Biophysics and Physiological Modeling
Chapter 2: Algorithms and pain relief

v.5.0 © pHn 2024

Introduction: Camp Omelet

The marble game we developed in **CHAPTER 1** is an excellent way to understand how diffusion works, but it's still just a game. In this chapter we'll learn how to turn the marble game into a realistic simulation (sim) of physiological situations ranging from single molecules up to the entire human body, e.g., drug elimination. In order to do that, we'll need to learn an essential skill for later chapters – how to design and implement quantitative models using algorithms. An **algorithm** is like the recipe for an omelet. It's a list of ingredients and a set of instructions written in a form that someone else can understand and follow to perform a complex task. Imagine…

> *Your family's going camping over Mother's Day weekend. The family tradition is for you to make Mom's favorite omelets for breakfast, but you're going to be out of town for an important interview at a research university. Your little brother Bert is willing to cook the omelets, but he's never made them before. You're about to get on a plane, but before you do, you need to email instructions to him. The family will be camping in a tent overnight, so Bert will need to collect **everything** he needs before they leave this afternoon, including the special ingredients – a pinch of dried tarragon and crumbled blue cheese that are lightly beaten into the egg mixture before it's cooked. Oh, and just to make life more interesting, the Fowlers (from next door) are probably coming, but Bert won't know how many to cook for until after you get on the plane.*

Q.2.1 **DISCUSSION QUESTION** *Write* the email for Bert. He's promised to follow your instructions carefully, but you'll have to make them as clear and complete as possible. Bert has a family reputation for being very meticulous and extremely literal. If you forget something in your email, he'll be sure to let everyone know that he was just following *your* instructions.

Hint: If you've never made an omelet before, you should **Google** "basic omelet recipe" to get an idea of how to do it.

About what you discovered: good recipes make good omelets

Writing instructions like this from memory is actually a significant challenge. A good strategy is to imagine being in Bert's shoes on Mother's Day morning, and then to picture in your mind what he will have to do. If you do that, you should realize that Bert will need to take some basic kitchen utensils with him to make the omelets, including a frying pan. **Heads up:** This question makes a good in-class activity, so take the task seriously. Your instructor might choose your recipe as an eggs-ample (sorry, I couldn't resist!).

Most people would find this task easier if they could do a practice run and actually make an omelet while writing out the instructions for Bert. The same applies to algorithms. That's why we'll always test out our algorithms by filling in an **output table** *at the same time* that we're writing out the algorithm. Using this procedure, we'll discover any deficiencies in our algorithm before we tangle with Excel. ❑

Algorithms are like recipes

In a lot of ways, writing an Excel spreadsheet is very much like writing the instructions for Bert. However, Excel is even more literal! Excel will *blithely* follow your instructions even if they don't make any sense at all! If you make a typo, Excel will do something unexpected or simply return a cryptic message like **#NAME?** to tell you that it didn't recognize what you typed. Writing an algorithm can be much more challenging than writing out instructions for something that you already know how to do (like cooking Mom's favorite omelet). In this book, we'll often want to write an algorithm for something that we've never even thought of before! Luckily, unlike the omelet example, we'll be able to test out our algorithm while we're writing it out. That way, if we make a mistake (like forgetting to include the number of people eating in the recipe) we'll be able to fix that *before* we start with the spreadsheet.

In **SECTION 2.1** of this chapter, we'll learn how to write an algorithm and then test it (by hand). Along the way, we'll learn about the two main parts of our algorithm: **parameters** and **variables**. Parameters are like the number of people eating and they don't change while we're cooking. Variables are like the eggs and do change while we're cooking the recipe. N (the total number of marbles) will be a parameter, that way we can easily change the number of marbles in the game (just like the number of people eating). We'll also add a **jump rate constant** k to describe how frequently (fast) the marbles jump from box to box. The main practical purpose of **SECTION 2.1** is for us to learn the composite process of **(a)** writing an algorithm; and **(b)** simultaneously testing that it works correctly by calculating an **output table** – by hand – using the algorithm – as we write it.

In **SECTION 2.2**, we'll use our (pretested) algorithm to write a spreadsheet to get Excel to do all the hard work. We'll also learn about how parameters should be referred to using "absolute cell references" in Excel so that each cell knows exactly where the parameter is located. If the spreadsheet isn't working correctly, we'll use our tested algorithm to figure out what went wrong. Getting the **bugs** out, or "**debugging** the spreadsheet", is much easier if you can compare what it's actually doing with what you know should happen (based on the algorithm). Everybody makes mistakes – nobody's perfect… including me! – I still find **bugs** when I'm working on new spreadsheets…

Once we've completed the spreadsheet and tested that it's working correctly, we'll use it in **SECTION 2.3** to discover how the sim properties depend on the number of marbles N – if everything else remains the same (including the jump rate constant k). In **SECTION 2.4** we'll discover that the marble game can be adjusted to model physiological systems ranging from the size of a single molecule up to the entire human body. We can do this by changing what the

marbles and boxes represent and by changing the value of the jump rate constants k to realistic values for those systems. The fastest rate constants go with the molecular demons of speed – ion channels and aquaporins (water channels) where the jumps take just nanoseconds, and the slowest rate constants can be associated with processes that can take decades – like removing heavy metals (like lead) from bone.

Finally, in **SECTION 2.5** we'll learn how to develop a marble game simulation of drug elimination from the body. Using this simple sim, we'll study the "pharmacokinetics" of elimination of a pain killer – TYLENOL® (whose generic name is acetaminophen in the USA and paracetamol almost everywhere else) and discover that our sim *predicts* the exponential decay observed clinically. This simple sim also explains a fundamental quantity in the subject of pharmacokinetics – the drug half-life. Let the games begin…

2.1 Generalizing the marble game

In Sections 1.4 and 1.5 of **CHAPTER 1** we discovered that changing the number of marbles from $N = 10$ to $N = 100$ or $N = 200$ dramatically changes how the game plays. However, to make that change we had to modify 1000 cells. Wouldn't it be great if we could change just one cell and get the same results? If we could also change the marble jumping rates, then the same basic model can be used for an amazing wide range of physiological systems from a single molecule to the entire human body (as we'll see in **SECTION 2.4** of this chapter). Rather than just diving in and changing the spreadsheet, we'll talk first about a fantastically useful technique for designing simulations (or for designing *any* process).

Algorithm development & explaining what a simulation does

An **algorithm** is a complete set of instructions that you (or your little brother) could follow to perform the simulation by hand. While it's fairly easy to construct a spreadsheet from an algorithm, it can be extremely difficult to do the reverse – to figure out the steps of a complicated algorithm, just by looking at a spreadsheet! From now on, we'll write an algorithm and then work through it by hand… before diving into Excel. While you're doing that, you'll sometimes realize that you're missing an important ingredient (like N in the example above). It's usually much easier to change a handwritten algorithm than it is to reorganize a large spreadsheet full of cross references.

Marble game simulation model

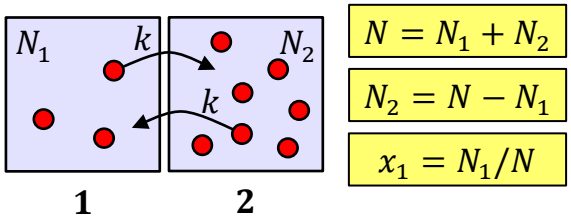

Fig.2.1 Marble game representation of the marble game kinetic Monte Carlo (kMC) simulation. The particles move just like in the original marble game, but the jumping rate is now characterized by a rate constant k.

In the **model** shown in Fig.2.1 we're thinking of a **system** of small particles (such as oxygen molecules) that are being constantly jostled by the surrounding molecules because they are all in **random thermal motion**. As a result, the particles appear to jump around randomly. This random jumping is called **Brownian motion** and occurs even for particles that are much larger than water molecules. It was first noticed by Robert Brown, way back in the late 1820's, when he observed pollen grains jiggling around while they were suspended in water. For more information refer back to Fig.1.4 and the related Brownian motion demo and MarbleGame sim. We'll be studying Brownian motion in more detail in **CHAPTER 10**.

After jiggling around for a while, a particle can end up in the other box. The average rate at which these inter-box **jumps** occur is k jumps per second *per particle*. The actual value of k for a particular physical system will depend on many of its properties, such as the identity of the particles, the size and location of the boxes and whether there's a membrane between the boxes. Just for now, we'll do the same thing that you did in chemistry – and assume that the jump rate constant k can be determined from experiment. For the sake of argument, let's think of a system like our original marble game, where each particle jumps to the other box with an average rate of three jumps per minute (or 1 jump per 20 seconds). This can be summarized mathematically by the **jump rate constant** k

$$k = \frac{3\ \cancel{\text{jumps}}}{\cancel{\text{min}}}\left(\frac{1\ \cancel{\text{min}}}{60\ s}\right) = 0.05\ \frac{1}{\text{s}} = 0.05\ \text{s}^{-1} \tag{2.1}$$

Where we've "done algebra" with the units to get the final equivalent form of $k = 0.05\ 1/\text{s}$. This means that the jump rate constant k has units of $1/\text{s}$ (or s^{-1}), which is pronounced "per second". This can be abbreviated as k [=] $1/\text{s}$, where the symbol [=] reads "**has units of**".

About what you discovered: units and unit conversions

The **jump rate constant** k is the number of jumps a marble takes per unit time *on average*. If we use minutes for time, k is three jumps per minute (per marble). Equation (2.1) converts this rate into a jump rate per second. The word "jumps" is struck out in equation (2.1) because jumps [=] 1 (have the same units as the number 1), i.e., "jumps" have no SI units because they're a simple count of how many (jumps) occur in one minute. The factor in parentheses in equation (2.1) is a **unit conversion**. If we multiply k by the same thing (1 min) over the same thing (60 s), that's the same as multiplying k by one, which doesn't change k. The "min"s cancel producing the final result of $3/60 = 0.05$ (jumps) per second. It's hard to overstate the importance of units in biophysics and physiological modeling. We'll talk more about units in **CHAPTER 3**. ❑

This model is identical to the original marble game, but now there's a definite amount of physical time that's associated with each **step** (turn) of the **simulation** (game). How much time? Well, if one particle has a jump rate of k times a second, then N particles will have a *combined* **jump rate** of Nk jumps per second. This means that the average time Δt between jumps (average time interval between the jump of one particle and the next jump of any one of the particles) is given by

$$\Delta t = \frac{1}{Nk} \tag{2.2}$$

where the symbol Δ is the uppercase Greek letter "D", which is pronounced "delta". The symbol Δ is a prefix that's used to indicate a "difference" or "change in" the symbol that follows it. Hence, the symbol Δt is the difference or change in time between jumps. Watch the Greek letters go green! video [Nelson 2013] for a review of the Greek alphabet and review **APPENDIX A.6**.

According to equation (2.2), if we know k and N for our simulation, then the average amount of time associated with each simulation step is completely determined. For example, if $k = 0.05\ 1/\text{s}$ and $N = 10$, then equation (2.2) tells us that

$$\Delta t = \frac{1}{Nk} = \frac{1}{(10)\left(0.05\ \frac{1}{\text{s}}\right)} = \frac{1}{0.5}\left(\frac{1}{\left(\frac{1}{\text{s}}\right)}\right) = 2\ \text{s} \tag{2.3}$$

… make sure that you understand how to do the calculation of both the number (2) and the units (s) in equation (2.3) – it's important for all that follows. Note that when we plugged in the numbers, we simply replaced the letter with the corresponding number-with-units in parentheses in $\frac{\text{over}}{\text{under}}$ fraction format. The answer $\Delta t = 2$ s means that 2 seconds elapse between turns in the marble game. This **timestep** Δt is the time between steps. Monte Carlo simulations that have a specific time interval Δt associated with each step are called **kinetic Monte Carlo** (**kMC**) sims to highlight that they have realistic kinetics k included.

About what you discovered: inline units $1/\text{s} = \text{s}^{-1}$

It's common in textbooks (including this one) to write units in inline form rather than as a fraction, e.g., to write s^{-1}, pronounced "per second" or "inverse seconds", instead of the equivalent $\frac{1}{\text{s}}$ or $1/\text{s}$. However, in the algorithms below, we'll use k $(1/\text{s})$ for the jump rate constant and its units because it's much easier to type k $(1/\text{s})$ in Excel than its equivalent k (s^{-1}). However, the National Institute of Standards and Technology (NIST) does recommend that complicated units be written in-line with half-high dots and that negative exponents should be used in complicated cases, e.g. the answer to the question "what's a unit of power" (in base SI units) is $\text{kg} \cdot \text{m} \cdot \text{s}^{-3}$. ❑

Q.2.2 Over the years, I've noticed that some of my students confuse units and their inverse – especially when they're written inline, e.g., s versus s^{-1}. As practice for distinguishing between them, please answer the following questions. You should write your answer as a number with inline units.

(a) How long does the second hand of a clock take to complete one circle? Your answer should be a number with units of either s or s^{-1}. You should make the correct choice for the units.

(b) What is the rate at which the second hand "ticks"? Use units of either s or s^{-1}.

(c) Re-answer part (a) using units of either min or min^{-1}.

(d) Re-answer part (b) using units of either min or min^{-1}.
(e) How often do leap days (February 29) occur? Use units of either yr or yr^{-1}.

About what you discovered: second or per second?

It's really important that you know how to answer Q.2.2 correctly. For example, the correct answer for Q.2.2(b) is that a second hand of a clock ticks at a rate of 1 tick per second, which is written mathematically as $1\ \text{s}^{-1}$. Similarly, the correct answer for Q.2.2(d) is that a second hand of a clock ticks at a rate of 60 ticks per minute, which is written mathematically as $60\ \text{min}^{-1}$. Finally, for Q.2.2(e), leap days occur once every four years, which is written mathematically as $1/(4\ \text{yr}) = 0.25\ \text{yr}^{-1}$. ❑

About what you discovered: how to do better on tests!

