CASM Tutorial

Brian Puchala[2],
John C. Thomas[1]
Anton Van der Ven[1]

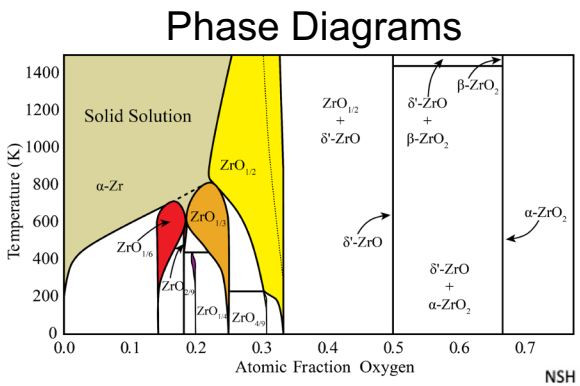
PRISMS Workshop
August 4-5, 2025

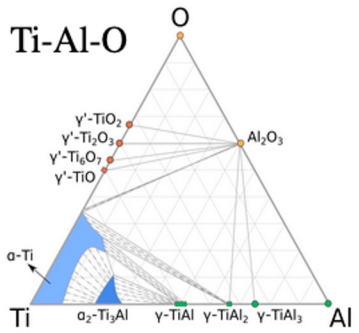
[1]Materials Department, University of California Santa Barbara
[2]Department of Materials Science & Engineering, University of Michigan

First-Principles Statistical Mechanics

Thermodynamics

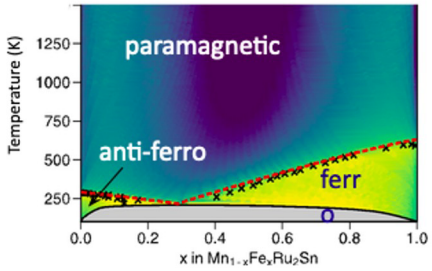
Phase Diagrams





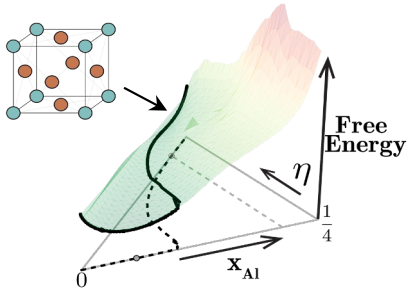
NSH Gunda, A. Van der Ven, Acta Materialia 191, 149 (2020)

Chemo-Magnetic phase stability



Decolvenaere, Levin, Seshadri, Van der Ven, Physical Review Materials 3, 104411 (2019)

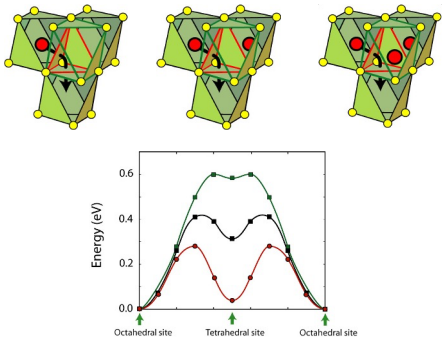
Free Energies



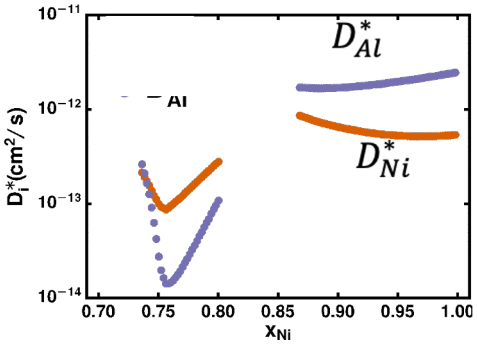
Disordered solid solution

Kinetics

Diffusion in disordered alloys

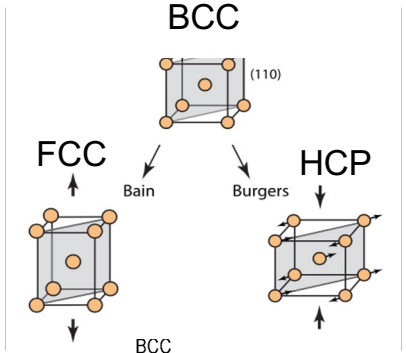


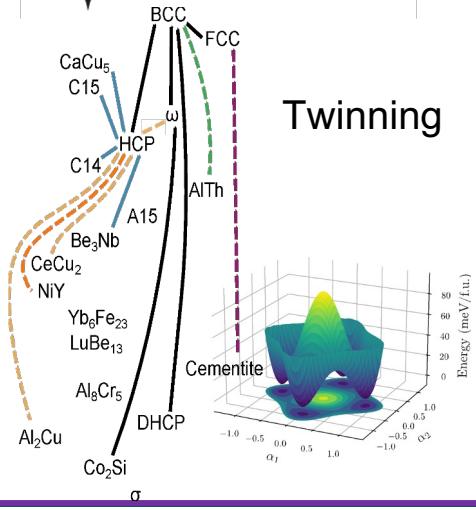
Diffusion Coefficients



Chemo-Mechanics

Structural transformations





Twinning

CASM

A Clusters Approach to Statistical Mechanics

Puchala, et al. Comp. Mater. Sci. 217, 111987 (2023).

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Full length article

CASM — A software package for first-principles based study of multicomponent crystalline solids

Brian Puchala [a,*,1], John C. Thomas [b,*,1], Anirudh Raju Natarajan [b], Jon Gabriel Goiri [b], Sesha Sai Behara [b], Jonas L. Kaufman [b], Anton Van der Ven [b,*]

[a] Department of Materials Science and Engineering, University of Michigan, Ann Arbor, United States of America
[b] Materials Department, University of California, Santa Barbara, United States of America

CASM v2 Overview:

https://prisms-center.github.io/CASMcode_pydocs/overview/latest/

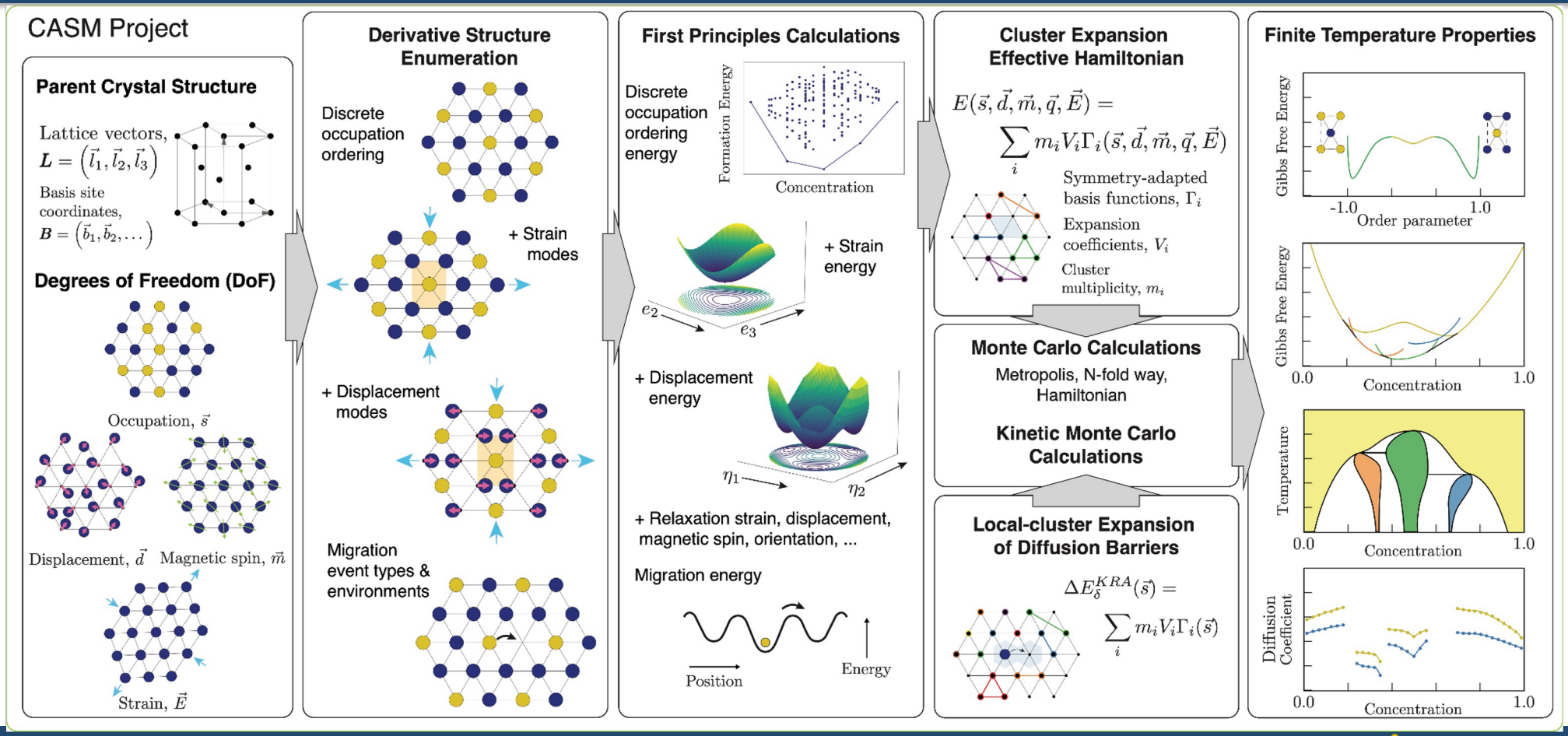
U.S. DEPARTMENT OF ENERGY | Office of Science

PRISMS

UCSB

UNIVERSITY OF MICHIGAN

CASM Workflow



Puchala, et al. Comp. Mater. Sci. 217, 111987 (2023).

PRISMS

CASMv2: About CASM

- CASM v2 consists of several Python packages
  - Easier to learn, to use, and to develop
- Python packages (libcasm-) with C++ extensions for core features
  - Fast symmetry application for enumeration
  - Efficient Monte Carlo calculations
  - Python interface using pybind11
- Pure python packages (casm-) for ease of use
  - casm-bset: Generate symmetry-adapted functions
  - casm-flow: (coming soon) Manage Monte Carlo calculations
  - casm-tools: CLI tools and helper methods
  - casm-project: Interact with project data **← Users start here**

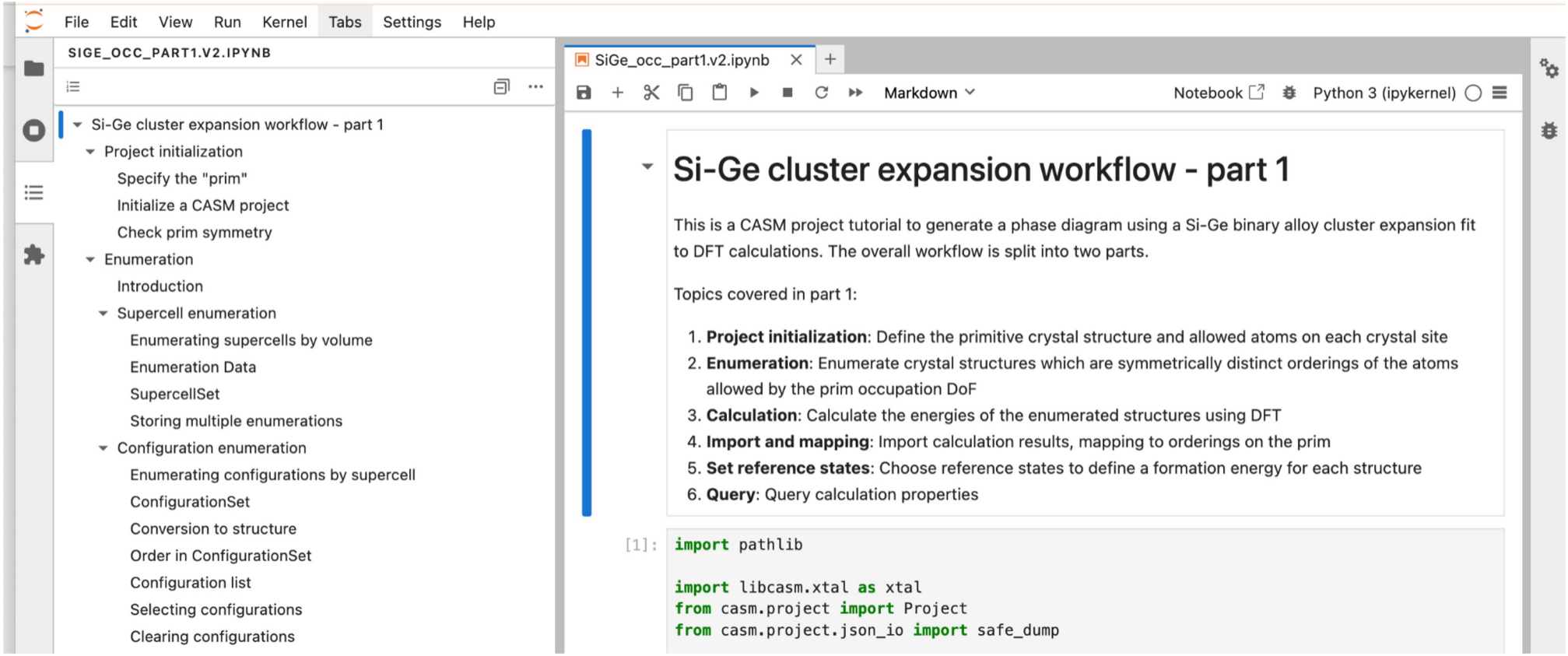
CASMv2: Installation for Si-Ge tutorials

```
# ~~ Install ~~
# load python3.11 and gcc/11.3.0
module load python/3.11
module load gcc/11.3.0

# install casm-project
git clone https://github.com/prisms-center/CASMcode_project.git
cd CASMcode_project
pip install .
pip install -r notebooks_requirements.txt

# ~~ Setup environment and test ~~
export PATH=/home/<username>/.local/bin:$PATH
export CASM_PREFIX=$(python -m libcasm.casmglobal --prefix)
casm-calc -h
python -m casm.bset --test

# ~~ Launch Jupyter ~~
jupyter lab
```

CASMv2: Getting Started

- Run Si-Ge cluster expansion tutorial notebooks in CASMcode_project:
  - notebooks/SiGe_occ_part1.v2.ipynb
    - CASM project initialization
    - Configuration enumeration
    - Calculation
    - Import and mapping
  - notebooks/SiGe_occ_part2.v2.ipynb
    - Cluster expansion construction & fitting
    - Monte Carlo calculation

CASMv2: Tutorials

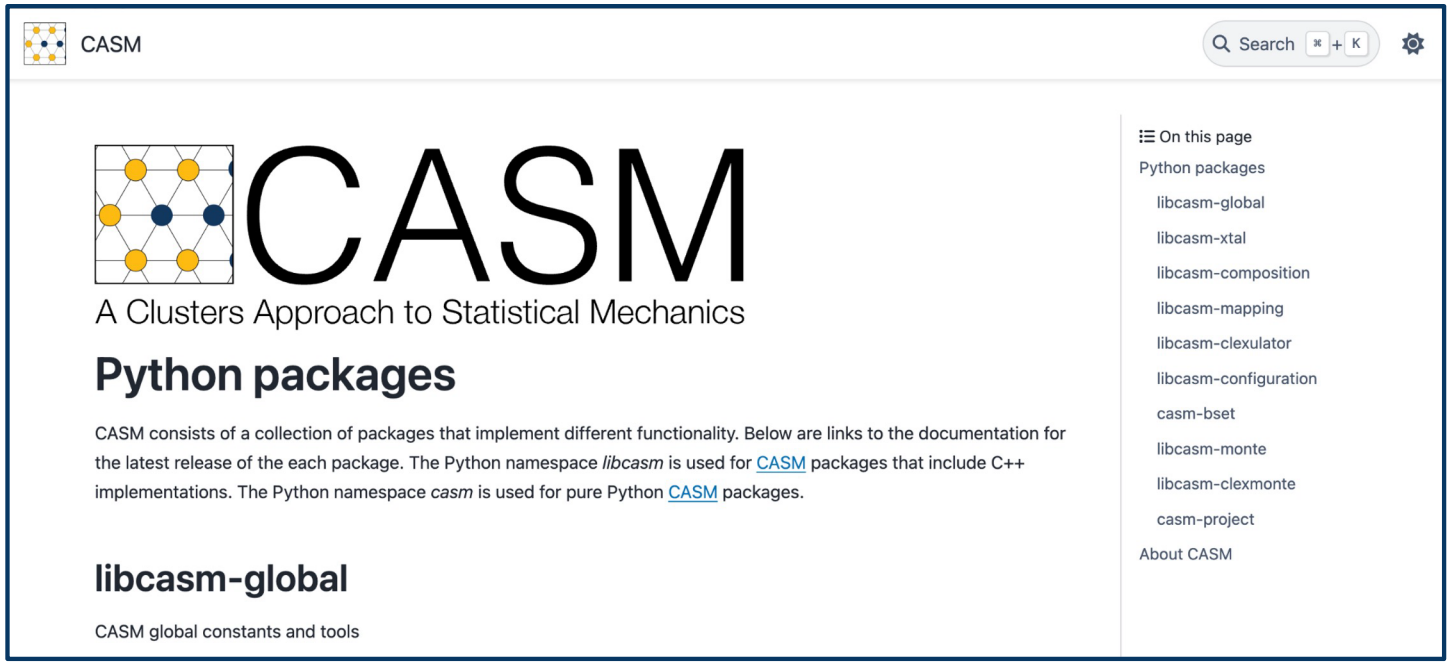


PRISMS

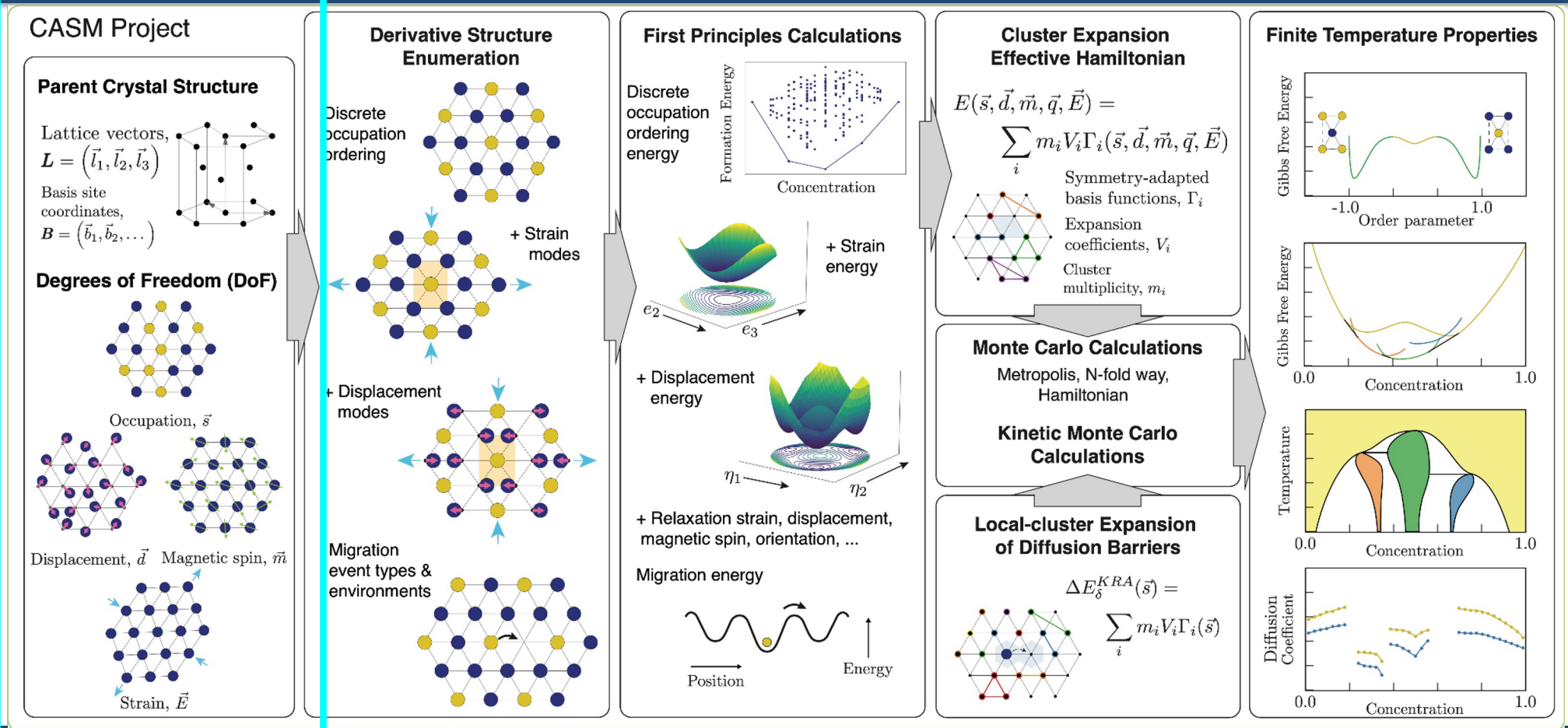
CASMv2: Docs

CASMv2 Python package docs:

- https://prisms-center.github.io/CASMcode_pydocs/overview/latest/



Project initialization

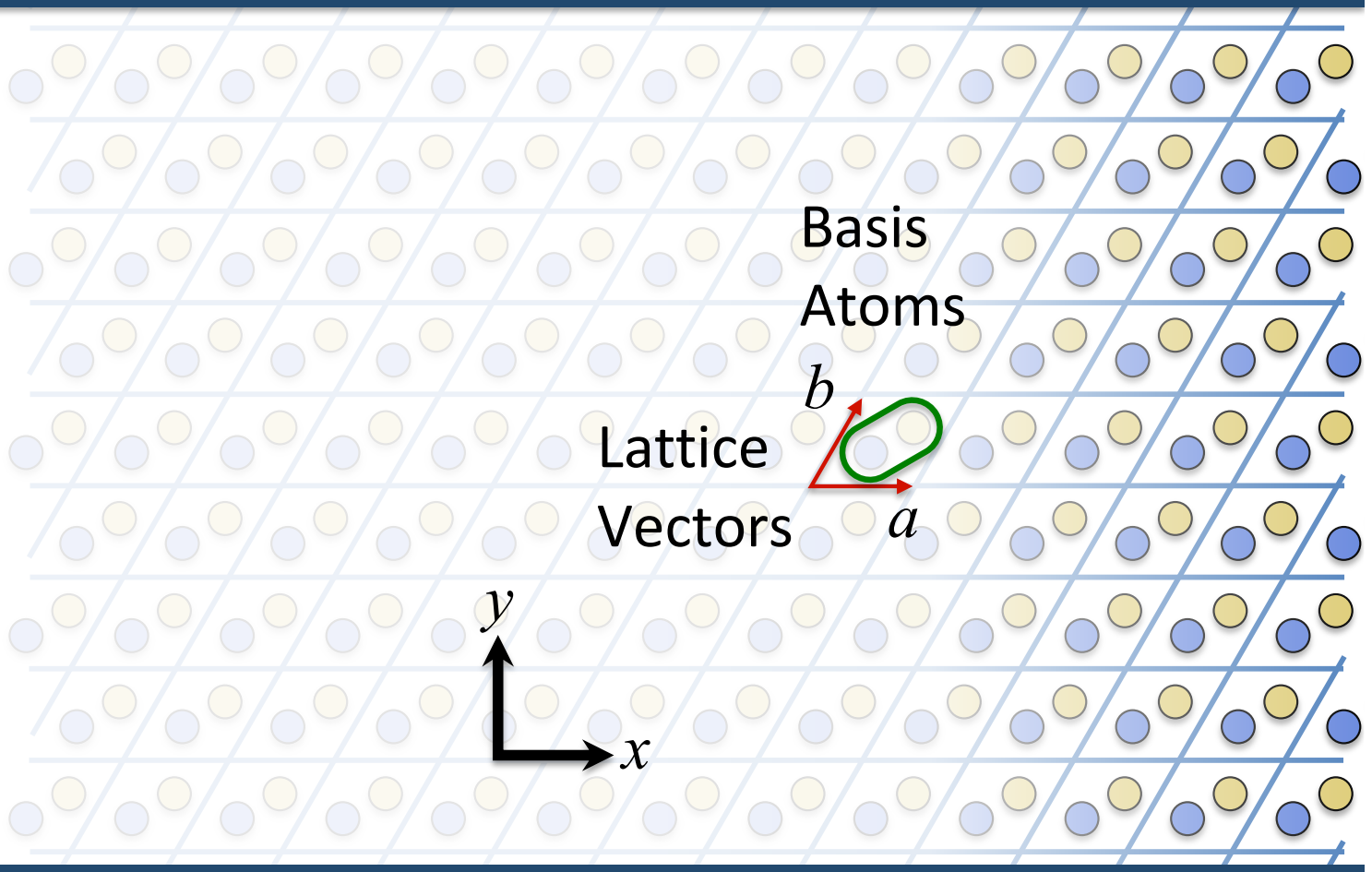


Puchala, et al. Comp. Mater. Sci. 217, 111987 (2023).

PRISMS

Project Initialization

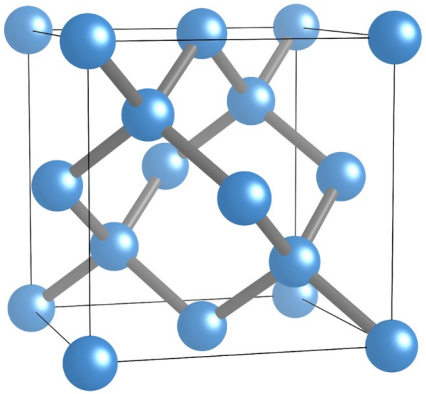
Crystal structure

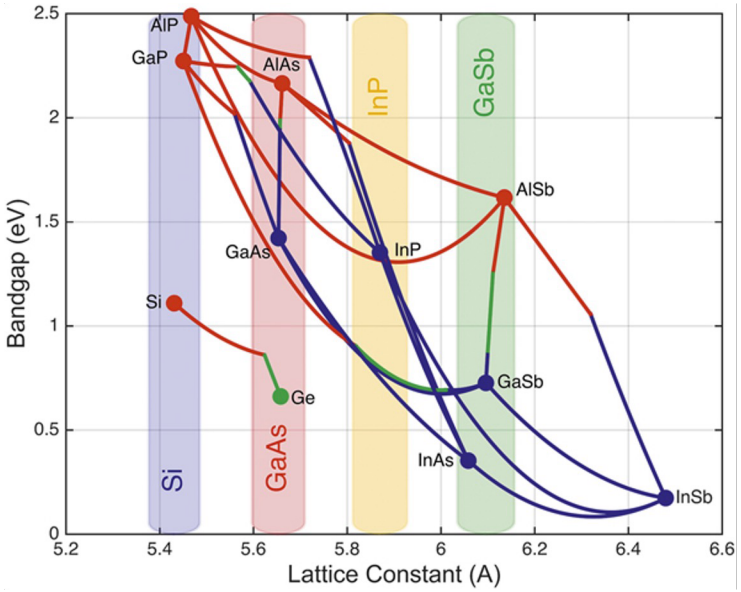


Tutorial Project: $Si_{1-x}Ge_x$ binary system

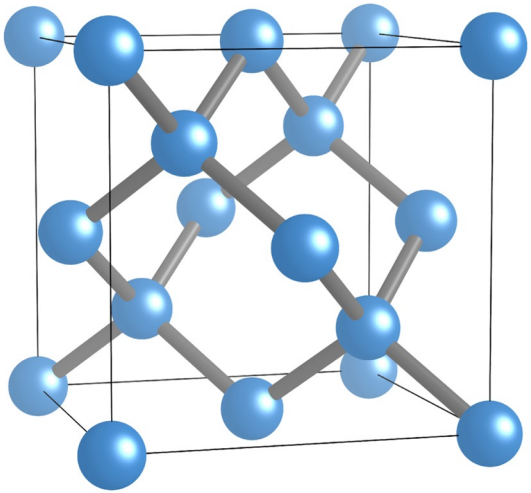
Important semiconductor alloy with tunable bandgap and lattice parameter.

