2018-10-03 @ 2nd ICCMS

Algorithms and Libraries in Computational Condensed Matter Physics

計算物性物理におけるアルゴリズムとライブラリ

Synge Todo (藤堂眞治)

Department of Physics, University of Tokyo

Self introduction - Synge Todo 藤堂眞治

- 1968 Born in Matsuyama in Shikoku Island
- 1996 Ph.D @ Department of Physics, University of Tokyo (Supervisor: Masuo Suzuki)
- 1996 Institute for Solid State Physics, University of Tokyo (Roppongi & Kashiwa)
- 2000 PostDoc in ETH Zürich, Switzerland
- 2002 Department of Applied Physics, University of Tokyo
- 2011 Institute for Solid State Physics, University of Tokyo (Kobe)

理化学研究所 計算科学研究機構 (AICS → R-CCS)

理化学研究所 計算科学研究機構 (AICS → R-CCS)

Self introduction - Synge Todo 藤堂眞治

- 1968 Born in Matsuyama in Shikoku Island
- 1996 Ph.D @ Department of Physics, University of Tokyo (Supervisor: Masuo Suzuki)
- 1996 Institute for Solid State Physics, University of Tokyo (Roppongi & Kashiwa)
- 2000 PostDoc in ETH Zürich, Switzerland
- 2002 Department of Applied Physics, University of Tokyo
- 2011 Institute for Solid State Physics, University of Tokyo (Kobe)
- 2014 Department of Physics, University of Tokyo

- Interests: computational physics, statistical physics, condensed-matter physics, data science, applied mathematics, development of open-source software, etc
- Hobby: debug of programs written by students, installation of open-source software in various computers

Libraries used in computational materials science

- Libraries for parallelization, I/O
  - MPI, HDF5
- "Numerical" Libraries
  - matrix operations, eigenvalue problems, linear equations (BLAS, LAPACK, ...)
  - fast Fourier transform, random number generator, optimization, etc
- Domain specific libraries
  - ALPS libraries for quantum lattice models
- Combination of simulation engines, algorithms, tools, etc
  - Data assimilation approach for crystal structure prediction
  - ALPS framework for large-scale parallel simulations
  - Conversion of application program into library

The ALPS project

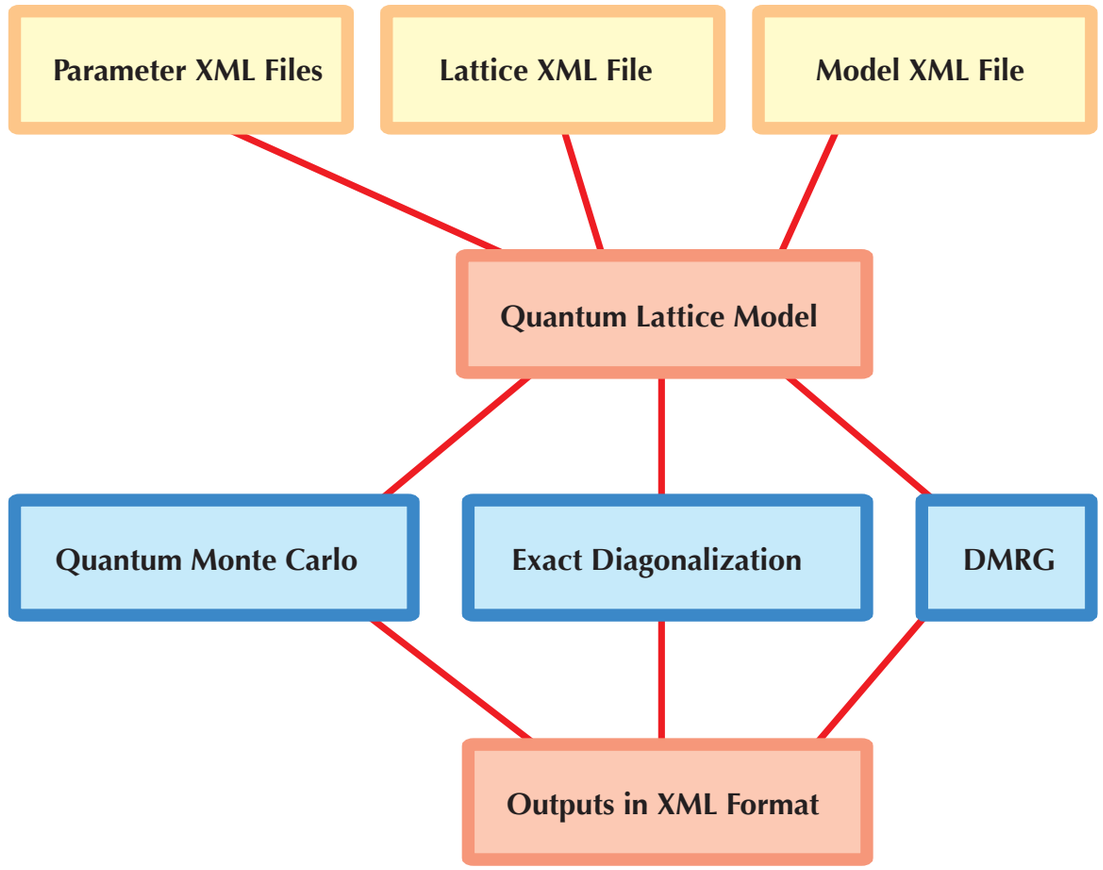
ALPS = Algorithms and Libraries for Physics Simulations

- International collaboration for developing open-source software for simulation of quantum lattice models, such as quantum spin systems, electron systems, etc

- ALPS Libraries = collection of generic C++ libraries

- ALPS Applications = collection of application packages using modern algorithms such as QMC, DMRG, ED, etc

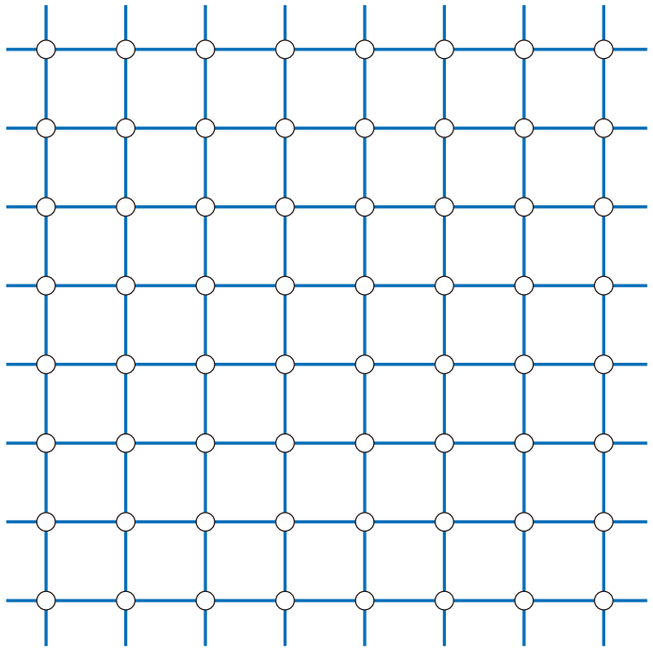
- ALPS Framework = environment for executing large-scale parallel simulations including XML, tools, scheduler, etc



Quantum lattice models

- Quantum spin model (XXZ model)

$$\mathcal{H} = \frac{J^{xy}}{2} \sum_{\langle i,j \rangle} (S_i^+ S_j^- + S_i^- S_j^+) + J^z \sum_{\langle i,j \rangle} S_i^z S_j^z$$



- Hubbard model (fermionic / bosonic)

$$\mathcal{H} = -t \sum_{\langle i,j \rangle \sigma} (c_{i\sigma}^\dagger c_{j\sigma} + \text{h.c.}) + U \sum_i n_{i\uparrow} n_{i\downarrow}$$

- t-J model

$$\mathcal{H} = -t \sum_{\langle i,j \rangle \sigma} (c_{i\sigma}^\dagger c_{j\sigma} + \text{h.c.}) + J \sum_{i,j} (\mathbf{S}_i \cdot \mathbf{S}_j - n_i n_j / 4)$$

