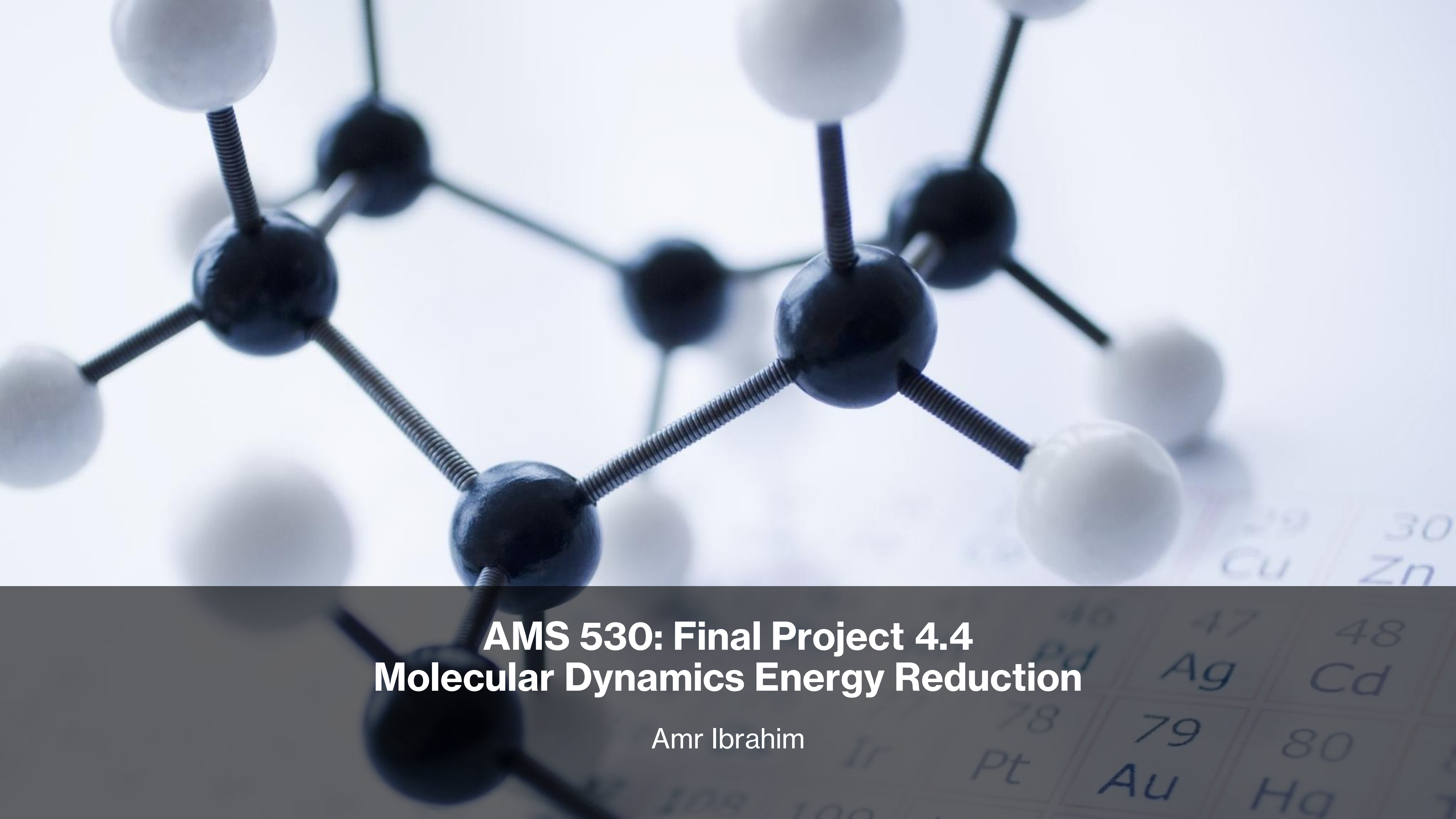


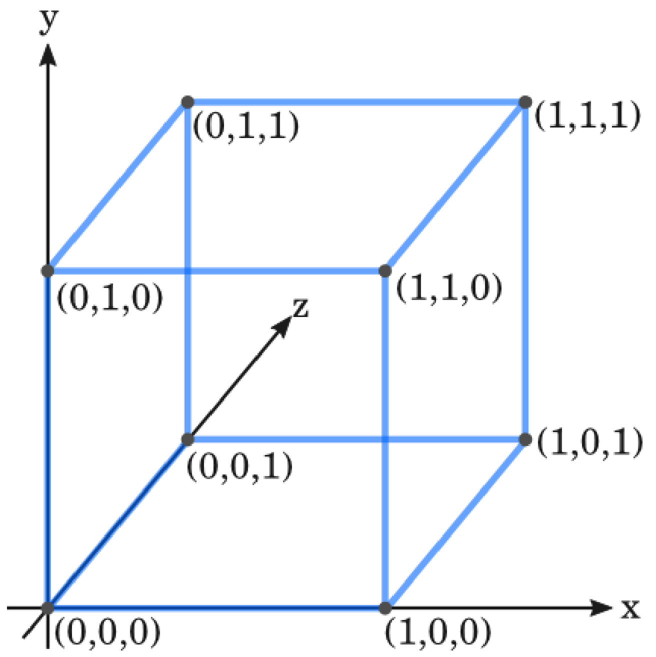
AMS 530: Final Project 4.4
Molecular Dynamics Energy Reduction

