Chapter 6

Multistrand : Output and Analysis

We have now presented the models and algorithms that form the continuous time Markov process simulator. Now we move on to discuss the most important part of the simulator from a user's perspective: the huge volume of data produced by the simulation, and methods for processing that data into useful information for analyzing the simulated system.

How much data are we talking about here? Following the discussion in the previous chapter, we expect an average of $O(N)$ moves per time unit simulated. This doesn't tell us much about the actual amount of data, only that we expect it to not change drastically for different size input systems. In practice this amount can be quite large, even for simple systems: for a simple 25 base hairpin sequence (similar to Fig 5.5D), it takes 4,000,000 Markov steps to simulate 1s of real time. For an even larger system, such as a 4-way branch migration system (Fig 5.5C) with 108 total bases, simulating 1s of real time takes 14,000,000 Markov steps.

What can we do with all the data produced by the simulator? In the following sections we discuss several different processing methods.

6.1 Trajectory Mode

This full trajectory information can be useful to the user in several ways: finding kinetic traps in the system, visualizing a kinetic pathway, or as raw data to be passed to another analysis tool.

Trajectory mode is Multistrand's simplest output mode. The data produced by this mode is a trajectory through the secondary structure state space. While many trajectories could be produced for a given system, for most analysis purposes discussed in this section

we are only concerned with a single trajectory. Similarly, these trajectories are infinite but unfortunately our computers have only a finite amount of storage so we must cut the trajectory off at some point.

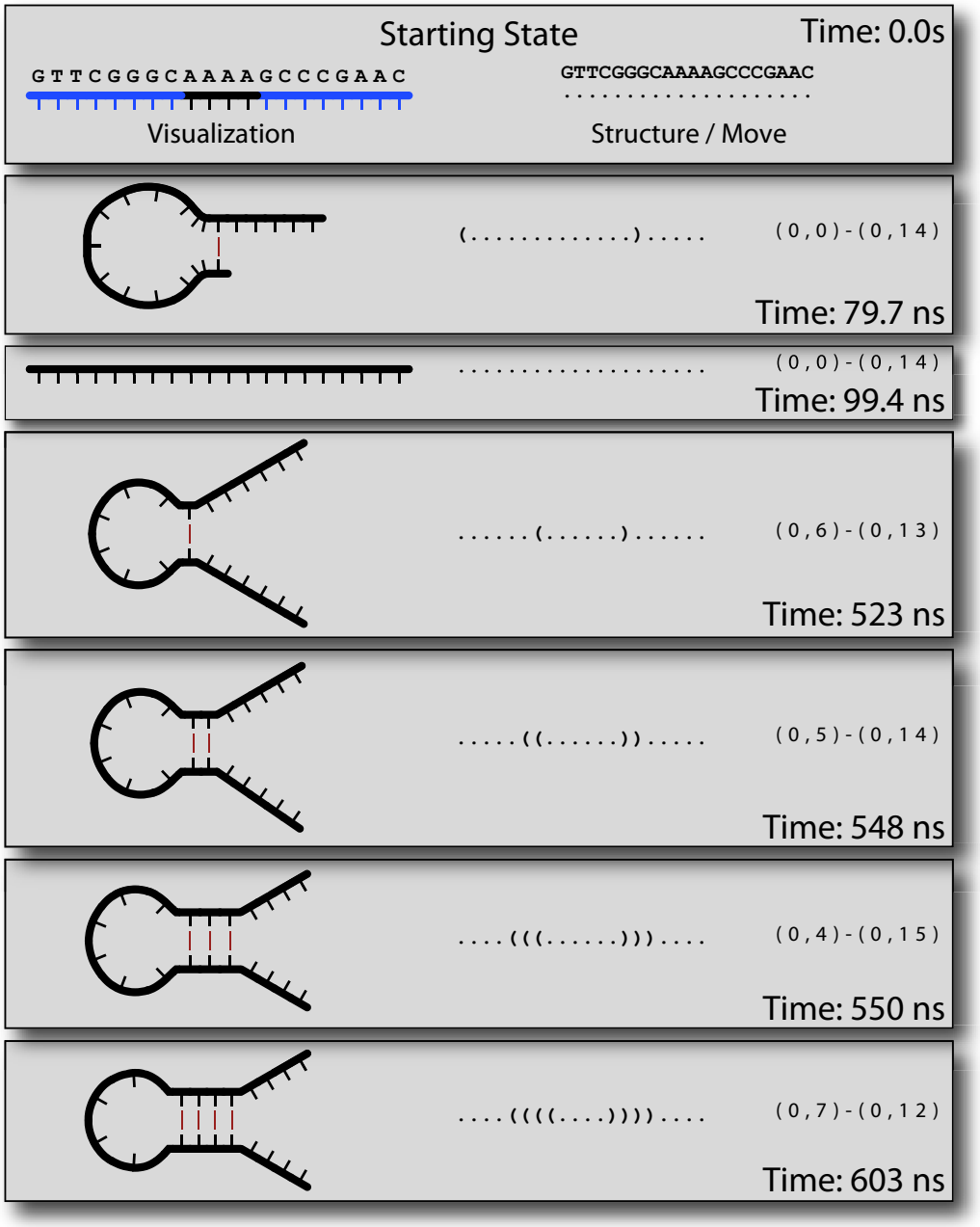


Figure 6.1: Trajectory Data

A trajectory is represented by a finite ordered list of (s, t) pairs, where s is a system

microstate, and t is the time in the simulation at which that state is reached. We call this time the *simulation time*, as opposed to the *wall clock time*, the real world time it has taken to simulate the trajectory up to that point. There are many different ways to represent a trajectory, as shown in Figure 6.1.

For practical reasons, we set up conditions to stop the simulation so that our trajectories are finite. There are two basic stop conditions that can be used, and the system stops when any condition is met:

1. Maximum simulation time. We set a maximum simulation time t' for a trajectory, and stop when the current simulation state (s, t) has $t > t'$. Note that the state (s, t) which caused the stopping condition to be met is not included in the trajectory, as it is different from the state at time t'.

2. Stop state. Given a system microstate s', we stop the trajectory when the current simulation state (s, t) has $s = s'$. This type of stopping condition can be specified multiple times, with a new system microstate s' each time; the simulation will stop when any of the provided microstates is reached.

We will now use an example to show how trajectory mode can be used to compare two different sequence designs for a particular system. The system is a straightforward three-way branch migration with three strands, with a six base toehold region and twenty base branch migration region, shown below (Fig 6.2).

