Final Project: Diffusion Simulator
Programming for Scientists

1. Introduction

1.1. Overviews

Diffusion is sometimes referred to the movement of molecules or atoms down a concentration gradient.

In modern textbooks, there is a division in the field of diffusion, namely the continuum diffusion (solving pde equations) and the atomic diffusion (particles). Typically speaking, it would be easier to start with classic (or continuum) diffusion. However, I am going to proceed in the opposite way, and start with discrete diffusion.

1.2. Project-specified backgrounds

(i) Random Walk (RW)

In a RW system, the particles can move $+\alpha$ or $-\alpha$ units at random for every time step. After n steps, the distribution of positions will be:

$$P(x) = \frac{1}{\sqrt{2\pi < x^2 >}} e^{\frac{-(x-<x>)^2}{2<x^2>}}$$

where $< x >$ is mean displacement, and $< x^2 >$ is mean square displacement. For the above RW system, $< x >= 0, < x^2 >= \alpha^2 n$. Note that n is the number of steps, not the actual time.

Final Project: Diffusion Simulator
Programming for Scientists

The Gaussian distribution shows that after n steps, the most probable position is $x = 0$, but also $|x| \neq 0$ is much more probable. The expected distance from the origin is $\sqrt{< x^2 >}$.

The Gaussian distribution has the local maximum entropy for a given $< x >$ and $< x^2 >$. This means that the RW distribution is a local equilibrium distribution (related to the 2nd law of thermodynamics).

(ii)Random Walk Mechanism

When it comes to real-world case, the atoms or the random walkers will interact with other atoms/walkers. It's quite difficult for random-walkers or atoms to perform direct exchange because it requires pushing the other atoms out of their original locations and distorting the crystal lattice. This process cost a great amount of energy (also called the activation energy for direct exchange).

Here we will adopt the vacancy mechanism which has a relatively low activation energy. The vacancy can "exchange" with its neighbor atoms or walkers with equal probability. This kind of substitutional diffusion is a RW only when the atom/walkers are next to vacancies and move. Every time an atom moves, so does a vacancy (we don't take interstitial diffusion into consideration even though it's quite common).

(iii)Diffusion Pattern

In 1855 Adolf Fick published a paper and described an empirical equation which is now called Fick's first law:

$$J = -D\nabla C$$

Fick's first law takes the form of generalized equations:

$$flow \propto gradient$$

$$velocity \propto driving\ force$$

Although Fick's law is empirical, it can still be proved by RW if we associate the number of random-walkers to the concentrations. Also, some cases will be studied in the following part. For most of the case, we assume that Ag diffuse in Au.

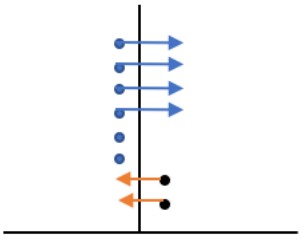

Fig 1.2.1 random-walkers

Case "surface": steady state diffusion through a membrane.

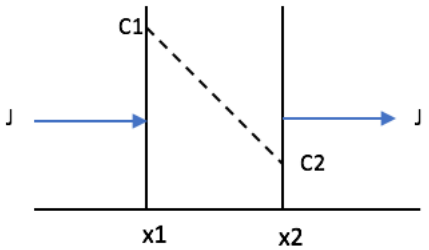

Fig 1.2.2 surface diffusion

C1, C2 held constant at the surfaces (see fig 2.2.2)

$$\Delta C = C2 - C1$$

$$\Delta x = x2 - x1$$

After a long time, the concentration profile reached steady state, we have:

$$J = -D\frac{\mathrm{d}C}{dx} = const \rightarrow \frac{\mathrm{d}C}{dx} = -\frac{J}{D}$$

So, when the steady state is reached, there will be a linear concentration profile

Case "thin": a thin film in an infinite slab (see fig.2.2.3)

The width of the thin film can be ignored. So, this is just a stack of 1D random walkers. Note that those random walkers can also RW in y axis as well, but those steps will on average cancel among all particles.

