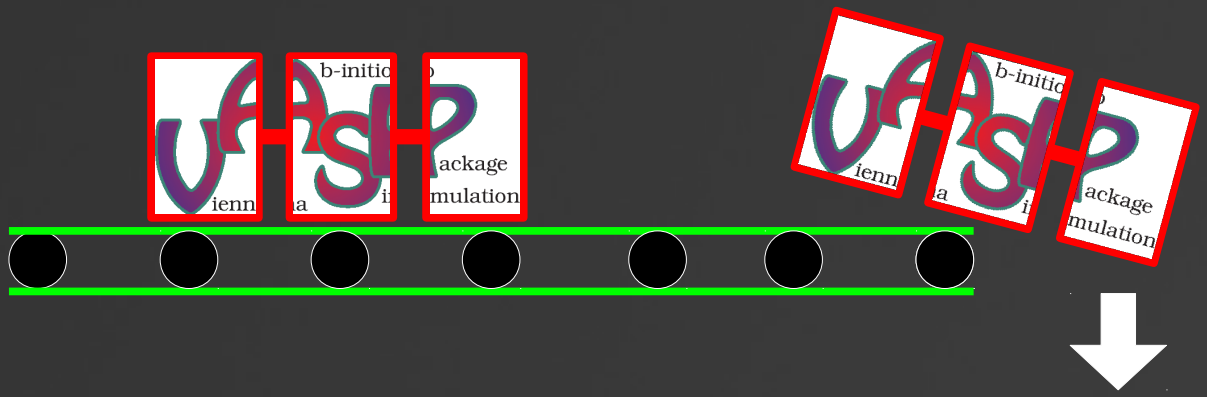
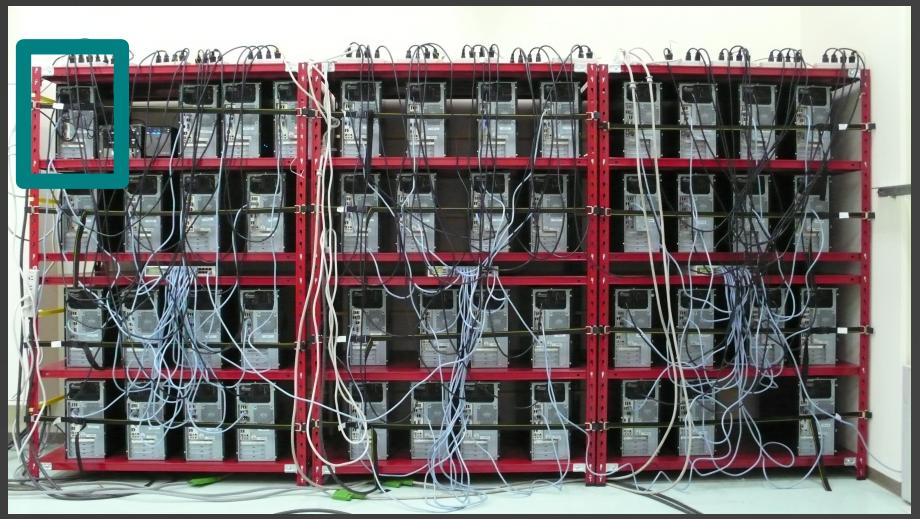
Automation system

Cogue

https://github.com/atztogo/cogue

Automation with queueing system

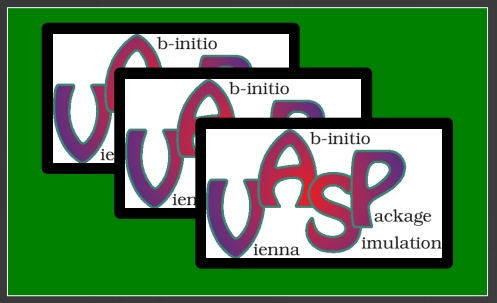




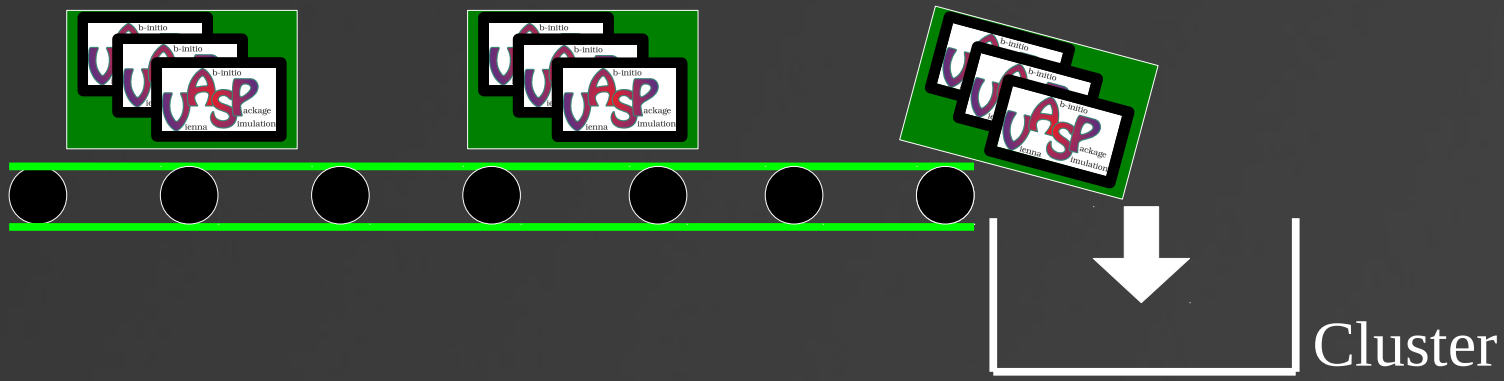
Automation

- Automation is hierarchical.
  - VASP calculation, well automated.

  - Bunch of VASP calculations may be automated by a shell-script.

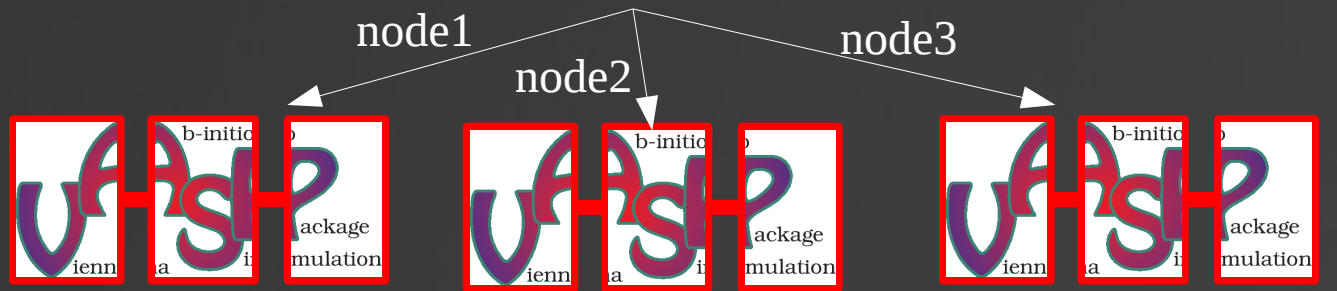


  - Shell-scripts may be automatically and flexibly executed on a computer cluster using a queueing system.

