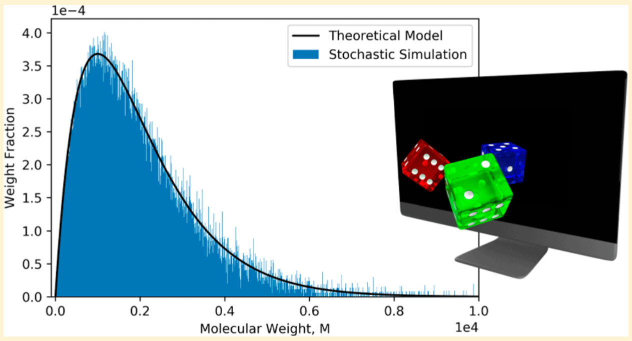
Introduction to Stochastic Simulations for Chemical and Physical Processes: Principles and Applications

Charles J. Weiss*

Department of Chemistry, Wabash College, Crawfordsville, Indiana 47933, United States

Supporting Information

ABSTRACT: An introduction to digital stochastic simulations for modeling a variety of physical and chemical processes is presented. Despite the importance of stochastic simulations in chemistry, the prevalence of turn-key software solutions can impose a layer of abstraction between the user and the underlying approach obscuring the methodology being employed. This article introduces the use of random variable generators in simulating physical and chemical processes suitable for use in undergraduate chemistry courses through detailed illustrative examples which include DNA sequences, Brownian motion, diffusion, chemical kinetics, and chain-growth polymerization. The results of the stochastic simulations are also compared to theoretical models. Jupyter notebooks containing Python code for running these simulations are provided in the Supporting Information section.



KEYWORDS: *Upper-Division Undergraduate, Interdisciplinary/Multidisciplinary, Computer-Based Learning, Kinetics, Polymerization*

INTRODUCTION

Stochastic[1] simulations employ random variables to dictate one or more aspects of the simulations. These are integral to modern chemical research[2] and are well-suited for such applications because of the inherent random nature of the underlying mechanisms for many physical and chemical processes such as diffusion and chemical kinetics.[3] Despite the prevalence, there is a paucity of introductory material on the subject appropriate for undergraduate chemistry courses. Many current examples that are at an undergraduate level use software that obscures the underlying stochastic process or employ dice/random value tables.[4−7] This article serves as an introduction to basic stochastic simulations for undergraduates and demonstrates how randomness guides physical and chemical processes with methods that can be implemented using computer software.

The first portion of this article provides the reader with an introduction to using random variables in simulations. The remainder of this article is devoted to detailed examples of stochastic simulations which include Brownian motion,[4] diffusion,[5] chemical kinetics,[6] and chain-growth polymerization.[7] Jupyter notebooks with the simulations coded in Python and NumPy are also provided in the Supporting Information for readers interested in experimenting with the simulations or for use in undergraduate courses. The methodologies described herein are not designed for computational efficiency or succinctness, but rather for simplicity and to mimic the physical or chemical processes being simulated to make them more approachable to chemistry students. These simulations are also designed to be simple to implement for students or instructors who know computer programming. They are also not intended to elucidate all the physical chemistry occurring in these examples but rather are focused on demonstrating how randomness guides behavior in many physical and chemical processes. The examples are appropriate for undergraduate-level curriculum and have been previously employed in an advanced, undergraduate course at Wabash College (see below for details).[8]

MATERIALS

The simulations provided in the Supporting Information are coded in Python[9] 3 and NumPy[10] which are both free, open source software applications available for Windows, Mac, and Linux.[11] The code for each simulation is provided in the Supporting Information as Jupyter notebooks[12,13] and in a PDF document for convenience. The plots included in this article were prepared using matplotlib[14] or seaborn,[15] which are free, open source Python plotting libraries, and the code for generating these plots is also included in the Supporting Information. Along with code, the Jupyter notebooks also contain markdown cells that contain explanatory text, equations, and instructions. Examples using the parameters presented in this article and in the Jupyter notebooks are not computationally intensive and complete within a few seconds on consumer grade laptop and desktop computers.

Received: June 8, 2017
Revised: September 10, 2017
Published: September 27, 2017

STOCHASTIC APPROACH

Solving Problems Stochastically

As an illustrative example of solving a problem stochastically, the statistical frequency of a specific pattern of nucleotides in random DNA strands can be determined using models generated from a random value generator. For example, it may be of interest to know the statistical frequency of the sequence GCAT in a DNA strand a thousand nucleotides long, in which the nucleotides (i.e., G, C, A, and T) occur randomly and with equal probability. This problem can be solved stochastically by observing the frequency of GCAT in multiple, randomly generated, thousand nucleotide DNA strands. If a large enough sample size is generated, the occurrences of GCAT will likely average to near the statistically expected value. This is a tedious process best left to a computer. If this simulation is run multiple times, there will be small variations in the values obtained due to the random element in the calculation, but all of the values are likely to be near 3.9 occurrences per strand. As more rounds are simulated, the average should ultimately converge on the theoretical value of 3.8945 occurrences (Figure 1). For example, a longer simulation using ten million rounds generated a mean of 3.8983 occurrences of the GCAT sequence per strand.