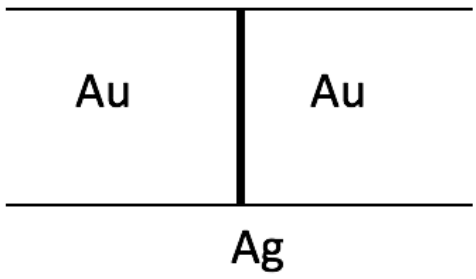

Fig 1.2.3 thin film diffusion

The concentration profile is straightforward: since each random walker contribute to a Gaussian distribution and they are indistinguishable, we can conclude that:

Final Project: Diffusion Simulator
Programming for Scientists

$$\mathrm{P(x)} = \mathrm{c(x)} = \frac{1 * a}{\sqrt{2\pi Dt}} e^{\frac{-x^2}{4Dt}}$$

Where a is a constant, which represent the total mass which related to the number of random walkers.

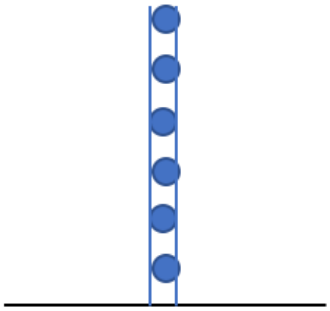

Fig 1.2.4 stack of random walkers

Case "Thick": Thick film is just a thin film whose width cannot be ignored (see fig 2.2.5).

Fig 1.2.5 thick film

Thick film is just the superposition of multiple thin films (or the Gaussian distribution). The concentration profile will be:

$$\mathrm{c(x)} = \frac{c_0}{2}\left[\mathrm{erf}\left(\frac{x+l}{\sqrt{4Dt}}\right) - \mathrm{erf}\left(\frac{x-l}{\sqrt{4Dt}}\right)\right]$$

where c_0 is the original concentration, the width of the thick film will be $2l$ and erf is error function.

Also, the concentration profile will evolve in Gaussian way.

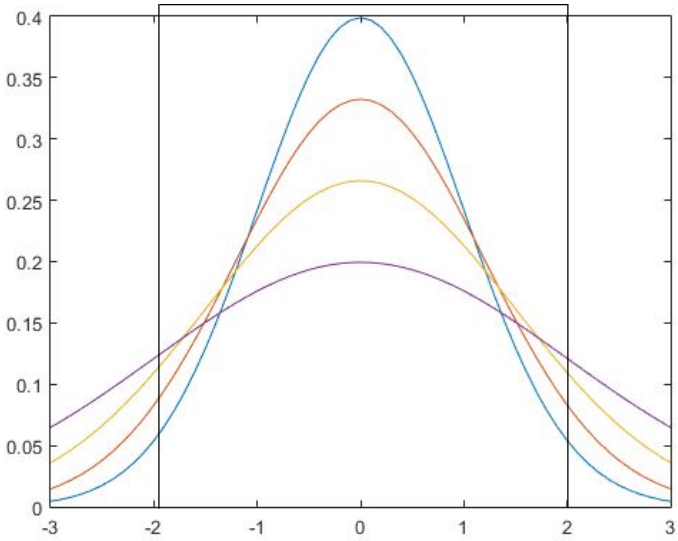

Fig 1.2.6 how thick film evolves

Case "Slab": a semi-infinite slab is just the superposition of multiple thin films in one side.

The concentration profile will be:

$$c(x) = \frac{c_0}{2}[1 - \mathrm{erf}\left(\frac{x}{\sqrt{4Dt}}\right)]$$

2. Simulations and Results

This simulator can perform two kinds of simulation. The first part will be numerical simulation, or FTCS (forward time, central space) to be exact. The second part will simulate the diffusion based on different mode.