Amr Ibrahim

Table of contents

Problem Statement

- Goal: Reduce energy of given system of particles by kicking them around
  - Emphasis on parallelization of energy reduction
- Total Energy is given by
  - $\frac{1}{2}\sum_{j\neq i}^{N} V_{ij}$
  - Lenard-Jones potential (ignoring constants)
    - $V_{ij} = \frac{1}{r_{ij}^{12}} - \frac{2}{r_{ij}^{6}}$
    - $r_{ij} = |x_i - x_j|$
- Initial conditions:
  - Box 10x10x10
  - N = 11x11x11 particles placed at box grid points

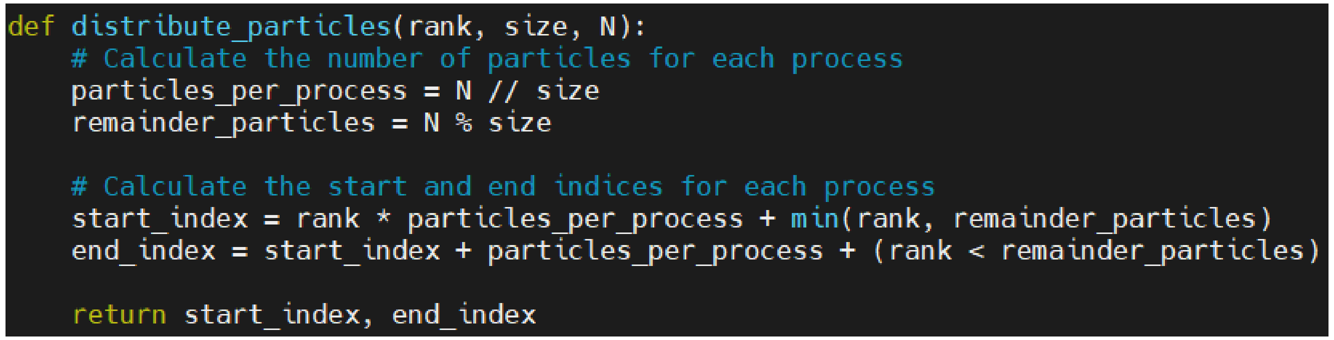


Program: General Design

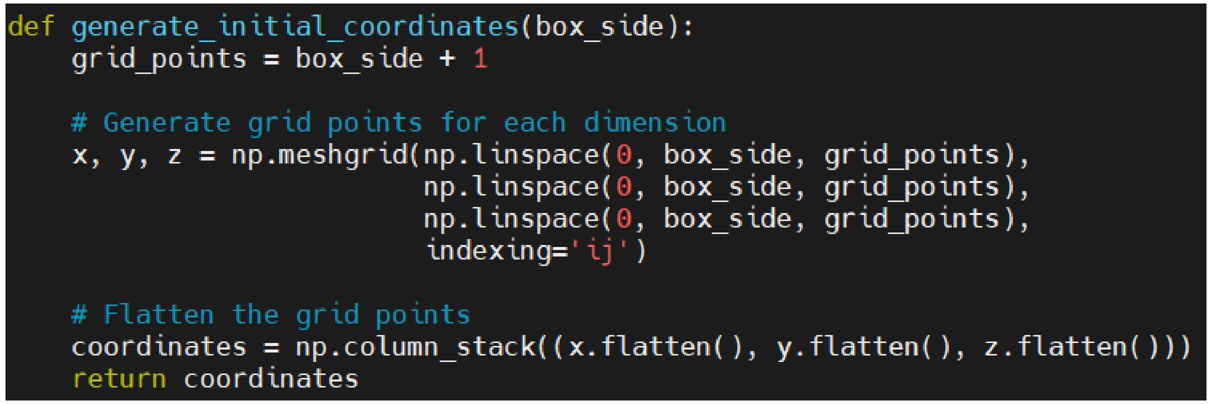
1. Generate initial coordinates & initial energy E_0
   - Initial coordinates generated by all processes
   - Particles distributed among processes
   - Energy calculation
2. Move a particle randomly with max distance 0.5
   - Moving a fraction of particles at a time
   - Discard moves that do not reduce energy
   - If particle moves outside the box return it from the opposite side
3. Repeat step 2 for 100 steps and print out energy if lowered