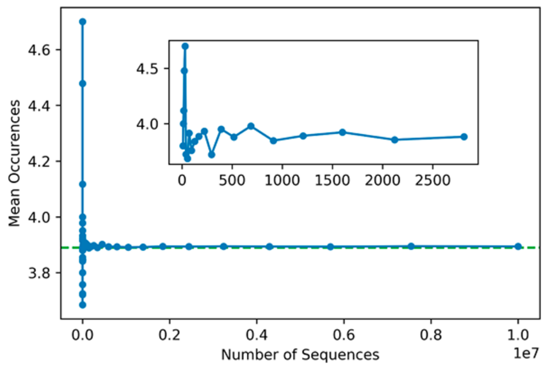


Figure 1. Plot of the mean occurrences of GCAT in simulated thousand nucleotide DNA stands with respect to the number of DNA strands simulated. An expanded view of values using up to three thousand generated strands is shown. The dashed horizontal line marks the theoretical value (see below).

Because this is a relatively simple, well-defined problem, it can also be solved analytically. Each nucleotide has a one in four chance of occurring at any position, so the chance of GCAT occurring in any four nucleotides is $(1/4)^4 = 1/256$. A DNA sequence of a thousand nucleotides will statistically have $997 \times (1/4)^4$ occurrences of the GCAT sequence which is approximately 3.8945 occurrences per sequence. The calculation uses 997 possible positions instead of a thousand because a four nucleotide sequence cannot begin in the last three nucleotides. This method of calculating frequency assumes that the chances of GCAT starting at any position in the DNA strand are independent of each other, but this fails to take into account that two GCAT sequences cannot overlap in a real DNA strand. The GCAT sequence occurrence in the DNA strand prohibits another GCAT sequence from starting in the next three positions (i.e., where the C, A, and T are located), so they are not truly independent. However, the theoretical probability of GCAT overlapping is small, so the occurrences of GCAT in any four nucleotides can be treated as *approximately* independent for this calculation.

Random Variable Generation

Random variables are the key driving force behind stochastic simulations, so it is important to be aware of the source and distributions of randomly generated variables. Variables can be generated in a variety of distributions to fit the needs of the simulation, and three key distributions that are often employed: uniform distribution, Poisson distribution, and binomial distribution. A *uniform distribution* is a series of random variables generated in a range where values appear with equal probability throughout the range.[16] A *Poisson distribution* describes the probability of a given number of independent events occurring in a specified time interval at a known mean rate. This is described by eq 1 where p is the probability of a given number of events (x) occurring in a time interval and λ is the known mean.[17]

$$p(x) = \frac{e^{-\lambda}\lambda^{x}}{x!} \tag{1}$$

This can be used, for example, to simulate total radiation detections per minute in a location where the mean detections per minute is known. The *binomial distribution* occurs when variables are randomly generated as a series of two possible outcomes,[18,19] and these two outcomes may be of equal or unequal probability and may include multiple trials. As is demonstrated below, this is useful for deciding if a molecule reacts or not or if a polymer chain terminates or continues to grow.

SIMULATION EXAMPLES

Brownian Motion

Brownian motion is the random movement of particles in a fluid[4b] as a result of collisions of the particles with the molecules of the fluid. If, for example, the position of a molecule of H_2 gas in a flask of N_2 gas could be monitored, as a thought experiment, its movement would appear to be a series of short, linear movements of constant velocity occurring between random changes in velocity (i.e., both speed and direction). This behavior is explained by *kinetic molecular theory* which describes a gas particle as moving with a constant velocity through space until it collides with another gas molecule. These collisions occur at seemingly random times, locations, and angles; depending upon the angle of the collision and velocity of the other particle, the original particle will change velocity in a seemingly random fashion.

The random movement of gas particles can be simulated using a method called a *random walk* which involves stochastically simulating a series of random molecular movements.[4] During every step of the simulation, a random variable, positive or negative, is added to the position of the particle causing it to move. To simulate Brownian motion in two dimensions (2D), a random number is added to the x coordinate and another random number to the y coordinate. The resulting x and y values are shown in Figure 2 (left), and this can easily be expanded to include a z-axis (see the Supporting Information for details) with an illustration of the results shown in Figure 2 (right).[20] The steps in this simulation are analogous to frames in an animation in that they represent movement forward in time.