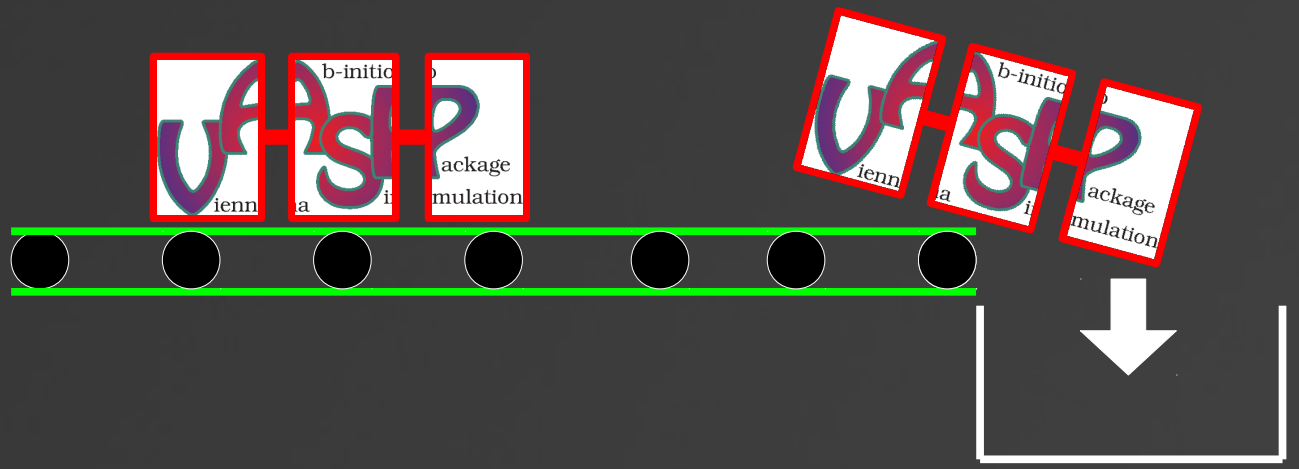


Distributed computation

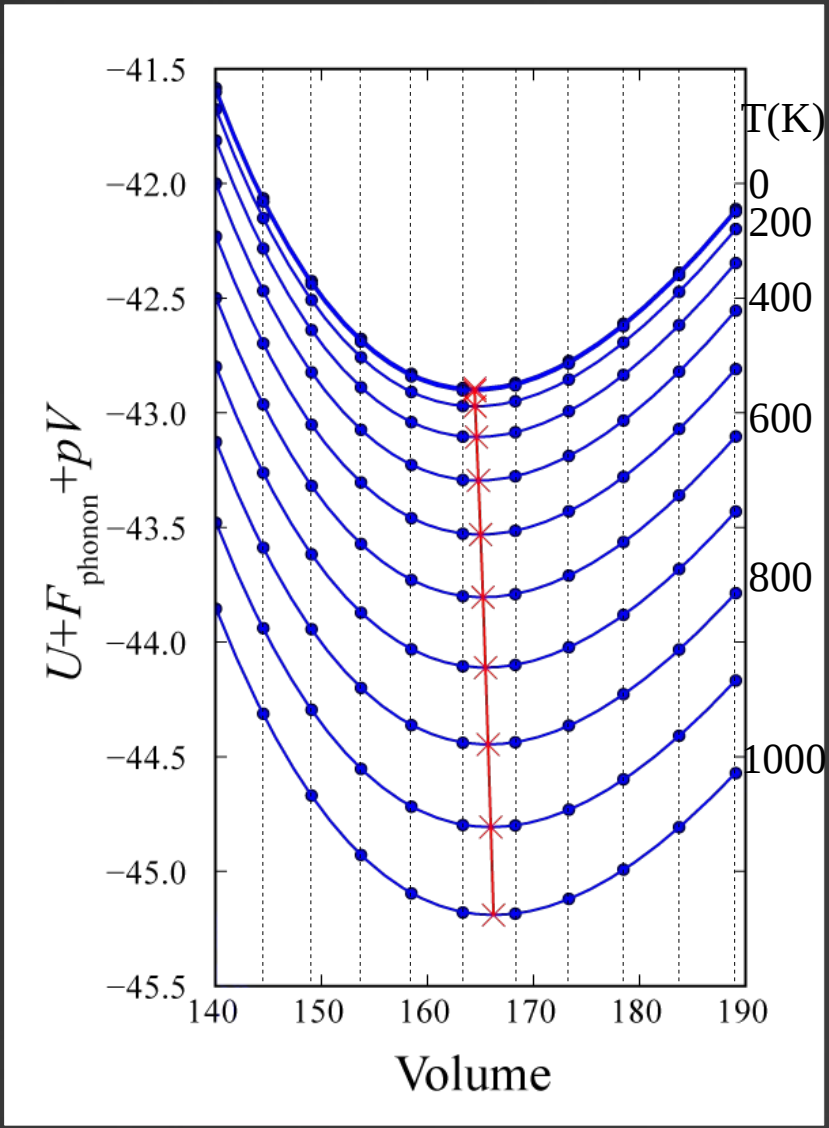
- Distribution is hierarchical.
  - VASP calculation, well distributed by MPI
  - Maybe distributed by a shell-script.



  - Maybe distributed using a queueing system.

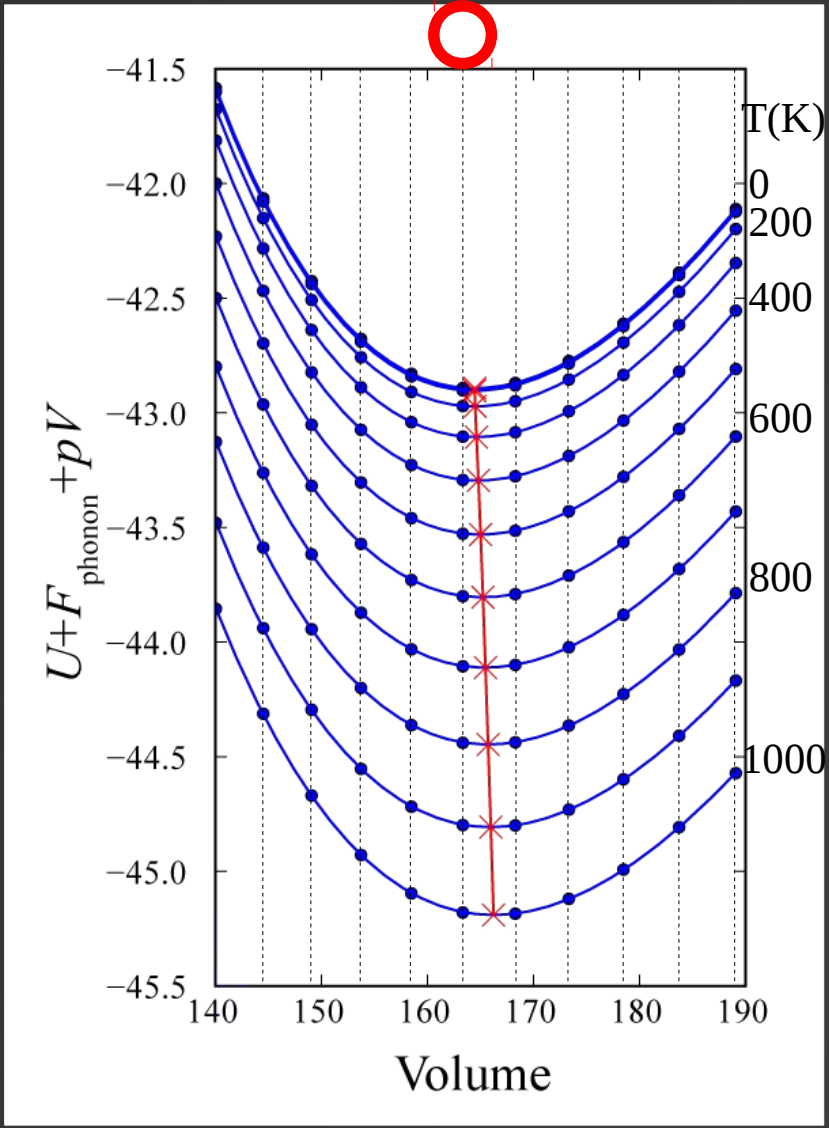


Example:
Thermal expansion calculation



1. Structure optimization of a unit cell
2. Create 10 unit cells with different volumes
3. Structure optimization of 10 unit cells
4. 11 phonon calculations
5. Quasi-harmonic fitting