Model alloying of Si and Ge on sites of diamond cubic crystal

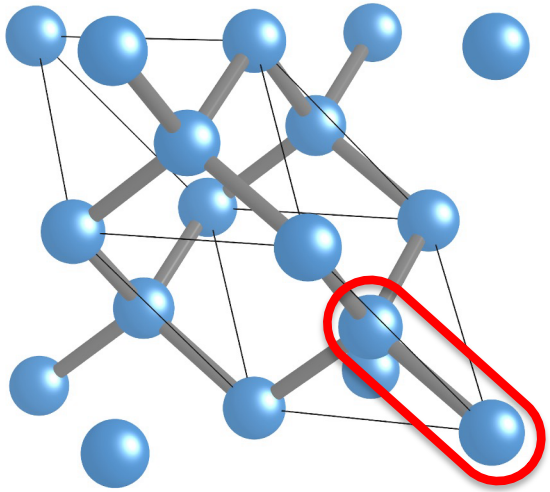




Tutorial Project: $Si_{1-x}Ge_x$ binary system



Conventional cubic cell



Primitive cell

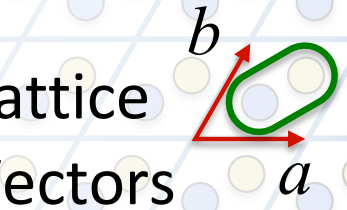
Project Initialization

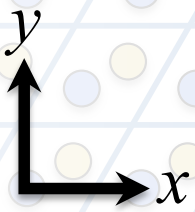
prim.json:

```
{
  "title" : "SiGe",
  "description" : "Si/Ge on diamond cubic crystal",
  "coordinate_mode" : "Fractional",
  "lattice_vectors" : [
    [ 0.0000, 2.8000, 2.8000 ],
    [ 2.8000, 0.0000, 2.8000 ],
    [ 2.8000, 2.8000, 0.0000 ]
  ],
  "basis" : [
    {
      "coordinate" : [ 0.0000, 0.0000, 0.0000 ],
      "occupant_dof" : [ "Si", "Ge" ]
    },
    {
      "coordinate" : [ 0.2500, 0.2500, 0.2500 ],
      "occupant_dof" : [ "Si", "Ge" ]
    }
  ],
}
```

Basis Atoms

Lattice Vectors





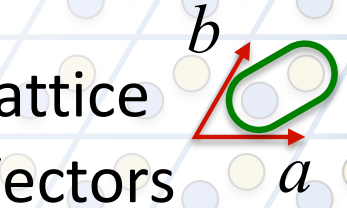
Project Initialization

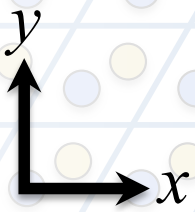
prim.json:

```
{
  "title" : "SiGe",
  "description" : "Si/Ge on diamond cubic crystal",
  "coordinate_mode" : "Fractional",
  "lattice_vectors" : [
    [ 0.0000, 2.8000, 2.8000 ],
    [ 2.8000, 0.0000, 2.8000 ],
    [ 2.8000, 2.8000, 0.0000 ]
  ],
  "basis" : [
    {
      "coordinate" : [ 0.0000, 0.0000, 0.0000 ],
      "occupant_dof" : [ "Si", "Ge" ]
    },
    {
      "coordinate" : [ 0.2500, 0.2500, 0.2500 ],
      "occupant_dof" : [ "Si", "Ge" ]
    }
  ],
}
```

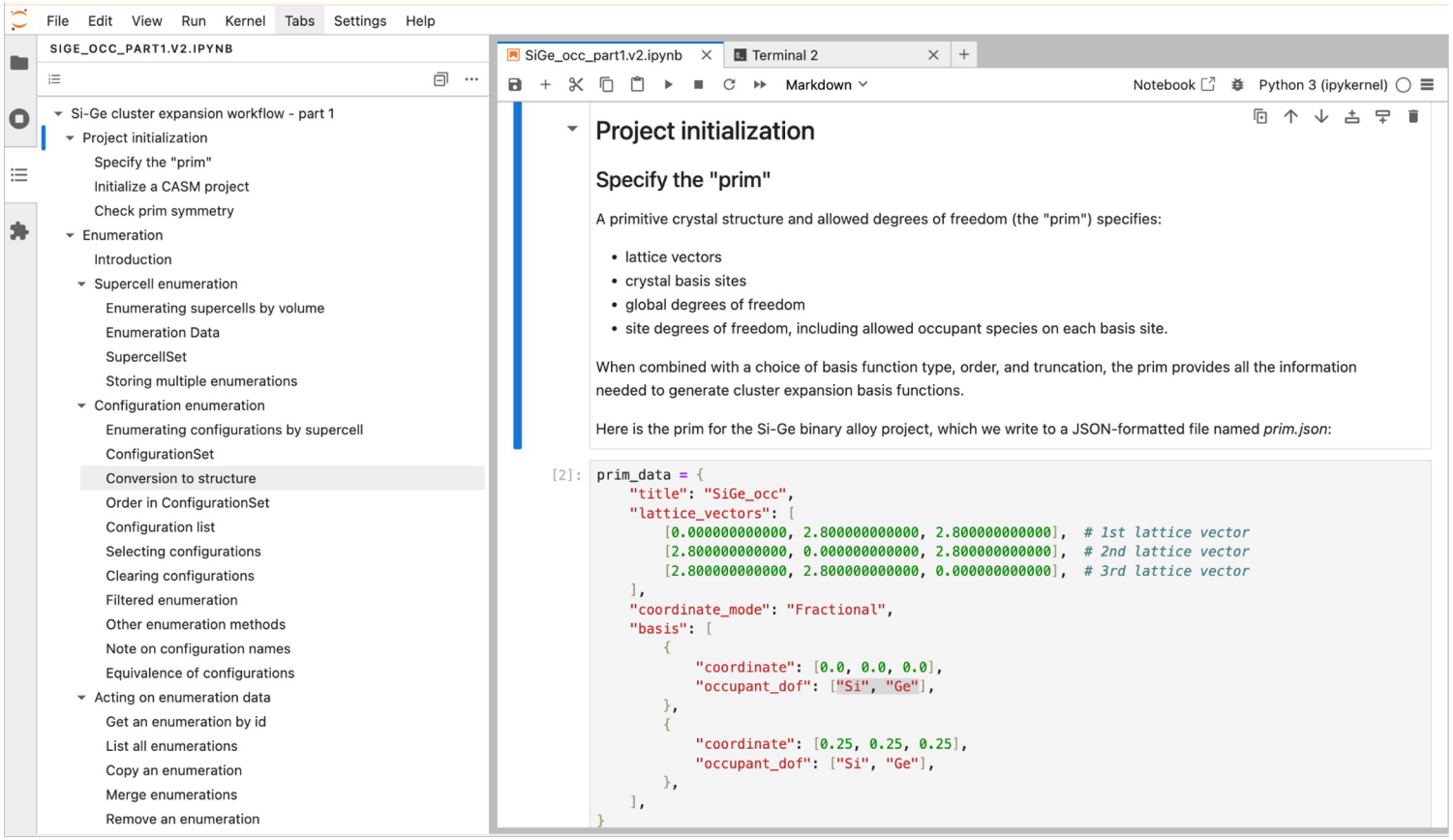
Basis Atoms

Lattice Vectors

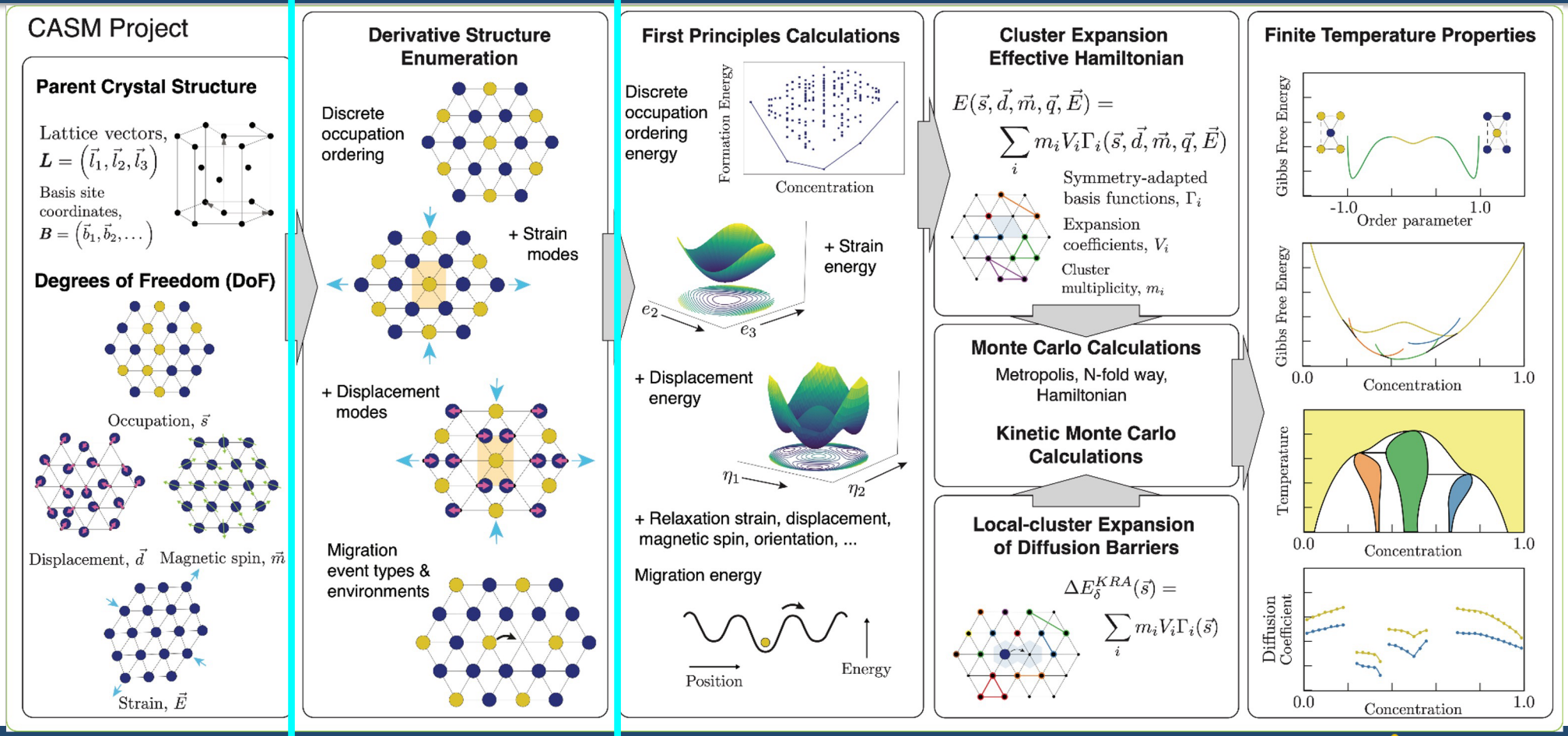




Project Initialization



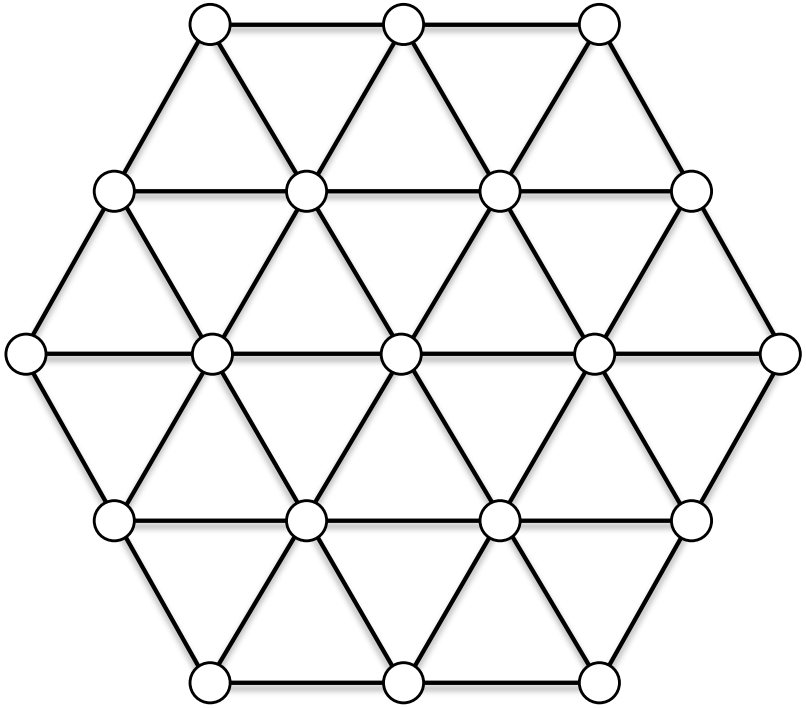
Enumeration



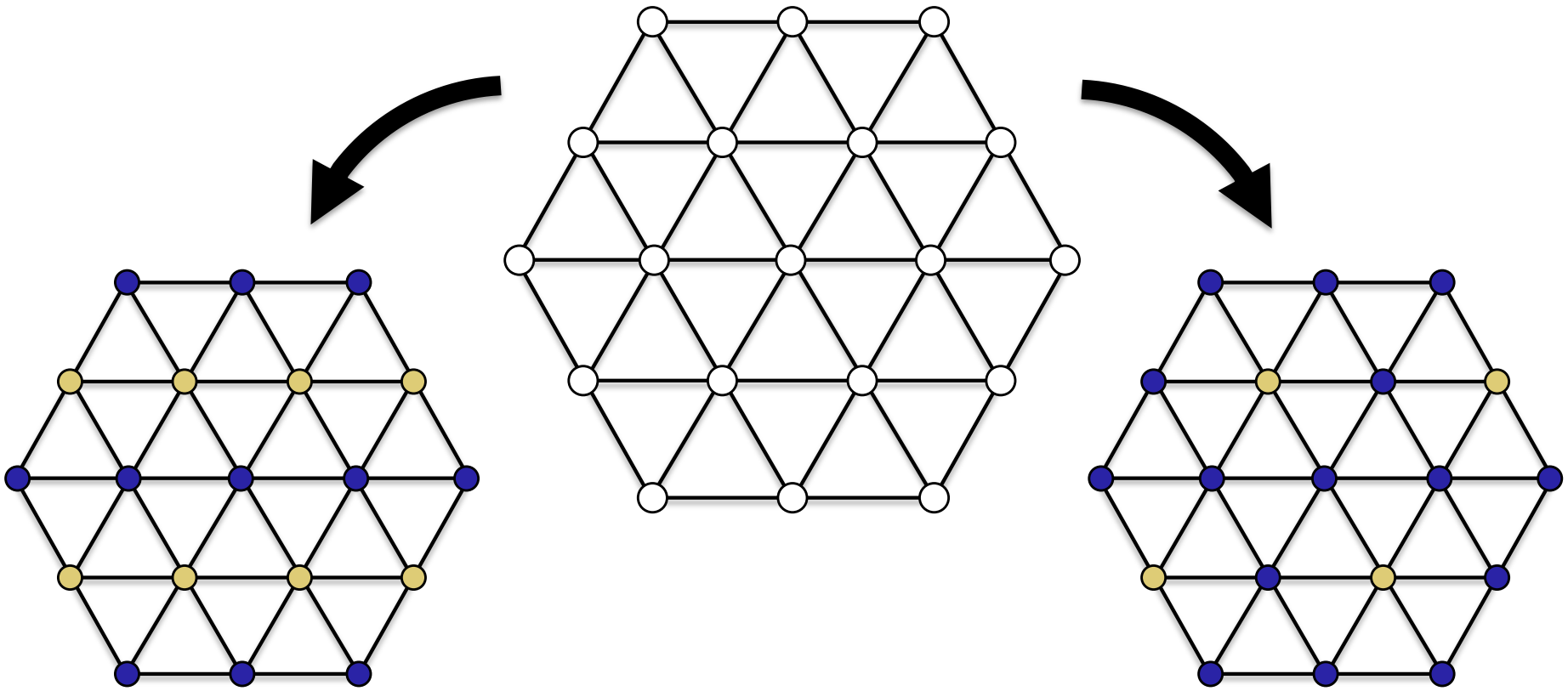
Puchala, et al. Comp. Mater. Sci. 217, 111987 (2023).

PRISMS

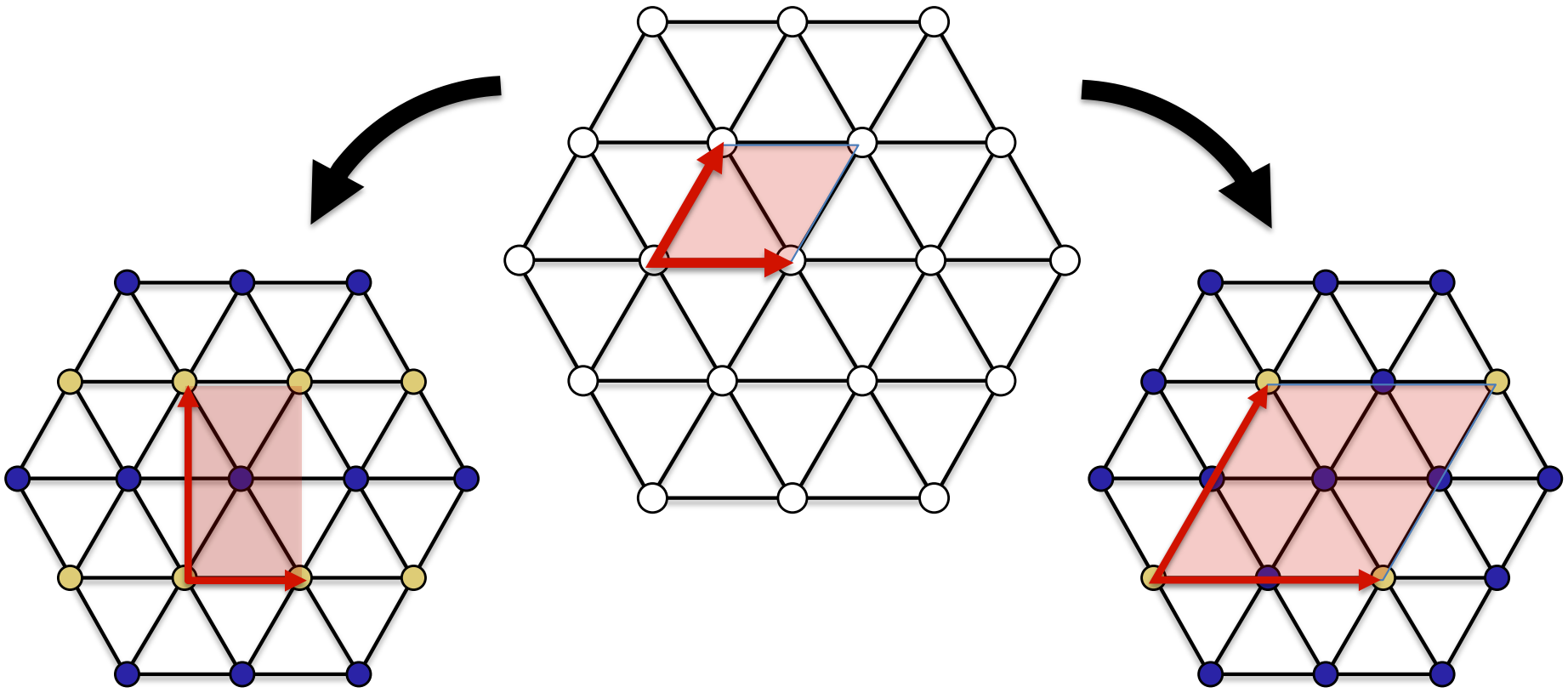
Crystal configurations



Crystal configurations



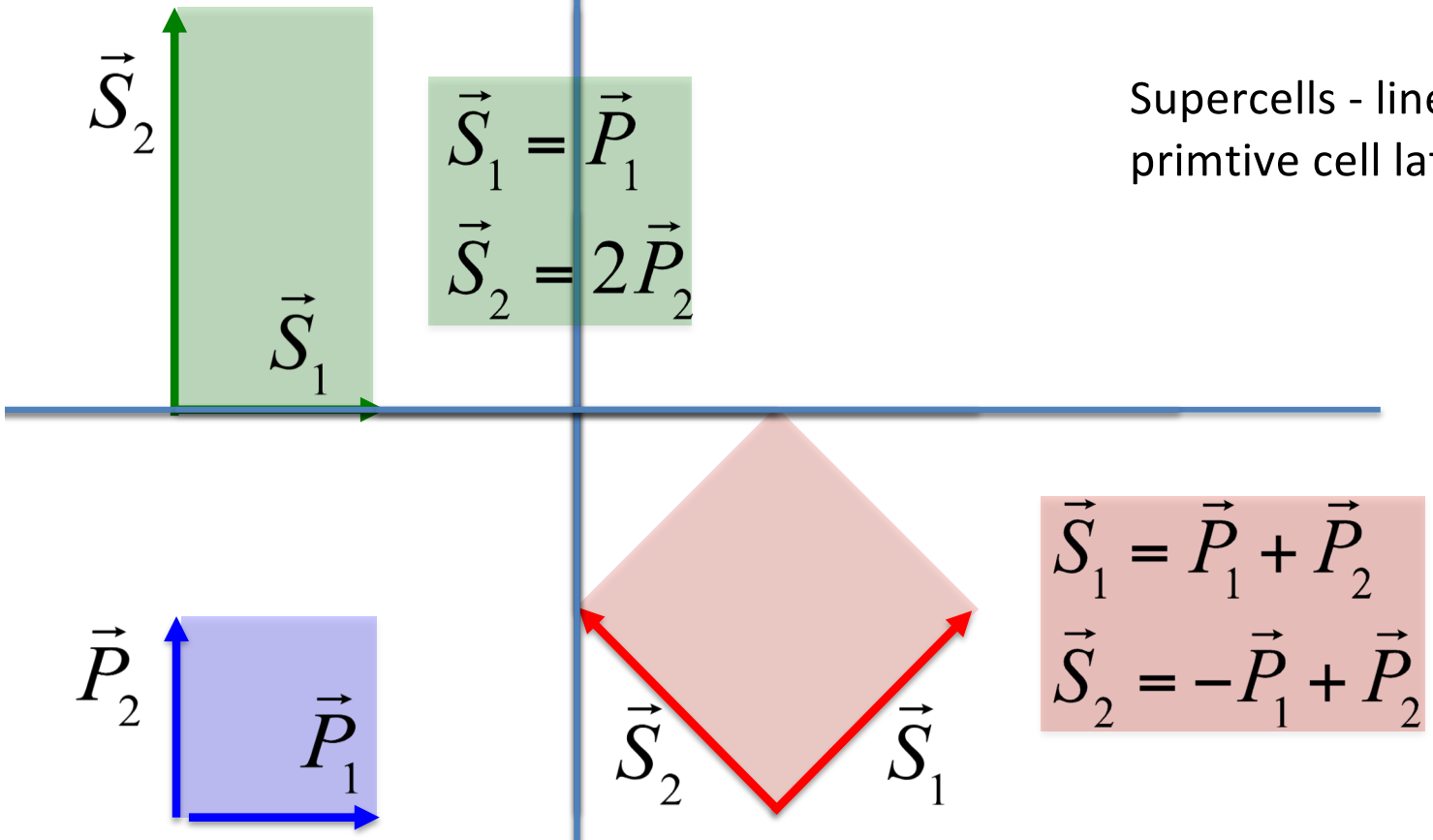
Crystal configurations



Vol 2 Supercell

Vol 4 Supercell

Unique supercell enumeration

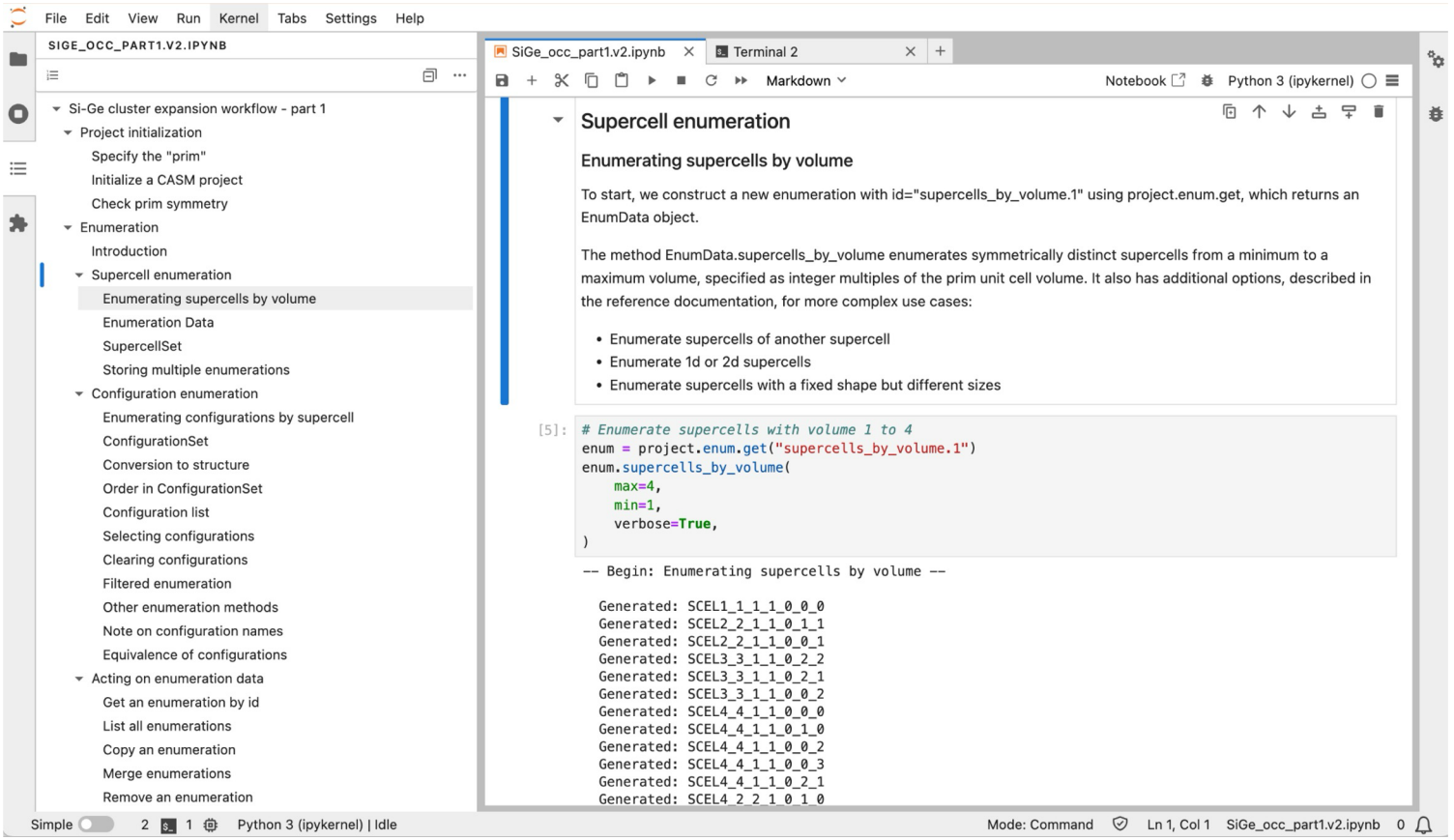


$$\vec{S}_1 = \vec{P}_1$$
$$\vec{S}_2 = 2\vec{P}_2$$

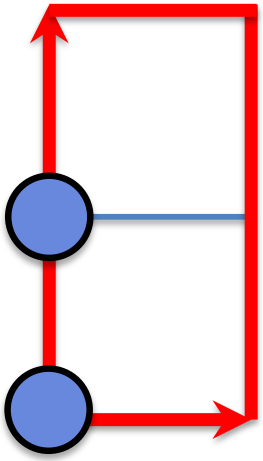
$$\vec{S}_1 = \vec{P}_1 + \vec{P}_2$$
$$\vec{S}_2 = -\vec{P}_1 + \vec{P}_2$$