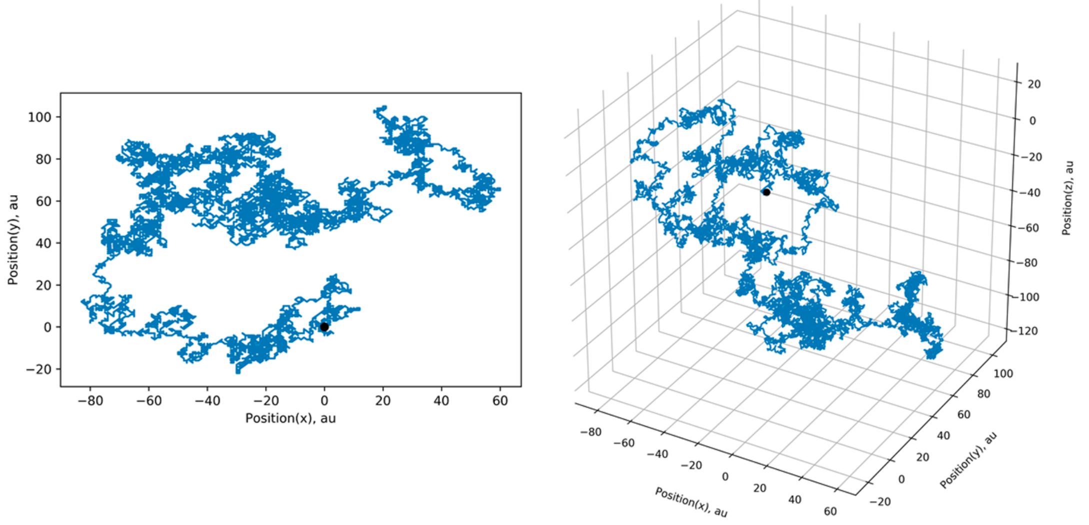


Figure 2. Results of both 2D (left) and 3D (right) random walk diffusion simulations with ten thousand steps. The black dots mark the starting positions in the simulations.

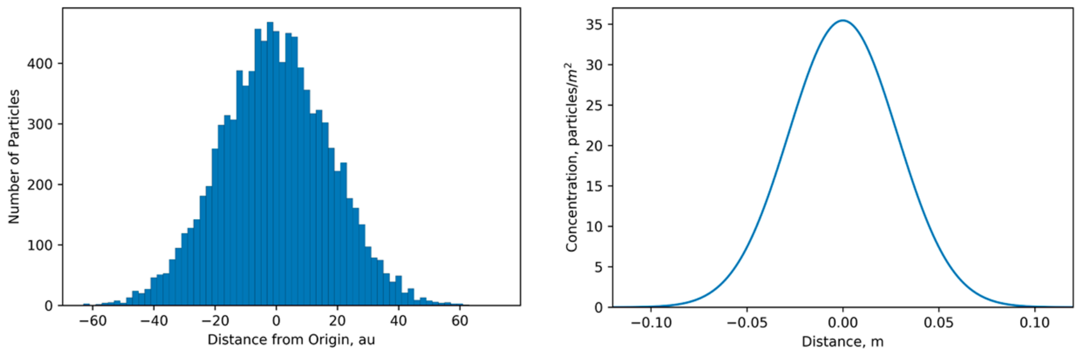


Figure 3. One-dimensional diffusion simulation for a thousand frames (left) and the theoretical model (right). The theoretical model shown here uses $D = 10^{-9}$ m^2/s, $A = 1$ m^2, $t = 4 \times 10^5$ s, and $n_0 = 1000$ particles.

Diffusion

Diffusion is the *net* movement of a solute from areas of high concentration to low concentration[21] as a result of Brownian motion until the solute concentration is homogeneous across the entire solution. Diffusion is an effect of a large number of solute particles, so instead of simulating one molecule of solute as is done in the Brownian motion simulation, this simulation requires a multitude of molecules. In contrast to the Brownian motion simulation, only the final positions of the molecules are important, so no attention is paid to where the solute molecules have been, only their final location. The diffusion of solute along a single axis can be simulated by adding a random value from the range [−1,1)[22] to each solute molecule position during each frame of the simulation. This causes the molecules to move along the diffusion axis. The results of the simulation are plotted in a histogram below (Figure 3, left) along with a plot of a theoretical distribution (Figure 3, right) based on eq 2.[23] In this equation, c is the concentration of particles at a given location and time, n_0 is the total number of particles, t is time in seconds, D is the diffusion coefficient, x is the position, and A is the area through which the particles are diffusing.

$$c(x, t) = \frac{n_0}{A\sqrt{\pi D t}} e^{-x^2/4Dt} \tag{2}$$

Similar to the Brownian motion simulations above, the diffusion simulation can be expanded to higher dimensions such as below (Figure 4) by adding additional axes (see Supporting Information for details).