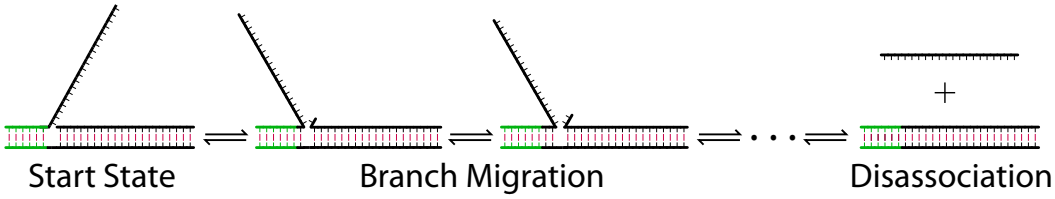


Figure 6.2: Three way branch migration system. The toehold region is in green, and the branch migration region is black. A few intermediate states along a sample trajectory are shown, with transition arrows indicating not a single base-pair step but a pair of steps that break one base-pair then form another. Many possible side reactions also exist, such as breathing of duplex regions and sequence depedent hairpin formation within the single-stranded region.

The simulation is started in the shown **Start State** using a toehold sequence of `GTGGGT` and a differing branch migration region for which we use the designs in Table 6.1. We then start trajectory mode for each design, with a stop condition of 0.05 s of simulation time, and save the resulting trajectories.

	Branch Migration Region
Design A	`ACCGCACGTCCACGGTGTCG`
Design B	`ACCGCACCACGTGGGTGTCG`

Table 6.1: Two different branch migration sequences

Rather than spam the interested reader with several thousand pages of trajectory printouts, since there are $5*10^6$ states in a 0.05 s trajectory for this system, we instead highlight one revealing section in each design's trajectory. Let us look at the state the trajectory is in after 0.01 s of simulation time, shown below in Figure 6.3 using a visual representation.

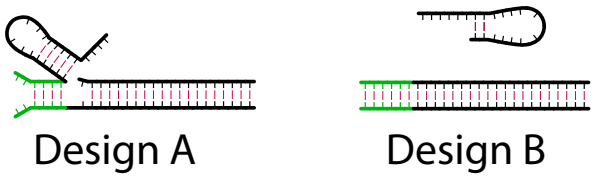


Figure 6.3: Structure after 0.01 s simulation time for two different sequence designs.

What happened? It appears that sequence design A has a structure that can form before the branch migration process initiates, that contains a hairpin in the single stranded branch migration region. Does this structure prevent the branch migration from completing? In the long run it shouldn't, as the equilibrium structure remains unchanged, but if we look at the final state in each trajectory (Figure 6.4), we see that design B has completed the process in 0.05 s of simulation time and indeed was complete at 0.01 s, where A is still stuck in that offending structure after the same amount of time. So for these specific trajectories, it's certainly slowing down the branch migration process.

Did this structure only appear because we were unlucky in the trajectory for design A? We could try running several more trajectories and seeing whether it appears in all or most of them, but a more complete answer is better handled using a different simulation mode, such as the first passage time mode discussed in Section 6.4.

A better type of question for trajectory mode is "How did this kinetic trap form?". In this example, we can examine the trajectory for design A and find the sequence of system

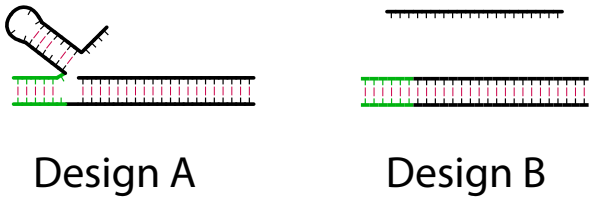


Figure 6.4: Final structure (0.05 s simulation time) for the two different sequence designs from Table 6.1. Branch migration regions: Design A: `ACCGCACGTCCACGGTGTCG`, Design B: `ACCGCACCACGTGGGTGTCG`.

microstates that lead to the first time the hairpin structure forms. This example has a straightforward answer: the competing structure forms before the branch migration starts, and is therefore in direct competition with the correct kinetic pathway.

We expect that the most common usage for trajectory mode is in providing the raw trajectory data for a separate tool to perform processing on. For example, taking the raw trajectory data and producing a movie of the structure's conformational changes can be very helpful in visualizing a system, and also is quite helpful for examining kinetic traps. A quick movie of the 3-way branch migration system could identify how the kinetic trap forms, rather than our examination of thousands of states by hand to locate that point.

6.1.1 Testing: Energy Model

We have also used the trajectory mode to aid in verifying that the kinetics model and energy model was implemented correctly. For the energy model, we can use an augmented output that includes the Multistrand-calculated energy for a given state, and compare that to the energy predicted by NUPACK [24] (or whichever tool / source we are using for our energy parameter dataset). This can be done using trajectory mode, with a cutoff time of 0 s, so the initial state is the only one in each trajectory. Multistrand's energy model was verified to be consistent with NUPACK for every sequence and structure in a comprehensive test set of secondary structures (part of the NUPACK package) that covers all possible loop configurations.

6.1.2 Testing: Kinetics Model

Testing the kinetics model can be done by testing that the *detailed balance* condition in fact holds: We know that at equilibrium, if our kinetics model obeys detailed balance,

the distribution of states seen by our simulator (after sufficient time to reach equilibrium) should agree with the Boltzmann distribution on each system microstate's energy. There are several ways we could extract this information from trajectory mode, such as recording all microstates seen in the trajectory (perhaps after some minimum time) and the amount of time spent in each one.

For our testing of the detailed balance condition we use a different method that is simpler to implement: we run many trajectories with a fixed maximum simulation time t and record only the final state in the trajectory (note that this is the state at time t in the trajectory, **not** the state which caused the stopping condition to be met). Assuming that the time t is large enough for us to reach equilibrium, we can compare the probability distribution over the final states seen by the simulation to that predicted using the NUPACK partition function and energy calculation utilities. In particular, for each final state observed in a trajectory we count the number of times it occured as a final state in our simulation, and use that to compute the simulation probability for that state. We then calculate the thermodynamic probability of observing that state using the NUPACK tools. Finally, we take the absolute value of the difference between the thermodynamic probability and the simulation probability for each final state observed and sum those quantities to obtain the total probability difference between our simulator and the thermodynamic predictions.

For our test cases we found this probability difference to be less than 1% when running a sufficient number of trajectories (approximately 10^5). This measure steadily decreases with increased trajectory count, and does not change when the simulation time is exponentially increased, indicating that our chosen t was enough to reach an equilibrium state and the probability difference is due to the stochastic nature of the simulation. The states which we observed accounted for 99.95% of the partition function, and that percentage also increases with increased number of trajectories.