Final Project: Diffusion Simulator
Programming for Scientists

2.1 Numerical Methods

For running the numerical part, one should use the following command line:

simulator.go ftcs numNodes numSteps c

where

- ftcs is the simulation mode, note that “FTCS” or “Ftcs” would be fine.
- numNodes is the nodes in x direction, should be an integer >0.
- numSteps is the steps to run, should be an integer >0.
- c is the boundary concentration, usually be 0 or 0.5.
- Other settings: suppose that by ion implantation, you could create a parabolic concentration profile through a thin film, the initial concentration profile is $c(x,0) = -0.5 * x^2 + x + 0.5$ where $0 \leq x \leq 2$ and the diffusion coefficient is $D = 0.05$

The FTCS method gives out that:

$$c_i^{m+1} = c_i^m + D\frac{\Delta t}{(\Delta x)^2}[c_{i+1}^m - 2c_i^m + c_{i-1}^m]$$

where m is time. So if we know $c(x,0)$ we can evaluate $c(x,t>0)$ for all time t. Also in our calculations, when $D\frac{\Delta t}{(\Delta x)^2} \geq 0.5$ then the FTCS method is called “unstable”, and the instability will accumulate.

When the steady state is obtained, the compositions will keep constant with time.

Sample Problem: Assume that you can embed your film in an atmosphere that maintain surface concentration (c=0.5); or presume that you vacuum anneal a similar film (c=0)

Input: the checking points, processing time and the concentration

Output: the evolvement of concentration distribution

Key steps to solve the problem:

1. Provide the command line arguments: numNodes<-n, numSteps<-t and boundary<-c
2. If $D\frac{\Delta t}{(\Delta x)^2} < 0.5$, then

   compositions= [] float64 //index=x axis; value=concentration

   InitializeFTCS(c, numNodes) //with boundary and original compositions.
3. For step=1 to numSteps do

   EvolevFTCS(compositions,n,t,c) //need to keep boundary

   OutputFTCS(compositions)

   CheckSteady(compositions) //check if steady state

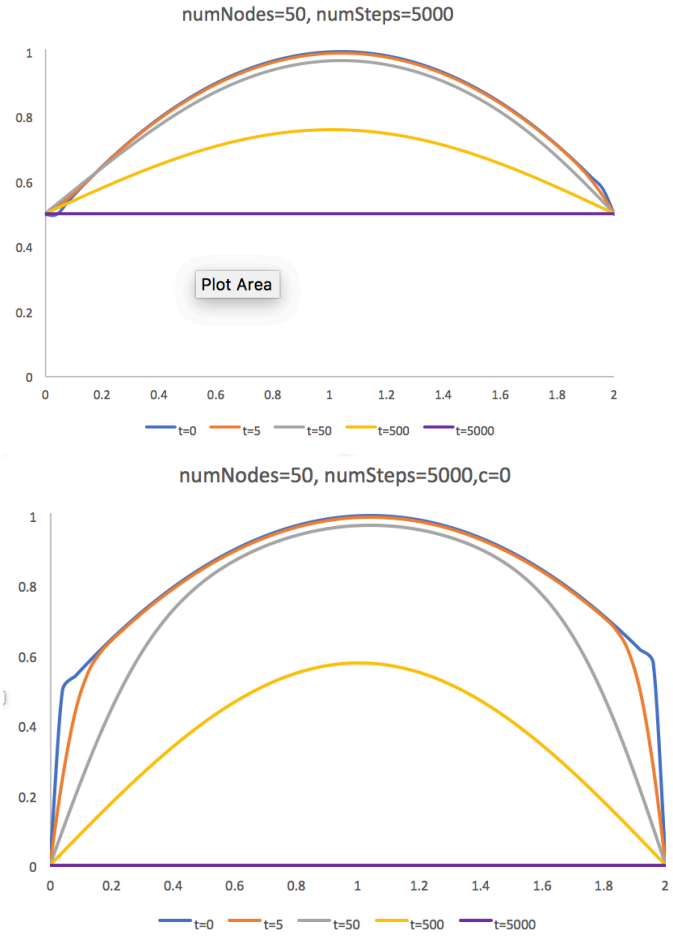

Fig 2.1.1 FTCS simulation

Conclusion: after a long time, the concentration profile will finally remain constant versus positions in both cases, which means the diffusion is almost done. The plots agreed with expected results.

2.2 Different diffusion modes(pattern)

This part aims to study the behavior of the random walkers from two perspectives: the change in position and concentration.

Final Project: Diffusion Simulator
Programming for Scientists

For running the simulation part, one should use the following command line:

simulator.go mode numParticles numSteps randomStep…

where

• numParticles is the number of random walkers, should be an integer >0.

