PRISMS Center

Center for PRedictive Integrated Structural Materials Science

CASM Tutorial – Section 2

Brian Puchala, John C. Thomas, Anton Van der Ven

Anirudh Natarajan, John Goiri, Min-Hua Chen,
Max Radin, Jonathon Bechtel, Elizabeth Decolvenaere, Anna Belak,
Naga Sri Harsha Gunda, Sanjeev Kolli

TMS ICME 2017
Thurs. May 24, 2017
Website: prisms-center.org

UCSB

U.S. DEPARTMENT OF ENERGY | Office of Science

Center for PRedictive Integrated Structural Materials Science

PRISMS

CASM Tutorial Outline

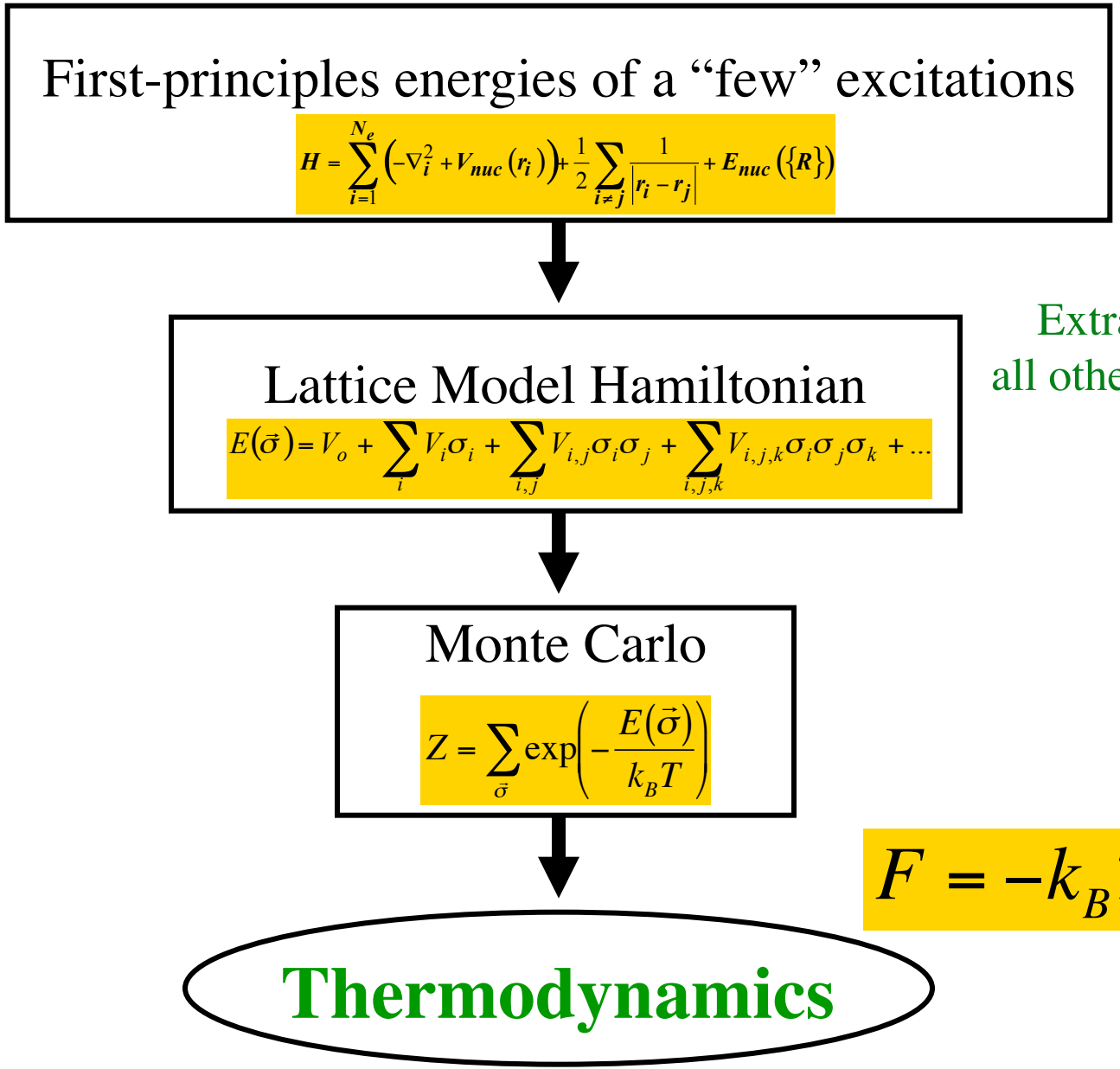
1:30 – 3:00:

- Crystal structure identification & specification
- Symmetry analysis
- Using CASM to manage data
  - Configuration selections and query syntax
  - Maintaining sanity with cluster expansion profiles
- Cluster expansion basics
  - Configuration enumeration
  - Cluster expansion basis sets & correlation evaluation

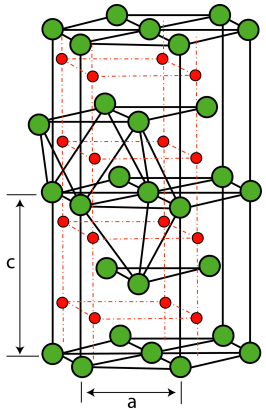
3:30 – 5:00:

- Parameterizing cluster expansions
- Predicting thermodynamic properties with grand canonical Monte Carlo
- Free energy integration and phase diagram construction

CASM: Clusters Approach to Statistical Mechanics



Zr-O binary system



Zr:

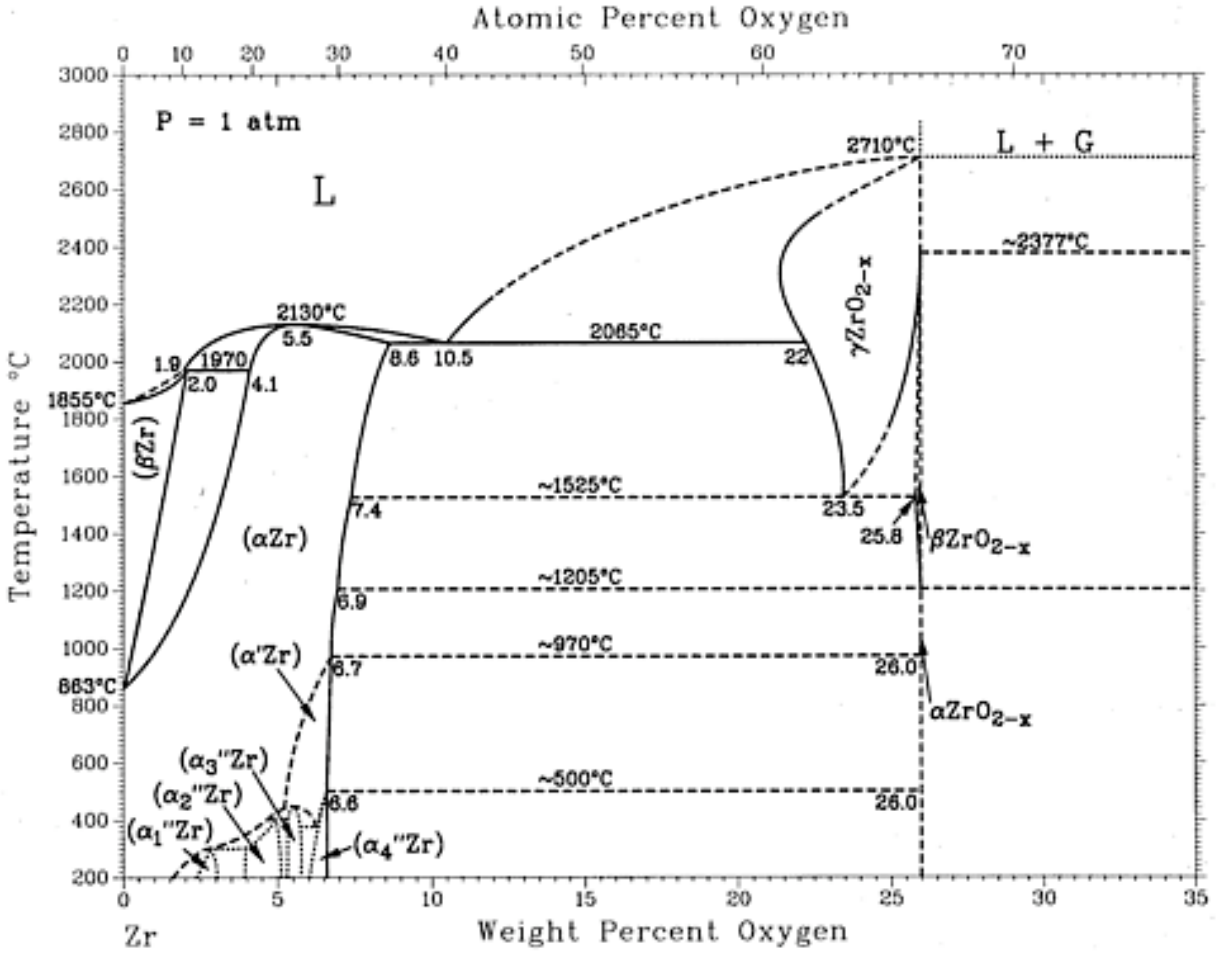
- Very high O solubility

ZrO_x

- Series of suboxide phases

ZrO_{2-x}

- Oxide



J.P. Abriata, J. Garcés, and R. Versaci, Bulletin of Alloy Phase Diagrams Vol. 7 No. 2 1986 .

