BIOREPS Problem Set #1

Switching dimensions: beating the diffusion speed limit

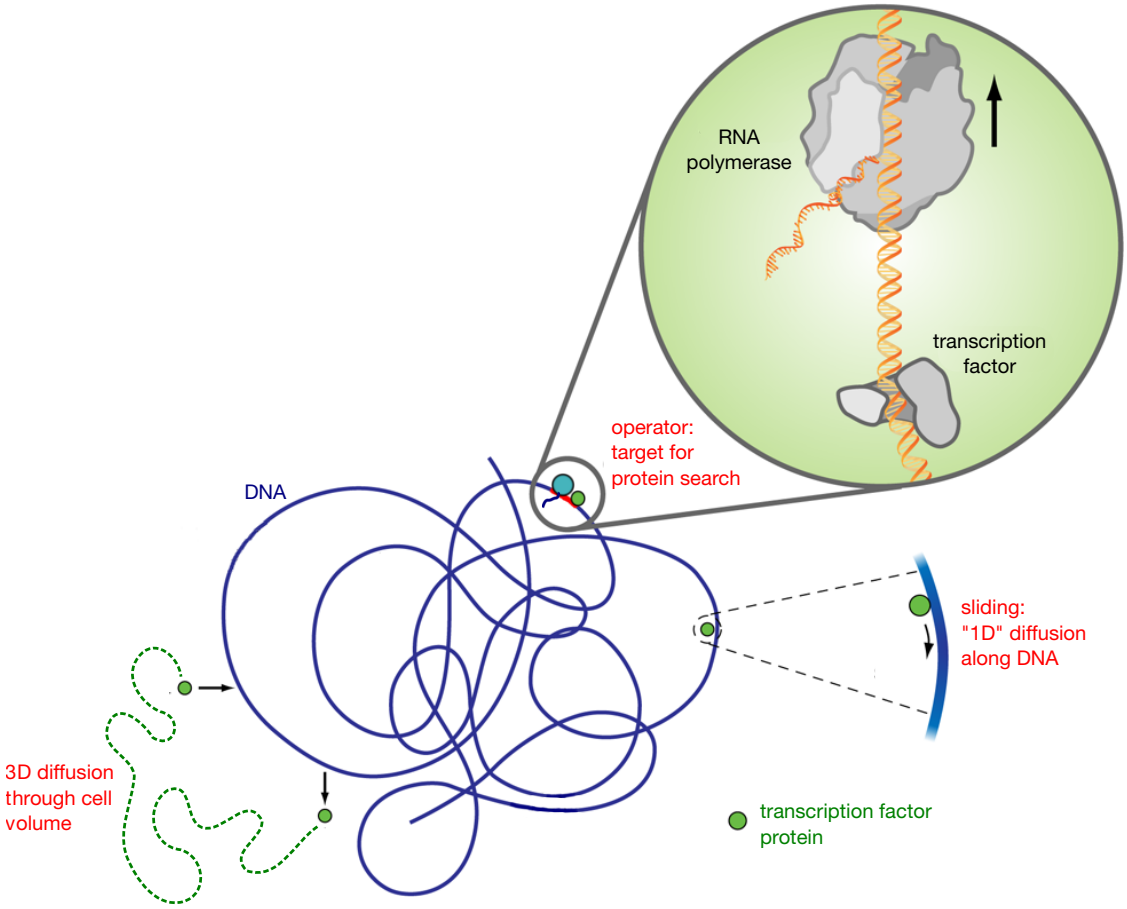

Figure 1: Proteins on the hunt for their target on DNA. Illustration adapted from Alan Stonebraker at: http://physics.aps.org/articles/v2/36.

1 Background

In order for any biological processes to take place, molecules need to find each other in the crowded, thermally agitated interior of the cell. Our first encounter with this problem will be one of the central questions of genetic regulation: how can a protein known as a transcription factor (TF) quickly locate and bind to a specific target sequence on DNA? This is quite literally a needle-in-a-haystack problem: finding a single, unique string of ~ 10 base pairs on a coiled strand of DNA that could be millions of base pairs or longer. Once the TF finds its target, it can have one of two effects: (i) some TF's are *activators*, meaning that they enhance the expression of genes downstream from the target by recruiting RNA polymerase to the target. The polymerase is a molecular machine that transcribes DNA into RNA, and the RNA will eventually be translated into the proteins encoded by the genes. (ii) On the other hand, some TF's are *repressors*, meaning

that they prevent RNA polymerase from binding and transcribing the genes near the target, thus downregulating gene expression.

