Welcome to
CHE 384T: Computational Methods in Materials Science

Random Walk Diffusion

LeSar Ch. 2, App. B7, C5, I2-I3

The University of Texas at Austin
McKetta Department
of Chemical Engineering
Cockrell School of Engineering

Announcements

HW 1 report due 09/12, 5pm to Canvas

Programming day 2: 09/06

Programming Days (approximately every other Friday):

L3	F Aug 30	Installation/set up; Jupyter, Modules and packages, environments What is object oriented programming? Why Python? global v local variables, manipulating lists and arrays, operators, (formatting strings), sets, tuples, lists, dictionaries, dataframes
L5	F Sep 6	conditions, loops, functions, classes and objects
L12	F Sep 20	opening a github account, testbeds, measuring speed and optimizing code, C libraries, documentation/sphinx, PEP8
L18	F Oct 4	ASE calculators
L24	F Oct 18	Python extras: list comprehension, exception handling decorators, lambda functions, regular expressions Peer sharing of Python tricks
6	F Nov 1	DFT tutorial: convergence, scf, relaxation, band structure advanced: phonon calculation, magnetic materials, surface properties

Approximate Schedule and Reading list for CHE384T

L1	Intro to the Course	Ch. 1, Appendix A
L2, L5	Random Walk Diffusion	Ch. 2, Appendix B7, C5, I2-I3
L7, L8	Intro to crystal structure, defect in materials	Appendix B1-B5
L10	Simulating finite systems	Ch. 3
L11, L13 L14	Interatomic potentials	Ch. 5
L16-L22	Molecular dynamics	Ch. 6, Appendix I4 Appendix G
L23, L25	Monte Carlo	Ch. 7, Appendix C4, D1-D4
L25-L32	Electronic structure and DFT	Ch. 4, Appendix F, Supplemental reading
L34	Materials informatics	
L35	Kinetic Monte Carlo	Ch. 9
L37	Monte Carlo as mesoscale Cellular automata	Ch. 11
L38	Quantum computing	

Lecture Outline

What is diffusion
Examples of diffusion in materials science

Connection with continuum description

Random Walk model for Diffusion

Coding considerations:
- Random number generators
- Binning probability distributions

Fick's First and Second Law

Fick's first law:

$$J = -D \bar{\nabla} c_i$$

diffusivity

Fick's second law:

$$\frac{\partial c_i}{\partial t} = +D_i \bar{\nabla}^2 c_i$$

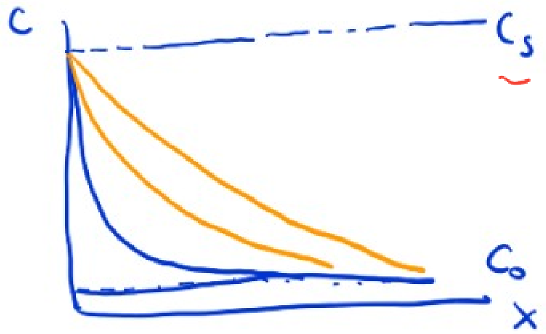
See also Appendix B7

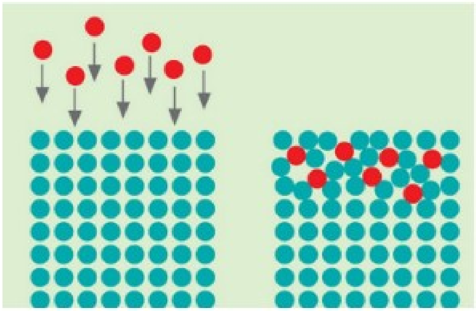
Example: Silicon wafer processing

Intentional incorporation of impurities (e.g., boron, phosphorous)

Step 1: Steady-state gas diffusion or ion implantation

$$\frac{c(x,t)-C_0}{C_s-C_0} = 1-\mathrm{erf}\left(\frac{x}{\sqrt{4Dt}}\right)$$

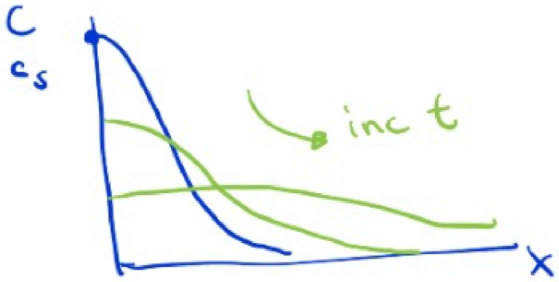




Step 2: Drive-in process (higher temperature)