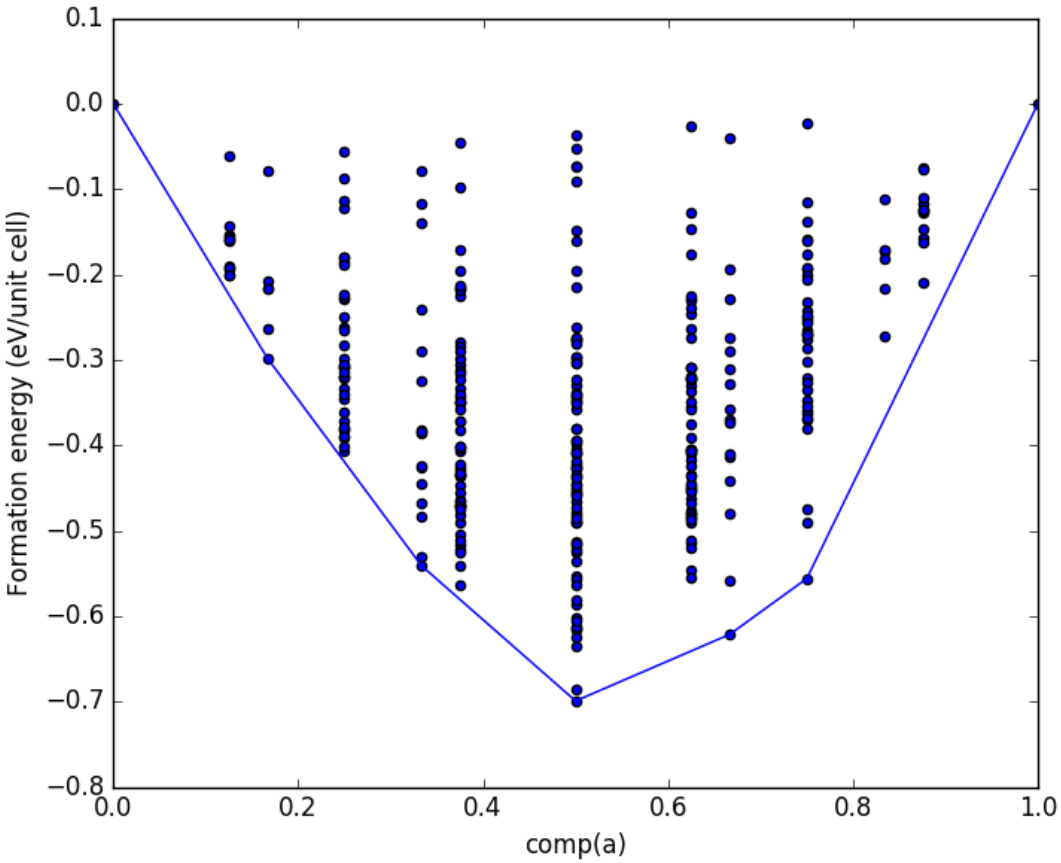
Query calculation results

- 'casm query –c CALCULATED –k 'comp formation_energy hull_dist(CALCULATED)'
  - ––output <filename> to output to file

# configname	selected	comp(a)	formation_energy	hull_dist(CALCULATED,atom_frac)
SCEL1_1_1_1_0_0_0/0	1	0.00000000	-0.00000000	0.00000000
SCEL1_1_1_1_0_0_0/1	1	0.50000000	-0.49027640	0.06970721
SCEL1_1_1_1_0_0_0/2	1	1.00000000	-0.00000000	0.00000000
SCEL2_1_1_2_0_0_0/0	1	0.25000000	-0.12190088	0.11903972
SCEL2_1_1_2_0_0_0/1	1	0.50000000	-0.09185887	0.20251305
SCEL2_1_1_2_0_0_0/2	1	0.75000000	-0.19365904	0.10354095
SCEL2_1_2_1_0_0_0/0	1	0.25000000	-0.33262694	0.03474930
SCEL2_1_2_1_0_0_0/1	1	0.50000000	-0.27791027	0.14049591
SCEL2_1_2_1_0_0_0/2	1	0.75000000	-0.24317736	0.08939286
SCEL2_1_2_1_0_0_0/3	1	0.50000000	-0.63584603	0.02118400
SCEL2_1_2_1_1_0_0/0	1	0.25000000	-0.37224950	0.01890027
SCEL2_1_2_1_1_0_0/1	1	0.50000000	-0.52465847	0.05824651
SCEL2_1_2_1_1_0_0/2	1	0.75000000	-0.47426092	0.02336898
SCEL3_1_1_3_0_0_0/0	1	0.16666667	-0.07860740	0.09400358
SCEL3_1_1_3_0_0_0/1	1	0.33333333	-0.24026118	0.11279629
SCEL3_1_1_3_0_0_0/2	1	0.33333333	-0.07917269	0.17320448
SCEL3_1_1_3_0_0_0/3	1	0.50000000	-0.05205108	0.21578231

Plot calculation results, via Python

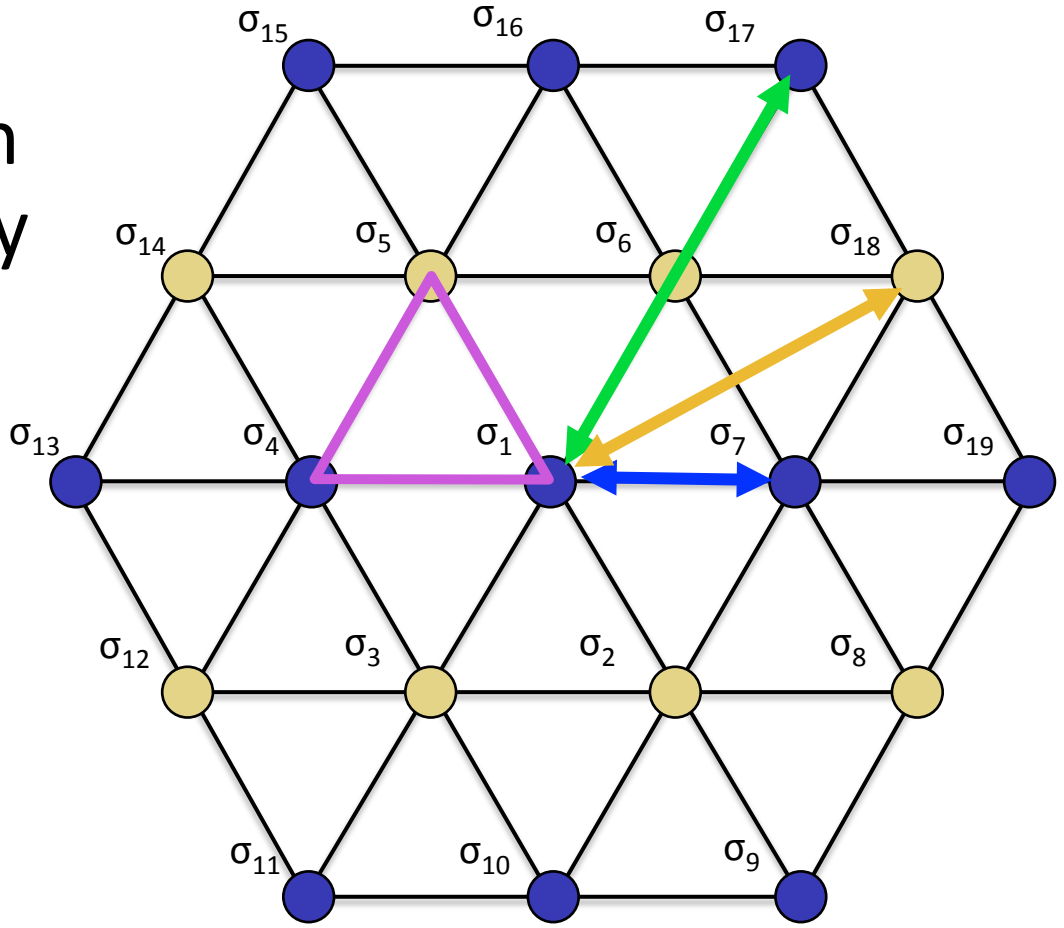
```
from casm.project import Project, Selection
import matplotlib.pyplot as plt

# construct a 'Selection'
proj = Project()
sel = Selection(proj, 'CALCULATED', all=False)

# query results into a pandas.DataFrame
comp = 'comp(a)'
Ef = 'formation_energy'
hull_dist = 'hull_dist(CALCULATED,atom_frac)'
sel.query([comp, Ef, hull_dist])

# get convex hull configurations,
# sorted for nice plotting
df = sel.data.sort_values([comp])
hull_tol = 1e-6
df_hull = df[df[hull_dist] < hull_tol]
print df_hull.to_string()

# plot formation energies with convex hull
plt.scatter(sel.data[comp], sel.data[Ef],
  marker='o')
plt.plot(df_hull[comp], df_hull[Ef], 'b.-')
plt.xlabel(comp)
plt.ylabel('Formation energy (eV/unit cell)')
plt.xlim([0.,1.])
plt.show()
```



The Cluster Expansion

Interactions between sites are described by products of σ_i.

Fit interaction parameters to a database of DFT energies.