One of the most widely studied transcription factors is the *lac* repressor protein (LacI) of *E. coli*. These bacteria have the ability to digest the sugar lactose when it is present in the environment. However, if they find themselves in a situation without lactose, LacI allows them to turn off the genes encoding the protein enzymes that metabolize lactose, thus conserving energy by not producing unnecessary proteins. About ten copies of the repressor protein are always present in the cell, which is normally sufficient to keep the lactose metabolism genes turned off. If *E. coli* wants to switch lactose digestion back on (when lactose becomes available again), it converts some of the lactose into a form that binds LacI and prevents its association with DNA. This is the canonical example of a so-called *genetic switch*, and in fact it was the first one discovered, described in a landmark paper in 1961 by François Jacob and Jacques Monod, future Nobel laureates in medicine [1]. (The same paper predicted the existence of messenger RNA, the product of RNA polymerase reading the DNA, which was soon thereafter confirmed experimentally.)

This paper set off an explosion of research to identify and characterize protein transcription factors and their interactions with DNA. Surprisingly, the unassuming *lac* repressor maintained its place in the spotlight: not content to be merely first, it turned out to be also uncannily *fast*. About a decade after the discovery of LacI, Riggs and coworkers measured the rate at which LacI associated with its target, an astonishing $k \sim 10^{10}$ M^{-1} s^{-1}. If this was solely due to three-dimensional diffusion—the protein wandering randomly through the cell until it collides with the tiny bit of DNA that is the target sequence—we could use the Smoluchowski reaction rate equation to estimate the corresponding value of D, the protein diffusion constant:

$$k_{\text{Smol}} = 4\pi DR$$

Here R is the radius of the target, $R \sim 1$ nm, which translates to an apparent $D \approx 1300$ μm^2/s. This is an exceedingly large diffusion constant, essentially impossible for a protein inside the cell: it is at least ten times larger than the average D for proteins in water, and more than a hundred times larger than the typical value for proteins in the crowded cell interior, $D \sim 10$ μm^2/s. To make the result more concrete, imagine a single LacI protein and a single target in the *E. coli* cell volume, $V \approx 1$ μm^3. The average time from the protein to first reach that target by three-dimensional diffusion alone is

$$\tau_{\text{3D}} = \frac{V}{4\pi DR},$$

assuming the initial separation of the two is much larger than R. The difference between $D \approx 10$ and 1300 μm^2/s is the difference between having a search time of 8 s versus 0.06 s. Given that D cannot be 1300 μm^2/s, how can LacI achieve a short search time such as 0.06 s?

If we accept that $D \approx 10$ μm^2/s for the protein and that $\tau_{\text{3D}} = 8$ s, clearly three-dimensional diffusion alone (what we will call the "3D strategy") is insufficient to explain the experimental results. So the protein must employ another strategy. It turns out that transcription factors have regions on their surface that are positively charged, and when these regions come within a few nanometers of the negatively-charged backbone of DNA, there is an electrostatic attraction. This attraction is strong enough to keep the protein within the vicinity of the DNA for some time, but weak enough that the protein can slide back and forth randomly along the DNA contour, which effectively looks like a diffusive random walk on a one-dimensional track defined by the DNA chain. If the protein does find its target sequence, hydrogen bonding interactions can form in