Program

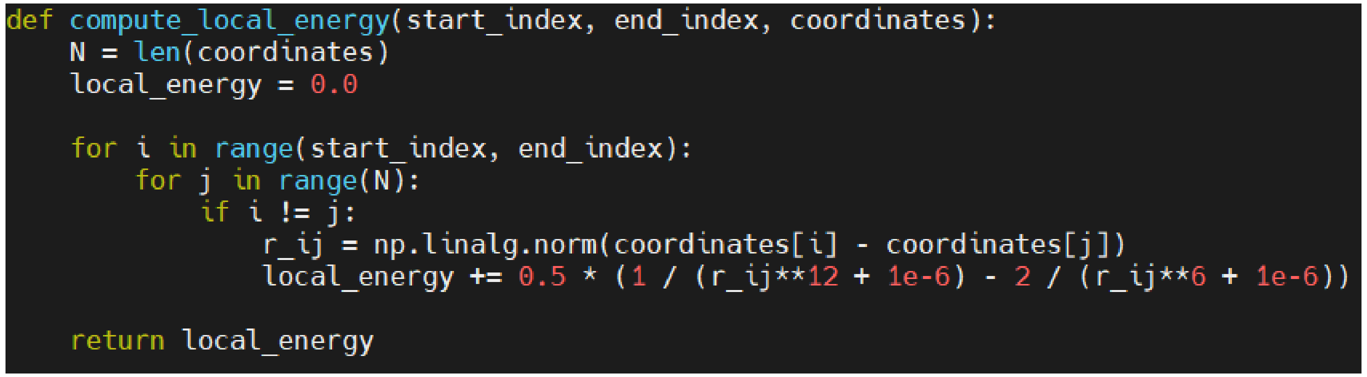
- Distribute particles
  - Based on N // size
  - Add remainder particles evenly across processes
- Generate initial coordinates
  - For all processes
  - All processes require full coordinates to compute initial energy
- Compute local energy
  - Compute interactions of local particles
  - Total energy = $\sum_{i=0}^{P} local\ energy_i$



```python
def distribute_particles(rank, size, N):
    # Calculate the number of particles for each process
    particles_per_process = N // size
    remainder_particles = N % size

    # Calculate the start and end indices for each process
    start_index = rank * particles_per_process + min(rank, remainder_particles)
    end_index = start_index + particles_per_process + (rank < remainder_particles)

    return start_index, end_index
```

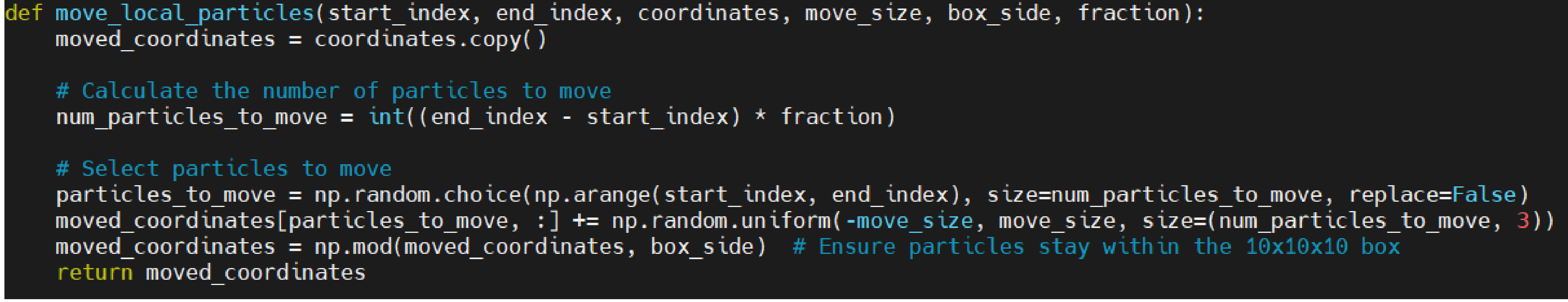


```python
def generate_initial_coordinates(box_side):
    grid_points = box_side + 1

    # Generate grid points for each dimension
    x, y, z = np.meshgrid(np.linspace(0, box_side, grid_points),
                          np.linspace(0, box_side, grid_points),
                          np.linspace(0, box_side, grid_points),
                          indexing='ij')

    # Flatten the grid points
    coordinates = np.column_stack((x.flatten(), y.flatten(), z.flatten()))
    return coordinates
```



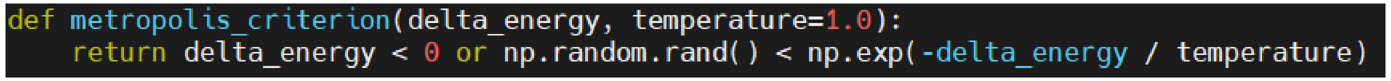
```python
def compute_local_energy(start_index, end_index, coordinates):
    N = len(coordinates)
    local_energy = 0.0

    for i in range(start_index, end_index):
        for j in range(N):
            if i != j:
                r_ij = np.linalg.norm(coordinates[i] - coordinates[j])
                local_energy += 0.5 * (1 / (r_ij**12 + 1e-6) - 2 / (r_ij**6 + 1e-6))

    return local_energy
```