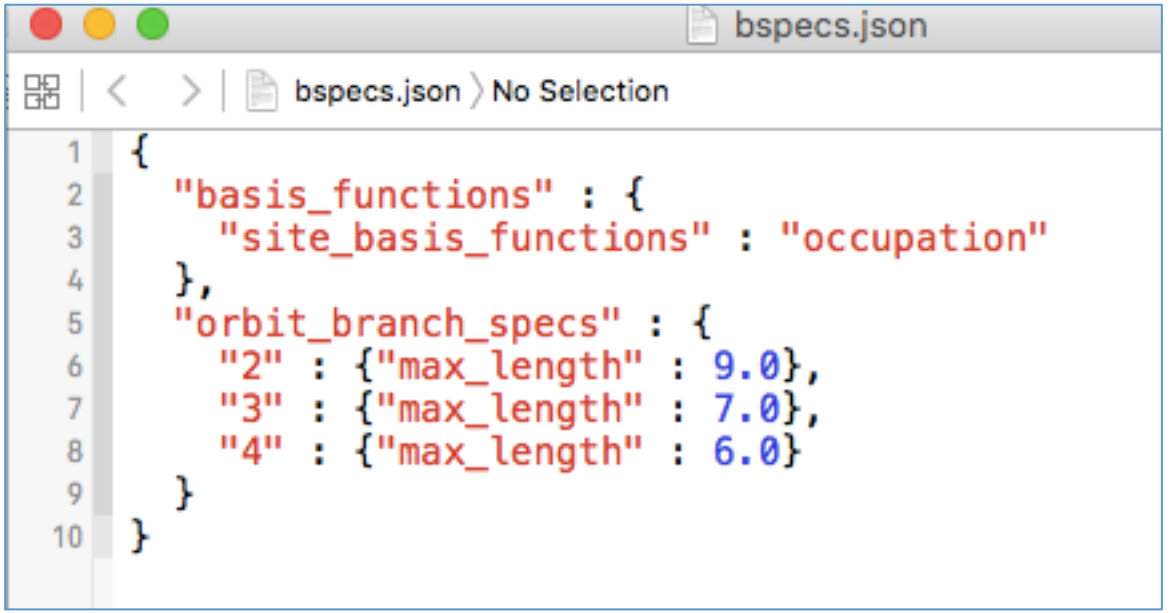


$$E = J_1\sigma_1 + J_{1,7}\sigma_1\sigma_7 + J_{1,18}\sigma_1\sigma_{18} + J_{1,17}\sigma_1\sigma_{17} + J_{1,4,5}\sigma_1\sigma_4\sigma_5 + \ldots$$

Then use symmetry to average over the entire crystal

The bspecs.json file

- Specifications for generating the cluster expansion basis functions



Viewing cluster basis functions

- `'casm bset –functions'` will print basis function formulas

```
** Branch 2 **
     ** 2 of 74 Orbits **  Orbit: 2 0  Points: 2  Mult: 2  MinLength: 2.5843392  MaxLength: 2.5843392
          Prototype of 2 Equivalent Clusters in Orbit 2
                     0.3333333    0.6666666    0.7500000 Va O  basis_index: 3  clust_index: 0
                     0.3333333    0.6666666    0.2500000 Va O  basis_index: 2  clust_index: 1

          Basis Functions:
            \Phi_2 = \phi_3_0(s_0)\phi_2_0(s_1)


     ** 3 of 74 Orbits **  Orbit: 2 1  Points: 2  Mult: 6  MinLength: 3.2339869  MaxLength: 3.2339869
          Prototype of 6 Equivalent Clusters in Orbit 3
                     0.3333333    0.6666666    0.7500000 Va O  basis_index: 3  clust_index: 0
                    -0.6666667   -0.3333334    0.7500000 Va O  basis_index: 3  clust_index: 1

          Basis Functions:
            \Phi_3 = \phi_3_0(s_0)\phi_3_0(s_1)
```

Query correlations

- `'casm query -k corr(0:5)'`
  - will print correlations, $\vec{\Gamma}^{(i)}$, the symmetry-invariant basis function averages

# configname	selected	corr(0)	corr(1)	corr(2)	corr(3)	corr(4)	corr(5)
SCEL1_1_1_1_0_0_0/0	1	1.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
SCEL1_1_1_1_0_0_0/1	1	1.00000000	0.50000000	0.00000000	0.50000000	0.00000000	0.50000000
SCEL1_1_1_1_0_0_0/2	1	1.00000000	1.00000000	1.00000000	1.00000000	1.00000000	1.00000000
SCEL2_1_1_2_0_0_0/0	1	1.00000000	0.25000000	0.00000000	0.25000000	0.00000000	0.00000000
SCEL2_1_1_2_0_0_0/1	1	1.00000000	0.50000000	0.25000000	0.50000000	0.25000000	0.00000000
SCEL2_1_1_2_0_0_0/2	1	1.00000000	0.75000000	0.50000000	0.75000000	0.50000000	0.50000000
SCEL2_1_2_1_0_0_0/0	1	1.00000000	0.25000000	0.00000000	0.08333333	0.00000000	0.25000000
SCEL2_1_2_1_0_0_0/1	1	1.00000000	0.50000000	0.50000000	0.16666667	0.16666667	0.50000000
SCEL2_1_2_1_0_0_0/2	1	1.00000000	0.75000000	0.50000000	0.58333333	0.50000000	0.75000000
SCEL2_1_2_1_0_0_0/3	1	1.00000000	0.50000000	0.00000000	0.16666667	0.33333333	0.50000000
SCEL2_1_2_1_1_0_0/0	1	1.00000000	0.25000000	0.00000000	0.08333333	0.00000000	0.00000000
SCEL2_1_2_1_1_0_0/1	1	1.00000000	0.50000000	0.25000000	0.16666667	0.25000000	0.00000000
SCEL2_1_2_1_1_0_0/2	1	1.00000000	0.75000000	0.50000000	0.58333333	0.50000000	0.50000000
SCEL3_1_1_3_0_0_0/0	1	1.00000000	0.16666667	0.00000000	0.16666667	0.00000000	0.00000000
SCEL3_1_1_3_0_0_0/1	1	1.00000000	0.33333333	0.00000000	0.33333333	0.00000000	0.16666667
SCEL3_1_1_3_0_0_0/2	1	1.00000000	0.33333333	0.16666667	0.33333333	0.16666667	0.00000000
SCEL3_1_1_3_0_0_0/3	1	1.00000000	0.50000000	0.33333333	0.50000000	0.33333333	0.16666667
SCEL3_1_1_3_0_0_0/4	1	1.00000000	0.66666667	0.33333333	0.66666667	0.33333333	0.50000000
SCEL3_1_1_3_0_0_0/5	1	1.00000000	0.33333333	0.00000000	0.33333333	0.00000000	0.00000000
SCEL3_1_1_3_0_0_0/6	1	1.00000000	0.50000000	0.16666667	0.50000000	0.16666667	0.16666667

- Next step: Fit cluster expansion paramters, $\vec{V}$, to the DFT calculated energies

$$H\left(\vec{\Gamma}^{(i)}\right) = \vec{V}^T \vec{\Gamma}^{(i)}$$

casm-learn

- Acts as a wrapper around 'scikit-learn', the Python machine learning package, and incorporates some additional features:
  - Genetic algorithms, via 'deap', the Distributed Evolutionary Algorithm Package
  - Stores and compares potential results in a 'Hall Of Fame'
  - Convex hull checking
  - Generate casm selection files of configurations that you might want to calculate based on the cluster expansion
  - Writes 'eci.json' file containing calculated parameters

scikit-learn: http://scikit-learn.org
deap: http://deap.readthedocs.io

casm-learn: getting started

- `'casm-learn --help'`
  - Print available options
- `'casm-learn --desc'`
  - Print overall usage description
- `'casm-learn --exSomething'`
  - Print example input files
- `'casm-learn -settings-format'`
  - Print input file help
- `'casm-learn -s <filename>'`
  - Run, using <filename> as the input settings file

Step 1) Problem specification

- Solve X*b = y for b
  - X: 2d matrix of correlations
  - y: 1d vector of formation energies (or other property)
  - b: 1d vector, of fitting parameters, the effective clusters interactions (ECI)
- Weighted least squares
  - X*b = y → L*X*b = L*y,
  - W = L*L^T, W typically being a diagonal matrix where the entries along the diagonal are the weights associated with each configuration

Step 1) Problem specification

- Cross-validation:
  - Seperate data into several training and testing sets
  - For each training set, solve $X^{train}*b = y^{train}$ for b
  - Use b to predict $y^{test} = X^{test}*b$ for the remaining data
  - Calculate cross-validation score, root average MSE over test sets:

$$CV = \sqrt{\frac{\sum_{j}^{k}\left(\frac{1}{n}\sum_{i}^{n}\left(\hat{y}_i^{(j)} - y_i^{(j)}\right)^2\right)}{k}} + p * \left\|b\right\|_0$$

  - 'casm-learn' can optionally add an additional 'penalty', p, to prefer results with fewer terms

Step 1) Problem specification

- Cross-validation schemes:
  - KFold cross validation: split data into 'k' "folds"
    - Train the model on k-1 folds
    - Test on the remaining fold
    - CV score = Average score over each test fold
  - LeaveOneOut: k = number of configurations
  - ShuffleSplit:
    - Create 'n_iter' random splits of the data, with user control of the fraction of configurations in the train/ test sets
  - Others:
    - http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

Step 1) Problem specification

```
"problem_specs": {
  "data": {
    "y": "formation_energy",
    "X": "corr",
    "kwargs": null,
    "type": "selection",
    "filename": "train"
  },
  "cv": {
    "penalty": 0.0,
    "method": "KFold",
    "kwargs": {
      "n_folds": 10,
      "shuffle": true
    }
  },
  "weight": {
    "method": "wHullDist",
    "kwargs": {
      "A": 9.0,
      "B": 1.0,
      "kT": 0.05
    }
  }
},
```

name of a 'casm select' output file, or "CALCULATED", or "MASTER",

optional penalty per non-zero eci, as documented in 'casm-learn –settings-format'

scikit-learn class name and keyword arguments, as documented on scikit-learn.org

weighting method name and keyword arguments, as documented in 'casm-learn –settings-format'

Step 2) Estimator and Feature Selection Methods

- Estimator:
  - determines how to solve for b, given X and y
  - Examples: LinearRegression, Lasso, Ridge
  - http://scikit-learn.org/stable/modules/linear_model.html
- Feature Selection:
  - Determines which features (basis functions) to include in the model (which subset of b_i are non-zero)
  - Examples: GeneticAlgorithm, RFE (recursive feature elimination), SelectFromModel (used with Lasso, for example)
  - http://scikit-learn.org/stable/modules/feature_selection.html