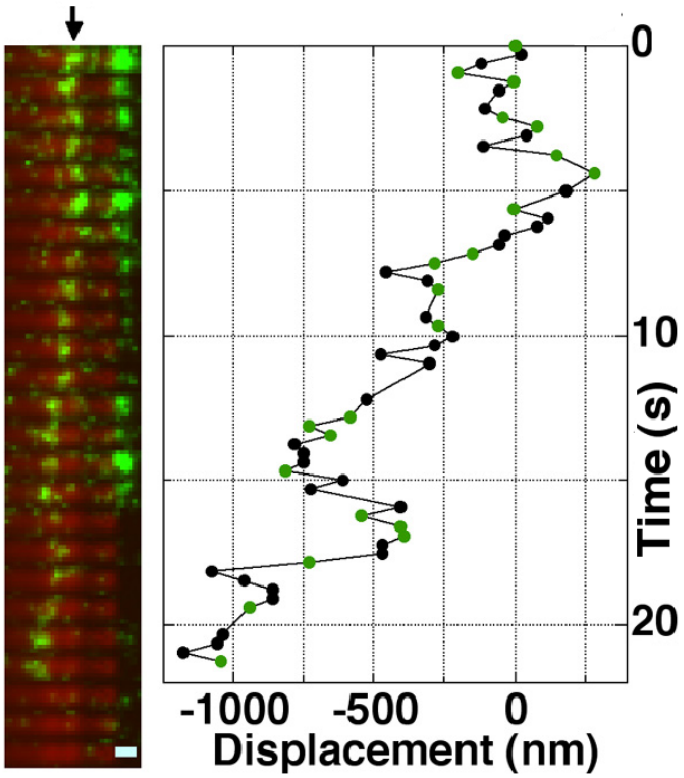

Figure 2: Experimental evidence of 1D protein diffusion on DNA, taken from Ref. [5]. The blurry red lines represent a segment of DNA, labeled by red fluorescent markers. Each row from top to bottom is a snapshot at a particular time, indicated by the scale on the right. The left green dot, highlighted by the arrow at $t = 0$, is a fluorescently labeled LacI protein (the right green dot is a labeled reference point). The protein jumps onto the DNA, diffuses randomly to the left and right for some time, and then jumps off. The motions of the protein can be converted to the trajectory shown on the left. Hundreds of these trajectories can be gathered and analyzed to calculate a sliding diffusion constant D_{slide} for the *lac* repressor.

addition to electrostatic attraction, which lock the protein strongly into place and stop the 1D diffusion.

What if the protein was always electrostatically attached to the DNA? We could call this the pure "1D strategy" of target search, since the protein never detaches to wander through the full 3D volume of the cell. There is a problem with this strategy as well: sliding along the DNA is an extremely slow process, with a diffusion constant $D_{\text{slide}} \approx 0.01\ \mu\text{m}^2/\text{s}$, two orders of magnitude smaller than a free protein diffusing through the cell interior. As you will calculate below, the search time $\tau_{1\text{D}}$ for the 1D strategy as a result is also much larger than the experimentally observed value. You will find that $\tau_{1\text{D}}$ is in fact comparable to $\tau_{3\text{D}}$. Even though being confined to one dimension restricts the number of possible paths you can take to find the target, the slow diffusion on the 1D track counteracts this potential advantage.

So we have two strategies that do not quite work, and are about equally slow. What about adopting a mixture of the two? The protein spends some time diffusing on the DNA, but it can also dissociate, wander across the cell, and reattach to another part of the DNA. This process

is repeated until the target is found. By tuning the balance of 1D and 3D diffusion just right, it turns out we can achieve target search times that are smaller than either of the two strategies alone. In other words, we can beat the 3D (and the 1D) diffusion speed limits! This somewhat counterintuitive result was laid out in a classic paper by Berg, Winter, and von Hippel in 1981. In this problem set, we will prove a simplified version of their argument, following the elegant derivation of Mirny *et al.* [4]. This mixed 1D+3D strategy has come to be known as *facilitated diffusion*.

Though the broad outlines of the theory are well established, figuring out the physical details of the search process continues to be an active research area. The remarkable thing about the Berg-Winter-von-Hippel model is that theory predicted a physical mechanism in 1981 that could not be directly verified by experiment for another 31 years. It was a pure triumph of mathematical intuition. But the major question remained: was the 1D+3D strategy actually used by cells to speed up the search process? Only in the last decade have experimentalists been able to directly confirm that proteins can indeed slide along DNA: by attaching a fluorescent label to individual proteins and then monitoring their diffusive motion, they could watch single proteins jump onto a strand of DNA, diffuse for a certain time along its contour, and then jump off. One example of this, taken from a 2006 study by Wang *et al.* [5], is shown in Fig. 2. Finally, in 2012 Johan Elf and collaborators were able to observe facilitated diffusion of LacI in a living cell [6], the end of an experimental saga that began four decades earlier.

Note: All references are available in the Resources section of the course website.

References

[1] Jacob, F. and Monod, J. Genetic regulatory mechanisms in the synthesis of proteins. *J. Mol. Biol.* **3**, 318–356 (1961).