Chemical Kinetics

The rates of chemical reactions are determined by random, microscopic processes such as the productive collision of two reactants or a single molecule achieving enough energy to surmount an activation barrier. The two examples below simulate single-step (i.e., elementary step), monomolecular reactions using random values to guide the underlying random processes. In a monomolecular reaction, the conversion of a reactant molecule A to product P (eq 3) is a random process where any molecule of reactant has a fixed probability of converting to product at any time.

$$\mathrm{A} \rightarrow \mathrm{P} \tag{3}$$

While the probability is independent of the *concentration* of A, the greater the *quantity* of A, the more A that will convert to product during a given period of time. Quantity is proportional to concentration in these simulations. Examples of this type of reaction include nuclear decay and some intramolecular chemical reactions.[24]

The goal of the kinetic simulations is to be able to plot the concentration of the reactants with respect to frames (i.e., proportional to time) which requires that the concentration(s) during each frame of the simulation be recorded. The general approach of the simulation outlined below mimics the approach

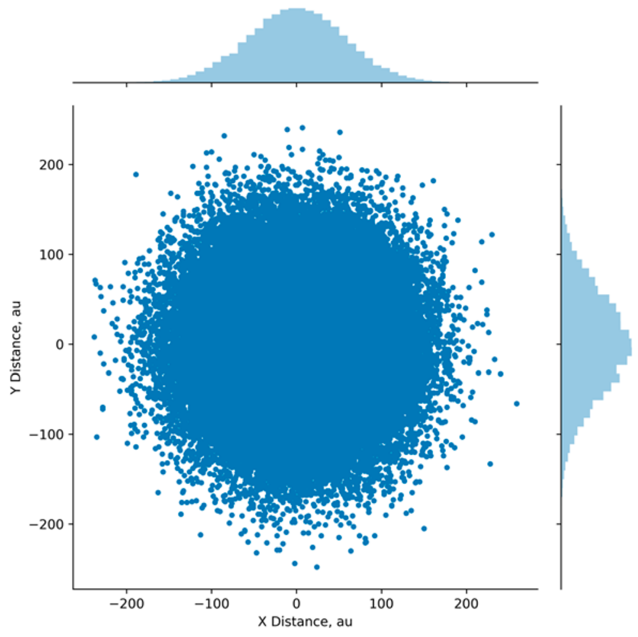


Figure 4. A plot of a two-dimensional (2D) diffusion simulation for a thousand frames. Histograms for the x and y axes are shown.

described by Rabinovitch.[25] An array of ones is created with each value representing a reactant molecule, and in each step/frame of the simulation, a random variable generator selects a position in the array. If the selected position contains a reactant molecule (i.e., a one), the value is changed to zero representing a reaction. An analogy for this approach is where a dart is thrown randomly and repeatedly at a wall covered in pieces of paper. If the dart hits a piece of paper, the piece is taken down representing a molecule reacting. One convenience of this approach is that as molecules of reactant are consumed, the chance of the random variable selecting a remaining reactant molecule inherently decreases simulating the first-order dependence in reactant A.

Results from this simulation are plotted below (Figure 5) with the results of the first-order[26] integrated rate law[27] displayed on the same axes. The stochastic simulation exhibits excellent agreement with the theoretical model. This is the result of a large number of simulated molecules and frames averaging to a predictable mean.

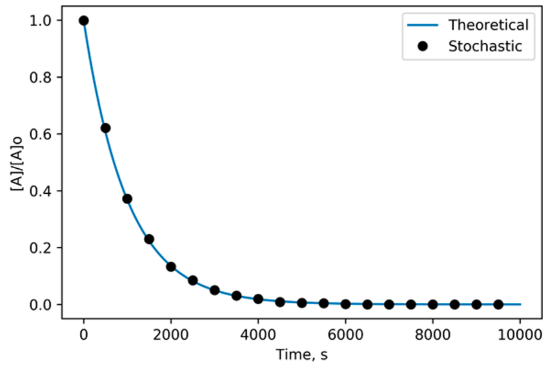


Figure 5. Results of the first-order, stochastic simulation of a thousand molecules and first-order integrated rate law (using k = 0.001 s^{-1}).[28] One of every five hundred points from the stochastic simulation is shown for clarity.

Two first-order reactions competing with each other can also be simulated with an analogous approach. In this reaction scenario, a single reactant A can form two possible products, B and C (eq 4).

$$C \leftarrow A \rightarrow B \tag{4}$$

Simulating this competition is similar to the single reaction above except the number of each product type also needs to be monitored and recorded. This is accomplished by changing a starting value of zero to either a one or two, representing the two possible products.[29] If the selected value in the array is already a one or two, no change occurs.