Step 2) Estimator and Feature Selection Methods

```
"estimator": {
  "method": "LinearRegression"
},
"feature_selection": {
  "method": "GeneticAlgorithm",
  "kwargs": {
    "constraints_kwargs": {
      "fix_on": [],
      "n_features_min": 5,
      "fix_off": [],
      "n_features_max": "all"
    },
    "cxUniformProb": 0.5,
    "mutFlipBitProb": 0.01,
    "evolve_params_kwargs": {
      "n_population": 100,
      "n_generation": 10,
      "n_repetition": 5,
      "n_halloffame": 50,
      "n_features_init": 5
    },
    "selTournamentSize": 3
  }
},
```

scikit-learn class name and keyword arguments, as documented on scikit-learn.org

scikit-learn class name or casm.learn.evolve class name and keyword arguments, as documented on scikit-learn.org and in 'casm-learn – settings-format'

Step 2) Estimator and Feature Selection Methods

- Run 'casm-learn': 'casm-learn -s fit_1.json'
  - First time: stores problem specs in 'fit_1_specs.pkl'
  - First time, and subsequent times: Runs and stores results in 'fit_1_halloffame.pkl'
- For a single problem specification, you may run repeatedly with the same or different estimator and feature selection methods
  - Individual results with best CV scores stored in hall of fame
- View results: 'casm-learn -s fit_1.json --hall'
  - '--format type', with type=details, json, csv
  - '--indiv 0 1 ...', to only print particular individuals in the hall of fame

Step 3 / 4) Analyze Results, Run Monte Carlo

- 'casm-learn -s fit_1.json --checkhull'
  - Check if predicted ground state configurations agree with the DFT calculated ground state configurations
  - Print configuration selection files with configurations that you may want to calculate
- Compare results from different problem specs
  - add more calculated configurations
  - select different training configurations
  - different weighting schemes
  - different CV schemes
- Compare Monte Carlo results
  - 'casm-learn –s fit.json –select 0' to write 'eci.json' file with ECI to use to evaluate the cluster expansion
  - Run Monte Carlo calculations and check phase stability predictions

The 'basis.json'/'eci.json' file

- Contains descriptions of the site basis functions (not shown) and cluster basis functions
- The 'basis.json' file is generated when using 'casm bset –update'
- The 'eci.json' file is a copy of the 'basis.json' file with "eci": value added to cluster functions with non-zero eci.


```
"cluster_functions": [
  {
    "prototype_function": "1",
    "prototype": {
      "min_length": 0.0,
      "max_length": 0.0,
      "sites": []
    },
    "mult": 1,
    "linear_function_index": 0,
    "orbit": [0, 0, 0]
  },
  {
    "prototype_function": "\\phi_3_0(s_0)",
    "orbit": [1, 0, 0],
    "eci": -1.484351045075109,
    "linear_function_index": 1,
    "prototype": {
      "min_length": 0.0,
      "max_length": 0.0,
      "sites": [
        [3, 0, 0, 0]
      ]
    },
    "mult": 2
  },
```

CASM: Clusters Approach to Statistical Mechanics

First-principles energies of a "few" excitations

$$H = \sum_{i=1}^{N_e}\left(-\nabla_i^2 + V_{nuc}(r_i)\right) + \frac{1}{2}\sum_{i\neq j}\frac{1}{|r_i - r_j|} + E_{nuc}(\{R\})$$

Lattice Model Hamiltonian

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

Extrapolate to all other excitations

Monte Carlo

$$Z = \sum_{\vec{\sigma}} \exp\left(-\frac{E(\vec{\sigma})}{k_B T}\right)$$

$$F = -k_B T \ln(Z)$$

Thermodynamics

Composition conversions

$$\vec{n} = \vec{n}^o + \boldsymbol{M}\vec{x}$$

$$\vec{x} = \boldsymbol{M}^+(\vec{n} - \vec{n}^o)$$

$\vec{n}$: #atoms per unit cell

$\vec{n}^o$: #atoms per unit cell at composition axes origin

$\vec{x}$: formula parameters determined from user specified composition axes

$\boldsymbol{M}$: conversion matrix

$\boldsymbol{M}^+$: left pseudo-inverse $\boldsymbol{M}^+ = (\boldsymbol{M}^\mathsf{T}\boldsymbol{M})^{-1}\boldsymbol{M}^\mathsf{T}$.

Parametric chemical potential

Define:

$$\begin{aligned} \vec{\xi} &\equiv \boldsymbol{M}^{\mathsf{T}}\vec{\mu} \\ \xi^{o} &\equiv \vec{\mu}^{\mathsf{T}}\vec{n}^{o}, \end{aligned}$$

$\vec{\xi}$: parametric chemical potential, along composition axes
- 'param_chem_pot' in casm input / output

ξ^{o} : unit cell reference potential

Example: Binary alloy, no Va

First choice of composition axes:

1) $n_B = x_a$ and $n_A = 1 - x_a$. (Pure A is the origin)

- $\vec{n}^o = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\vec{n}^a = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
- $\boldsymbol{M}^+ = \begin{bmatrix} -0.5 & 0.5 \end{bmatrix}$, $\boldsymbol{M} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$
- $\xi_a = \vec{\mu}^\mathsf{T} \boldsymbol{M}_{:0} = \mu_{n_B} - \mu_{n_A}$.
- $\xi^o = \vec{\mu}^\mathsf{T} \vec{n}^o = \mu_{n_A}$.

Example: Binary alloy, no Va

Second choice of composition axes:

2) $n_A = x_a$ and $n_B = 1 - x_a$. (Pure B is the origin)

- $\vec{n}^o = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $\vec{n}^a = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
- $\boldsymbol{M}^+ = \begin{bmatrix} 0.5 & -0.5 \end{bmatrix}$, $\boldsymbol{M} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$
- $\xi_a = \vec{\mu}^{\mathsf{T}} \boldsymbol{M}_{:0} = \mu_{n_A} - \mu_{n_B}$.
- $\xi^o = \vec{\mu}^{\mathsf{T}} \vec{n}^o = \mu_{n_B}$.

Ternary Alloy

1) $n_A = 1 - x_a - x_b$, $n_B = x_a$, and $n_C = x_b$. (Pure A is the origin)

- $\vec{n}^o = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, $\vec{n}^a = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, $\vec{n}^b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
- $\boldsymbol{M}^+ = \begin{bmatrix} -1/3 & 2/3 & -1/3 \\ -1/3 & -1/3 & 2/3 \end{bmatrix}$, $\boldsymbol{M} = \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$
- $\xi_a = \vec{\mu}^{\mathsf{T}} \boldsymbol{M}_{:0} = \mu_{n_B} - \mu_{n_A}$.
- $\xi_b = \vec{\mu}^{\mathsf{T}} \boldsymbol{M}_{:1} = \mu_{n_C} - \mu_{n_A}$.
- $\xi^o = \vec{\mu}^{\mathsf{T}} \vec{n}^o = \mu_{n_A}$.

2) $n_B = 1 - x_a - x_b$, $n_A = x_a$, and $n_C = x_b$.

3) $n_C = 1 - x_a - x_b$, $n_A = x_a$, and $n_B = x_b$.

Semi-grand canonical ensemble

By first & second laws, the fundamental equation is:

$$dU = TdS + \vec{\mu}^{\mathsf{T}} d\vec{N},$$

with a constraint on the total number of sites:

$$\sum_i N_i = N^{basis} N^{unit}.$$

Characteristic potential in semi-grand canonical ensemble

$$\begin{aligned}\Phi &= U - TS - N^{unit}\vec{\xi}^{\mathsf{T}}\vec{x}. \\ &= U - TS - N^{unit}\vec{\mu}^{\mathsf{T}}(\vec{n} - \vec{n}^{o}).\end{aligned}$$

And partition function:

$$\begin{aligned}Z &= \sum_{\vec{\sigma}} \exp\left(-\frac{e^{f}(\vec{\sigma}) - \vec{\xi}\vec{x}(\vec{\sigma})}{kT} N^{unit}\right) \\ &= \sum_{\vec{\sigma}} \exp\left(-\frac{e^{f}(\vec{\sigma}) - \vec{\mu}^{\mathsf{T}}(\vec{n}(\vec{\sigma}) - \vec{n}^{o})}{kT} N^{unit}\right).\end{aligned}$$

Equations of state

Combining differentials of the fundamental equation and characteristic potential:

$$d\Phi = -SdT - N^{unit} d\vec{\xi}^{\mathsf{T}} \vec{x} + \xi^o dN^{unit}$$

$$S = -\left(\frac{\partial \Phi}{\partial T}\right)_{\vec{\xi}, N^{unit}}$$

$$N^{unit} x_i = -\left(\frac{\partial \Phi}{\partial \xi_i}\right)_{T, \mu_{j \neq i}, N^{unit}}$$

$$\xi^o = \left(\frac{\partial \Phi}{\partial N^{unit}}\right)_{T, \vec{\xi}}$$

Thermodynamic Integration

Free energy: $$\beta\Phi = -\ln Z$$

Partition function: $$Z = \sum_{\vec{\sigma}} \exp\left(-\frac{e^f(\vec{\sigma}) - \vec{\xi}\vec{x}(\vec{\sigma})}{kT} N^{unit}\right)$$

$$\Omega = e^f(\vec{\sigma}) - \vec{\xi}^{\mathsf{T}}\vec{x}(\vec{\sigma})$$

Ω : potential energy per unit cell

$$\langle X\rangle = \frac{1}{Z}\sum_{\vec{\sigma}} X \exp(-\Omega\beta N^{unit})$$