6.2 Macrostates

In section 2.3 we defined a *system microstate*, which represents the configuration (primary and secondary structure) of the strands in the simulation volume. In this section, we will define a *macrostate* of the system and show how these objects can help us analyze a system by providing better stop states, as well as allowing new avenues of analysis, as discussed in

section 6.3. To make things simpler in this section, when we refer to a *microstate* we always mean a system microstate unless stated otherwise.

Formally, we define a *macrostate* m as a non-empty set of microstates: $m = \{s_1, s_2, ..., s_n\}$, where each s_i is a microstate of the system. Now we wish to derive the free energy of a macrostate, $\Delta G(m)$ in such a way that the probability of observing the macrostate m at equilibrium is consistent with probability of observing any of the contained microstates.

$$\begin{aligned} Pr(m) = & \quad Pr(s_1) + Pr(s_2) + \ldots + Pr(s_n) \\ = & \quad \sum_{1 \leq i \leq n} Pr(s_i) \\ = & \quad \sum_{1 \leq i \leq n} \frac{1}{Q} * e^{-\Delta G_{box}(s_i)/RT} \\ = & \quad \frac{1}{Q} * \sum_{1 \leq i \leq n} e^{-\Delta G_{box}(s_i)/RT} \end{aligned} \tag{6.1}$$

Now, letting $Q_m = \sum_{1 \leq i \leq n} e^{-\Delta G_{box}(s_i)/RT}$, the partition function of the macrostate m, we have $Pr(m) = \frac{Q_m}{Q}$. Similarly, in terms of the energy of the macrostate, we can express $Pr(m)$ as $\frac{1}{Q} * e^{-\Delta G(m)/RT}$, and plugging into (6.1) and solving for $\Delta G(m)$, we get:

$$\begin{aligned} \frac{1}{Q} * e^{-\Delta G(m)/RT} = & \quad \frac{1}{Q} * Q_m \\ e^{-\Delta G(m)/RT} = & \quad Q_m \\ -\Delta G(m)/RT = & \quad \log Q_m \\ \Delta G(m) = & \quad -RT * \log Q_m \end{aligned} \tag{6.2}$$

Now that we have the formal definition out of the way, let's look at an example macrostate using the same three-way branch migration system as in the previous section, figure 6.2.

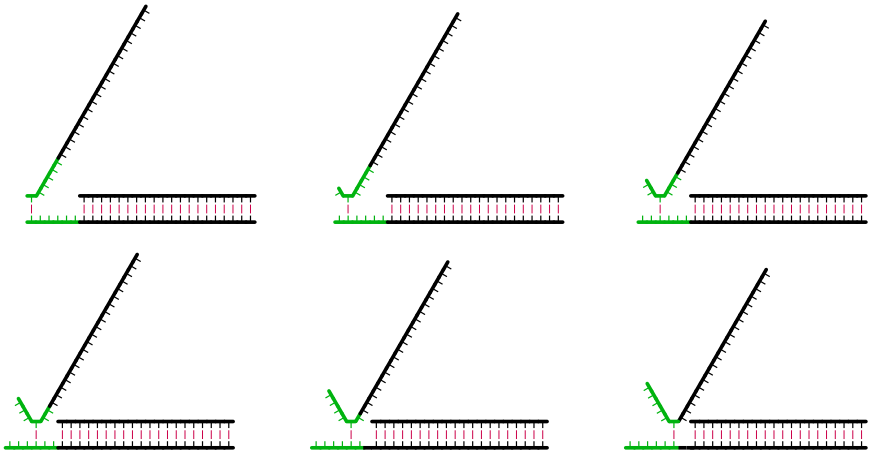
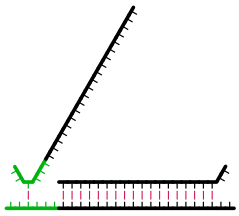


Figure 6.5: Example Macrostate

What does this macrostate represent? It's a set of microstates that has exactly one basepair formed in the toehold region, but it's not every such microstate – every microstate shown has the entire branch migration region fully formed. Thus the following microstate isn't included in the macrostate, but it does have exactly one basepair formed in the toehold region:



Why are these general macrostates interesting? Previously, we defined stop states as being microstates of the system, and we can use any number of them as part of the simulator's stop conditions. From that, it's easy to see that any given macrostate m could be used as a stop state of the system by simply expanding it out into the list of microstates contained within and using those as individual stop states.

Of particular interest to us are several classes of macrostates which can be described in very simple terms and also checked efficiently by the simulator *without* having to individually check for each microstate within those macrostates. The ability to check for a macrostate efficiently is very important: if we allowed the branch migration region in the previous example to have any structure, the macrostate would contain over 2^{22} microstates, and

even if we allowed only a limited number of bases in the branch migration region to be breathing (such as 3 base pairs, e.g. 6 bases) this is still 1140 microstates.

One useful tool in defining these classes of macrostates is a distance metric for comparing two complex microstates c_i, c_j. The distance $d(c_i, c_j)$ is defined as ∞ if c_i and c_j do not have the same set of strands and strand ordering, and otherwise as the number of bases which are paired differently between the two structures: e.g. if base x is paired with base y in c_i, but base x isn't paired with y in c_j, or if base x is unpaired in c_i, but base x is paired in c_j. This distance metric has been used in other work, using a slightly different but equivalent formulation for example [5, 12] and references therein. Some examples are shown below, in table 6.2.

c_i	Structure	Distance
c_0	....((((_))))....	
c_1	((((((((_))))))))	$d(c_0, c_1) = 8$
c_2	....((((_)).....))	$d(c_0, c_2) = 6$
c_3	......((_)).....	$d(c_0, c_3) = 4$
c_4	....((((_))).....)	$d(c_0, c_4) = 3$
c_5	.(....)(_)........	$d(c_0, c_5) = 7$

Table 6.2: Distance metric examples, for complex microstates on the two strand complex with sequences AGCTAGCT,AGCTAGCT. Bases that differ from the structure c_0 are shown in red.

Now that we have a distance metric, we define several common macrostates that can be used in the simulator as stopping conditions.

6.2.1 Common Macrostates

Disassoc: Given a set of strands ST and an ordering π^* on those strands, we define the **Disassoc** macrostate m as the set of all system microstates s which contain a complex microstate c with exactly the strands ST and ordering π^*. Recall that a complex microstate (Section 2.2) is defined by three quantities, the strands contained in the connected complex, the ordering on those strands, and the base pairs present; thus this definition implies no particular set of base pairs are present, though it does require that the complex be connected. Note that this macrostate can only be reached by either a association or disassociation step, allowing it to be efficiently checked as we only need to do so when encountering a bimolecular move. It's called **Disassoc** in

light of its most common usage, but it could also be used to stop after an association event.