Diagram of thermal expansion calculation

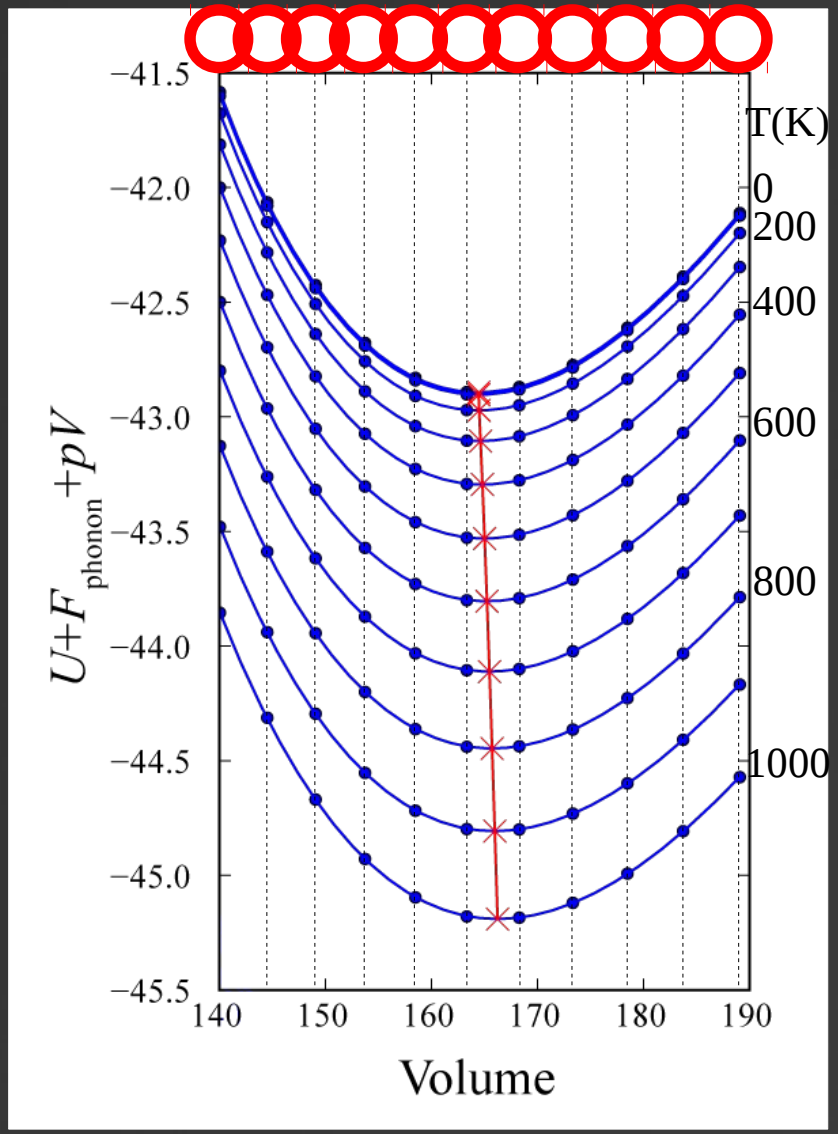


Structure optimization

Stage 1

Determine initial volume

Diagram of thermal expansion calculation



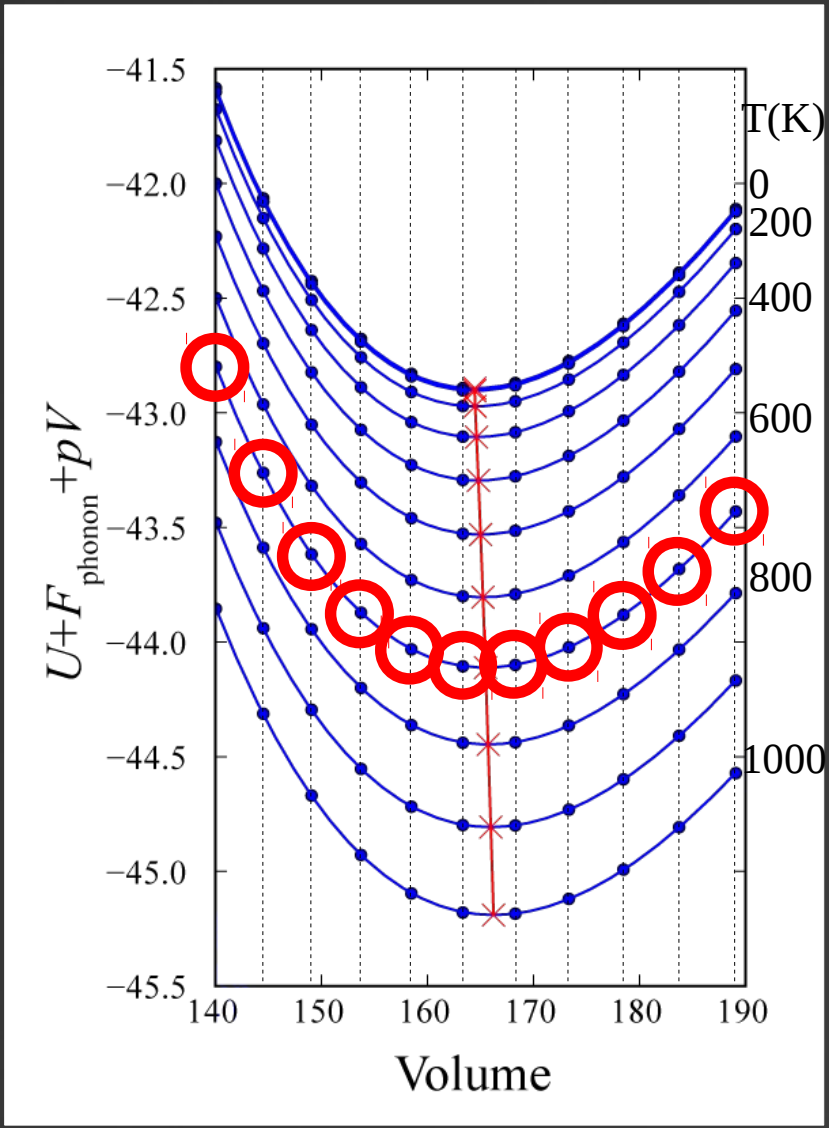
Structure optimization

Stage 1

Determine initial volume

Create unit cells

Diagram of thermal expansion calculation



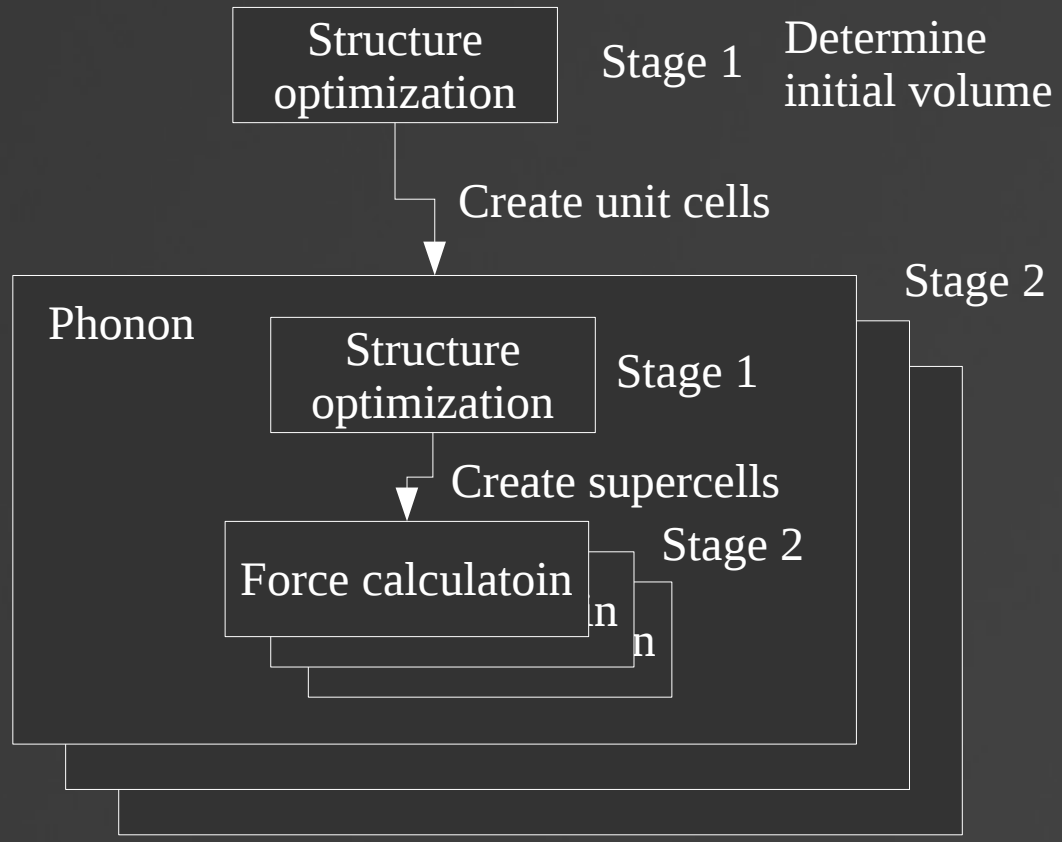
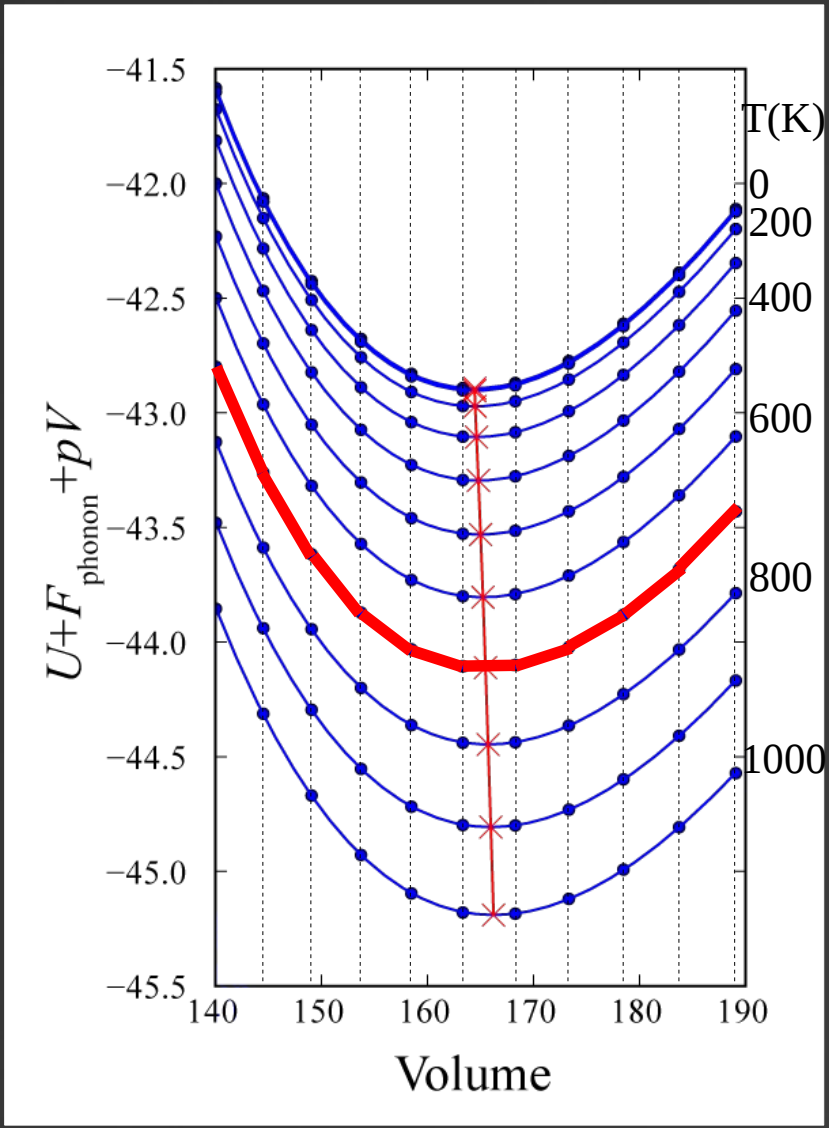