Program



```python
def move_local_particles(start_index, end_index, coordinates, move_size, box_side, fraction):
    moved_coordinates = coordinates.copy()

    # Calculate the number of particles to move
    num_particles_to_move = int((end_index - start_index) * fraction)

    # Select particles to move
    particles_to_move = np.random.choice(np.arange(start_index, end_index), size=num_particles_to_move, replace=False)
    moved_coordinates[particles_to_move, :] += np.random.uniform(-move_size, move_size, size=(num_particles_to_move, 3))
    moved_coordinates = np.mod(moved_coordinates, box_side)  # Ensure particles stay within the 10x10x10 box
    return moved_coordinates
```

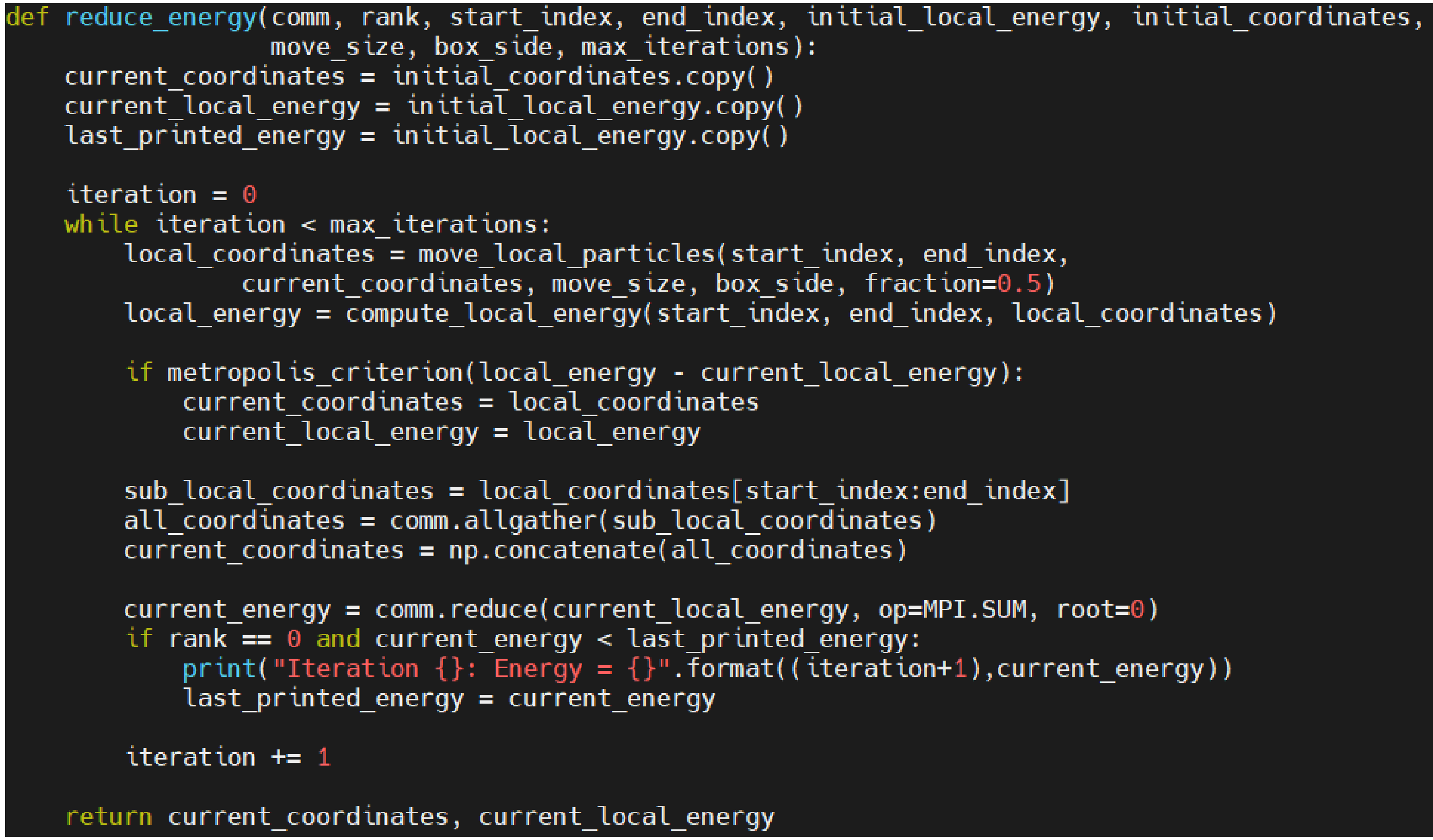
- Move local particles
  - Move a fraction of randomly chosen particles
  - Move with distribution U(-0.5, 0.5)
  - Mod coordinates
- Metropolis criterion (Simulated Annealing)
  - Update local coordinates & local energy if
    - $E_{local_i} - E_{local_0} < 0$
    - $rand < e^{-(E_{local_i} - E_{local_0})/T}, T = 1$
  - In case stuck at local min



```python
def metropolis_criterion(delta_energy, temperature=1.0):
    return delta_energy < 0 or np.random.rand() < np.exp(-delta_energy / temperature)
```