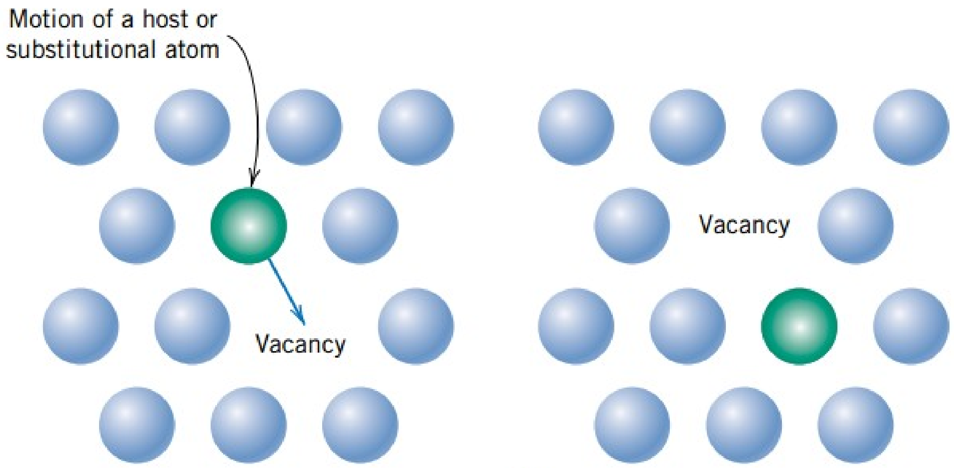
$$c(x,t) = \frac{2C_s}{\pi}\sqrt{\frac{D_p t_p}{Dt}}\exp\left(-\frac{x^2}{4Dt}\right)$$

depend on step 1



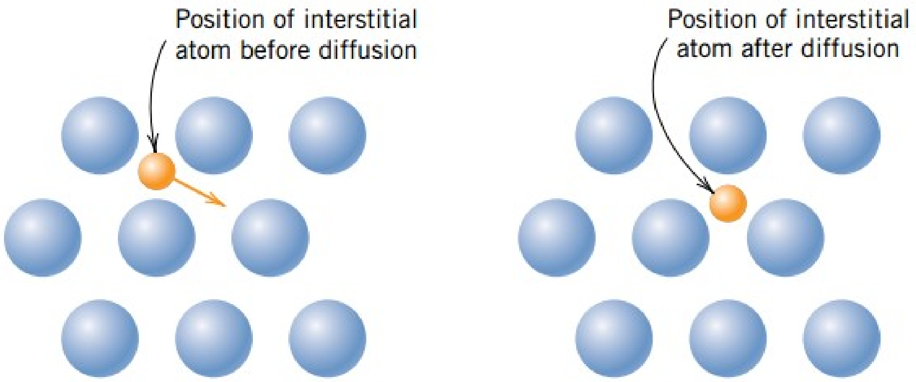
Types of Diffusion

Vacancy diffusion



(a)

Interstitial diffusion



(b)

