Final Project Report

PARALLELIZATION OF MONTE CARLO SIMULATION FOR ADSORPTION-DESORPTION AND SOLVING HEAT CONDUCTION PROBLEMS

Keerthiharan A
CH22B026

ABSTRACT

Kinetic Monte Carlo (KMC) simulations in combination with first-principles (1p)-based calculations are rapidly becoming the gold-standard computational framework for bridging the gap between the wide range of length scales and time scales over which heterogeneous catalysis unfolds. 1p-KMC simulations provide accurate insights into reactions over surfaces, a vital step toward the rational design of novel catalysts. In the first part of this project I have modelled the Adsorption-Desorption Kinetics governed by **Langmuir Adsorption** theorem on a 2D square lattice. We tend to find the equilibrium Surface coverage for the given rates on the adsorbent surface. Since it may require a large number of iterations to converge, for a relatively large lattice parallelization is inevitable to ensure faster execution. Also by executing on multiple threads we can validate our model and ensure its accuracy and precision.

In the second part we have considered simple **stochastic model based on a discrete random walk** to model transient heat conduction problems in a **homogeneous one-dimensional region** for all two different boundary conditions,

1. Zero Temperature at the walls and
2. Zero Heat flux at the walls.

By implementing the discrete random walk model under these boundary conditions, we aim to investigate how heat spreads through the material, how different boundary conditions influence the temperature distribution dynamics. I have used **OpenMP** to parallelize the algorithms and it showed a significance improvement in the execution time.

NOMENCLATURE

The symbols used in the report are specified below.

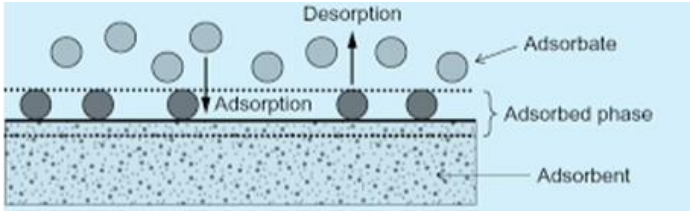
R_A : Rate of Adsorption
R_D : Rate of Desorption
α : Strength of Near Neighbor interactions
θ : Fraction of occupied sites on the adsorbent
$T(x, t)$: Temperature in the position x at time t,
a : Thermal diffusivity,
ε : Strength of the near neighbor bond.
k : Boltzmann Constant ($1.38 \times 10^{-23} m^2 kgs^{-2} K^{-1}$)
T : Temperature of the adsorbent surface
Φ : Deposition rate of surface of adsorbate

INTRODUCTION

Monte Carlo (MC) methods are broadly defined as computational techniques that make use of random numbers with the aim of generating samples from a given probability distribution and sometimes of estimating expectations of the functions under this distribution.

For the KMC Simulation we have used Langmuir Adsorption theorem which assumes,

- The adsorbent surface is flat
- All sites of adsorption are equivalent
- Each site can hold to utmost one atom and
- There are no interactions among adjacent atoms.



In case of Heat Conduction Problems, for transient conditions we have the equation,

$$\partial T(x, t)/\partial t = a\, \partial^2 T(x, t)/\partial x^2$$

This can be discretized using FTCS (Forward Time Central Space) and written as,

$$T(x, t + \Delta t) = \mu T(x - \Delta x, t) + (1 - 2\mu)T(x, t) + \mu T(x + \Delta x, t).$$

Where, μ is the dimensionless mesh ratio given by $a\frac{a\Delta t}{\Delta x^2}$

In probabilistic interpretation this equation is the **Smoluchowski equation** for a free Brownian particle, which is randomly moving one step in positive or negative direction to (x ± Δx), with probability p = μ, or remaining at the same position (x), with probability q = (1 − μ).

We will use one of the simplest types of discrete random walk, in which **p = 1/4** and **q = 1/2**. Space and time will be discretized using

Dimensionless time $t = s\Delta t a/L^2$,s = 0, 1,...