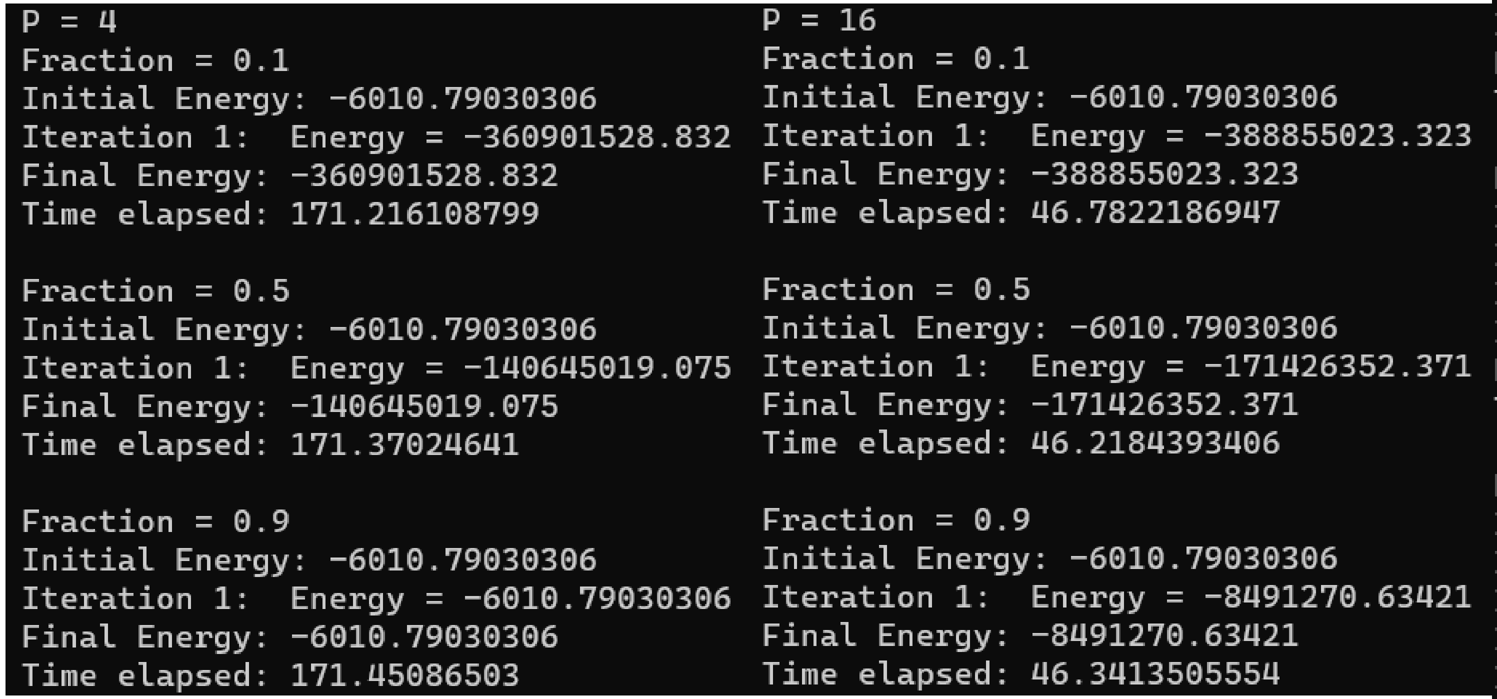
Program

- Reduce energy
  - Move particles
  - Compute local energy
  - Metropolis criterion
  - Extract new local coordinates
    - Each process get their respective local particle coordinates
  - All gather new coordinates
    - Updated coordinates for *next iteration*
  - Compute new total energy
    - New total energy = $\sum_{i=0}^{P} local\ energy_i$



