbate*, *bonds=None*, *index=0*, *auto_construct=True*, ***kwargs*)
Add and adsorbate to a slab. If the auto_constructor flag is False, the atoms object provided will be attached at the active site.

Parameters

- **adsorbate** (*gratoms object*) – Molecule to connect to the surface.
- **bonds** (*int or list of 2 int*) – Index of adsorbate atoms to be bonded.
- **index** (*int*) – Index of the site or edge to use as the adsorption position. A value of -1 will return all possible structures.
- **auto_construct** (*bool*) – Whether to automatically estimate the position of atoms in larger molecules or use the provided structure.

Returns **slabs** – Slab(s) with adsorbate attached.

Return type gratoms object

add_adsorbates(*adsorbates*, *indices*)

catkit.gen.adsorption.**get_adsorption_sites**(*slab*, *surface_atoms=None*, *symmetry_reduced=True*, *tol=1e-05*)
Get the adsorption sites of a slab as defined by surface symmetries of the surface atoms.

Parameters

- **slab** (*Atoms object*) – The slab to find adsorption sites for. Must have surface atoms defined.
- **surface_atoms** (*array_like (N,)*) – List of the surface atom indices.
- **symmetry_reduced** (*bool*) – Return the symmetrically unique sites only.
- **adsorption_vectors** (*bool*) – Return the adsorption vectors.

Returns

- **coordinates** (*ndarray (N, 3)*) – Cartesian coordinates of activate sites.
- **connectivity** (*ndarray (N,)*) – Number of bonds formed for a given adsorbate.
- **symmetry_index** (*ndarray (N,)*) – Arbitrary indices of symmetric sites.

catkit.gen.adsorption.**symmetry_equivalent_points**(*fractional_coordinates*, *atoms*, *tol=1e-05*)
Return the symmetrically equivalent points from a list of provided fractional coordinates.

Parameters

- **fractional_coordinates** (*ndarray (N ,3)*) – Fractional coordinates to find symmetrical equivalence between.

- **atoms** (*Atoms object*) – Atoms object to use the unit cell, positions, and pbc of.
- **tol** (*float*) – Float point precision tolerance.

Returns **symmetry_match** – Indices of fractional coordinates which are unique or matching.

Return type ndarray (N,)

catkit.gen.geometry module

catkit.gen.molecules module

catkit.gen.molecules.**bin_hydrogen**(*hydrogens=1*, *bins=1*)
Recursive function for determining distributions of hydrogens across bins.

catkit.gen.molecules.**get_3D_positions**(*atoms*, *bond_index=None*)
Return an estimation of the 3D structure of a Gratoms object based on its graph.

WARNING: This function operates on the atoms object in-place.

Parameters

- **atoms** (*Gratoms object*) – Structure with connectivity matrix to provide a 3D structure.
- **bond_index** (*int*) – Index of the atoms to consider as the origin of a surface bonding site.

Returns **atoms** – Structure with updated 3D positions.

Return type Gratoms object

catkit.gen.molecules.**get_topologies**(*symbols*, *saturate=False*)
Return the possible topologies of a given chemical species.

Parameters

- **symbols** (*str*) – Atomic symbols to construct the topologies from.
- **saturate** (*bool*) – Saturate the molecule with hydrogen based on the default.radicals set.

Returns **molecules** – Gratoms objects with unique connectivity matrix attached. No 3D positions will be provided for these structures.

Return type list (N,)

catkit.gen.molecules.**hydrogenate**(*atoms*, *bins*, *copy=True*)
Add hydrogens to a gratoms object via provided bins

catkit.gen.pathways module

class catkit.gen.pathways.**ReactionNetwork**(*db_name='reaction-network.db'*, *base_valence=None*, *nbond_limits=None*)
A class for accessing a temporary SQLite database. This function works as a context manager and should be used as follows:

with ReactionNetwork() as rn: (Perform operation here)

This syntax will automatically construct the temporary database, or access an existing one. Upon exiting the indentation, the changes to the database will be automatically committed.

create_table()
Create the SQLite database table framework.

load_3d_structures(*ids=None*)
Return Gratoms objects from the ReactionNetwork database.

Parameters **ids** (*int or list of int*) – Identifier of the molecule in the database. If None, return all structure.

Returns **images** – All Gratoms objects in the database.

Return type list

load_molecules(*ids=None*, *binned=False*)
Load 2D molecule graphs from the database.

Parameters **binned** (*bool*) – Return the molecules in sub-dictionaries of their corresponding composition and bonding tags.

Returns **molecules** – All molecules present in the database.

Return type dict

load_pathways(*broken_bonds=False*)
Save enumerated pathways the ReactionNetwork database. More than two reactants or two products is not supported.

Parameters **broken_bonds** (*bool*) – Return the index information of which bond was broken. Only supported for elementary steps.

Returns **pathways** – All pathways present in the database.

Return type list

molecule_search(*element_pool={'C': 2, 'H': 6}*, *load_molecules=True*, *multiple_bond_search=False*)
Return the enumeration of molecules which can be produced from the specified atoms.