Dimensionless coordinate x = mΔx/L, m = 0, ±1, ±2,…

where L is a unit length, divided into N equal size bins.

The probability distribution for it can be obtained by a computer, or by flipping simultaneously two fair coins. If they differ, then the particle will stay in the same bin. If there are two heads, or two tails, the particle will move to the left, or to the right, respectively.

We have shown that the discrete random walk with properly chosen boundary conditions (Zero Temperature and Zero Heat flux B.C), can be used to model a broad range of heat conduction problems in 1-D regions. We will assume that the particle can be imagined as a point particle and the probability P(m, s) remains uniform within a bin.

KMC SIMULATION

We have generalized the model for adsorption desorption kinetics to the case where near-neighbor interactions are included in the analysis. For 2D square lattice, a given site can be empty or occupied by an atom 0, 1, 2, 3 or 4 neighbors.

- If a site is empty, an atom can adsorb to the site at $\mathbf{R_A = 1s^{-1}}$
- If a site is occupied, then atom at the site can desorb at $\mathbf{R_{Di} = R_D \alpha^i}$, where i = 0,1..4 denotes the number of neighbors and $\mathbf{R_D = 2s^{-1}}$. Here, $\boldsymbol{\alpha = e^{-\varepsilon/kBT}}$

We have considered a **10x10 lattice** and assume initially all the sites are empty. For the cases α = 1 and 0.5 we will plot the fraction of occupied sites (θ(t)) as a function of time until a steady state is reached. At any given instance of time, there are 6 possible events on the surface:

(1) Adsorption and

(2-6) Desorption from a 0, 1, 2, 3, 4 coordinated sites.

We will use periodic boundary conditions along both the coordinates.

EXECUTION FLOW:

1. Categorize and tabulate all the possible types of events and the atoms that can make each more. Total rate of anything happening on the surface is
   $\mathbf{R = \sum n_i r_i + \Phi}$
   Therefore, on average an event happens in dt = 1/R
2. Probability of a process in that time is $P_i = n_i r_i/R$. Hence, find the probabilities of all the events.
3. Choose which process will occur by weighting its relative probability and by choosing a random number between 0 and 1
4. Go to the list it events with chosen rate r_i. Chose a candidate from the list randomly.
5. Execute the chosen event e.g., a jump.
6. Update all the lists to reflect the change in surface configuration.
7. Update time by increment dt = 1/R
8. Repeat this process for specified time.
9. This is repeated N (=100) times and the result is averaged to get a more accurate and stabilized result.

PARALLELIZATION IDEA AND CONSTRAINTS:

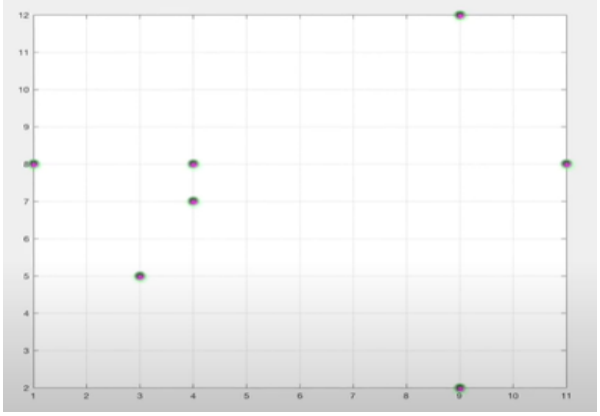
For each execution of each simulation the current step depends on the occupancy of lattice in the previous step, this creates data dependency and prohibits the parallelization of the inner process loop. Whereas the loop which runs over the lattice has potential scope for parallelization but since we have applied periodic boundary condition it is required to transfer the data from one end to another end in case of OpenMPI, but as each process updates the state of points, we must use barriers and do data transfers to update this in all the processors which worsens the execution time and computational complexity. This parallelization is highly necessary in case of high dimensional lattices hence future study on this is inevitable.