- bose Hubbard model, spinless fermions, Kondo lattice model, etc

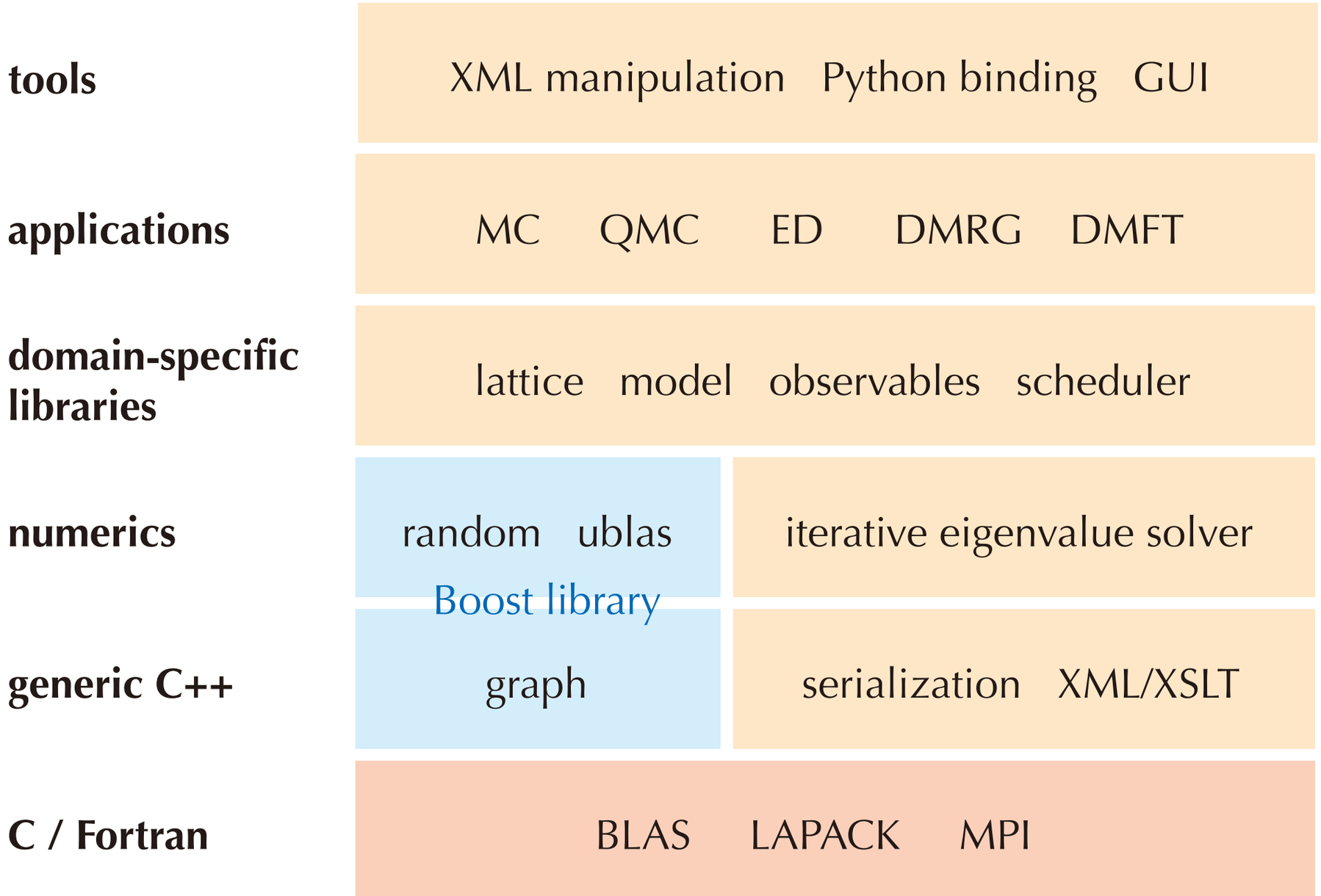
Simulation algorithms for quantum lattice model

- Exact diagonalization
  - full-diagonalization (Householder method), Lanczos method
- Classical Monte Carlo
  - Metropolis method, cluster algorithm, etc
- Quantum Monte Carlo
  - loop algorithm, directed loop algorithm, worm algorithm, etc
- Matrix-product and tensor-network methods
  - DMRG (density matrix renormalization group), (i)TEBD, (i)PEPS, MERA, etc
- Dynamical mean-field theory
  - QMC solver, diagonalization solver, etc

Distinctive features of ALPS

- Arbitrary lattice structures
  - defining lattice structure in XML format
  - construction of lattice by repeating unit cell structures
  - arbitrary finite graph (= set of vertices and edges) can also be defined
- Arbitrary Hamiltonian (model)
  - defining quantum numbers and operators in XML format
  - defining Hamiltonian by symbolic expression
- State-of-the-art algorithms (applications): ED, CMC, QMC, tensor network, DMFT
- Common input format for all the ALPS applications
- Generic output format, analysis and plot tools in Python

ALPS Libraries and Applications



ALPS/lattice library

- Lattices are represented as mathematical structures called “graphs”

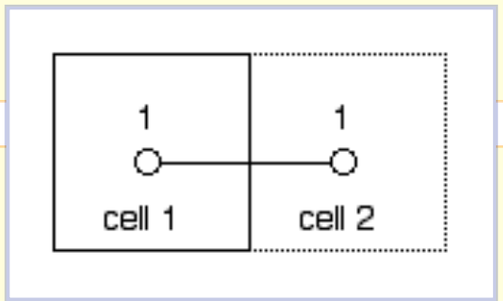
lattice	graph
site	vertex
bond	edge

- ALPS:Lattice library provides a useful “wrapper” on generic graph libraries (e.g. BGL)
  - Input/Output in XML format
  - constructing a lattice by repeating unit cell
  - coordinates, parity, reciprocal vectors, etc
- Predefined lattices:
  - “chain lattice”, “square lattice”, “triangular lattice”, “honeycomb lattice”, “simple cubic lattice”, etc

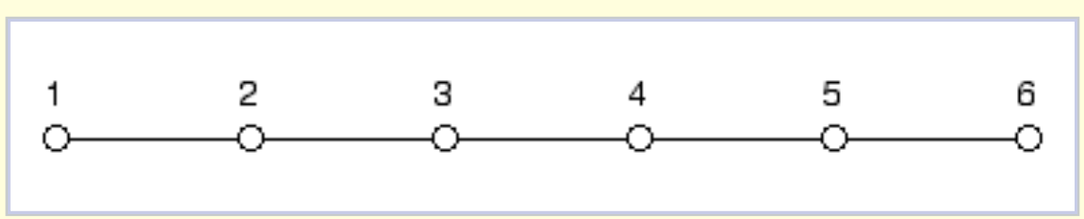
Example: finite lattice + embedded unit cell

```
<LATTICE name="chain lattice" dimension="1">
  <BASIS><VECTOR> 1 </VECTOR></BASIS>
</LATTICE>
```

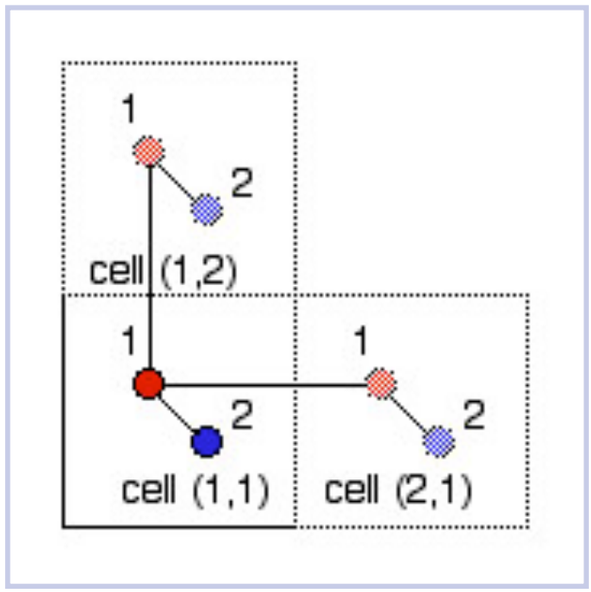
```
<UNITCELL name="simple1d" dimension="1" vertices=”1”>
  <EDGE>
    <SOURCE vertex="1" offset="0"/><TARGET vertex="1" offset="1"/>
  </EDGE>
</UNITCELL>
```




```
<LATTICEGRAPH name = "chain lattice">
  <FINITELATTICE>
    <LATTICE ref="chain lattice"/>
    <PARAMETER name=”L”/>
    <EXTENT size ="L"/>
    <BOUNDARY type="periodic"/>
  </FINITELATTICE>
  <UNITCELL ref="simple1d"/>
</LATTICEGRAPH>
```



A more complex example: 2D Kondo lattice




```
<UNITCELL name="kondo2d" dimension="2">
  <VERTEX type=”0”><COORDINATE> 0.3 0.7 </COORDINATE></VERTEX>
  <VERTEX type=”1”><COORDINATE> 0.6 0.3 </COORDINATE></VERTEX>
  <EDGE><SOURCE vertex="1"/><TARGET vertex="1" offset="1 0"/></EDGE>
  <EDGE><SOURCE vertex="1"/><TARGET vertex="1" offset="0 1"/></EDGE>
  <EDGE type=”1”><SOURCE vertex="1"/><TARGET vertex="2"/></EDGE>
</UNITCELL>
```