Parameters

- **element_pool** (*dict*) – Atomic symbols keys paired with the maximum number of that atom.
- **load_molecules** (*bool*) – Load any existing molecules from the database.
- **multiple_bond_search** (*bool*) – Allow atoms to form bonds with other atoms in the molecule.

path_search(*reconfiguration=True*, *substitution=True*)
Search for reaction mechanisms from a pre-populated database of molecules. By default, only single bond addition pathways are enumerated (Also called elementary steps).

Parameters

- **reconfiguration** (*bool*) – Search for reconfiguration paths. Reconfiguration paths are all those where only the bond order is changed. R1 –> P1.
- **substitution** (*bool*) – Search for substitution paths. Substitution paths are all those where one bond is broken and one bond is formed simultaneously. R1 + R2 –> P1 + P2.

plot_reaction_network(*file_name=None*)
Plot the reaction network present in the database.

save_3d_structure(*gratoms*, *overwrite=False*)
Save Cartesian coordinates into the ReactionNetwork database.

Parameters

- **gratoms** (*Atoms object*) – Structure containing Cartesian coordinates to be saved.
- **overwrite** (*bool*) – Allow the database to overwrite a matching index.

save_molecules(*molecules*)
Save enumerated molecules to the ReactionNetwork database.

Parameters **molecules** (*dict*) – Molecules to be saved to the database.

save_pathways(*pathways*, *broken_bonds=None*)
Save enumerated pathways the ReactionNetwork database. More than two reactants or two products is not supported.

Parameters

- **pathways** (*list*) – Sorted pathways in the form [R1, R2, P1, P2].
- **broken_bonds** (*list*) – Comma separated strings of index associated with the two atoms whos bond is broken. List order must match pathways.

catkit.gen.route module

catkit.gen.route.**get_heppel_sellers**(*nu*, *terminal*)
Returns an array of linearly independent reaction routes as described by Heppel-Sellers reaction route enumeration.

Parameters

- **nu** (*ndarray (n, m)*) – The stoichiometric matrix of n species and m mechanisms.
- **terminal** (*ndarray (j,)*) – Indices of the m species to be considered as terminal

Returns **sigma** – Linearly independent set of Heppel-Sellers reaction routes.

Return type ndarray (m, k)

catkit.gen.route.**get_reaction_routes**(*nu*, *sigma*, *empty_routes=True*, *independent_only=False*)
Returns an array of reaction routes. Returns all full reaction routes by default.

Parameters

- **nu** (*ndarray (n, m)*) – The stoichiometric matrix of n species and m mechanisms.
- **sigma** (*ndarray (m, j)*) – A linearly independent set of reaction routes.
- **empty_routes** (*bool*) – Return the empty routes along with the full routes.
- **independent_only** (*bool*) – Return only a linearly independent set of full reaction routes. Can take less time.

Returns

- **FR** (*ndarray (m, k)*) – Enumerated full reaction routes.
- **ER** (*ndarray (m, l)*) – Enumerated empty reaction routes.

catkit.gen.route.**get_response_reactions**(*epsilon*, *selection=None*, *species=False*)
Returns an array of possible response reaction routes for a given chemical formula array.

Parameters

- **epsilon** (*ndarray (n, m)*) – Chemical formula array of n elements by m molecular species.

- **selection** (*ndarray (j,)*) – Indices of the m species to be considered as terminal
- **species** (*bool*) – Return the indices of the chemical species used.

Returns

- **RER** (*ndarray (m, k)*) – Possible response reaction routes.
- **index** (*ndarray (j, k)*) – Indices of the k terminal species use to produce the l response reactions.

catkit.gen.surface module

class catkit.gen.surface.**SlabGenerator**(*bulk*, *miller_index*, *layers*, *vacuum=None*, *fixed=None*, *layer_type='ang'*, *attach_graph=True*, *standardize_bulk=False*, *primitive=True*, *tol=1e-08*)

Bases: object

Class for generation of slab unit cells from bulk unit cells.

Many surface operations rely upon / are made easier through the bulk basis cell they are created from. The SlabGenerator class is designed to house these operations.

Return the miller indices associated with the users requested values. Follows the following steps:

- Convert Miller-Bravais notation into standard Miller index.
- (optional) Ensure the bulk cell is in its standard form.
- Convert the indices to the cell for the primitive lattice.
- Reduce the indices by their greatest common divisor.

Parameters

- **bulk** (*Atoms object*) – Bulk system to be converted into slab.
- **miller_index** (*list (3,) or (4,)*) – Miller index to construct surface from. If length 4, Miller-Bravais notation is assumed.
- **layers** (*int*) – Number of layers to include in the slab. A slab layer is defined as a unique z-coordinate.
- **vacuum** (*float*) – Angstroms of vacuum to apply to the slab.
- **fixed** (*int*) – Number of slab layers to constrain.
- **layer_type** (*'angs', 'trim', 'stoich', or 'sym'*) – Determines how to perform slab layering.