Factors that affect diffusion: Diffusion coefficient

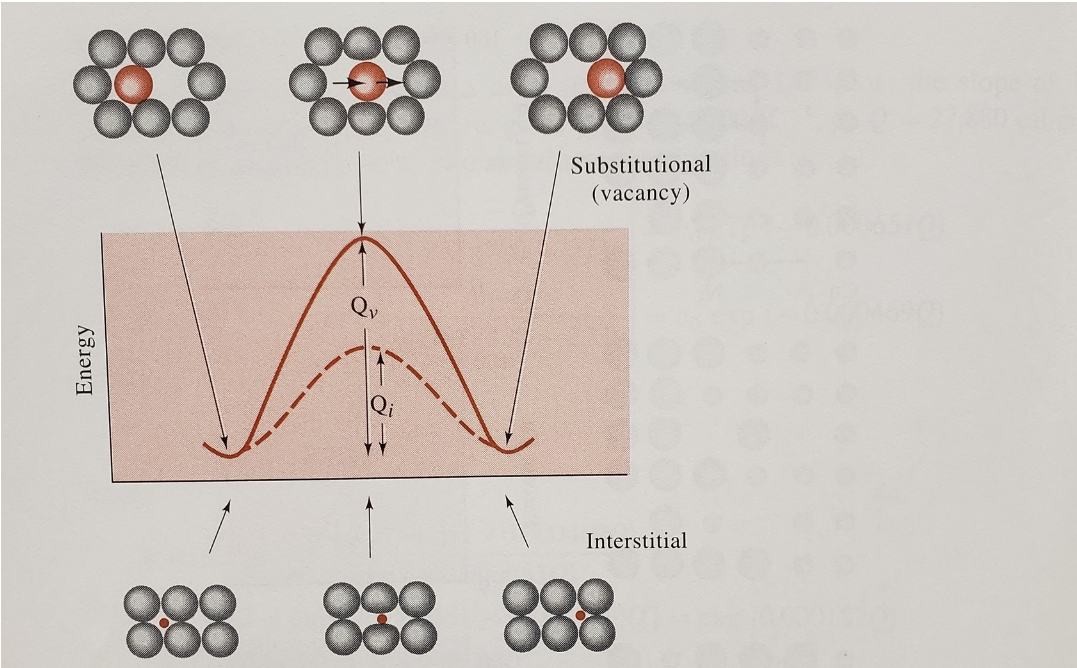


FIGURE 5-4 A high energy is required to squeeze atoms past one another during diffusion. This energy is the activation energy Q. Generally more energy is required for a substitutional atom than for an interstitial atom.

$$D = D_0 \exp\left(-\frac{Q}{k_B T}\right)$$

~ activation

$T \to \infty$

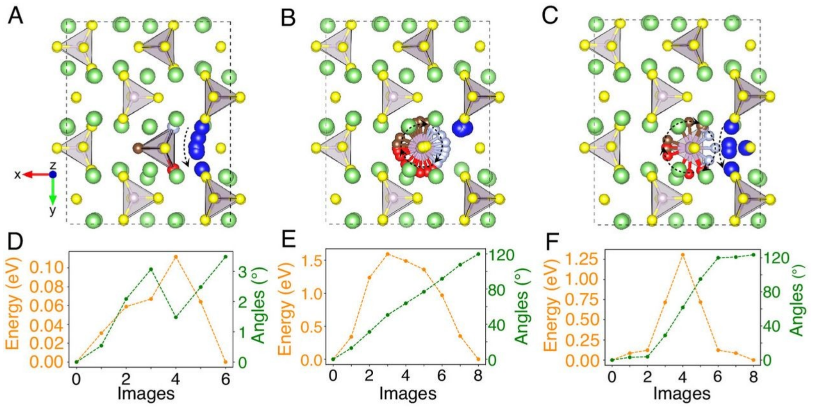
materials dep factor

↳ crystal structure

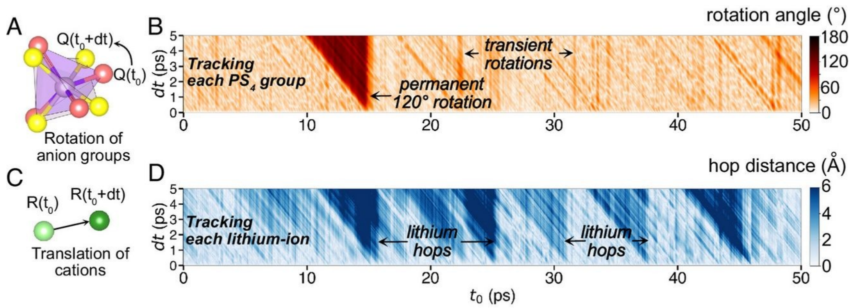
↳ chemical species identity

Donald R. Askeland. The Science and Engineering of Materials 3rd ed.