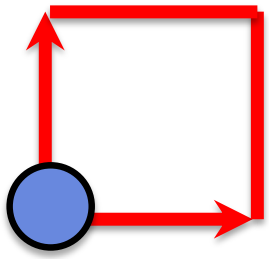
Supercells - linear combiations of primtive cell lattice vectors

Supercell enumeration



Unique configuration enumeration



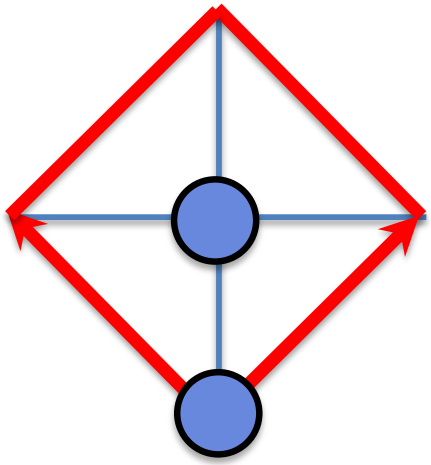


A:

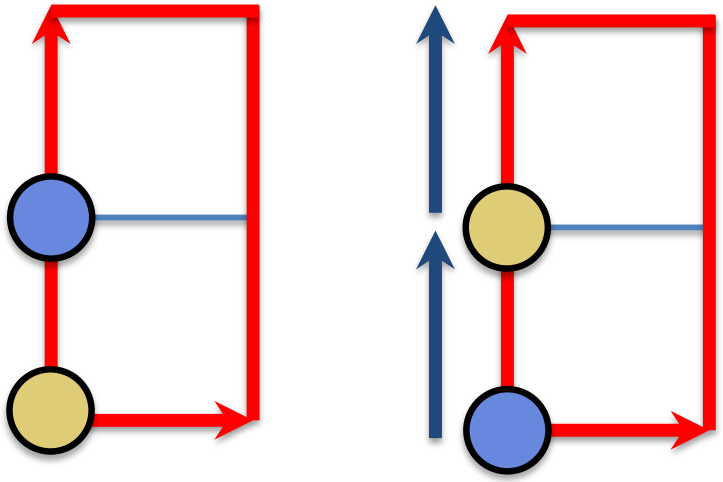
B:

Equivalent configurations:

- Supercells of the volume 1 configuration



Unique configuration enumeration



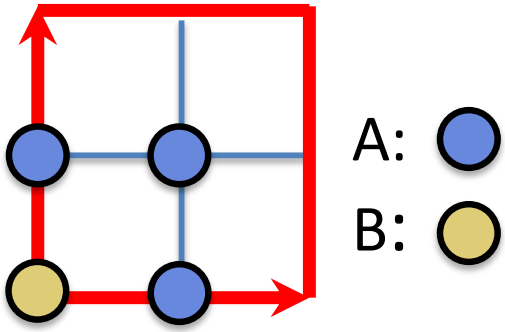
A:

B:

Equivalent configurations:

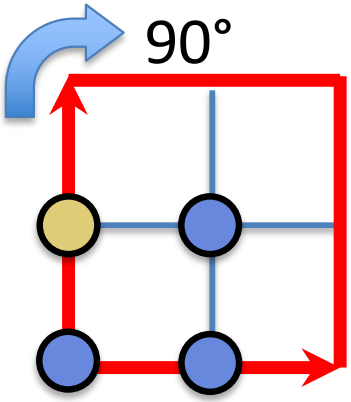
- Translation maps one configuration onto the other

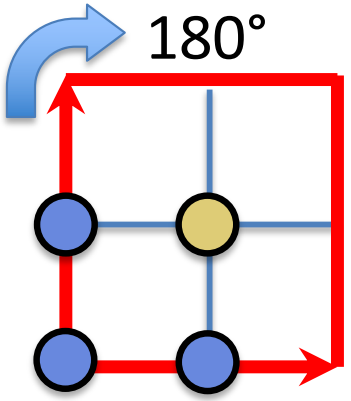
Unique configuration enumeration

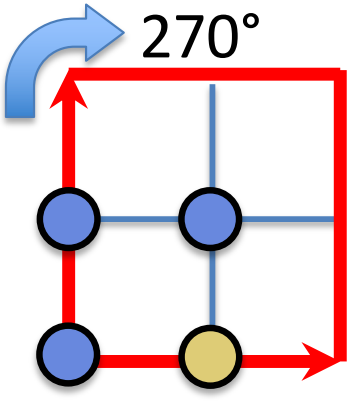


Equivalent configurations:

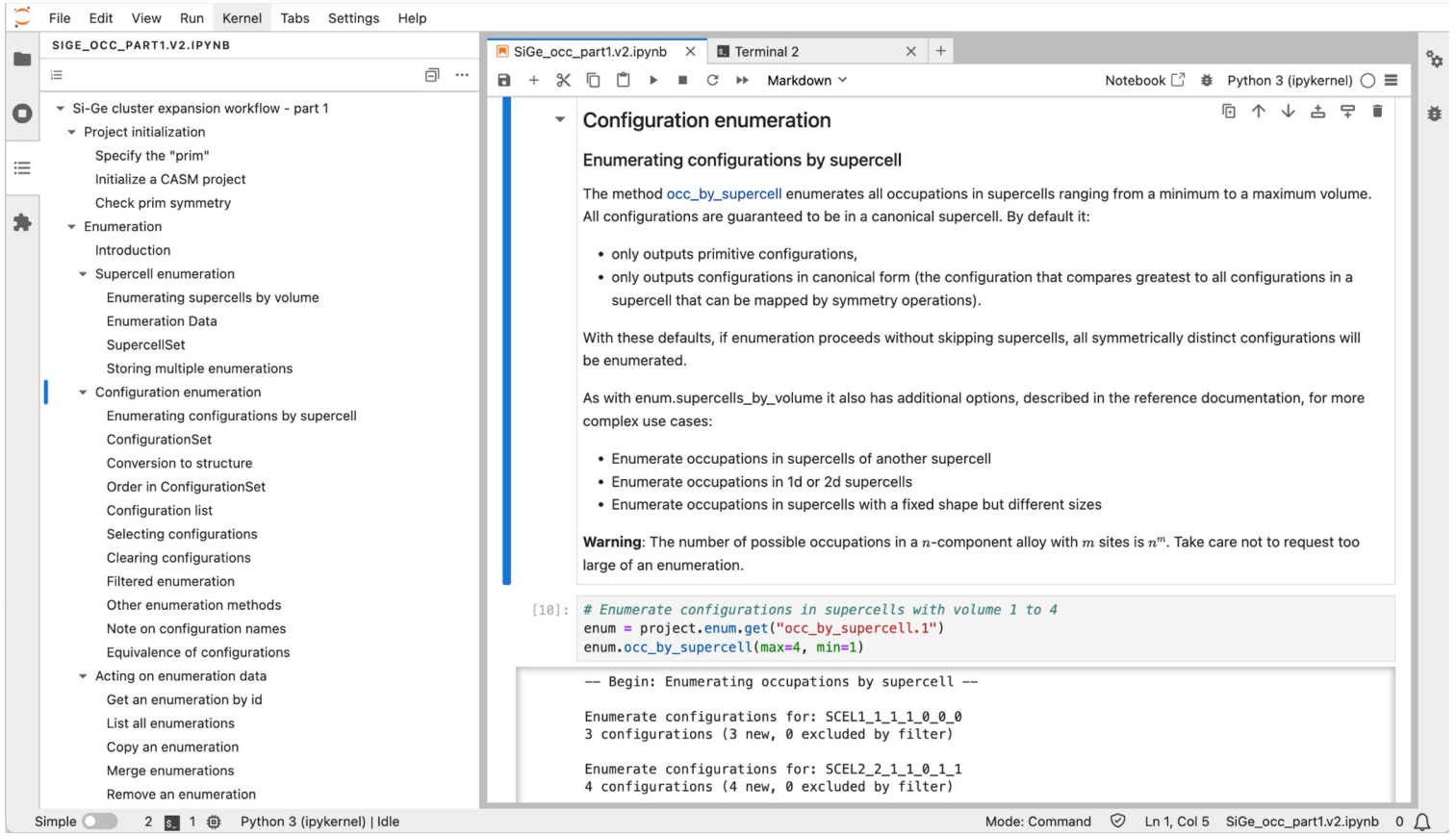
- Rotation or translation maps occupants



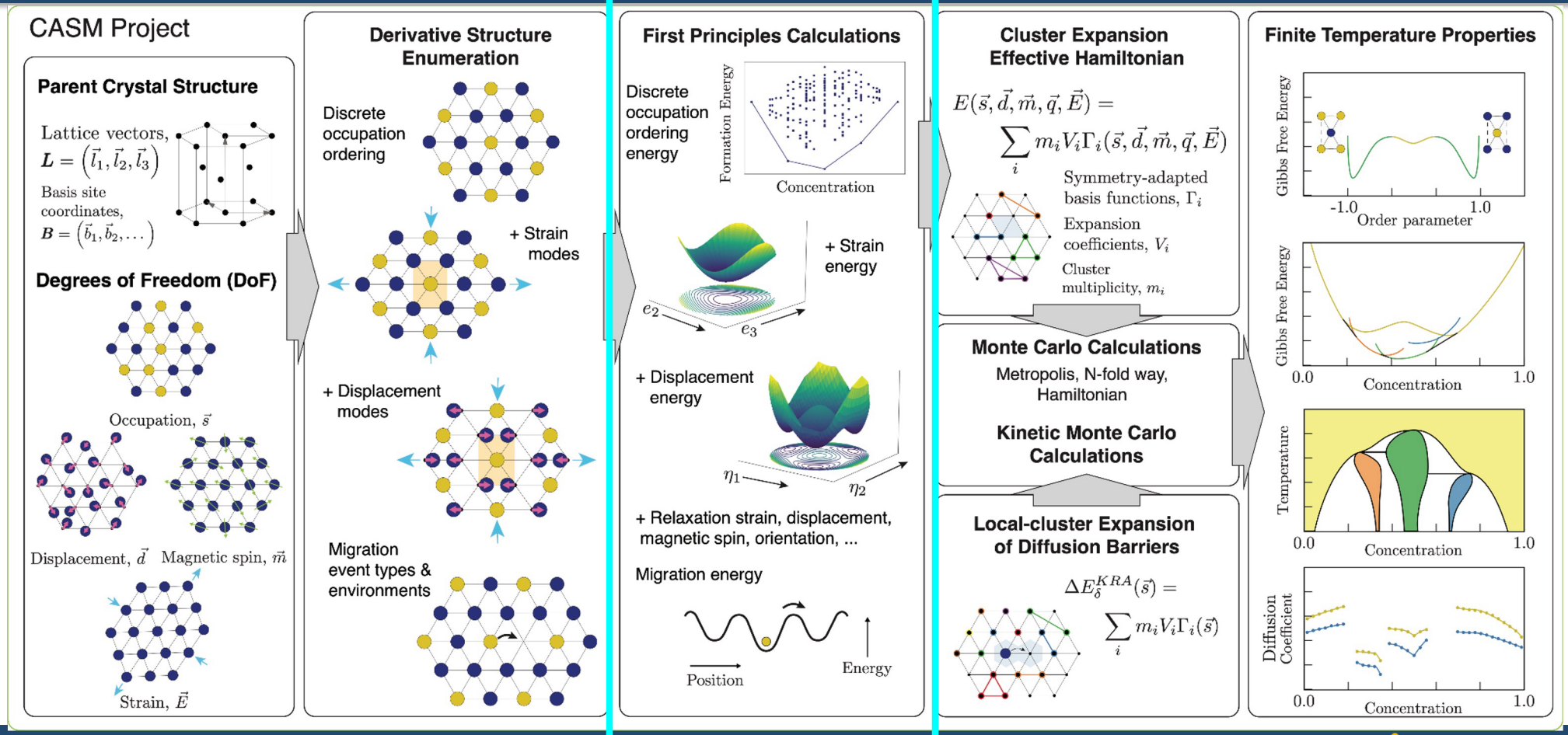




Configuration enumeration



Calculation / Import and Mapping



Puchala, et al. Comp. Mater. Sci. 217, 111987 (2023).

PRISMS

Calculation / Import and Mapping

CASM Configuration

→ CASM Structure

→ DFT Calculation Input (VASP INCAR, KPOINTS, POSCAR, POTCAR)

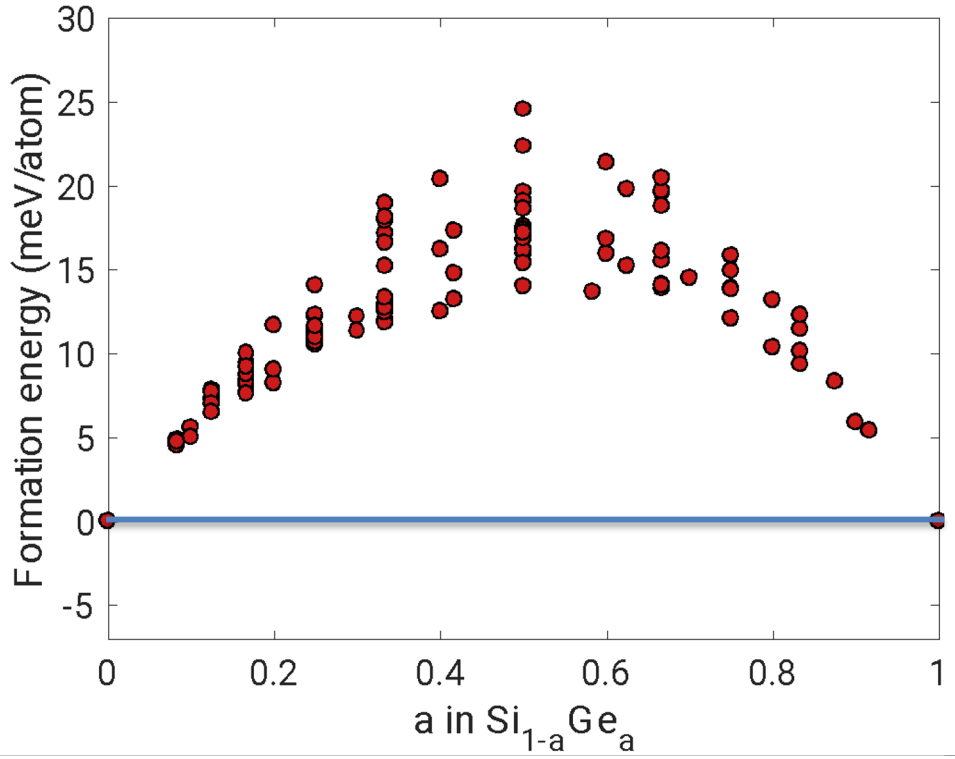
→ DFT Calculation Output (VASP OUTCAR)

→ Import → CASM Structure w/ properties

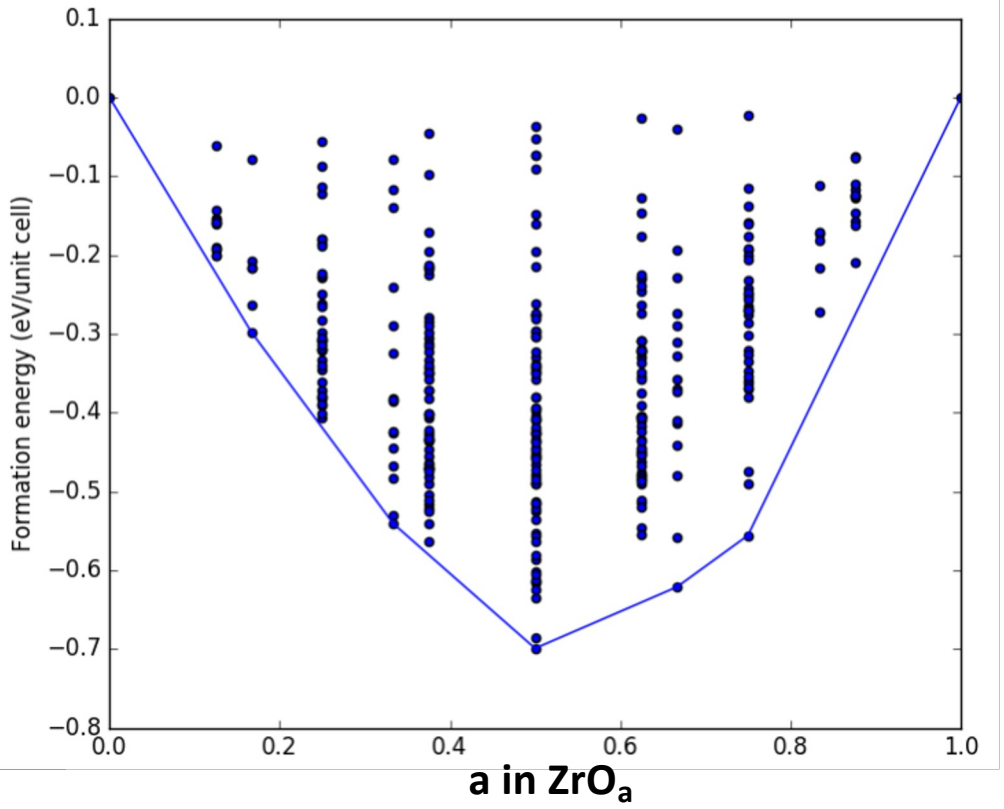
→ Mapping → CASM Configuration w/ properties

Typical DFT results

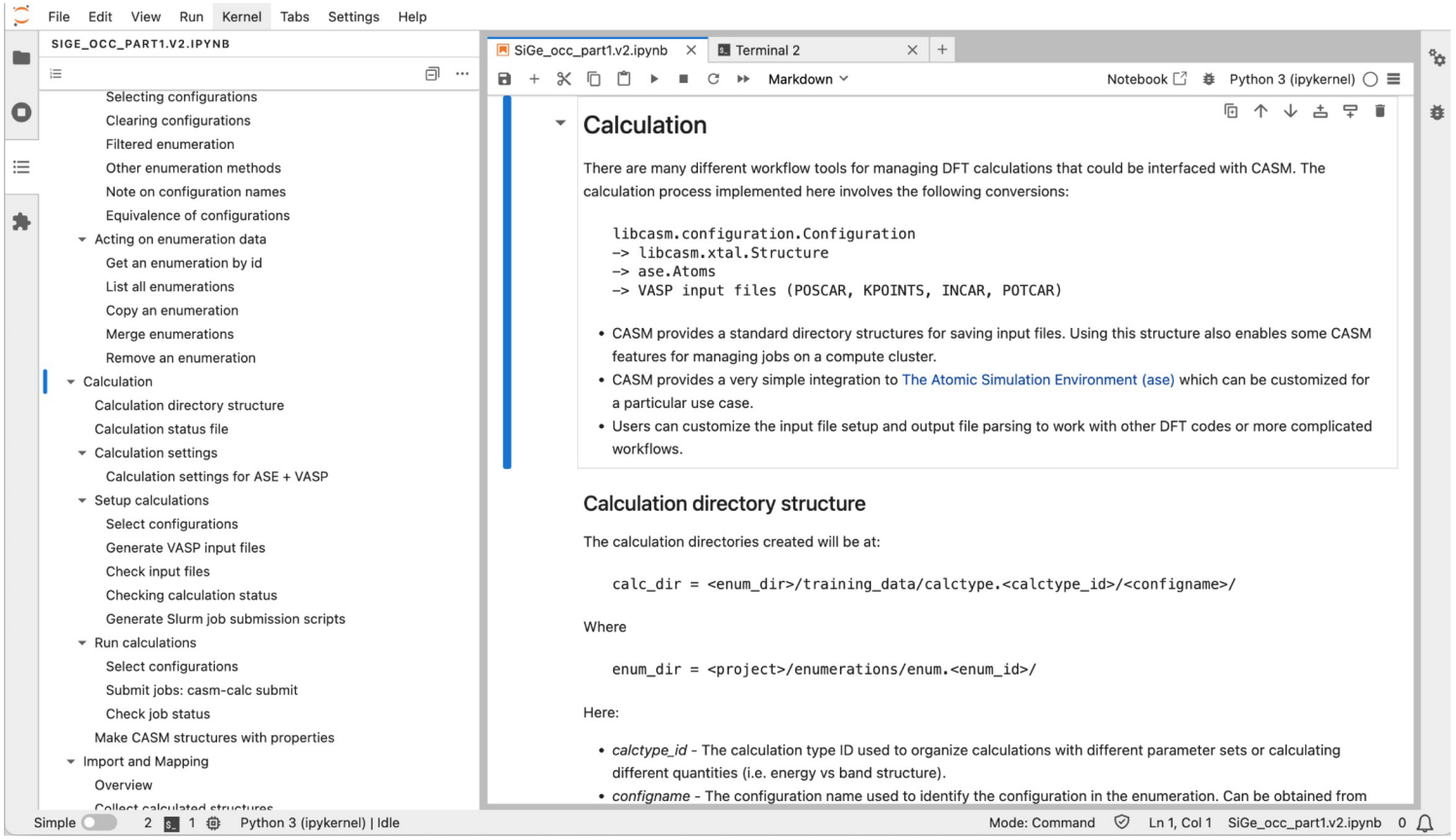
Miscibility Gap



Ordered groundstates



Calculation



PRISMS

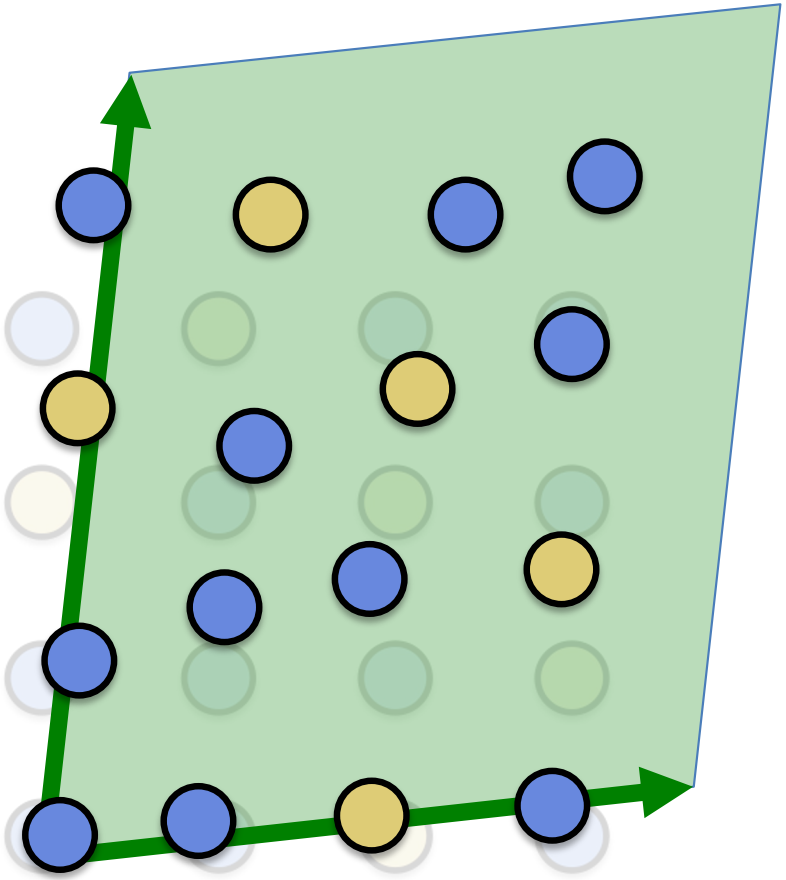
Import and Mapping

Very large relaxations may change the configuration.

There is no single valid mapping from a deformed structure and configuration.

CASM searches for mappings and scores them based on lattice deformation and atomic displacement.