Equation (2.2) is an example of an equation that you might forget on a test in one of your science classes. However, if you remember that the definition of k is the rate at which one particle jumps, so that it $[=]\ 1/\text{s}$ (has units of per second), then you should be able to figure out the equation for Δt – the average time between jumps of N particles using **dimensional analysis**. The N particles jump at a combined rate of Nk. Now $N\ [=]\ 1$, meaning N has the units of a number (e.g., 1), i.e., *no units*. Hence, if one particle jumps with rate $k\ [=]\ \text{s}^{-1}$ then N particles jump at rate $Nk\ [=]\ \text{s}^{-1}$. The quantity that we're looking for (timestep) has units of seconds ($\Delta t\ [=]$ s). Hence, just by comparing the units, we can guess that $\Delta t = 1/(Nk)$ because this equation is dimensionally correct. There is no absolute guarantee that your equation is correct in the end, but it is a really good guess! Hence, you should always remember that **dimensional analysis** can help you do better on tests! ❑

If we focus on just one of the marbles in our simulation game, we can ask how long it takes for it to jump to the other box on average. This **mean residence time** τ (average time a marble stays in a box) is given by equation (2.2) with $N = 1$ and $\Delta t = \tau$, i.e.

$$\tau = \frac{1}{k} \tag{2.4}$$

where the symbol τ is the lowercase Greek letter "t", which is pronounced "tau" [Nelson 2013]. Using the Greek letter tau helps to distinguish the mean residence time from all the other times that we'll be talking about. Equation (2.4) states that the jump rate constant k and mean residence time τ provide the same basic information about the timescale of the jumps. If you know one, then you can calculate the other. I.e., if $k = 1/(20\ \text{s})$ then $\tau = 20\ \text{s}$.

About what you discovered: equations describe relationships

Consider the following question and answer about familial relationships: *Who is the daughter of your grandmother's son's sister?* If a person answers *"I am."* then we can infer two things. (i) The person is female and (ii) she has an uncle on her mother's side of the family. The question and answer provide relationships that can be combined to infer (i) and (ii). In an analogous manner,

equations describe **mathematical relationships** between physical things. When these relationships are combined using algebra, they allow us to infer new relationships. As an example, equation (2.2) $\Delta t = 1/(Nk)$ can be combined with equation (2.4) $\tau = 1/k$ to infer that

$$\Delta t = \frac{\tau}{N} \tag{2.5}$$

i.e., the timestep in the kMC simulation is proportional to the mean residence time τ and **inversely proportional** to the number of marbles N. From equation (2.4) we can also infer that the mean residence time τ does not depend on N, i.e., there's no N in equation (2.4), so if you change N then τ doesn't change. Hence, an individual marble jumps at the same rate *independent of how many other marbles there are*. The fact that τ is a property of a single marble (independent of N) is an important physical feature of our marble game – **marble independence**. ❑

About what you discovered: residence time, jump time, dwell time etc.

This concept of how long a molecule stays in a particular place (or state) is becoming a much more commonly discussed concept in biophysics, molecular biology and biochemistry as science becomes increasingly focused on the behavior of individual molecules. ❑

Finally, as we'll be interested in being able to change the number of marbles in the system (i.e., N) and then to compare different systems, we'll introduce a variable x_1 that's **defined** as

$$x_1 \equiv \frac{N_1}{N} \tag{2.6}$$

for the **fraction of marbles** in box 1 (Fig.2.1). As we mentioned in **CHAPTER 1**, The symbol $\equiv$ is an alternate $=$ sign that is pronounced "**is defined as**". Also, when you read equation (2.6), you should recall that N is the total number of marbles in the system, which is related to the number in each box by the **bookkeeping equation**

$$N = N_1 + N_2 \tag{1.1}$$

About what you discovered: translating math into English

Equation (2.6) translates into English as "the fraction of marbles in box 1 is defined as the number of marbles in box 1 divided by the total number of marbles", which is a lot of words to be represented by just four symbols. It's important that you always consciously translate the math into English every time you read an equation or other mathematical expression in this book … until you become fluent in the language of math. If you can't remember what a symbol means, go back and reread the previous section(s) or refer to the glossary of symbols in **APPENDIX B.1**, which lists all the variables used in the book in alphabetical order – math vocabulary! ❑

To see why x_1 is useful, think about $x_1 = 0.5$. Any simulation that has an equal number of particles in each of the two boxes has $x_1 = 0.5$, independent of the number of marbles N in the

marble game. N_1 tells us exactly the same information, but its value depends on the total number of marbles N. x_1 is an **intensive property**, because it doesn't depend on "system size" (in this case the total number of marbles), whereas N_1 is an **extensive property**, because it does depend on the total number of marbles in the game. Both ways of describing the location of the particles are useful.

We're now going to talk about how to write an algorithm that *completely describes* the sim. When you're working on an algorithm you should write out your answer using pencil and paper (just like working physics problems). In my experience, most students find it much easier that way… you can take a photograph of (or scan) the handwritten answer and paste it into a Word document with the rest of your answer. Most people find that typing algorithms in Word is awkward because of the complex formatting that's needed. It also takes *way longer* than writing it out by hand. When we're done writing the algorithm, it should look something like Fig.2.2.

The algorithm is just like the recipe for Mom's favorite omelet. It can be broken down into a list of ingredients (**parameters** and **variables**) and a set of instructions (**step 0**, **step 1**, **step 2**…) – see Fig.2.2. In the following sections, we'll go through the complete algorithm and explain what all the parts are. Writing algorithms is a skill that we'll use over and over in this book.

The algorithm

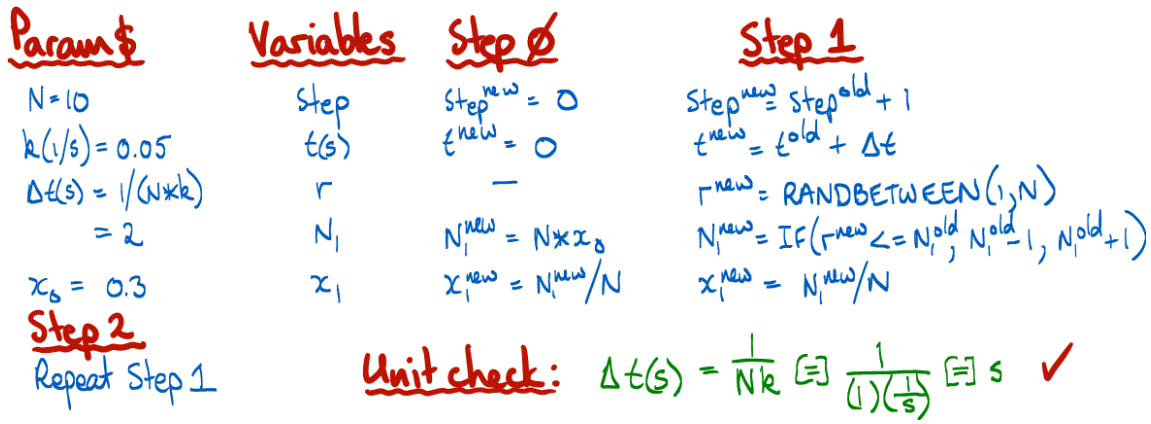

Fig.2.2 Algorithm for the marble game kMC sim shown in Fig.2.1.

The algorithm in Fig.2.2 describes our kinetic Monte Carlo sim of the marble game with $N = 10$, $k = 0.05$ 1/s, and $x_0 = 0.3$, where x_0 is the **initial fraction** of marbles in box 1. The subscript 0 is pronounced "naught" – for zero time. The discussion of Tables 2.1 – 2.6 below explains what the algorithm should include. You don't need to include the "Comment or explanation", but for full credit you must include the **units** with any parameter or variable when it's introduced (see Fig.2.2).

Table 2.1 Parameter list for the marble game kMC algorithm

Parameter (units) = value	Comment or explanation
$N = 10$	total number of particles

$k\ (1/\text{s}) = 0.05$	jump rate constant (per particle per second)
$\Delta t\ (\text{s}) = 1/(N * k) = 2$	time between jumps
$x_0 = 0.3$	initial value of the fraction in box 1

The numerical things that you need to know before the simulation can start are called **parameters** – see Table 2.1. If you change a parameter, you end up changing the whole simulation. N is the total number of particles in the marble game system. $N = 10$ in the original marble game. N has no units. It's a **dimensionless quantity** because it's a counting number with no units. k is the rate constant for jumps of any particle. We've written this out in the format $k\ (1/\text{s}) = 0.05$. The parameter letter k appears first and is followed by the units in parentheses $(1/\text{s})$ so that the following number 0.05 means one twentieth of a jump per second (or 1 jump per 20 seconds). All parameters that have units should be written in this format… The reason for that should become clearer in **SECTION 2.2** below. The basic idea is that "$k\ (1/\text{s})$" as a heading for the number that appears below it in the spreadsheet (see Fig.2.3 below). If you include the units in the heading, then you don't need units with the number itself.

Δt is the amount of time (in seconds) that elapses between steps in the simulation. This is a **calculated parameter** that depends on the values of the two previous parameters. Your algorithm should specify that the equation $\Delta t\ (\text{s}) = 1/(N * k)$ will be used to calculate it in the spreadsheet. If you entered the value of $\Delta t\ (\text{s}) = 2$ then you'd have to change the algorithm every time you changed either N or k! Entering the formula, Δt remains correct even if you change N or k (or both). This is like the instructions for Bert saying, "bring three eggs per person". The "$\Delta t\ (\text{s})$" will be a heading in Excel so it needs units. The expression on the right "$= 1/(N * k)$" doesn't need units because we've already defined k and N. In the algorithm, the fraction is written as an **inline equation** with parentheses. A $*$ sign is used for multiplication because that's the format that Excel understands. Recall that the purpose of writing an algorithm is to set out what you'll type into Excel. Everywhere else in this book we'll use the usual math format $\frac{1}{Nk}$ which doesn't require the parentheses or the asterisk.

Warning: If you type $= 1/N * k$ into Excel – you'll get the wrong answer! The parentheses in Table 2.1 are *required* for the inline equation to work correctly in Excel. That's why I'm so particular about you writing inline equations in algorithms. Okay, so now that you've been warned, don't fall into this common trap for new players!

About what you discovered: algorithms are formatted to match Excel input

The algorithm format used in Fig.2.2 is designed to match what you'll input into Excel. That way after you've written the algorithm you can focus on simply typing it into Excel correctly without having to rearrange equations or work out heading formats. While you're writing an algorithm, you should remember its purpose: to make entering the spreadsheet as painless and error free as possible. Hence, don't forget to use the rules for inline equations. ❑

x_0 tells us how many particles to put in box 1 at $Step$ 0 (zero) of the simulation. For example, $N = 10$ and $x_0 = 0.2$ will give $N_1 = 2$ particles in box 1 at the beginning of the simulation.

Table 2.2 Variable list for the marble game kMC algorithm

Variable(units)	Comment or explanation
$Step$	counter for number of simulation steps
t (s)	current elapsed time (in seconds)
r	random number of chosen marble
N_1	number of marbles in box 1
x_1	fraction in box 1

The list of ingredients that change in the simulation are the **variables**. These are listed in Table 2.2. While you're writing out an algorithm, it's a good idea to leave extra space in the parameter and variable lists. It's common to discover that you've forgotten a key ingredient that wasn't obvious at the beginning. For example, while writing out the recipe for Mom's favorite omelet, you might not remember that salt and pepper (to taste) are ingredients until the omelet is cooked.

Table 2.3 Instructions for **Step 0** of the marble game

Instruction	Comment or explanation
$Step^{\text{new}} = 0$	initialize step count to zero
$t^{\text{new}} = 0$	initialize simulation clock to zero
$N_1^{\text{new}} = N * x_0$	calculate initial value of N_1 using parameters N and x_0
$x_1^{\text{new}} = N_1^{\text{new}}/N$	calculate initial fraction in box 1

With the parameters and variables at hand, we're now ready to write out the steps of the simulation. The first thing we need to do is set up the starting configuration of the sim. We'll call this $Step$ 0 as this **initialization** occurs before the simulation actually begins. As shown in Table 2.3, $Step^{\text{new}} = 0$ sets the value of $Step$ to zero at the beginning of the simulation. The superscript "new" indicates that we're referring to the current step ($Step$ 0). Similarly, $t^{\text{new}} = 0$ sets the first value of time to zero. **Note:** You don't need to include the units here as they are already specified in the variable list. The equation $N_1^{\text{new}} = N * x_0$ calculates the **initial value** of N_1 (the value of N_1 in $Step$ 0) from the parameters N and x_0. In our example, this equation calculates as $N_1 = 10 * 0.3 = 3$. **Note:** N and x_0 do not need a "new" superscript as they are parameters that don't change during the simulation. The last thing we'll do in $Step$ 0 is to calculate the initial value of x_1 using the current value of N_1 (this matches what we'll do in $Step$ 1).

Table 2.4 Instructions for **Step 1** of the marble game

Instruction	Comment or explanation

$Step^{\text{new}} = Step^{\text{old}} + 1$	new $Step$ is the previous value plus 1
$t^{\text{new}} = t^{\text{old}} + \Delta t$	increment clock using timestep Δt
$r^{\text{new}} = \text{RANDBETWEEN}(1, N)$	randomly choose one of the N marbles (call it r^{new})
$N_1^{\text{new}} = \text{IF}(r^{\text{new}} <= N_1^{\text{old}}, N_1^{\text{old}} - 1, N_1^{\text{old}} + 1)$	if r^{new} is less than or equal to N_1^{old} jump 1 → 2 (N_1^{new} decreases), else jump 2 → 1 (N_1^{new} increases)
$x_1^{\text{new}} = N_1^{\text{new}}/N$	calculate new fraction in box 1

As shown in Table 2.4, the $Step$ 1 instruction $Step^{\text{new}} = Step^{\text{old}} + 1$ calculates the value of $Step$ for the current step by adding one to the previous value of $Step$. $t^{\text{new}} = t^{\text{old}} + \Delta t$ calculates the current time by adding Δt to the time in the previous step. You should note that the superscript "new" always refers to the value of the variable in the current step, and the superscript "old" always refers to the previous step. The instruction

$$r^{\text{new}} = \text{RANDBETWEEN}(1, N) \tag{2.7}$$

selects one of the N particles at random with equal **probability**, just like rolling the ten-sided dice. Note that this instruction, like the rest of the algorithm, uses N rather than the number 10. That way if we change the parameter N, the whole sim changes accordingly. The next instruction calculates the new value of N_1 based on the particle that was just chosen.

$$N_1^{\text{new}} = \text{IF}(r^{\text{new}} <= N_1^{\text{old}}, N_1^{\text{old}} - 1, N_1^{\text{old}} + 1) \tag{2.8}$$

In words, if r^{new} is less than or equal to the value of N_1 in the *previous* step $\left(N_1^{\text{old}}\right)$, then the *new* value (N_1^{new}) (in the current step) should be $N_1^{\text{old}} - 1$ (as a particle jumps out of box 1 making N_1^{new} smaller by one), otherwise the new value of N_1 should be $N_1^{\text{old}} + 1$ (as a particle jumps into box 1 making N_1^{old} bigger by one). You should note that these last two instructions describe exactly what we did in the original marble game. Finally, we calculate x_1^{new} using the exact same formula as in $Step$ 0 (you should note that in this instruction we calculate x_1^{new} using the new value N_1^{new} and *not* the old value).