Competitive reactions do not necessarily occur at the same rates. To differentiate the two rates, one rate can be decreased by adding an additional condition for that reaction to occur. For the reaction of A → C, a second random variable, which generates a zero or one in equal probability, is added. Only when an unreacted molecule is selected in the array *and* the second random variable is a one does an A react to form a C. This reduces the rate of A → C by half. Results from this simulation are plotted below (Figure 6) and agree with the

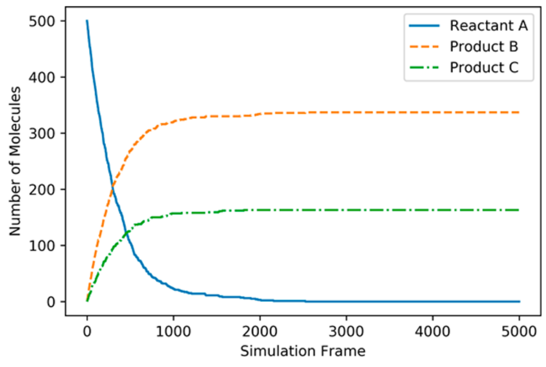


Figure 6. Stochastic simulation results for two first-order competitive reactions of A → B and A → C.

theoretical model.[30] The rate of formation of B is approximately[31] twice as fast as C resulting in twice the amount of B present at the completion of the reaction.

Chain-Growth Polymerization

Chain-growth polymerization is a typical mechanism for radical-mediated and metal-catalyzed reactions, and occurs by the successive addition of single monomer units onto live polymer chains.[32] The chains stop growing upon termination, which occurs at random times during the reaction, resulting in polymer strands of varying lengths. Chain-growth polymerization can be stochastically simulated to examine the distribution of chain lengths. During the simulation, an array stores the lengths of the simulated polymer strands, and in each step of the simulation, polymer lengths are incremented by one. At the end of each step, some polymer strands are randomly selected for termination by a binomial random variable generator with a set probability. Once a polymer strand has been terminated, its length remains unchanged for the remainder of the simulation.

The results of a simulation with one hundred thousand polymer strands, a 0.1% probability of chain termination in each step, and ten thousand steps are shown below (Figure 7). A histogram plot of the number of chains versus polymer chain weight, represented as the number of monomer units, shows that longer polymer chains are significantly less prevalent than

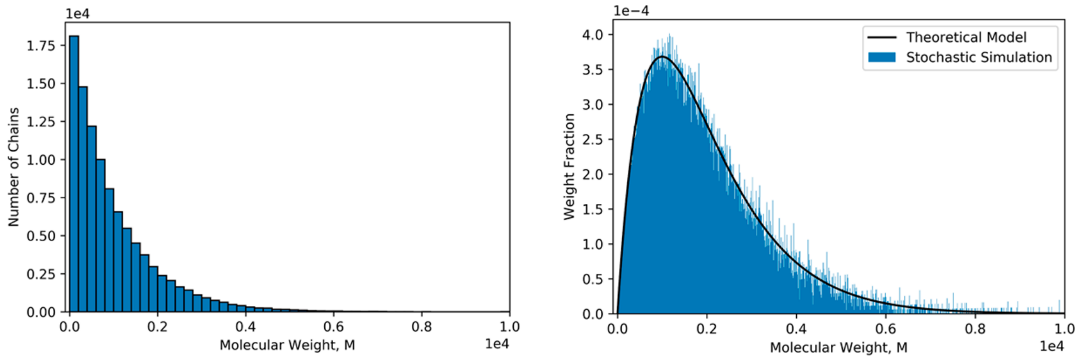


Figure 7. Histograms of chain length distributions (left) and weight fractions (right) from the stochastic simulation. The stochastic weight fraction values are divided by 10 to compensate for a histogram bin width of 10.[35] The theoretical weight fraction distribution (black line) is based on eq 5. Molecular weights (M_w's) are in terms of monomer units.

the shorter chains (Figure 7, left). Polymer molecular weight distributions are often represented in terms of *weight fraction*, *W*, which is the fraction of the total polymer sample weight that is from a particular polymer molecular weight. Weight fraction can be calculated by eq 5 where M_w is the molecular weight in monomer units, $\bar{M}_w$ is the average molecular weight, and $N(M_w)$ is the number fraction of chains of a given molecular weight.[33] Equation 6 describes the theoretical weight fraction for a given molecular weight based on the probability of growth versus termination (p).[33,34] The results are plotted along with the stochastic simulation (Figure 7, right) and exhibit excellent agreement between the two models.