[2] Riggs, A. D., Bourgeois, S. and Cohn, M. The Lac repressor-operator interaction: 3. Kinetic studies. *J. Mol. Biol.* **53**, 401–417 (1970).

[3] Berg, O. G., Winter, R. B. and von Hippel, P. H. Diffusion-driven mechanisms of protein translocation on nucleic acids: 1. Models and theory. *Biochemistry* **20**, 6929–6948 (1981).

[4] Mirny, L., Slutsky, M., Wunderlich, Z., Tafvizi, A., Leith, J. and Kosmrlj, A. How a protein searches for its site on DNA: the mechanism of facilitated diffusion. *J. Phys. A: Math. Theor.* **42**, 434013 (2009).

[5] Wang, Y. M., Austin, R. H. and Cox, E. C. Single molecule measurements of repressor protein 1d diffusion on DNA. *Phys. Rev. Lett.* **97**, 048302 (2006).

[6] Hammar, P., Leroy, P., Mahmutovic, A., Marklund, E. G., Berg, O. G. and Elf, J. The lac repressor displays facilitated diffusion in living cells. *Science* **336**, 1595–1598 (2012).

2 Questions

1D diffusion of a protein on DNA

To explore facilitated diffusion, we will use a combination of analytical and numerical approaches. Let us first focus on characterizing pure 1D diffusion of a protein sliding along the DNA, without detachment. Consider a single strand of DNA of length L. We will represent the position of the protein on the DNA by x, where $0 \leq x \leq L$. For our analytical calculations we will treat x as a continuous variable, but for computations it is more convenient to discretize x as $x = ia$, where $i = 1, \ldots, N$ and $N \equiv L/a$. For bacterial DNA, the true L may be on the order of mm, but we will set $L = 1000$ nm to make the numerics faster. We choose $a = 1$ nm, making $N = 1000$. The DNA in bacteria is a closed loop, so $x = 0$ is equivalent to $x = L$. In simulations, this means that when the protein jumps from $i = N$ to $i = N + 1$, the program should relabel $N + 1$ as $i = 1$. (And similarly $i = 0$ becomes $i = N$.)

a) Write a program (in your favorite language) to simulate diffusion of a single protein along the DNA, starting from an initial state $i = i_0 = N/2$. The script should be in the form of a loop, from time 0 to a certain end time t in steps of $\delta t = 10^{-5}$ s. In each run of the loop, the system updates the state i according to the following dynamics:

$$\begin{aligned} &i \to i + 1 \text{ with probability } w\delta t \\ &i \to i - 1 \text{ with probability } w\delta t \\ &i \to i \qquad \text{with probability } 1 - 2w\delta t \end{aligned}$$

The transition rate w is related to the sliding diffusion constant by $w = D_{\text{slide}}/a^2$. Use the value $D_{\text{slide}} \approx 0.01\,\mu\text{m}^2/\text{s} = 10^4\,\text{nm}^2/\text{s}$. Please see the *Computational Hints* section below for additional information on how to implement a simulation of random dynamics. Note that in these types of algorithms, δt has to be chosen small enough that the transition probabilities above are all between 0 and 1. That is why we set δt to such a small value of 10^{-5} s. (We can make δt even smaller, but our program would take longer to run.) To check that your program works, calculate the mean squared displacement (MSD) $\langle \Delta_i^2 \rangle_t = a^2 \langle (i - i_0)^2 \rangle_t$ for several different values of t. Keep the range of t small enough that the protein is not able to reach $i = 1$ (or equivalently $i = N + 1$). For each value of t, calculating the MSD involves running several hundred trajectories of length t, determining $(i - i_0)^2$ for the final state i in each trajectory, and then taking the average. Plot $\langle \Delta_i^2 \rangle_t$ versus t, and compare it to the theoretical prediction $\langle \Delta_i^2 \rangle_t = 2D_{\text{slide}}t$. If the numerics and theory do not agree, check that you have used enough trajectories to get convergent averages.

b) Let us model a 1D search for a target site on the DNA. Make the target site $i = 1$ (equivalent to $i = N + 1$), and modify your program from part a) in the following way: choose the starting position i_0 at $t = 0$ randomly from the possible range 1 to N. Then run the program until either $i = 1$ or $i = N + 1$ is reached. Record the time t when this happens. Repeat this procedure several hundred times to find the average time it takes to reach the target from a random position on the DNA. This average time is what we called τ_{1D} above.