In the algorithm, it's essential that we specify whether we're talking about the new value (current step) or the old value (previous step). If you use the old value instead of the new value or *vice versa*, the algorithm will not work correctly! Mixing up references between new and old values is very common trap for new players. We'll avoid this error by always explicitly labeling variables as being "old" or "new" in algorithm steps. As we've already discussed, parameters don't need new or old labels as they don't change from step to step.

Table 2.5 Instructions for **Step 2** of the marble game (same as Step 1)

Instruction	Comment or explanation
$Step^{\text{new}} = Step^{\text{old}} + 1$	new $Step$ is the previous value plus 1

$t^{\text{new}} = t^{\text{old}} + \Delta t$	increment clock using timestep Δt
$r^{\text{new}} = \text{RANDBETWEEN}(1, N)$	randomly choose one of the N marbles (call it r^{new})
$N_1^{\text{new}} = \text{IF}(r^{\text{new}} <= N_1^{\text{old}}, N_1^{\text{old}} - 1, N_1^{\text{old}} + 1)$	if r^{new} is less than or equal to N_1^{old} jump $1 \rightarrow 2$ (N_1^{new} decreases), else jump $2 \rightarrow 1$ (N_1^{new} increases)
$x_1^{\text{new}} = N_1^{\text{new}}/N$	calculate new fraction in box 1

If you look closely at Table 2.5, you'll notice that $Step\ 2$ is identical to $Step\ 1$. The only difference is that "new" now refers to $Step\ 2$ and "old" now refers to $Step\ 1$. This did not happen by accident. When I wrote out $Step\ 1$, I made sure that the same algorithm statement would work for all the following steps $2, 3, 4\ldots$ Because $Step\ 2$ just repeats $Step\ 1$, we can write it and the remaining steps as shown in Table 2.6.

Table 2.6 Alternate instruction for Step 2, 3…

repeat Step 1 for the desired number of steps

Q.2.3 **(a)** *Write out* a complete kinetic Monte Carlo algorithm, including a unit check, to simulate a marble game system and calculate $x_1(t)$, i.e., how x_1 changes with time t, with $N = 10$, $k = 0.05\ 1/\text{s}$, and $x_0 = 0.3$.
(b) Assume that the function **RANDBETWEEN(1,10)** produces the following sequence of **fixed random numbers** r = 6, 4, 4, 7, 6, 3, 10… Using your algorithm, *calculate by hand* what happens for steps 0 through 4 and *write your answer out by hand* in the form of an **output table**.
Really long hint: Your answer to part (a) should look exactly like Fig.2.2. However, please don't just copy Fig.2.2 … the point is for you to discover if you can start out with a description of the system and then figure out the algorithm for yourself – *from scratch*. If you get stuck, use Fig.2.2 to remind yourself of what we're aiming for. While you're writing your algorithm for part (a), you should *simultaneously* fill in the corresponding entries in your output table for part (b) as shown in Table 2.7. As you write out the list of variables, you should transcribe them into your output table as headings. As you write out the instructions for step 0, you should then follow those instructions and fill in the corresponding row in the output table. Similarly, you should follow your instructions to fill in the output table row for step 1. Once that is working, you should then check to make sure that it also works correctly for the step 2 row. The point of doing parts (a) and (b) together is to check that your algorithm is working correctly – while you're writing it. Your algorithm should also include a unit check for any equations using parameters or variables that have units. In this algorithm, the only calculation involving a unit change is equation (2.2), which is used to calculate Δt from k and N. Getting the units wrong is one of the most common traps for newbies and experts alike, so always include an explicit unit check in your algorithm (Fig.2.2), even if it's not asked for.

Table 2.7 Output table for the marble game kMC algorithm

$Step$	t (s)	r	N_1	x_1
0	0	–	3	0.3
1	2	6	4	0.4
2	4	4	3	0.3

When you're done, you should carefully check that each entry in your output table matches the values in Table 2.7. **Note:** There is no random number r needed for $Step$ 0. The first random number is used in $Step$ 1, you should then use the fixed random numbers in order until you complete $Step$ 4. You don't need all the random numbers, just the first four. You should note that the heading for the time column is t (s). This exactly matches the listing t (s) in your algorithm in Fig.2.2 and Table 2.2. The part in parentheses (s) is necessary to specify the units for the numbers in the time column. If the units are missing from the column heading, then the answer is just … *wrong*!

About what you discovered: writing and testing algorithms

Writing an algorithm is *much easier* if you combine writing the algorithm with testing it. I.e., you should do parts (a) and (b) of Q.2.3 *at the same time*. You should use a **pencil** – because you'll probably have to change or rearrange something along the way. Imagine writing a cookbook. The only way to ensure that the recipe is correct (and complete) is to use the recipe to actually cook the omelet. You then taste it to make sure that it's good! ❑

Q.2.4 It's extremely important that you understand how the instruction for N_1^{new} works in the fourth line of $Step$ 1. To check that you really do understand how it works, rewrite the instruction for N_1^{new} in $Step$ 1, by *filling in the blanks* in the following:

$$N_1^{\text{new}} = \text{IF}(r^{\text{new}} > N_1^{\text{old}}, \_\_\_\_\_, \_\_\_\_\_) \tag{2.9}$$

(note the greater than sign in the **Logical_test** of the **IF()** function). Your answer should be written as a complete instruction in the style used in Fig.2.2.
Hint: You need to figure out what the **value_if_true**, and the **value_if_false** should be in the **IF()** function so that the new game rule works correctly.

Q.2.5 *Briefly explain* what would happen if you used the following instruction by mistake

$$N_1^{\text{new}} = \text{IF}(r^{\text{new}} >= N_1^{\text{old}}, N_1^{\text{old}} + 1, N_1^{\text{old}} - 1) \tag{2.10}$$

(note the greater-than-or-equal-to sign in the **Logical_test** in the **IF()** function) instead of $r^{\text{new}} > N_1^{\text{old}}$. Explain why this instruction would *not* always play the game correctly.
Hint: *Think* about how this instruction would work if $N_1^{\text{old}} = 3$ and you rolled $r = 3$. Would a marble go from box $1 \rightarrow 2$?

2.2 The spreadsheet – putting Excel to work for you

We're now going to use the algorithm that we just developed to help us write a spreadsheet to get Excel to execute the kMC sim for us. After following the procedure outlined below, my spreadsheet looked like Fig.2.3. This spreadsheet is organized into different sections. **Column A** (cells **A2:A12**, which is pronounced cells **A2** through **A12**) contains the sim parameters. **Row 1** (cells **A1:G1**)(cells **A1** through **G1**) contains table headings. **Row 2** (cells **C2:G2**) contains column headings. **Row 3** (cells **C3:G3**) contains the values (and formulas) for *Step* 0 (zero). **Row 4** (cells **C4:G4**) contains the formulas for *Step* 1. **Row 5** (cells **C5:G5**) contains the formulas for *Step* 2 etc…

	A	B	C	D	E	F	G
1	**Param$**		**Marble game kMC sim table**				
2	***N***		***Step***	***t* (s)**	***r***	N_1	x_1
3	10		0	0		3	0.3
4			1	2	2	2	0.2
5	***k* (1/s)**		2	4	4	3	0.3
6	0.05		3	6	9	4	0.4
7			4	8	10	5	0.5
8	**Δ*t* (s)**		5	10	7	6	0.6
9	2		6	12	2	5	0.5
10			7	14	10	6	0.6
11	x_0		8	16	4	5	0.5
12	0.3		9	18	10	6	0.6

Fig.2.3 Excel screenshot of the spreadsheet for the marble game kMC sim.

About what you discovered: careful reading is required!

As you may have noticed reading the preceding paragraph, you absolutely must read the instructions very carefully. If you don't have two screens you might like to print this section out. There are lots of technical terms that must be interpreted correctly. For example, **Column A** means all the cells directly under the A heading and **Row 1** means all the cells directly to the right of the 1 heading. The notation **A2:A12** is used in Excel to mean all the cells between cells **A2** and **A12** inclusive. ❑

Open the **Excel** app. on your laptop. Then *create* (*open*) a **Blank workbook** (spreadsheet). *Create* a folder your files, then *save* your spreadsheet with a systematic file name, e.g., BPM.Q.2.6.xlsx. We'll now go through what I did, so that you can enter and format your spreadsheet like mine. I'll also mention some of the tricks I've learned for entering the spreadsheet. There are lots of ways of doing things in Excel. If you prefer a different way, that's fine. All we really care about is whether the spreadsheet works correctly.

In Fig.2.3, cells **A1** and **C1:G1** contain table headings. *Type* in the text **Param$** for cell **A1**. Then *select* cell **A1** and go to the **Styles** tab and *select* the **cell style Heading 1** like I did for Fig.2.3. To

widen column **A** *double click* on the line between the **A** and **B** column labels. Then *click drag* the line between the **B** and **C** column labels, to make column **B** a narrow gap between the **parameters table** and the **sim table**. *Type* the sim table heading **Marble game kMC sim table** in cell **C1**, then *select* cells **C1:G1** and *select* the cell style **Heading 1**. In the **Alignment** tab *select* **Merge & Center** to center the sim table heading.

Type **N** in cell **A2** and *select* the **cell style** **Heading 2** like I did for Fig.2.3. Cell **A2** containing ***N*** is a **heading**. In all headings you need to *select* the **cell style** for the cells first, before you start formatting subscripts and italics etc. Note that letters that stand for numbers are formatted in italics.

Next *type* the **value** **10** for N in cell **A3**. *Type* the heading ***k* (1/s)** into cell **A5** and *type* the corresponding value **0.05** into cell **A6**. Note that the heading contains the units for the value in parentheses **(1/s)**, so that people reading the spreadsheet will know the units for k. This corresponds to the definition of k in the algorithm $k\ (1/\mathrm{s}) = 0.05$ (see Fig.2.2 and Table 2.2). *Type* the heading **Δ*t* (s)** into cell **A8**. You can insert the Greek letter Δ (Delta) using **[Insert] > [Symbols]**. Cell **A9** contains the formula $= 1/(N * k)$. Recall from the algorithm, that we used a formula for Δt so that if N or k are changed, then the value for Δt will automatically update. The algorithm formula for Δt is entered into cell **A9** as follows: *type* **=1/(** into cell **A9**, you can then *left-click* on the numerical value for N in cell **A3**. This automatically inserts **A3** into the **Formula bar**. Then *type* ***** and *left-click* on the numerical value of k. Then *type* **)** and hit the ENTER key. *Check* that you have the correct formula by *left-clicking* in cell **A9**. The **Formula bar** should now read **=1/(A3*A6)**. Don't forget the parentheses or the asterisk (I usually call it "star") – you need them!

Type the heading **x0** into cell **A11**, *format* the cell with the **Heading 2** style and then *format* the ***x*** in italics and the **$_0$** as a subscript. Then *type* **0.3** the value of x_0 in cell **A12**. When entering spreadsheets from now on, we'll put all the parameter instructions under their corresponding headings (with units) in column **A** in the format that we just learned …

Each sim variable gets its own column in the sim table. **Row 2** contains the headings for all the sim variables: $Step$, t (s), r, N_1 and x_1. *Type* **Step**, **t (s)**, **r**, **N1** and **x1** into cells **C2:G2**, then *format* them with the **Heading 2** style and then *format* them with subscripts and italics etc. as appropriate. The headings include units to indicate the units for all the numerical values in their respective column. *Type* the instructions for $Step$ 0 into cells **C3:G3** of **row 3** of the spreadsheet. For the **Step** and ***t* (s)** columns, just *type* **0** (zero) into cells **C3** and **C4**, respectively. *Leave* the cell for r empty (see Fig.2.3). For N_1, *enter* the formula from your algorithm, but refer to the numerical *values* for N and x_0 in cells **A3** and **A12**. When you're done, the formula in cell **F3** should read **=A3*A12**, and the value in cell **F3** should be **3**. For x_1 in $Step$ 0 enter the formula from the algorithm and refer to the numerical value of N_1 in the *current* row (that's what N_1^{new} means) and the value of N in the **Param$** column. When you're done, the formula in cell **G3** should read **=F3/A3**, and the value in cell **F3** should be **0.3**.

The value in the *Step* **column** tells us the step number for that entire **row** of the sim table. The algorithm formula for *Step* works just like *turn* in the original marble game. **Row 4** (cells **C4:G4**) should have the formulas for *Step* 1 listed in the algorithm. Enter the algorithm instruction for Step in cell **C4**. The spreadsheet formula that you end up with in cell **C4** should be **=C3+1**. Now enter the algorithm formula for t^{new}. You probably entered **=D3+A9**. This works just fine for *Step* 1, but it won't work correctly for *Step* 2!… let's see why…

Excel absolute cell references

The spreadsheet formula **=D3+A9** uses the standard **relative cell referencing** method. *Copy* this formula from cell **D4** to cell **D5** using the standard *left-click-drag* method discussed in **CHAPTER 1** and you'll get a value of $t = 2$ s for *Step* 2 instead of the correct value of $t = 4$ s, which is what it should be. So, what went wrong? Unfortunately, this kind of error happens quite often when we're working with Excel. So, let's take this opportunity to learn how to **debug** our spreadsheet. We've identified that cell **D5** is not working the way we wanted, so left-click in cell **D5** and then in the **Formula bar**, you should then see something similar to Fig.2.4.

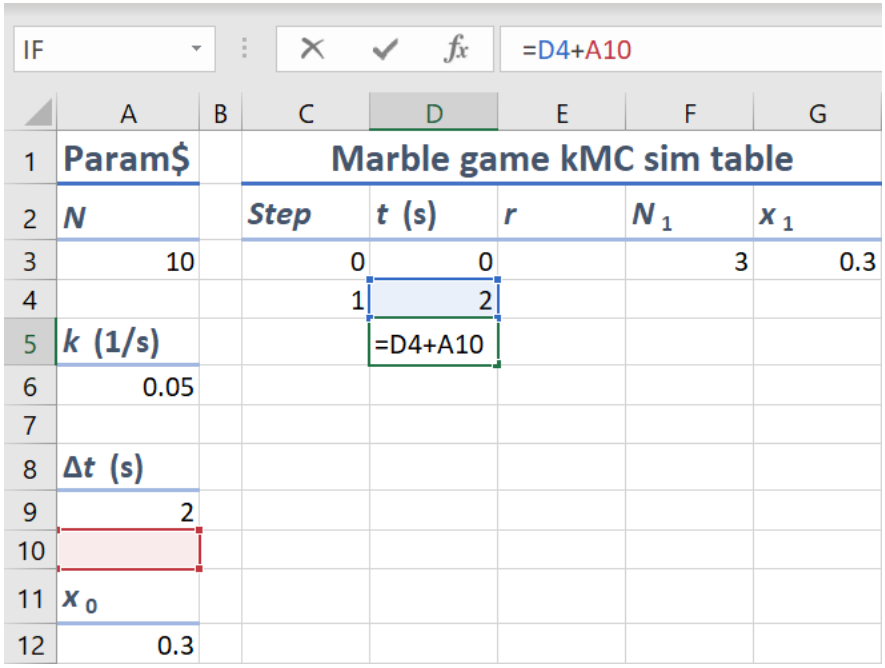

Fig.2.4 Screenshot from Excel. **Debugging** the kMC sim by using the **Formula Bar** to show the cells actually used by Excel were not the ones we intended.