We have parallelized the outer loop which executes the simulation multiple number of times using OpenMP which shows significant performance improvement. We have used 3D array to store the results of all the threads and calculate the final average, this prevents the racing condition in updating the same value in case of a 2D array.

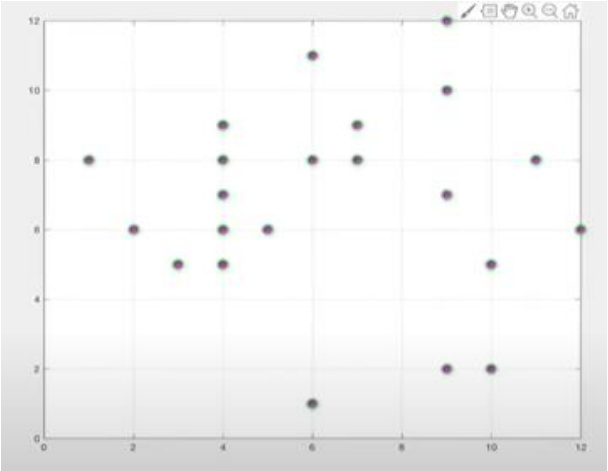
RESULTS:

To visualize how the surface looks, the following images of surface at various time steps are taken (for α = 1),

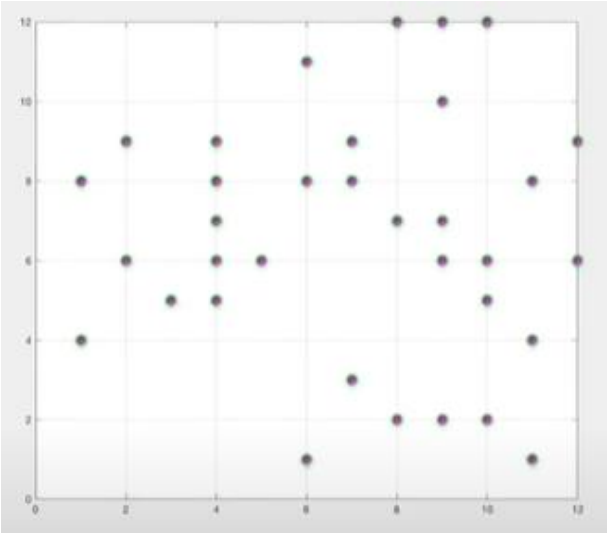
At t = 0.078,



At t = 0.58,

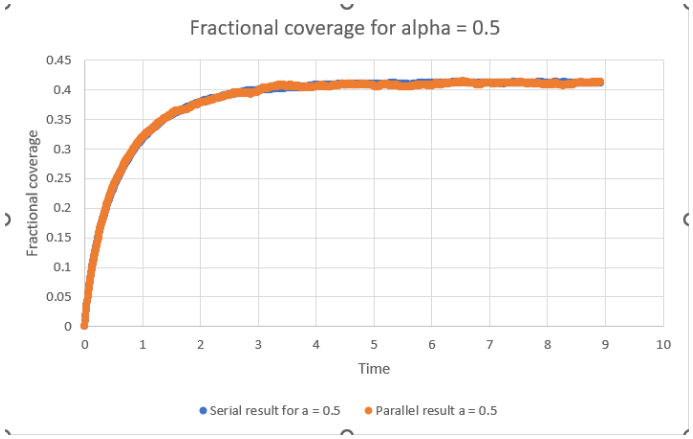


At t = 2.365,

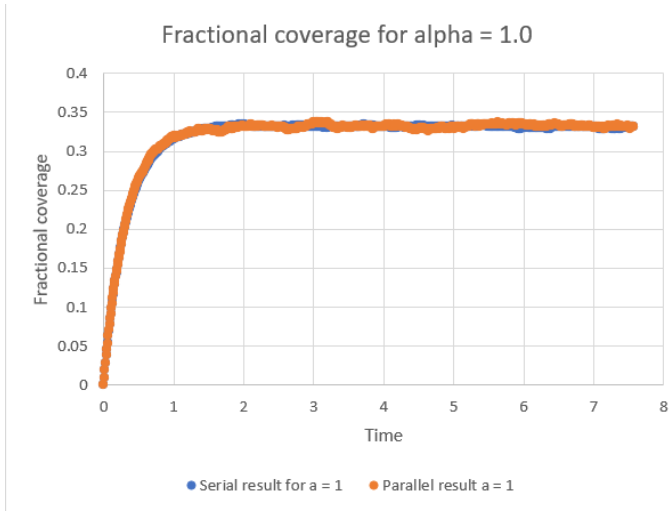


Fractional occupancy vs time:

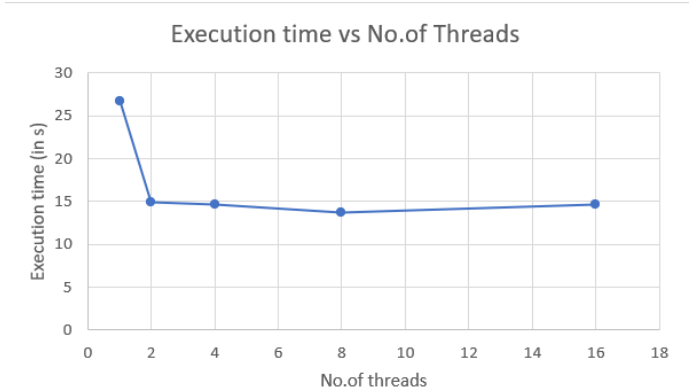
For α = 0.5



For α = 1.0

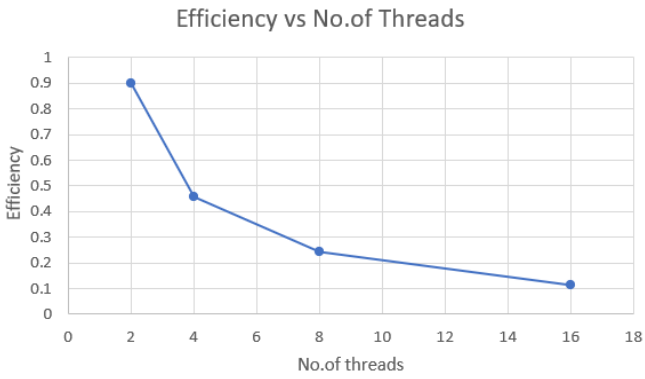


Execution time vs Number of threads:



Efficiency vs Number of threads:

Efficiency of execution is given by, Speedup / No.of threads
Where, Speedup = **Seq time / Parallel time**



No. of threads	Execution Time	Efficiency
1(Serial)	26.761 s	NIL
2	14.89 s	0.898623
4	14.663 s	0.456267
8	13.734 s	0.243565
16	14.658 s	0.114106

- Due to limited availability of threads, it creates hyperthreading when I use more than 2 threads, hence there isn't much performance improvement on using a greater number of threads, but when we use a system with better specifications, we can expect a good scaling.

HEAT CONDUCTION

1. Zero Temperature at Boundaries

Boundary condition is given by

$$T(0, t) = 0, T(1, t) = 0, \text{ for } t > 0$$

The boundaries are bouncing back the temperature contributions from heat sources with exactly the same, but negative probability amplitude. That is, the boundaries are reflecting the particle back, while changing its contribution sign. This type of boundary condition is also called anti-reflective.

ALGORITHM AND PARALLELIZING STRATEGY:

The temperature distribution can be simulated using the discrete random walk with the anti-reflective boundary conditions added at p = 1, and p = N. Given the number of bins per unit length (N), the number of time steps, the number of samples (B), and the initial position index (k), the first for loop initializes the probability distribution vector P to its initial values at time zero.