Conflicts can be resolved using either min-energy or min-deformation rule

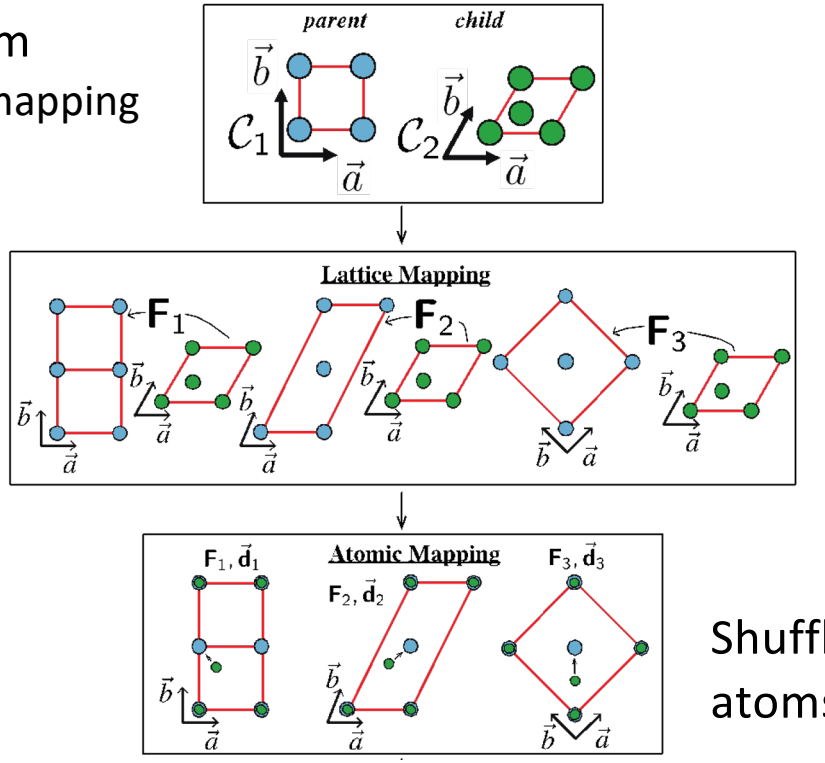


CASM
A Clusters Approach to Statistical Mechanics

CASM mapping algorithm

CASM mapping algorithm

- Available in libcasm-mapping



Make supercells of the parent

Strain the lattice vectors

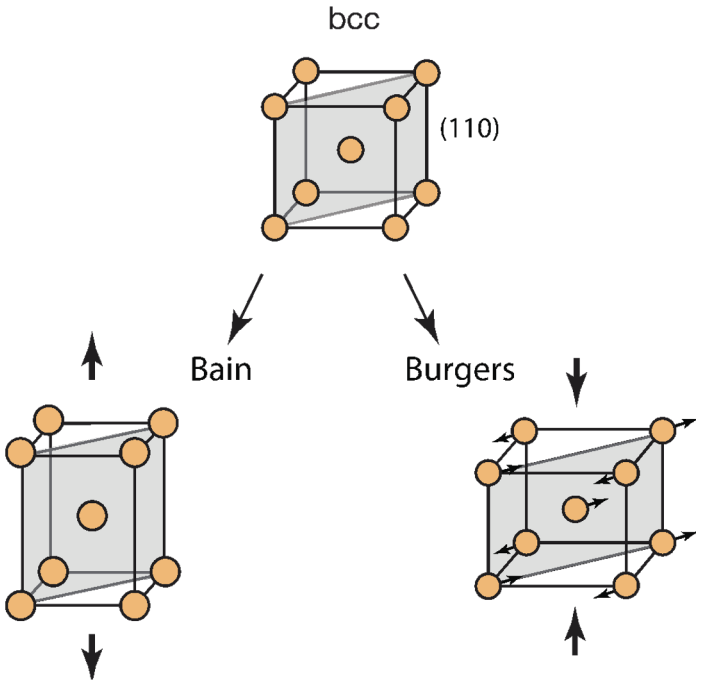
Shuffle the basis atoms

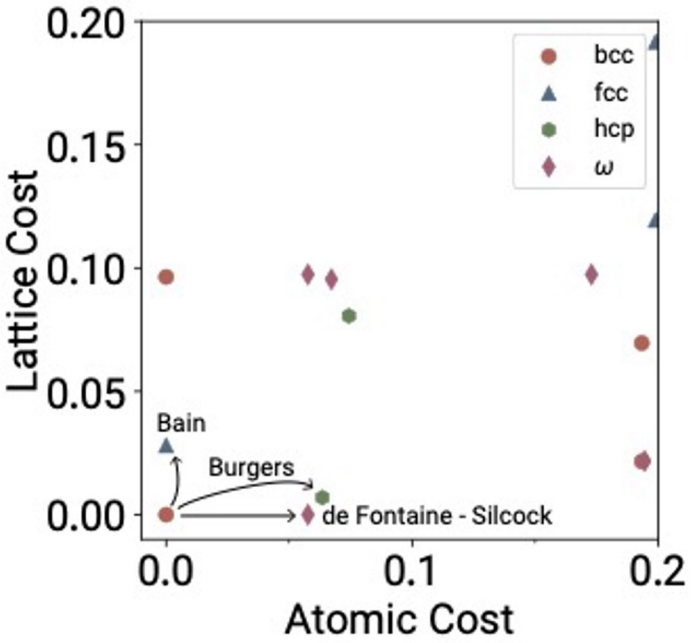
J.C. Thomas, A. R. Natarajan, A. Van der Ven, npj Comp. Mat. 7, 164, (2021).

CASM
A Clusters Approach to Statistical Mechanics

CASM mapping algorithm

Example: BCC parent

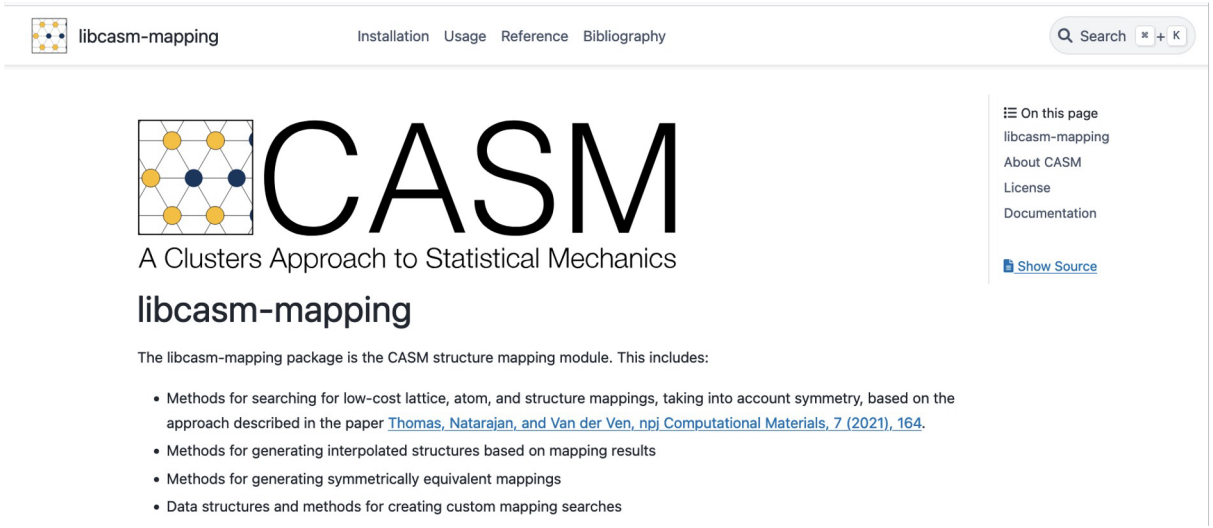




J.C. Thomas, A. R. Natarajan, A. Van der Ven, npj Comp. Mat. 7, 164, (2021).

libcasm-mapping

- Map structures
- Map lattices
  - Allow any orientation, or do not allow re-orientaion
- Map atoms to sites
- Find *k*-best mappings
- Make interpolated structures after finding a mapping
- *libcasm.mapping.mapsearch*: Build your own mapping search algorithm, with custom constraints on the search



https://prisms-center.github.io/CASMcode_pydocs/libcasm/mapping/2.0/index.html

CASM

A Clusters Approach to Statistical Mechanics

LatticeMapping

```
class libcasm.mapping.info.LatticeMapping(
    self: LatticeMapping,
    deformation_gradient: numpy.ndarray[numpy.float64[3, 3]],
    transformation_matrix_to_super: numpy.ndarray[numpy.float64[3, 3]],
    reorientation: numpy.ndarray[numpy.float64[3, 3]],
)
```

A lattice mapping has the form:

$$FL_1TN = L_2,$$

where:

- L_1 is a shape=(3,3) matrix with columns containing the reference "parent" lattice vectors
- L_2 is a shape=(3,3) matrix with columns containing the "child" lattice vectors
- F is the parent-to-child deformation gradient tensor, a shape=(3,3) matrix. F can be decomposed as:

  $$F = QU = VQ$$

  into a pure rigid transformation component, Q, (called an isometry) and either a right stretch tensor, U, or a left stretch tensor, V. U and V are both positive definite and symmetric and include all of the non-rigid lattice deformation. Isometries, Q, have the property that $Q^{-1} = Q^{\mathsf{T}}$, where $^{\mathsf{T}}$ is used to indicate matrix transpose.

- T is an integer transformation matrix that generates a superlattice of L_1
- N is a unimodular reorientation matrix that generates a lattice equivalent to L_1T with reoriented lattice vectors

AtomMapping

```
class libcasm.mapping.info.AtomMapping(
    self: AtomMapping,
    displacement: numpy.ndarray[numpy.float64[m, n]],
    permutation: list[int],
    translation: numpy.ndarray[numpy.float64[3, 1]],
)
```

An atom mapping is defined in the context of an existing `LatticeMapping`. An atom mapping has the form:

$$F\left(\vec{r}_1(i) + \vec{d}(i)\right) = \vec{r}_2(p_i) + \vec{t}$$

where:

- $\vec{r}_1(i)$ is the Cartesian coordinates of the i-th atom in the parent superstructure. The parent superstructure can be constructed using *libcasm.xtal.make_superstructure*:

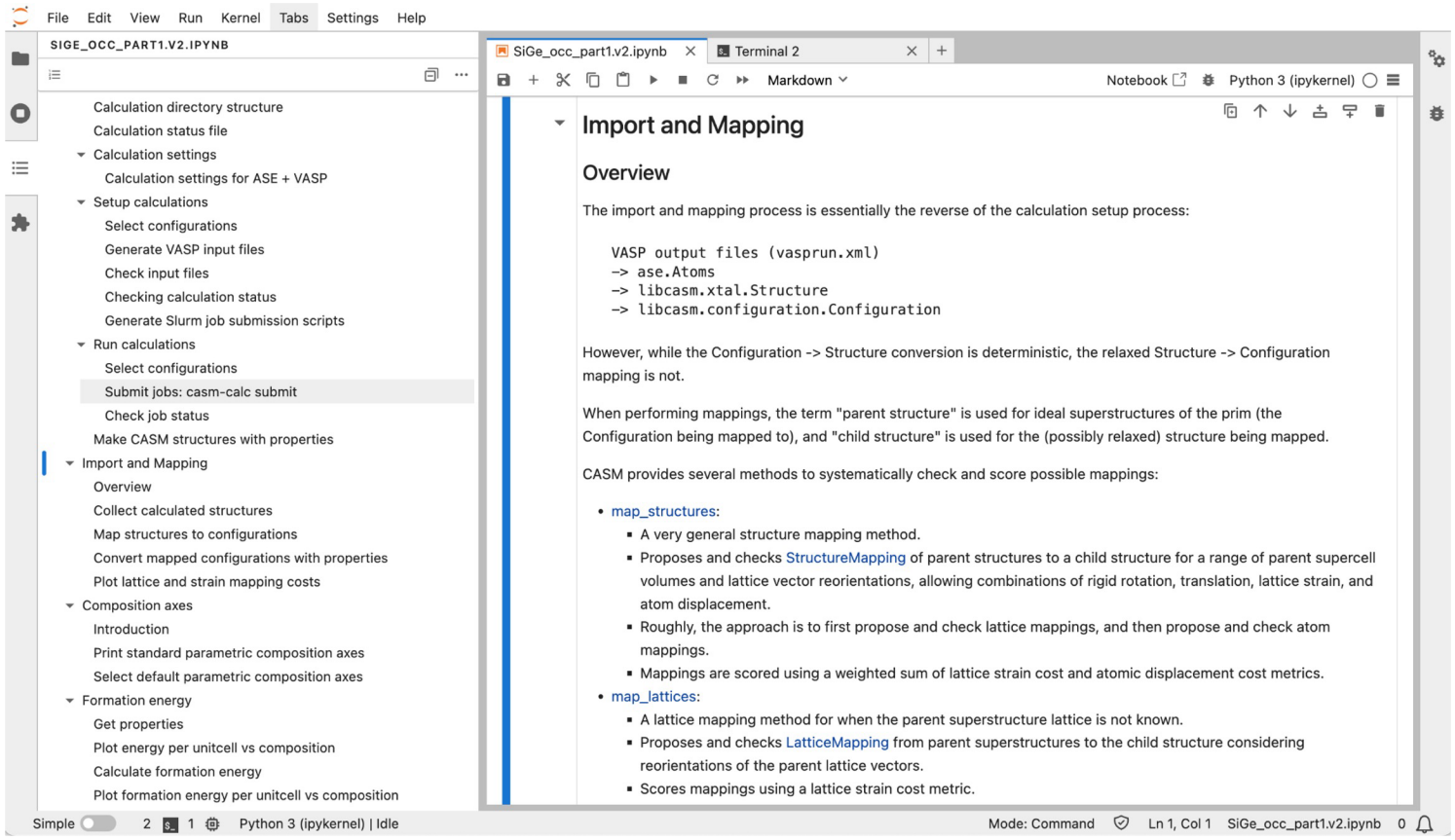
  ```
  import libcasm.xtal as xtal
  parent_superstructure = xtal.make_superstructure(
      T * N, parent_structure)
  ```

  where T, and N are from a `LatticeMapping`. Then the i-th atom coordinate, $\vec{r}_1(i)$, is equal to:

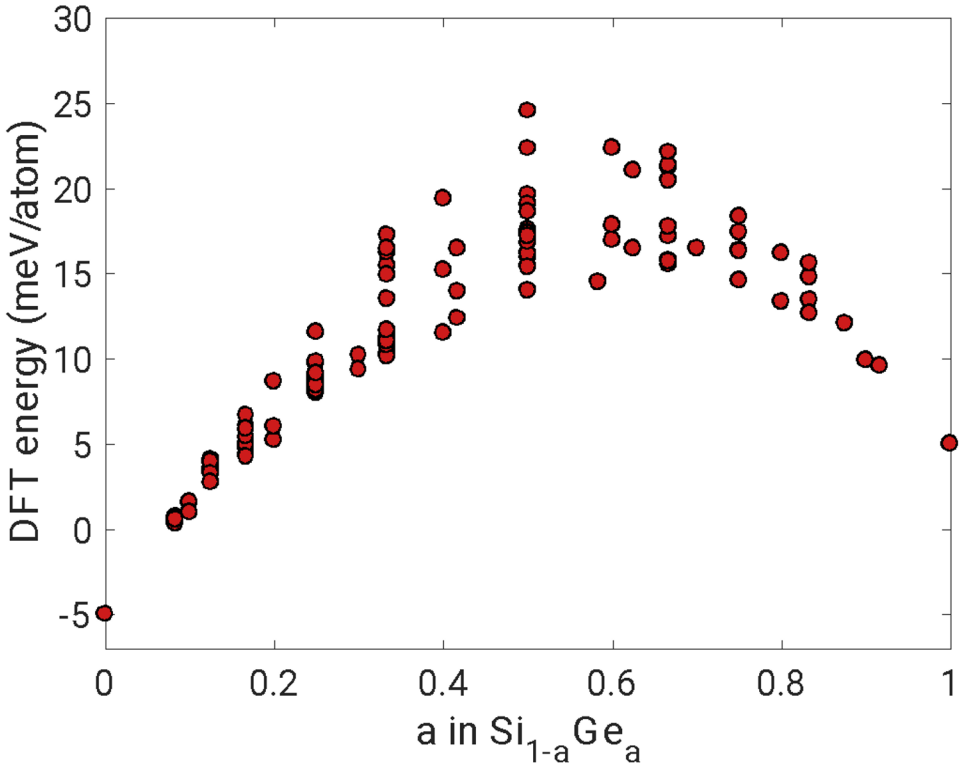
  ```
  parent_superstructure.atom_coordinate_cart()[:,i]
  ```

- $\vec{r}_2(i)$ is the Cartesian coordinates of i-th atom in the child structure.
- F is the parent-to-child deformation gradient tensor from a `LatticeMapping`.
- p_i is a permutation vector, specifying which atom in the child structure (p_i) is mapped to the i-th site of the parent superstructure. Values of p_i greater than the number of atoms in the child structure indicate inferred vacancies in mappings to `Prim` with vacancies allowed.
- $\vec{t}$ is a translation vector, in Cartesian coordinates, usually chosen so the average displacement is zero.
- $\vec{d}(i)$**: The Cartesian displacement associated with the atom** at the i-th site in the parent superstructure.

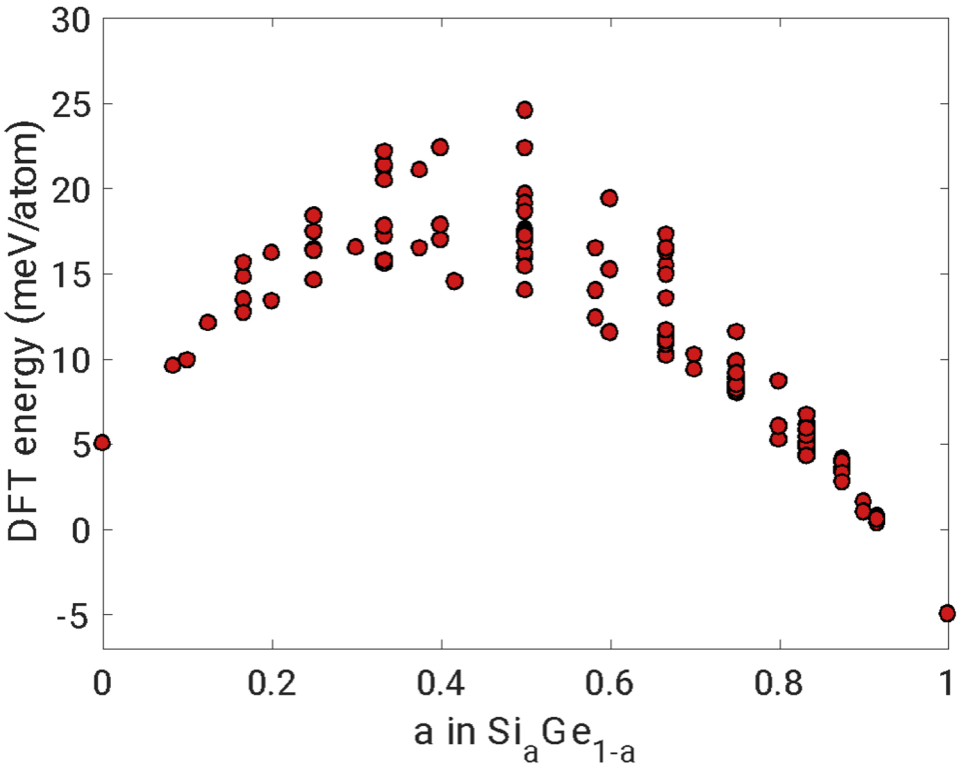
Import and Mapping



Composition Axes

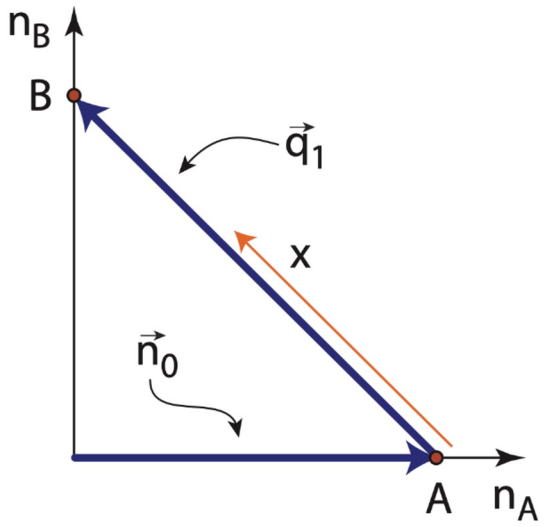


Pure Si as a=0 reference

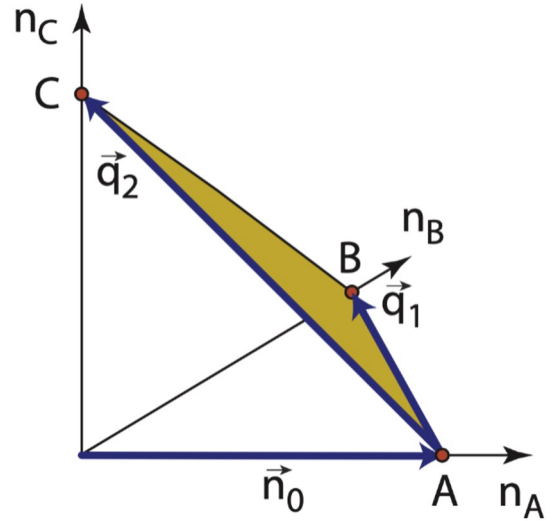


Pure Ge as a=0 reference

Composition Axes



Binary



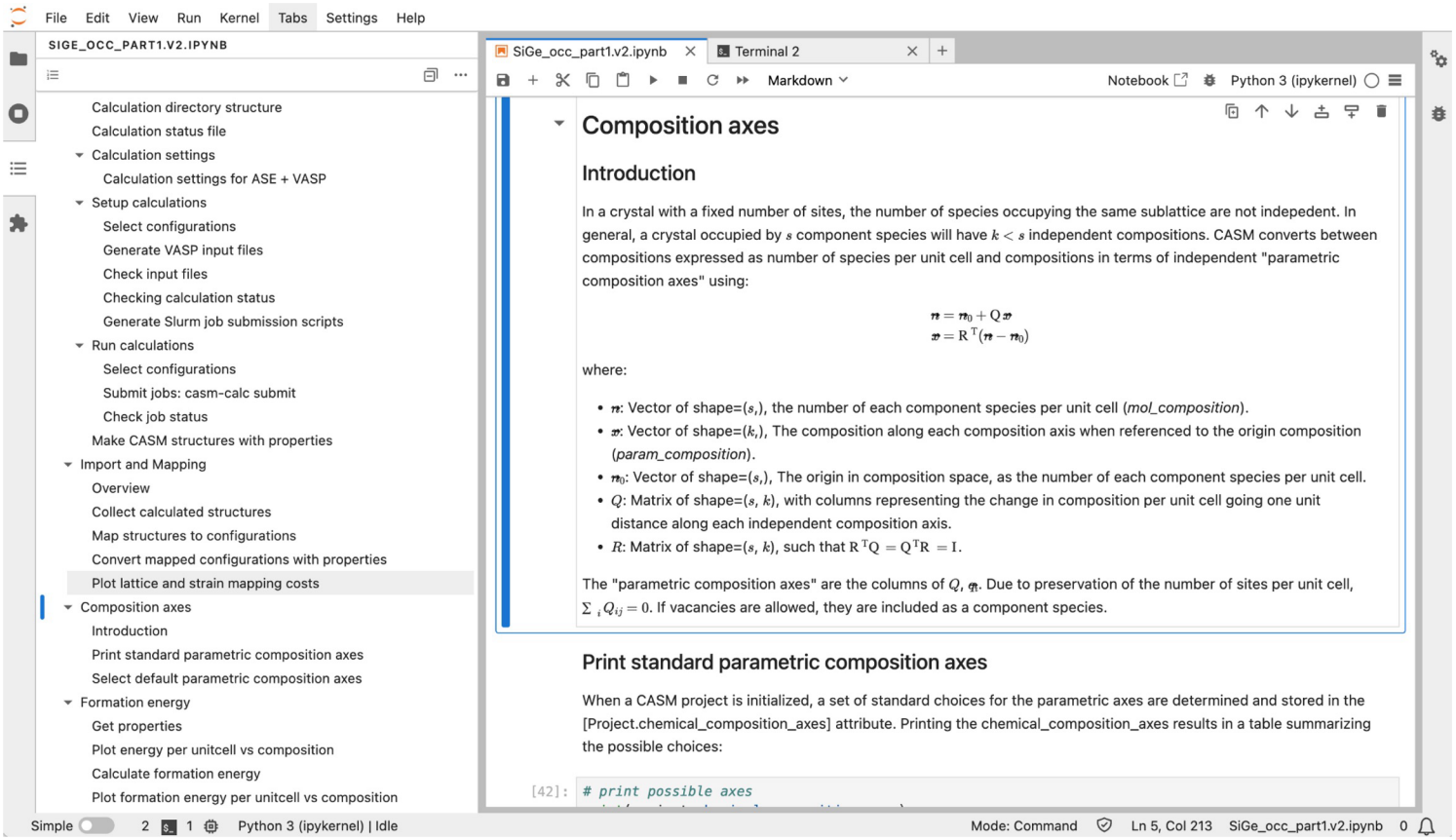
Ternary

Selecting composition Axes

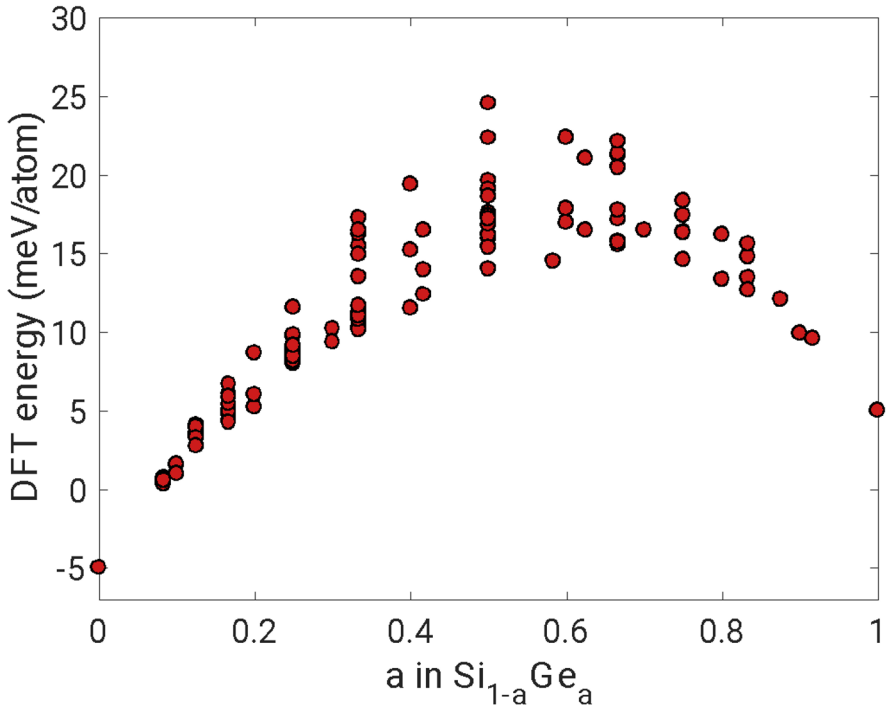
```
# select axes with <key>, unset with None
project.chemical_composition_axes.set_current_axes(1)

# print possible axes and formulas for current choice
print(project.chemical_composition_axes)

# commit current choice
project.chemical_composition_axes.commit()
```

```
KEY  ORIGIN    a     GENERAL FORMULA
---  --------  ----  -----------------
  0  Ge(2)     SiGe  Si(a)Ge(2-a)
  1  Si(2)     SiGe  Si(2-a)Ge(a)

Currently selected composition axes: 1

Parametric composition:
  comp(a) = -0.5*(comp_n(Si) - 2)  + 0.5*comp_n(Ge)

Composition:
  comp_n(Si) = 2 - 1*comp(a)
  comp_n(Ge) = 1*comp(a)

Parametric chemical potentials:
  param_chem_pot(a) = -chem_pot(Si) + chem_pot(Ge)
```