If you look closely, you'll see that the formula Excel copied to cell **D5** correctly refers to the previous value of t in cell **D4**, note the blue box around cell **D4** and the blue **D4** in the **Formula bar**. However, the formula Excel copied to cell **D5** refers to **A10** and not **A9** as we intended (note the red box around cell **A10** and the red **A10** in the **Formula bar**). The reason this happened was because Excel assumed that you wanted to use a **relative reference** for Δt, i.e., "*two cells to the left and six cells down*". What we actually want to do is to "*always use cell* **A9**". To do this we need to enter **=D4+$A$9** into cell **D5**. The **$** in front of the **A** means *always use* column **A**. The **$** in front of the **9** means *always use* row **9**. **$A$9** is an **absolute cell reference** because we *absolutely*

(only) want to use **A9**. In our spreadsheet, all the parameters are in column **A**. Hence, as a general rule, any cell references that start with **A** should be absolute references. The heading **Param$** is intended to you remind you that the parameters in column **A** need absolute references using the dollar sign (not because Param is a wannabe rapper – Param Dolla $ign ☺).

Addressing modes

There are four kinds of cell addresses in Excel: **(i) A9** means the standard relative addressing for both column and row; **(ii) $A$9** means always use column **A** and always use row **9**; **(iii) A$9** means use relative addressing for the column, but always use row **9**; and **(iv) $A9** means always use column **A**, but use relative addressing for the row. On a PC, or newer Mac, you can cycle between the four choices after you've entered a cell address in a cell or in the **Formula bar,** or a dialog box, by pressing the F4 function key on the keyboard (you may need to press and hold the Fn key first). On older Macs press ⌘ + **-** + **t** (i.e., hold down the ⌘ command key and then press the **-** minus key and the **t** key). We'll use some of these choices in later chapters.

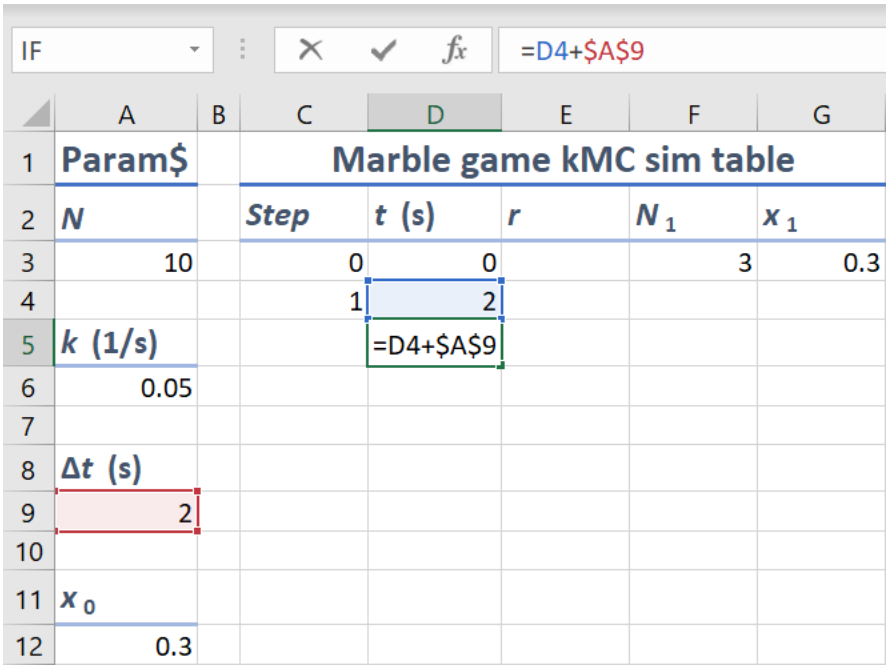

Fig.2.5 Screenshot from Excel showing that the debugged spreadsheet correctly uses **absolute referencing**.

Let's see what happens if we go back and use an absolute reference for the value of Δt in cell **D4**. *Delete* the contents of cells **D4** and **D5** then *enter* the formula for cell **D4**. As mentioned above, you can enter **$A$9** into a formula by *left-clicking* on cell **A9** and then *pressing the F4 function key*. After you're finished, the formula should read **=D3+$A$9**. If you *copy* this formula from cell **D4** into cell **D5** using the *left-click-drag* method, you'll get the desired result in cell **D5** i.e., **=D4+$A$9**. *Confirm* this by *left-clicking* in cell **D5** (after the copy) and then *left-click* in the **Formula bar**. Your spreadsheet should look something like Fig.2.5. Notice that in the copied formula, the reference to t changes to **D4** after the copy – because we entered a relative cell reference **D3**, whereas the reference to Δt does not change after the copy – because we entered an

absolute cell reference **$A$9**. Also note that by clicking in the **Formula bar**, the cells that are actually being used are highlighted by colored boxes. This is the best way to check that any formula copied by Excel is actually the that one we intended. We've now successfully **debugged** the problem in our spreadsheet.

About what you discovered: debugging references

Unfortunately, this type of error – forgetting to use absolute addressing is a fairly common problem! If you just copied a section of a spreadsheet and you get weird error messages like **#DIV/0**, **#VALUE!** or **#NUM!** then there's a good chance that they're caused by a problem similar to the one that we just fixed … Now you know what to do! ❑

Completing step 1

Okay, so now that we've figured out how to correctly refer to parameters in the spreadsheet, let's finish $Step\ 1$. Delete the contents of **D5** as they relate to $Step\ 2$ – we got a little ahead of ourselves. (Don't worry; we haven't lost anything – the same formula is still in cell **D4**.) *Enter* the algorithm instruction for r in cell **E4** … you should end up with **=RANDBETWEEN(1,$A$3)**, note the **$A$3**.

The main logic of the algorithm goes into the cell for N_1. *Enter* the algorithm instruction for N_1 in cell **F4.** Remember that while you're entering the formula you can left-click on a cell to insert its reference. After you've finished entering the formula from your algorithm, the value in cell **F4** should be either **2** or **4**. If you *left click* in cell **F4** and then in the **Formula bar**, you can check that the formula you entered is doing what you intended. The formula should read **=IF(E4<=F3,F3-1,F3+1)** …don't forget the first **=**. *Enter* the algorithm instruction for x_1 in cell **G4** … the formula for x_1 should read **=F4/$A$3** when you're done entering it.

That completes all of $Step\ 1$. To copy it, *select* all of $Step\ 1$ in your spreadsheet and *left-click-drag copy* it to make a 4-step simulation (i.e., the last row in the **Marble game kMC sim table** should start with $Step = 4$.

The first thing we want to do with this spreadsheet is to check that it's working correctly by comparing it with what we did by hand in Q.2.3.

Q.2.6 *Replace* the random numbers in steps 1 through 4 with the *fixed* random numbers r = 6, 4, 4, 7 by simply typing over the contents. Then *compare* the spreadsheet with your answer to Q.2.3 to make sure that it's working properly. When it is working correctly, *record it* in **Normal Mode** by highlighting cells **C1:G7** and copying (CTRL+C) and then **[Paste as Picture]** into MS Word.

Q.2.7 *Delete* steps $2-4$ and replace the fixed random number in $Step\ 1$ with $r^{\text{new}} = \text{RANDBETWEEN}(1, N)$. *Select* all of $Step\ 1$ and *click-drag copy* to make a 4-step simulation. *Record* the spreadsheet in **Formula Auditing Mode** (CTRL+`) by using **[Paste as Picture]** into Word.
Hint: Your answer should look something like Fig.2.6.

	A	B	C	D	E	F	G
1	Param$		Marble game kMC sim table				
2	*N*		*Step*	*t* (s)	*r*	N_1	x_1
3	10		0	0		=A3*A12	=F3/$A$3
4			=C3+1	=D3+$A$9	=RANDBETWEEN(1,$A$3)	=IF(E4<=F3,F3-1,F3+1)	=F4/$A$3
5	*k* (1/s)		=C4+1	=D4+$A$9	=RANDBETWEEN(1,$A$3)	=IF(E5<=F4,F4-1,F4+1)	=F5/$A$3
6	0.05		=C5+1	=D5+$A$9	=RANDBETWEEN(1,$A$3)	=IF(E6<=F5,F5-1,F5+1)	=F6/$A$3
7			=C6+1	=D6+$A$9	=RANDBETWEEN(1,$A$3)	=IF(E7<=F6,F6-1,F6+1)	=F7/$A$3
8	Δ*t* (s)						
9	=1/(A3*A6)						
10							
11	x_0						
12	0.3						

Fig.2.6 Screenshot from Excel showing **Formula Auditing Mode**.

Check to make sure that your last row for $Step$ etc… looks *exactly* the same as Fig.2.6. If it does, then you've got a working spreadsheet for the marble game simulation model, and you should *return* the spreadsheet to **Normal Mode** (CTRL+`). Now would be an excellent time to save a copy of your spreadsheet using the method discussed in **CHAPTER 1**.

About what you discovered: writing and debugging spreadsheets

Implementing and debugging spreadsheets from an algorithm is one of the most important skills you'll learn here. In some ways, this process is just like cranking through the algebra of a physics word problem. If you make one little mistake, the answer will be wrong, and you'll have to go back and figure out what went wrong. If you can't find the bug in a reasonable amount of time, then it's probably time to stop hitting your head against that brick wall. First go back and double check that the algorithm really is working correctly by checking it again carefully by hand. If the algorithm is correct, you should then rewrite the spreadsheet again starting *from scratch* using the algorithm by opening a **blank workbook**. As you enter each cell, check that it's working correctly and that it's producing exactly the same number you calculated in your output table. We used *fixed* random numbers in Q.2.3 and Q.2.6 so that this check will work. ❑

2.3 Approach to equilibrium – does not depend on *N*

Once you get your spreadsheet working and you've replaced the fixed random numbers with $r^{\text{new}} = \text{RANDBETWEEN}(1, N)$, *extend* your simulation to 2000 steps and *plot* $x_1(t)$, which means you should plot x_1, the fraction in box 1, as a function of time t as a **Scatter with Straight Lines** chart (no markers). x_1 should be on the y-axis and t should be on the x-axis. *Make sure* to label your axes and include units (if needed). *Press* DELETE in a blank cell about ten times to get a feel for what the $x_1(t)$ graph looks like on average (it should look similar to the original marble game). To see how changing the number of particles changes the simulation, *change* the value of the total number of particles from $N = 10$ to $N = 500$ with the jump rate constant held fixed at $k = 0.05\ \text{s}^{-1}$. Excel should automatically recalculate the timestep to be $\Delta t = 0.04$ s. You should *check* that this actually happened in your spreadsheet. Then *format* the time axis to have a fixed

Maximum Bound of $t = 80$ s and *format* the x_1 axis to have a fixed maximum bound of $x_1 = 0.7$, so that the scale of the graph remains constant between different simulations. *Set* the parameter for the initial fraction in box 1 to $x_0 = 0$ so that we start out as far from equilibrium as possible. Once you've done that, *press* the DELETE key in a blank cell about ten times to get a feel for what the graph looks like on average. Once again, you should *notice* that (on average) the graph appears to approach the same **equilibrium** value (on average).

On the graph, add a dotted line representing your best guess for this constant value… one way to do that, or to add *any* **mathematical function** to an existing graph, is to *add* a two-column table *anywhere* that's convenient on **Sheet1** of your spreadsheet. I recommend putting it in columns **I** and **J** to separate it from the main simulation table. In this case your **theory table** should look something like Fig.2.7. The heading "**Theory table**" is included to remind us that this table contains theory values. The t (s) column contains the x-values we want to plot. Cell **I3** is simply the number **0**. The cell containing **80** is produced by a formula **=D2003**, which is a reference to the last value of time t in column **D** of the spreadsheet. We used a formula so that the value of t will always match the last value of t in the $x_1(t)$ graph – even when we change N or k and hence Δt. Whenever Δt changes, the duration of the simulation in seconds also changes. The $\langle x_1 \rangle$ column contains the y-values for the line, $\langle x_1 \rangle = 0.5$, that we're plotting series (not the x-values as you might have expected). This function is a constant straight line. As a result, we only need two rows in our theory table and the two values for $\langle x_1 \rangle$ are simply the number **0.5**. However, if we wanted to plot a more complicated function, then we'd need to have enough x-values to give a smooth looking curve using a **Scatter with Straight Lines** chart. See the "Plotting functions or shapes in Excel" section in the end-of-chapter summary.

Theory table	
t (s)	$\langle x_1 \rangle$
0	0.5
80	0.5

Fig.2.7 Screenshot from Excel showing a **theory table** for equilibrium.

To add the theory table data as a **Scatter with Straight Lines** (no markers) series to the chart, *right-click* on the chart (remember Excel calls graphs "charts") and *choose* **[Select Data…] > [Add]** (**[+]** on a Mac). An **Edit Series** window then opens. For the **Series name** *enter* **equilibrium**; then *select* **[Series X values:]** and *left-click drag* on the time *values* in your new table; then *select* **[Series Y values:]** and *left-click drag* on the $\langle x_1 \rangle$ *values* in your new table. You will probably need to use **Format Data Series…** to get the desired format with no markers. You should also *change* the **[Layout]** to include a **Legend**, see the following AWYD. If you haven't already, you should also *change* the **Series name** for **Series1** to something like **kMC sim** to indicate that it's for the marble game sim.

Q.2.8 For your $N = 500$ marble game with $k = 0.05$ s^{-1} (so that $\Delta t = 0.04$ s) and $x_0 = 0$, *press* DELETE in a blank cell about ten times to get an idea of what's going on. When

you find an $x_1(t)$ graph that seems representative of what happens on average, *record it*. Remember in Word, you should **[Paste as Picture]** into the Word document.