Collective motion of lithium with crystal lattice

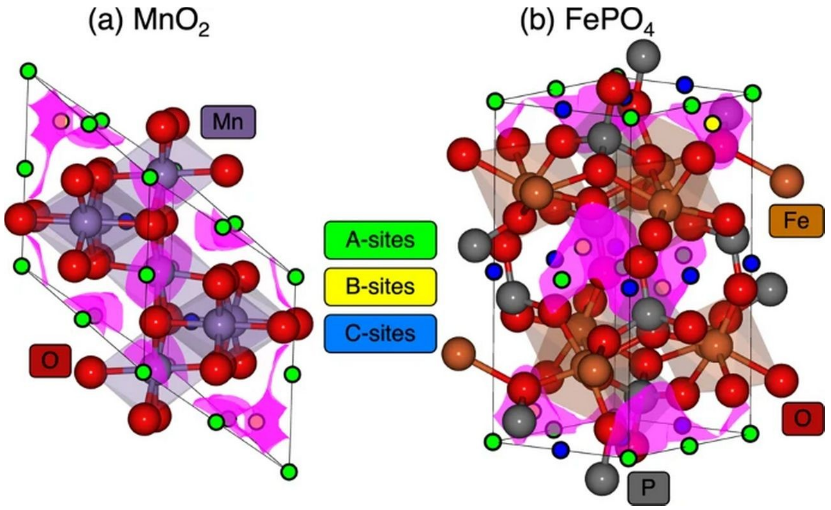


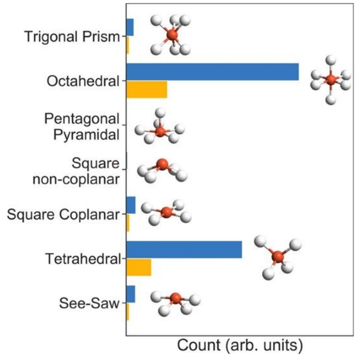
β-Li_3PS_4



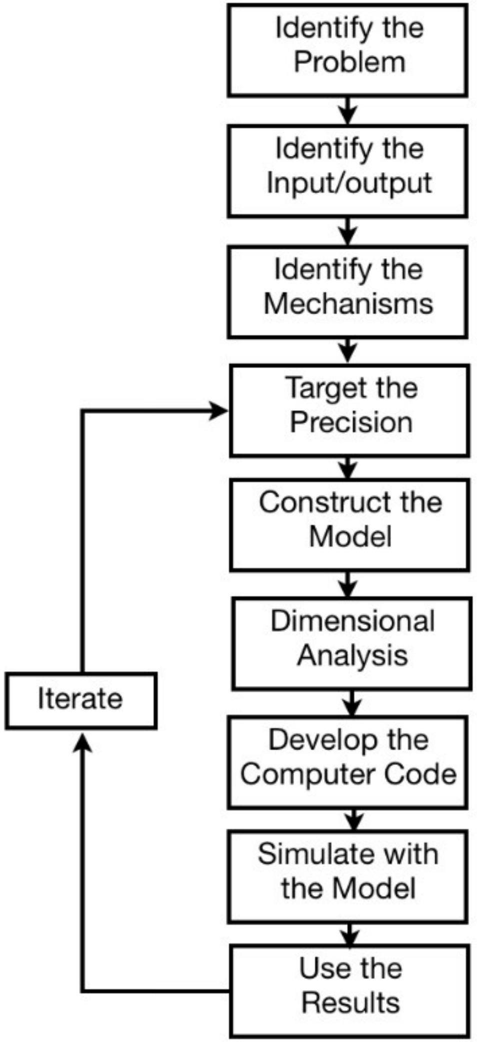
PNAS. 121 (18) e2316493121 (2024).

Intercalation of lithium in novel cathode materials





npj Comput Mater 6, 161 (2020).



Developing a model

“Science and Statistics.” George E.P. Box (1976)

“All models are wrong, but some are useful.”

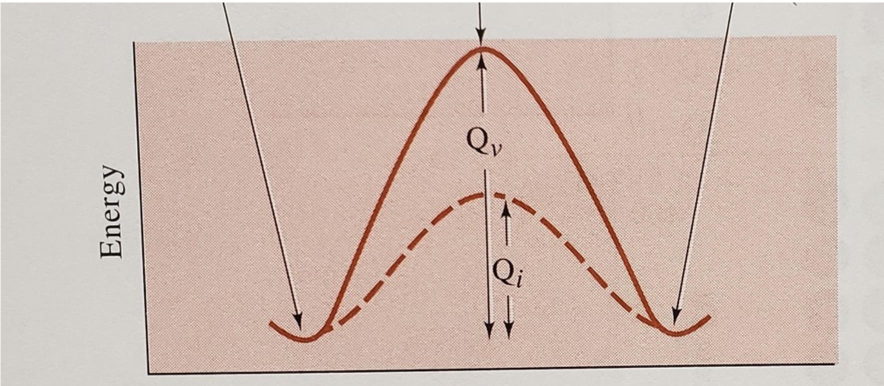
2.3 Parsimony

Since all models are wrong the scientist cannot obtain a “correct” one by excessive elaboration. On the contrary following William of Occam he should seek an economical description of natural phenomena. Just as the ability to devise simple but evocative models is the signature of the great scientist so overelaboration and overparameterization is often the mark of mediocrity.