$$W(M_w) = \frac{M_w}{\bar{M}_w} N(M_w) \tag{5}$$

$$W(M_w) = (1 - p)^2 M_w p^{M_w - 1} \tag{6}$$

CLASSROOM INCORPORATION

The above simulations have been included in the half-semester Scientific Computing for Chemists course taught at Wabash College.[8] This course is designed with the overall goal of teaching undergraduate chemistry students to use advanced computing tools to perform simulations and process, analyze, and visualize digital data. The results of a previously reported survey[8] show that students felt the content of the course significantly increases their abilities and interest in working with digital data and computer programming. Among the units, the course includes a two-day unit introducing stochastic simulations as a means of exposing students to the use of random values in the simulation of physical and chemical processes and how randomness guides the outcomes.

Prior to the stochastic unit, students are introduced to digital random value generation and different value distributions.[36] To give students practice, homework assignments include the generation of different types of random value distributions, the simulation of background radiation with a Geiger counter (see above), and the estimation of a numerical value of π using a random value generator (see the Supporting Information for details).[37]

The stochastic unit includes two class periods with the first day covering the Brownian motion, diffusion, and chemical kinetics simulations; the second day serves as an in-class, group activity on simulating chain-growth polymerization. Both days take place in a computer lab where each student sits at a computer workstation with all the required software already installed. The students are provided with Jupyter notebooks containing markdown cells that include explanatory text, equations, and instructions (available in the Supporting Information). This makes it convenient for students to add/modify code in the same document that contains the instructions and background information.

On the first day, the instructor presents the simulation methodologies and codes simulations in a Jupyter notebook projected on the front of the classroom while the students code simulations in Jupyter notebooks at their own computer workstations. The second day of the unit begins with a brief presentation of chain-growth polymerization chemistry followed by students forming groups of two or three to design and code their own stochastic polymerization simulations. Along with their stochastic simulations, students are asked to compare their simulation results to the theoretical model.[33]

SUMMARY

An understanding of stochastic methods is valuable in providing chemistry students with additional tools for solving problems and a deeper understanding of the underlying role of randomness in physical and chemical processes. While nonrandom on a macroscopic level, many physical and chemical changes are driven by randomness at a microscopic level. The examples in this article are suitable for undergraduate curriculum and may be used to introduce students to thinking about randomness and how to stochasticly simulate processes using computers.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available on the ACS Publications website at DOI: 10.1021/acs.jchemed.7b00395.

Software instructions, student handout, Python simulation code (PDF)

Jupyter notebooks (ZIP)

AUTHOR INFORMATION

Corresponding Author

*E-mail: weissc@wabash.edu.

ORCID

Charles J. Weiss: 0000-0003-4102-7683

Notes

The author declares no competing financial interest.

■ ACKNOWLEDGMENTS

The author thanks Wabash College and the Department of Chemistry for their support and thanks Dr. Jonathan M. Darmon, Dr. Scott E. Feller, and Dr. Chad R. Westphal for helpful comments and discussions.

■ REFERENCES

(1) Stochastic means something randomly determined or derived.

(2) For examples, see: (a) Pascal, T. A.; He, Y.; Jiang, S.; Goddard, W. A., III Thermodynamics of Water Stabilization of Carboxybetaine Hydrogels from Molecular Dynamics Simulations. *J. Phys. Chem. Lett.* **2011**, 2 (14), 1757−1760. (b) Molina, L. A.; Freire, J. J. Monte Carlo Simulation of Many-Chain Star Polymer Solutions. *Macromolecules* **1999**, *32* (2), 499−505. (c) Carnevali, P.; Tóth, G.; Toubassi, G.; Meshkat, S. N. Fast Protein Structure Prediction Using Monte Carlo Simulations with Modal Moves. *J. Am. Chem. Soc.* **2003**, *125* (47), 14244−14245.

(3) It is possible that processes that appear random may have an underlying nonrandom driving force that is simply not known about. However, many physical and chemical processes are close enough to random to be modeled through stochastic simulations.

(4) For examples of stochastic Brownian motion simulations, see: (a) Kottonau, J. An Interactive Computer Model for Improved Student Understanding of Random Particle Motion and Osmosis. *J. Chem. Educ.* **2011**, *88* (6), 772−775. (b) Muranaka, K. Simulation of One-Dimensional Brownian Motion by Stochastic Differential Equations. *J. Chem. Educ.* **1999**, *76* (7), 994−998. (c) Kirksey, H. G. Brownian Motion: A Classroom Demonstration and Student Experiment. *J. Chem. Educ.* **1988**, *65*, 1091−1093.