The second for loop is repeating the sampling process for a total of B number of samples. The first line of this loop contains the particle initial position and the embedded for loop calculates the position where the particle will be after s randomly chosen steps. The step is either -1, 0, or 1, with probabilities 0.25, or 0.5, or 0.25, respectively. Anti-reflective boundary conditions are in 6 lines 11 and 15. When the final position, given by the index p, is determined, the value of Pp is increased (tallied) by 1. At the end, vector P is multiplied by a factor N/B and returned as a result.

Pseudo code:

```
for i=1 to s + k do
    Pi ← 0
end for
for j=1 to B do
    p ← k
    c ← 1
    for n=1 to s do
        step ∈ {−1, 0, +1}
        p ← p + step
        if p < 1 then
            p ← 1
            c ← −c
        end if
        if p > N then
            p ← N
            c ← −c
        end if
    end for
    Pp ← Pp + c
end for
return Pi,s ← PiN/B
end
```

Temperature distributions calculated using Algorithm 1 for a unit instantaneous pulse in k = 13, for N = 50, B = 10^5,

s = 200, 400 and 800 are compared with analytical curves where $x_0 = 0.25$ and t = 0.02, 0.04 and 0.08, respectively.

2. Zero Heat Flux Boundary Conditions

The temperature distribution T(x,t|x0) in a finite region (0<x<1), initially at zero temperature, with an instantaneous unit heat source in x0 described by $\delta(x_0 - x)\delta(t)$, and zero heat flux at the boundaries:

$$\partial T(0, t)/\partial x = 0 \text{ and } \partial T(1, t)/\partial x = 0$$

This type of boundaries is called fully-reflective.

Pseudo code:

```
for i = 1 to s + k do
   Pi ← 0
end for
for j = 1 to B do
   p ← k
   for n = 1 to s do
      step ← random{-1, 0, 1}
      p ← p + step
      if p < 1 then
         p ← 1
      end if
      if p > N then
         p ← N
      end if
   end for

   Pp ← Pp + 1
end for
return Pi,s,k ← PiN/B
end
```

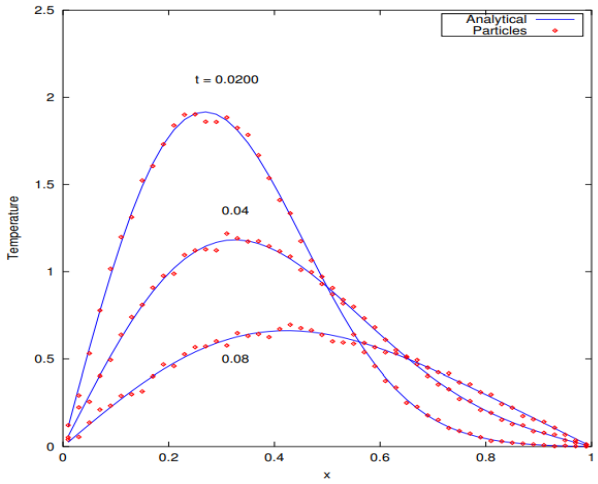
PARALLELIZATION STRATEGY:

Given the algorithm is pretty simple to execute in general B value might be very large, which requires parallelization to improve the execution time. I parallelized the main loop with OpenMP parallel for clause in both the cases.

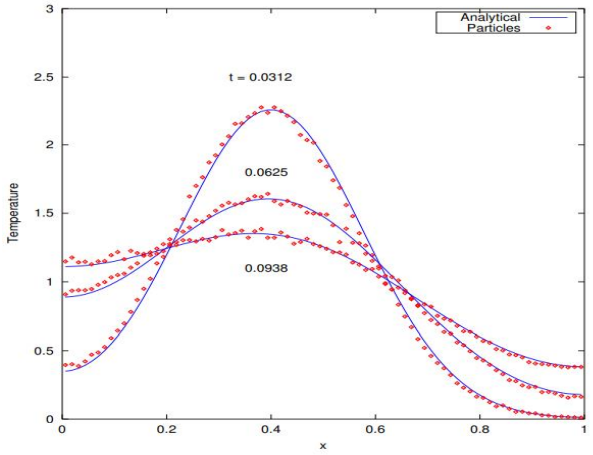
RESULTS:

Comparing the solutions obtained with the exact solution.