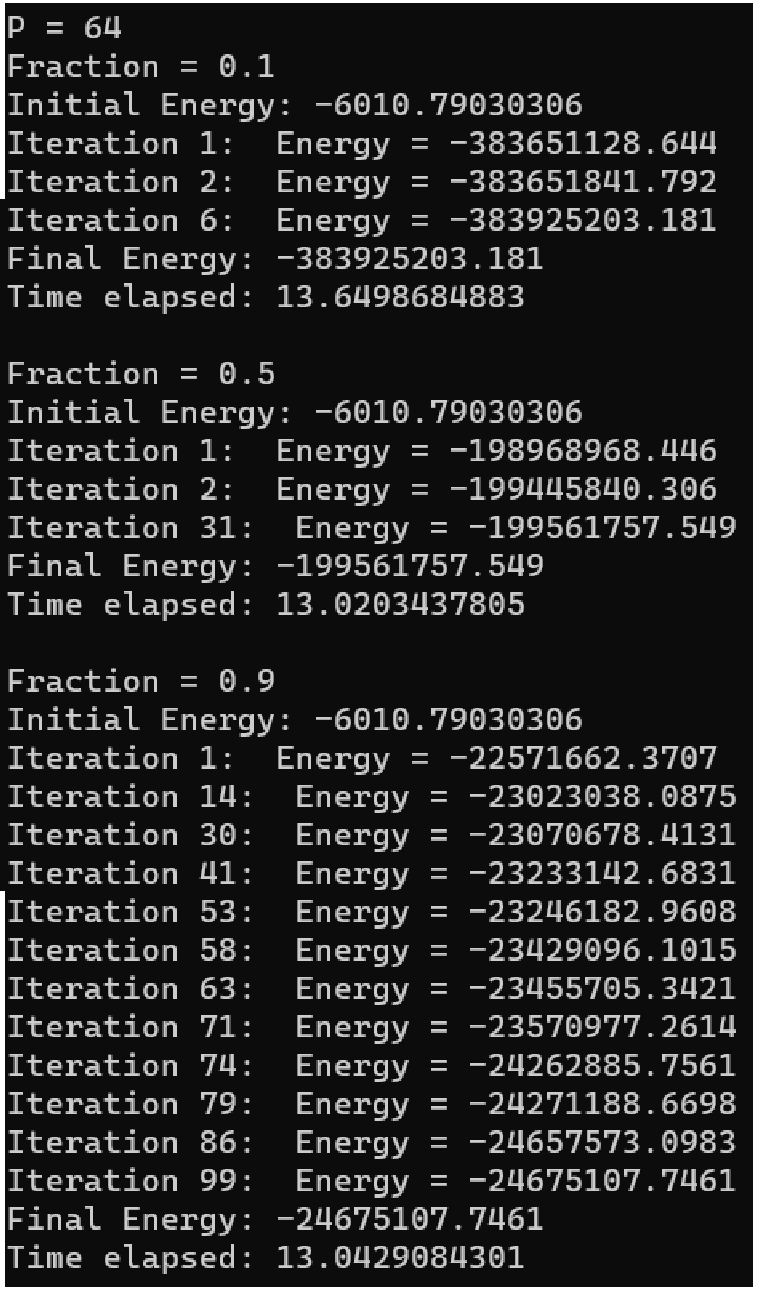
```python
def reduce_energy(comm, rank, start_index, end_index, initial_local_energy, initial_coordinates,
                  move_size, box_side, max_iterations):
    current_coordinates = initial_coordinates.copy()
    current_local_energy = initial_local_energy.copy()
    last_printed_energy = initial_local_energy.copy()

    iteration = 0
    while iteration < max_iterations:
        local_coordinates = move_local_particles(start_index, end_index,
                current_coordinates, move_size, box_side, fraction=0.5)
        local_energy = compute_local_energy(start_index, end_index, local_coordinates)

        if metropolis_criterion(local_energy - current_local_energy):
            current_coordinates = local_coordinates
            current_local_energy = local_energy

        sub_local_coordinates = local_coordinates[start_index:end_index]
        all_coordinates = comm.allgather(sub_local_coordinates)
        current_coordinates = np.concatenate(all_coordinates)

        current_energy = comm.reduce(current_local_energy, op=MPI.SUM, root=0)
        if rank == 0 and current_energy < last_printed_energy:
            print("Iteration {}: Energy = {}".format((iteration+1),current_energy))
            last_printed_energy = current_energy

        iteration += 1

    return current_coordinates, current_local_energy
```

Results



```
P = 4
Fraction = 0.1
Initial Energy: -6010.79030306
Iteration 1:  Energy = -360901528.832
Final Energy: -360901528.832
Time elapsed: 171.216108799

Fraction = 0.5
Initial Energy: -6010.79030306
Iteration 1:  Energy = -140645019.075
Final Energy: -140645019.075
Time elapsed: 171.37024641

Fraction = 0.9
Initial Energy: -6010.79030306
Iteration 1:  Energy = -6010.79030306
Final Energy: -6010.79030306
Time elapsed: 171.45086503
```