ALPS/model library

- Enables us to define a Hamiltonian in XML format
  - quantum numbers, operators
  - accepts symbolic expressions for site-/bond-Hamiltonians

```
Jz*Sz(i)*Sz(j)+Jxy/2*(Splus(i)*Sminus(j)+Sminus(i)*Splus(j))
```

    - generates local Hamiltonians in a matrix form
- Predefined models: “spin”, “boson Hubbard”, “hardcore boson”, “fermion Hubbard”, “alternative fermion Hubbard”, “spinless fermions”, “Kondo lattice”, “t-J”

Model XML for describing Hamiltonian

```
<HAMILTONIAN name="spin">
  <PARAMETER name="Jz" default="J"/>
  <PARAMETER name="Jxy" default="J"/>
  <PARAMETER name="J" default="1"/>
  <PARAMETER name="Jz'" default="J'"/>
  <PARAMETER name="Jxy'" default="J'"/>
  <PARAMETER name="J'" default="0"/>
  <PARAMETER name="h" default="0"/>
  <BASIS ref="spin"/>
  <SITETERM site="i"> -h * Sz(i) </SITETERM>
  <BONDTERM type="0" source="i" target="j">
    Jz * Sz(i) * Sz(j) + Jxy * (Splus(x)*Sminus(y)+Sminus(x)*Splus(y)) / 2
  </BONDTERM>
  <BONDTERM type="1" source="i" target="j">
    Jz' * Sz(i) * Sz(j) + Jxy' * (Splus(x)*Sminus(y)+Sminus(x)*Splus(y)) / 2
  </BONDTERM>
</HAMILTONIAN>
```

$$\mathcal{H} = \sum_{\langle i,j \rangle} [J_z S_i^z S_j^z + \frac{J_{xy}}{2}(S_i^+ S_j^- + S_i^- S_j^+)] - \sum_i h S_i^z$$

ALPS/alea library for physical observables

- Template library for accumulating and analyzing correlated time-series observagles from Markov process

```
alps::RealObservable mag2("Magnetization^2");
...
mag2 << m * m; // at each MC step
```

```
std::cout << mag2 << std::endl;
```

- binning analysis for mean, error, and autocorrelation time

```
Magnetization^2: 3.142 +/- 0.001; tau = 10.3
```

- jackknife error analysis for correlated quantities

```
alps::RealObsEvaluator binder = mag2 * mag2 / mag4;
std::cout << binder.mean() << ' ' << binder.error() << '\n';
```

Writing Monte Carlo code for classical Ising model

- Initialization of lattice and alea
- main loop is just a few lines

```
for (int s = 0; s < num_sites(); ++s) {
  double diff = 2 * field * spins[s];
  neighbor_bond_iterator itr, itr_end;
  for (boost::tie(itr, itr_end) = neighbor_bonds(s); itr != itr_end; ++itr) {
    int t = s ^ source(*itr) ^ target(*itr); // calculate index of neighbor spin
    diff += 2 * coupling[bond_type(*itr)] * spins[s] * spins[t];
  }
  if (uniform_01() < 0.5 * (1 + std::tanh(-0.5 * beta * diff))) spins[s] = -spins[s];
}
```

- + measurement of physical quantities (energy, magnetization, etc)
- works for any lattice structures

Libraries used in computational materials science

- Libraries for parallelization, I/O
  - MPI, HDF5
- "Numerical" Libraries
  - matrix operations, eigenvalue problems, linear equations (BLAS, LAPACK, ...)
  - fast Fourier transform, random number generator, optimization, etc
- Domain specific libraries
  - ALPS libraries for quantum lattice models
- Combination of simulation engines, algorithms, tools, etc
  - Data assimilation approach for crystal structure prediction
  - ALPS scheduler for large-scale parallel simulations
  - Conversion of application program into library

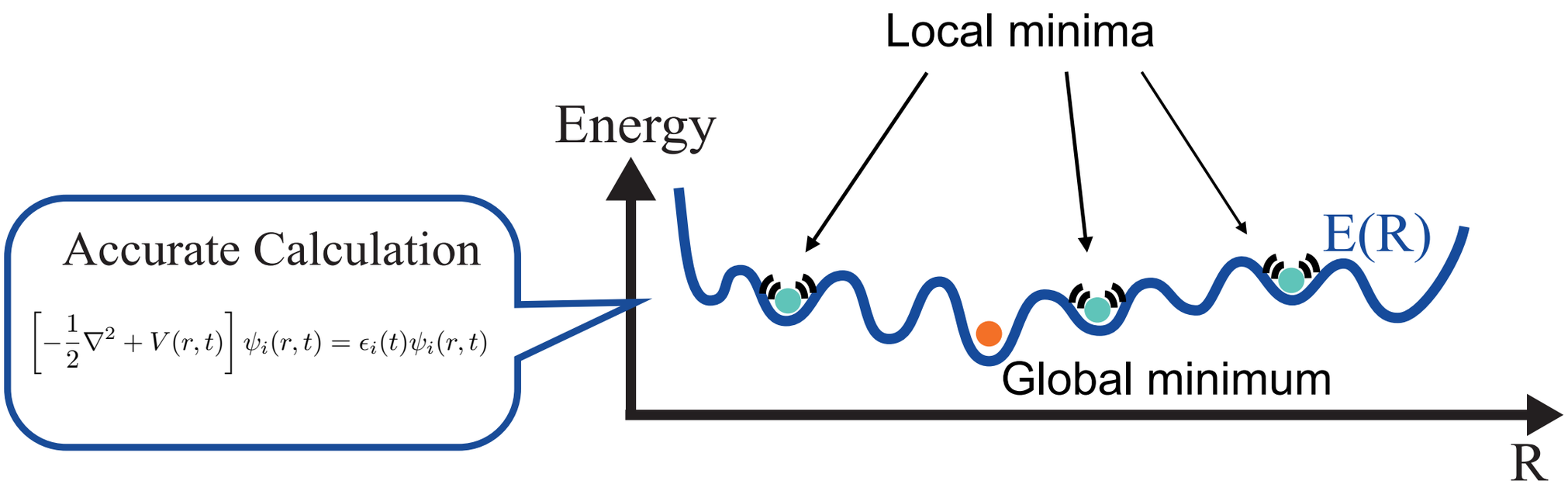
ALPS ライブラリ・フレームワークを使う利点

- エラーバーと自己相関時間の自動計算 (ALPS/alea)
- 自動並列化 & ロードバランシング (ALPS/scheduler)
- チェックポイント作成とリスタート機能 (ALPS/scheduler)
- XML・HDF5形式による計算結果の自動出力とこれらを扱うツール
- 入力パラメータの柔軟な扱い (ALPS/parameter)
- 擬似乱数生成器 (ALPS/random)
- 格子の容易な取扱い (ALPS/lattice)
- 量子ハミルトニアンの容易な取扱い (ALPS/model)

- アプリケーションの「ALPS化」のチュートリアルを用意

Structure prediction from first-principles calculation

- Local structure can be predicted very accurately
  - by using density functional theory, or highly-accurate classical force field
- Energy landscape again has many local minima



- Naive gradient decent method does not work
- Several optimization techniques have been developed: simulated annealing, Wang-Landau sampling, particle-swarm optimization, evolutionary algorithm, Bayesian optimization, etc

Calculation of powder X-ray diffraction pattern

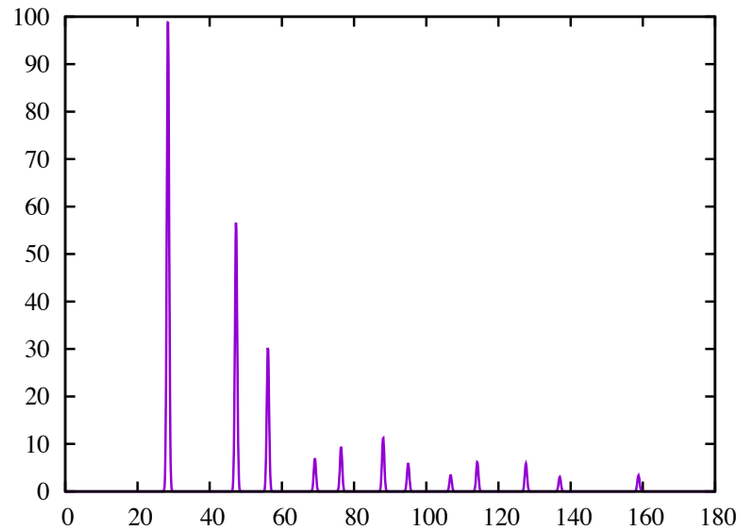
- Powder X-ray diffraction peaks

$$I^{\mathrm{peak}}(\theta) = \sum_k \delta(\theta - \theta(k))|F(k)|^2 Lp(\theta(k)),$$