  'angs': Layers denotes the thickness of the slab in Angstroms. 'trim': The slab will be trimmed to a number of layers equal to the exact number of unique z-coordinates. Useful for precision control. 'stoich' : Constraints any slab generated to have the same stoichiometric ratio as the provided bulk. 'sym' : Return a slab which is inversion symmetric. i.e. The same on both sides.
- **attach_graph** (*bool*) – Attach the connectivity graph generated from the bulk structure. This is only necessary for fingerprinting and setting it to False can save time. Surface atoms will be found regardless.

- **standardize_bulk** (*bool*) – Covert the bulk input to its standard form before and produce the cleave from it. This is highly recommended as Miller indices are not defined for non-standard cells.
- **tol** (*float*) – Tolerance for floating point rounding errors.

adsorption_sites(*slab*, ***kwargs*)

Helper function to return the adsorption sites of the provided slab.

Parameters **slab** (*atoms object*) – The slab to find adsorption sites for. Assumes you are using the same basis.

Returns **output** – Coordinates and connectivity of the adsorption sites. The symmetry indices can also be returned.

Return type tuple (n, n) | (n, n, n)

align_crystal(*bulk*, *miller_index*)

Return an aligned unit cell from bulk unit cell. This alignment rotates the a and b basis vectors to be parallel to the Miller index.

Parameters

- **bulk** (*Atoms object*) – Bulk system to be standardized.
- **miller_index** (*list (3,)*) – Miller indices to align with the basis vectors.

Returns **new_bulk** – Standardized bulk unit cell.

Return type Gratoms object

get_slab(*size=1*, *iterm=0*)

Generate a slab from the bulk structure. This function is meant specifically for selection of an individual termination or enumeration through surfaces of various size.

This function will orthogonalize the c basis vector and align it with the z-axis which breaks bulk symmetry along the z-axis.

Parameters

- **size** (*int, array_like (2,) or (2, 2)*) – Size of the unit cell to create as described in *set_size()*.
- **iterm** (*int*) – A termination index in reference to the list of possible terminations.

Returns **slab** – The modified basis slab produced based on the layer specifications given.

Return type Gratoms object

get_slab_basis(*iterm=0*, *maxn=20*)

Return a list of all terminations which have been properly shifted and with an appropriate number of layers added. This function is mainly for performance, to prevent looping over other operations which are not related the size of the slab.

This step also contains periodically constrained orthogonalization of the c basis. This implementation only works if the a anb b basis vectors are properly aligned with the x and y axis. This is strictly to assist the correct identification of surface atoms.

Only produces the terminations requested as a lazy evaluator.

Parameters

- **iterm** (*int*) – Index of the slab termination to return.
- **maxn** (*int*) – The maximum integer component to search for a more orthogonal bulk.

Returns **ibasis** – Prepared, ith basis.

Return type Gratoms object

get_unique_terminations()
Determine the fractional coordinate shift that will result in a unique surface termination. This is not required if bulk standardization has been performed, since all available z shifts will result in a unique termination for a primitive cell.

Returns **unique_shift** – Fractional coordinate shifts which will result in unique terminations.

Return type array (n,)

make_symmetric(*slab*)
Returns a symmetric slab. Note, this will trim the slab potentially resulting in loss of stoichiometry.

set_size(*slab*, *size*)
Set the size of a slab based one of three methods.

1. An integer value performs a search of valid matrix operations to perform on the ab-basis vectors to return a set which with a minimal sum of distances and an angle closest to 90 degrees.

2. An array_like of length 2 will multiply the existing basis vectors by that amount.

3. An array of shape (2, 2) will be interpreted as matrix notation to multiply the ab-basis vectors by.

Parameters

- **slab** (*Atoms object*) – Slab to be made into the requested size.
- **size** (*int, array_like (2,) or (2, 2)*) – Size of the unit cell to create as described above.

Returns **supercell** – Supercell of the requested size.

Return type Gratoms object

catkit.gen.surface.**convert_miller_index**(*miller_index*, *atoms1*, *atoms2*)
Return a converted miller index between two atoms objects.

catkit.gen.surface.**generate_indices**(*max_index*)
Return an array of miller indices enumerated up to values plus or minus some maximum. Filters out lists with greatest common divisors greater than one. Only positive values need to be considered for the first index.

Parameters **max_index** (*int*) – Maximum number that will be considered for a given surface.

Returns **unique_index** – Unique miller indices

Return type ndarray (n, 3)

catkit.gen.surface.**get_degenerate_indices**(*bulk*, *miller_index*)
Return the miller indices which are degenerate to a given miller index for a particular bulk structure.

Parameters

- **bulk** (*Atoms object*) – Bulk structure to get the degenerate miller indices.
- **miller_index** (*array_like (3,)*) – Miller index to get the degenerate indices for.

Returns **degenerate_indices** – Degenerate miller indices to the provided index.

Return type array (N, 3)

catkit.gen.surface.**get_unique_indices**(*bulk*, *max_index*)
Returns an array of miller indices which will produce unique surface terminations based on a provided bulk structure.

Parameters

- **bulk** (*Atoms object*) – Bulk structure to get the unique miller indices.
- **max_index** (*int*) – Maximum number that will be considered for a given surface.

Returns **unique_millers** – Symmetrically distinct miller indices for a given bulk.