Diagram of thermal expansion calculation



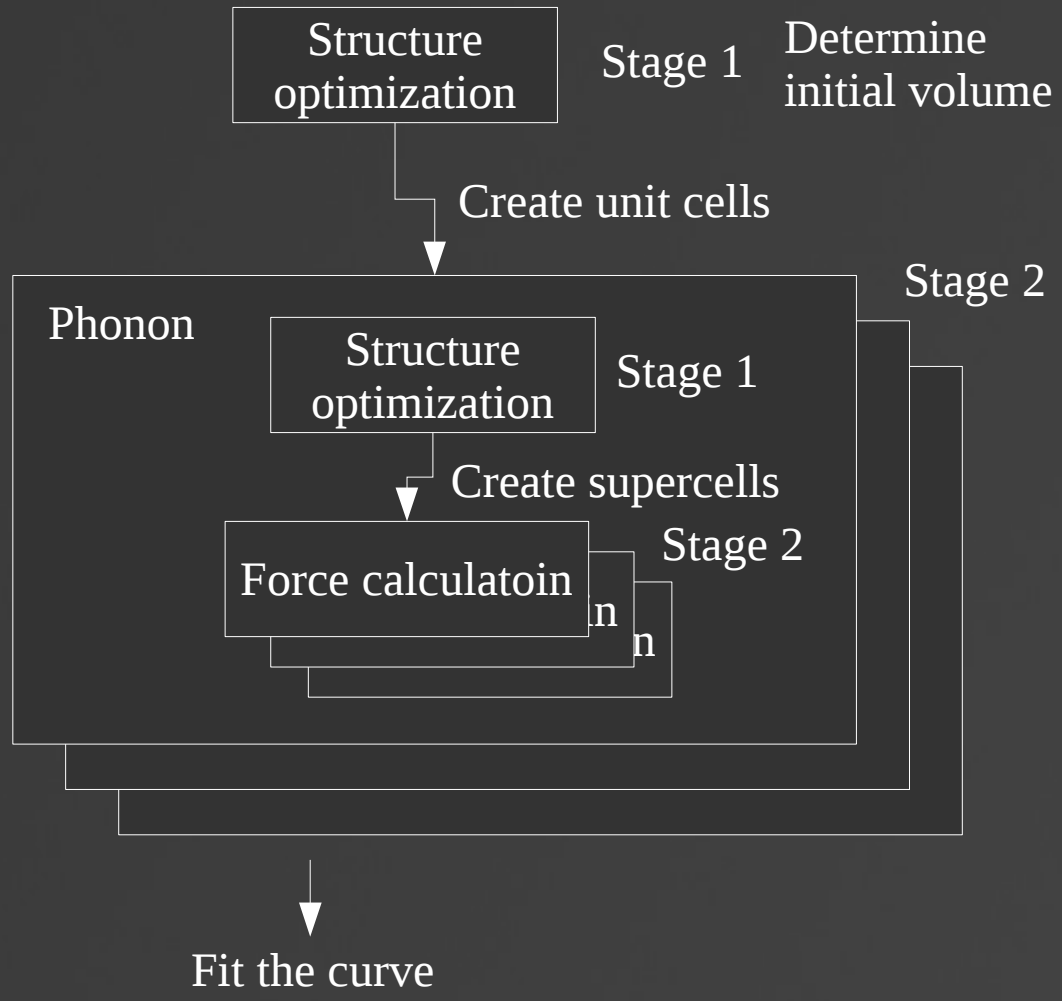
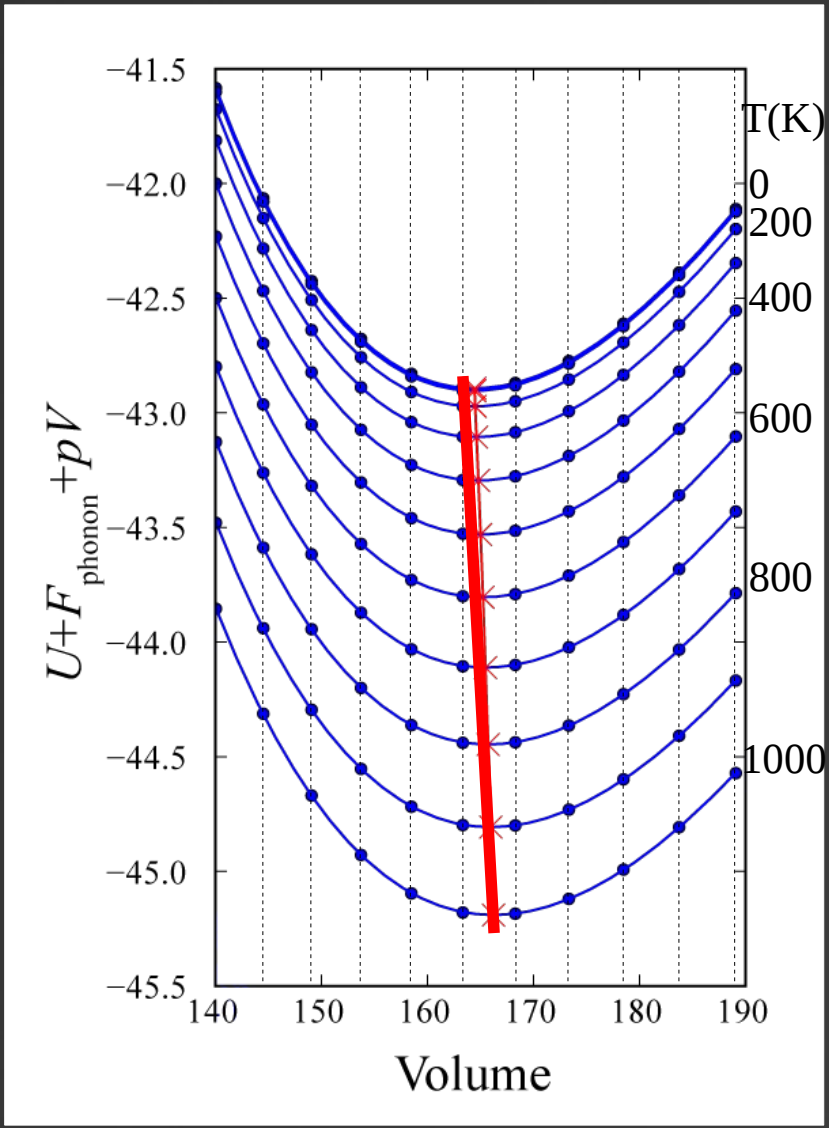
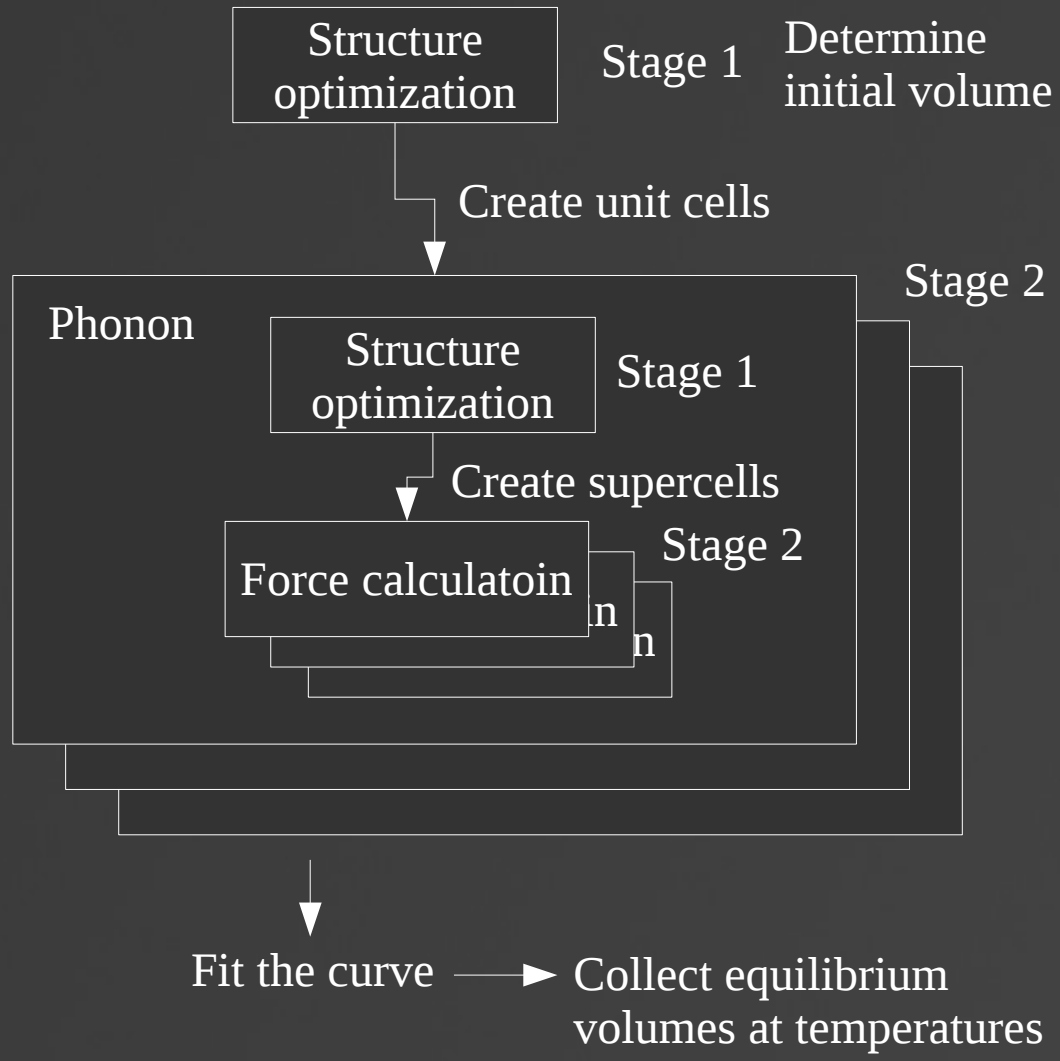