(5) For examples of stochastic diffusion simulations, see: (a) Grubbs, W. T. The Diffusion Game—Using Symbolic Mathematics Software To Play the Game on a Large Scale. *J. Chem. Educ.* **2006**, *83* (11), 1727. (b) Pérez-Rodríguez, P.; Gameiro, D.; Pérez-Pérez, M.; Lourenço, A.; Azevedo, N. F. Single Molecule Simulation of Diffusion and Enzyme Kinetics. *J. Phys. Chem. B* **2016**, *120* (16), 3809−3820. (c) Anderson, L. B.; Reilly, C. N. Teaching Electroanalytical Chemistry: Diffusion Controlled Processes. *J. Chem. Educ.* **1967**, *44* (1), 9−16.

(6) For examples of stochastic chemical kinetic simulations, see: (a) Bentenitis, N. A Convenient Tool for the Stochastic Simulation of Reaction Mechanisms. *J. Chem. Educ.* **2008**, *85* (8), 1146−1150. (b) Mira, J.; Fernández, C. G.; Urreaga, J. M. Examples of Deterministic versus Stochastic Modeling of Chemical Reactions. *J. Chem. Educ.* **2003**, *80* (12), 1488−1493. (c) Gillespie, D. T. Exact Stochastic Simulation of Coupled Chemical Reactions. *J. Phys. Chem.* **1977**, *81* (25), 2340−2361. (d) Moebs, M. D.; Haglund, E. A. A Simple Monte Carlo Method for Teaching Kinetics. *J. Chem. Educ.* **1976**, *53* (8), 506−507. (e) Para, A. F.; Lazzarini, E. Some Simple Classroom Experiments on the Monte Carlo Method. *J. Chem. Educ.* **1974**, *51* (5), 336−342. (f) Schultz, E. Dice-Shaking as an Analogy for Radioactive Decay and First-Order Kinetics. *J. Chem. Educ.* **1997**, *74* (5), 505−507.

(7) For examples of stochastic polymerization simulations, see: (a) Pintos, E.; Sarmoria, C.; Brandolin, A.; Asteasuain, M. Modeling of RAFT Polymerization Processes Using an Efficient Monte Carlo Algorithm in Julia. *Ind. Eng. Chem. Res.* **2016**, *55* (31), 8534−8547. (b) Sosnowski, S. Software for Demonstration of Features of Chain Polymerization Processes. *J. Chem. Educ.* **2013**, *90* (6), 793−795. (c) Pastor, R. W.; Karplus, M. Parametrization of the Friction Constant for Stochastic Simulations of Polymers. *J. Phys. Chem.* **1988**, *92* (9), 2636−2641. (d) Shahrouzi, J. R.; Guillaume, D.; Rouchon, P.; Da Costa, P. Stochastic Simulation and Single Events Kinetic Modeling: Application to Olefin Oligomerization. *Ind. Eng. Chem. Res.* **2008**, *47* (13), 4308−4316. (e) Tobita, H.; Takada, Y.; Nomura, M. Molecular Weight Distribution in Emulsion Polymerization. *Macromolecules* **1994**, *27* (14), 3804−3811.

(8) Weiss, C. J. Scientific Computing for Chemists: An Undergraduate Course in Simulations, Data Processing, and Visualization. *J. Chem. Educ.* **2017**, *94* (5), 592−597.

(9) Python Software Foundation. https://www.python.org (accessed Sep 2017).

(10) NumPy. http://www.numpy.org (accessed Sep 2017).

(11) Software was installed using Anaconda freely available at https://www.continuum.io/downloads (accessed Sep 2017).

(12) (a) *Jupyter Home Page.* https://jupyter.org/ (accessed Sep 2017). (b) Rossant, C. *Learning IPython for Interactive Computing and Data Visualization*; Packt Publishing Ltd.: Birmingham, U.K., 2013.

(13) The Jupyter notebook, formerly known as the IPython notebook, is an open, interactive document for holding code, explanatory text, the output of code, and other interactive elements. The notebook supports a range of programming languages and is becoming increasingly common in chemistry. For examples, see: (a) Srnec, M. N.; Upadhyay, S.; Madura, J. D. A Python Program for Solving Schrödinger's Equation in Undergraduate Physical Chemistry. *J. Chem. Educ.* **2017**, *94* (6), 813−815. (b) Cruzeiro, V. W. D.; Roitberg, A.; Polfer, N. C. Interactively Applying the Variational Method to the Dihydrogen Molecule: Exploring Bonding and Antibonding. *J. Chem. Educ.* **2016**, *93* (9), 1578−1585. (c) Srnec, M. N.; Upadhyay, S.; Madura, J. D. Teaching Reciprocal Space to Undergraduates via Theory and Code Components of an IPython Notebook. *J. Chem. Educ.* **2016**, *93* (12), 2106−2109.