$\langle X\rangle$: ensemble average

Thermodynamic Integration

From free energy:

$$\left(\frac{\partial \beta \Phi}{\partial \beta}\right)_{\xi_i, N^{unit}} = -\frac{1}{Z}\frac{\partial Z}{\partial \beta}$$

From the partition function and ensemble average:

$$\frac{1}{Z}\left(\frac{\partial Z}{\partial \beta}\right)_{\xi_i, N^{unit}} = -N^{unit}\langle \Omega \rangle$$

$$\beta^{end}\Phi(\beta^{end}, \xi_i) = \beta^{begin}\Phi(\beta^{begin}, \xi_i) + N^{unit}\int_{\beta^{begin}}^{\beta^{end}} \langle \Omega \rangle d\beta.$$

Thermodynamic Integration

From free energy:

$$\left(\frac{\partial \beta \Phi}{\partial \xi_i}\right)_{\beta, \xi_{j \neq i}, N^{unit}} = -\frac{1}{Z}\frac{\partial Z}{\partial \xi_i}$$

From the partition function and ensemble average:

$$\frac{1}{Z}\left(\frac{\partial Z}{\partial \xi_i}\right)_{\beta, \xi_{j \neq i}, N^{unit}} = \beta N^{unit} \langle x_i \rangle$$

$$\Phi(\xi_i{}^{end}, \beta, \xi_{j \neq i}) = \Phi(\xi_i{}^{begin}, \beta, \xi_{j \neq i}) - N^{unit} \int_{\xi_i{}^{begin}}^{\xi_i{}^{end}} \langle x_i \rangle d\xi_i$$

Reference states

- Low temperature expansion
  - At low T, only the lowest energy configurations (ground state, and most stable perturbations) will contribute to the partition function. Evaluate partition function explicitly by enumerating 1, 2, ... occupant perturbations. CASM implements the approximation with only 1 occupant perturbations.

$$Z = \sum_{\vec{\sigma}} \exp\left(-\frac{e^f(\vec{\sigma}) - \vec{\xi}\vec{x}(\vec{\sigma})}{kT} N^{unit}\right)$$

Reference states

- Dilute concentration:

$$\Phi = N^{unit} \langle \Omega \rangle - TS$$

$$S = -k \sum_i \left\langle \frac{n_i}{N^{basis}} \right\rangle \ln \left(\left\langle \frac{n_i}{N^{basis}} \right\rangle \right) N^{basis} N^{unit}$$

Importance sampling

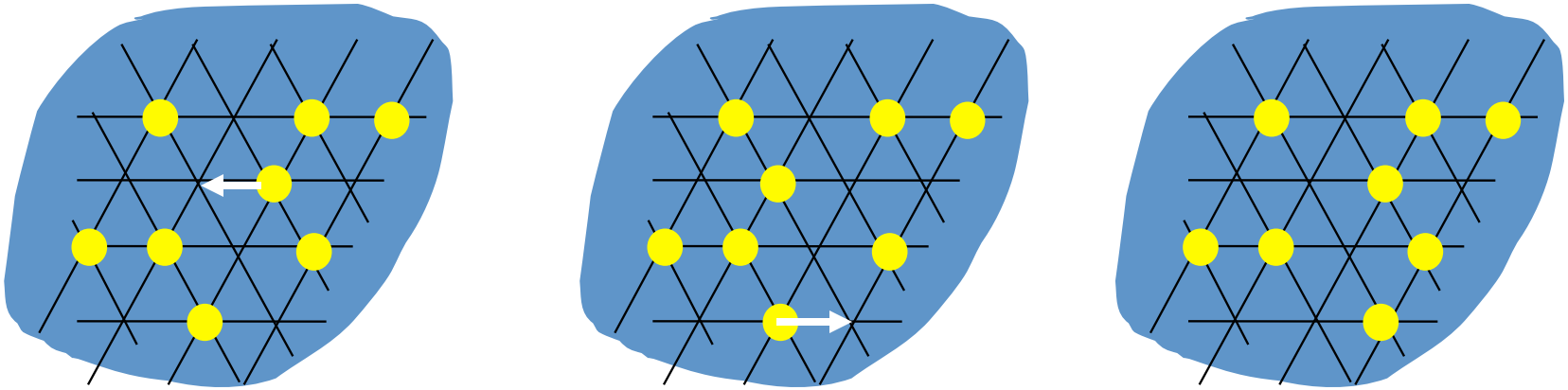
To numerically average thermodynamic quantities,
you don't want to randomly sample $\vec{\sigma}$ from the 2^N possibilities

Instead you want to sample configurations $\vec{\sigma}$
with their probability of occurrence

$$\mathrm{P}(\vec{\sigma}) = \frac{1}{Z}\exp\left(-\frac{E(\vec{\sigma}) - \mu_O N_O}{k_B T}\right)$$

$$Z = \sum_{\vec{\sigma}} \exp\left(-\frac{E(\vec{\sigma}) - \mu_O N_O}{k_B T}\right)$$

Monte Carlo Algorithm



A Monte Carlo Algorithm

1. Start with some configuration
2. Choose perturbation of the system
3. Compute energy for that perturbation

4. If $\Delta E < 0$ -> accept perturbation
   If $\Delta E > 0$ -> accept perturbation if $\exp\left(\frac{-\Delta E}{kT}\right) > R$ a random number
5. Choose next perturbation

Property will be average over these states

Monte Carlo output

(Semi-grand Canonical)

Thermodynamic variables controlled in Monte Carlo (i.e. input)

$$\beta, \vec{\xi}$$

Output from Monte Carlo

$$\langle \Omega \rangle (\beta, \vec{\xi}) \qquad \langle \vec{x} \rangle (\beta, \vec{\xi})$$

Monte Carlo output

(Semi-grand Canonical)

$$\texttt{heat\_capacity} = C_{\vec{\mu}_x}/N^{unit} = \frac{\langle \Omega^2 \rangle - \langle \Omega \rangle^2}{kT^2} N^{unit}$$

$$\texttt{susc\_x(i,j)} = \chi^{\vec{x}}_{ij}/N^{unit} = \frac{\langle x_i x_j \rangle - \langle x_i \rangle \langle x_j \rangle}{kT} N^{unit}$$

$$\texttt{susc\_x(S,i)} = \chi^{\vec{x}}_{S,i}/N^{unit} = \frac{\langle \Omega x_i \rangle - \langle \Omega \rangle \langle x_i \rangle}{kT} N^{unit}.$$

$$\texttt{susc\_n(i,j)} = \chi^{\vec{n}}_{ij}/N^{unit} = \frac{\langle n_i n_i \rangle - \langle n_i \rangle \langle n_i \rangle}{kT} N^{unit}$$

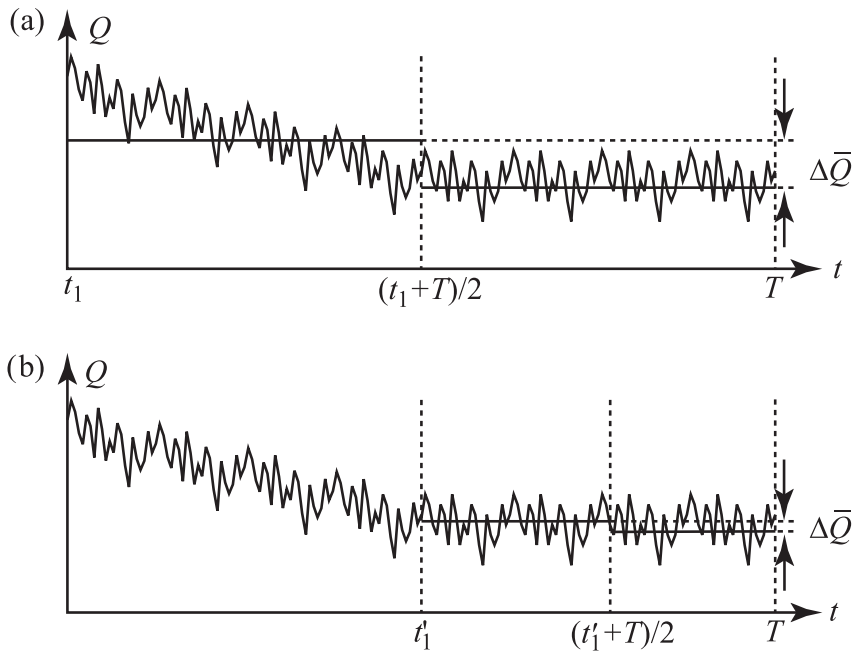
$$\texttt{susc\_n(S,i)} = \chi^{\vec{n}}_{S,i}/N^{unit} = \frac{\langle \Omega n_i \rangle - \langle \Omega \rangle \langle n_i \rangle}{kT} N^{unit}.$$