Composition Axes

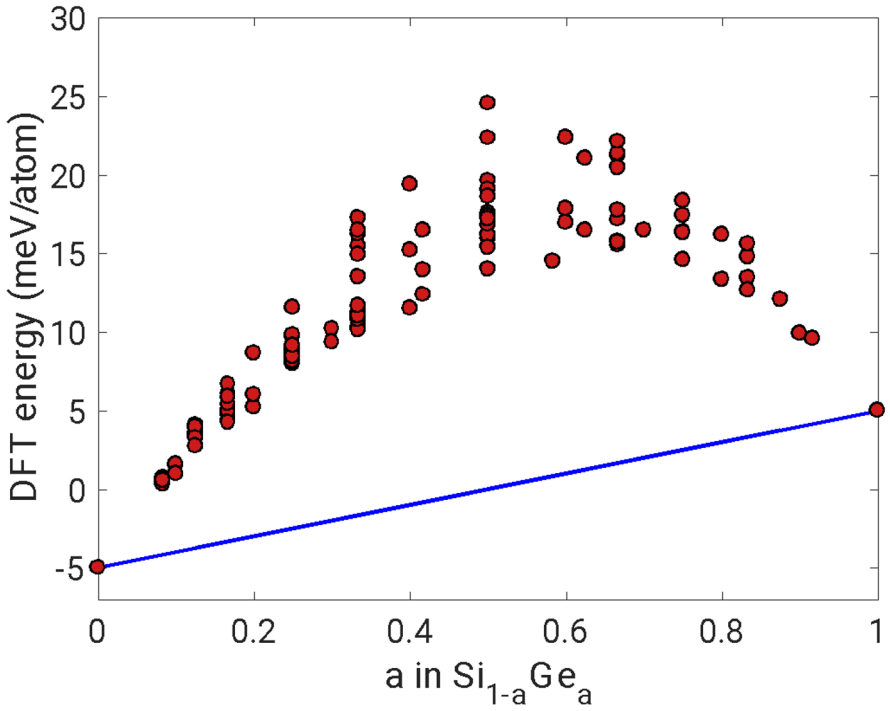


Setting an energy reference

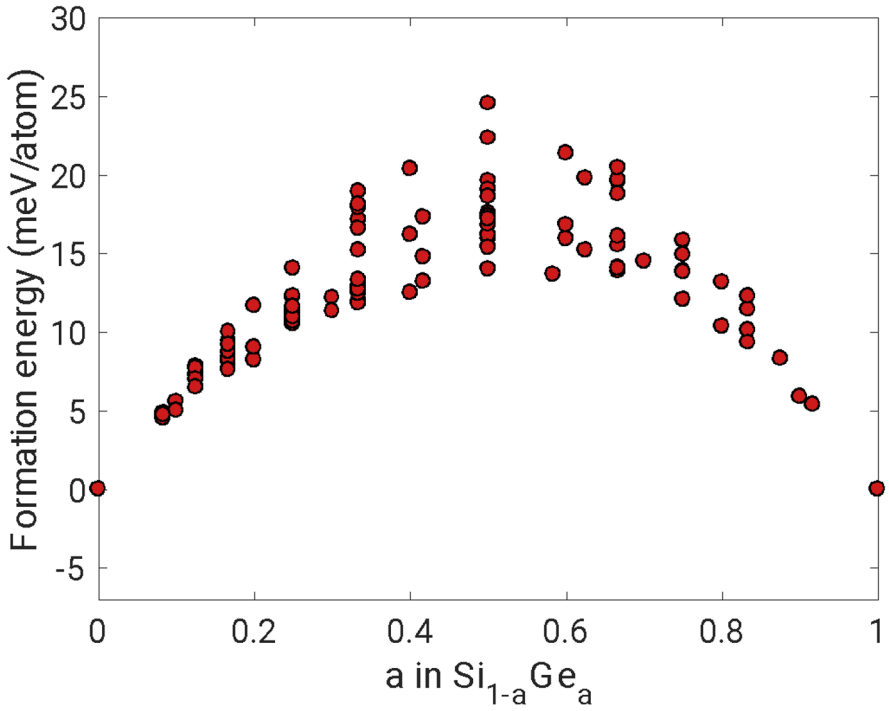


Chemical energy can only be measured relative to reference compounds

Setting an energy reference

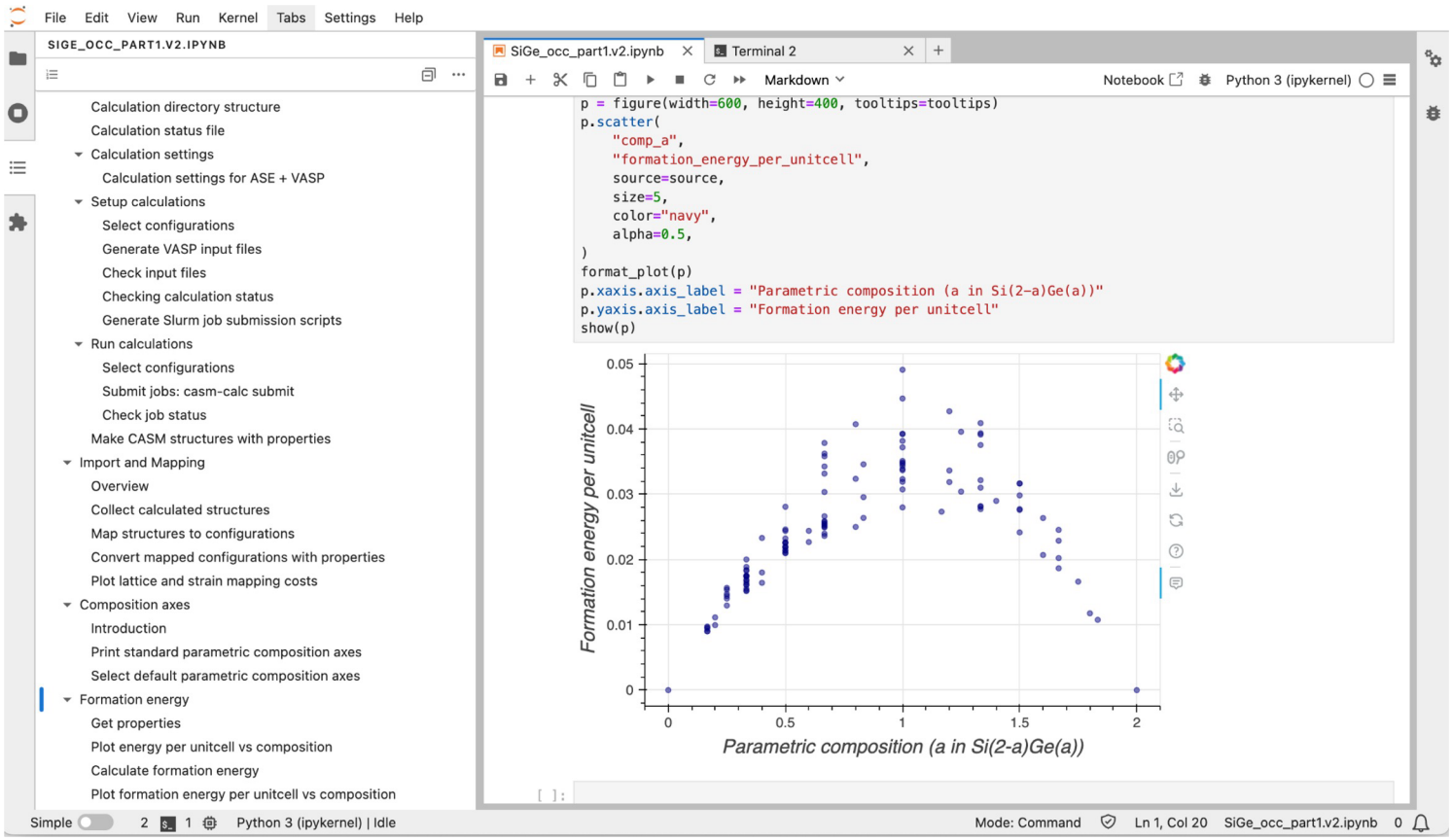


DFT energy



Formation energy

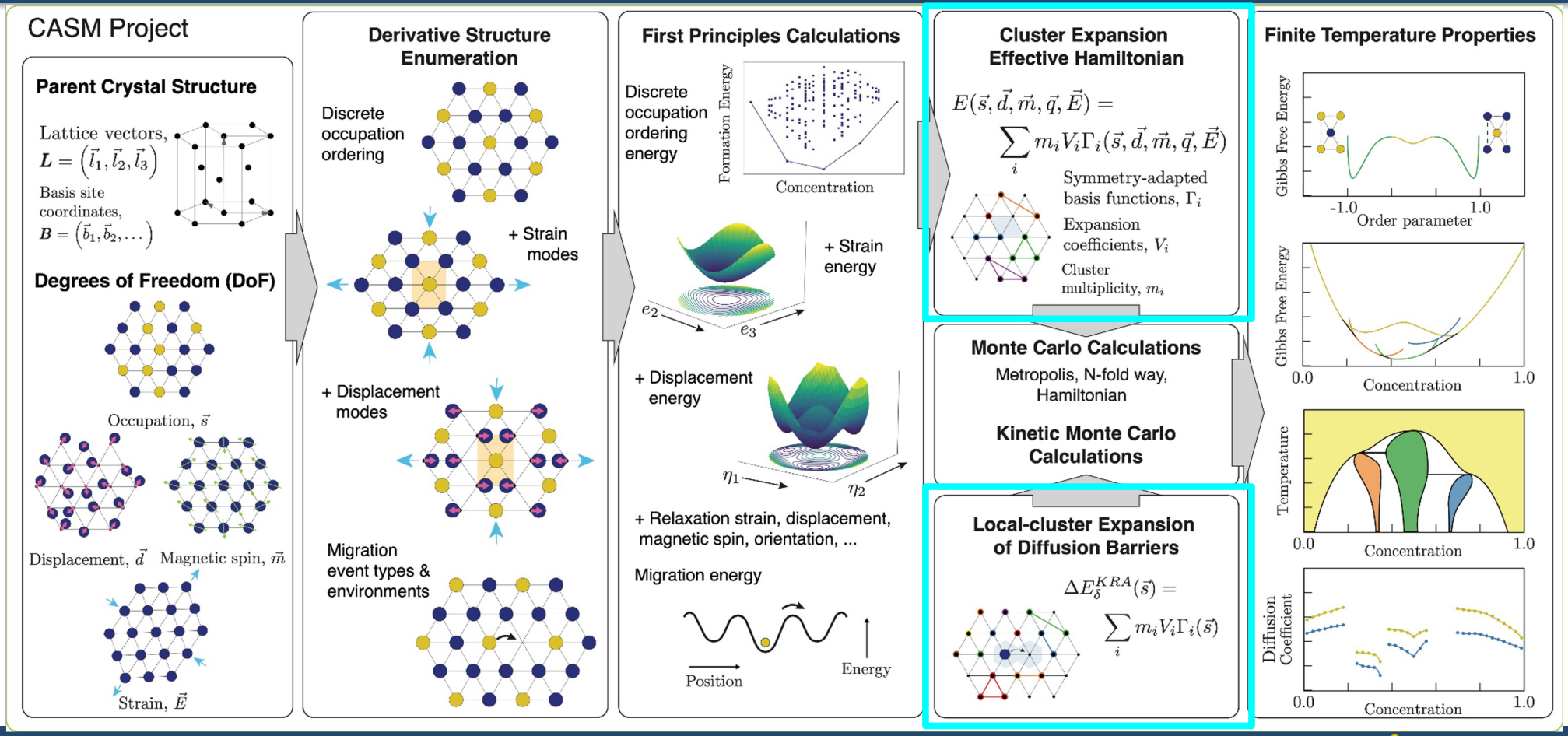
Formation energy



Puchala, et al. Comp. Mater. Sci. 217, 111987 (2023).

PRISMS

Cluster expansion construction & fitting

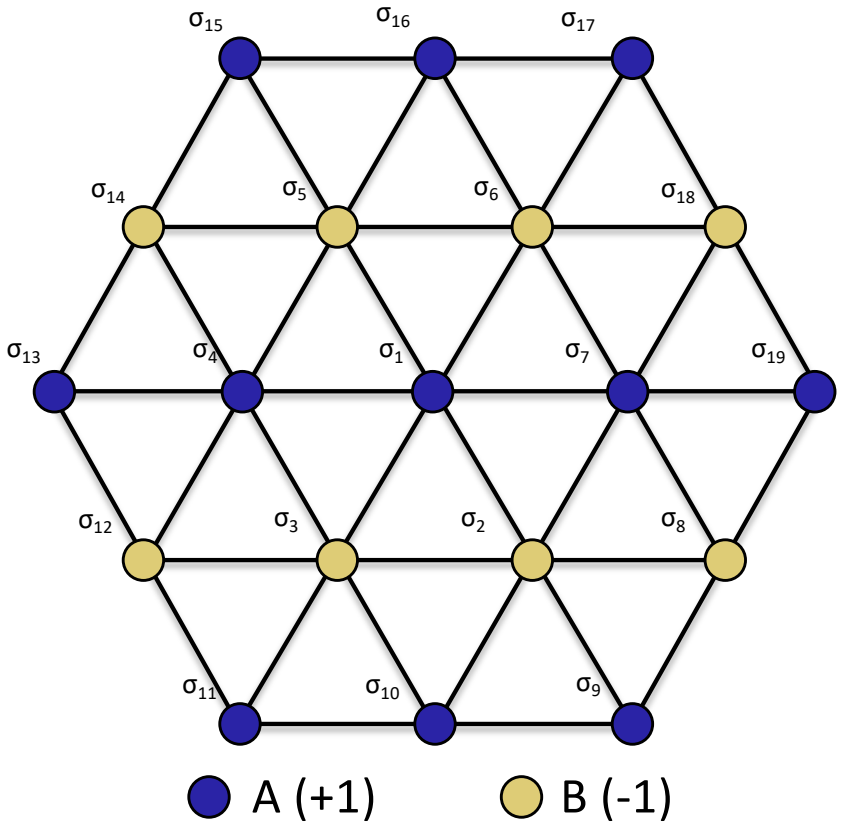


Puchala, et al. Comp. Mater. Sci. 217, 111987 (2023).

PRISMS

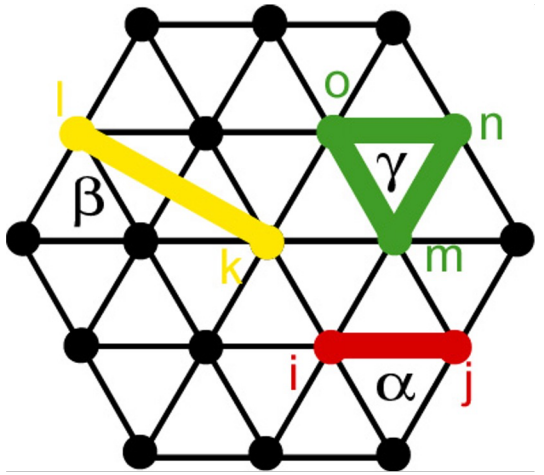
Cluster expansion

A particular ordering is tracked with occupation variables σ_i assigned to each site i



Cluster expansion

$$E(\vec{\sigma}) = V_o + \sum_i V_i \sigma_i + \sum_{i,j} V_{i,j} \sigma_i \sigma_j + \sum_{i,j,k} V_{i,j,k} \sigma_i \sigma_j \sigma_k + \ldots$$



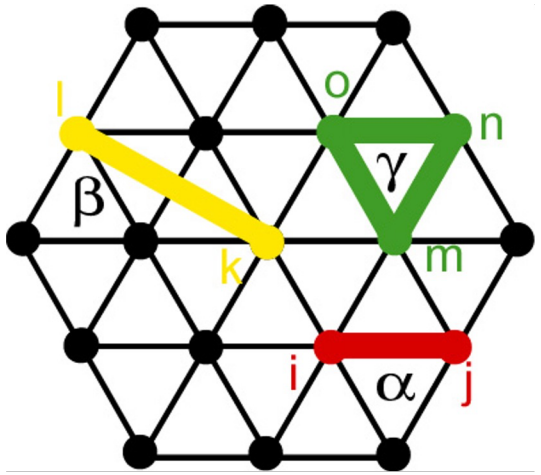
Cluster expansion

$$E(\vec{\sigma}) = V_o + \sum_i V_i \sigma_i + \sum_{i,j} V_{i,j} \sigma_i \sigma_j + \sum_{i,j,k} V_{i,j,k} \sigma_i \sigma_j \sigma_k + \ldots$$

Can be written more compactly as

$$E(\vec{\sigma}) = V_0 + \sum_\alpha V_\alpha \, \phi_\alpha(\vec{\sigma})$$

$$\phi_\alpha(\vec{\sigma}) = \prod_{i \in \alpha} \sigma_i$$



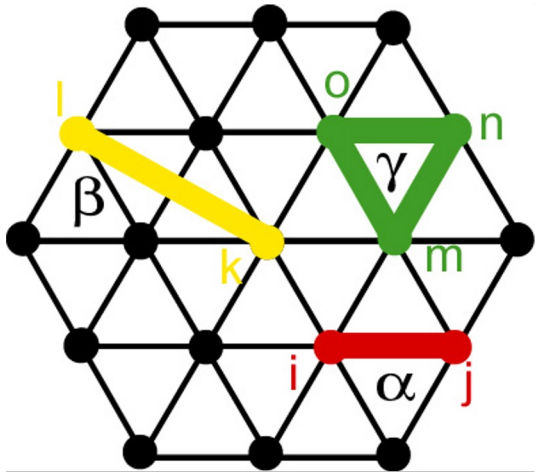
Cluster expansion

$$E(\vec{\sigma}) = V_o + \sum_i V_i \sigma_i + \sum_{i,j} V_{i,j} \sigma_i \sigma_j + \sum_{i,j,k} V_{i,j,k} \sigma_i \sigma_j \sigma_k + \ldots$$

Can be written more compactly as

$$E(\vec{\sigma}) = V_0 + \sum_\alpha V_\alpha \, \phi_\alpha(\vec{\sigma})$$

$$\phi_\alpha(\vec{\sigma}) = \prod_{i \in \alpha} \sigma_i$$



Two clusters α and δ that are equivalent by symmetry have the same expansion coefficient