About what you discovered: checking your graph

Before you *check* on your answer by looking in the chapter appendix, you should first *review* your own graph to make sure that it has all the key features and that they make sense. *Check* your chart title, axis titles (including units if applicable), axis labels and tick marks. Your graph should also include a **Legend** to indicate which line corresponds to the kMC sim and which one corresponds to equilibrium. After you've done that, *compare* your answer with Fig.A2.1. Your graph won't look identical because of the randomness of the kMC sim, but if yours looks significantly different then you should consider whether there is a problem with your graph.
Hint: If your graph doesn't match Fig.A2.1, but you think your spreadsheet is correct, use **[Select Data…]** or right click on the series in the chart to check that your series data (for both **X** and **Y**) start at $Step$ 0 and end at $Step$ 2000. It's important that **X** and **Y** match each other and that you don't include the headings. It surprised me (and my students) that including the headings can make the graph plot incorrectly! ❑

Once it's working correctly, *save* a fresh copy of your spreadsheet, e.g., as BPM.Q.2.8 – you'll need it again in this chapter and in **CHAPTER 3**.

About what you discovered: one sheet is enough

Important: I recommend this procedure of using and saving a separate **file** for each question. That way if you have to go back to a particular question, all you have to do is open the **file** and *voila!* There it is. In the past, I have had students try to use multiple **sheets** in a single Excel **workbook**. This *did not go well*! It's very easy to inadvertently make links between the **sheets**, which can be a nightmare to debug … because of the tangled **sheets**! Similarly, it's also a good idea to close any spreadsheets that you're not currently working on. ❑

Q.2.9 *Format* the x_1-axis in your chart to have a fixed maximum of $x_1 = 1$ and then *sample* a range of possible values of x_0 to see if the equilibrium value of x_1 depends on the starting value. *Try* x_0 values of $0, 0.3, 0.5, 0.8$, and 1.0. What can you conclude about the equilibrium value, does it depend on x_0? *Briefly explain*.

Q.2.10 **DISCUSSION QUESTION** Okay, so now that you've confirmed that the simulation always approaches an equilibrium value of $\langle x_1 \rangle = 0.5$. *Briefly hypothesize* (i.e., briefly guess) why you think the simulation actually does this.
Hint: *Think* about what the algorithm actually does. At each step, each of the marbles is relabeled. Marbles labeled 1 through N_1 are in box 1 and marbles $N_1 + 1$ through N are in box 2. The algorithm then selects any one of the N marbles with an even probability and moves it to the other box. Why does that result in $\langle x_1 \rangle = 0.5$? **BONUS POINTS** for a mathematical explanation.

About what you discovered: why ½?

From a kinetic point of view, it's relatively easy to see why $x_1 = 0.5$ is the most favored value. If the value of x_1 is less than 0.5, then the number of particles in box 2 is higher. Hence, there's a greater probability of a particle in box 2 being the next one to jump, which moves x_1 closer to 0.5 *on average*. Alternatively, if x_1 is greater than 0.5, then the number in box 1 is higher, and a particle jumps out of box 1 more often than not, bringing the system back towards $x_1 = 0.5$. The only time when there is no **bias** in the type of jump is when we are at *exactly* $x_1 = 0.5$. BTW There are quite a few ways of explaining why $\langle x_1 \rangle = 0.5$, maybe you can come up with a different one for the bonus points in Q.2.10. ❑

Q.2.11 DISCUSSION QUESTION With the jump rate constant set to $k = 0.05$ 1/s, *set* the initial fraction of particles in box 1 to $x_0 = 0$ with fixed-scale axes having a maximum of 0.7 for x_1 and 80 s for t. Then *vary* the number of marbles in the system in the range $N = 100$ to $N = 500$ and *compare* the average shapes of the kMC $x_1(t)$ curves for different numbers of marbles. Check each number N by pressing DELETE in a blank cell about ten times and *visually compare* the general shapes of the $x_1(t)$ graphs. *Briefly summarize* how the following properties change with the number of marbles in the game N:

(a) the jaggedness of the graph; and

(b) the average general shape of the graph (ignore changes in jaggedness); and

(c) By discussing the mean residence time $\tau = 20$ s, *briefly hypothesize* why the average shape of the $x_1(t)$ graph (ignoring changes in jaggedness) doesn't depend on the number of marbles N.

Hint: The average time to reach equilibrium should be related to the time needed to "randomize" the location a marble, which should happen after each marble has jumped more than once, on average.

About what you discovered: effect of number of marbles on kinetics

With fixed x_1 and t axes, you should have noticed that while the jaggedness (variability) of the curves increases with a decreasing number of marbles, the basic shape of the average curve does not depend on the number of marbles. It's important that you consider what happens on average when you look at the curves, so be sure to hit DELETE a bunch of times and look for what happens on average.

In the physical system that we're simulating, the particles are jiggled around and eventually they end up jumping to the other box because of their random thermal motion. The rate of this jiggling (and hence the jump rate k) doesn't depend on how many particles there are in solution – so long as the solution is still "dilute." That means that the mean residence time $\tau = 1/k$ does *not* depend on how many particles N there are in total. That means that every single particle leaves the box at the same rate k as every other molecule. As we'll discover in **CHAPTER 3**, it's then possible to prove mathematically that the shape of the $x_1(t)$ curve doesn't depend on N. This **particle independence** is a simple idea, but it's not like many everyday situations. For example, exiting the front door from the rear of an airplane (or a bus) can take much longer if the plane is full. The

distinction in the marble game is that the jump rate constant k is **independent** of how many marbles there are in the box, whereas on a plane you can't even start to leave until all those people at the front have deplaned! ❑

Q.2.12 With $x_0 = 0$ and the same fixed axes as in Q.2.11, change N to be twenty thousand particles. The best way to enter twenty thousand is to type **2e4** into the spreadsheet. Excel recognizes this as the number 2×10^4.
(a) *Briefly describe* what the $x_1(t)$ graph looks like.
(b) In light of what you discovered in Q.2.11, *briefly explain* why the graph looks the way it does.
Hint: The time axis has numbers on it, what are they telling you? How much time elapses in the 2000 $Step$s that you can see in your graph? In your answer, be careful to distinguish between simulation $Step$s and the physical time t – they are *not* the same thing!

About what you discovered: Microsoft messes with your stuff!

When you enter **2e4** into the spreadsheet, Excel recognized this as the number 2×10^4. Excel also automatically converted it into the Excel standard format for scientific numbers and displayed it as **2.00E+04** because Excel assumed that this was what you wanted. If you want to change it back to the default **General** format you'll need to go to the **[Home] > [Number]** group and change [**Scientific]** to [**General]**, or you can right click on the cell and select **[Format Cells…] > [Number] > [General]**. BTW The default **Scientific** format displays only two decimal places and rounds the number before displaying it. This can sometimes cause problems, to see why, *type* **20049** into the cell when it's in the scientific format. Excel displays **2.00E+04** – which is off by nearly 50. However, it's not quite as bad as it seems, if you *left-click* in the cell and then look in the **Formula bar**, you'll see that Excel is actually storing the exact number you typed in. As a result, seeing-is-not-always-believing in Excel. You may run across other examples of this type of thing as we continue to work with Excel. Sometimes a number will not fit in its cell because the cell is too narrow, and you'll see something like **###** displayed in the cell. If this happens, click on the cell, and look in the **Formula bar**. If you widen the cell, you should be able to correctly see the number stored there. ❑

With $x_0 = 0$ and $N = 20{,}000$, your $x_1(t)$ graph looks truncated. To fix that, go to the **Format Axis** window for the time- and x_1-axes and **[Reset]** the **Maximum** for the time- and x_1-axes to **Auto** and *delete* the equilibrium $\langle x_1 \rangle$ line from the graph.

About what you discovered: seeing what's in graphs clearly

The problem of an improperly scaled graph that we just fixed is one that often comes up when we're working with Excel, or any other graphing software. It is *always* up to you to *adjust* the graph axes so that you can see the data in a manner that you think is appropriate. ❑

Q.2.13 **(a)** *Briefly explain* why the rescaled $x_1(t)$ graph looks like a straight line starting at zero and pointing up to the right.

(b) *Describe mathematically* how the slope (= rise/run) is related to the jump rate constant k.
(c) *Briefly explain* if this graph represents the equilibrium behavior of the system. Look carefully at the x_1-axis. Are we even close to the equilibrium value of $\langle x_1 \rangle = 0.5$?

About what you discovered: large systems

What you just discovered applies to nearly all simulations. Let's see how it applies to one of the most important challenge problems in biophysics – protein folding. With the advent of genomics, new DNA sequences are discovered every day that code for the amino acid chains that make up protein molecules. For the protein to be biologically active, it must fold up into a useful shape. In principle, this folded structure can be predicted from the amino acid sequence using advanced computer simulations of proteins in water. However, just like in our marble game simulation, the amount of computer time required to reach equilibrium increases with the number of atoms N. The result is that this approach is not viable for most proteins because they are just too big for computers to simulate a long enough time to get to the equilibrium folded structure … An additional complication is that sometimes other proteins (chaperones) aid in the folding process.❑

2.4 Approach to equilibrium – kinetics and dependence on jump rate constant k

Jump rate constant units

Before we talk about how the jump rate constant k affects the marble game, let's talk first about its most important property – **units**! I know we've talked about this before, but I know some of my students have had trouble with this in the past … so here's another explanation of this tricky concept. **Rate constants** and **rates** tell us how frequently something happens. E.g., how often do cars cross the intersection at the entrance to campus on College Road? The answer to that isn't a rate constant, it's a **rate** – e.g., "10 cars per minute". This rate can be written as

$$R_{\text{cars}} = 10 \, \frac{1}{\text{min}} = 10 \, \text{min}^{-1} \tag{2.11}$$

I prefer the first way of writing the value of R_{cars}, with an over-under fraction $\frac{1}{\text{min}}$ for the units "per minute". This is how you should write units when you're plugging and chugging through a calculation, but you should note that scientific publishers and writers usually prefer to use the mathematically equivalent inline notation min^{-1} because it doesn't mess up their line spacing. This is common in scientific writing, so it's important that you take notice of the $^{-1}$ (negative one exponent), which means **per minute** in the units min^{-1}. It's always really important that you pay attention to units. In summary, R_{cars} does *not* have units of time. It has units of **inverse time** $\frac{1}{\text{min}} = \text{min}^{-1}$.

Q.2.14 Open the copy of the spreadsheet you saved in Q.2.8 that has *fixed axis maxima* of $x_1 = 0.7$ and $t = 80$ s. Make sure that $x_0 = 0$ and set the number of marbles to $N =$

250. Then *sample* different values of k to see the effect of this parameter on the simulation. In particular, *try* values of $k = 0.1\ \mathrm{s}^{-1}$, $k = 0.05\ \mathrm{s}^{-1}$ and $k = 0.025\ \mathrm{s}^{-1}$. Don't forget to hit DELETE in a blank cell a few times to see what happens on average.
(a) As you decrease the value of k, *briefly describe* the change you see in the graph.
(b) *Briefly describe* what's wrong with the time axis when you make $k = 0.2\ \mathrm{s}^{-1}$ or
(c) $k = 0.01\ \mathrm{s}^{-1}$?

Q.2.15 To correct the problem you discovered in Q.2.14(b)&(c), **[Reset]** the **Maximum** for the time-axis to **Auto**. Then sample possible values of k to see if the overall shape of the x_1 vs. t curve depends on the jump rate constant k. Try values in the range $k = 1 \times 10^{-9}\ \mathrm{s}^{-1}$ to $k = 1 \times 10^{9}\ \mathrm{s}^{-1}$.
Hint: the first number should be entered as **1e-9** and the second one as **1e9**.
(a) What can you conclude about the overall shape of the curve, does it depend on the value of k? *Briefly explain*.
(b) What does change? *Briefly explain*.
(c) With $N = 250$, *calculate* the length of time (duration) of a 2000-step sim when $k = 1 \times 10^{-9}\ \mathrm{s}^{-1}$, $k = 1 \times 10^{9}\ \mathrm{s}^{-1}$ and $k = 0.05\ \mathrm{s}^{-1}$.

Q.2.16 Let's use the symbol t_{sim} for the duration of the simulation in seconds and N_{steps} for the duration of the sim in $Steps$, e.g. $t_{\mathrm{sim}} = 80\ \mathrm{s}$ and $N_{\mathrm{steps}} = 2000$ in the kMC sim of Fig.A2.1.
(a) *Write out* an equation for the sim duration t_{sim} in terms of the timestep Δt.
Hint: If one step takes Δt, how long do $N_{\mathrm{steps}} = 2000$ steps take?
(b) Use the equation you derived in part (a) to *calculate* the duration of a 2000-step sim with $N = 500$ when $k = 1 \times 10^{-9}\ \mathrm{s}^{-1}$, $k = 1 \times 10^{9}\ \mathrm{s}^{-1}$ and $k = 0.05\ \mathrm{s}^{-1}$.

About what you discovered: approach to equilibrium

What you should have noticed in Q.2.15 is that the shape of the curve doesn't seem to depend on the value of k. The autoscaling feature may squish or stretch the graph in the time direction a little, but taking this into account, you should have been able to convince yourself that the basic shape does not change. One thing that does change *dramatically* is the scale of the time axis, and hence the time required to reach equilibrium. The smallest rate constant corresponds to jumps that take *decades* to happen on average, e.g., removing heavy metals like lead from bone. The largest rate constants are for the speed demons of molecular transport: **aquaporins** and **ion channels**, which are membrane proteins that transport water and ions respectively, on a timescale of $1 - 100$ *nanoseconds*. In Q.2.16 you should have discovered that the physical duration of the simulation is given by

$$t_{\mathrm{sim}} = N_{\mathrm{steps}}\Delta t = \frac{N_{\mathrm{steps}}}{Nk} \tag{2.12}$$

❑

Q.2.17 DISCUSSION QUESTION Using what you discovered in Q.2.15 and Q.2.16, *briefly summarize* in your own words the effects of the jump rate constant k on each of the following:
(a) the number of simulation steps required to reach equilibrium
(b) the amount of physical time required to reach equilibrium
(c) the duration of the simulation t_{sim} for a fixed number of simulation steps N_{steps}.

Q.2.18 DISCUSSION QUESTION In your own words, *briefly summarize* the effects of the number of particles N on each of the following:
(a) the number of simulation steps required to reach equilibrium
(b) the amount of physical time required to reach equilibrium
(c) the duration of the simulation t_{sim} for a fixed number of simulation steps N_{steps}.