- Structure factor

$$F(k) = \int \rho(r) \exp(2\pi i r \cdot k)\, dr$$

$$\simeq \sum_{\mathrm{unit\ \ cell}} f_{\mathrm{atom}}(k) \exp(2\pi i r_{\mathrm{atom}} \cdot k)$$



- Lorentz polarization factor

$$Lp(\theta) = \frac{1 + \cos^2(2\theta)}{\sin(2\theta)\sin(\theta)}$$

- Information of phase and lattice direction is lost in diffraction pattern

Direct space method

- Minimize R-factor by changing atomic position (reverse MC, Rietveld analysis)

$$R = \frac{\int |I_{\mathrm{calc}}(\theta) - I_{\mathrm{exp}}(\theta)| d\theta}{\int |I_{\mathrm{exp}}(\theta)| d\theta}$$

- Problem
  - Search space is high dimensional: 3(N-1) or 3(N-1)+6
  - “Energy” surface has many local minima (double occupied configuration, etc)
  - Very sensitive to noise in experimental data
- Need of some “regularization” term

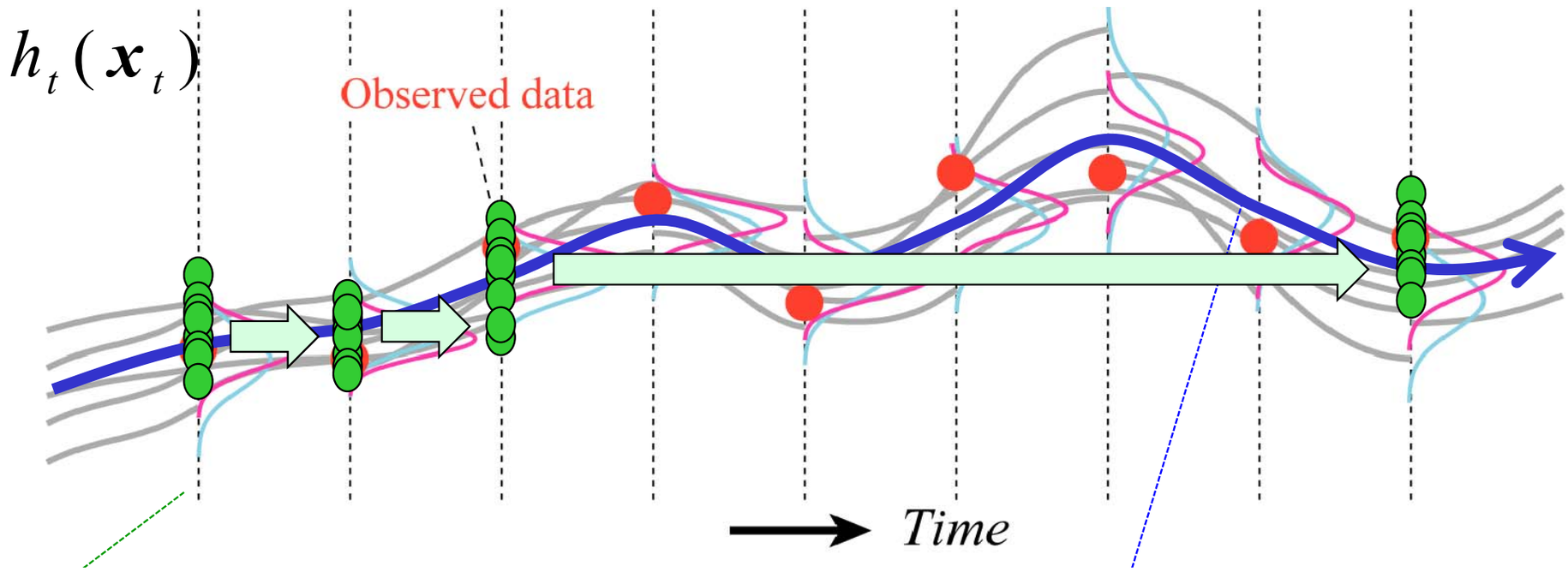
$$R(r) + \lambda P(r)$$

  - e.g. repulsive force between atoms: H. Putz, et al., J. Appl. Cryst. 32, 864 (1999), O. J. Lanning, et al., Chem. Phys. Lett. 317, 296 (2000).

Our strategy - data assimilation in materials science

- To propose “good” cost function
  - hybrid approach: combination of two cost functions
  - can take short-range atomic force and long-range periodic structure into account simultaneously
  - with less local minima
  - with less control parameters
  - robust against noise in experimental data
- To develop “good” optimization method for new class of optimization problem
  - new simultaneous optimization method for two (or more) cost functions that share the same global minimum but have different local-minima distributions

Data assimilation in weather forecast



- gray lines and green dots: simulation results based on some model
  - can predict physical quantities (temperature, wind, etc) on every grid point
  - uncertainty in initial conditions
- red dots: observed data
  - accurate
  - number of data points are limited
- blue line: result of data assimilation

http://www.data-assimilation.riken.jp/jp/events/ithes_da_2016fall/slides/160912_iTHES_Higuchi.pdf

Data assimilation and Bayes' theorem

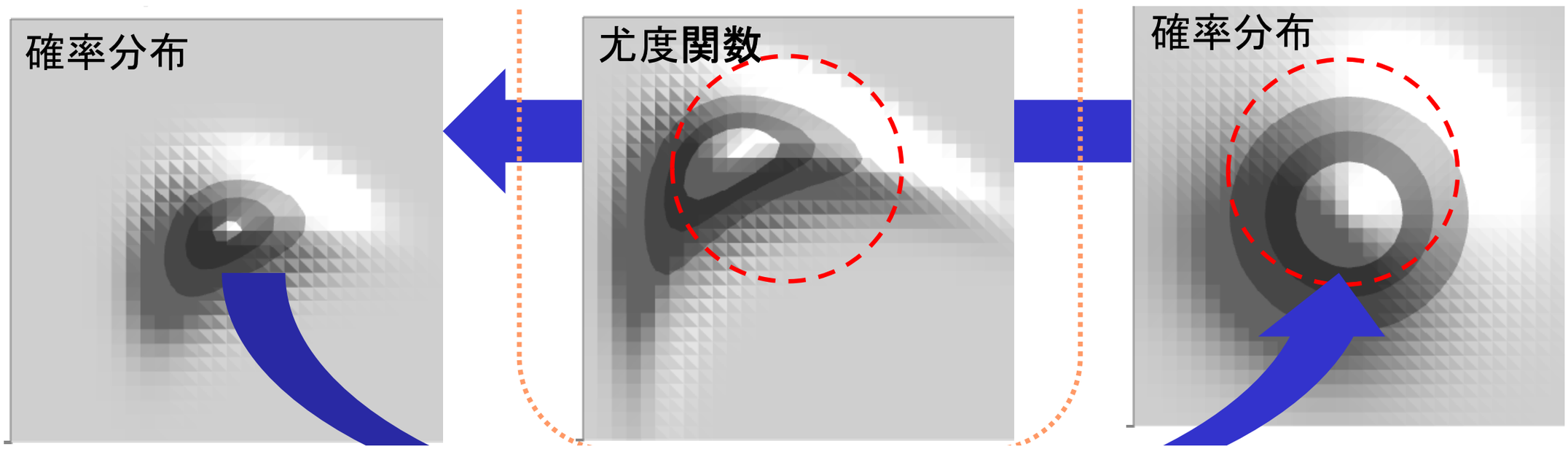
- Bayes' theorem

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \sim p(y|x)p(x)$$

posterior:
improved knowledge about x

likelihood:
feasibility of y for given x

prior:
belief about x



http://www.data-assimilation.riken.jp/jp/events/ithes_da_2016fall/slides/160912_iTHES_Higuchi.pdf