2.4 Worrying Selectively

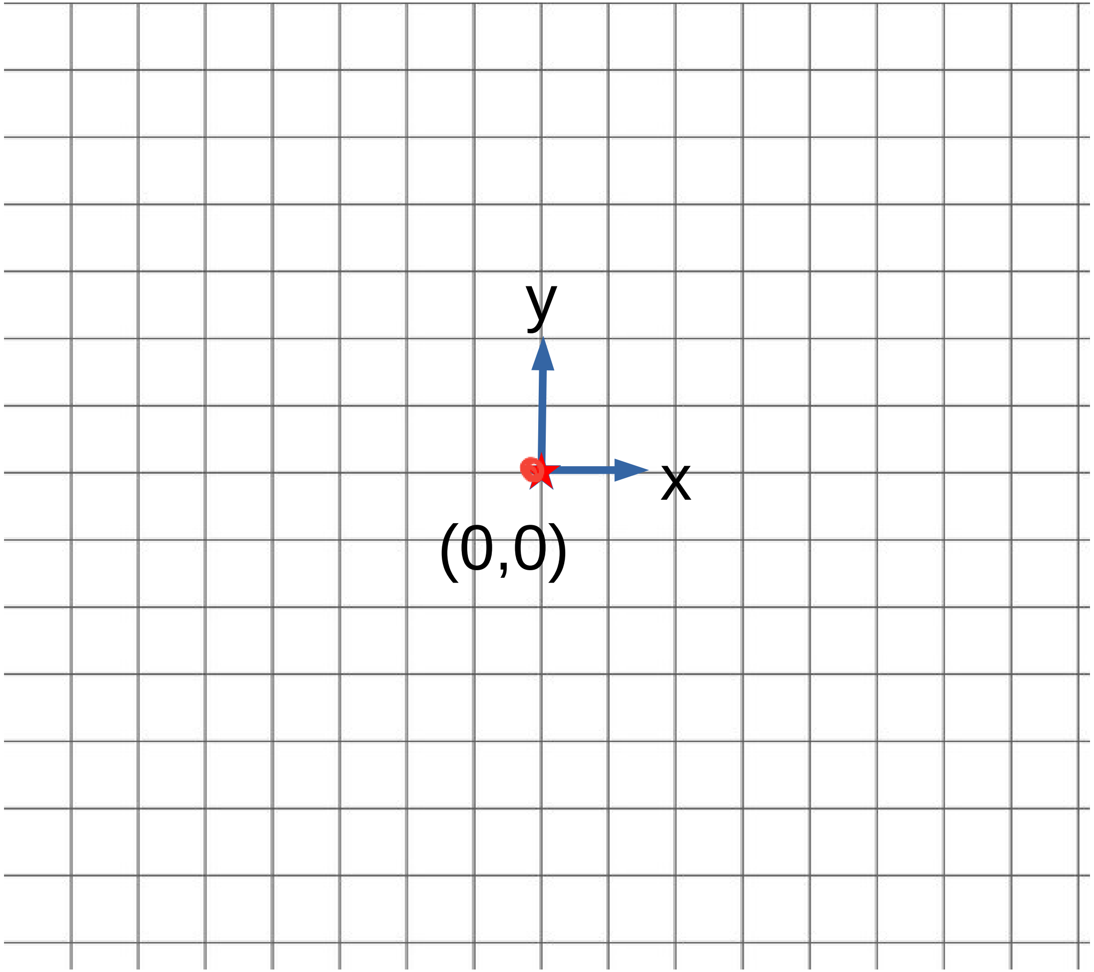
Since all models are wrong the scientist must be alert to what is importantly wrong. It is inappropriate to be concerned about mice when there are tigers abroad.

Simplifications of the Random Walk Model



- Reduced dimensionality (to 2D or 1D)
- Simplification of the crystal structure
- Model the hops the atoms take as random (for a particular atom)