c) Does the numerical result of part b) make sense? To check this, let us solve the problem analytically. In class, we saw that in the continuum approximation we can write down the following

equation for the average time $\tau(x)$ to go from a starting position x to our target site:

$$D_{\text{slide}} \frac{d^2\tau(x)}{dx^2} = -1.$$

The boundary conditions are $\tau(0) = 0$ and $\tau(L) = 0$, since $x = 0$ (equivalent to $x = L$ for the DNA loop) is our target location. Solve this equation to find $\tau(x)$. (Make sure the boundary conditions are satisfied!) Since $\tau(x)$ is the average time from a particular starting point x, we need to average this over all possible starting locations. In the continuum limit, this average is just an integral over x, divided by L, the range of possible x:

$$\tau_{\text{1D}} = \frac{1}{L} \int_0^L dx\, \tau(x).$$

Find an analytical expression for τ_{1D}. Plug in the parameter values, and you should get the same numerical answer as in part b), within the margin of statistical errors. If everything checks out, breathe a sigh of relief: physics works!

1D+3D search strategy

Now we can start exploring whether it is possible to get search times which are smaller than τ_{3D} and τ_{1D}. Modify the search algorithm in part b) in the following way: introduce a new parameter γ, which represents the transition rate at which the protein dissociates from the DNA (when it is at some point on the DNA that is not the target). We will not actually simulate the 3D diffusion realistically, because this would be too computationally intensive for a problem set. But we an include it in a semi-realistic way by defining something called a "3D jump" (more on that below). The program in part b) should now simulate the following dynamics:

$$\begin{aligned}
&i \to i+1 \text{ with probability } w\delta t \\
&i \to i-1 \text{ with probability } w\delta t \\
&\text{3D jump with probability } \gamma\delta t \\
&i \to i \qquad \text{with probability } 1-(2w+\gamma)\delta t
\end{aligned}$$

What happens when the program chooses to make a "3D jump"? Physically, the protein dissociates, wanders through the cell volume, and then randomly finds another part of the DNA where it can attach. From the point of view of our program, this means that every time a 3D jump occurs, i is set to a random integer in the range from 1 to N. Of course it takes some time for this 3D diffusion to occur. Though in reality this time may vary from jump to jump, as an approximation we will assume there is a fixed duration $\tau_{\text{3D jump}}$ for each 3D jump. This duration is related to τ_{3D}, the average time it takes for the protein to find the target site by 3D diffusion. Keep in mind that the 3D jump is less specific: it can land on *any* of the N parts of the DNA. Since there are N possible "targets" for the 3D jump, the search time should be N times smaller than the time for a single target. Hence $\tau_{\text{3D jump}} = \tau_{\text{3D}}/N = 0.008$ s. In your program, every time a jump occurs, add $\tau_{\text{3D jump}}$ to the current value of t. You still have the regular loop that updates t in steps of δt for all the other dynamical transitions, but the 3D jumps are the exception where the time step becomes $\delta t + \tau_{\text{3D jump}}$.

d) Let us call $\tau_{\text{1D+3D}}(\gamma)$ the average search time to the target for a certain value of γ. You already know the result for $\gamma = 0$, since this is just what you found numerically in part b), where the protein can never detach to make a 3D jump. In other words, $\tau_{\text{1D+3D}}(0) = \tau_{\text{1D}}$. You also know the result when $\gamma \to \infty$, because this just means the protein does not spend any time sliding on the DNA (it instantaneously detaches if not at the target site), so $\tau_{\text{1D+3D}}(\infty) = \tau_{\text{3D}}$. What about intermediate values of γ? Use your new program to find $\tau_{\text{1D+3D}}(\gamma)$ for $\gamma = 1\ \text{s}^{-1}$, $10\ \text{s}^{-1}$, $100\ \text{s}^{-1}$, and $1000\ \text{s}^{-1}$. You should see that $\tau_{\text{1D+3D}}(\gamma)$ first decreases and then increases with γ, and the minimum is smaller than either τ_{1D} and τ_{3D}. You have just numerically demonstrated facilitated diffusion!

Note: The minimum value of $\tau_{\text{1D+3D}}$ you find numerically is not quite the same as the experimental value of 0.06 s quoted above. This reflects the fact that we tailored the parameters to make the computations faster, and the simplifications in our approach do not completely capture the physics of facilitated diffusion in the cell. However the qualitative result that a speed-up can occur with a mix of 1D and 3D diffusion is clearly evident.