Data assimilation in crystal structure prediction

- Bayes' theorem

$$p(R|\theta) \sim p(\theta|R)p(R)$$

- R: atomic structure

$$p(R) \sim \exp\{-\beta E(R)\}$$

- θ: X-ray diffraction pattern

$$p(\theta|R) \sim \exp\{-\beta D(\theta, \theta_{\mathrm{calc}}(R))\}$$

- posterior for atomic structure

$$p(R|\theta) \sim \exp\{-\beta[E(R) + D(\theta, \theta_{\mathrm{calc}}(R))]\}$$

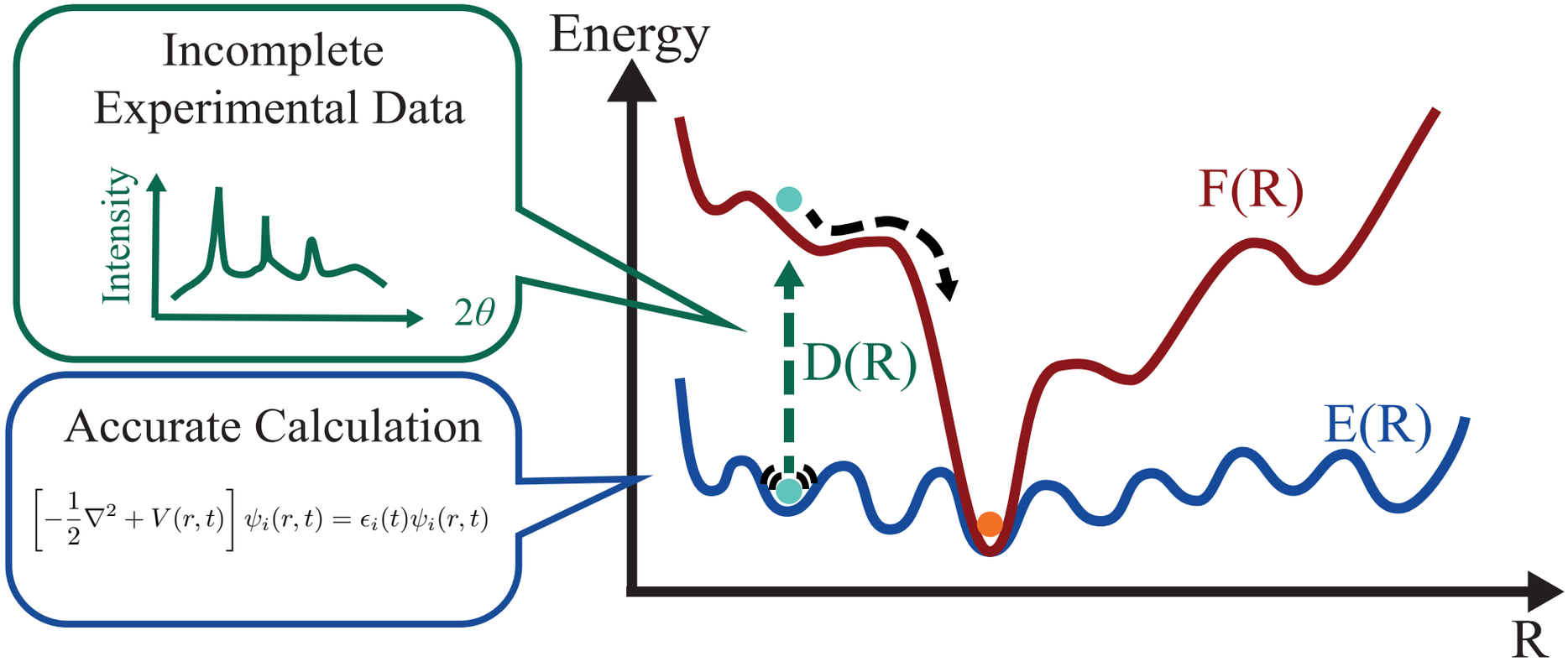
- MAP (maximum a posteriori)
  - → minimize $F(R) = E(R) + D(\theta, \theta_{\mathrm{calc}}(R))$

Hybrid cost function

- Optimizing E(R) and D(R) simultaneously

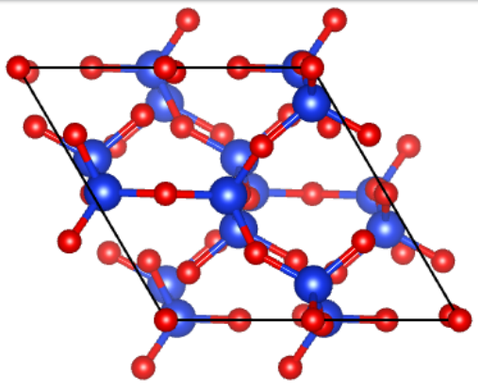
$$F(\mathbf{R}) = E(\mathbf{R}) + \alpha N D[g(\mathbf{R}), g_{\mathrm{obs}}]$$

- R: position of atoms (3N dimensions)
- E(R): accurate atomic energy calculation (by DFT or classical force field)
- D(R): penalty function calculated from (insufficient) diffraction data



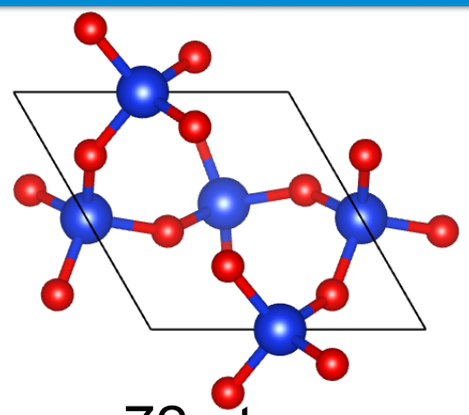
Structure prediction for polymorphs of SiO_2