Physiological applications

The marble game simulation we've been investigating is much more than the simple children's game we studied at the beginning of **CHAPTER 1**. As we discussed in **SECTION 1.5** of **CHAPTER 1**, the marble game can be used to **model** (and hence understand) a wide variety of biological systems. Similar metabolic processes to oxygen uptake, carbon dioxide removal, and glucose transport occur at membranes throughout the body with differing levels of complexity. The **two-box model** based on the marble game provides a jumping off point for our study of these biophysical systems. As you'll see, we can modify this two-box model to investigate transport of almost any molecule or ion in the body to understand a variety of **metabolic** and **transport** processes in physiology. The limitation of this approach is that the underlying mechanism for transport must be a random process. This usually means **passive** processes and *excludes* **active transport** that requires the use of stored chemical energy, e.g., in the form of ATP molecules. Also *excluded* are **convective flows** – such as breathing or blood flow. These can be distinguished from the marble game because all the molecules move together in a concerted manner – the marbles don't jump randomly between stationary boxes – in convection the boxes are moving!

Q.2.19 DISCUSSION QUESTION *List* five physiological situations that you think might be modeled using the marble game model. See if you can come up with examples that you think are interesting, or that you think are important. Specify the system by *describing*:
(a) The boxes. I.e., do they correspond to a tiny portion of cytoplasm, a whole cell, an organ, or the whole body?
(b) The identity of the particles. I.e., what molecules correspond to the marbles? Be creative; see if you can come up with some interesting and important examples where molecules move from place to place randomly.
Hint: Physiology can be described in terms of systems that exchange material with the external environment: respiratory, digestive, urinary, reproductive and integumentary (skin). There are also internal systems that coordinate or perform functions within the body: musculoskeletal, circulatory, immune, nervous, and endocrine systems. All of these

systems include numerous passive processes occurring across epithelial surfaces that might be modeled by the marble game.

2.5 Eliminating TYLENOL® from the body

In the original marble game sim, the rate of **events** (jumps) didn't change during the simulation. A marble's rate of jumping didn't depend on which box the marble was in, and the total number of marbles didn't change. Hence, the **total rate of events** λ was a *constant*. In the original marble game, λ is calculated using

$$\lambda = Nk \tag{2.13}$$

where N is the total number of marbles and k is the marble jump rate constant. As both N and k are constant parameters, equation (2.13) tells us that the total rate of events must also be a constant in the original marble game sim.

In other sims, the total rate of events might *not* be constant. For these **variable timestep simulations**, λ becomes a system *variable* because it changes during the sim. This implies that the timestep Δt is also a variable and is given by

$$\Delta t = \frac{1}{\lambda} \tag{2.14}$$

so that both λ and Δt are variables that change from step to step. The simplest possible example of this is when one of the jump rates is zero. This occurs in a simple – but surprisingly realistic – model of how some drugs are eliminated from your body.

Marble game model of drug elimination

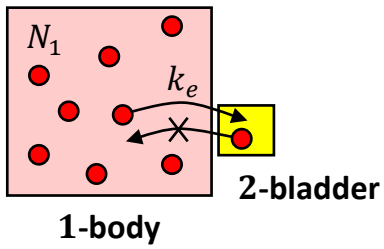

Fig.2.8 Marble game representation of the two-box model of drug elimination.

In the model system shown in Fig.2.8, box 2 represents the bladder and box 1 represents the rest of the body. The rate constant for jumps from box $1 \rightarrow 2$ is k_e, the **drug elimination rate constant**. We're using the subscript e to remind us that the rate constant is for elimination. The rate constant for jumps from box $2 \rightarrow 1$ is zero, as jumps from box 2 back into box 1 can't happen as indicated by the crossed-out arrow in Fig.2.8. In other words, once drug molecules get into the bladder they can't get back into the rest of the body. In this case, the **total rate of jumps** λ is

$$\lambda = N_1 k_e \tag{2.15}$$

λ is now a variable because it depends on N_1, which steady decreases. The specific drug we'll consider is the drug known as **acetaminophen** in the USA, e.g., the Tylenol® brand. Elsewhere it's known as **paracetamol**, e.g., the Panadol® brand.

The model shown in Fig.2.8 is extremely simplified! There's clearly much more going on than a simple "jump" of a molecule through the kidneys and ureters into the bladder. For example, acetaminophen molecules are usually modified by the liver before being eliminated. We'll leave the discussion of the physiological implications of the model to **CHAPTER 4**. For the time being, we'll focus on discovering how this idealized **model system** behaves.

Q.2.20 *Show that* equation (2.14) reduces to equation (2.2) in the *original* marble game.

About what you discovered: the purpose of "Show that…" problems

Questions that include **show that** or **derive** are asking you to write out a math answer to the problem. In this case, you need to algebraically combine the general equation (2.14) for Δt with the specific equation (2.13) for λ in the original marble game sim. According to the question, the answer should be equation (2.2). The purpose of this question is to help you *discover* the relationship between the more general way of talking about kMC sims in equation (2.14) and the original marble game, equation (2.2). To communicate that you really do understand how the connection is made, you should carefully write out your answer including all the elementary algebraic steps required to get from the starting point(s) to the desired equation. In this case you could write

$$\Delta t = \frac{1}{\lambda} \text{ and } \lambda = Nk \implies \Delta t = \frac{1}{Nk}$$

or in words

Substituting equation (2.13) into (2.14) gives $\Delta t = \frac{1}{Nk}$, which is equation (2.2).

The bottom line here is that you need to make it clear in your answer that you really do understand how equation (2.14) is a consequence of combining the relationships in equations (2.2) and (2.13).

Reminder: translating math into English

As we discussed in **CHAPTER 1**, it's essential that whenever you read a mathematical symbol in the text of this book, you should make sure that you understand what concept the symbol represents. I.e., you need to explicitly translate math into English, e.g. k into "jump rate constant" until you become fluent in using math symbols. As an example, you probably don't need to translate the texting phrase "LOL" into "laugh(ing) out loud" anymore because you're fluent in texting abbreviations, but if you came across "TIL" in a blog you might need to look it up to discover it means "today I learned".

Show that problems provide another opportunity for you to practice your math language skills – understanding the symbolic representation of key ideas (**math vocab**) and how we can use **math grammar** to combine and manipulate relationships. When you see Δt you should pronounce it "time step", λ is the "total rate of events" and N is the "total number of marbles". If you always

do the translation in your head, you'll find it much easier to remember the meaning of all those symbols… if you need a reminder, there's a complete list of symbols in **APPENDIX B.1**. ❑

Nothing random!

There is only one kind of event in this kMC sim – the jump of a Tylenol molecule from box 1 → 2. As a result, *we always know what will happen next* – a molecule will jump from box 1 → 2, making N_1 decrease by one. We also know from equation (2.14) that the average time until that jump is $\Delta t = 1/(N_1 k_e)$. This means that our kMC simulation is completely **deterministic**! Meaning that there is *no randomness* in this Monte Carlo simulation at all!

Q.2.21 **(a)** Using equations (2.15) and (2.14), *write out* a complete algorithm in a style similar to Fig.2.2 (including unit checks) for a kMC simulation of $N_1(t)$ for Tylenol elimination with an elimination rate constant of $k_e = 0.31\ \text{h}^{-1}$. Start with $N_0 = 100$ Tylenol molecules in box 1.
(b) *Calculate* steps 0 through 4 of your algorithm *by hand*, i.e., using a calculator. *Write* your answer in the form of an output table.
Really long hint: You should do parts (a) and (b) of this question together. Your answer to part (b) will help you figure out what to write in part (a). For example, $Step\ 0$ should describe how to fill in row 0 of the table and $Step\ 1$ should describe how to fill in row 1. Your sim algorithm should include the following *parameters*

- $N_0 = 100$ Number of Tylenol molecules in the body at time 0
- $k_e\ (1/\text{h}) = 0.31$ Rate constant for elimination (per hour)

… and the following *variables*

- $Step$ Counter for the number of simulation steps
- $\lambda\ (1/\text{h})$ Average rate of jumps (per hour)
- $\Delta t\ (\text{h})$ Timestep (in hours) **Note:** Δt is a variable in this algorithm
- $t\ (\text{h})$ Time since the dose was taken (in hours)
- N_1 Number of Tylenol molecules in the body (box 1)

When you're done, your answer for steps 0 through 2 should look *exactly* like those shown in Table 2.8. **Note:** No time elapses during $Step\ 0$.

Table 2.8 Output table for the marble game kMC algorithm of drug elimination

$Step$	$\lambda\ (1/\text{h})$	$\Delta t\ (\text{h})$	$t\ (\text{h})$	N_1
0	–	–	0	100
1	31	0.032258	0.032258	99
2	30.69	0.032584	0.064842	98

Even if you think you got the algorithm correct, check out the following AWYD (after you're done) …

About what you discovered: if you're having trouble…

If you're having trouble writing out the algorithm, try answering the following questions first.

1. If there are N_1^{old} Tylenol molecules in box 1, at what rate λ^{new} do they jump to box 2?
2. If the total rate of events is λ^{new}, what is the equation for Δt^{new} the timestep associated with the current jump?
3. If there are N_1^{old} Tylenol molecules in box 1 in the previous step, what is the equation for the number N_1^{new} at the end of the current step of the sim?

In this kMC simulation, λ and Δt are variables. The overall rate λ^{new} at which molecules jump from box $1 \rightarrow 2$ in the current step is $\lambda^{\text{new}} = k_e * N_1^{\text{old}}$ and the timestep for the current jump is $\Delta t^{\text{new}} = 1/\lambda^{\text{new}}$. The new value of time is $t^{\text{new}} = t^{\text{old}} + \Delta t^{\text{new}}$, and the new number of molecules is always $N_1^{\text{new}} = N_1^{\text{old}} - 1$ as a single molecule always leaves box 1 during each step of this sim. ❑

Q.2.22 *Implement* your algorithm in a blank spreadsheet and check that you get exactly the same results that you calculated by hand in Q.2.21(b). *Add* steps (**rows** in Excel) to *extend* your sim in time until you reach $N_1 = 0$. Then plot $N_1(t)$, i.e., a graph of the number of molecules in the body versus time, as a **Scatter with only Markers** series. You should put N_1 on the y-axis and t on the x-axis, until you reach $N_1 = 0$. You should then **Format Data Series… > Series Options > Marker > Fill** to **No fill**. That will let you see the close-together markers better. *Record your graph*.

Q.2.23 **(a)** By *inspecting* the numbers in your spreadsheet, *calculate* how long (in hours) it takes for the number of molecules N_1 to go from $100 \rightarrow 50$?
(b) *Calculate* how long it takes for the number of molecules to go from $50 \rightarrow 25$?
(c) *Calculate* the average of your answers to parts (a) and (b).
Hint: Use the *sim numbers*. In addition, you should remember that numerical answers must always include units.

As we'll discover in **CHAPTER 4**, **pharmacokinetic** theory predicts that our **model system** should have an **exponential decay** for the amount of Tylenol as a function time

$$N_1 = N_0 e^{-k_e t} \tag{2.16}$$

Theory table		
	t (h)	N_1
δt_{theory} (h)	0	100
5	5	21.2248
	10	4.50492

Fig.2.9 Excel screenshot of a **theory table** for plotting equation (2.16). δt_{theory} is a parameter for this table.

Add a two-column **theory table** to your spreadsheet (see Fig.2.9). This table should have a heading "Theory Table". This heading will distinguish the t and N_1 in this table from those listed in the **simulation table**. Under the table heading there should be one column for time t (h) and another column for N_1. The entries in the first row should be $t^{\text{new}} = 0$ and $N_1^{\text{new}} = N_0 * \text{EXP}(-k_e * t^{\text{new}})$. The following rows should be calculated using the instructions shown in Table 2.9, where $\delta t_{\text{theory}} = 5$ h is an adjustable parameter for the theory table that specifies the spacing between times in the theory table. We've used the lowercase Greek letter δ (small delta) to distinguish it from the sim uppercase Δt (big delta t) and because – as we'll discover – this δt needs to be "small", so you should have a couple of hundred rows in the table. In Fig.2.9, I've put δt_{theory} next to the table. In each row of the theory table, a theoretical value of N_1 is calculated using equation (2.16). Note that **EXP()** is the Excel function for the **exponential function** e^x. Also note, we remembered to use a $*$ for multiplications in Excel. Also see the "Plotting functions or shapes in Excel" section in the end-of-chapter summary.

Table 2.9 Instructions for a **theory table** to plot a theory curve using equation (2.16)

Rows 1, 2, 3…	Comment or explanation
$t^{\text{new}} = t^{\text{old}} + \delta t_{\text{theory}}$	increment time by δt_{theory}
$N_1^{\text{new}} = N_0 * \text{EXP}(-k_e * t^{\text{new}})$	calculate N_1^{new} using equation (2.16)

Q.2.24 *Add* a **Scatter with Straight Lines** series to your graph for the theory using a value of $\delta t_{\text{theory}} = 5$ h and *confirm* that your time axis has a maximum value of $t = 20$ h.
(a) *Briefly explain* what's wrong with your theory series.
(b) *Change* the theory series to the banned style **Scatter with Smooth Lines**. *Briefly explain* why this **Smooth Lines** curve is extremely misleading.
(c) *Change* the theory series back to **Scatter with Straight Lines** and *decrease* δt_{theory} until the theory curve looks like a smooth function. You should end up with at least 200 t_{theory} values in your theory table. *Record* the value of δt_{theory} that you found.
Hint: You'll need to extend the **series** in your graph to show all the new points in your theory table.
(d) *Record your graph*.

About what you discovered: using Excel to plot functions

Fig.2.10 shows what your answer to Q.2.24(d) should look like. You should have noticed that the kMC sim actually reaches zero Tylenol molecules at a finite time of about 17 hours, whereas the theory still has about half a molecule left at this time. Which one is correct?

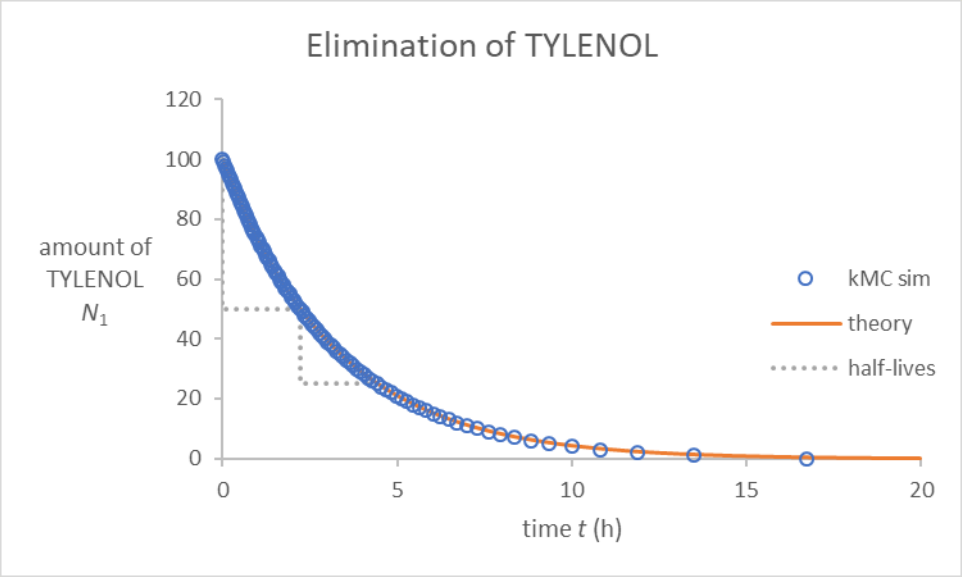

Fig.2.10 Excel chart of drug elimination. Open circles show the kMC results for an $N_0 = 100$ sim, the solid line shows the pharmacokinetic theory prediction from equation (2.16) and the dotted line indicates the first two half-lives.