Return type ndarray (n, 3)

catkit.gen.surface.**transform_ab**(*slab*, *matrix*, *tol=1e-05*)

Transform the slab basis vectors parallel to the z-plane by matrix notation. This can result in changing the slabs cell size. This can also result in very unusual slab dimensions, use with caution.

Parameters

- **slab** (*Atoms object*) – The slab to be transformed.
- **matrix** (*array_like (2, 2)*) – The matrix notation transformation of the a and b basis vectors.
- **tol** (*float*) – Float point precision tolerance.

Returns **slab** – Slab after transformation.

Return type Atoms object

catkit.gen.utils module

catkit.gen.utils.**get_voronoi_neighbors**(*atoms*, *cutoff=5.0*)

Return the connectivity matrix from the Voronoi method. Multi-bonding occurs through periodic boundary conditions.

Parameters

- **atoms** (*atoms object*) – Atoms object with the periodic boundary conditions and unit cell information to use.
- **cutoff** (*float*) – Radius of maximum atomic bond distance to consider.

Returns **connectivity** – Number of edges formed between atoms in a system.

Return type ndarray (n, n)

catkit.gen.utils.**get_cutoff_neighbors**(*atoms*, *cutoff=None*, *atol=1e-08*)

Return the connectivity matrix from a simple radial cutoff. Multi-bonding occurs through periodic boundary conditions.

Parameters

- **atoms** (*atoms object*) – Atoms object with the periodic boundary conditions and unit cell information to use.
- **cutoff** (*float*) – Cutoff radius to use when determining neighbors.
- **atol** (*float*) – Absolute tolerance to use when computing distances.

Returns **connectivity** – Number of edges formed between atoms in a system.

Return type ndarray (n, n)

catkit.gen.utils.**get_integer_enumeration**(*N=3, span=[0, 2]*)

Return the enumerated array of a span of integer values. These enumerations are limited to the length N.

For the default span of [0, 2], the enumeration equates to the corners of an N-dimensional hypercube.

Parameters

- **N** (*int*) – Length of enumerated lists to produce.
- **span** (*list | slice*) – The range of integers to be considered for enumeration.

Returns **enumeration** – Enumeration of the requested integers.

Return type ndarray (M, N)

catkit.gen.utils.**trilaterate**(*centers*, *r*, *zvector=None*)

Find the intersection of two or three spheres. In the case of two sphere intersection, the z-coordinate is assumed to be an intersection of a plane whose normal is aligned with the points and perpendicular to the positive z-coordinate.

If more than three spheres are supplied, the centroid of the points is returned (no radii used).

Parameters

- **centers** (*list | ndarray (n, 3)*) – Cartesian coordinates representing the center of each sphere
- **r** (*list | ndarray (n,)*) – The radii of the spheres.
- **zvector** (*ndarray (3,)*) – The vector associated with the upward direction for under-specified coordinations (1 and 2).

Returns **intersection** – The point where all spheres/planes intersect.

Return type ndarray (3,)

catkit.gen.utils.**get_unique_xy**(*xyz_coords*, *cutoff=0.1*)

Return the unique coordinates of an atoms object for the requrested atoms indices. Z-coordinates are projected to maximum z-coordinate by default.

Parameters

- **xyz_coords** (*ndarray (n, 3)*) – Cartesian coordinates to identify unique xy positions from.
- **cutoff** (*float*) – Distance in Angstrons to consider xy-coordinate unique within.

Returns **xy_pos** – Unique xy coordinates projected onto a maximal z coordinate.

Return type ndarray (m, 3)

catkit.gen.utils.**expand_cell**(*atoms*, *cutoff=None*, *padding=None*)

Return Cartesian coordinates atoms within a supercell which contains repetitions of the unit cell which contains at least one neighboring atom.

Parameters

- **atoms** (*Atoms object*) – Atoms with the periodic boundary conditions and unit cell information to use.
- **cutoff** (*float*) – Radius of maximum atomic bond distance to consider.
- **padding** (*ndarray (3,)*) – Padding of repetition of the unit cell in the x, y, z directions. e.g. [1, 0, 1].

Returns

- **index** (*ndarray (N,)*) – Indices associated with the original unit cell positions.
- **coords** (*ndarray (N, 3)*) – Cartesian coordinates associated with positions in the supercell.
- **offsets** (*ndarray (M, 3)*) – Integer offsets of each unit cell.

catkit.gen.utils.**matching_sites**(*position*, *comparators*, *tol=1e-08*)
Get the indices of all points in a comparator list that are equal to a given position (with a tolerance), taking into account periodic boundary conditions (adaptation from Pymatgen).

This will only accept a fractional coordinate scheme.

Parameters

- **position** (*list (3,)*) – Fractional coordinate to compare to list.
- **comparators** (*list (3, n)*) – Fractional coordinates to compare against.
- **tol** (*float*) – Absolute tolerance.

Returns **match** – Indices of matches.

Return type list (n,)

catkit.gen.utils.**get_basis_vectors**(*coordinates*)
Return a set of basis vectors for a given array of 3D coordinates.