coesite



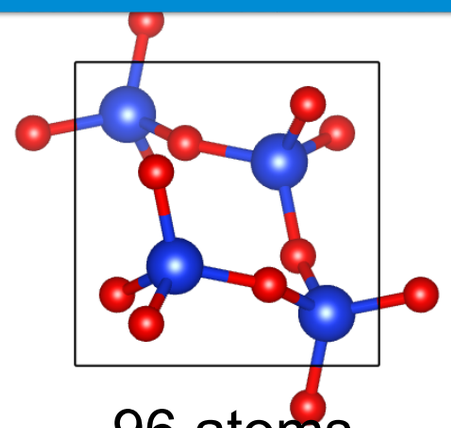
96 atoms

low-quartz

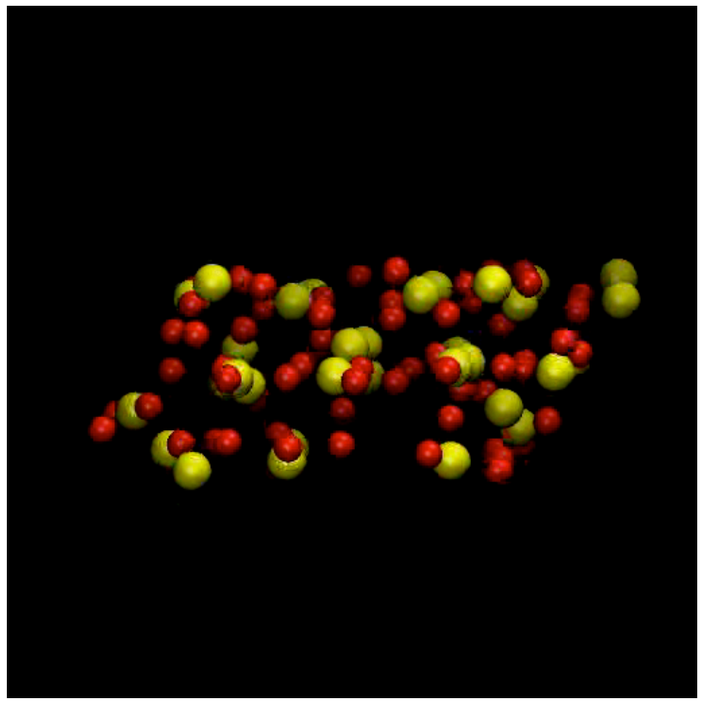


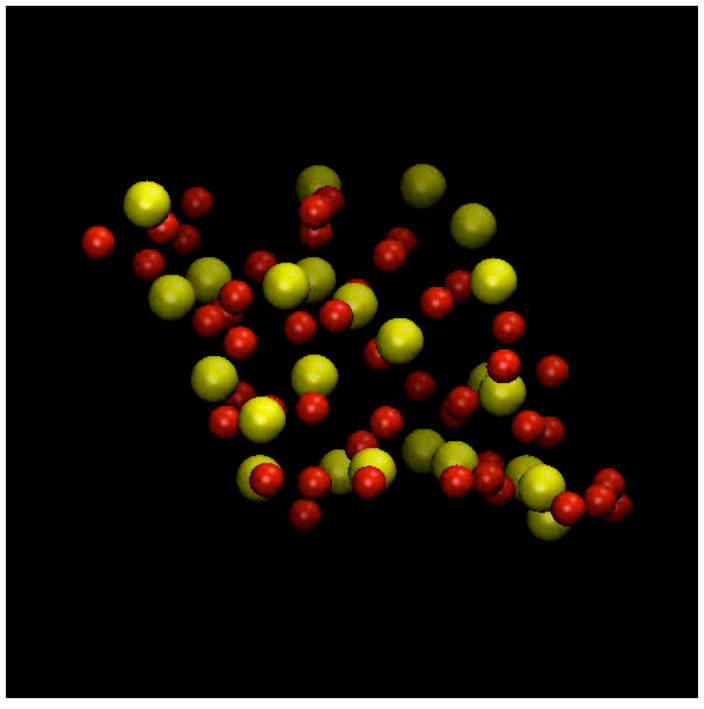
72 atoms

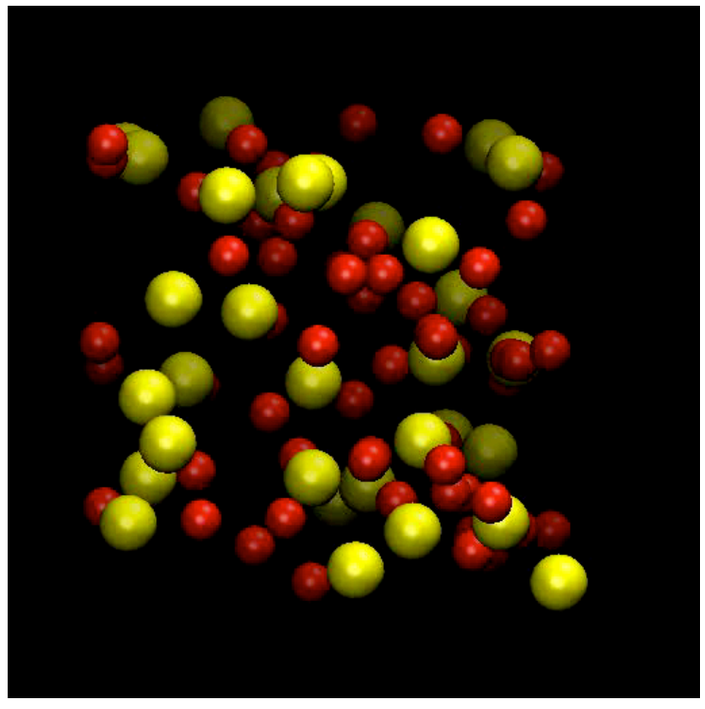
low-cristobalite



96 atoms

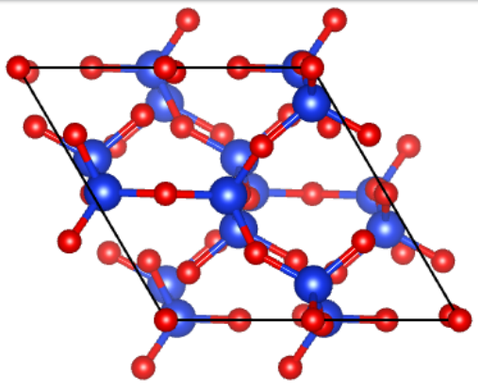






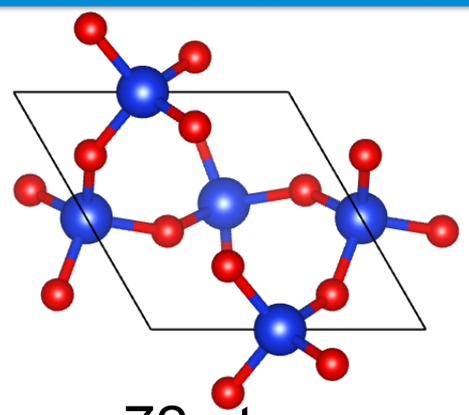
Structure prediction for polymorphs of SiO_2