Bound: Given a single strand S, we define the **Bound** macrostate m as the set of all system microstates s which contain a complex microstate c with set of strands ST that has $S \in ST$ and $|ST| > 1$.

Count: Given a complex microstate c and an integer count k, we define the **Count** macrostate m as the set of all system microstates s which contain a complex microstate c' for which $d(c, c') \leq k$. Note that c' which meet this criteria must have the same strands and strand ordering, as $d(c, c') = \infty$ if they do not. For convenience, instead of using the integer count k we allow passing a percentage p which represents a percentage of the total number of bases N in the complex c. If this is done, we use a cutoff $k = \lceil p * N \rceil$.

Loose: Given a complex microstate c, a integer count k and a set of bases B that is a subset of all the bases in c, we define the **Loose** macrostate m as the set of all system microstates s which contain a complex microstate c' for which $d_B(c, c') \leq k$, where we define d_B as the distance metric d over only the set of bases B in c. Similar to the **Count** macrostate, we allow a percentage p instead of k, for which we set $k = \lceil p*|B| \rceil$. This macrostate allows us to specify a specific region of interest in a microstate, such as just a toehold region we wish to be bound without caring about other areas in the complex microstate.

Note that each of these macrostates is based on the properties of a single complex microstate occuring within a system microstate; thus if desired we could make a stopping condition which uses several of these in conjunction. For example, we might make a stopping conditions that has **Disassoc** for strand A and **Disassoc** for strand B, thus creating a macrostate which can be described in words as "strand A is in a complex by itself, and strand B is in a complex by itself, and we don't care about any other parts of the system". Similarly we can implement disjunction simply by using multiple independent stopping conditions. Though the **NOT** operation is not currently implemented for these stop conditions, it may be added in the future, allowing us to have the full range of boolean operations on these common macrostates. As it is, we can easily implement the original example macrostate

simply by using an **OR** of the six exact system microstates. Or we could use **Loose** macrostates to implement the one we might have intended, where we didn't care very much about the branch migration region (and thus allowed it to have some breathing base pairs), only that a single base of the toehold had been formed.

6.3 Transition Mode

What is transition mode? The basic idea is that instead of **every** system microstate being an interesting piece of the trajectory, we provide (as part of the input) a list T of *transition states* of the system, the states which we think are interesting, and the output is then the times when we enter or leave any transition state in the list T. These transition states can be exact states of the system (e.g. system microstates), or macrostates of the system (e.g. a combinition of common macrostates such as **Dissasoc** or **Loose** macrostates), and we note that they are not required to be technical "transition states" as in chemical reaction theory – we are interested in how trajectories move (i.e. transition) from macrostate to macrostate, no matter how those macrostates are defined. One way to look at this form of output is as a trajectory across transition state membership vectors. We note that since these transition states are defined in exactly the same way as stop states, we generally lump them both together in the list of transition states that get reported (after all, you'd like to know what state caused the simulation to finish, right?), with a special labelling for which transition states are also stop states.

What is transition mode good for? The simplest answer is that it allows us to ask questions about specific kinetic pathways. Here's an example of this: Given a simple sequence that forms a hairpin, does it form starting from the bases closest to the 5'/3' ends (Fig 6.6B), or starting from the bases closest to the hairpin region (Fig 6.6C)?

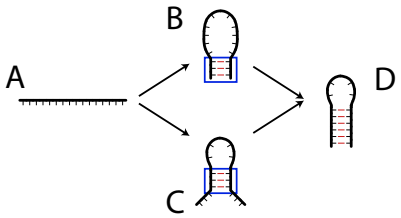


Figure 6.6: Hairpin Folding Pathways. Blue boxes indicate regions of interest used in loose structure definitions (Table 6.3). A) Starting State. B) Bases near the 5'/3' ends form first. C) Bases near the hairpin region form first. D) Final hairpin structure.

How do we represent these pathways in terms of transition states? Here we take advantage of the common macrostate definitions (Section 6.2.1) to define the intermediate structures B and C, using loose macrostates with a distance of 2, while A and D are defined with exact microstates.

Transition State Label	Sequence / Structure	State Type
	GCATGCAAAAGCATGC	
A (start)	.....	Exact
B	(((**********)))	Loose, $d \leq 2$
C	***(((****)))***	Loose, $d \leq 2$
D (stop)	((((((....))))))	Exact

Table 6.3: Transition states for hairpin pathway example. State type of **Exact** is exactly the given structure as a system microstate, and **Loose** is a loose macrostate covering only the bases in blue (or alternately, the bases not marked with "*").

Why is using a loose macrostate for these transition states useful? First, we note that we produce output any time the transition state membership changes, hence each step of the pathway is the set of all transition states which match the system microstate. Let's look at a possible pathway to the stop state where the first bases that form are near the 5' and 3' ends and the base pairs are added sequentially without ever being broken. With exact states this would result in the following transition pathway: $\{A\} \rightarrow \emptyset \rightarrow \{B\} \rightarrow \emptyset \rightarrow \{D\}$ and with the loose macrostates it would be this transition pathway: $\{A\} \rightarrow \emptyset \rightarrow \{B\} \rightarrow \{B, C\} \rightarrow \{B, C, D\}$. So far, so good. What about if we form two bases of B, then all of C, then the last base of B? For loose states, this is the exact same transition pathway - recall that we use a distance of 2, and two base pairs formed in B is exactly that distance away from the given structure. But for exact states, this is now the (very boring) pathway $\{A\} \rightarrow \emptyset \rightarrow \{D\}$, which doesn't answer our question about which part of the helix formed first!

Two possible transition pathways, using either the loose structures for B and C, or exact structures:

Time	Transition States
0.00	A
$3.63 * 10^{-7}$	$\emptyset$
$1.03 * 10^{-6}$	A
$1.40 * 10^{-6}$	$\emptyset$
$1.78 * 10^{-6}$	B
$1.92 * 10^{-6}$	B, C
$2.15 * 10^{-6}$	B, C, D

(a) Sample Transition Pathway (Loose)

Time	Transition States
0.00	A
$9.02 * 10^{-7}$	$\emptyset$
$1.31 * 10^{-6}$	A
$2.26 * 10^{-6}$	$\emptyset$
$2.72 * 10^{-6}$	D

(b) Sample Transition Pathway (Exact)

Table 6.4: Two different transition pathways via transition mode simulation, using either the given B and C states with the loose macrostate definitions from Table 6.3, or exact system microstates using the states from the same table with all "*" replaced by "." (unpaired) and distance set to 0, effectively. Note that the times listed are the times of first entering the given state.

