TUTORIAL SYNOPSIS

Boron diffusion in bulk Silicon

Quantum Wise
When every atom matters

In this tutorial we will use ATK's Adaptive Kinetic Monte Carlo (AKMC) algorithm with DFT to investigate the diffusion of a single B atom in a bulk Si lattice. Specifically, we will investigate how mobile B is in a defect-free Si lattice, with B in a substitutional lattice site as the initial state. In order to do this, we will follow these steps:

- Optimize the silicon bulk to find the correct lattice constant for our computational model;
- create the B-doped Si crystal;
- run the AKMC simulation.

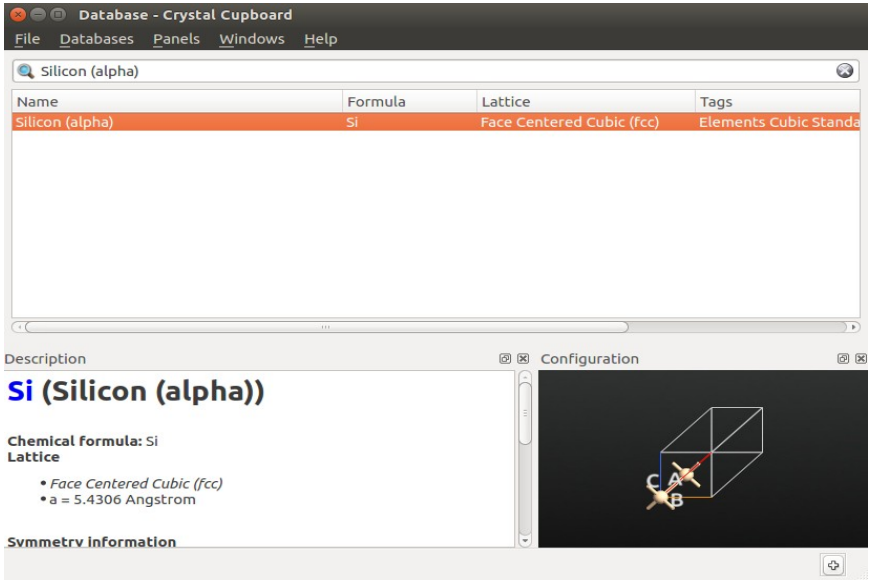
Optimizing the bulk Si

Builder

❖Go to the Database using Add → From Database.

❖Search for “Silicon (alpha)” and add it to the Stash.

❖Send the structure to the **Script Generator.**



Optimizing the bulk Si

Script Generator

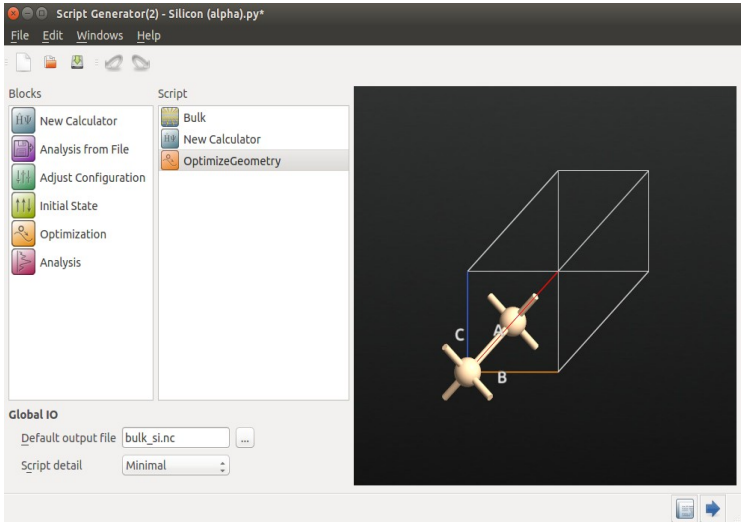
❖Add the following Blocks:

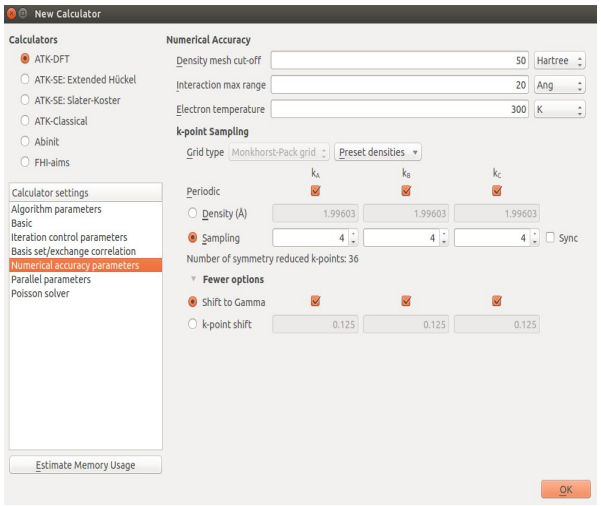
❖New Calculator

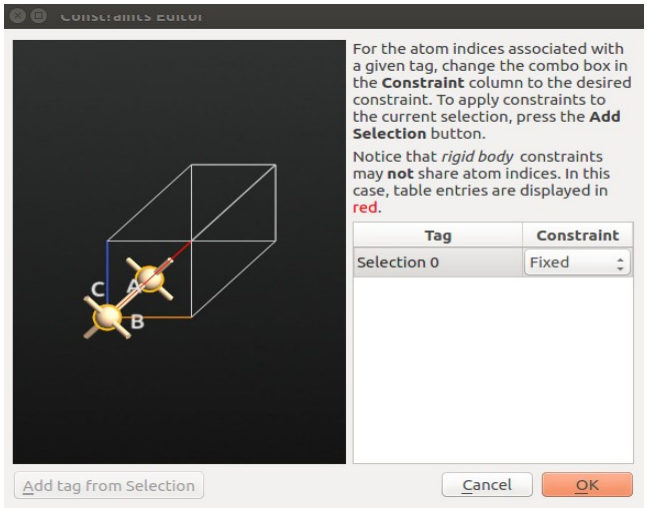
- *Exchange and correlation*: PBES
- *Basis set*: SingleZetaPolarized
- K-point sampling: 4x4x4
- -Shifted to Gamma
- Mesh cut-off: 50 Hartree

❖OptimizeGeometry

- Untick “Constrain Lattice Vectors”
- Click “Atomic Constraint Editor” and fix the fractional coordinates of Si atoms.







Creating the B-doped crystal

Builder

❖In the LabFloor, select the optimized structure (with *gID001*) and drop it onto the **Builder**.

❖Make the cell orthogonal:

- click on *Bulk Tools → Supercell* ;
- click on “Conventional” and then “Transform”.

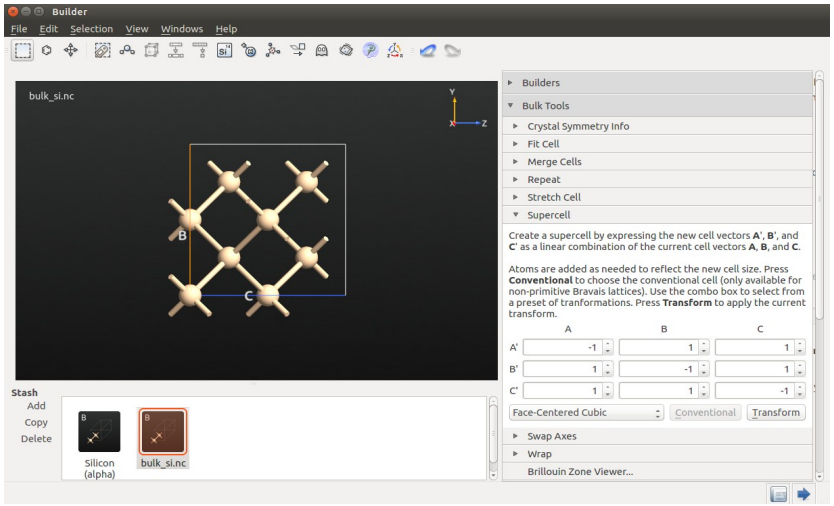
❖Make a 64 atom cell: click on *Bulk Tools → Repeat* and increase A, B and C to 2 before clicking “Apply”.

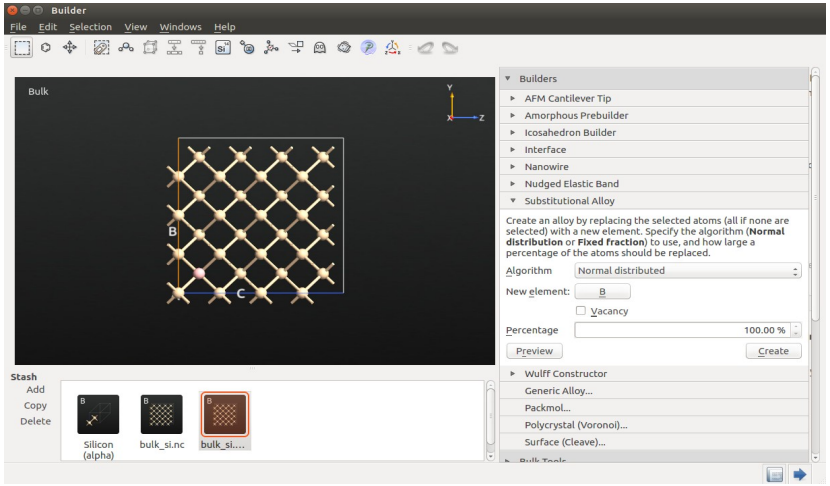
❖Select the Si atom you would like to replace with B.