In my graph, I also added a dotted line series to show the first two half-lives, using the numbers from Q.2.23. I did that by adding a **half-life table** to the spreadsheet (see Fig.2.11). I collected the spreadsheet data for the graph *by hand*. To make the stepped shape you need to have two staggered points for each t and N_1. The first two rows make the first vertical drop at time $t = 0$, the second and third rows make the horizontal tread of the first step and the third and fourth rows make the second vertical drop and the fourth and fifth rows make the second horizontal tread. The take-home message is that using a two-column table of data you can make any shape you like in Excel – see Q.2.25.

Half-life table	
t (h)	N_1
0	100
0	50
2.21991	50
2.21991	25
4.423933	25

Fig.2.11 Excel screenshot of a half-life table for plotting half-lives as a step graph.

❑

Q.2.25 You can use the method discussed in the previous AWYD to plot *any* mathematical function in Excel using an **X Y (Scatter)** chart. You can also use it to make fun shapes. With just two series, I made the square "have a nice day" smiley face shown in Fig.2.12. *Open* a blank Excel spreadsheet and – using a data table and line and/or marker series – *draw* a smiley face or another line drawing of your choice and *record* your artwork.

Fig.2.12 Excel chart made using two series. The eyes were made with a "Scatter with only Markers" series. The rest of the figure was made with a single "Scatter with Straight Lines" series. FYI if you leave a blank row in the table you get a gap in the straight-line chart. Inspired by the Bon Jovi album and song "Have a Nice Day".

Q.2.26 *Change* the y-axis of your graph from Q.2.24 to a **log scale**, and *set* the maximum on the time axis to 8 hours. *Add* a step graph for the half-lives you discovered in Q.2.23 and *record* your **semi-log graph**.
Hint: In the following AWYD there's a sample graph in the desired format. To display your graph in this format, you'll need to select the **Format Axis... > Axis Options** for the N_1-axis and check the box for **Logarithmic scale**. It's also a good idea to add "**(semi-log)**" to the **Chart Title** to let readers know from the title that the chart is a **semi-log graph**.

About what you discovered: exponential decay and half-lives

Your $N_1(t)$ graph for Q.2.26 should look something like Fig.2.13, but you weren't asked to plot the experimental data for TYLENOL liquid or caplets, and you were only asked for two half-lives. The title makes it clear that the chart is a **semi-log graph**. As shown in the graph **legend**, the blue circles represent the kMC results, the solid orange line is the theoretical curve, equation (2.16), and the grey dotted lines indicate the times required for the number of molecules to halve and then halve again and again. The experimental data and kMC results are shown as **only Markers** to make it clear that they are discrete data. As you can see, the kMC simulation (circles) and the theory (solid line) are almost identical.

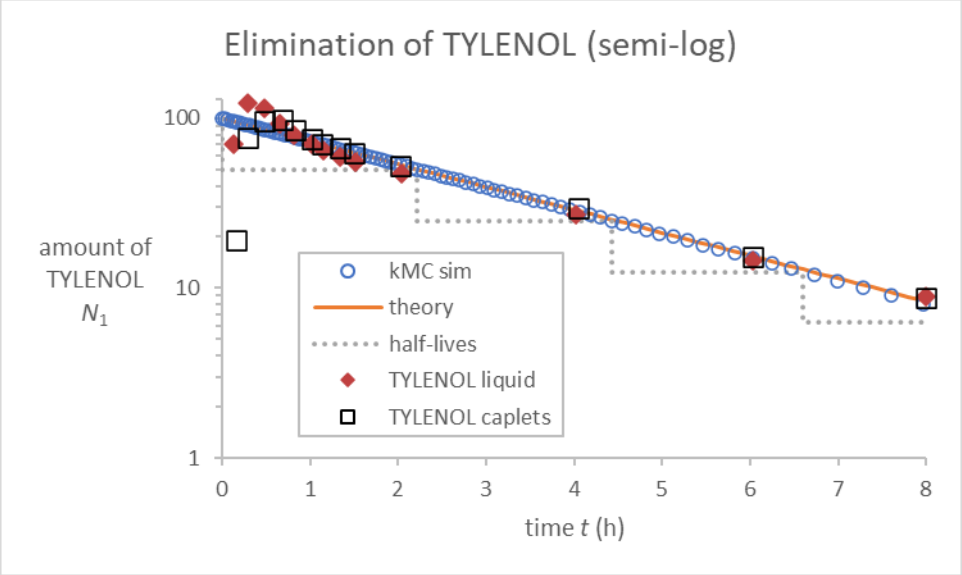

Fig.2.13 Excel semi-log chart showing of drug elimination with clinical data from tylenolprofessional.com. Note that the experimental data are discrete and hence are plotted as **only Markers**. The theory curve represents a continuous function that is plotted as **Straight Lines**. Marble game kMC data are also discrete and are plotted as **only Markers**.

My graph also includes some experimental (clinical) data points derived from the mean blood plasma concentrations of acetaminophen in 24 male subjects following oral administration of 1000 mg of acetaminophen dosed as either 30 mL of Extra Strength TYLENOL Liquid or as two Extra Strength TYLENOL Caplets (tylenolprofessional.com). The clinical data appear to match the model predictions fairly well after about an hour, indicating that our model provides a reasonable explanation of Tylenol elimination after the first hour. You should also note that the half-life steps appear to be uniform (the same size) in this semi-log plot. ❑

Q.2.27 *Compare* your answers to Q.2.23(a) and Q.2.23(b).
Hint: The following AWYD explains how to compare numbers quantitatively.

About what you discovered: talking numbers

Percentage comparisons

Q.2.27 asks you to compare two numerical values. This is an important part of technical discussions. As a scientist, engineer, or medical professional, you should be able to talk about numerical comparisons in a clear and concise manner. Your answers to Q.2.23(a) and Q.2.23(b) should have been 2.2199 h for the first half-life and 2.2040 h for the second half-life, respectively. You could compare these numbers by simply reporting them both as I just did, but that leaves the comparison and its significance up to the reader. A good answer to Q.2.27 is:

A.2.27 ***The two half-lives are almost the same. The second half-life is 0.7% less than the first half-life.***

So how did I calculate the 0.7%? Technically this is a **percent change**. The formula for a percent change from a to b is

$$\Delta\% = \frac{b-a}{a}\left(\frac{100\%}{1}\right) \tag{2.17}$$

For example, imagine that a major brand of aspirin costs $a = \$10.00$ and a bottle of generic aspirin costs $b = \$7.00$. Most people have no trouble in calculating that there is a 30.0% savings for the generic. Check for yourself that equation (2.17) gives this answer: $\Delta\% = -30.0\%$. **Note:** The *minus sign* means that the generic is 30.0% *less than* the major brand. Hence, there is a 30.0% decrease (**percent decrease** or savings) switching to the generic. You should also notice that the term in parentheses in equation (2.17) is simply a unit conversion, one over one, as $100\% = 1$.

The order of comparison is important in equation (2.17). For example, if we use equation (2.17) to calculate the **percent increase** when switching from the generic to the major brand, we get $+42.9\%$. The reason for the larger magnitude (42.9% *vs.* 30.0%) is because of the smaller reference value ($7.00 *vs.* $10.00). To avoid any ambiguity, the language of your answer should make clear which price, the generic or the brand name, is being used as the reference value a, see A.2.27 above.

Equation (2.17) is also used in science when we want to compare an observed value o from a simulation or an experiment with theoretical or expected value e. In this case the equation becomes

$$\Delta\% = \frac{o-e}{e}\left(\frac{100\%}{1}\right) \tag{2.18}$$

and the result is often called the **percent error**. The word "error" in this comparison implies that we are comparing an observed value with an expected value. However, the word "**error**" is unfortunate. The "error" can be non-zero even when there was no mistake made in the experiment or the simulation because the variation is due to normal random fluctuations, like in in the marble game.

On the internet, and in some textbooks, you will also find a comparison that is unfortunately called **percent difference** (or **percentage difference**).

$$\%\text{Diff} = \frac{|a-b|}{(a+b)/2}\left(\frac{100\%}{1}\right) \tag{2.19}$$

This equation can be used to compare two experimental (or simulation) values, but the reference value that is used is the average $(a+b)/2$.

To avoid confusion with this quantity in your answers to "*compare*" questions in this book, or in your later technical writing, I recommend that you always write your answer in the form of a

complete sentence that explicitly states the reference value used for the comparison – see sample answer A.2.27 above.

You might find the preformatted spreadsheet BPM.Ch02_Talking_numbers.xlsx useful when you answer "*compare*" questions in this book.

Factor comparisons

Percentage comparisons really only make sense if the percentages are less than about 100%. If the differences are large, it makes more sense to use a **factor** to compare the two values. For example, if an epi-pen used to cost \$50.00 and it now costs \$500.00, then we can say that the new price is ten-times (or a factor of 10.00 larger than) the old price. If the epinephrine in the injector costs \$1.00, then the epi-pen costs 500-times as much as the drug it delivers. ❑

Q.2.28 *Briefly discuss* how your estimate for the half-life of Tylenol in Q.2.23(c) compares with the published values of the half-life of Tylenol for acetaminophen and paracetamol, i.e., look up the half-life of acetaminophen and paracetamol on the web or in a drug fact book.
Hint: Your answer to this type of question should quantitatively compare your value with the literature value. As always you should cite references. If you find a range of published values, state how your value compares with the range, e.g., does it fall within the range?

Alternate kMC sim of drug elimination

An alternate way to implement the marble game model of drug elimination in a kMC sim is to modify the original marble game by changing the instruction that changes N_1 in $Step\ 1, 2, 3, \ldots$ to

$$N_1^{\text{new}} = \text{IF}(r^{\text{new}} <= N_1^{\text{old}}, N_1^{\text{old}} - 1, N_1^{\text{old}}) \tag{2.20}$$

in which the second choice in the IF is changed from $N_1^{\text{old}} + 1$ to N_1^{old}. I.e., N_1^{new} never increases.

Open the copy of the spreadsheet you saved in Q.2.8, **[Reset]** the x_1-axis to **Auto**, *remove* the equilibrium line, *set* the rate constant to be 0.31 h^{-1}, *change* the rate constant symbol from k to k_e for elimination, *set* the initial fraction in box 1 to $x_0 = 1$, *set* the total number of marbles to $N = 100$ and *modify* the instructions for N_1 according to equation (2.20). You'll also need to *change* the units for time from seconds (s) to hours (h) in the spreadsheet headings and in the $x_1(t)$ graph. *Set* the time axis **Maximum** to 20 h. Then *add* a **theory table** (like the one shown in Fig.2.9) to the spreadsheet for the theoretical prediction of $x_1 = x_0 \exp(-k_e t)$ using the techniques you learned in **SECTION 2.5** and *plot* the theory as a dotted line together with the kMC values of x_1 as a function of time. Press DELETE in a blank cell several times to get an idea of what the sim does on average.

Q.2.29 **DISCUSSION QUESTION** **(a)** *Briefly explain in words* what the new kMC simulation is actually doing and *briefly explain* where the randomness comes from.

Hint: A good answer will *compare* this simulation with the kMC sim in Q.2.24.
(b) With $N = 100$, and the x_1-axis set to **Auto**, *change* the initial fraction in box 1 to $x_0 = 0.1$. *Press* DELETE in a blank cell at least ten times and *record* a representative $x_1(t)$ graph.
(c) In your answer to part (b), the 20 h of time in the graph corresponds to 620 kMC $Steps$. By considering what's happening to the 100 marbles in the game, *briefly explain why* there are level portions in the graph followed by abrupt falls in x_1.
(d) *Sample* values of N in the range from $N = 10$ to $N = 500$ and *briefly describe* how the graph depends on N.
(e) *Set* $x_0 = 1$ and sample values of N in the range from $N = 10$ to $N = 500$. *Briefly explain why* the graphs look like part (d).

Conclusions – about what you discovered

Congratulations! If you made it here, then you've successfully learned how to write an algorithm, test it, and then implement it in a spreadsheet. An algorithm is useful for formalizing *any* procedure and then communicating it to others. (I even found one on the internet for cardiac rehabilitation.) Algorithms are also a key step in writing more complex computer applications, e.g., in genomics, bioinformatics, or engineering. Algorithm development is one of the most important skills that you'll learn in this book! If you can write an algorithm, you can then get a spreadsheet to do the number crunching for you. BTW you should note that questions like Q.2.3-Q.2.5 and Q.2.21 make good test questions, because they test your ability to write and understand algorithms.

In **SECTION 2.1**, we learned how to write an algorithm and test it by hand. We wrote an algorithm for the original marble game and identified **parameters** and **variables**. Parameters like k, N and Δt *do not change* during the marble game sim whereas variables such as t, r, N_1, and x_1 *do change*, usually at every step. As we learned in **SECTION 2.1**, it's best to **(b)** test the algorithm by filling in an **output table** while you're **(a)** writing it. It's also important to remember that whether something is a parameter or variable depends on the sim. For example, in the TYLENOL sim, Δt became a variable because it changed at each step of the sim.

In **SECTION 2.2** we learned how to implement our pretested algorithm in a spreadsheet. Parameters all go in **column A** of the spreadsheet and are referred to using **absolute addressing** in the rest of the spreadsheet. Each of the variables gets its own **column** in the spreadsheet with each **row** representing the current step. As we discovered, writing spreadsheets does not always go well the first time. If you do run into problems, you'll have to **debug** the spreadsheet using the **Formula Bar** and **Formula Auditing Mode** and carefully compare what the spreadsheet is actually doing with what you planned in your **algorithm** and then tested in your **output table**.