Does this mean every simulated trajectory takes these transition pathways? Definitely not! The stochastic nature of the simulator means we're likely to see many different transition pathways if we run many trajectories. So, let's now answer the original question: which transition pathway is more likely? We do this by accumulating statistics over many kinetic trajectories as follows: For each transition path trajectory (such as those in Table 6.4) we break down the trajectory into pieces which have non-empty sets of transition states, separated only by zero or one empty set of transition states. So for example, the path shown in Table 6.4a breaks down into four separate reactions: $\{A\} \to \emptyset \to \{A\}$, $\{A\} \to \emptyset \to \{B\}$, $\{B\} \to \{B, C\}$, and $\{B, C\} \to \{B, C, D\}$. For our statistics, we'll group reactions of the form $x \to \emptyset \to y$ with those of the form $x \to y$, and for every possible reaction, we record the number of times it occurred and the average time it took to occur. So for the single pathway in Table 6.4a we get the following statistics:

Reaction	Average Time	Number of Occurences
$A \to A$	$1.03 * 10^{-6}$	1
$A \to B$	$7.43 * 10^{-7}$	1
$B \to B, C$	$1.47 * 10^{-7}$	1
$B, C \to B, C, D$	$2.29 * 10^{-7}$	1

Table 6.5: Statistics for the single transition pathway shown in Table 6.4a.

Now that we've seen an example of these statistics for a single kinetic trajectory, let's look at the same statistics over a hundred kinetic trajectories, again using the system with loose macrostates.

Reaction	Average Time	Number of Occurences
$A \to A$	$2.48 * 10^{-6}$	829
$A \to B$	$2.17 * 10^{-7}$	37
$A \to C$	$2.53 * 10^{-7}$	73
$B \to A$	$1.09 * 10^{-6}$	5
$B \to B$	$1.46 * 10^{-7}$	2
$B \to B, C$	$3.78 * 10^{-7}$	33
$C \to A$	$5.63 * 10^{-7}$	5
$C \to C$	$2.48 * 10^{-7}$	7
$C \to B, C$	$5.84 * 10^{-7}$	77
$B, C \to B$	$4.32 * 10^{-7}$	1
$B, C \to C$	$1.21 * 10^{-7}$	9
$B, C \to B, C, D$	$2.10 * 10^{-7}$	100

Table 6.6: Statistics for 100 simulated trajectories using the transition states from Table 6.3.

What can we conclude from these statistics? Both pathways do occur, but it is much more likely that the first bases formed are those closest to the hairpin region. The average times for each pathway are roughly within an order of magnitude of each other, and our selection of transition states was good: we didn't see any unexpected pathways, such as $\{A\} \to \{D\}$.

We could use these "reactions" as to create a coarse-grained representation of the original system as a chemical reaction network, using $\frac{1}{\text{avg time}}$ as the reaction rate constants. Whether this will be an accurate representation or not depends on the choice of transition states and the structure of the energy landscape. For example, if we were to try this using the average times for this system, we would end up with a formal CRN in which the $A \to A$ reaction is taken far less frequently than shown in Table 6.6. Finding appropriate coarse-grained representations is a deep and subtle topic [13].

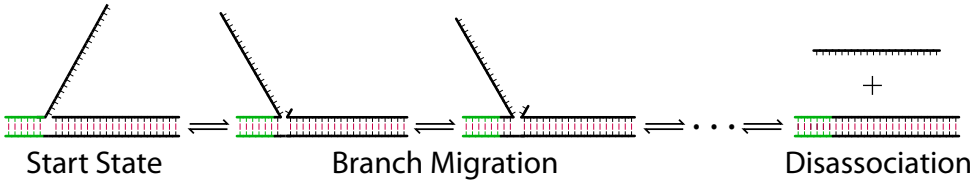
6.4 First Passage Time Mode

First passage time mode is the most basic simulation mode in Multistrand. It produces a single piece of data for each trajectory simulated: the first passage time for reaching any stop state in the system, and which stop state was reached. This is a rather striking difference from our previous simulation modes in the amount of data produced for each individual trajectory, but it is still quite powerful!

This first passage time data could be produced via trajectory mode: we can just discard

all the output until a stop state is reached. There is a distinct efficiency advantage to making it a separate simulation mode: we don't have to pay the overhead of reporting every piece of trajectory data only for it to be discarded. Similarly, we could generate the same data using transition mode by only using stop states in our list of transition states. We implement this as a distinct simulation mode in order to better separate the reasons for using each simulation mode: for transition mode, we are interested in the pathway our system takes to reach a stop state, and for first passage time mode we are interested in how quickly the system reaches the stop state(s).

What does first passage time data look like? Let's revisit our example system from section 6.1 (Figure 6.2):



We start the system as shown, and use two different stop states: the **complete** stop condition where the incumbent strand has disassociated (as shown in the figure), and the **failed** stop condition where the invading strand has disassociated without completing the branch migration. Both of these are done using *Disassoc* macrostates, which makes it very efficient to check the stop states. Note that we include the invading strand disassociating as a stop state so that if it occurs (which should be very rarely), we can find out easily without waiting until the maximum simulation time or until the strands reassociate and complete the branch migration.

The following table (Table 6.7) shows a five trajectories worth of data from first passage time mode on the example system, using sequence design B (Table 6.1) for the branch migration region.

Note that we have included a third piece of data for each trajectory, which is the pseudorandom number generator seed used to simulate that trajectory. This allows us to produce the exact same trajectory again using a different simulation mode, stop states or other output conditions. For example, we might wish to run the fifth trajectory in the table again using trajectory mode, to see why it took longer than the others, or run the first

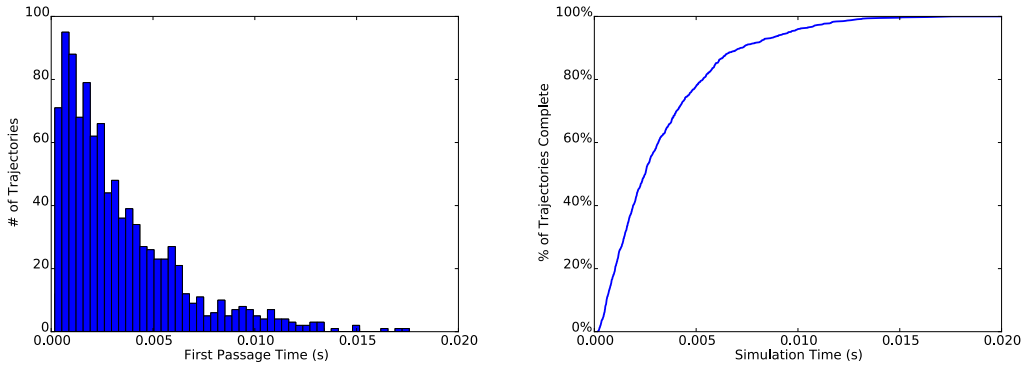
Random Number Seed	Completion Time	Stop Condition
`0x790e400d`	$3.7 * 10^{-3}$	failed
`0x38188213`	$3.8 * 10^{-3}$	complete
`0x47607ebf`	$2.1 * 10^{-3}$	complete
`0x02efe7fa`	$2.8 * 10^{-3}$	complete
`0x7c590233`	$6.7 * 10^{-3}$	complete

Table 6.7: First passage time data for the example three way branch migration system. Stop conditions are either "complete", indicating the branch migration completed successfully, or "failed", indicating the strands fell apart before the branch migration could complete.

trajectory to see what kinetic pathway it took to reach the **failed** stop condition.