❖Go to Builders → Substitutional Alloy

- *New element*: B
- *Percentage*: 100%

❖Send the structure to the **Script Generator**

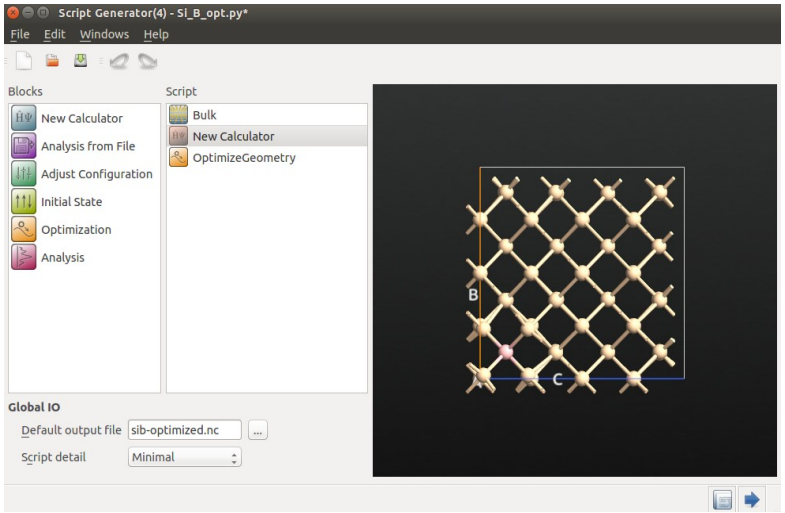


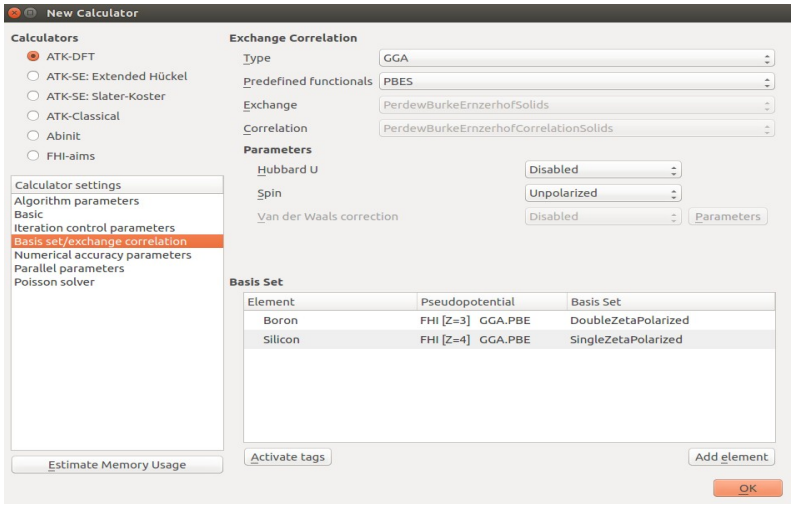


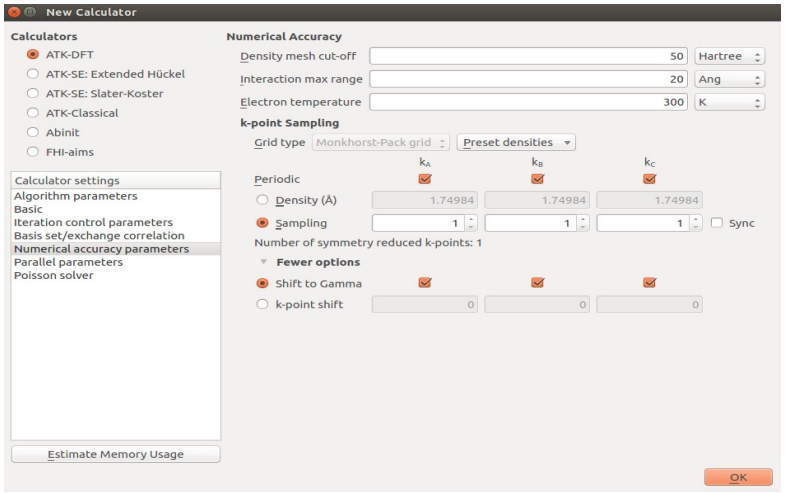
Setup the script

Script Generator

- Add the following blocks:
- New Calculator:
  - *K-point sampling*: 1x1x1
  - *Basis set:* SingleZetaPolarized for Si and DoubleZetaPolarized for B.
  - *Remaining settings*: The same as before.
- OptimizeGeometry:
  - Keep the default settings.
- Send the script to the **Job Manager** and run it.