• numSteps is the steps to run, should be an integer >0.

• randomStep is a series of steps that a random walker can move, like -1 and +1.

Sample Problem: Figure out the basic statistical behaviors of 1D random walker in free space.

Input: numParticles, numSteps, randomSteps

Output: The positions information versus numSteps

In this problem, the output will be a slice of discrete points. Since those data are lineally correlated, then a method called **linear regression analysis** is applied to study and estimate the results (or the slope) and compare them to theoretical values. That's what AnalyzeData() method will do.

Final Project: Diffusion Simulator
Programming for Scientists

Key steps:

1. numParticles<-n; numSteps<-t; randomSteps<-[]int{step1…}
   meanSquarePosition=[]float64
   finalPosition=[]float64
2. for i=1 to n do //for each particle
   for j=1 to t do //for each time steps
   step=ChooseStep(randomSteps)
   position=Move(position,step)
   meanSquarePosition[j]= meanSquarePosition[j]+position^2
   finalPosition[position]+=1 //occur one time
   if minPosition>=position then
   minPosition=position
   if maxPosition<=position then
   maxPosition=position
3. for k=1 to len(meanSquarePosition) do //to get the mean value
   meanSquarePosition[k]/=numParticles
   finalPosition[k]/=numParticles
4. AnalyzeData (meanSquarePosition, finalPosition)

	Linear regression analysis	Expected value
Slope	0.9942	1
Mean square displacement	98.47983	$\alpha^2 n = 100$

Final Project: Diffusion Simulator
Programming for Scientists

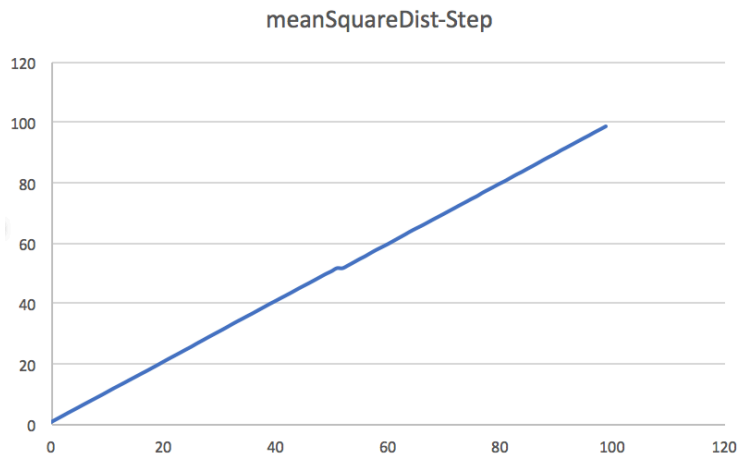

Figure 2.2.1 mean Square Displacement-time Step, n=10000

Conclusion: the plot of mean Square Distance versus Step agrees well with the theoretical equation: $< x^2 >= \alpha^2 n = n$

Optional problems to do: In fact, we can study many other random-walk-related problems with different inputs as followed:

•**Statistical accuracy problem**: give n>>t, we can calculate both the n-particles-t-steps and t-particles-n-steps systems to see if n and t are interchangeable, and find out which factor has greater influence on the simulation.

•**Step size and frequency problem:** in the sample problem, I take the step size to be 1. One can choose like $\alpha = \pm 2$ and $\alpha = 0$ as well as their combinations to figure out the behavior of the random walkers.

•**Predict Field Effect**: there could be some external (like magnetic and electrical) field effects that the random walkers will move in an

unbalanced way. Say the particles can move one step forward and two steps backward, $\alpha = [-2,0,1,1]$ or they can move mostly toward on direction $\alpha = [-1,1,1,1]$

In addition to the position changes, it's apparent and prominent that for a system, or the real-world materials, with the movement of random walkers, the composition profile inside the materials will change simultaneously.

Sample Problem: Say we want to produce a kind of materials by diffusion, like Al-a%Cu, where a is the amount of Cu. How can we design a manufacturing process?

Input: The processing mode, the concentration of Cu, processing time, and steps

Output: the evolvement of concentration distribution

For this problem, the critical point is to simulate the movements of multiple random-walkers at the same time. I separate the space into two parts and use goroutine to let the particles in both parts move concurrently.