Parameters **coordinates** (*array_like (3, 3) | (2, 3)*) – Cartesian coordinates to determine the basis of. If only 2 positions are given 3rd is chosen as the positive y-axis.

Returns **basis_vectors** – Automatically generated basis vectors from the given positions.

Return type ndarray (3, 3)

catkit.gen.utils.**connectivity_to_edges**(*connectivity*, *indices=None*)
Convert a Numpy connectivity matrix into a list of NetworkX compatible edges.

catkit.gen.utils.**isomorphic_molecules**(*graph0*, *graph1*)
Check whether two molecule graphs are isomorphic.

catkit.gen.utils.**matching_coordinates**(*position*, *comparators*, *tol=1e-08*)
Get the indices of all points in a comparator list that are equal to a given position (with a tolerance), taking into account periodic boundary conditions (adaptation from Pymatgen).

This will only accept a Cartesian coordinate scheme. TODO: merge this with matching_sites.

Parameters

- **position** (*list (3,)*) – Fractional coordinate to compare to list.
- **comparators** (*list (3, N)*) – Fractional coordinates to compare against.
- **tol** (*float*) – Absolute tolerance.

Returns **match** – Indices of matches.

Return type list (N,)

catkit.gen.utils.**get_unique_coordinates**(*atoms*, *axis=2*, *tag=False*, *tol=0.001*)
Return unique coordinate values of a given atoms object for a specified axis.

Parameters

- **atoms** (*object*) – Atoms object to search for unique values along.
- **axis** (*int (0, 1, or 2)*) – Look for unique values along the x, y, or z axis.
- **tag** (*bool*) – Assign ASE-like tags to each layer of the slab.
- **tol** (*float*) – The tolerance to search for unique values within.

Returns **values** – Array of unique positions in fractional coordinates.

Return type ndarray (n,)

catkit.gen.utils.**get_reciprocal_vectors**(*atoms*)
Return the reciprocal lattice vectors to a atoms unit cell.

catkit.gen.utils.**plane_normal**(*xyz*)
Return the surface normal vector to a plane of best fit.

Parameters **xyz** (*ndarray (n, 3)*) – 3D points to fit plane to.

Returns **vec** – Unit vector normal to the plane of best fit.

Return type ndarray (1, 3)

catkit.gen.utils.**running_mean**(*array*, *N=5*)
Calculate the running mean of array for N instances.

Parameters

- **array** (*array_like | ndarray (N,)*) – Array of values to have a average taken from.
- **N** (*int*) – Number of values to take an average with.

Returns **running_mean** – Mean value of the running average.

Return type ndarray (N + 1,)

catkit.gen.utils.**to_gratoms**(*atoms*, *edges=None*)
Convert and atom object to a gratoms object.

catkit.gen.utils.**get_atomic_numbers**(*formula*, *return_count=False*)
Return the atomic numbers associated with a chemical formula.

Parameters

- **formula** (*string*) – A chemical formula to parse into atomic numbers.
- **return_count** (*bool*) – Return the count of each element in the formula.

Returns

- **numbers** (*ndarray (n,)*) – Element numbers in associated species.
- **counts** (*ndarray (n,)*) – Count of each element in a species.

catkit.gen.utils.**get_reference_energies**(*species*, *energies*)
Get reference energies for the elements in a set of molecules.

Parameters

- **species** (*list (n,)*) – Chemical formulas for each molecular species.
- **energies** (*list (n,)*) – Total energies associated with each species.

Returns

- **elements** (*ndarray (n,)*) – Atomic elements associated with all species.
- **references** (*ndarray (n,)*) – Reference energies associated with each element.

catkit.gen.utils.**parse_slice**(*slice_name*)
Return a correctly parsed slice from input of varying types.

catkit.gen.utils.**ext_gcd**(*a*, *b*)
Extension of greatest common divisor.

catkit.gen.utils.**list_gcd**(*values*)
Return the greatest common divisor of a list of values.

Module contents

Catalysis Generator.

catkit.pawprint package

Submodules

catkit.pawprint.generator module

class catkit.pawprint.generator.**Fingerprinter**(*images=None*)

Parent class for all fingerprint generators.

get_fp(*parameters*, *operation_list*)

Return the fingerprints for a list of images of single atoms object for the given parameters. Convolutions will be performed for specific operations if the parameters is provided as a list of lists.

Parameters

- **parameters** (*list of str | list of lists (M,)*) – Names of seeding parameters available in the parameters database. If a list lists is provided, the number of lists must be equal to the number of operations.
- **operation_list** (*list of func | str (M,)*) – A list of operation functions to produce the fingerprints from. The names (str) of operations functions can also be used.

Returns **fingerprints** – Fingerprints for the images produced from the provided seed parameters.

Return type ndarray (N, X)

catkit.pawprint.generator.**get_connectivity**(*atoms*, *method=None*)

Returns an estimate of the connectivity matrix for a given atoms-object from CatGen.

Parameters

- **atoms** (*object*) – Molecular structure with out without adsorbates.
- **method** (*str (None or 'voronoi')*) – Method for estimating the connectivity matrix:

  None - standard cutoff radius method. voronoi - best suited for bulk characterization.