In **SECTION 2.3** we discovered how the sim approaches equilibrium, and how that approach doesn't seem to depend on the number of particles N, i.e., the average shape of the N_1 vs. t curve does not depend on the number of marbles. The physical explanation is that the jiggling due to

Brownian motion, and hence the rate of jumps, does not depend on how many particles there are in a box. As a result, the mean residence time τ, the average time between jumps of a specific molecule, does *not* depend on how many there are in total. This means that the marbles move **independently** of each other, which is why the mean residence time doesn't depend on how many marbles there are in the box. The behavior of single molecules is now an extremely active area of biophysics. Sometimes the variability of this behavior is critical to understanding how small systems behave. We will return to this fascinating topic again in later chapters.

In **SECTION 2.4** we learned that parameters make the simulation real. The rate constants $k = 40\ \mathrm{s}^{-1}$ for CO_2 removal and $k_e = 0.31\ \mathrm{h}^{-1}$ for TYLENOL elimination were chosen to match the physiological systems we were considering. For physical systems, the **jump rate constant** k is *the* parameter that determines the amount of physical time required to reach equilibrium. As we learned, this time could range from nanoseconds up to decades depending on the system represented by the simulation. This **equilibration time** does not depend on N. However, when we implement the system in a kMC sim, the amount of **physical time** represented by the kMC sim does depend on N – see equation (2.12). These two effects should not be confused. The physical time that elapses is not determined solely by the **number of steps** in the sim N_{steps}, it also depends on the rate constant k and number of marbles N. **Hint:** For this discussion to make sense – you have to *look at* equation (2.12).

Finally, in **SECTION 2.5** we generalized the marble game model to allow for the jump rates to be different. By setting $k_2 = 0$, we developed an extremely simple model of how drugs are eliminated from the body. The concept of a drug half-life *originates* from this model, as it predicts an exponential decrease in the amount of drug in the body, so that the drug half-life doesn't depend on the dose. The half-life of a drug is an important factor in determining the dosage **frequency** of drugs. In **CHAPTER 4** we'll investigate drug elimination in more detail and discover how to fit the model to experimental data and obtain the value of k_e that matches the real situation, e.g., the value of $k_e = 0.31\ \mathrm{h}^{-1}$ used in Fig.2.13. Our marble game model of drug elimination is the simplest possible model. We are ignoring nearly all of the fantastically complicated intricacies of human physiology, but the result is an excellent **first-order approximation** (starting explanation) for the elimination of the drug TYLENOL.

As you've probably noticed already, reading this book is not like reading pulp fiction! You can't just skim read it and expect to get the main plot details. **Active reading** is required! As you read, you should be constantly asking yourself: What are we doing? Why are we doing it? Does this make sense? The questions are included in the main body of the text to help you discover what the models predict and what they imply. You should always try to work through the question before reading any related AWYD. If you discover things for yourself… you'll understand them much better! If you have a **problem** with any of the instructions, the most likely reason is that you missed something that was explained earlier. **Solution:** Check out any related AWYD and then go back and reread the relevant sections. Pay particular attention to **definitions** of **parameters** and **variables** and the explanations of the equations in which they appear. Whenever you read a symbol like k_e, it should be pronounced "the drug elimination rate constant". If a thing is given a

symbol, then it must be important! Make sure you *recall what a symbol stands for* when you read it.

We also learned how to use Excel to plot a theoretical curve using any mathematical function or any other numerical data. The process is very simple and is *always the same* – make a two-column **table** where you enter the **X** and **Y** you want to plot.

In this chapter we learned how to use algorithms to develop quantitative models and implement them in a spreadsheet. The most fundamental parameter in the quantitative marble game model is the jump rate constant k. In addition to telling us how frequently molecules jump between the boxes, it also determines the mean residence time τ of an individual molecule and the size of the simulation timestep Δt. By modifying the algorithm for the original marble game in two different ways we were able to develop simple kMC models of drug elimination and discovered that they predict the exponential decay and half-life that you'll find in traditional pharmacokinetic theory.

Reading and writing algorithms are essential components of our scientific literacy. We'll be using algorithms extensively in later chapters.

References

http://www.tylenolprofessional.com/pharmacology.html (2009)
Nelson, P. H. (2013) *Greek letters go green!* https://circle4.com/biophysics/videos/

Summary: Algorithms and pain relief

Key computational concepts

Algorithm

- An **algorithm** is a human-readable recipe for implementing the model in a spreadsheet. It is a list of ingredients separated into **parameters** and **variables**, and a list of instruction **steps**.
- **Parameters** – go in column **A** and don't change during the sim. If you change a parameter, the whole sim changes… Don't forget to include **units**!
- **Variables** – have a **column** of their own with a heading including the variable name and **units**. Variables usually change at each step. Don't forget to include **units**!
- **Step 0** – a set of **instructions** for the **initial values** of the variables (first **row**)
- **Step 1** – a set of instructions for how the variables change in later **rows**
- **old** & **new** – indicate **variables** in previous and current step as **old** and **new**

Addressing modes (press F4)

- **Relative addressing** e.g., **=C3+1** keeps the **relationship** the same after a copy, e.g., *add one to the cell above*
- **Absolute addressing** e.g., **+$A$9** always refers to the **same cell** after a copy i.e., *always add cell* **A9**

Debugging (press CTRL+`)

- When you see **#DIV/0**, **#VALUE!** or **#NUM!** in your spreadsheet, something has gone wrong (probably with the addressing during a copy). To check an individual cell, left click in the **cell** and then in the **Function bar**. To check the entire spreadsheet, use **Formula Auditing Mode** (CTRL+`). If you see **###** in a cell, the contents are too wide to be displayed properly – widen the column by double-clicking on the vertical line in the heading
- Enter large numbers in Excel as **2e4** for 2×10^4

Plotting functions or shapes in Excel

- By entering a two-column **theory table** in your spreadsheet you can plot any shape you like in an Excel **Scatter with Straight Lines** chart. To make a mathematical function look **smooth** (rather than **chunky**) you need to make δx small enough. ($\delta x = 1$ is way too big in the $y = A\sin(x/L)$ example shown in Fig.2.14 and the graph is too **chunky**).

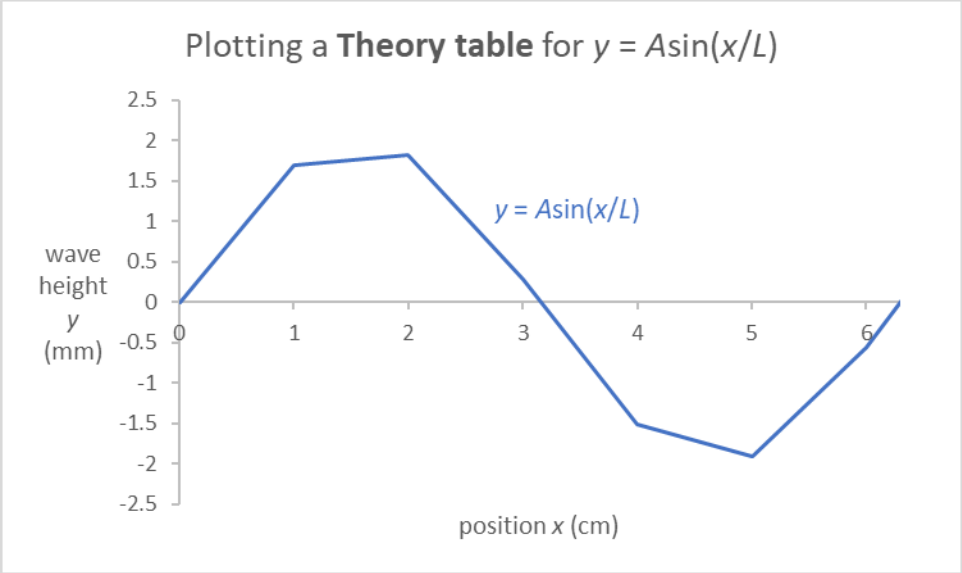

Fig.2.14 Excel chart plotting a theory table for the function $y = A\sin(x/L)$, with $A = 2$ mm and $L = 1$ cm. The value of $\delta x = 1$ cm used in the $y(x)$ graph is too big. The parameter δx should be reduced to provide a smooth curve. Download the preformatted spreadsheet BPM.Ch02_Theory_table.xlsx.

Key physiology concepts

Kinetic Monte Carlo (kMC) simulation of the marble game

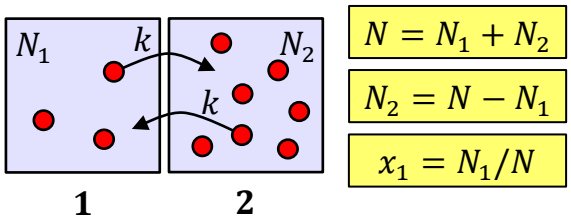

Fig.2.1 Marble game representation of the marble game kinetic Monte Carlo (kMC) simulation. The particles move just like in the original marble game, but the jumping rate is now characterized by a rate constant k (repeated from main text).

- The **jump rate constant** k [=] s^{-1} provides the timescale for the **kMC sim**.
- The **mean residence time** $\tau = 1/k$ is the average time between jumps of a particular particle.

$$\tau = \frac{1}{k} \tag{2.4}$$

- The **timestep** Δt is the time between jumps of any marble (irrespective of which marble jumps). Its $\Delta t = \tau/N$ or

$$\Delta t = \frac{1}{Nk} \tag{2.2}$$

Equation (2.2) means that the amount of physical time associated with each step in the kMC sim depends on both N and k. **Note:** The number of steps is *not* the same as physical time.

- The **fraction of marbles in box 1** is defined by

$$x_1 \equiv \frac{N_1}{N} \tag{2.6}$$

It's an **intensive property** that can be used to compare systems with different numbers of marbles N.

- N is the **total number of marbles** in the system, which is related to the number in each box by the **bookkeeping equation**

$$N = N_1 + N_2 \tag{1.1}$$

- At **equilibrium** $\langle x_1 \rangle = 0.5$ *on average* independent of k, N or x_0.
- The **variability** of the x_1 vs. time curve depends on the number of marbles N, with small systems showing greater variability.
- The **average shape** of the $x_1(t)$ curve *does not depend* on N or k – it only depends on x_0 the initial fraction in box 1.
- The **duration** of the sim is determined by

$$t_{\text{sim}} = N_{\text{steps}} \Delta t = \frac{N_{\text{steps}}}{Nk} \tag{2.12}$$

Marble game model of TYLENOL® elimination

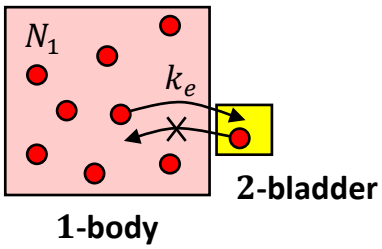

Fig.2.8 Marble game representation of the two-box model of drug elimination (repeated from main text).

- Tylenol molecules are **eliminated** from the body with a **rate** $k_e N_1$ for **jumps** from box $1 \rightarrow 2$. Jumps in the reverse direction from box $2 \rightarrow 1$ *can't happen*.
- The **total rate of jumps** λ is given by

$$\lambda = N_1 k_e \tag{2.15}$$

- where, once again, N_1 is the number of molecules in box 1
- k_e is the **drug elimination rate constant** (jump rate constant for jumps from box $1 \rightarrow 2$)

- the time between jumps (**events**) is

$$\Delta t = \frac{1}{\lambda} = \frac{1}{N_1 k_e} \tag{2.14}$$

- In this sim, both λ and Δt are simulation **variables** that change at every step.

Exponential decay and half-life

- Use **EXP()** for e^x
- A **semi-log graph** of N_1 vs. t yields a **straight line** as the model predicts an **exponential decay** in N_1.

$$N_1 = N_0 e^{-k_e t} \tag{2.16}$$

- The concept of a drug **half-life** *originates* from this model. The idea is that the time required for half to disappear (half-life) doesn't depend on the starting amount. Many real drugs exhibit this type of behavior (approximately).

Key scientific concept

Numerical comparisons

- Use **percent change** $\Delta\%$ when comparing two similar numbers.
- Use a **factor** when comparing numbers that differ by a factor of 2 or more.
- In your comparison, explicitly state the reference value as part of a sentence. E.g., the generic aspirin is 30% less than the major brand.
- You might find the preformatted spreadsheet BPM.Ch02_Talking_numbers.xlsx useful when you answer "*compare*" questions in this book.

Appendix - Answer graphs

Figure A2.1 – Checking your graph

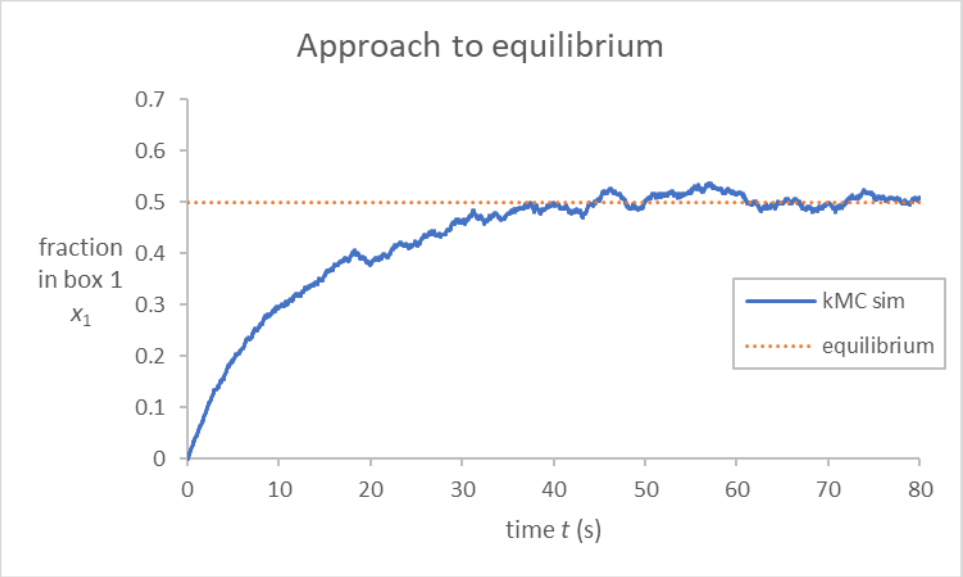

Fig.A2.1 Excel chart of an $N = 500$ marble game kMC simulation with $k = 0.05\ \mathrm{s}^{-1}$. The dotted line indicates the expected equilibrium value. **Note:** Both $x_1(t)$ curves are of the **Scatter with Straight Lines** type. Remember – **Smooth Lines** are banned!

Return to main text

In coming transmission…

…algorithms making your **head ache**?

…TYLENOL® the pain reliever hospitals use most!

This material is based upon work supported by the National Science Foundation under Grant Nos. 0836833, 1817282, and 2306506. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.