Diagram of thermal expansion calculation

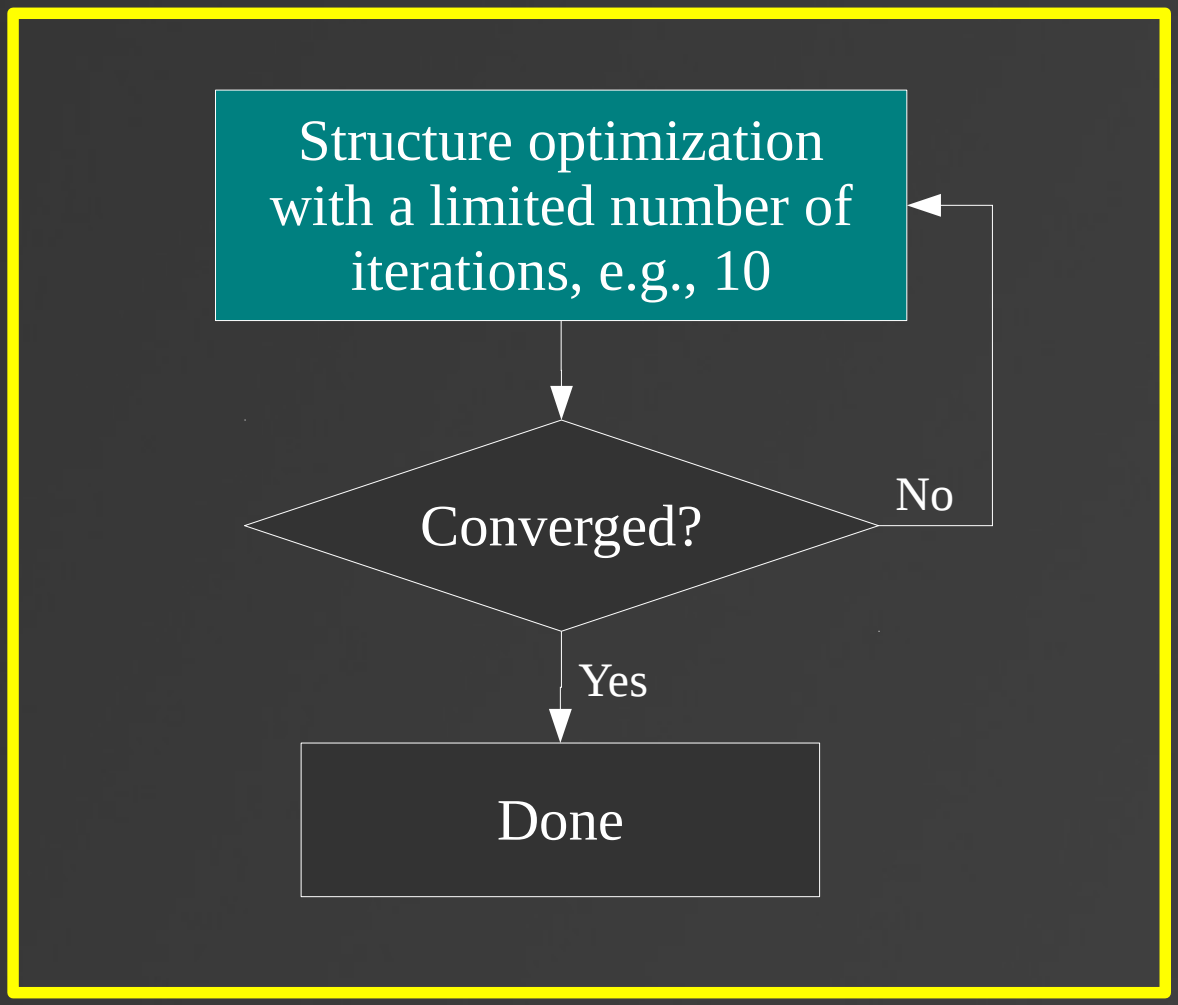




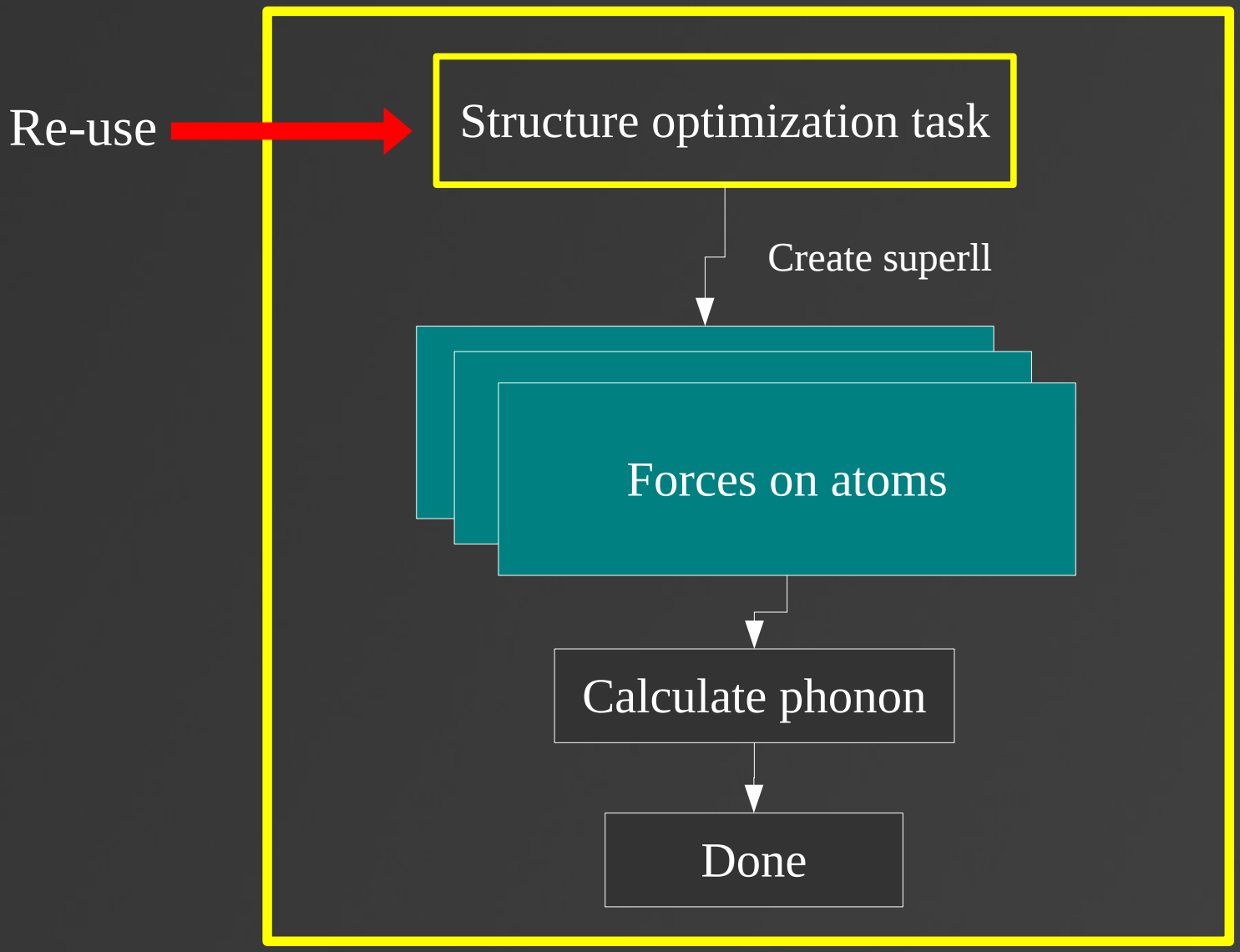
Implementations of execution tasks that are submitted to queueing system

Structure optimization with a limited number of iteration, e.g., 10	Forces on atoms
Energy	Stress
Elastic constants	Dielectric constant Born effective charges

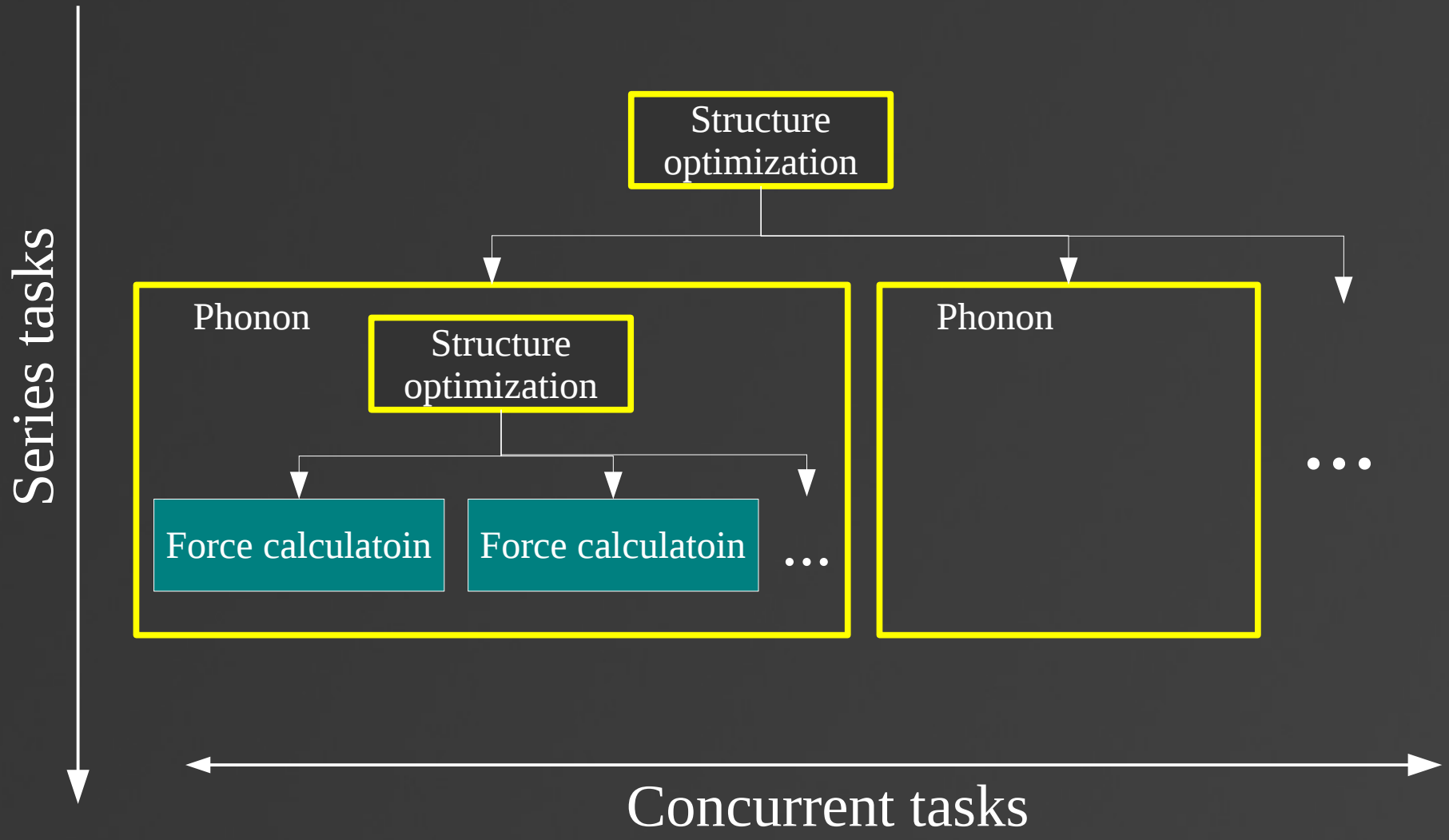
Implementation of general structure optimization task



Implementation of phonon task



Implementation of thermal expansion calculation



Kernel of automation system

```
def _deep_run(self, task):
    cwd = self._chdir_in(task.get_directory())

    subtasks = task.get_tasks()
    if subtasks: # taskset
        for subtask in subtasks:
            if not subtask.done():
                self._deep_run(subtask)

    else: # job task
        self._queue.set_job_status(task)

    task.set_status()
    if task.done():
        try:
            next_subtasks = task.next()
        except StopIteration:
            task.end()
        else:
            for next_subtask in next_subtasks:
                self._deep_begin(next_subtask)

    self._chdir_out(cwd, task.get_status())
```

Concurrent tasks

Recursive call of subtask

Series task

*Standardization

To control tasks, each task has to hold the following methods:

Task.begin()	Task.end()
Task.next()	Task.done()

Script for bulk modulus calculation

```python
#!/usr/bin/env python

import numpy as np
import cogue
import cogue.calculator.vasp as vasp
import cogue.qsystem.gridengine as ge

task_name = "sno2"

# Crystal structure
symbols = ['Sn'] * 2 + ['O'] * 4
lattice = [[4.75, 0, 0],
           [0, 4.75, 0],
           [0, 0, 3.25]]
points=np.transpose([[0.0, 0.0, 0.0],
                     [0.5, 0.5, 0.5],
                     [0.3, 0.3, 0.0],
                     [0.7, 0.7, 0.0],
                     [0.2, 0.8, 0.5],
                     [0.8, 0.2, 0.5]])
cell = cogue.cell(lattice=lattice,
                  points=points,
                  symbols=symbols)

# Vasp settings
ps_map = {'Sn': 'Sn_PBE',
          'O': 'O_PBE'}
incar = vasp.incar()
incar.set_structure_optimization()
incar.set_encut(400)
incar.set_prec("Normal")
```