Returns **connectivity** – Estimated connectivity matrix where n is the number of atoms in the atoms-object.

Return type ndarray (N, N)

catkit.pawprint.operations module

catkit.pawprint.operations.**autocorrelation**(*atoms=None*, *atoms_parameters=None*, *connectivity=None*, *d=0*)

Autocorrelation convolution for systems without pbc.

catkit.pawprint.operations.**bonding_convolution**(*atoms=None*, *atoms_parameters=None*, *connectivity=None*)

Perform convolution of metal atoms with bonded adsorbates.

catkit.pawprint.operations.**derived_fp**(*atoms=None*, *atoms_parameters=None*, *connectivity=None*, *fp_1=None*, *fp_2=None*, *n_1=None*, *n_2=None*, *op=None*)

NOTE : This is a work in progress. I'll redesign the whole thing to allow for arithmetic manipulation of two fingerprints.

Given two fingerprints vector, it will perform arithmetic operation to design new fingerprints.

Parameters **op** (*str ('add' | 'subtract' | 'divide' | 'multiply')*) – add - Adds two fingerprints of equal length raised to their given power subtract - subtracts two fingerprints of equal length raised to

their given power

mulltiply - multiply two fingerprints of equal length raised to their given power

divide - divide two fingerprints of equal length raised to their given power

catkit.pawprint.operations.**layered_sum**(*atoms=None*, *atoms_parameters=None*, *connectivity=None*)

Sum of the properties in a layer as indicated by catkit tags.

catkit.pawprint.operations.**local_ads_metal_fp**(*atoms=None*, *atoms_parameters=None*, *connectivity=None*, *fuse=False*)

Sum of the differences in properties of the atoms in the metal-adsorbate interface

catkit.pawprint.operations.**periodic_convolution**(*atoms=None*, *atoms_parameters=None*, *connectivity=None*, *d=0*, *normalize=False*)

Return the square of each property with each atom. For distance 1 the covolutions returns the multiple of each property for all neighboring atom pairs.

catkit.pawprint.operations.**raw_properties**(*atoms=None*, *atoms_parameters=None*, *connectivity=None*)

Return all atom properties without manipulation.

Module contents

class catkit.pawprint.**Fingerprinter**(*images=None*)

Parent class for all fingerprint generators.

get_fp(*parameters*, *operation_list*)

Return the fingerprints for a list of images of single atoms object for the given parameters. Convolutions will be performed for specific operations if the parameters is provided as a list of lists.

Parameters

- **parameters** (*list of str | list of lists (M,)*) – Names of seeding parameters available in the parameters database. If a list lists is provided, the number of lists must be equal to the number of operations.
- **operation_list** (*list of func | str (M,)*) – A list of operation functions to produce the fingerprints from. The names (str) of operations functions can also be used.

Returns **fingerprints** – Fingerprints for the images produced from the provided seed parameters.

Return type ndarray (N, X)

class catkit.pawprint.**FingerprintDB**(*db_name='fingerprints.db'*, *verbose=False*)

A class for accessing a temporary SQLite database. This function works as a context manager and should be used as follows:

with FingerprintDB() as fpdb: (Perform operation here)

This syntax will automatically construct the temporary database, or access an existing one. Upon exiting the indentation, the changes to the database will be automatically committed.

create_table()
Creates the database table framework used in SQLite. This includes 3 tables: images, parameters, and fingerprints.

The images table currently stores ase_id information and a unqiue string. This can be adapted in the future to support atoms objects.

The parameters table stores a symbol (10 character maximum) for convenient reference and a description of the parameter.

The fingerprints table holds a unique image and parmeter ID along with a float value for each. The ID pair must be unique.

fingerprint_entry(*ase_id*, *param_id*, *value*)
Enters a fingerprint value to the database for a given ase and parameter id.

Parameters

- **ase_id** (*int*) – The unique id associated with an atoms object in the database.
- **param_id** (*int or str*) – The parameter ID or symbol associated with and entry in the parameters table.
- **value** (*float*) – The value of the parameter for the atoms object.

get_fingerprints(*ase_ids=None*, *params=[]*)
Get the array of values associated with the provided parameters for each ase_id.

Parameters

- **ase_id** (*list*) – The ase-id associated with an atoms object in the database.
- **params** (*list*) – List of symbols or int in parameters table to be selected.

Returns **fingerprint** – An array of values associated with the given parameters (a fingerprint) for each ase_id.

Return type array (n,)

get_parameters(*selection=None*, *display=False*)
Get an array of integer values which correspond to the parameter IDs for a set of provided symbols.

Parameters

- **selection** (*list*) – Symbols in parameters table to be selected. If no selection is made, return all parameters.
- **display** (*bool*) – Print parameter descriptions.

Returns **parameter_ids** – Integer values of selected parameters.

Return type array (n,)

image_entry(*d*, *identity=None*)
Enters a single ase-db image into the fingerprint database. The ase-db ID with identity must be unique. If not, it will be skipped.

This table can be expanded to contain atoms objects in the future.