Let's now look at a much larger data set for first passage time mode. Here we again use the three way branch migration system with sequence design B for the branch migration region and increase the toehold region to be ten bases, to minimize the number of trajectories that reach the **failed** stop condition. We run 1000 trajectories, using a maximum simulation time of 1 s, though no trajectory actually used that much as we shall shortly see.

Instead of listing all the trajectories in a table, we graph the first passage time data for the **complete** stop condition in two different ways: first (Figure 6.7a) we make a histogram of the distribution of first passage times for the data set, and second (Figure 6.7b) we graph the percentage of trajectories in our sample that have reached the **complete** stop condition as a function of the simulation time.



(a) Histogram of first passage times

(b) Percent completion by simulation time

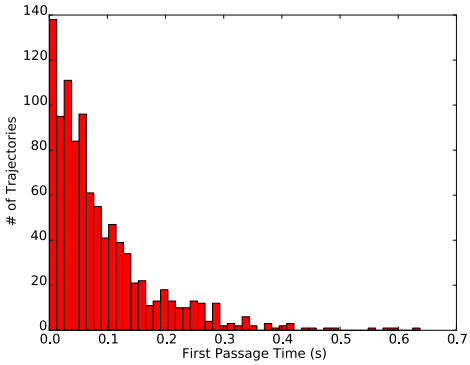
Figure 6.7: First passage time data for the three way branch migration system, using sequence design B (Table 6.1) and with a ten base toehold sequence. 1000 trajectories were simulated and all of them ended with the **complete** stop condition.

While there are many ways to analyze these figures, we note two particular observations. Firstly, the histogram of the first passage time distribution looks suspiciously like an exponential distribution, possibly with a short delay. This is not always typical (as we shall shortly see), but the shape of this histogram can be very helpful in inferring how we might wish to model our system based on the simulation data; e.g. for this system, we might decide that the three way branch migration process is roughly exponential (with some fitted rate parameter) and so we could model it as a one step unimolecular process.

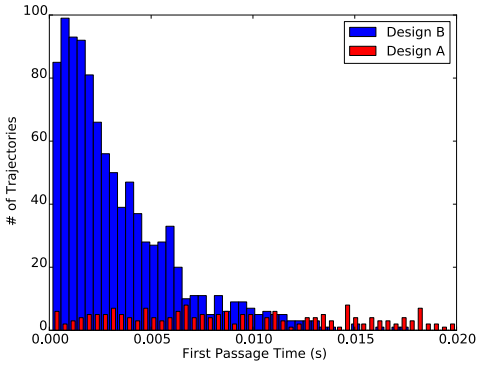
The second observation is that while the percentage completion graph looks very similar to an experimental fluourescence microscopy curve, they should **NOT** be assumed to be directly comparable. The main pitfall to watch out for is when comparing fluourescence curves from systems where the reactions are bimolecular: in these the concentration of the relevant molecules are changing over time, but in our stochastic simulation the bimolecular steps are at a fixed volume/concentration (reflected in the ΔG_{volume} energy term) and data is aggregated over many trajectories.

6.4.1 Comparing Sequence Designs

A common usage of first step mode is in the comparison of sequence designs, as we previously brought up in Section 6.1. We now run another 1000 trajectories on the same three way branch migration system as in the previous section, including the increased toehold length, but using the sequence design A (Table 6.1) for the branch migration region. Note the change in x-axis scale; this design is indeed much slower than design B!



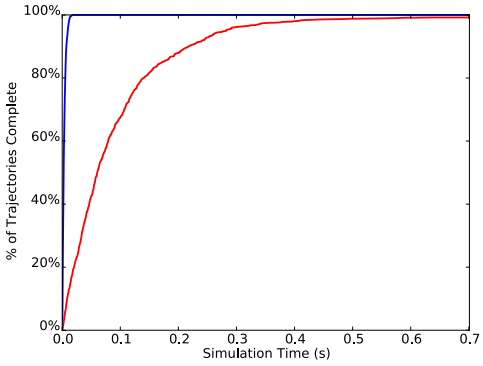
(a) Histogram of first passage times, design A



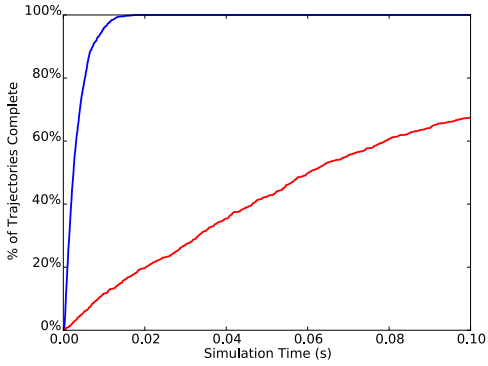
(b) Histogram of first passage times, both designs

Figure 6.8: First passage time data for the three way branch migration system, comparing sequence designs using histograms. For figure (b), we compare the two designs on the range of times from 0 s to 0.02 s. The buckets for sequence design A have been reduced in visual size to show overlapping regions, but overall bucket sizes are consistent between the two designs (though they are slightly different from those in Figure 6.7a).

Let's also look at the same data but using the percentage completion as a function of simulated time graphs:



(a) Design comparison, percent completion graph



(b) Design comparison, zoomed

Figure 6.9: First passage time data for the three way branch migration system, comparing sequence designs using percent completion graphs. Figure (b) is a zoom in on the range of times from 0 s to 0.1 s from Figure (a).