Random walk diffusion: an atomic model for diffusion

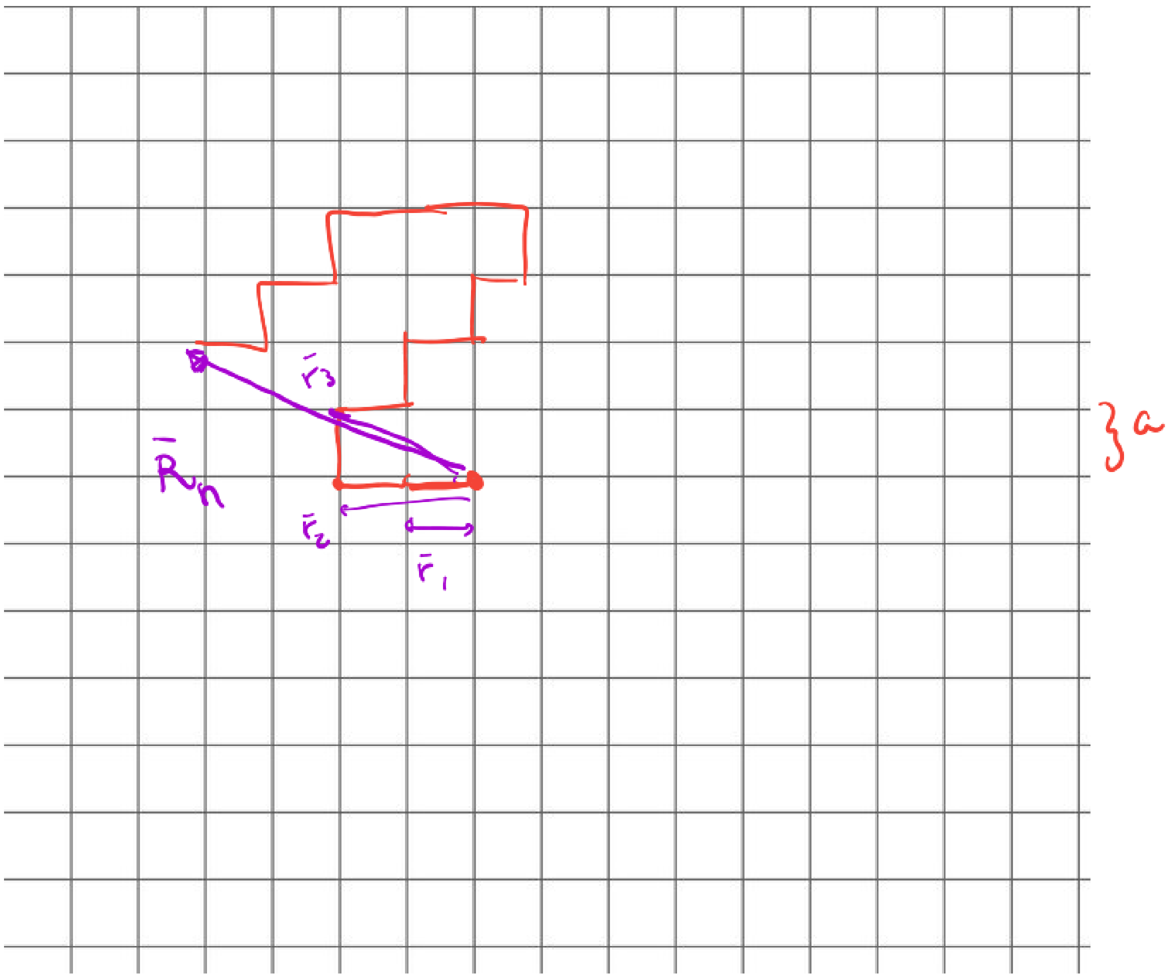


Rules for the random walker:

- divide time into *nt* discrete steps spaced by Δ*t* time, where *nt* is an integer and Δ*t* is a number
- can only move 1 space at each time step
- equal and random probability of moving up, down, left, right

$t_{tot} = n_{tot} \Delta t$

Random walk diffusion: an atomic model for diffusion



random jump

$$\Delta_i = \begin{cases} (0,1)\,a & 1 \\ (0,+1)\,a & 2 \\ (-1,0)\,a & 3 \\ (+1,0)\,a & 4 \end{cases}$$

$$\bar{R}_n = \bar{r}_0 + \sum_{i=1}^{n} \Delta_i$$

$\equiv$

$(0,0)$

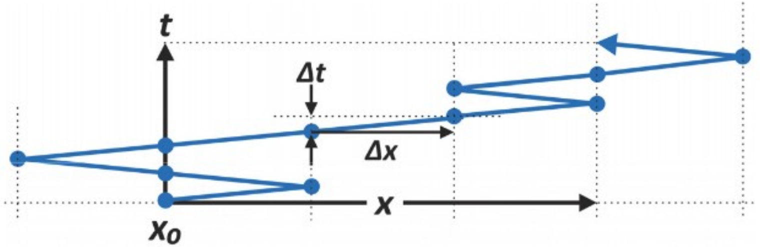
$$= \bar{r}_1 + \bar{r}_2 + \cdots\cdots \bar{r}_n$$

Random walk diffusion: a small simulation

https://rwd2d-mercury.runmercury.com/

Connection between Random Walk and Diffusivity

1D case



$x_n = x_{n-1} + \Delta x$

$\Delta x = \begin{cases} +1 \\ -1 \end{cases}$

① avg position

$\langle x_n \rangle = \langle x_{n-1} + \Delta x \rangle = \langle x_{n-1} \rangle + \cancel{\langle \Delta x \rangle}^{\,0}$

$= 0$

② avg spread of particles

$\langle x_n^2 \rangle = \langle x_{n-1}^2 + \cancel{2 x_{n-1} \Delta x}^{\,0} + \Delta x^2 \rangle$

$= \langle x_{n-1}^2 \rangle + \langle \Delta x^2 \rangle$

$= \langle x_{n-1}^2 \rangle + \Delta x^2$

$\langle x_0^2 \rangle = 0$

$\langle x_1^2 \rangle = \Delta x^2$

$\langle x_2^2 \rangle = 2 \Delta x^2$

$\vdots$

$\langle x_n^2 \rangle = n \Delta x^2$

$D \equiv \dfrac{\Delta x^2}{2 \Delta t}$

$\langle x(t)^2 \rangle = 2Dt$

$t = n \Delta t$

in 2D $\langle r^2 \rangle = \langle x^2 \rangle + \langle y^2 \rangle = 4Dt$

$= 2Dt + 2Dt$

in 3D $\langle r^2 \rangle = \langle x^2 \rangle + \langle y^2 \rangle + \langle z^2 \rangle = 6Dt$