Monte Carlo

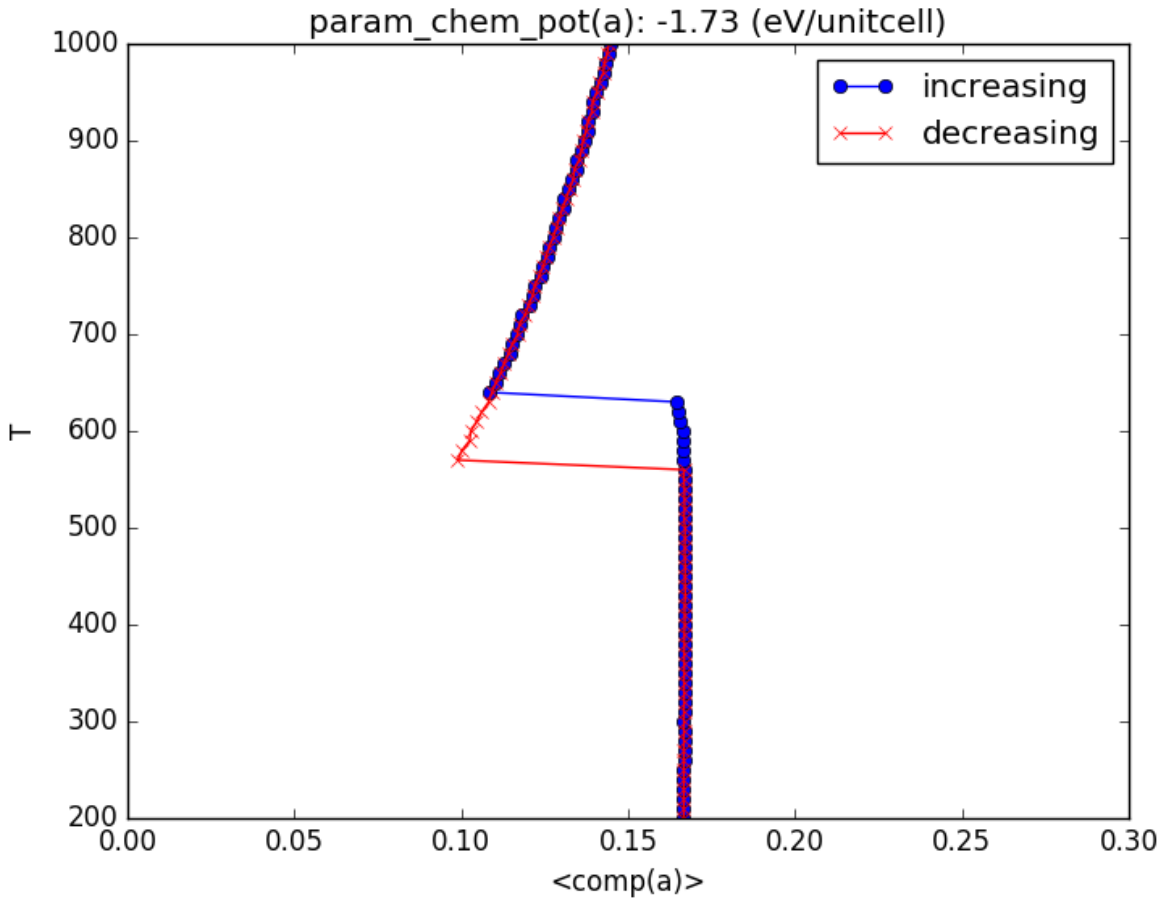
- See 'casm monte -h' and 'casm format --monte' for help setting up an input file.
- The input file lets you specify:
  - The cluster expansion
  - Supercell and initial configuration
  - Pathway in $T, \vec{\xi}$ space, with dependent or independent runs
  - Monte Carlo ensemble:
    - "grand_canonical" (only option currently) for semi-grand canonical ensemble
  - Monte Carlo method:
    - 'metropolis' or 'lte1' (low temperature expansion)
  - Quantities to sample & convergence criteria
  - Enable saving configurations encountered during Monte Carlo into your configuration list

Monte Carlo Convergence

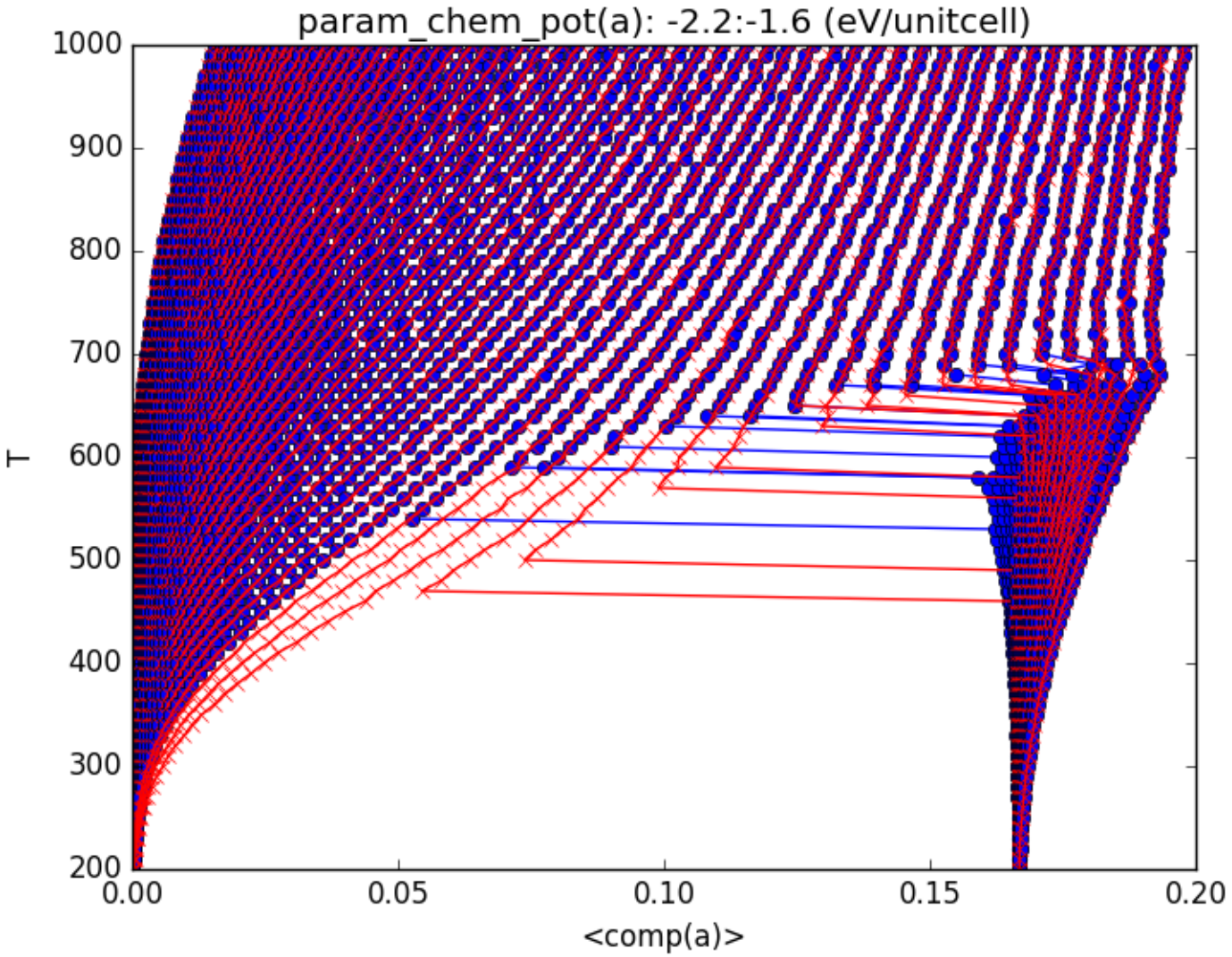
- All quantities that are given a requested precision must first equilibrate, and then the estimated precision on the average must converge to the requested precision
  - A. van de Walle, M. Asta, *Modell. Simul. Mater. Sci. Eng.* **10**, 521 (2002)