Running the AKMC simulation

Script

❖Script provided(*sib-with-akmc.py*). Download it and save it in the project folder.

1. SetVerbosity: Removes the logging of the DFT calculation from output and log-files.
2. The optimized configuration is read from the saved file, *sib-optimized.nc*.
3. Calculator setup is identical to the one used for the structural relaxation, except for the addition the *ParallelParameters,* which is used to ensure that just one process will be used for each saddle search.

❖*Note: It might make sense to use more than one process per saddle search for a bigger system, but the optimal distribution depends on your system and the setup and queuing rules of your supercomputer.*

```
setVerbosity(CALCULATOR_UPDATE=False)
setVerbosity(PROGRESS_BARS=False)

# ----------------------------------------------------------------
# Bulk Configuration
# ----------------------------------------------------------------

# Read in configuration
bulk_configuration = nlread('sib-optimized.nc', BulkConfiguration)[-1]

# ----------------------------------------------------------------
# Calculator
# ----------------------------------------------------------------
#----------------------------------------
# Basis Set
#----------------------------------------
basis_set = [
    GGABasis.Boron_DoubleZetaPolarized,
    GGABasis.Silicon_SingleZetaPolarized,
    ]

#----------------------------------------
# Exchange-Correlation
#----------------------------------------
exchange_correlation = GGA.PBES

k_point_sampling = MonkhorstPackGrid(
    na=1,
    nb=1,
    nc=1,
    shift_to_gamma=[True, True, True],
    )
numerical_accuracy_parameters = NumericalAccuracyParameters(
    k_point_sampling=k_point_sampling,
    density_mesh_cutoff=50.0*Hartree
    )

parallel_parameters = ParallelParameters(
    processes_per_saddle_search=1,
    )

calculator = LCAOCalculator(
    basis_set=basis_set,
    exchange_correlation=exchange_correlation,
    numerical_accuracy_parameters=numerical_accuracy_parameters,
    parallel_parameters=parallel_parameters,
    )

bulk_configuration.setCalculator(calculator)
nlprint(bulk_configuration)
bulk_configuration.update()
```

Running the AKMC simulation

Script

❖Final part of the script: The AKMC simulation

4.- First two blocks: check if a previous AKMC simulation has already been run, in order to reuse that information, or initialize new objects if no previous results are present.

5.- SaddleSearchParameters: We restrict the NEB calculation to more than 5 images (for more complicated systems, more than 5 images might be needed).

6.- The actual AKMC simulation: Starts with the number of saddle searches as a parameter. As AKMC is a stochastic method, it is important to run enough saddle searches to ensure that the reaction space is adequately sampled. This script is prepared for making additional runs.

6.1.- Assumed_prefactor: For solid state systems it is often a good approximation to assume the prefactor, which is very expensive to calculate.

```
# ------------------------------------------------------------
# AKMC
# ------------------------------------------------------------

# Reusing existing MarkovChain object if it already exists, otherwise a new one is created
if os.path.isfile('akmc_markov_chain.nc'):
    markov_chain = nlread('akmc_markov_chain.nc')[0]
else:
    markov_chain = MarkovChain(bulk_configuration, TotalEnergy(bulk_configuration).evaluate())

# Reusing existing KMC object if it already exists, otherwise a new one is created
if os.path.isfile('akmc_kmc.nc'):
    kmc = nlread('akmc_kmc.nc')[0]
else:
    kmc = None

# Modify the default maximum number of NEB images to 5.
saddle_search_parameters = SaddleSearchParameters(max_neb_images=5)

# Setup the AKMC simulation.
akmc = AdaptiveKineticMonteCarlo(markov_chain,
                                 calculator=calculator,
                                 kmc_temperature=300*Kelvin,
                                 md_temperature=3000*Kelvin,
                                 saddle_search_parameters=saddle_search_parameters,
                                 constraints=[0],
                                 confidence=0.99,
                                 assumed_prefactor=1e13/Second,
                                 write_searches=False,
                                 write_kmc=True,
                                 write_markov_chain=True,
                                 write_log=True)

# Run 100 saddle searches.
akmc.run(47)
```

For more information on the AKMC parameters see the tutorial “Modeling Vacancy Diffusion in Si0.5 Ge0.5 with AKMC”.