Connection between Random Walk and Diffusivity

$$D = \frac{1}{6t}\langle R^2\rangle \quad \text{(in 3D)}$$

$\langle R^2 \rangle \sim t$

For the random walk model

$$|\bar{R}_n|^2 = \bar{R}_n \cdot \bar{R}_n = (\bar{r}_1 + \bar{r}_2 + \bar{r}_3 + \ldots \bar{r}_n)(\bar{r}_1 + \bar{r}_2 + \bar{r}_3 + \ldots \bar{r}_n)$$

$$= \underbrace{\sum_{k=1}^{n} \bar{r}_k^2}_{na^2} + \underbrace{2 \sum_{k=1}^{n-1} \sum_{j=k+1}^{n} \bar{r}_k \cdot \bar{r}_j}_{j \neq k}$$

$\bar{r}_k \cdot \bar{r}_j = r_k r_j \cos\theta_{ij}$

$$\langle R_n^2 \rangle = \left\langle na^2 \left(1 + \cancel{\frac{2}{n} \sum_{k=1}^{n-1} \sum_{j=k+1}^{n} \cos\theta_{kj}}\right)\right\rangle$$

avg to zero for truly random jumps

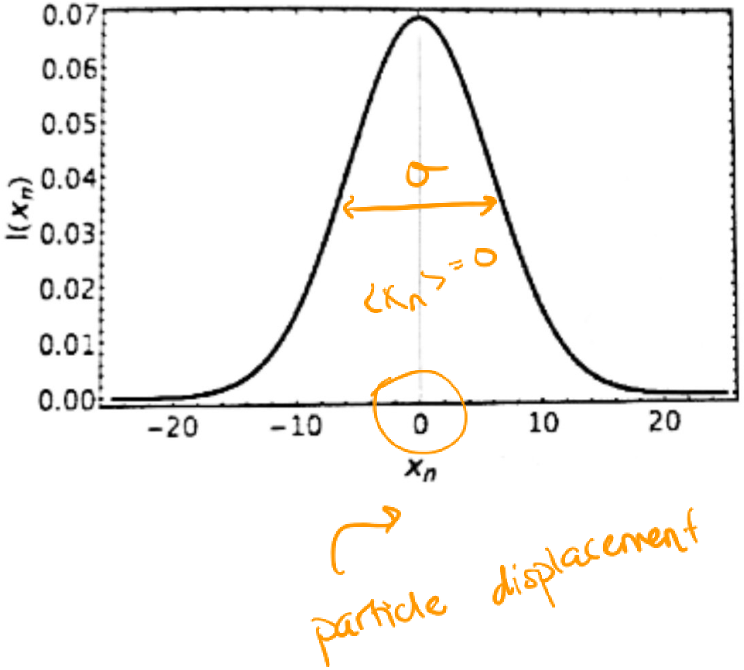
$$\boxed{\langle R_n^2 \rangle = na^2}$$

Statistics of the Random Walk Model: End-to-end distribution

1D case

Statistics of the Random Walk Model: End-to-end distribution

(In 1D)



$$\mathcal{P}(n_{tot}, q) = \frac{1}{2}^{n_{tot}} \frac{n_{tot}!}{(\frac{n_{tot}+q}{2})!(\frac{n_{tot}-q}{2})!}$$

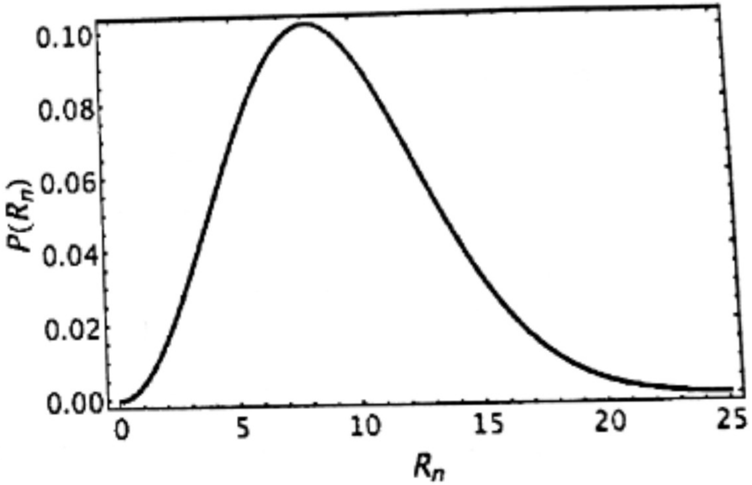
$$I(x_n) = \left(\frac{3}{2\pi \underbrace{na^2}}\right)^{1/2} \underbrace{\exp\left(-\frac{3}{2}\frac{x_n^2}{na^2}\right)}$$