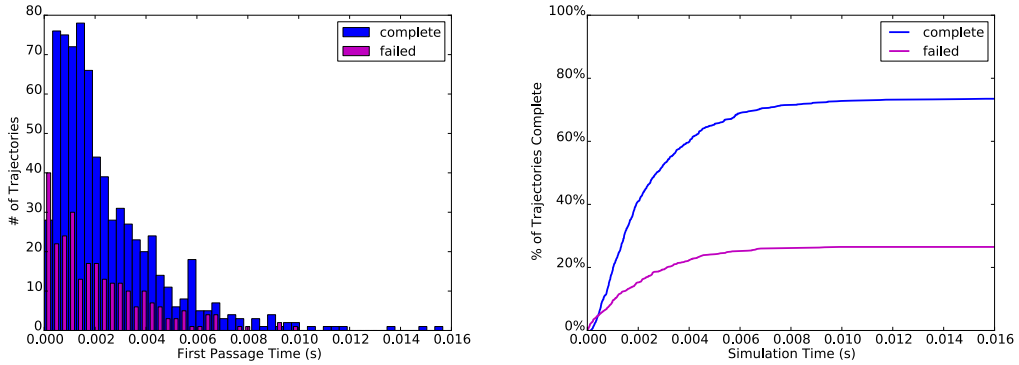
We can now clearly see just how different these two sequence designs are! Our results from the trajectory mode section were clearly not unusual: the majority of sequence design B's trajectories finish in under 0.01 s, whereas the same amount for sequence design A

is over ten times longer. This is highlighted in the combined graph (Figure 6.8b), which shows how vastly different the timescales are for each process. Looking at the percentage completion graphs, we note that sequence design A did not actually reach 100% – it actually had 8 trajectories reach the **failed** stop condition!

While both types of graphs are presenting the same information, they are frequently useful in different cases: the first passage time histograms are helpful for gaining an intuition into the actual distribution of times for the process we are simulating, while the percent completion graphs are better for looking at the relative rates of different designs, especially when working with more than two sequence designs.

6.4.2 Systems with Multiple Stop Conditions

What about our original three way branch migration system, which had toeholds of length six? Let's now look at this situation, where we have a system that has more than one competing stop state. We again run the system with 1000 trajectories and a 1 s maximum simulation time (though it's never reached). Instead of plotting only the first passage times for the **complete** stop condition, we overlay those with the first passage times for the **failed** stop condition.



(a) Histogram of first passage times

(b) Percent completion by simulation time

Figure 6.10: First passage time data for the three way branch migration system with 6 base toeholds, comparing sequence designs.

Competing stop conditions certainly make the data more interesting! We can pick out pieces of the kinetic pathways a lot easier using these graphs. For example, the competing pathway leading to the **failed** stop condition frequently occurs faster than the **complete**

stop condition pathway; this should be unsurprising, as one pathway involves a long random walk while the other is very unlikely to include one.

6.5 Fitting Chemical Reaction Equations

We are frequently interested in DNA systems which can be represented (with some choice of the appropriate internal parameters/sequences) by chemical reaction equations of the following form:

$$A + B \xrightarrow{k_{eff}} C + D \tag{6.3}$$

These systems usually involve an intermediate step, so typically the concentration is low enough for the above equation to actually be a good fit. Experimental observation of these systems tend to be in the range of concentrations where the above equation is an accurate characterization of the system.

Let us now associate the species in the equation above with actual DNA complexes (ordered collections of connected strands). We designate strands by unique letters, and indicate complementary strands with an asterisk. For the toehold-mediated 3-way branch migration example, the chemical equation then becomes:

$$x^*y + x \xrightarrow{k_{eff}} xx^* + y \tag{6.4}$$

Let us examine one possible DNA configuration for the state of the entire system that could follow this equation's implied dynamics. The left hand side could be the configuration given in figure 6.11, and the right hand side could be the configuration given in figure 6.12. Note that while we show particular exact secondary structures for each of these figures, there are actually many such secondary structures that represent the appropriate parts of the equation, which might be specified using **Loose** macrostates.

Finally, what would the equations look like if we did not expect them to fit equation 6.3? One possibility is as follows (as in Zhang, Winfree 2009 [25]):

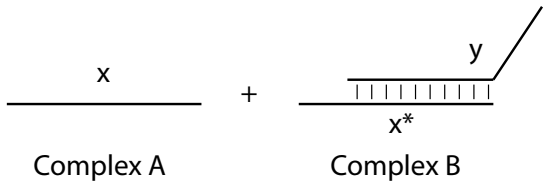


Figure 6.11: Starting Complexes and Strand labels

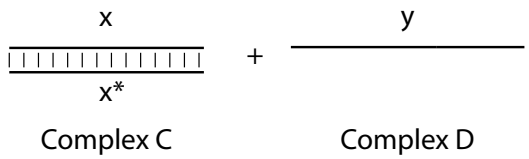


Figure 6.12: Final Complexes and Strand labels

$$A + B \underset{k_r}{\overset{k_f}{\rightleftharpoons}} AB \underset{k_{bm}}{\overset{k_{bm}}{\rightleftharpoons}} AB' \xrightarrow{k_r} C + D \tag{6.5}$$

Though this model is used in several experimental papers, it is difficult for us to define simulation macrostates in order to determine the various rates present in the equation. Instead, we look at a model where it is easy to separate out the steps into discrete components which can be individually simulated. Specifically, we look at a system where the molecules A and B can collide and form either a reactive molecule which will go to C and D, or a nonreactive molecule which will fall apart after some time[1]. We call this the **first step model**, and it is described by the equations below:

$$A + B \xrightarrow{k_1} AB \xrightarrow{k_2} C + D \tag{6.6}$$

$$A + B \underset{k_2'}{\overset{k_1'}{\rightleftharpoons}} AB' \tag{6.7}$$

We will use this model extensively to analyze the results of the **first step mode** simu-

[1]Thanks to Niles Pierce and Victor Beck for suggesting this approach.

lations discussed in section 6.6. Note that for low concentrations, the controlling step will be the bimolecular reactions and thus we should be able to also fit to a k_{eff} type model in those cases.

We discuss fitting first passage time data to the simple k_{eff} model (Equation 6.3, as well as using **first step mode** to generate data which can easily be fit to the first step model.

6.5.1 Fitting Full Simulation Data to the k_{eff} model

For this simulation mode, we start the simulation in the exact state shown in figure 6.11, and measure the first passage time to reach any state with the same complexes (but not exact secondary structure, i.e. we use a **Disassoc** stop state) as that shown in figure 6.12. This gives us a data set of first passage times Δt_i, where $1 \leq i \leq n$, and n is the total number of trajectories simulated. The simulation is done at a particular concentration z. Note that this simulation will require time inversely proportional to the simulation concentration, since we are simulating elementary steps. Thus we would prefer to simulate at higher concentrations and step downwards in concentration until we are in the region where the bimolecular reaction dominates, and equation 6.3 holds.