Is there a clever analytical argument to justify the numerical results of the previous part? Let's break down the search process in the following way: the protein spends some time t_{slide} on average sliding along the DNA, and then dissociates and does a 3D jump, which lasts $t_{\text{3D jump}}$. Thus one slide+jump routine lasts $t_{\text{slide}} + t_{\text{3D jump}}$ on average. Let's say the probability of finding the target during one slide is ρ. Depending on how big or small ρ is, the protein must repeat the slide+jump routine on average M times before finding the target (the smaller the ρ, the larger the M). Putting everything together, the average search time should be:

$$\tau_{\text{1D+3D}} = M(t_{\text{slide}} + t_{\text{3D jump}}). \tag{1}$$

The problem is we do not know what t_{slide}, ρ, and M are. Let's determine these one by one.

e) The average time spent sliding along the DNA, t_{slide}, clearly must depend on the dissociation rate γ (larger γ means shorter t_{slide}). To figure out t_{slide}, let us introduce a simple two-state model for the protein: it is either *on* the DNA, or *off* the DNA (in this model we don't care where exactly the protein is located). We are only interested in the detachment process, so we have a single probability rate γ for going from *on* to *off*. Using the escape time formalism introduced in class, what is the average escape time to get from *on* to *off*. This is what we will call t_{slide}, since it gives us how long the protein stays attached to the DNA before it detaches.

f) The probability ρ is related to how many sites the protein visits on the DNA before detaching. The larger the range of exploration, the higher the probability ρ that the target is in that range. During one round of 1D diffusion on the DNA, the range of exploration over time t_{slide} is proportional to the root-mean-squared displacement (RMSD) $\sqrt{\langle \Delta_i^2 \rangle_{t_{\text{slide}}}}$ of the protein from the beginning of the 1D diffusion round. So the range of exploration must be $c\sqrt{\langle \Delta_i^2 \rangle_{t_{\text{slide}}}}$, where c is some unknown constant of proportionality. Thus,

$$\rho \approx \frac{c\sqrt{\langle \Delta_i^2 \rangle_{t_{\text{slide}}}}}{N}$$

or in other words ρ is approximately the fraction of sites explored during the interval t_{slide} when

the protein is on the DNA. Using the theoretical prediction for the RMSD, and the result for t_{slide} from part e), write down an expression for ρ.

g) If the chance of success in each slide+jump is ρ, on average how many tries M does it take to find the target? (This is a standard probability problem: for example, if you have a six-sided die, on average how many times do you have to roll the die before you get a six?)

h) Plug the results of parts e), f), and g) into Eq. 1 for $\tau_{\text{1D+3D}}$. You now have an analytical expression for $\tau_{\text{1D+3D}}$ as it varies with γ. Find the γ at which this expression becomes minimum. This is the optimal rate of detachment to get the best search time. Compare your analytical prediction to the numerical results of part d). What is the rough value of the unknown constant c that makes the analytical model agree with the numerical values?

If you have reached this point, congratulations! Had you done this exercise six years ago, you could have gotten it published (see Ref. [4]).

3 Computational Hints

Several tricks are useful when designing the numerical programs in the problem set. The first is how to choose between different possible outcomes if you are given a list of probabilities. For example, let us say there are three events that can happen (as in part a), where the probabilities of those events are p_1, p_2, and p_3, with $p_1 + p_2 + p_3 = 1$. To choose a particular event, generate a random number r between 0 and 1. If $r \leq p_1$, then event 1 happens. If $p_1 < r \leq p_1 + p_2$, then event 2 happens. And if $p_1 + p_2 < r \leq p_1 + p_2 + p_3 = 1$, then event 3 happens. By using this procedure, you guarantee that the events will occur with frequencies proportional to their probabilities. For example if p_2 is tiny, the range between p_1 and $p_1 + p_2$ is tiny, so it is unlikely for r to fall there, and hence event 2 happens infrequently. This approach can be generalized to any number of possible events, so long as you know the probability of each event.

The second trick that comes in handy is to be able to choose a random integer between 1 and N (for example when assigning a random value to the start position i_0, or assigning a random i after the 3D jump). To do this, generate a random number r between 0 and 1. Set i to be ceil(Nr), where ceil(x) is the standard ceiling function, the smallest integer greater or equal to x.