In conclusion, the random diffusion pattern will be:

Final Project: Diffusion Simulator
Programming for Scientists

(1)where to move: depends on ChooseDirection() method

(2)when to move: depends on goroutine or the concurrency

(3)How to move: depends on RandomStep() method

Key steps:

1. mode<-mode, numParticles<-n, numSteps<-t,

randomSteps=[]int{step1,step2…}

2. board=InitializedBoard(maxDist)

//initialize the walkers' configurations

board.InitializeWalkers(mode)

//for each time step, all particles perform RW

for i=1 to t do

board.Diffuse(yStart1,yEnd1,mode)

// goroutine to simulate the simultaneous random movement in different zones

go board.Diffuse(yStart2,yEnd2,mode)

//then translate the number of walkers into concentration, assume that the original part is 1

3. board.ParseConcentration(n)

Simulating with similar instruction: **thin 1000 1000 -1 1**

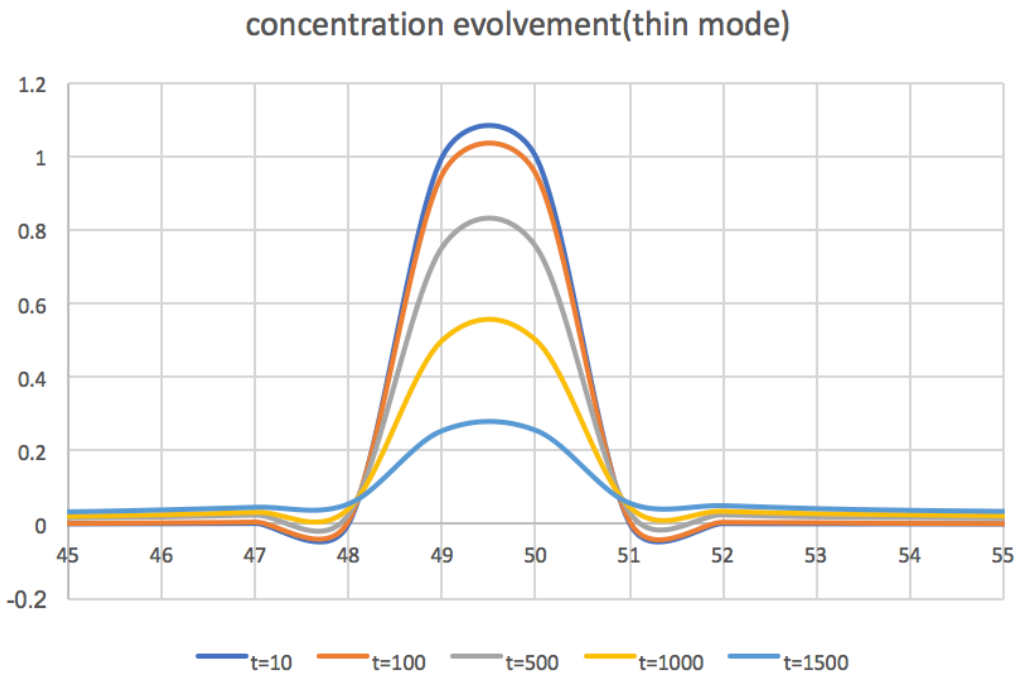

Fig 2.2.2 evolvement of thin mode

It's expected that the concentration profile of the thin film will evolve in Gaussian way (see introduction) and the above results agree with that.

Next improvements to do:

1. One can separate the diffusion board into four or more parts and activate them concurrently to get a better simulation of "randomness"
2. One can modify the board.InitializeWalkers()method to accommodate more kinds of atoms to diffuse.

Final Project: Diffusion Simulator
Programming for Scientists

3. Summary

This project aims at simulating some of the typical problems in diffusion based on random walk theory. The creative point is using goroutine to simulate random time.

After testing with some base cases whose results agree well with theoretical explanations, we can now conclude that:

1. The classic (or continuum) diffusion theory can be proved by or derived from discrete diffusion.
2. This simulator can be used to explore systems that we don't yet have analytical results for, or to predict some potential behaviors.