$$V_\alpha = V_\delta$$

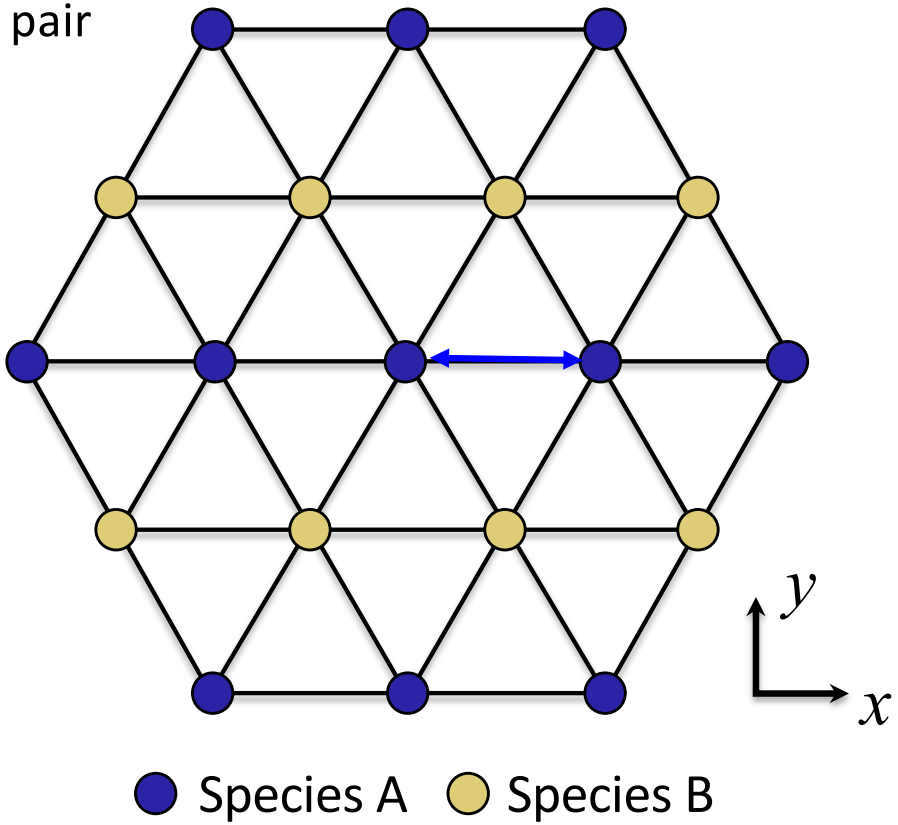
Cluster expansion

Example of the nearest neighbor pair

$$E(\vec{\sigma}) = V_0 + \sum_{\alpha} V_\alpha \, \phi_\alpha(\vec{\sigma})$$

$$\phi_\alpha(\vec{\sigma}) = \prod_{i \in \alpha} \sigma_i$$

All these pair clusters will have the same expansion coefficient V



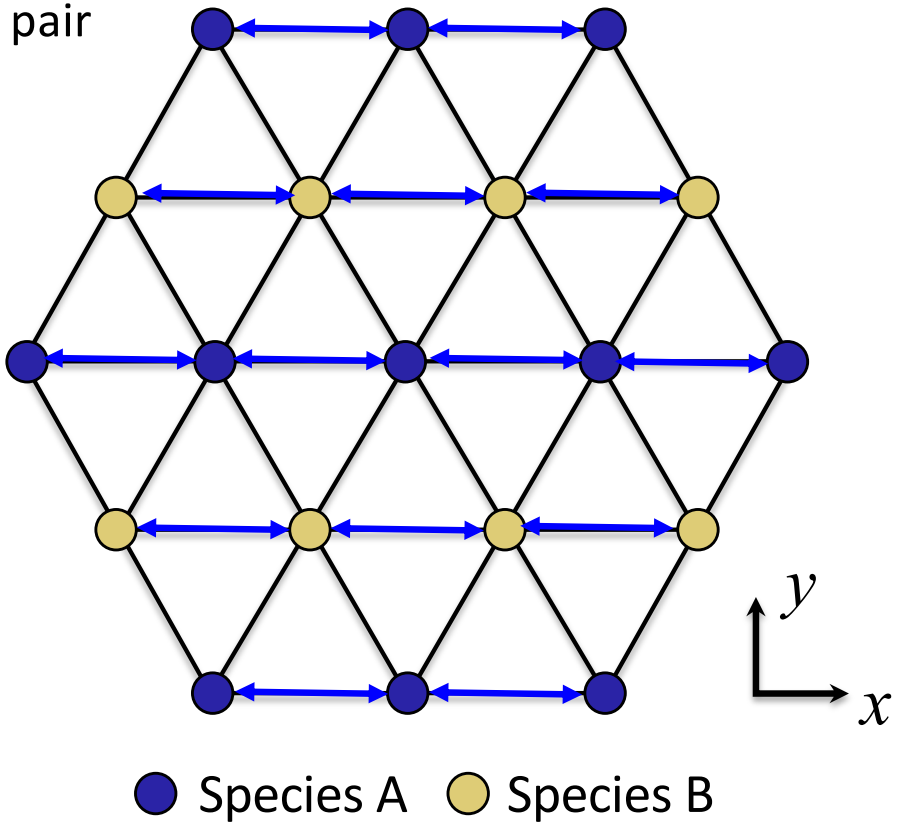
Cluster expansion

Example of the nearest neighbor pair

$$E(\vec{\sigma}) = V_0 + \sum_{\alpha} V_\alpha \, \phi_\alpha(\vec{\sigma})$$

$$\phi_\alpha(\vec{\sigma}) = \prod_{i \in \alpha} \sigma_i$$

All these pair clusters will have the same expansion coefficient V



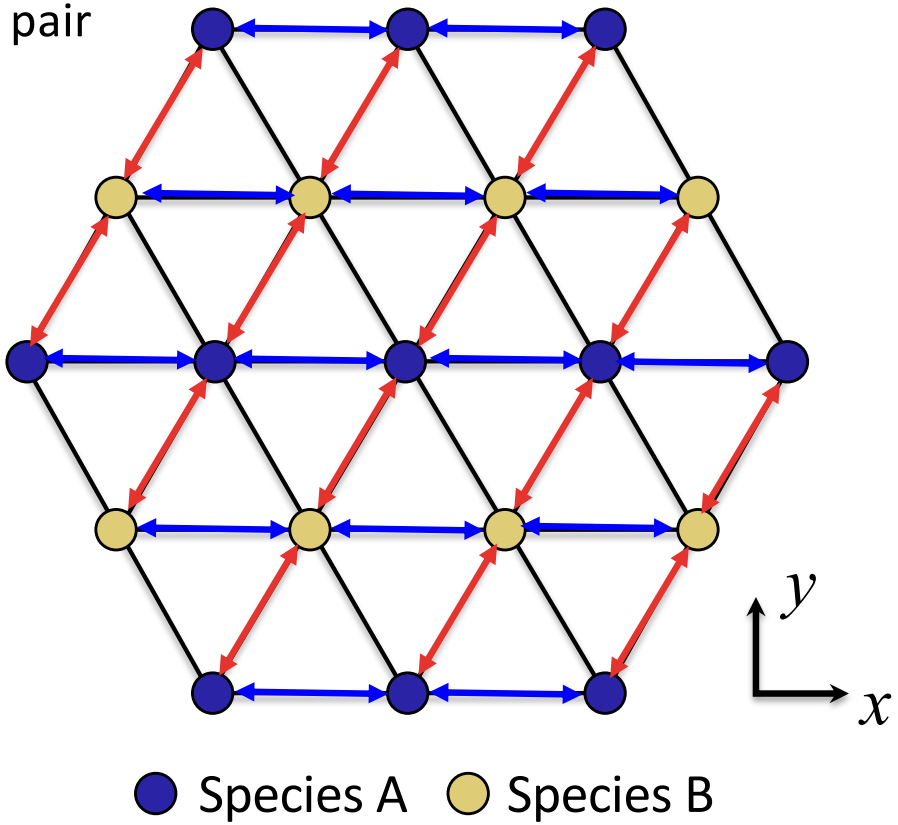
Cluster expansion

Example of the nearest neighbor pair

$$E(\vec{\sigma}) = V_0 + \sum_{\alpha} V_\alpha \, \phi_\alpha(\vec{\sigma})$$

$$\phi_\alpha(\vec{\sigma}) = \prod_{i \in \alpha} \sigma_i$$

All these pair clusters will have the same expansion coefficient V



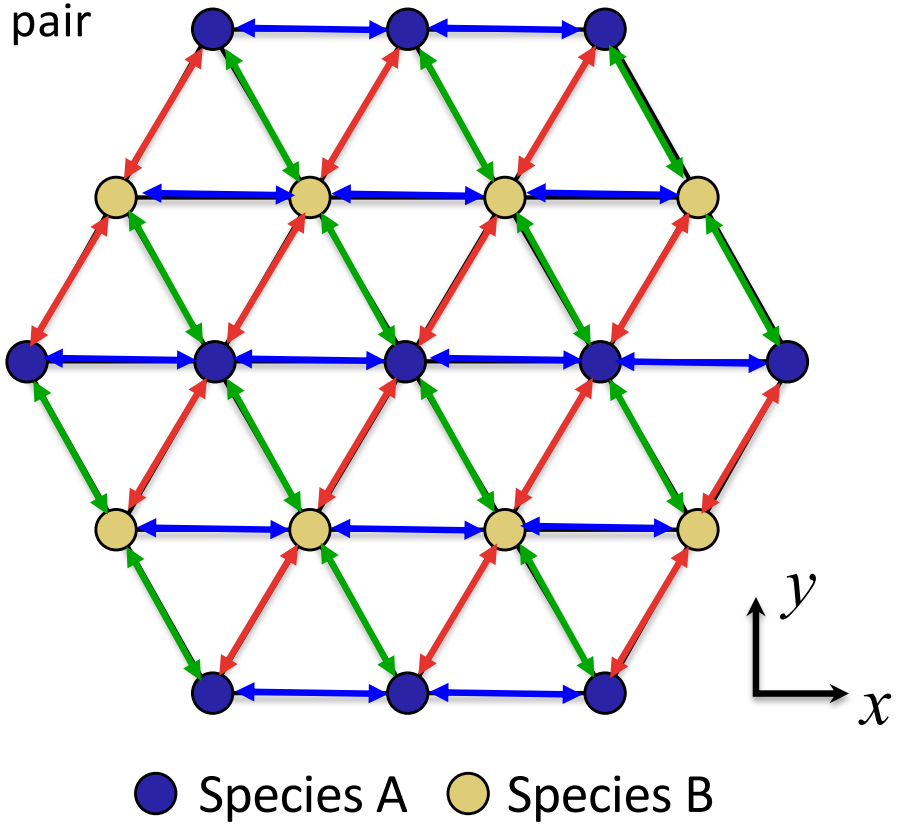
Cluster expansion

Example of the nearest neighbor pair

$$E(\vec{\sigma}) = V_0 + \sum_{\alpha} V_\alpha \, \phi_\alpha(\vec{\sigma})$$

$$\phi_\alpha(\vec{\sigma}) = \prod_{i \in \alpha} \sigma_i$$

All these pair clusters will have the same expansion coefficient V



Symmetry equivalence of clusters lets us simplify the cluster expansion

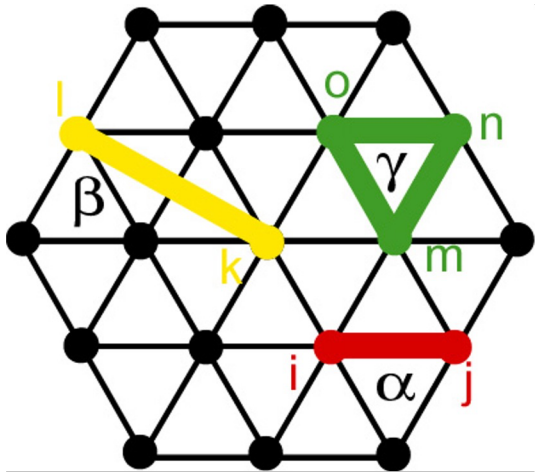
$$E(\vec{\sigma}) = V_0 + \sum_{\alpha} V_\alpha \, \phi_\alpha(\vec{\sigma})$$

$\phi_\alpha(\vec{\sigma}) = \prod_{i \in \alpha} \sigma_i$

Using symmetry

$$e(\vec{\sigma}) = v_0 + \sum_{\alpha} v_\alpha \Gamma_\alpha(\vec{\sigma})$$

$$\Gamma_\alpha = \frac{1}{N m_\alpha} \sum_{\beta \in \Omega_\alpha} \phi_\beta(\vec{\sigma})$$

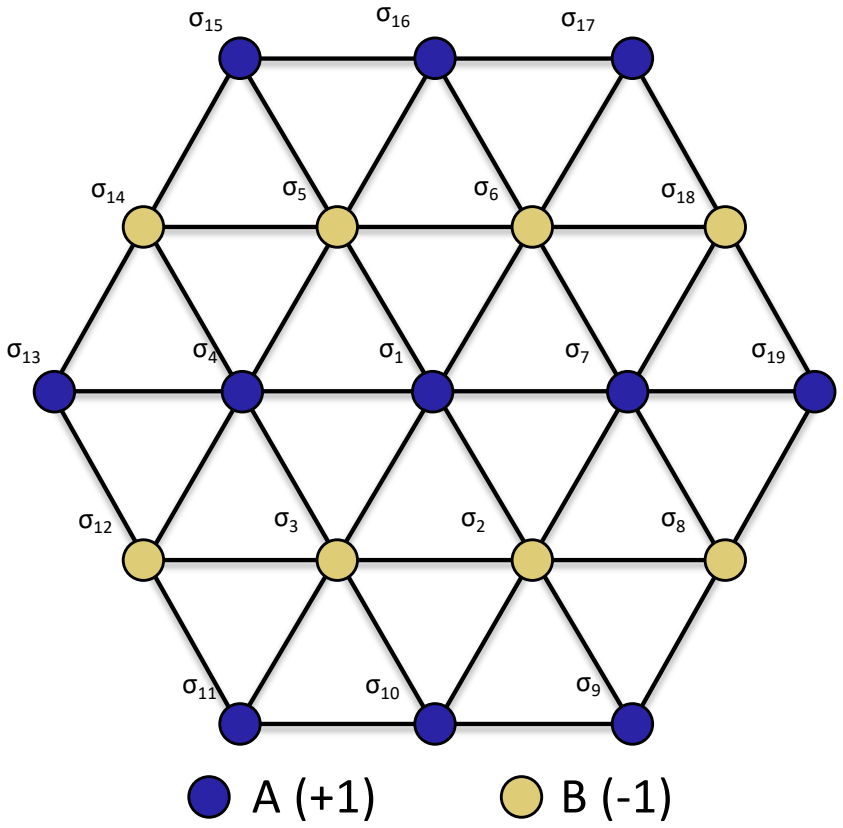


Cluster expansion

$$\Gamma_\alpha = \frac{1}{N m_\alpha} \sum_{\beta \in \Omega_\alpha} \phi_\beta(\vec{\sigma})$$

σ_i = +1 if species A

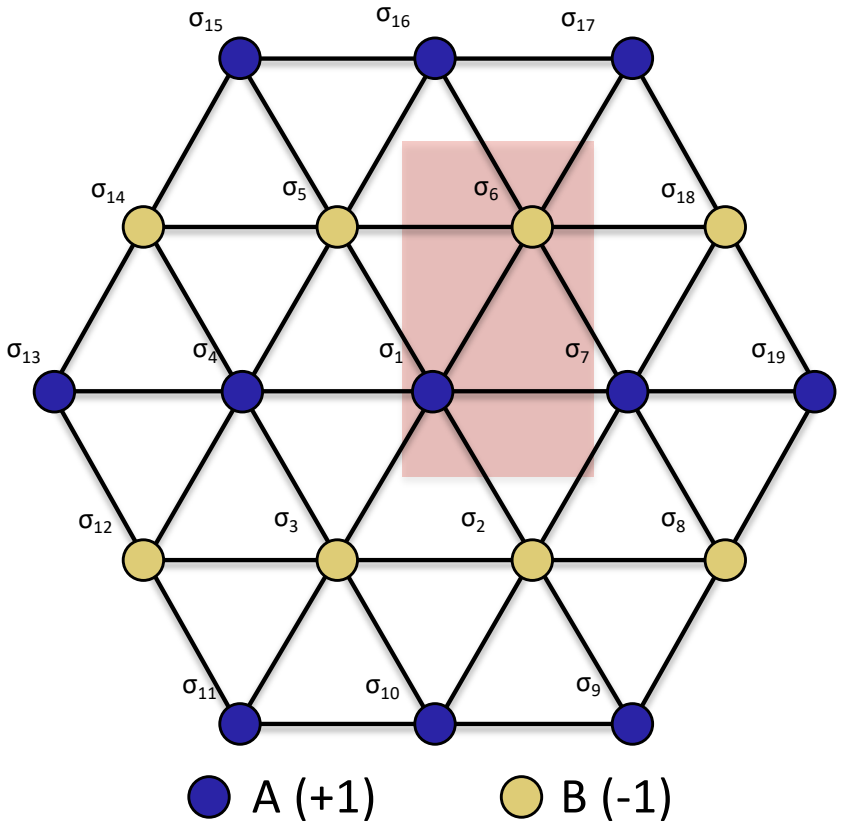
σ_i = -1 if species B



Cluster expansion

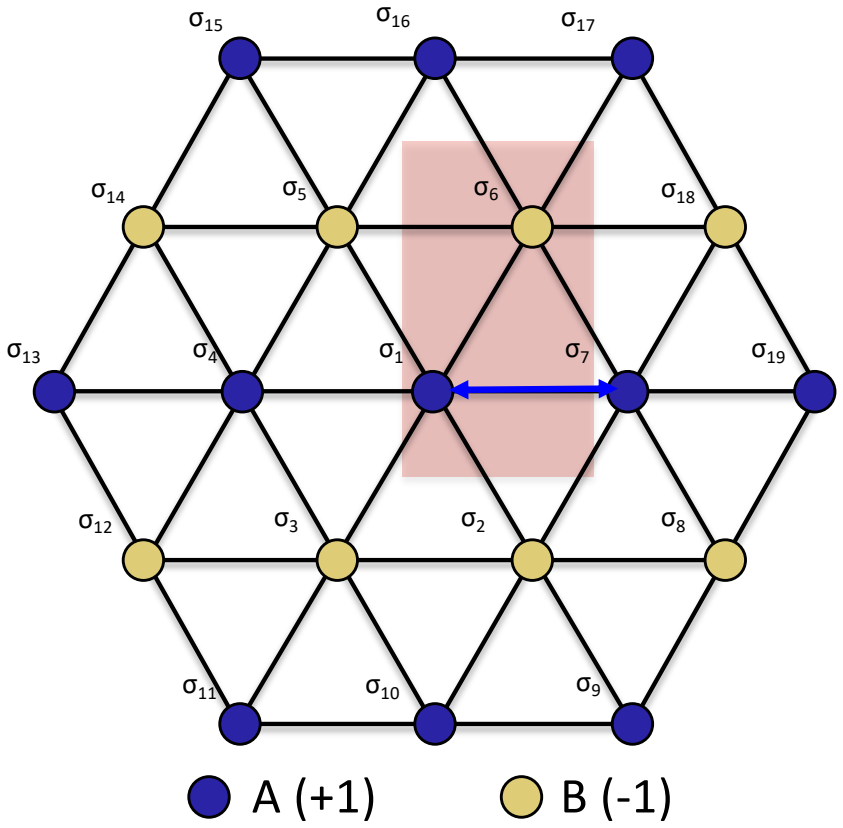
Correlation function
for the point cluster

Average site occupation is
$\Gamma_1=(\sigma_1+\sigma_6)/2=0$



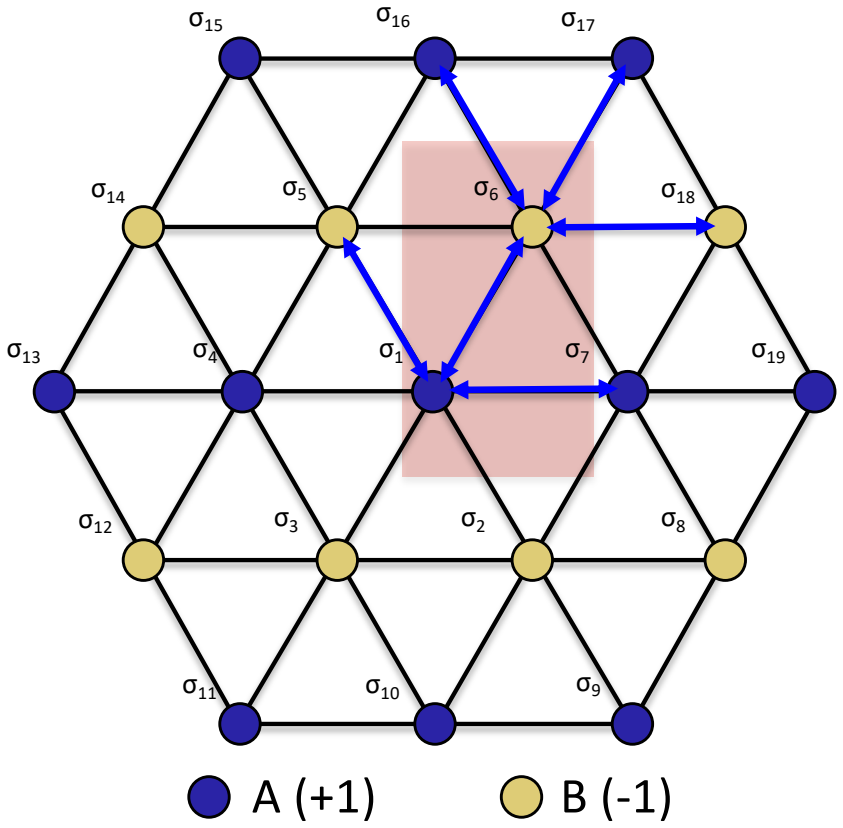
Cluster expansion

Correlation function for the nearest neighbor pair cluster



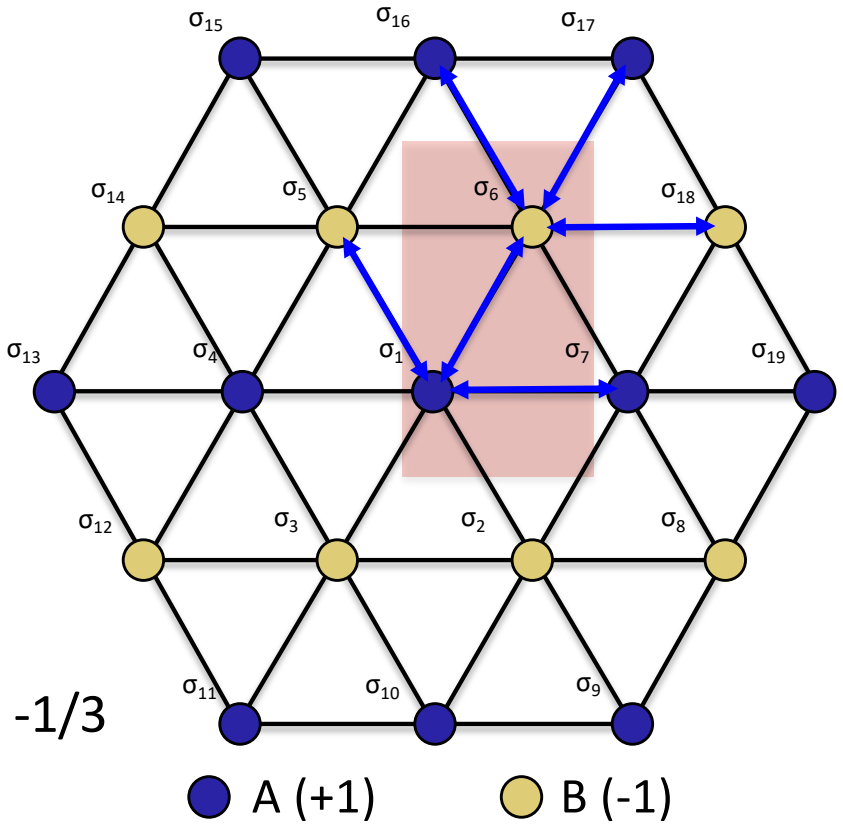
Cluster expansion

Correlation function for the nearest neighbor pair cluster



Cluster expansion

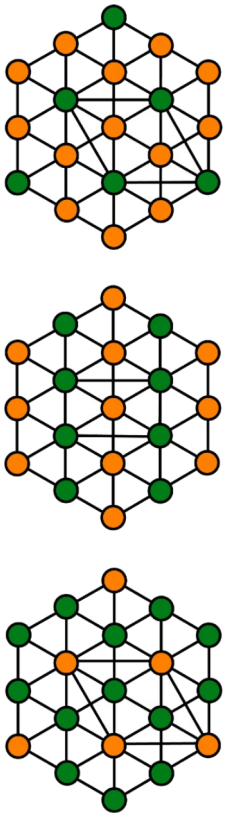
Correlation function for the nearest neighbor pair cluster



Average NN pair correlation is

$\Gamma_2=(\sigma_1\sigma_7+\sigma_1\sigma_6+\sigma_1\sigma_5+\sigma_6\sigma_{18}+\sigma_6\sigma_{17}+\sigma_6\sigma_{16})/6 = -1/3$

Fitting Coefficients

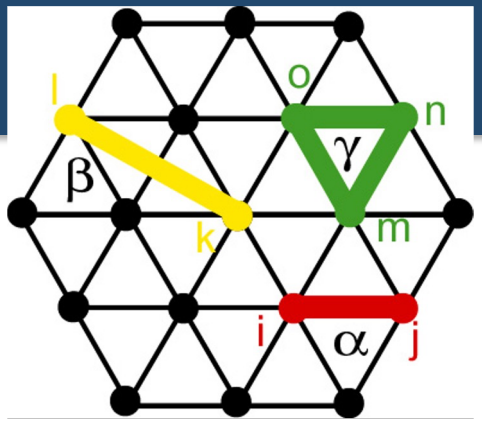


Cluster

$$\begin{pmatrix} e(\vec{\sigma}_1) \\ \vdots \\ e(\vec{\sigma}_L) \\ \vdots \\ e(\vec{\sigma}_M) \end{pmatrix} = \begin{pmatrix} 1 & \Gamma_\alpha(\vec{\sigma}_1) & \Gamma_\beta(\vec{\sigma}_1) & \Gamma_\gamma(\vec{\sigma}_1) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & \Gamma_\alpha(\vec{\sigma}_L) & \Gamma_\beta(\vec{\sigma}_L) & \Gamma_\gamma(\vec{\sigma}_L) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & \Gamma_\alpha(\vec{\sigma}_M) & \Gamma_\beta(\vec{\sigma}_M) & \Gamma_\gamma(\vec{\sigma}_M) & \cdots \end{pmatrix} \begin{pmatrix} v_o \\ v_\alpha \\ v_\beta \\ v_\gamma \\ \vdots \end{pmatrix}$$

$$\vec{Y} = X\vec{V}$$

Fitting Coefficients

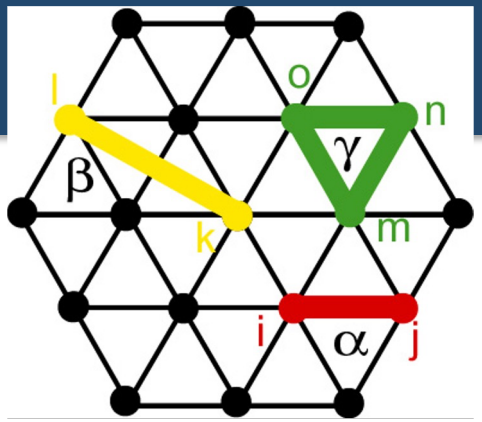


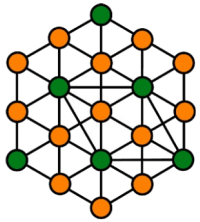
$$\frac{E(\vec{\sigma})}{N} = \frac{V_0}{N} + \sum_{\alpha} m_\alpha V_\alpha \left(\frac{1}{N m_\alpha} \sum_{\delta \in \Omega_\alpha} \phi_\delta(\vec{\sigma}) \right)$$

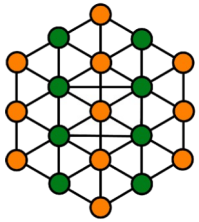
$\Gamma_\alpha(\vec{\sigma})$ Correlation function

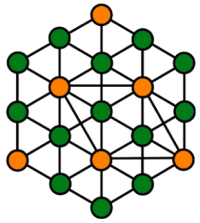
$$e(\vec{\sigma}) = v_0 + \sum_{\alpha} v_\alpha \Gamma_\alpha(\vec{\sigma})$$

Fitting Coefficients

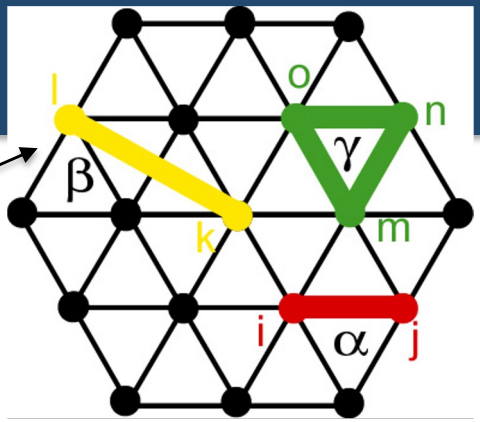




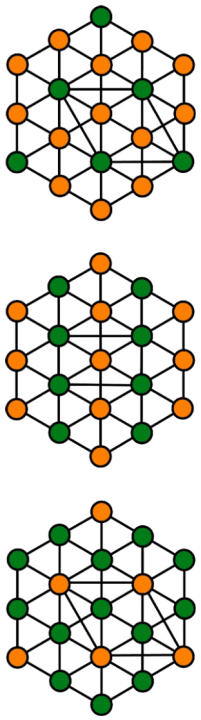




Fitting Coefficients



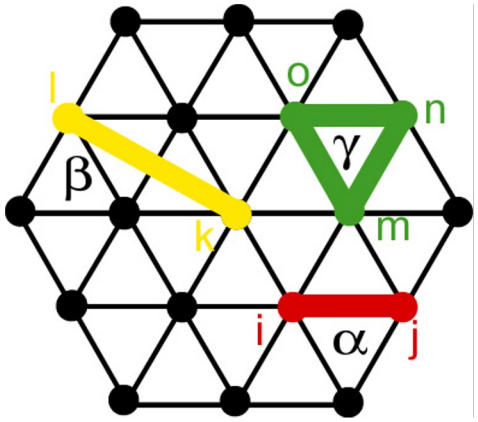
Cluster



$$\begin{pmatrix} e(\vec{\sigma}_1) \\ \cdot \\ \cdot \\ \cdot \\ e(\vec{\sigma}_L) \\ \cdot \\ \cdot \\ \cdot \\ e(\vec{\sigma}_M) \end{pmatrix} = \begin{pmatrix} 1 & \Gamma_\alpha(\vec{\sigma}_1) & \Gamma_\beta(\vec{\sigma}_1) & \Gamma_\gamma(\vec{\sigma}_1) & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \vdots & \vdots & \vdots & \vdots & \cdot & \cdot \\ 1 & \Gamma_\alpha(\vec{\sigma}_L) & \Gamma_\beta(\vec{\sigma}_L) & \Gamma_\gamma(\vec{\sigma}_L) & \cdot & \cdot \\ \vdots & \vdots & \vdots & \vdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & \Gamma_\alpha(\vec{\sigma}_M) & \Gamma_\beta(\vec{\sigma}_M) & \Gamma_\gamma(\vec{\sigma}_M) & \cdot & \cdot \end{pmatrix} \begin{pmatrix} v_o \\ v_\alpha \\ v_\beta \\ v_\gamma \\ \cdot \\ \cdot \end{pmatrix}$$

$$\vec{Y} = X\vec{V}$$

Exercise: Using a varying number of terms

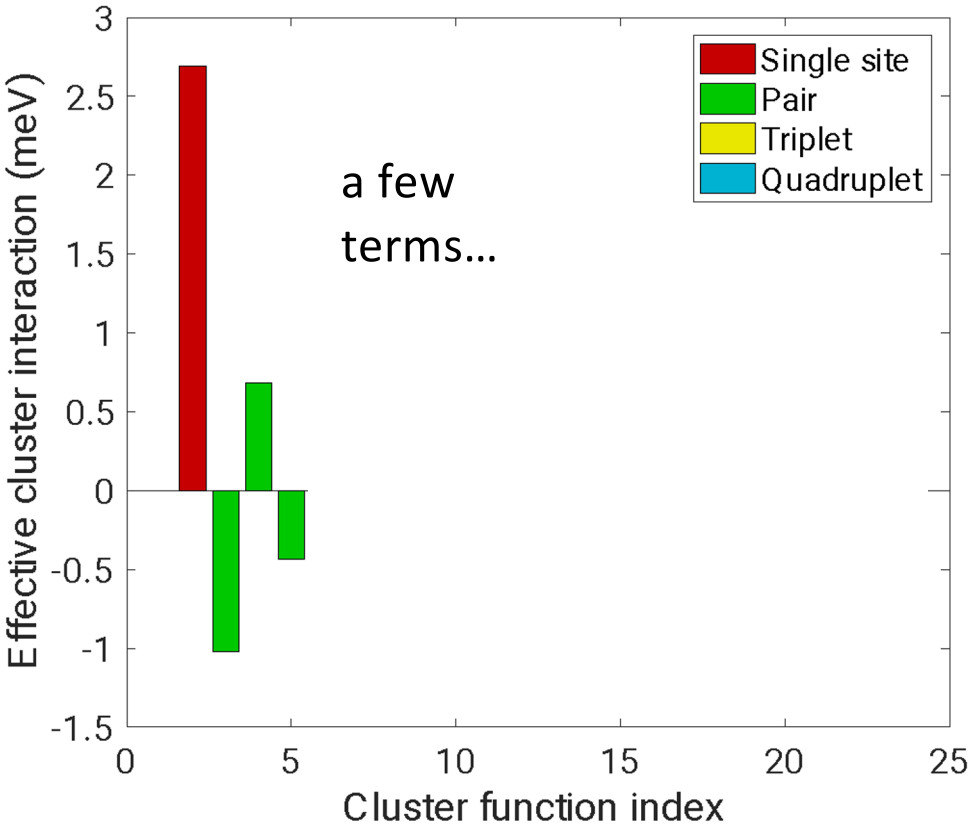


$$\Phi_\alpha = \sigma_i \sigma_j$$

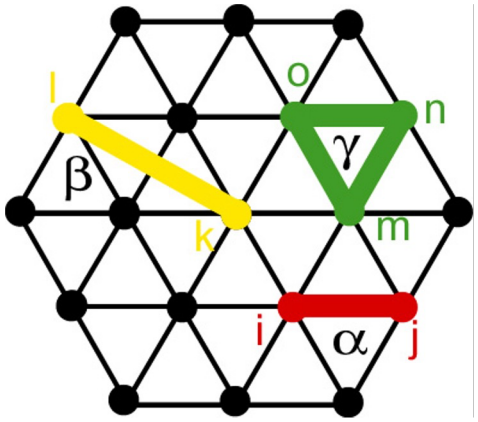
$$\Phi_\beta = \sigma_l \sigma_k$$

$$\Phi_\gamma = \sigma_m \sigma_n \sigma_o$$

$$E(\vec{\sigma}) = V_o + \sum_i V_i \sigma_i + \sum_{i,j} V_{i,j} \sigma_i \sigma_j + \sum_{i,j,k} V_{i,j,k} \sigma_i \sigma_j \sigma_k + \ldots$$