coesite



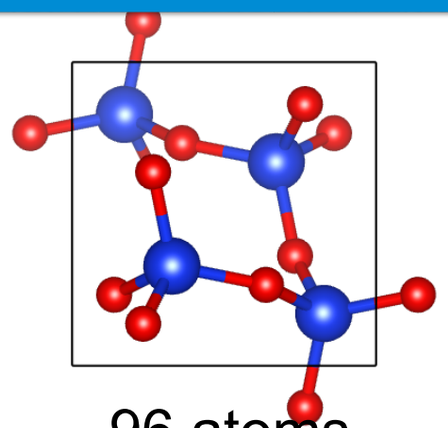
96 atoms

low-quartz

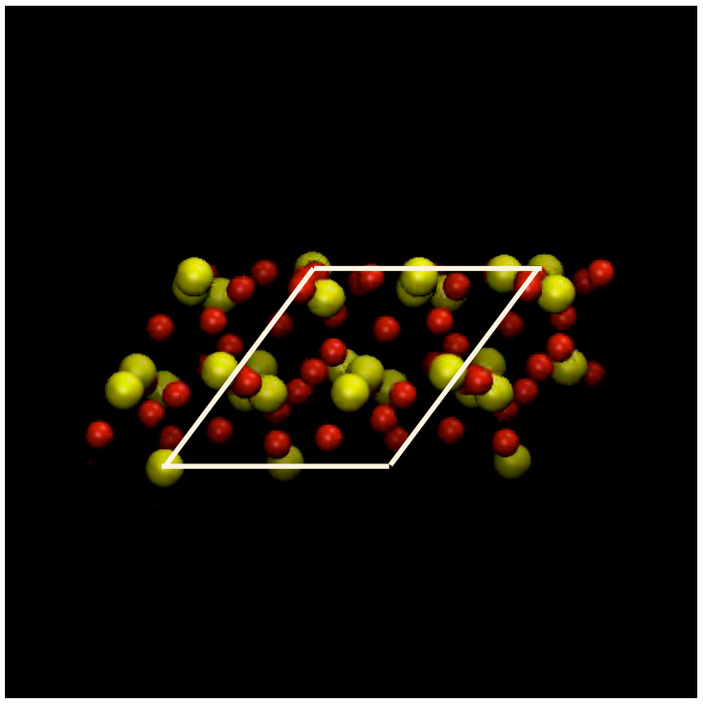


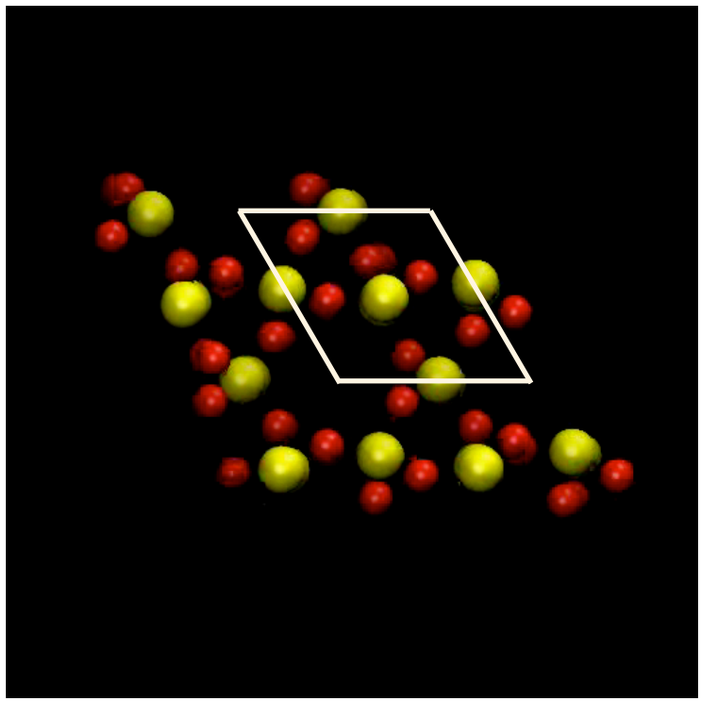
72 atoms

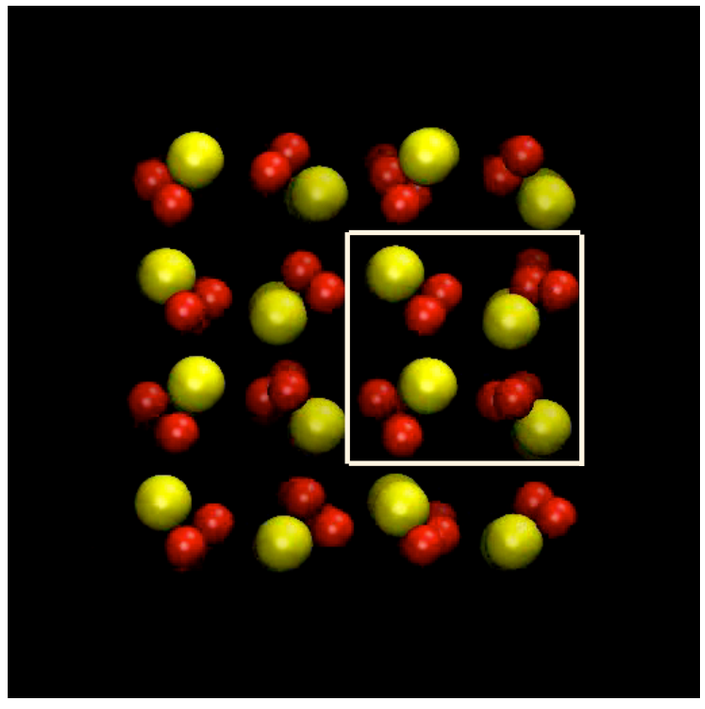
low-cristobalite



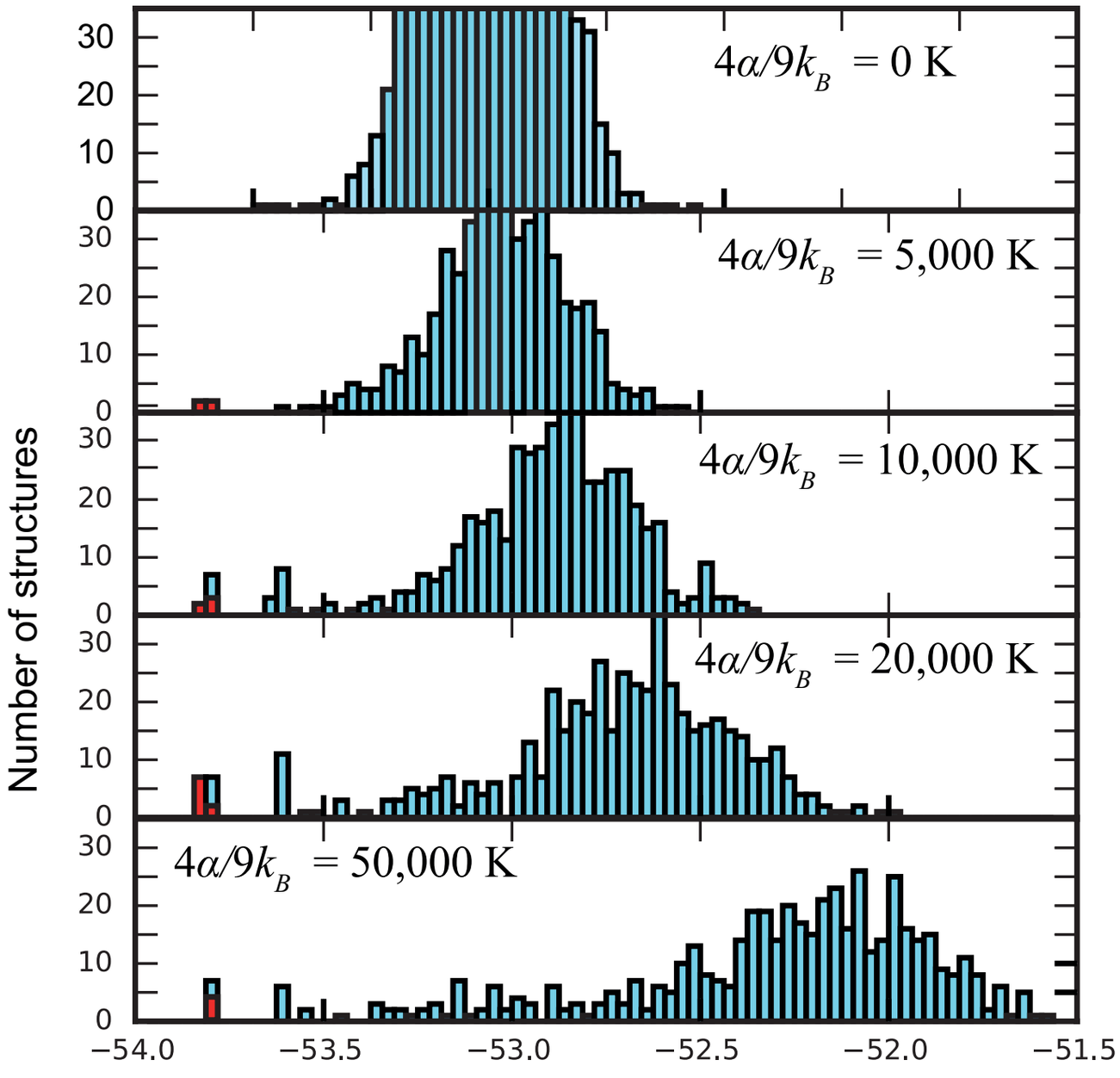
96 atoms





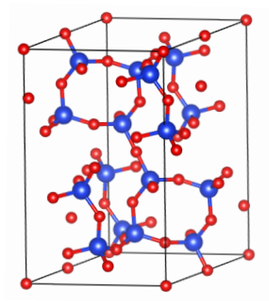


α dependence of cost function



$$F(\mathbf{R}) = E(\mathbf{R}) + \alpha N D[g(\mathbf{R}), g_{\mathrm{obs}}]$$

- **Systems** : unit cell of coesite (48 atoms)

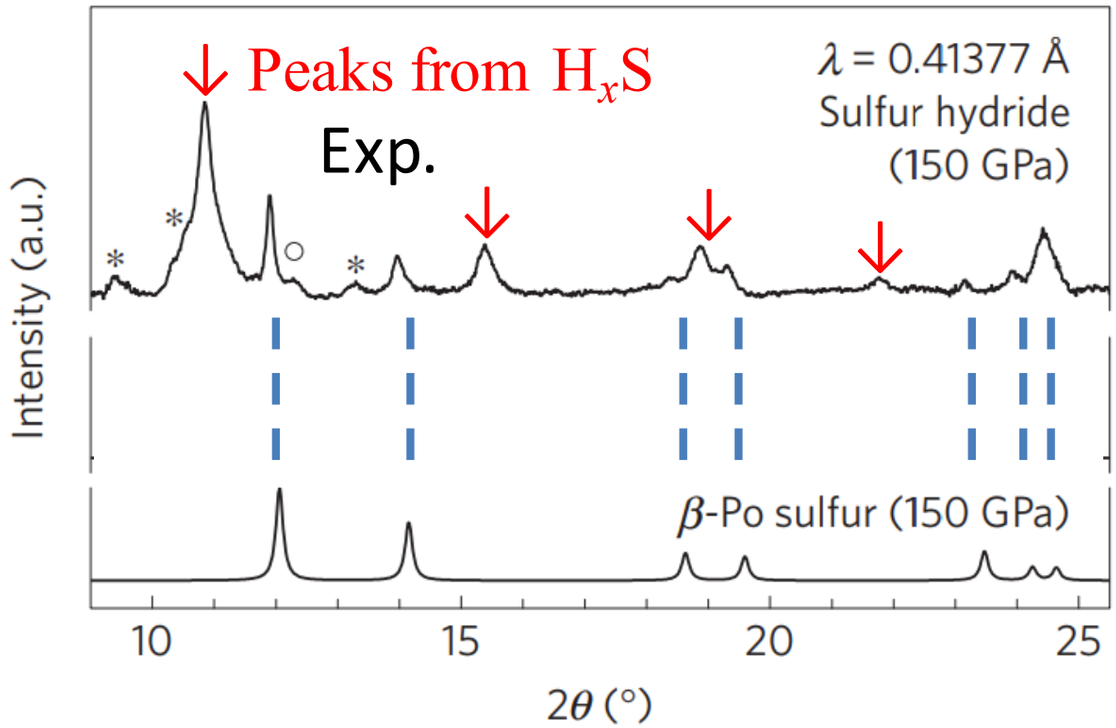


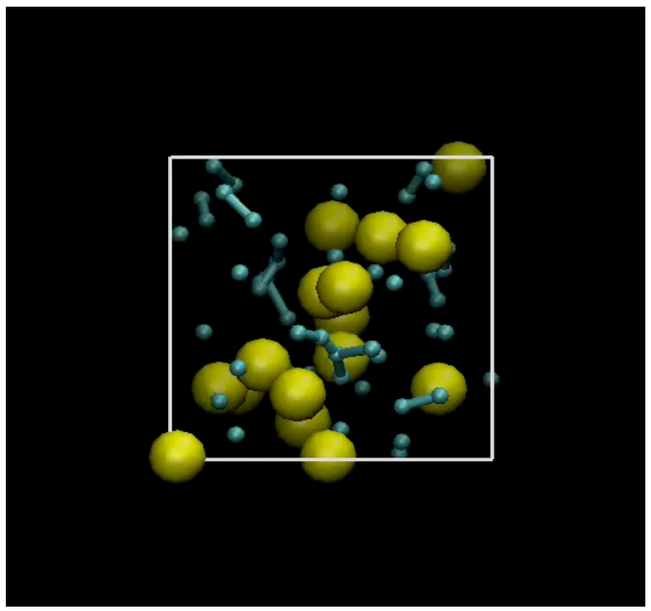
- **Optimization** :