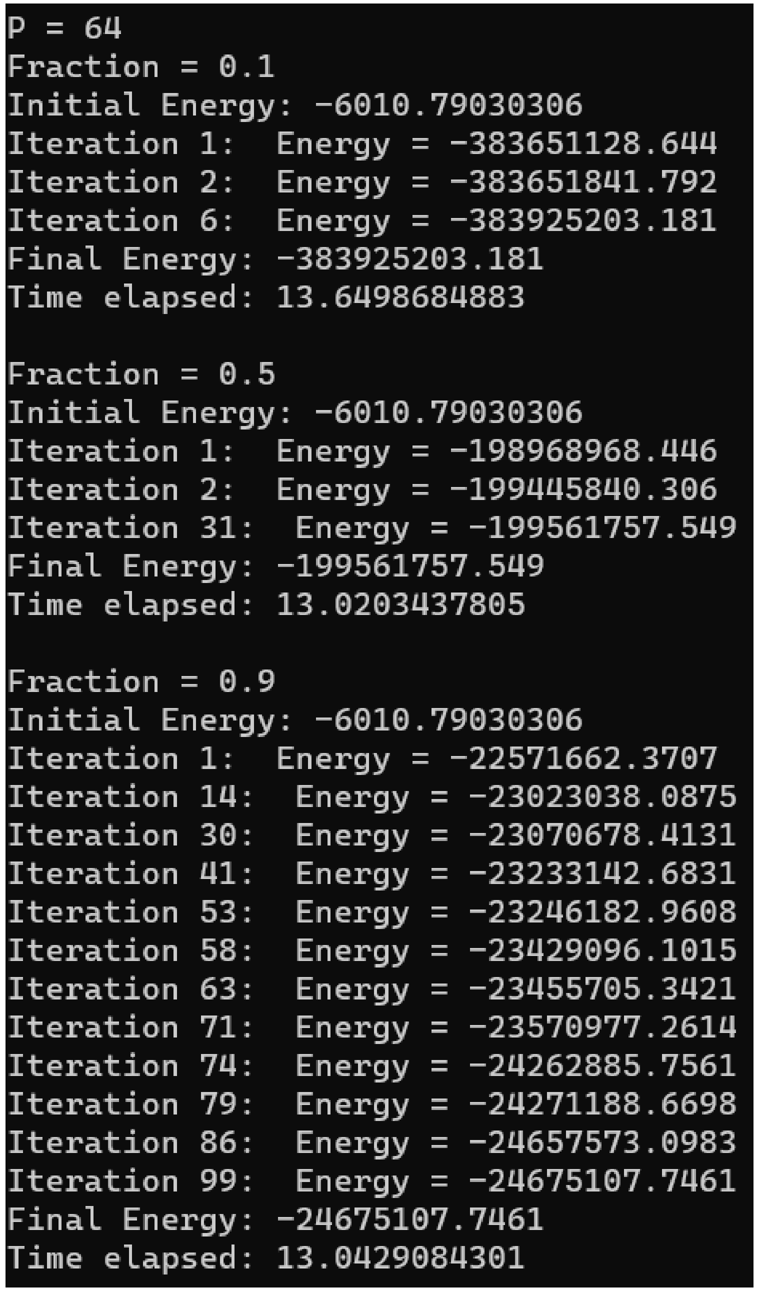
```
P = 16
Fraction = 0.1
Initial Energy: -6010.79030306
Iteration 1:  Energy = -388855023.323
Final Energy: -388855023.323
Time elapsed: 46.7822186947

Fraction = 0.5
Initial Energy: -6010.79030306
Iteration 1:  Energy = -171426352.371
Final Energy: -171426352.371
Time elapsed: 46.2184393406

Fraction = 0.9
Initial Energy: -6010.79030306
Iteration 1:  Energy = -8491270.63421
Final Energy: -8491270.63421
Time elapsed: 46.3413505554
```



```
P = 64
Fraction = 0.1
Initial Energy: -6010.79030306
Iteration 1:  Energy = -383651128.644
Iteration 2:  Energy = -383651841.792
Iteration 6:  Energy = -383925203.181
Final Energy: -383925203.181
Time elapsed: 13.6498684883

Fraction = 0.5
Initial Energy: -6010.79030306
Iteration 1:  Energy = -198968968.446
Iteration 2:  Energy = -199445840.306
Iteration 31:  Energy = -199561757.549
Final Energy: -199561757.549
Time elapsed: 13.0203437805

Fraction = 0.9
Initial Energy: -6010.79030306
Iteration 1:  Energy = -22571662.3707
Iteration 14:  Energy = -23023038.0875
Iteration 30:  Energy = -23070678.4131
Iteration 41:  Energy = -23233142.6831
Iteration 53:  Energy = -23246182.9608
Iteration 58:  Energy = -23429096.1015
Iteration 63:  Energy = -23455705.3421
Iteration 71:  Energy = -23570977.2614
Iteration 74:  Energy = -24262885.7561
Iteration 79:  Energy = -24271188.6698
Iteration 86:  Energy = -24657573.0983
Iteration 99:  Energy = -24675107.7461
Final Energy: -24675107.7461
Time elapsed: 13.0429084301
```