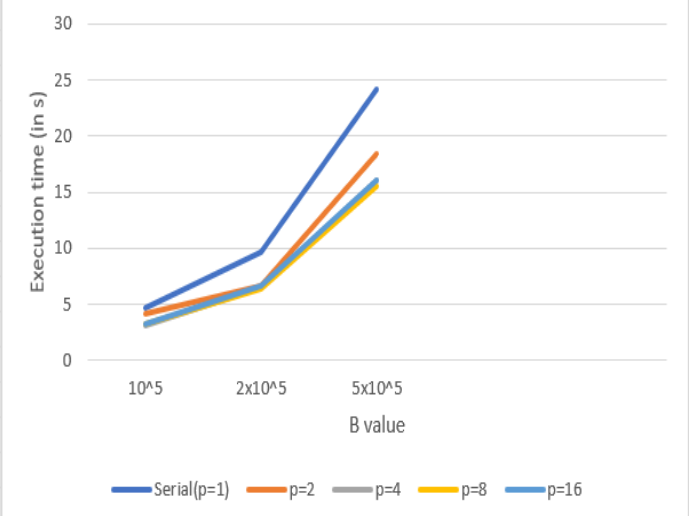
Case-1



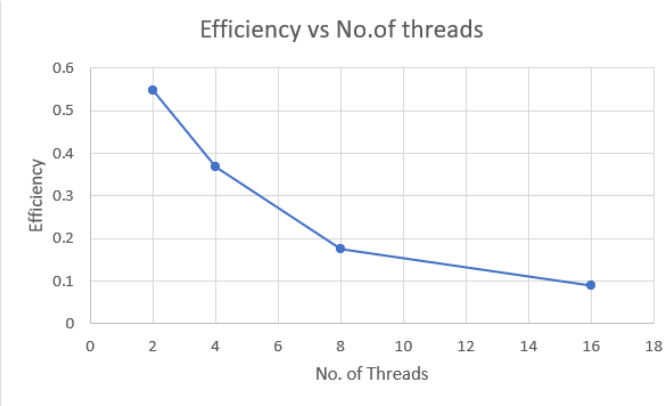
Case-2



Comparing the execution time vs number of threads for different sizes:



Efficiency vs Number of threads:



We got nearly same performance in both the cases in terms of efficiency and execution time.

The following table shows the execution time (in s) for various number of threads and size,

B value	Serial	p=2	p=4	p=8	p=16
1×10^5	4.632	4.23	3.148	3.285	3.256
2×10^5	9.652	6.62	6.485	6.429	6.632
5×10^5	24.122	18.385	16.079	15.497	16.107

CONCLUSION:

We saw how parallelization can drastically improve the performance, using OpenACC might give an even better performance, but due to lack of computing resources we could not do that, but the potential of parallelization here is enormous. The highly parallelizable nature of the problem gives scope for lot of improvements. By using more efficient data structures or with a better implementation we can utilize the OpenMPI to parallelize the lattice traversal in case of KMC. With this approach as the base, we can build on more advanced cases with lesser assumptions to model a more realistic system, which can be a lot more scalable.

ACKNOWLEDGMENTS

Thanks to Dr. Kameswararao Anupindi the course instructor for his support throughout the course and the project.

REFERENCES

[1] Molecular Simulations in Mechanics and Physics by Dibakar Datta and Vivek Shenoy.
[2] A Practical Guide to Surface Kinetic Monte Carlo Simulations, Mie Andersen and Chiara Panosetti
[3] Introduction to the Kinetic Monte Carlo Method, Arthur F. Voter
[4] Efficient Monte Carlo simulation of heat conduction problems for integrated multi-physics applications, Valeryi Sizyuk and Ahmed Hassanein.
[5] A. Haji-Sheikh, E. M. Sparrow, The solution of heat conduction problems by probability methods.