  NVT at 500 K for 5000 steps
  and then quenched to 0 K for 200 steps

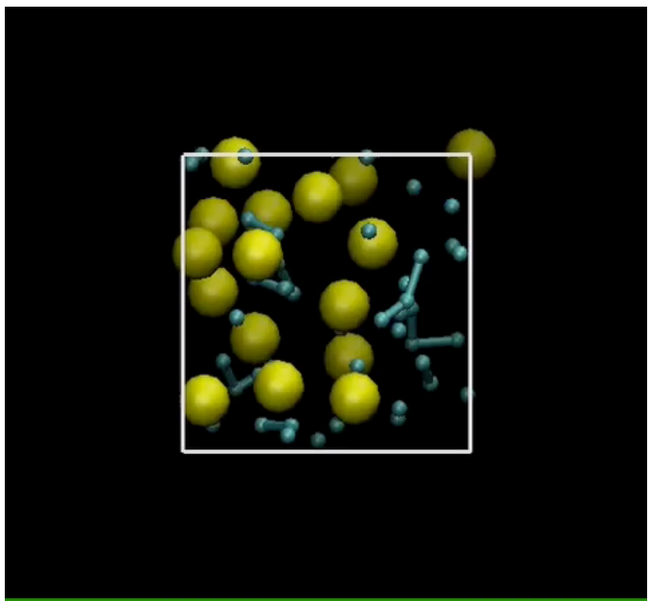
N. Tsujimoto, D. Adachi, R. Akashi, ST, S. Tsuneyuki, Phys. Rev. Materials 2, 053801 (2018)

Structure simulation of high-Tc superconductor H_xS





simulated annealing



present method

- Position of hydrogen atoms cannot be determined from X-ray diffraction
- Exp. diffraction data are taken from M. Einaga et al, Nature Phys. (2016)

- Ordinary simulated annealing is trapped to local minimum and fails to reproduce H_3S crystal structure
- Potential energy is calculated by VASP

Data assimilation approach

- Combination of multiple simulation engines/tools and algorithms
  - Engine 1: total energy calculation by first-principles calculation / classical force field from atomic positions
  - Engine 2: X-ray diffraction pattern calculation from atomic positions
  - Optimizer: joint optimization of multiple cost functions
    - gradient descent, random search, simulated annealing, parallel tempering, particle-swam optimization, evolutionary algorithm, Bayesian optimization
    - sum of two or more cost functions that share the same global minimum but have different local-minima distributions → new class of optimization problem
    - COM (combined optimization method) for new class of optimization

      D. Adachi, N. Tsujimoto, R. Akashi, ST, S. Tsuneyuki, preprint arXiv:1808.06846
- Engines are called from optimizer at each iteration step
  - simulation engines have to be converted into libraries

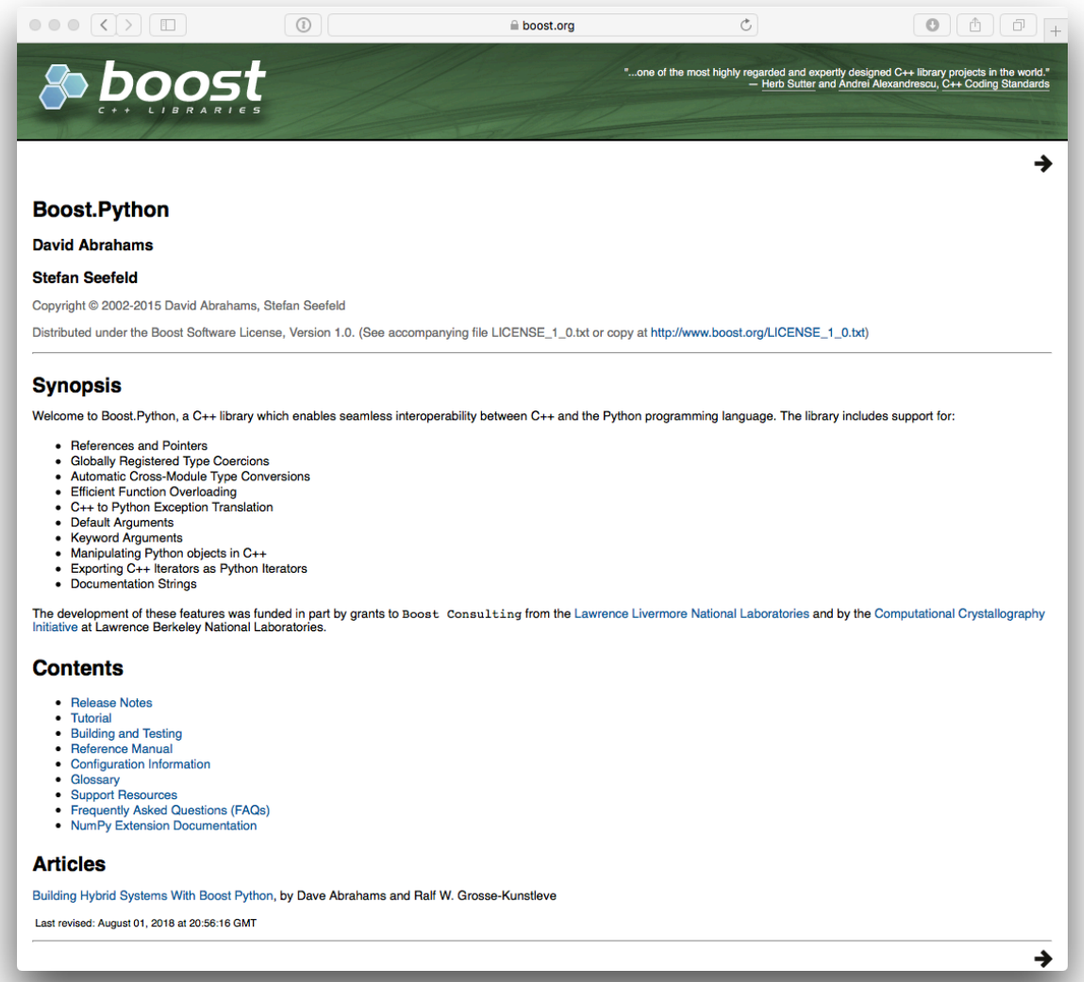
Using ALPS/scheduler framework

- Benefit of using ALPS/scheduler
  - automatic parallelization on parameter set
  - checkpoint/restart
  - load balancing
  - output in XML/HDF5 and analysis tools in Python
- Users have to write their simulation code as "worker class" in C++ [example]
  - constructor, init_observables member function for initialization
  - run, is_thermalized, progress member functions for proceed simulation
  - save, load for checkpoint/restart
- ALPS/scheduler will construct, and call run again and again until progress member function returns 1.

Conversion of application program into library

- "Modularization" of simulation engine
  - Execute as an external program, e.g. ASE (Atomic Simulator Environment)
    - prepare input file, run, and parse output file
    - noninvasive, but fragile against small change in output format
    - large initialization overhead when called multiple times
  - Some simulation codes provide their internal functions as libraries, e.g. LAMMPS
  - Conversion of third-party application programs
    - remove "main" and generate static/shared library archive file (xxx.a or xxx.so)
    - prepare header file
    - depend heavily on application design - thread safety? multiple instances?
- Writing wrapper for Python

pybind11 & Boost.Python



- Can wrap your C++ functions/classes and provide them as python modules
- pycxx: Templates for Python-C++ bindings https://github.com/todo-group/pycxx

Libraries used in computational materials science

- Libraries for parallelization, I/O
  - MPI, HDF5
- "Numerical" Libraries
  - matrix operations, eigenvalue problems, linear equations (BLAS, LAPACK, ...)
  - fast Fourier transform, random number generator, optimization, etc
- Domain specific libraries
  - ALPS libraries for quantum lattice models
- Combination of simulation engines, algorithms, tools, etc
  - Data assimilation approach for crystal structure prediction
  - ALPS framework for large-scale parallel simulations
  - Conversion of application program into library