Parameters

- **d** (*ase-db object*) – Database entry to parse.

- **identity** (*str*) – An identifier of the users choice.

Returns **ase_id** – The ase id collected.

Return type int

parameter_entry(*symbol=None*, *description=None*)
Enters a unique parameter into the database.

Parameters

- **symbol** (*str*) – A unique symbol the entry can be referenced by. If None, the symbol will be the ID of the parameter as a string.
- **description** (*str*) – A description of the parameter.

catkit.flow package

Submodules

catkit.flow.fwespresso module

catkit.flow.fwio module

catkit.flow.fwio.**array_to_list**(*data*)
A function to covert all arrays in a structure of embeded dictionaries and lists into lists themselves.

catkit.flow.fwio.**atoms_to_encode**(*images*)
Converts an list of atoms objects to an encoding from a .traj file.

catkit.flow.fwio.**encode_to_atoms**(*encode*, *out_file='input.traj'*)
Dump the encoding to a local traj file.

catkit.flow.hpcio module

catkit.flow.qeio module

Module contents

Catalysis WorkFlow.

class catkit.flow.**Laminar**(*host*, *username=None*, *name=None*, *password=None*)
Simple submission script helper for CatFlow.

bulk_relaxation(*atoms*, *parameters*, *spec=None*)
Run a relaxation of a given DB entry or atoms object. If a database object is used, the calculation will automatically store the keys and data for later retrieval.

The entries uuid will also be stored and $data.calculator_parameters$ will be used as the calculation parameters.

Parameters

- **images** (*Atoms object*) – Initial atoms to perform workflow on.
- **parameters** (*dict*) – Calculation parameters to use.
- **workflow_name** (*str*) – Name of the fireworks calculation to be used.

- **spec** (*dict*) – Additional fireworks specifications to pass to the database.

submit_relaxation(*image*, *workflow_name*, *parameters=None*, *spec=None*)
Run a relaxation of a given DB entry or atoms object. If a database object is used, the calculation will automatically store the keys and data for later retrieval.

The entries uuid will also be stored and $data.calculator_parameters$ will be used as the calculation parameters.

Parameters

- **images** (*Atoms object | AtomsRow object*) – ASE database entry or atoms object to relax.
- **workflow_name** (*str*) – Name of the fireworks calculation to be used.
- **parameters** (*dict*) – Calculation parameters to use. Will be pulled from a database entry $data.calculator_parameters$.
- **spec** (*dict*) – Additional fireworks specifications to pass to the database.

submit_relaxation_db(*database*, *spec=None*)
Submit each entry of an ASE database for relaxation. This requires that the calculation parameters be stored in the data under $data.calculator_parameters$.

Parameters

- **database** (*str*) – Path to ASE database to be looped over for relaxation.
- **spec** (*dict*) – Additional specification to be passed to Fireworks.

catkit.hub package

Subpackages

catkit.hub.ase_tools package

Submodules

catkit.hub.ase_tools.gas_phase_references module

Module contents

Submodules

catkit.hub.ase_connect module

catkit.hub.cathubsqlite module

catkit.hub.cli module

catkit.hub.convert_traj module

catkit.hub.create_user module

catkit.hub.db2server module

catkit.hub.folder2db module

catkit.hub.folder_check module

catkit.hub.folderreader module

catkit.hub.maintain_server module

catkit.hub.make_folders_template module

catkit.hub.organize module

catkit.hub.postgresql module

catkit.hub.psql_server_connect module

catkit.hub.query module

catkit.hub.tools module

Module contents

1.1.2 catkit.build module

catkit.build.**bulk**(*name*, *crystalstructure=None*, *primitive=False*, ***kwargs*)

Return the standard conventional cell of a bulk structure created using ASE. Accepts all keyword arguments for the ase bulk generator.

Parameters

- **name** (*Atoms object | str*) – Chemical symbol or symbols as in 'MgO' or 'NaCl'.
- **crystalstructure** (*str*) – Must be one of sc, fcc, bcc, hcp, diamond, zincblende, rocksalt, cesiumchloride, fluorite or wurtzite.
- **primitive** (*bool*) – Return the primitive unit cell instead of the conventional standard cell.

Returns **standardized_bulk** – The conventional standard or primitive bulk structure.

Return type Gratoms object

catkit.build.**molecule**(*species*, *bond_index=None*, *vacuum=0*)

Return list of enumerated gas-phase molecule structures based on species and topology.

Parameters

- **species** (*str*) – The chemical symbols to construct a molecule from.
- **bond_index** (*int*) – Construct the molecule as though it were adsorbed to a surface parallel to the z-axis. Will bond by the atom index given.
- **vacuum** (*float*) – Angstroms of vacuum to pad the molecules with.

Returns **images** – 3D structures of the requested chemical species and topologies.

Return type list of Gratoms objects

catkit.build.**surface**(*elements*, *size*, *miller=(1, 1, 1)*, *termination=0*, *fixed=0*, *vacuum=10*, *orthogonal=False*, ***kwargs*)

A helper function to return the surface associated with a given set of input parameters to the general surface generator.