$\langle R^2 \rangle$ Gaussian

σ

Statistics of the Random Walk Model: End-to-end distribution

(In 3D)



$P(\bar{R}_n) = I(x_n)\, I(y_n)\, I(z_n)$

$\downarrow$

convert spherical polar coord

$\int d^3r = \int r^2 \sin\theta \, dr\, d\theta\, d\phi$

$$\mathbb{P}(\bar{R}_n) = \left(\frac{3}{2\pi n a^2}\right)^{3/2} 4\pi R_n^2 \exp\left(-\frac{3R_n^2}{2na^2}\right)$$

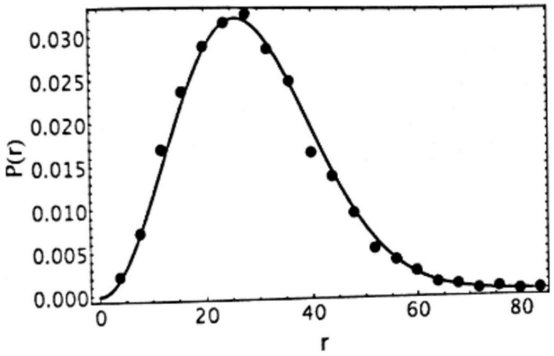
Coding considerations:

Binning distributions

Discretizing continuous functions
Appendix I.3

Choose n_{bin}

$$\Delta = \frac{R_n^{max} - R_n^{min}}{n_{bin}}$$



Random Number Generator

Appendix I.2

To reproduce a "random" result,
pick a consistent *seed*

Python:

numpy.random.rand(...): generates (pseudo-)random number over [0,1)

numpy.ceil(…): round to next highest integer

Donald R. Askeland. The Science and Engineering of Materials 3rd ed.

An implementation of the 2D Random Walk Diffusion

Objective	Concept	Python Representation
User chooses *nt* = number of time steps	Variable, integer	`nt`

An implementation of the 2D Random Walk Diffusion

Objective	Concept	Python Representation
User chooses *nt* = number of time steps	Variable, integer	`nt`

Keep track of position of random walker at each time step.
Let's assume it starts at the origin.
	Array (list of items)
	e.g., [3, 4.5, 8, -1]
	
	2D → *x* and *y* coordinate for each position
		Use the library numpy, shorthand is np:
		
		`x = np.zeros(nt+1)`
		`y = np.zeros(nt+1)`

An implementation of the 2D Random Walk Diffusion

Objective	Concept	Python Representation
User chooses *nt* = number of time steps	Variable, integer	`nt`

Keep track of position of random walker at each time step.
Let's assume it starts at the origin.
	Array (list of items)
	e.g., [3, 4.5, 8, -1]
	
	2D → *x* and *y* coordinate for each position
		Use the library numpy, shorthand is np:
		
		`x = np.zeros(nt+1)`
		`y = np.zeros(nt+1)`

Specify how the position changes at each time step.		`delx = np.array([?,?,?,?])` `dely = np.array([?,?,?,?])`

An implementation of the 2D Random Walk Diffusion

Objective	Concept	Python Representation
User chooses *nt* = number of time steps	Variable, integer	`nt`

Keep track of position of random walker at each time step.
Let’s assume it starts at the origin.
	Array (list of items)
	e.g., [3, 4.5, 8, -1]
	
	2D → *x* and *y* coordinate for each position
		Use the library numpy, shorthand is np:
		
		`x = np.zeros(nt+1)`
		`y = np.zeros(nt+1)`

Specify how the position changes at each time step.		`delx = np.array([?,?,?,?])` `dely = np.array([?,?,?,?])`

Save each new position of the diffusion path	Index the array
	i.e., access a specific element
	“zero index”
		`x = [1,2,3]`
		`x[0] = 1`
		`x[1] = 2`

An implementation of the 2D Random Walk Diffusion

Objective	Concept	Python Representation

Repeat for *nt* times	for loop
	range function
		Input:
		`for i in range(3):`
		`    print(i)`
		Output:
		`0`
		`1`
		`2`

Encode the random number to a change in position of the random walker	Generate a (pseudo)-random number	`np.floor(4* np.random.rand(nt))` Generate random number b/t 0 and 1 Random number b/t 0 and 4 Random integer: 0, 1, 2, 3