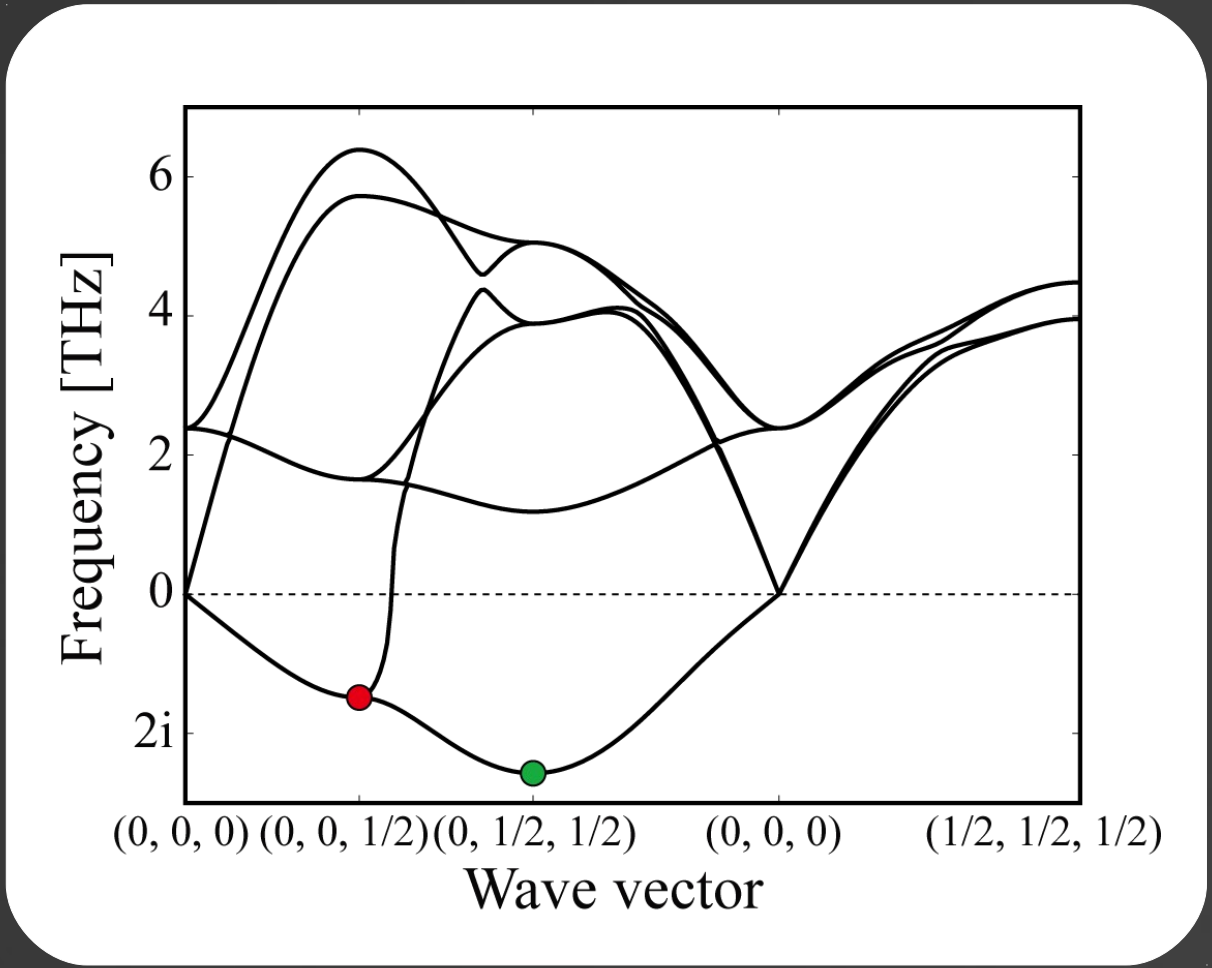
```python
# Queue
job = ge.job(script="vasp5212serial",
             shell="/bin/zsh",
             jobname=task_name,
             stdout="std.log",
             stderr="err.log")

# Task
task = vasp.bulk_modulus(max_iteration=2,
                         cell=cell,
                         pseudo_potential_map=ps_map,
                         k_mesh=[4, 4, 6],
                         incar=incar,
                         job=job)

# Automatic calculation
calc = cogue.autocalc()
calc.append(task_name, task) # More tasks can be appended.
calc.set_queue(ge.queue())
calc.run(check_period=5)
print "space group:", cogue.symmetry(cell)['international']
print "status:", task.get_status()
# 201.411956183 GPa
print "bulk modulus:", task.get_bulk_modulus(), "GPa"
```

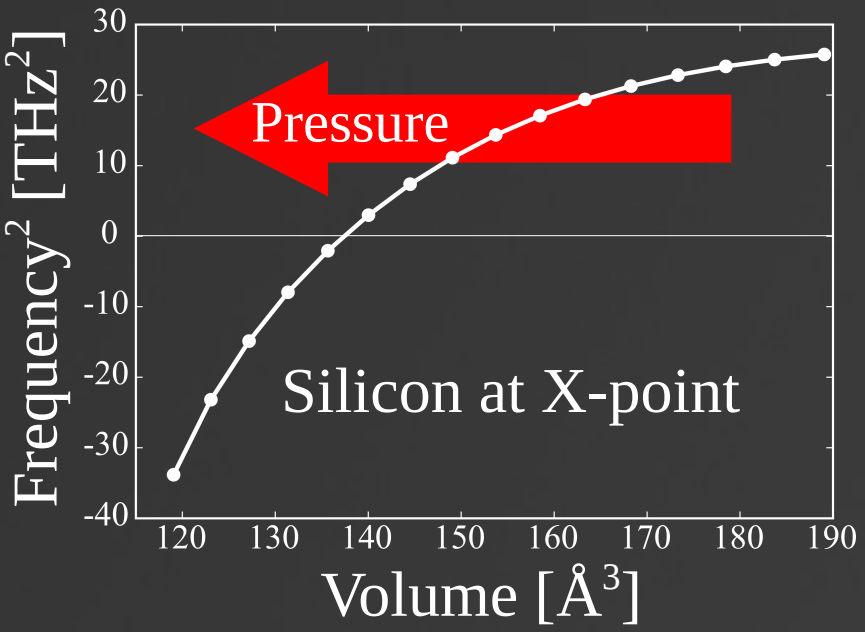
Application of metastable structure search

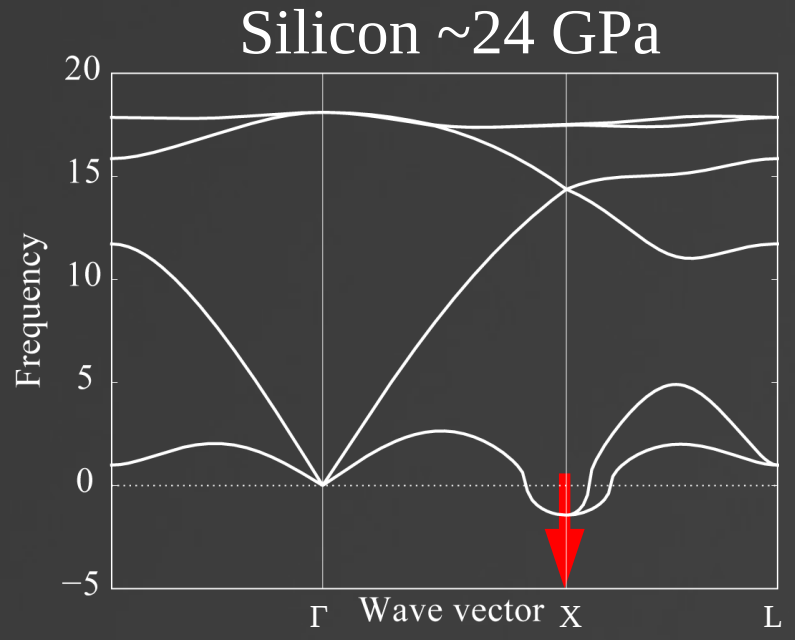
A hypothetical structure: CsCl-type NaCl at p=0

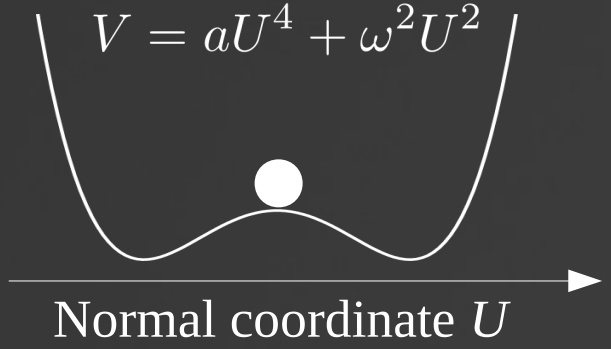


$\omega^2 < 0$: Instability of crystal potential

$$\mathrm{D}(\mathbf{q})\mathbf{e}(\mathbf{q}s) = [\omega(\mathbf{q}s)]^2\mathbf{e}(\mathbf{q}s)$$