Discussion

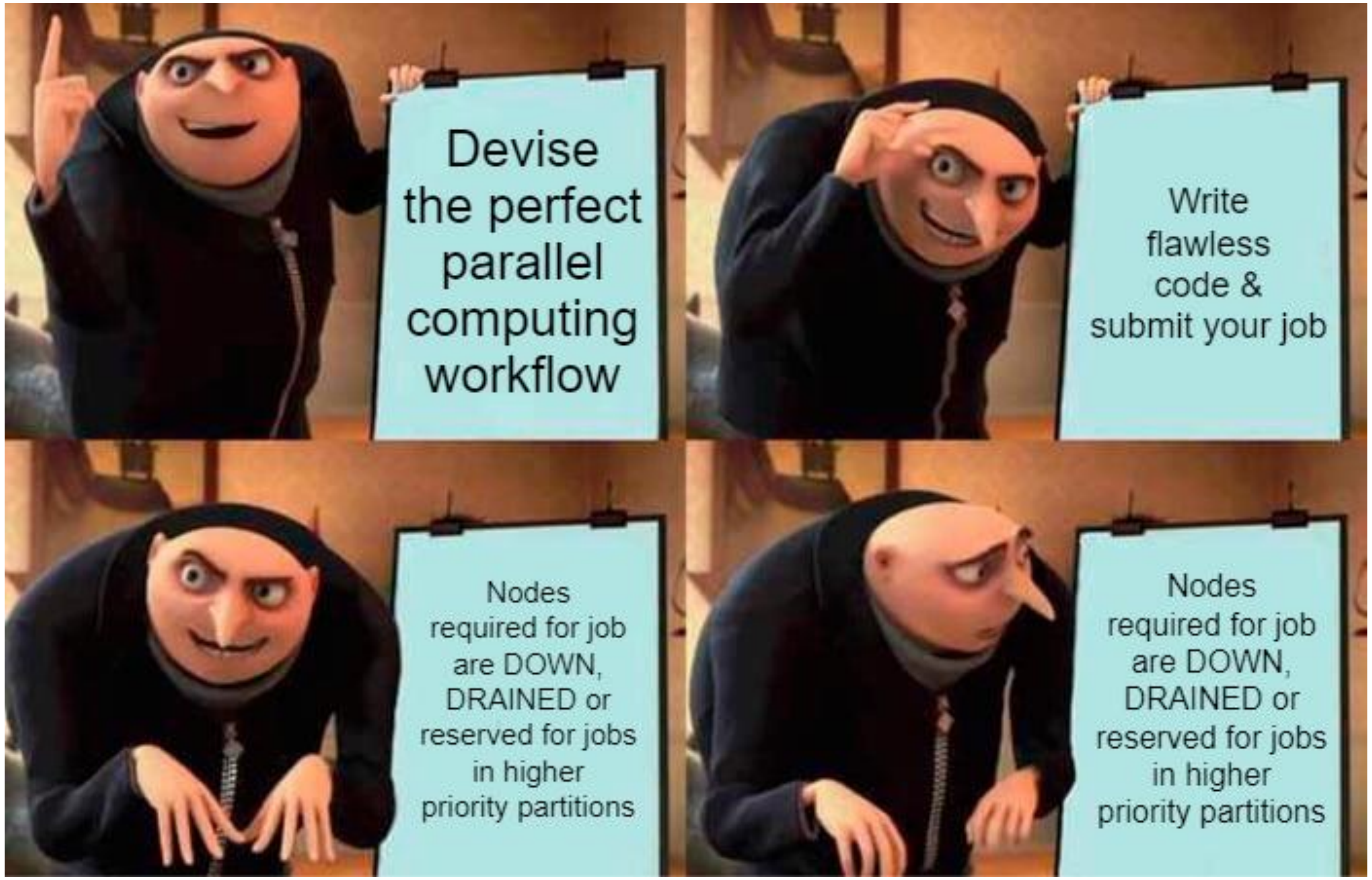
- Energy reduction
  - 1st step usually has a massive reduction
  - As fraction number increases reduction of energy is reduced
  - Among iterations reduction is small compared to 1st iteration
- Time elapsed
  - Within each P, time elapsed variance is very small
  - As P increases time elapsed decreases



```
P = 64
Fraction = 0.1
Initial Energy: -6010.79030306
Iteration 1:  Energy = -383651128.644
Iteration 2:  Energy = -383651841.792
Iteration 6:  Energy = -383925203.181
Final Energy: -383925203.181
Time elapsed: 13.6498684883

Fraction = 0.5
Initial Energy: -6010.79030306
Iteration 1:  Energy = -198968968.446
Iteration 2:  Energy = -199445840.306
Iteration 31:  Energy = -199561757.549
Final Energy: -199561757.549
Time elapsed: 13.0203437805

Fraction = 0.9
Initial Energy: -6010.79030306
Iteration 1:  Energy = -22571662.3707
Iteration 14:  Energy = -23023038.0875
Iteration 30:  Energy = -23070678.4131
Iteration 41:  Energy = -23233142.6831
Iteration 53:  Energy = -23246182.9608
Iteration 58:  Energy = -23429096.1015
Iteration 63:  Energy = -23455705.3421
Iteration 71:  Energy = -23570977.2614
Iteration 74:  Energy = -24262885.7561
Iteration 79:  Energy = -24271188.6698
Iteration 86:  Energy = -24657573.0983
Iteration 99:  Energy = -24675107.7461
Final Energy: -24675107.7461
Time elapsed: 13.0429084301
```

Future Direction

- Decomposition
  - This project uses particle decomposition
  - Spatial + particle decomposition
- Energy reduction
  - Method applied Monte Carlo
  - Gradient descent
  - Python’s scipy.optimize()


Devise the perfect parallel computing workflow
Write flawless code & submit your job
Nodes required for job are DOWN, DRAINED or reserved for jobs in higher priority partitions
Nodes required for job are DOWN, DRAINED or reserved for jobs in higher priority partitions