Parameters

- **elements** (*str or object*) – The atomic symbol to be passed to the as bulk builder function or an atoms object representing the bulk structure to use.
- **size** (*list (3,)*) – Number of time to expand the x, y, and z primitive cell.
- **miller** (*list (3,) or (4,)*) – The miller index to cleave the surface structure from. If 4 values are used, assume Miller-Bravis convention.
- **termination** (*int*) – The index associated with a specific slab termination.
- **fixed** (*int*) – Number of layers to constrain.
- **vacuum** (*float*) – Angstroms of vacuum to add to the unit cell.
- **orthogonal** (*bool*) – Force the slab generator to produce the most orthogonal slab.

Returns **slab** – Return a slab generated from the specified bulk structure.

Return type Gratoms object

1.1.3 catkit.enumeration module

catkit.enumeration.**surfaces**(*bulk*, *width*, *miller_indices=(1, 1, 1)*, *terminations=None*, *sizes=None*, *vacuum=10*, *fixed=0*, *layer_type='angs'*, ***kwargs*)

Return a list of enumerated surfaces based on symmetry properties of interest to the user. Any bulk structure provided will be standardized.

This function will take additional keyword arguments for the *catkit.gen.surface.SlabGenerator()* Class.

Parameters

- **bulk** (*str | Atoms*) – The atomic symbol to be passed to the as bulk builder function or an atoms object representing the bulk structure to use.
- **width** (*float*) – Minimum width of the slab in angstroms before trimming. Imposing symmetry requirements will reduce the width.
- **miller_indices** (*int | list (3,) | list of list (n, 3)*) – List of the miller indices to enumerate slabs for. If an integer is provided, the value is treated as the maximum miller index to consider for an enumeration of all possible unique miller indices.
- **terminations** (*int | array_like*) – Return the terminations associated with the provided indices. If -1, all possible terminations are enumerated.
- **sizes** (*None | int | array_like (n,)*) – Enumerate all surface sizes in the provided list. Sizes are integers which represent multiples of the smallest possible surface area. If None, return slabs with the smallest possible surface area. If an integer, enumerate all sizes up to that multiple.
- **vacuum** (*float*) – Angstroms of vacuum to add to the unit cell.
- **fixed** (*int*) – Number of layers to constrain.

- **layer_type** (*'angs' | 'trim' | 'stoich' | 'sym'*) – Method of slab layering to perform. See also: *catkit.gen.surface.SlabGenerator()*

Returns **slabs** – Return a list of enumerated slab structures.

Return type list of Gratoms objects

1.1.4 catkit.db module

1.1.5 catkit.gratoms module

class catkit.gratoms.**Gratoms**(*symbols=None*, *positions=None*, *numbers=None*, *tags=None*, *momenta=None*, *masses=None*, *magmoms=None*, *charges=None*, *scaled_positions=None*, *cell=None*, *pbc=None*, *celldisp=None*, *constraint=None*, *calculator=None*, *info=None*, *edges=None*)

Bases: ase.atoms.Atoms

Graph based atoms object.

An Integrated class for an ASE atoms object with a corresponding Networkx Graph.

adj

connectivity

copy()
Return a copy.

degree

edges

get_chemical_tags(*rank=2*)
Generate a hash descriptive of the chemical formula (rank 0) or include bonding (rank 1).

get_neighbor_symbols(*u*)
Get chemical symbols for neighboring atoms of u.

get_surface_atoms()
Return surface atoms.

get_unsaturated_nodes(*screen=None*)

graph

is_isomorph(*other*)
Check if isomorphic by bond count and atomic number.

nodes

set_surface_atoms(*top*, *bottom=None*)
Assign surface atoms.

1.1.6 catkit.learn module

catkit.learn.**online_learning**(*X, y, samples, factors=[1.0, 1.0], nsteps=40, plot=False*)
A simple utility for performing online learning. The main components required are a regression method and a scoring technique.

Currently, the scoring methodology and regressor are baked in. These need to be made modular.

Minimum 3 samples are required for 3 fold cross validation.

catkit.learn.**optimizer**(*obj_func*, *initial_theta*, *bounds*, *gradient=True*, *minimizer='L-BFGS-B'*, *hopping=0*, ***kwargs*)

Substitute optimizer in scikit-learn Gaussian Process function.

Note 'L-BFGS-B' is equivalent to the standard optimizer used in scikit-learn. This function allows for more direct control over the arguments. https://docs.scipy.org/doc/scipy/reference/optimize.html

Parameters

- **obj_func** (*function*) – scikit-learn objective function.
- **initial_theta** (*array (n,)*) – Hyperparameters to be optimized against.
- **bounds** (*list of tuples (n, 2)*) – Lower and upper bounds for each hyper parameter.
- **gradient** (*bool*) – Include the gradient for the optimization function.
- **minimizer** (*str*) – A scipy minimization method.
- **hopping** (*int*) – Perform a number of basin hopping steps.

Returns

- **theta_opt** (*list (n,)*) – Optimized hyperparameters.
- **func_min** (*float*) – Value of the minimized objective function.

Python Module Index

C

Index

G

H

I