(14) (a) Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* **2007**, *9* (3), 90−95. (b) *Matplotlib Home Page.* http://matplotlib.org/ (accessed Sep 2017).

(15) *Seaborn.* https://seaborn.pydata.org/ (accessed Sep 2017).

(16) Ghahramani, S. *Fundamentals of Probability*, 2nd ed.; Prentice Hall: Upper Saddle River, NJ, 2000; pp 247−248.

(17) Ref 16, pp 188−194.

(18) The NumPy binomial generator generates them as the outcomes zero and one.

(19) Ref 16, pp 174−175.

(20) The random numbers used in this simulation are the integers −1, 0, and +1 which fail to factor in much variation in the velocity of the molecule. This can be done instead by using randomly generated floats (i.e., decimal values) from [−1,1), but with no direct function to generate this exact spread of numbers, the values generated by existing functions must be altered such as by the following equation: rand_float = 2 × (np.random.rand() - 0.5).

(21) Atkins, P.; de Paula, J. *Physical Chemistry*, 7th ed.; W. H. Freeman and Company: New York, 2002; pp 815.

(22) The range of random values is generated from the range [−1, 1) which is inclusive of the lower value (−1) and exclusive of the upper value (+1). This is due to the way NumPy generates ranges of random values, but values from the range [−1, 1] could also be used.

(23) Ref 21, pp 848−849.

(24) For examples of intramolecular reactions with first-order or pseudo-first-order kinetics, see: (a) Leitch, D. C.; Platel, R. H.; Schafer, L. L. Mechanistic Elucidation of Intramolecular Aminoalkene Hydroamination Catalyzed by a Tethered Bis(ureate) Complex: Evidence for Proton-Assisted C−N Bond Formation at Zirconium. *J. Am. Chem. Soc.* **2011**, *133* (39), 15453−15463. (b) Yu, X.; Seo, S.; Marks, T. J. Effective, Selective Hydroalkoxylation/Cyclization of Alkynyl and Allenyl Alcohols Mediated by Lanthanide Catalysts. *J. Am. Chem. Soc.* **2007**, *129* (23), 7244−7245.

(25) Rabinovitch, B. The Monte Carlo Method: Plotting the Course of Complex Reactions. *J. Chem. Educ.* **1969**, *46* (5), 262−268.

(26) Because the simulated reactions are themselves elementary steps, their kinetic rate laws can be determined from the reaction stoichiometry.

(27) Meek, S. J.; Pitman, C. L.; Miller, A. J. M. Deducing Reaction Mechanism: A Guide for Students, Researchers, and Instructors. *J. Chem. Educ.* **2016**, *93* (2), 275−286.

(28) At the beginning of the simulation, nearly every random index results in a molecule of A reacting (i.e., 1 going to 0), so the rate is ~1/1000 per second. Therefore, the value of $k = 0.001\ s^{-1}$.

(29) The choice of number to use for reactants and products may be arbitrarily chosen but are chosen here as zero, one, and two for convenience when using the np.bincount() function to tally the number of zeros, ones, and two.

(30) Novak, I. Chemical Kinetics without Calculus. *J. Chem. Educ.* **1998**, *75* (12), 1574−1575.

(31) Because the results are dictated by a random number generator, the results will vary from simulation to simulation and are unlikely to be exactly the theoretical value. As more molecules are simulated, the results will asymptotically approach the theoretical values.

(32) Harris, F. W. Introduction to Polymer Chemistry. *J. Chem. Educ.* **1981**, *58* (11), 837−843.

(33) Horta, A.; Pastoriza, M. A. The Molecular Weight Distribution of Polymer Samples. *J. Chem. Educ.* **2007**, *84* (7), 1217−1221.

(34) This equation was originally developed for step-growth polymerization in the following references, but it also applies to chain-growth polymerization (see ref 33). (a) Flory, P. J. Molecular Size Distribution in Linear Condensation Polymers. *J. Am. Chem. Soc.* **1936**, *58* (10), 1877−1885. (b) Flory, P. J. Molecular Size Distribution in Ethylene Oxide Polymers. *J. Am. Chem. Soc.* **1940**, *62* (6), 1561−1565.

(35) With histogram bin widths of 10 monomer units wide, each bin contains *approximately* 10 times the number of polymer counts as would a histogram of the same data with bin widths of one polymer unit. The bin counts need to be divided by the bin width to compensate for this effect.

(36) Hill, C. *Learning Scientific Programming with Python*, 1st ed.; Cambridge University Press: Cambridge, U.K., 2016.

(37) *Monte Carlo Calculation of Pi*. http://mathfaculty.fullerton.edu/mathews/n2003/montecarlopimod.html (accessed Sep 2017).