Exercise: Using a varying number of terms

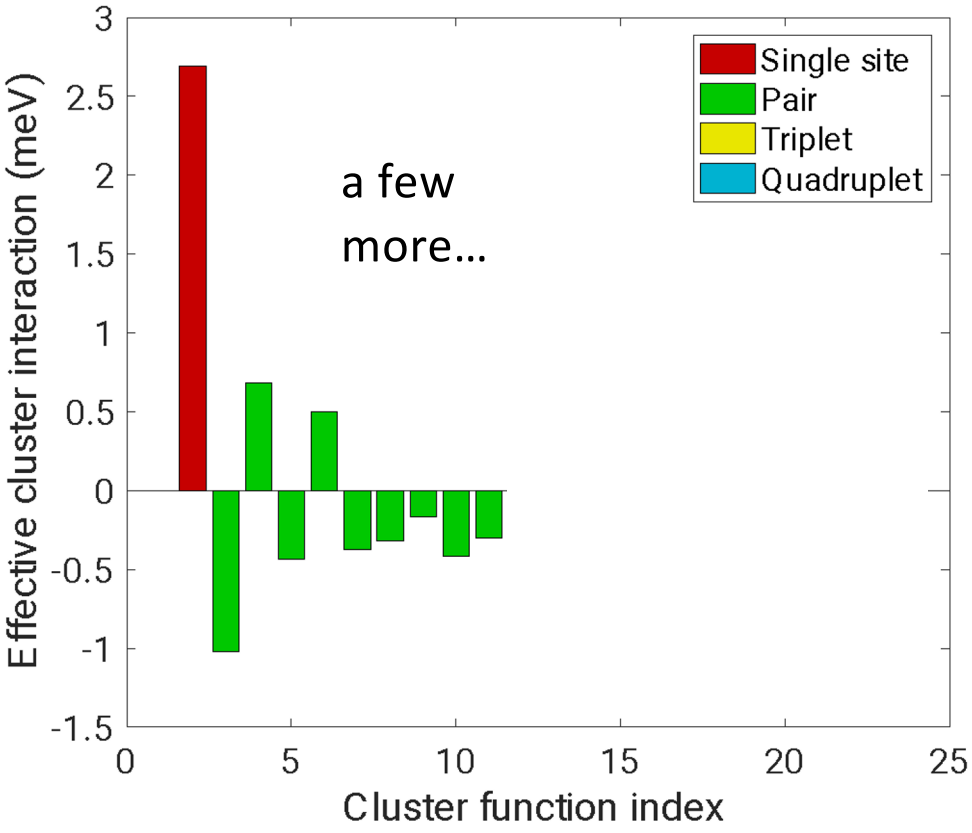


$$\Phi_\alpha = \sigma_i \sigma_j$$

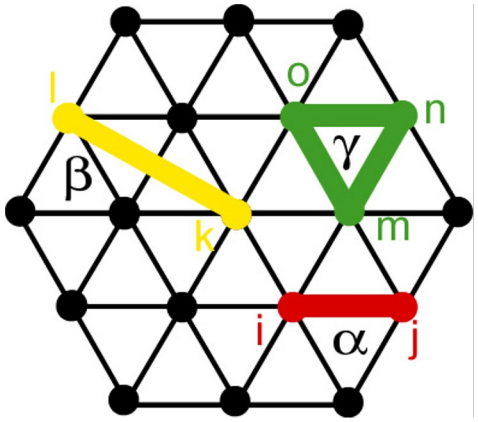
$$\Phi_\beta = \sigma_l \sigma_k$$

$$\Phi_\gamma = \sigma_m \sigma_n \sigma_o$$

$$E(\vec{\sigma}) = V_o + \sum_i V_i \sigma_i + \sum_{i,j} V_{i,j} \sigma_i \sigma_j + \sum_{i,j,k} V_{i,j,k} \sigma_i \sigma_j \sigma_k + \ldots$$



Exercise: Using a varying number of terms

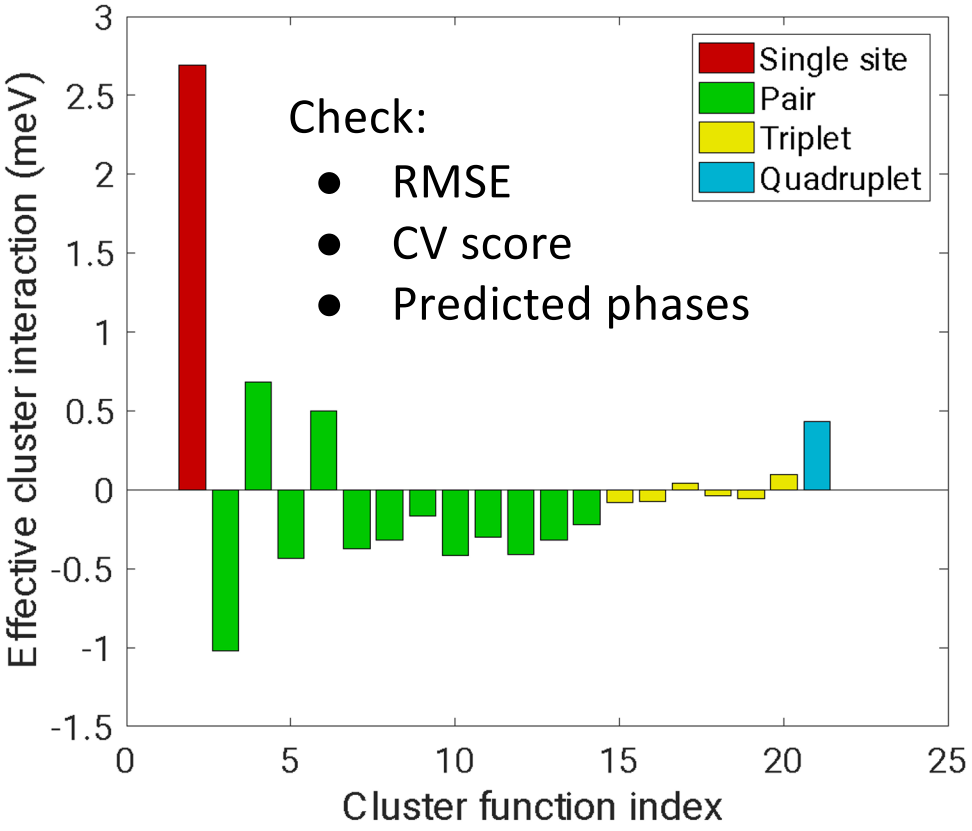


$$\Phi_\alpha = \sigma_i \sigma_j$$

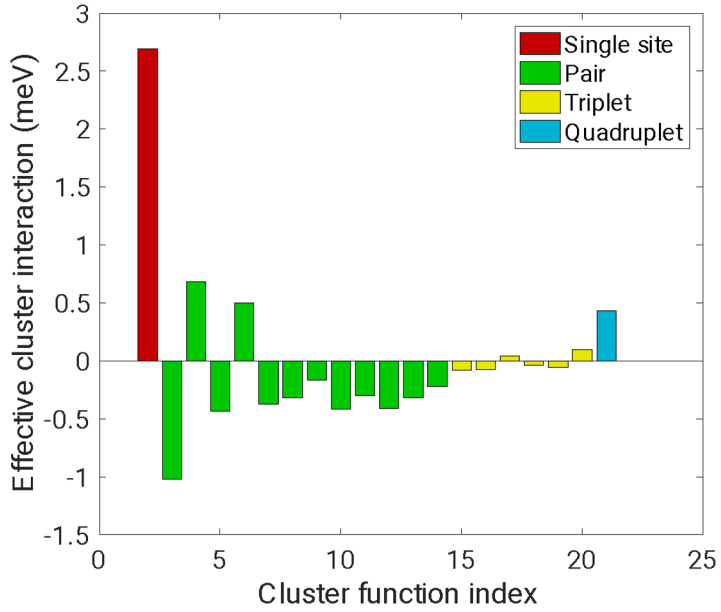
$$\Phi_\beta = \sigma_l \sigma_k$$

$$\Phi_\gamma = \sigma_m \sigma_n \sigma_o$$

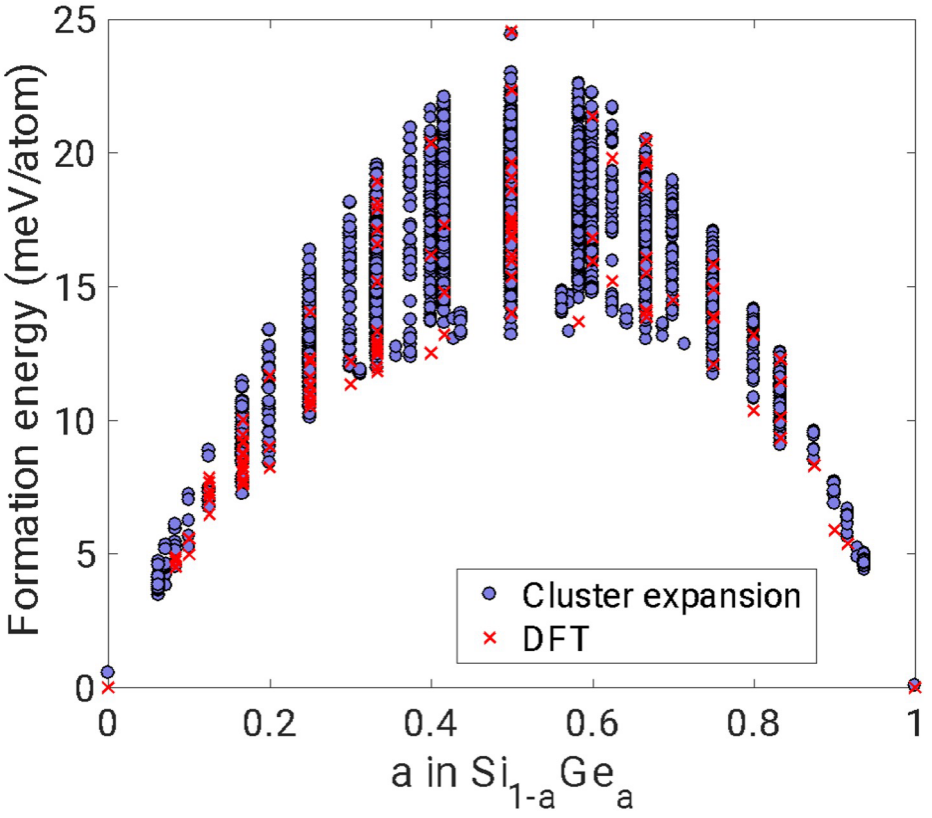
$$E(\vec{\sigma}) = V_o + \sum_i V_i \sigma_i + \sum_{i,j} V_{i,j} \sigma_i \sigma_j + \sum_{i,j,k} V_{i,j,k} \sigma_i \sigma_j \sigma_k + \ldots$$



Exercise: Using a varying number of terms

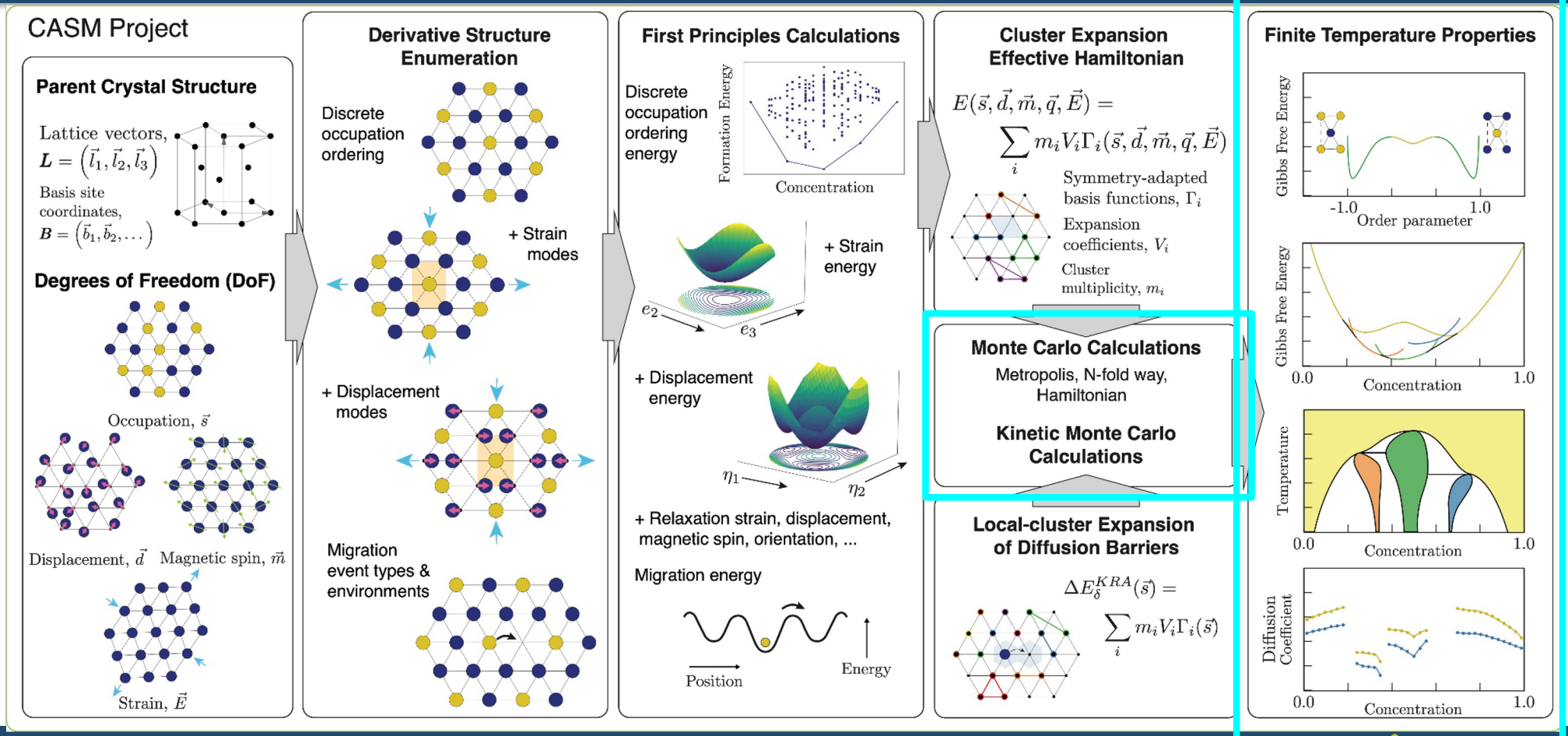


$$E(\vec{\sigma}) = V_o + \sum_i V_i \sigma_i + \sum_{i,j} V_{i,j} \sigma_i \sigma_j + \sum_{i,j,k} V_{i,j,k} \sigma_i \sigma_j \sigma_k + \ldots$$



PRISMS

Monte Carlo calculation & Finite Temp. Properties



Puchala, et al. Comp. Mater. Sci. 217, 111987 (2023).

PRISMS

Binary alloy, semi-grand canonical ensemble

Partition function: $Z = \sum_{\vec{\sigma}} e^{-\beta\Omega(\vec{\sigma})}$

$$\Omega(\vec{\sigma}) = E(\vec{\sigma}) - \Delta\mu N_B(\vec{\sigma})$$

Microstate formation energy

$$\Delta\mu = \mu_B - \mu_A$$

Free energy:

$$\beta\Phi = -\ln(Z)$$

$$\Phi = G - \Delta\mu N_B$$

Binary alloy, semi-grand canonical free energy

Gibbs free energy

Monte Carlo Output

Binary alloy, semi-grand canonical ensemble

Thermodynamic variables (Input): $\Delta\mu, T$

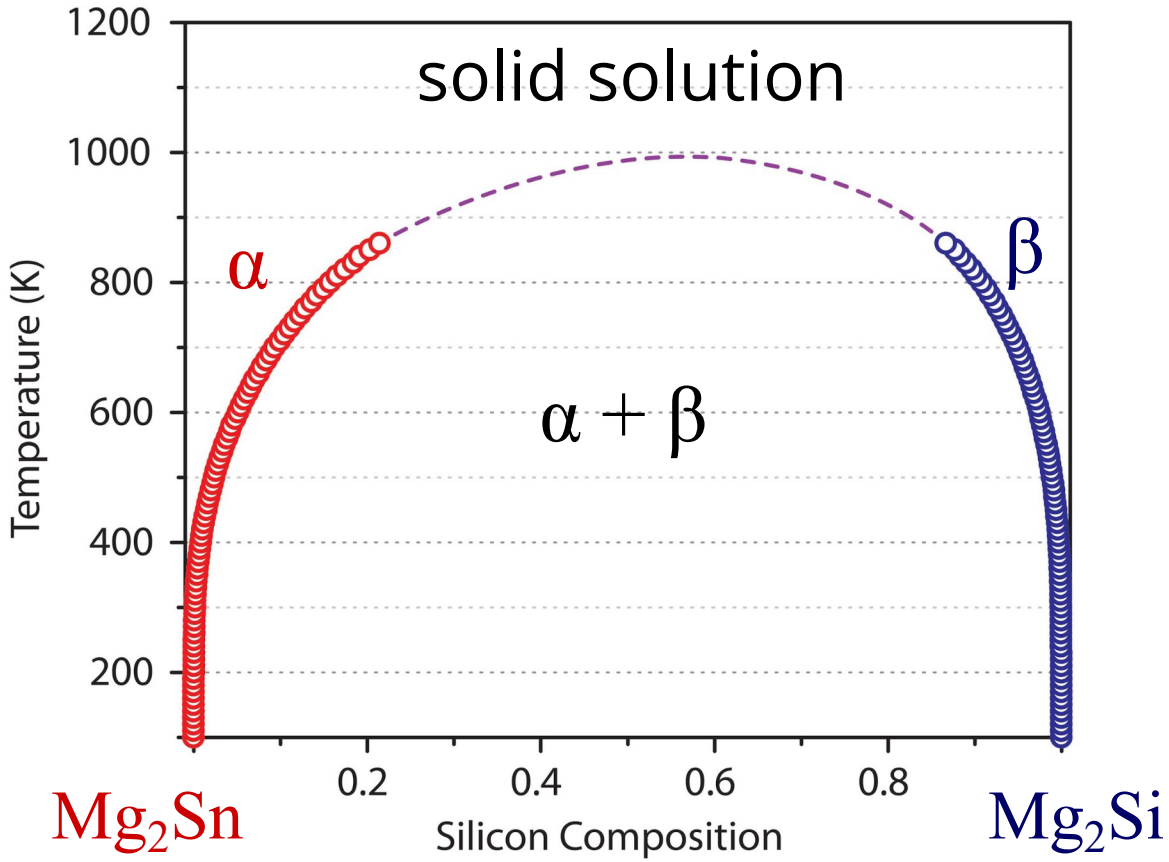
Ensemble averages (Output): $\overline{\Omega} = \sum_{\vec{\sigma}} P(\vec{\sigma})\Omega(\vec{\sigma})$

$$\overline{N}_B = \sum_{\vec{\sigma}} P(\vec{\sigma}) N_B(\vec{\sigma})$$

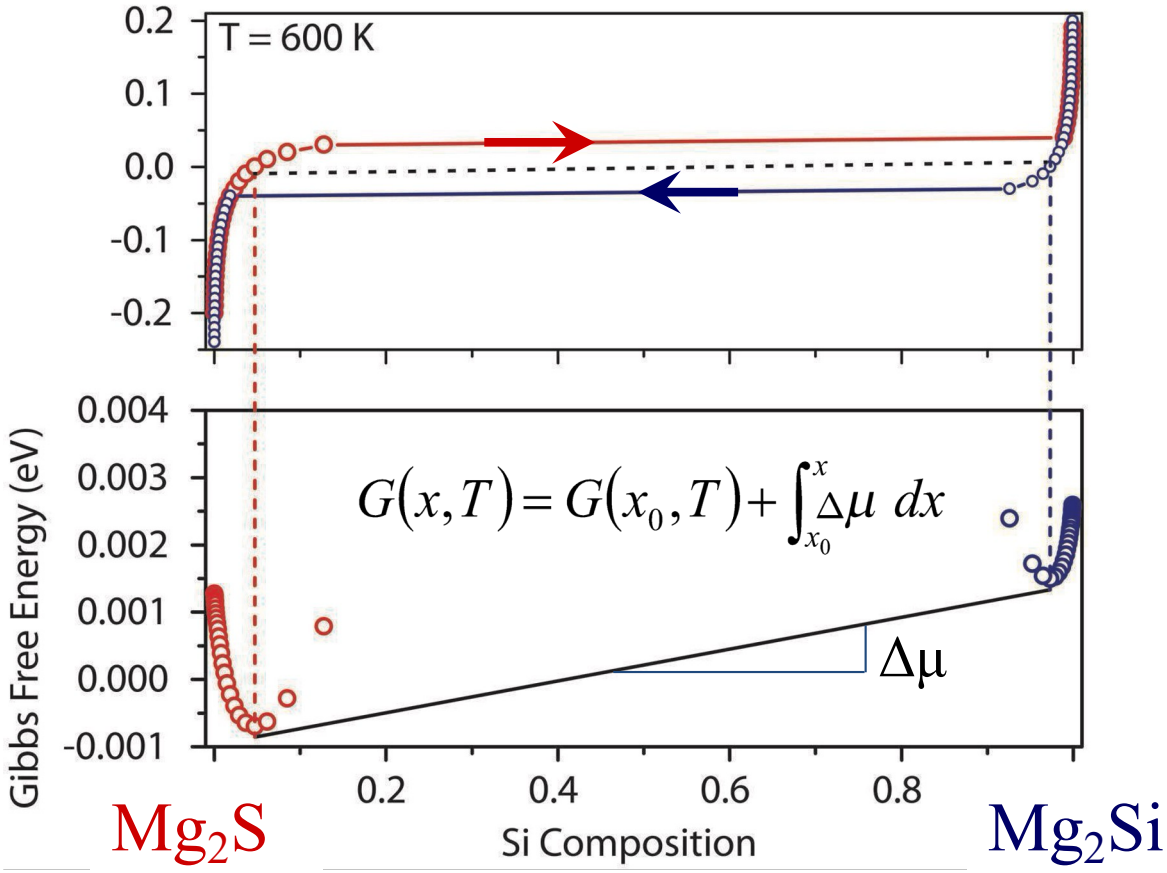
$$x = \overline{N}_B / N$$

Also measure correlations, order parameters, heat capacity, susceptibility, …

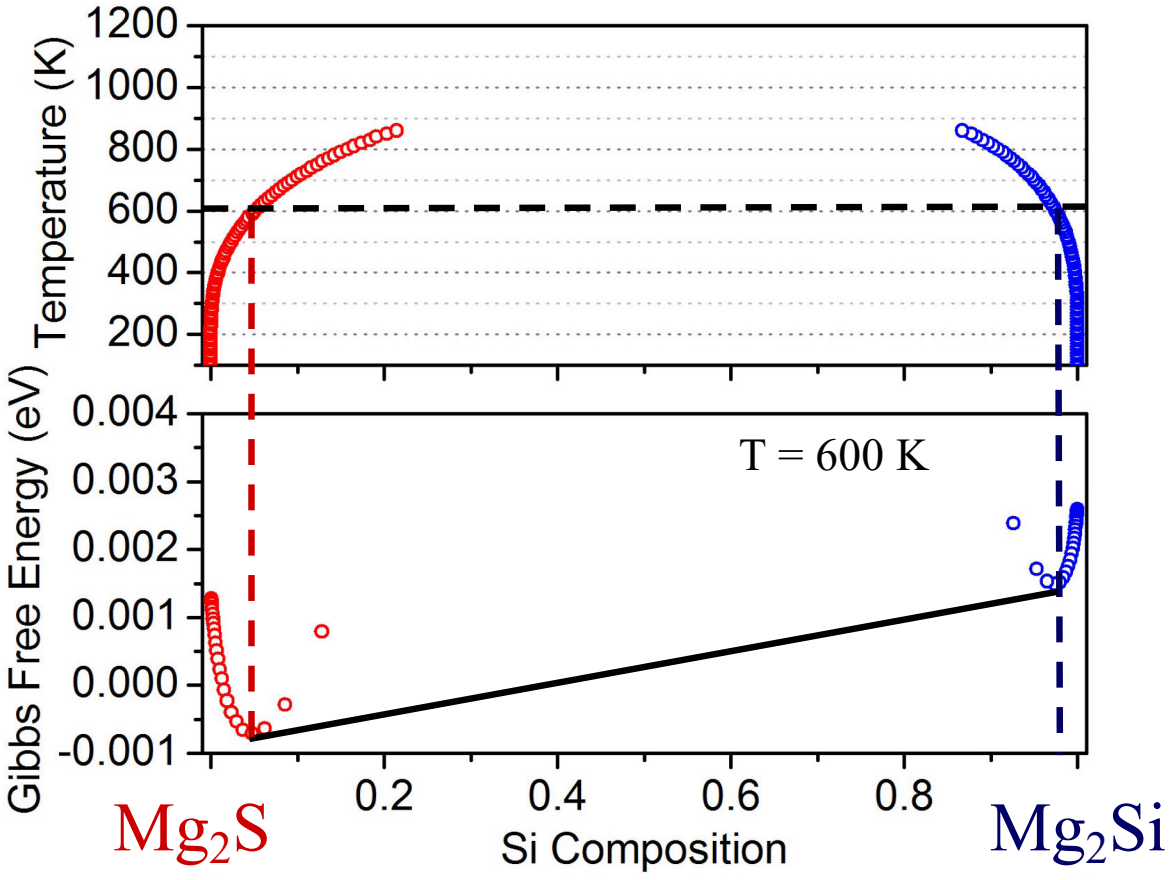
Example: miscibility gap $Mg_2Sn_{1-x}Si_x$