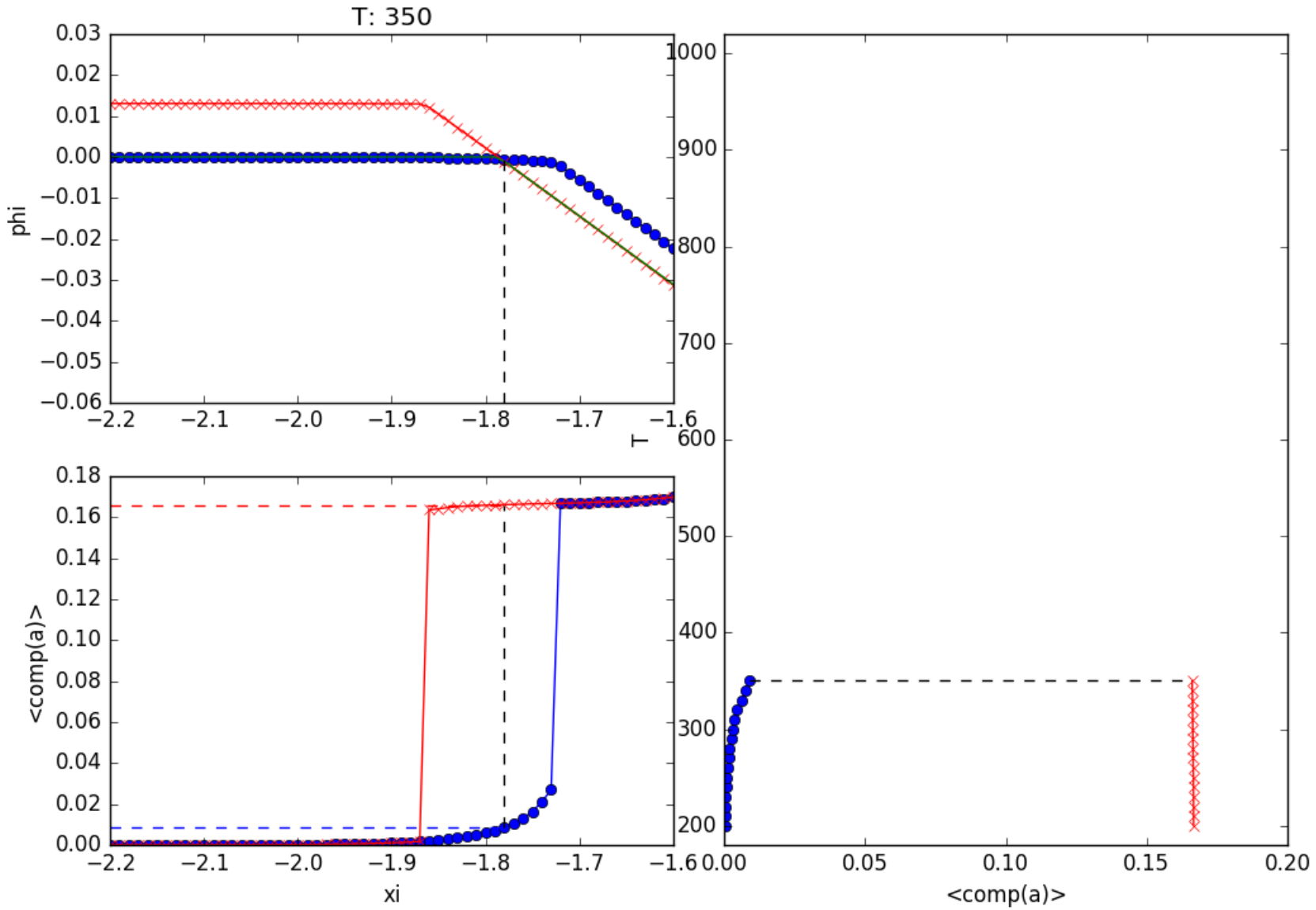
Phase diagram construction



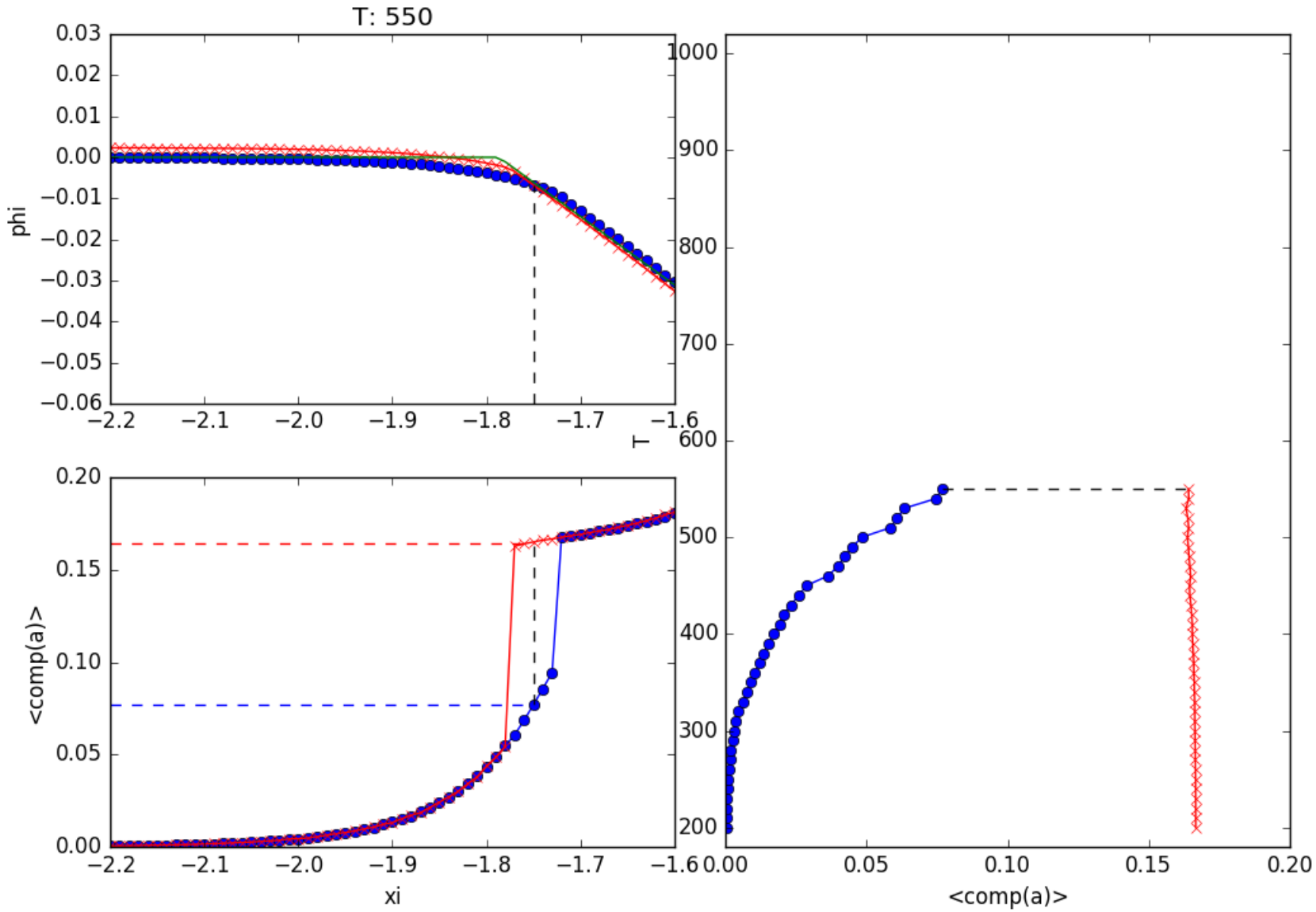
Phase diagram construction



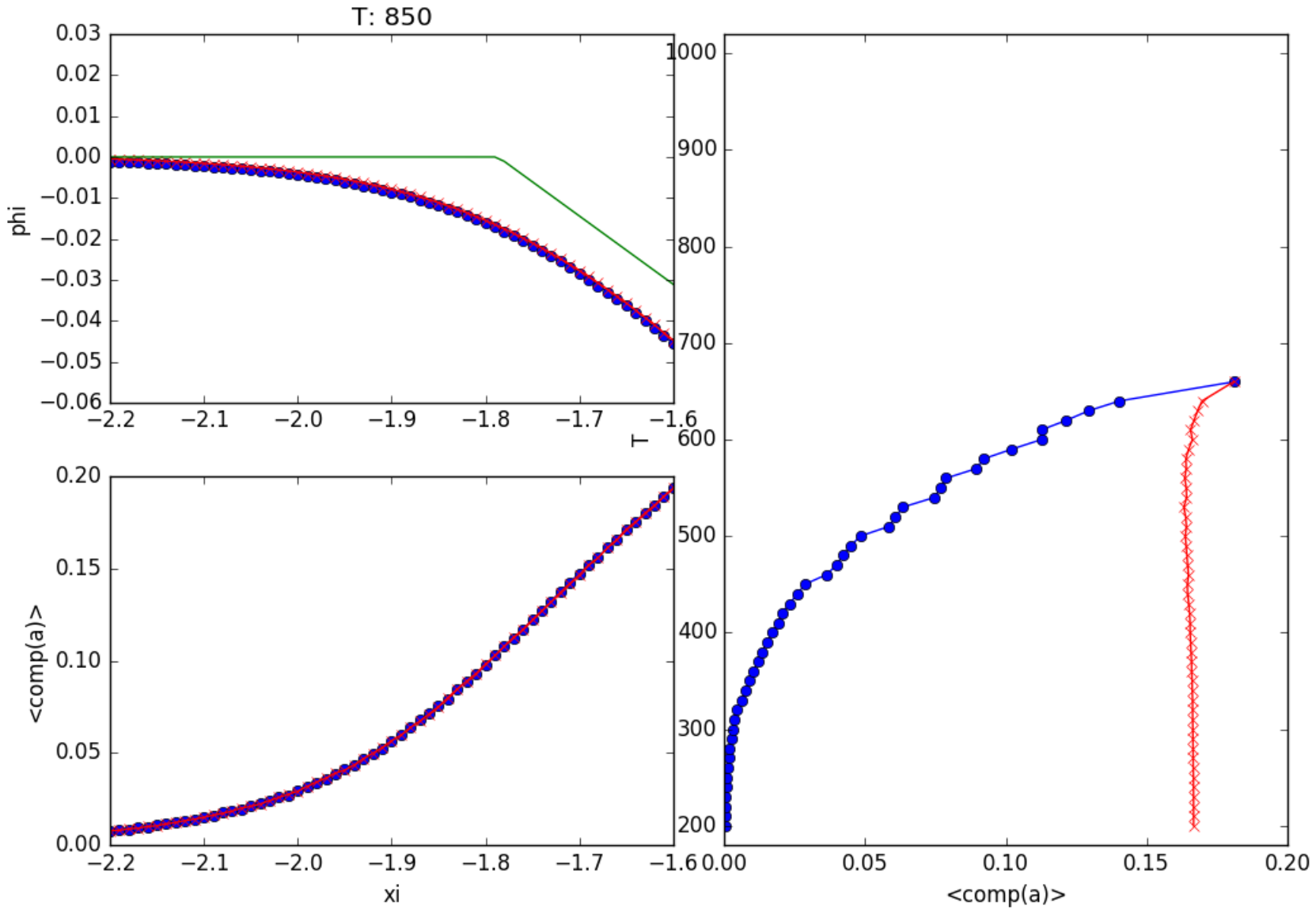
Phase diagram construction



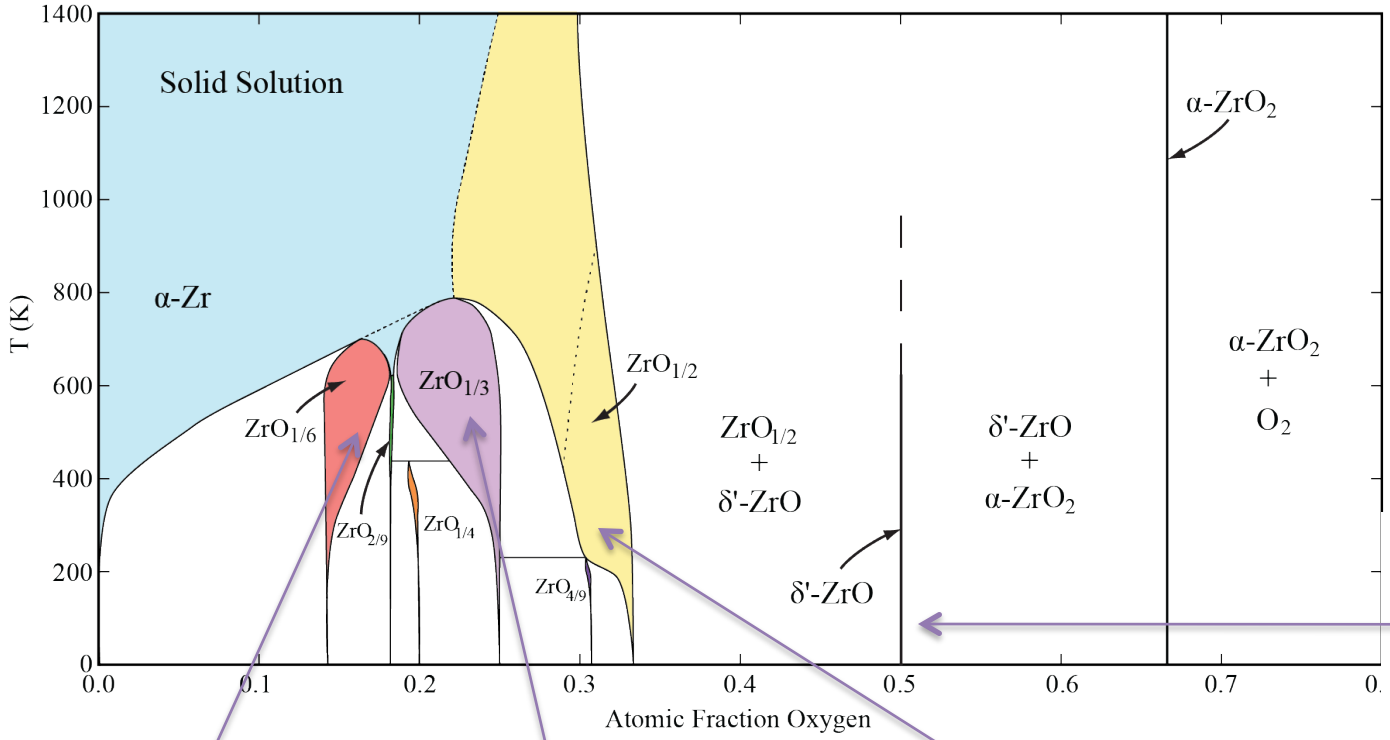
Phase diagram construction



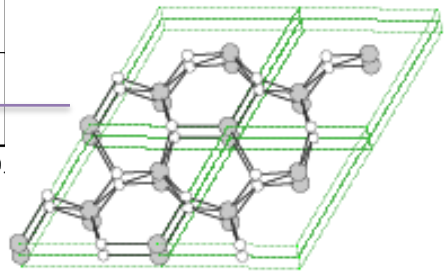
Phase diagram construction

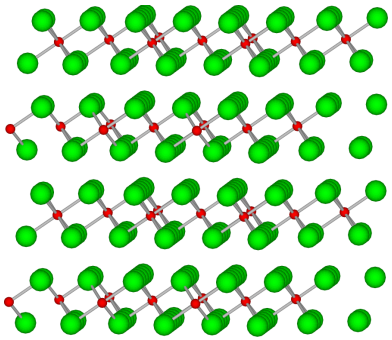


Calculated phase diagram

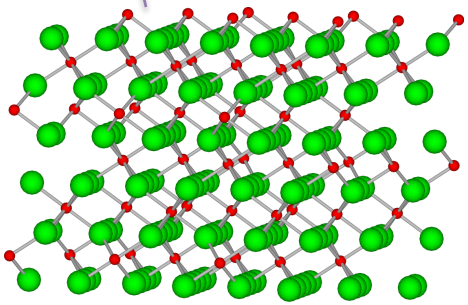


ZrO

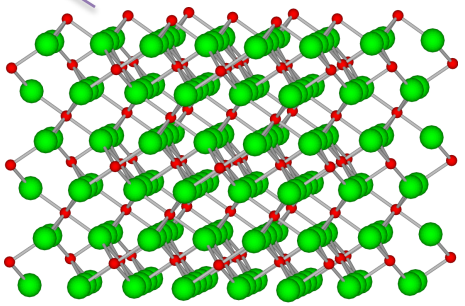




$ZrO_{1/6}$



$ZrO_{1/3}$



$ZrO_{1/2}$

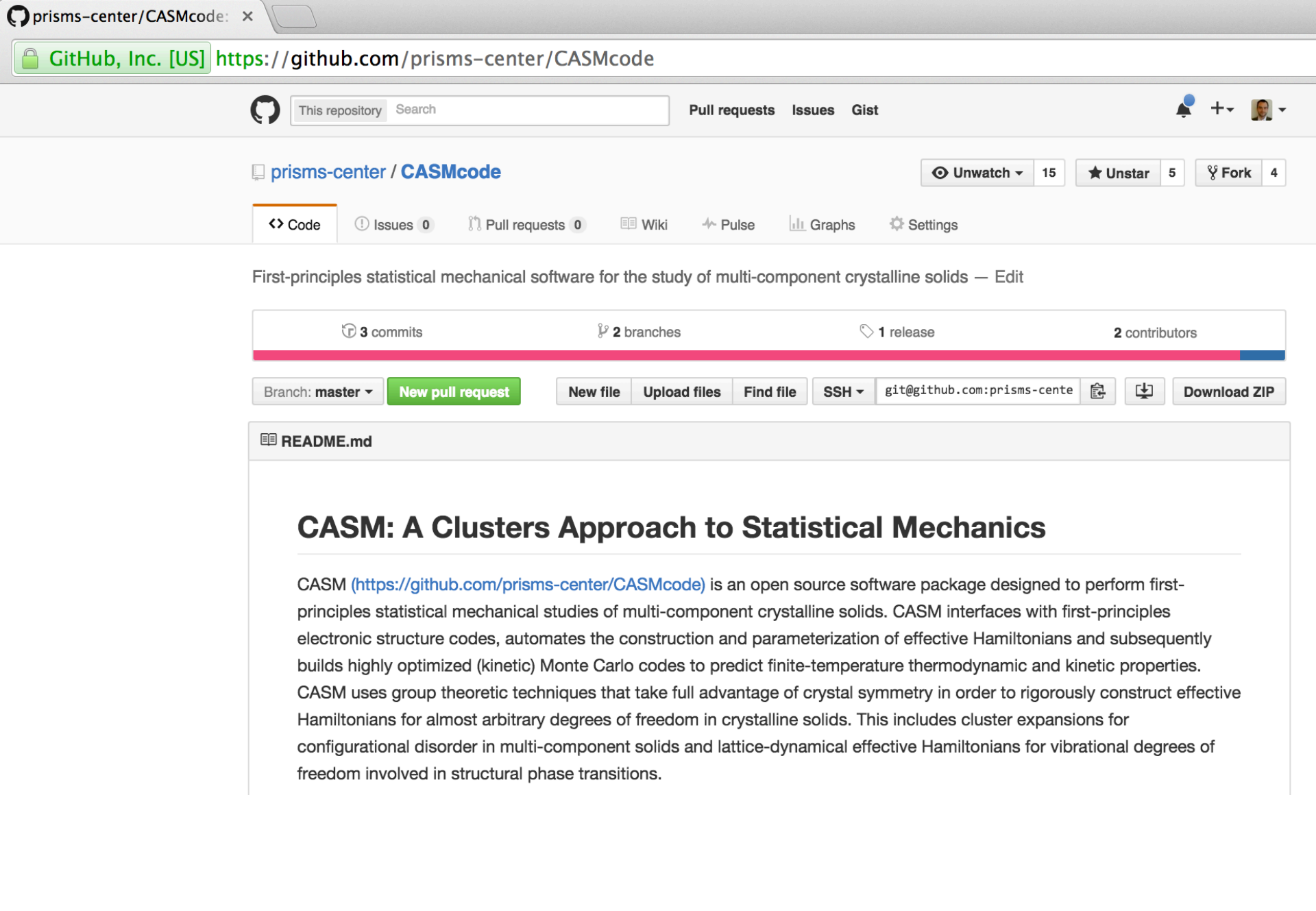
CASM Tutorial Outline

1:30 – 3:00:

- Crystal structure identification & specification
- Symmetry analysis
- Using CASM to manage data
  - Configuration selections and query syntax
  - Maintaining sanity with cluster expansion profiles
- Cluster expansion basics
  - Configuration enumeration
  - Cluster expansion basis sets & correlation evaluation

3:30 – 5:00:

- Parameterizing cluster expansions
- Predicting thermodynamic properties with grand canonical Monte Carlo
- Free energy integration and phase diagram construction


prisms-center/CASMcode
GitHub, Inc. [US] https://github.com/prisms-center/CASMcode
This repository Search
Pull requests Issues Gist
prisms-center / CASMcode
Unwatch 15 Unstar 5 Fork 4
Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings
First-principles statistical mechanical software for the study of multi-component crystalline solids — Edit
3 commits 2 branches 1 release 2 contributors
Branch: master New pull request
New file Upload files Find file SSH git@github.com:prisms-cente Download ZIP
README.md
CASM: A Clusters Approach to Statistical Mechanics
CASM (https://github.com/prisms-center/CASMcode) is an open source software package designed to perform first-principles statistical mechanical studies of multi-component crystalline solids. CASM interfaces with first-principles electronic structure codes, automates the construction and parameterization of effective Hamiltonians and subsequently builds highly optimized (kinetic) Monte Carlo codes to predict finite-temperature thermodynamic and kinetic properties. CASM uses group theoretic techniques that take full advantage of crystal symmetry in order to rigorously construct effective Hamiltonians for almost arbitrary degrees of freedom in crystalline solids. This includes cluster expansions for configurational disorder in multi-component solids and lattice-dynamical effective Hamiltonians for vibrational degrees of freedom involved in structural phase transitions.