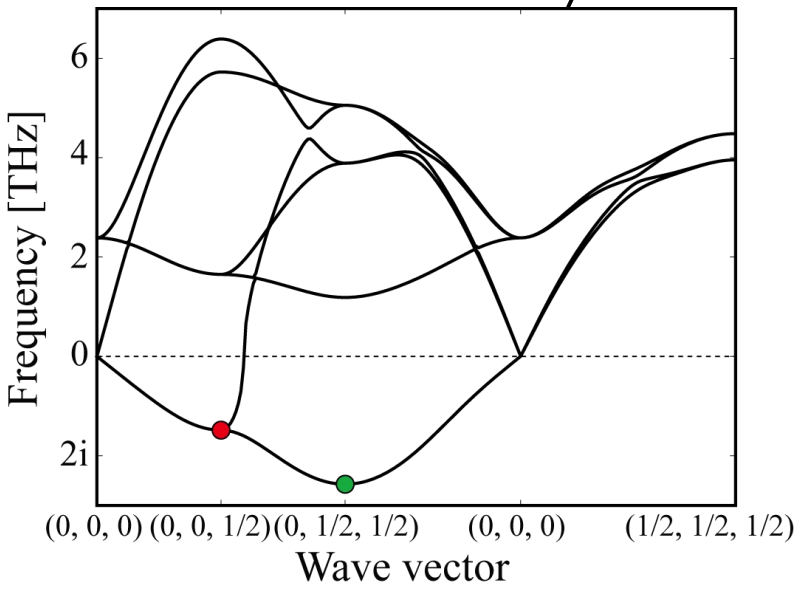


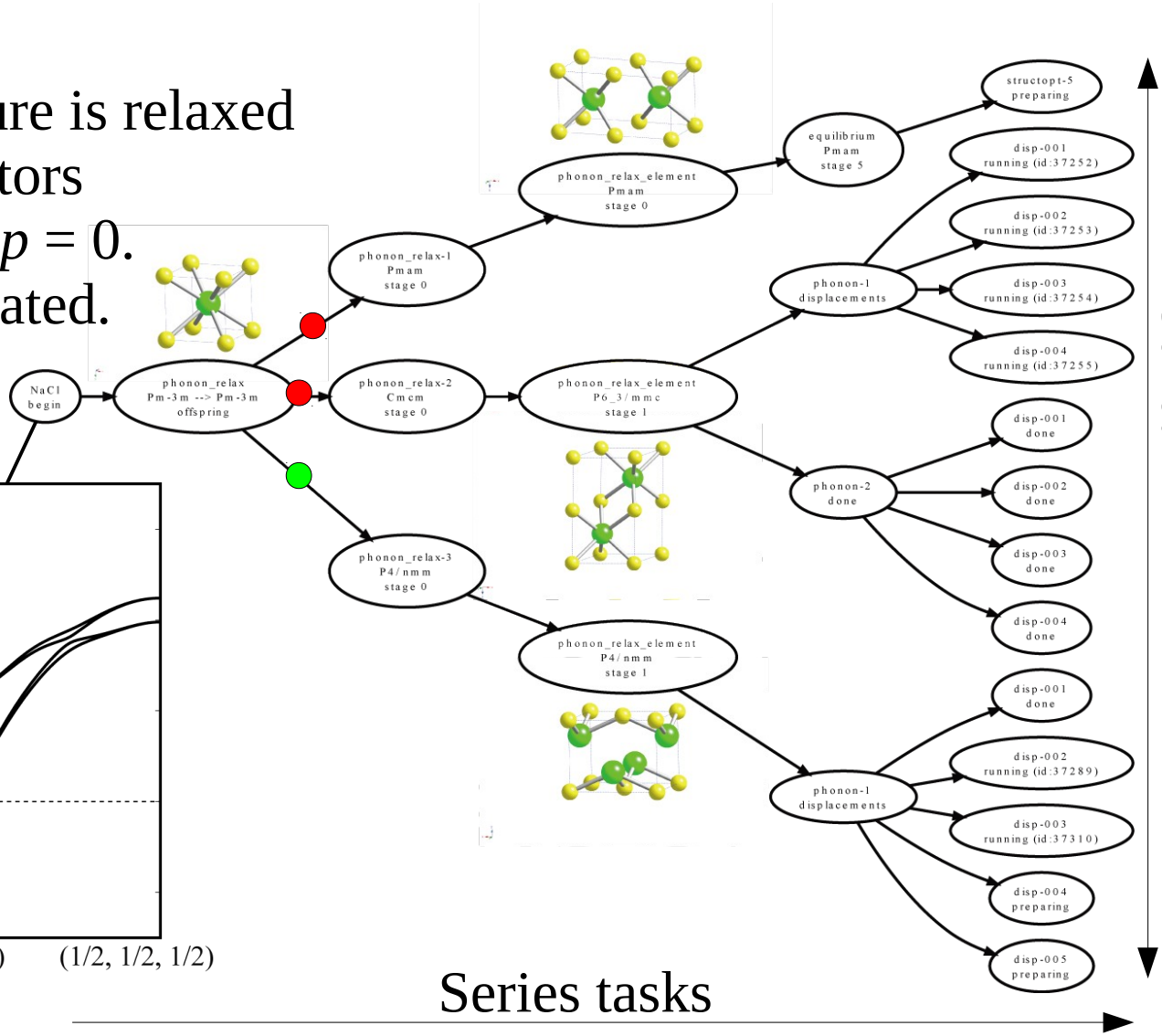


*Imaginary frequencies are shown by negative values.

Automatic structure relaxation

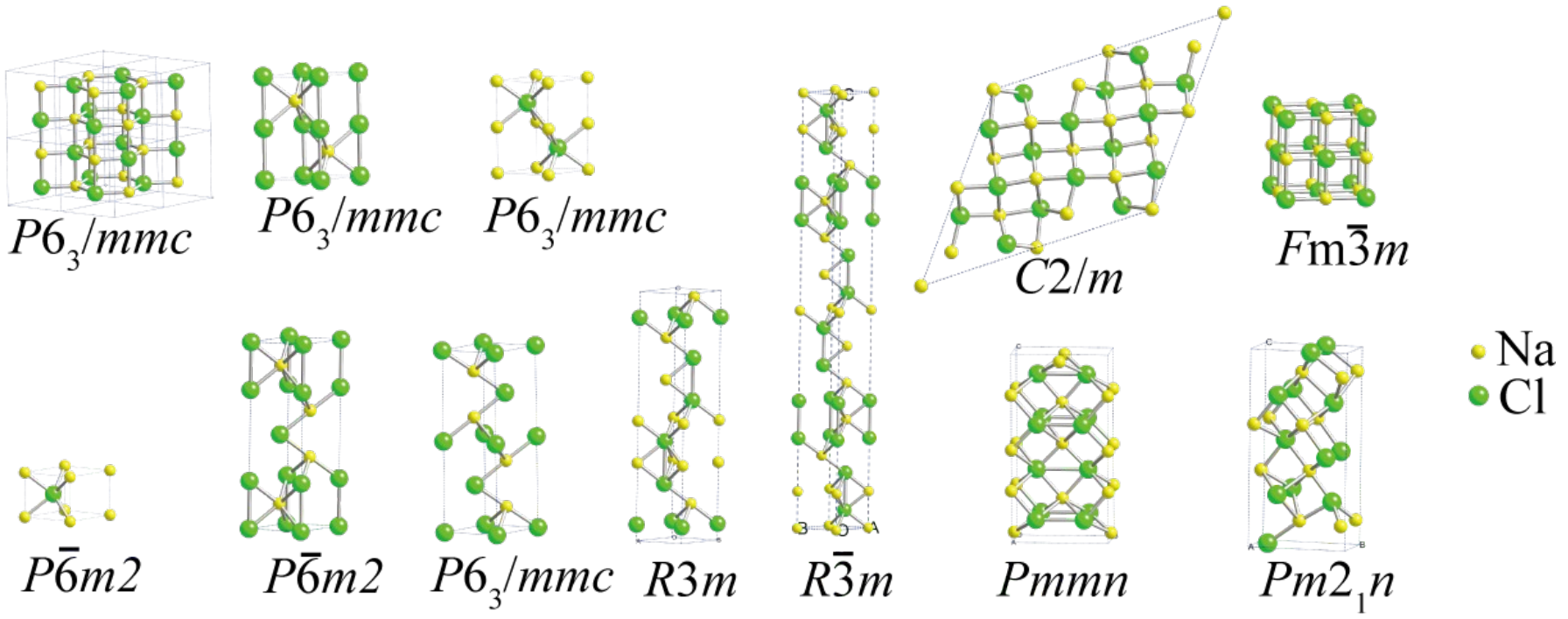
CsCl-type NaCl structure is relaxed following the eigenvectors of imaginary modes at $p = 0$. Calculations are automated.





*Number of series tasks increases on-the-fly.

Dynamically stable structures of NaCl



Most of the structures were found as close-packed alternate stacking of Na and Cl layers.

*See for more details, Togo and Tanaka, Phys. Rev. B 87, 184104 (2013)