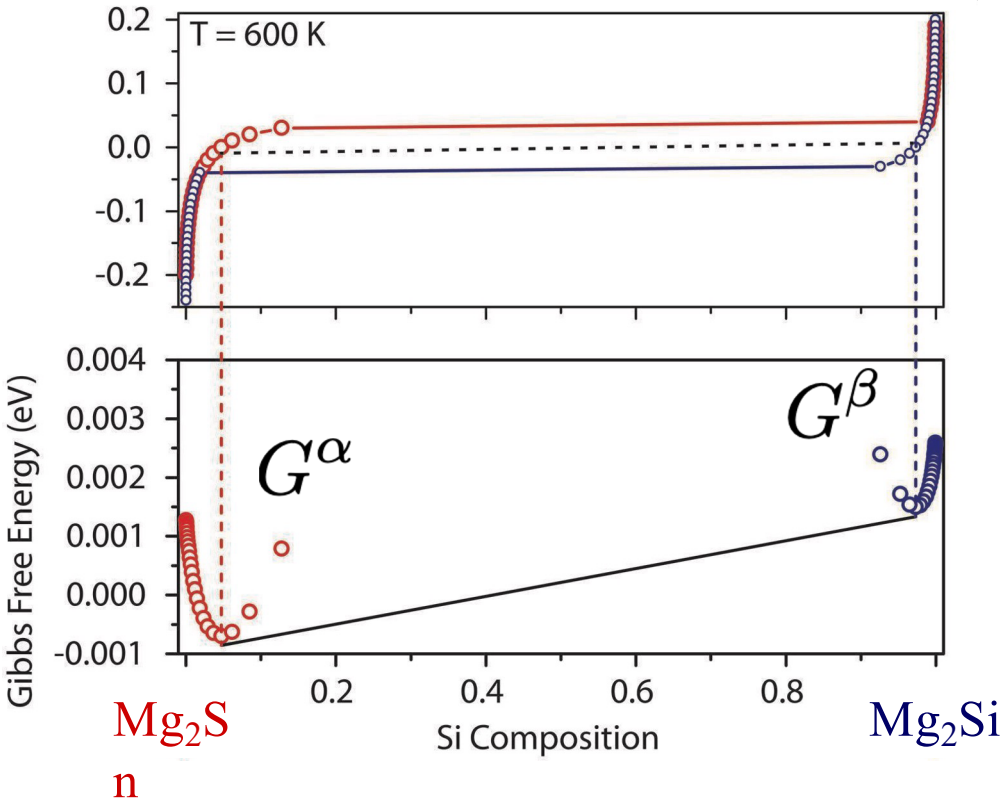
Chemical Potential and Free Energy



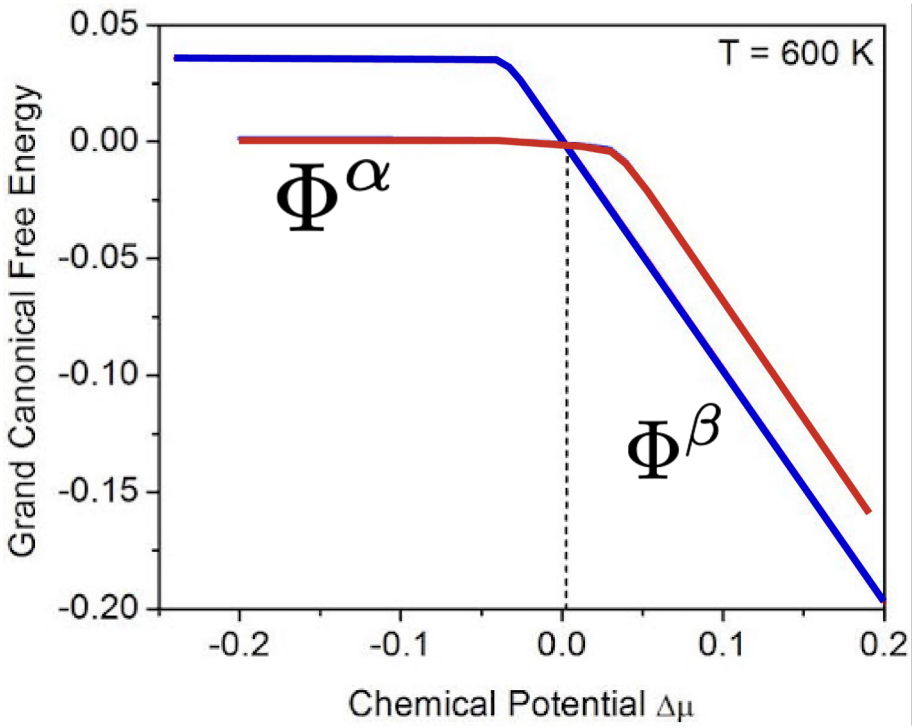
Drawing the Phase Diagram



In terms of grand canonical free energies

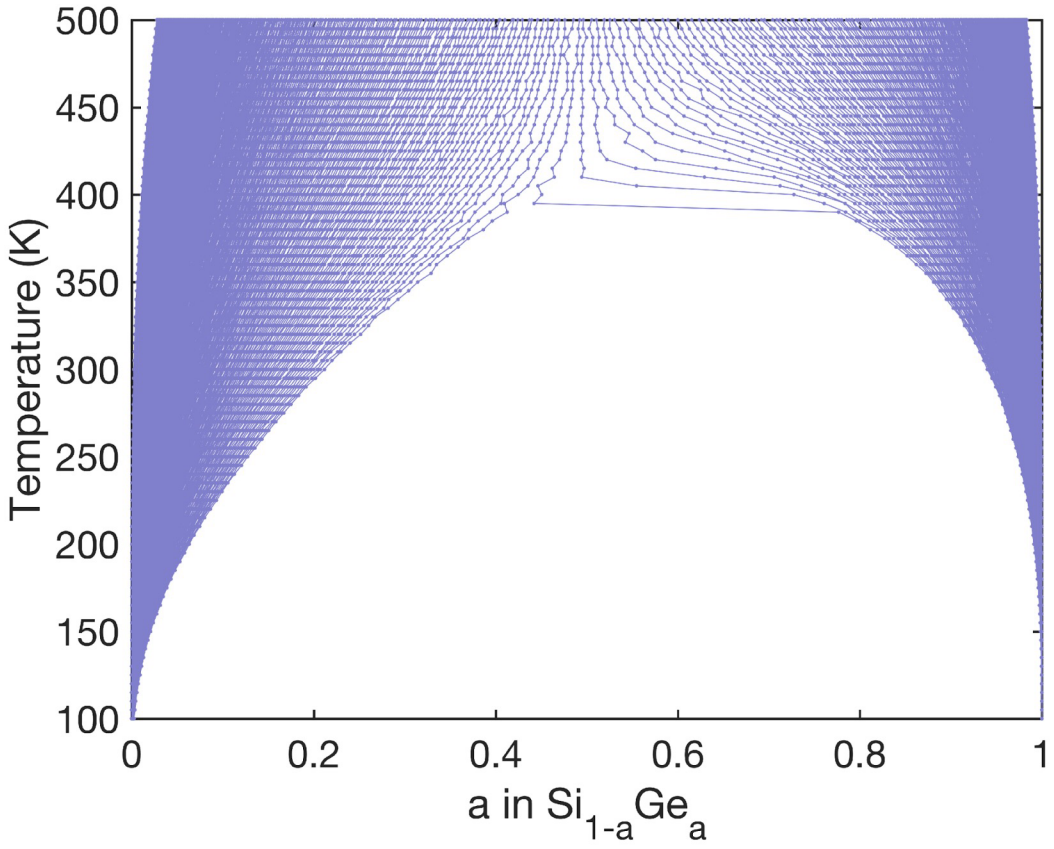


$$\Phi = G - \Delta\mu N_{Si}$$

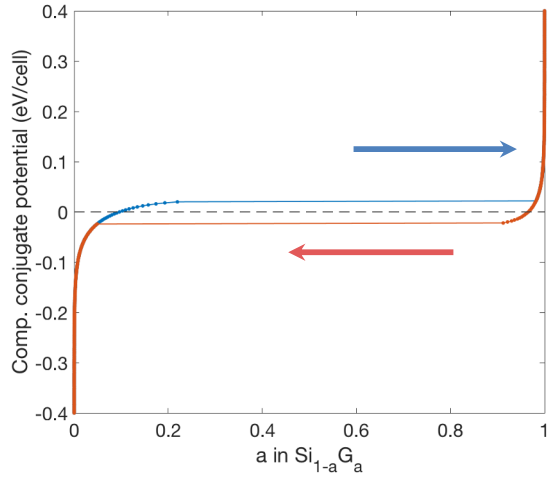


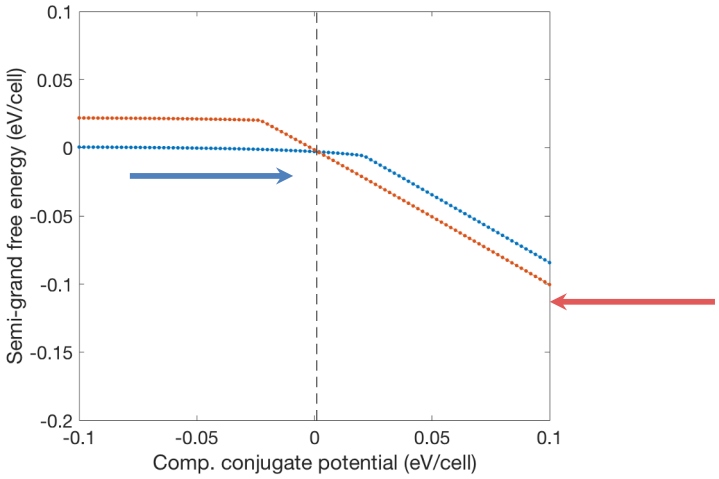
Phase diagram construction

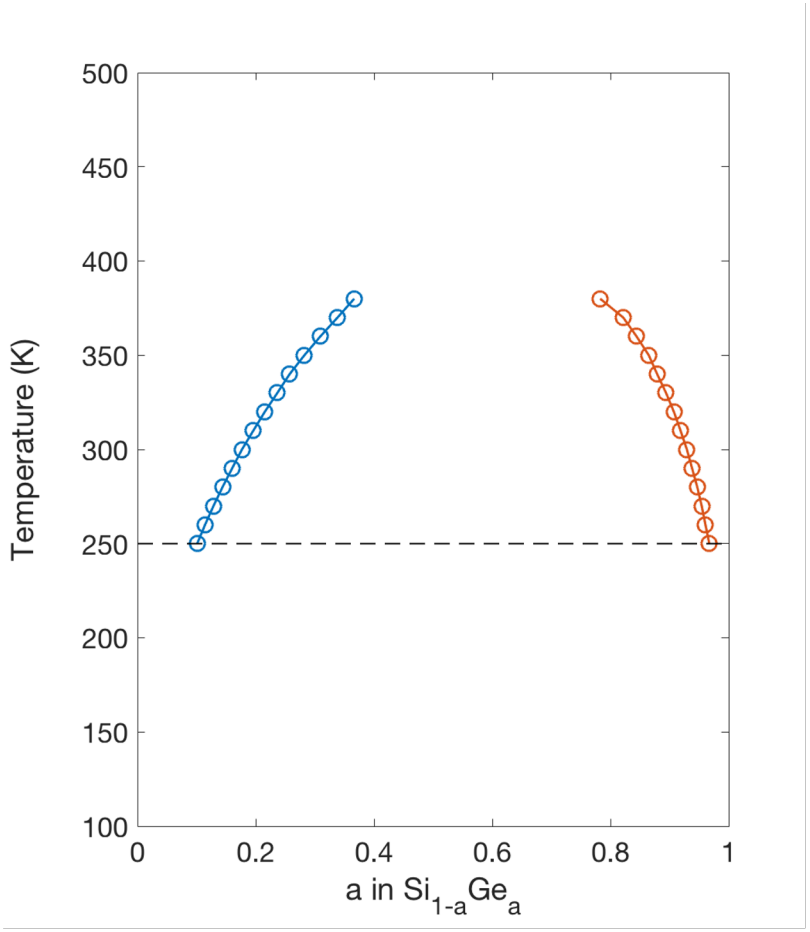
Cooling runs at constant chemical-potential reveal rough shape of miscibility gap



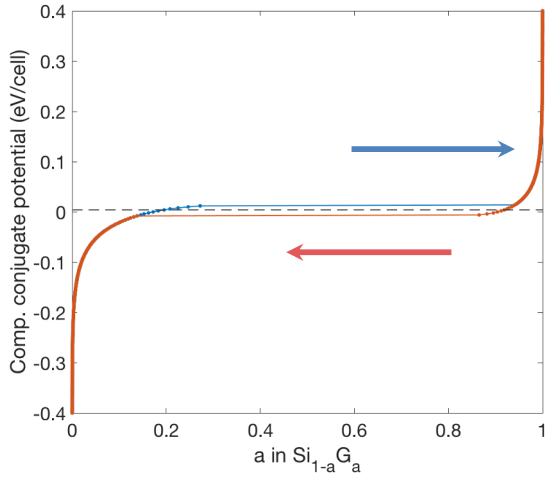
Phase diagram construction

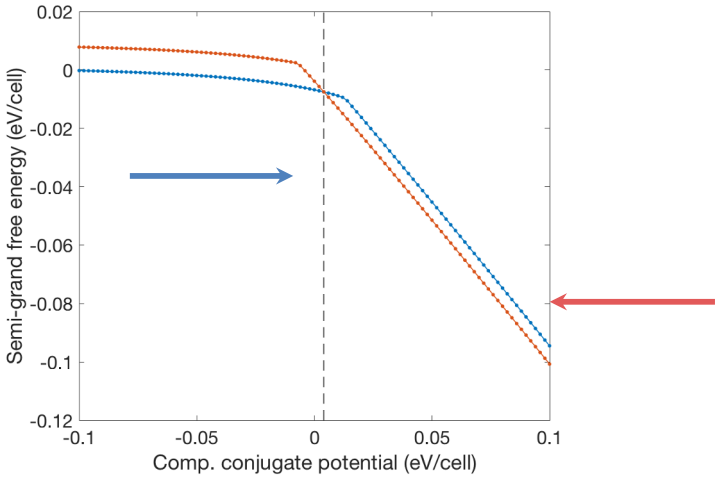


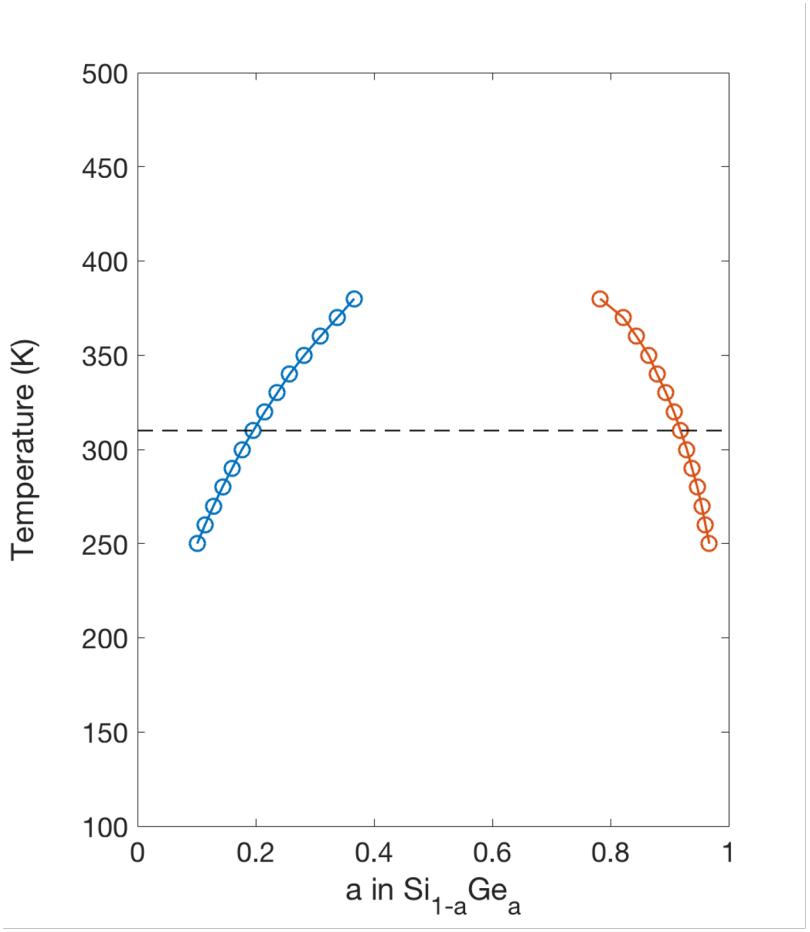




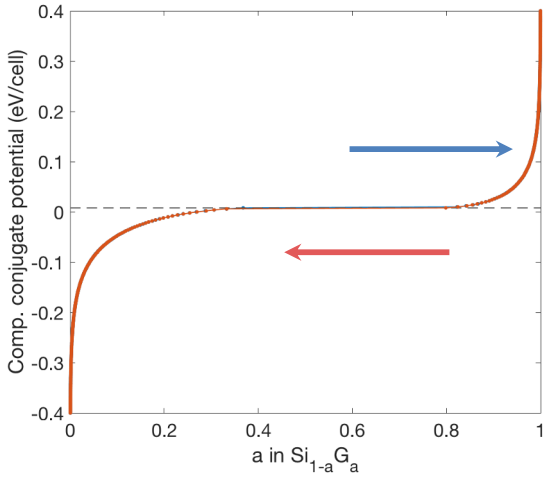
Phase diagram construction

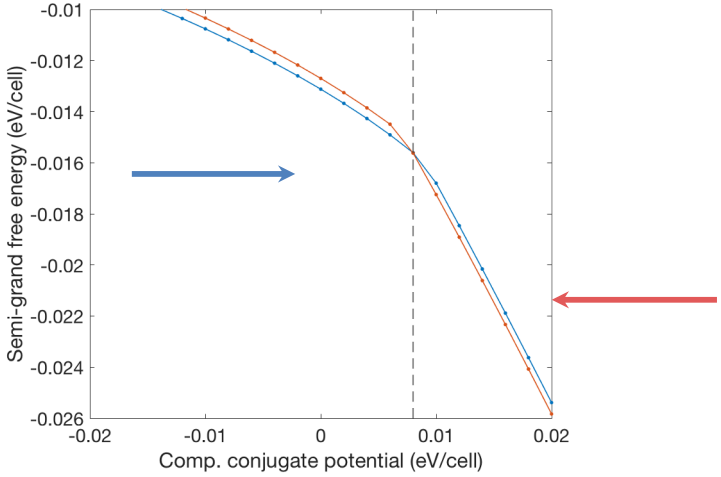


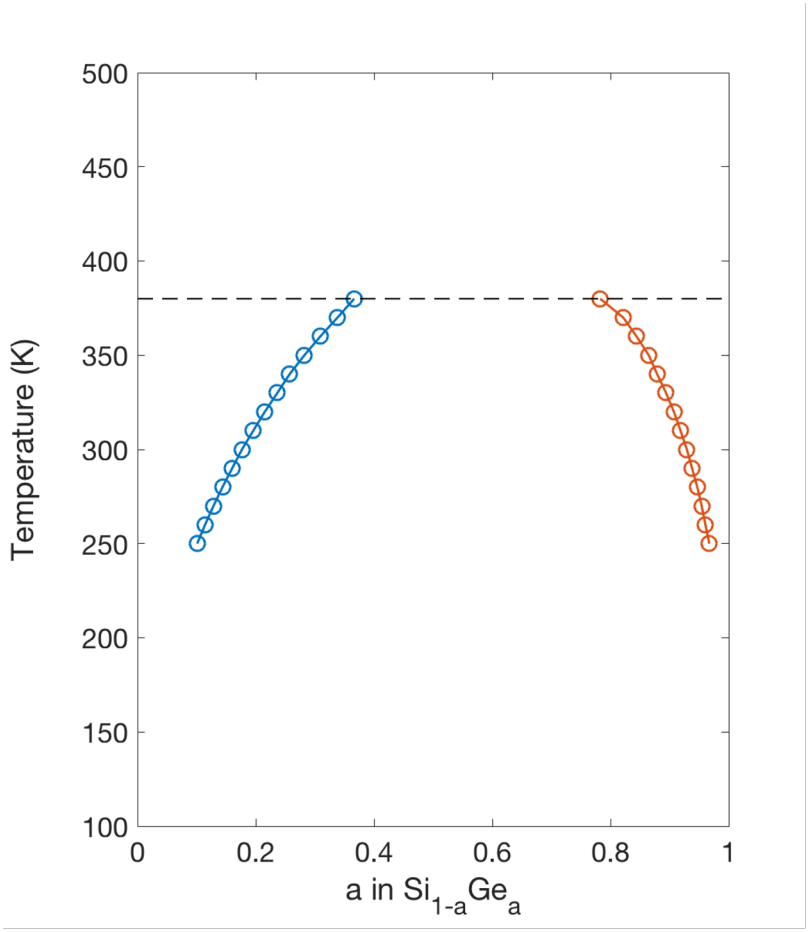




Phase diagram construction

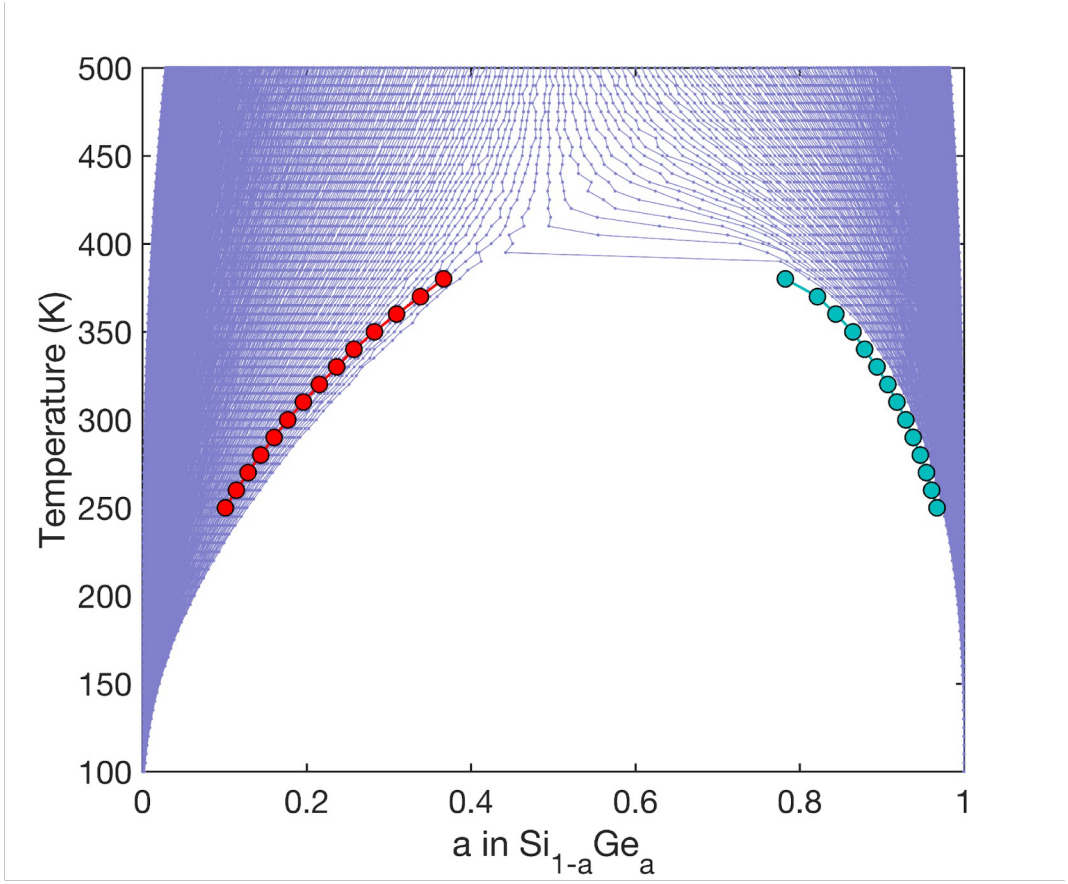






Phase diagram construction

Cooling runs at constant chemical-potential reveal rough shape of miscibility gap

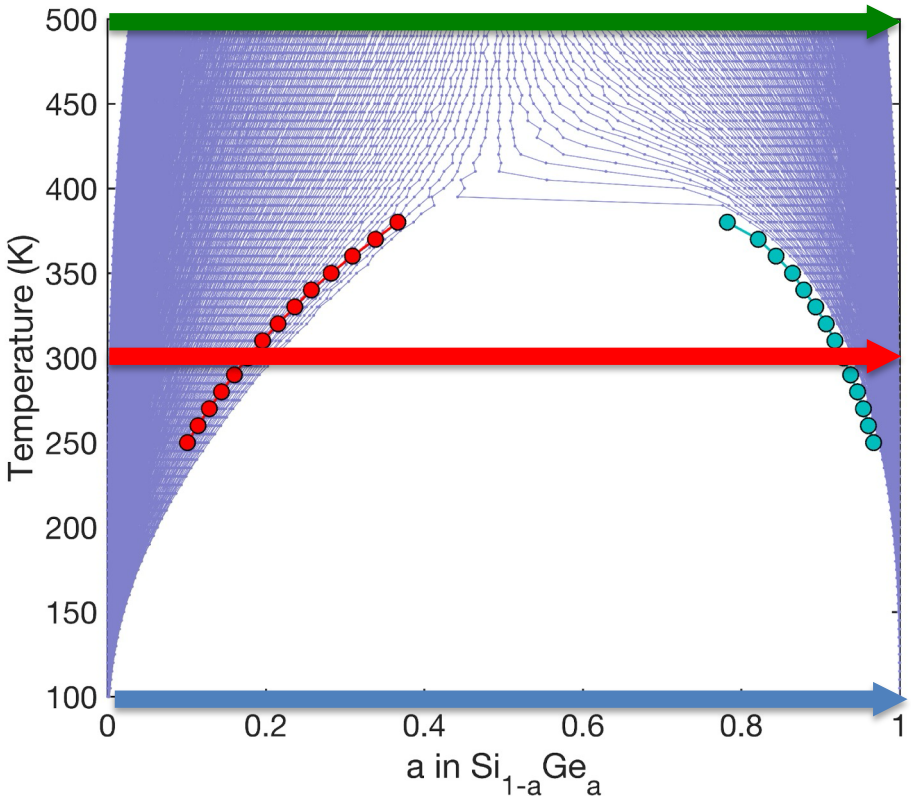


Exercise: Run at T=100, 300, 600 K

T=600

T=300

T=100



Monte Carlo Results