If we assume equation 6.3 holds and determines our distribution of first reaction times, we can fit our data to an exponential distribution in order to determine k_{eff}: Recall that (via Gillespie [8]) in a formal chemical reaction network, a state with total outgoing propensity a_0 will have a passage time (τ) distribution according to the probability density function $P(\tau) = a_0 * \exp(-a_0\tau)$. Since the propensity a_0 for a bimolecular reaction is the reaction rate times the concentration, we solve for k_{eff} as follows, where expfit is the function that takes a data set of first passage times (Δt_i) and returns the parameter a_0 of the exponential distribution given by those Δt_i, and z is the simulated concentration:

$$k_{eff} = \text{expfit}(\Delta t_i) * 1/z \tag{6.8}$$

If we are in the regime where equation 6.3 holds, we expect this to give us a consistent value for k_{eff}. If we are outside of that regime (and thus the unimolecular reactions from

equation 6.5 or equation 6.6 dominates), this will err by exactly the factor of $1/z$, thus the graph of k_{eff} versus concentration z should appear linear in this regime.

6.6 First Step Mode

This simulation mode makes a simple assumption: we always start the Markov simulation by making a "join" step, that is, one where a pair of molecules A and B come together and form a single base pair. The choice of secondary structure states for the molecules A and B before collision can be done either by using particular exact complex microstates, or by Boltzmann sampling the secondary structure space of the molecules. This sampling is valid when the bimolecular reaction rates are slow enough that the initial complexes reach equilibrium. In either case, we have a valid system microstate once we know the structure for the A and B molecules, and then choose a "join" step from those present in the system microstate and use the resulting system microstate (after making the join) as our start state for a trajectory. Since this mode runs many trajectories, we must make many such random choices for the join step and for the Boltzmann sampling of the initial molecules (if we are using the sampling rather than exact states).

The simulation then starts from this configuration, and we track two distinct end states: the molecules falling apart back into one of the $A + B$ configurations, or the molecules reacting into one of the $C + D$ configurations. Our data then consists of first passage times where we can separate each trajectory into one that reacted or one that failed. The advantage to this mode of simulation is that we no longer are directly simulating the "join" bimolecular steps, whose rates are proportional to the simulated concentration and thus are going to be very slow relative to the normal unimolecular steps. This allows us to use the simulator to concentrate on the trajectories where we have a collision, rather than spending (at low concentrations) most of the time simulating unimolecular reactions while waiting for the very rare bimolecular reaction.

6.6.1 Fitting the First Step Model

Our first step mode simulation produces the following pieces of data: Δt^i_{react}, the first passage times for reactive trajectories, Δt^i_{fail} the first passage times for trajectories did not react, the number of trajectories that reacted N_{react} and failed N_{fail}, and the rate constant

k_{coll}, the simulation's estimate of the rate of collision (in $/M/s$) of the A and B molecules. This k_{coll} is calculated based on *all* the join moves possible from the initial configuration of the A and B molecules, and thus is very likely to include many such moves which do not lead to very stable structures and thus disassociate quickly.

We then fit to the model given in equations 6.6 and 6.7, as follows:

$$k_1 = \frac{N_{react}}{N_{react} + N_{fail}} * k_{coll} \tag{6.9}$$

$$k_1' = \frac{N_{fail}}{N_{react} + N_{fail}} * k_{coll} \tag{6.10}$$

$$k_2 = \text{expfit}(\Delta t^i_{react}) \tag{6.11}$$

$$k_2' = \text{expfit}(\Delta t^i_{fail}) \tag{6.12}$$

Thus we can directly find each of the model parameters from the collected simulation data, in a natural way. We then use this model to predict the k_{eff} parameter we would observe if we assume that the simple chemical reaction model (eqn 6.3) is valid.

6.6.2 Analysis of First Step Model Parameters

We first show a natural (but inexact) way to calculate k_{eff} from the first step model parameters, using the assumption that the time used in "failed" collisions is negligible compared to that needed for a successful reaction. In this situation, we can estimate k_{eff} for a particular concentration z by calculating the expected average time for a reaction. We use the fact that the expected value of an exponential distribution with rate parameter λ will be $\frac{1}{\lambda}$ in order to derive k_{eff}:

$$k_{eff} = \frac{1}{z} * \frac{1}{\frac{1}{k_1 * z} + \frac{1}{k_2}} \tag{6.13}$$

Again, this makes the assumption that equation 6.3 holds and thus that the reaction dominated by the bimolecular step. The observation from the full simulation mode still holds: if we are not in this regime, we will err by a factor of $\frac{1}{z}$ and thus the graph should be linear. Note that since this simulation does not require a set concentration, we can run

one simulation (with a large number of trajectories) and use the extracted data to produce the same type of graph as the full simulation mode.

We now would like to remove the assumption that the "failed" collision time is negligible: though that assumption makes k_{eff} straightforward to calculate, many systems of interest will not satisfy that condition.

We now need to calculate the expected time for a "successful" reaction to occur based on both the "failed" and "reactive" collision parameters. We do this by summing over all possible paths through the reactions in equations 6.6 and 6.7, weighted by the probability of those reactions. Let $\Delta t_{coll} = \frac{1}{(k_1+k_1')*z}$ (the expected time for any collision to occur), $\Delta t_{fail} = \Delta t_{coll} + \frac{1}{k_2'}$ (the expected time needed for a failed collision to return to the initial state), $\Delta t_{react} = \Delta t_{coll} + \frac{1}{k_2}$ (the expected time for a reactive collision to reach the final state), $p(path)$ is the probability of a particular path occuring, and $\Delta t_{path} = n\Delta t_{fail} + \Delta t_{react}$ is the expected time for a path which has n failed collisions and then a successful collision. Finally, the quantity which we want to solve for is $\Delta t_{correct}$, the expected time it takes for a successful reaction to occur.

$$\Delta t_{correct} = \sum_{path} \Delta t_{path} * p(path) \tag{6.14}$$

$$= \sum_{n=0}^{\infty} (n\Delta t_{fail} + \Delta t_{react}) * (\frac{k_1'}{k_1 + k_1'})^n * \frac{k_1}{k_1 + k_1'} \tag{6.15}$$

To simplify the next step, let $\alpha = \frac{k_1}{k_1+k_1'}$ and $\alpha' = \frac{k_1'}{k_1+k_1'}$, and recall that for $\beta > 0$, $\sum_{n=0}^{\infty} \beta^n = \frac{1}{1-\beta}$ and $\sum_{n=0}^{\infty} n * \beta^n = \beta * \frac{1}{(1-\beta)^2}$, and we get:

$$\Delta t_{correct}) = \Delta t_{fail} * \frac{\alpha'}{(1-\alpha')^2} * \alpha + \Delta t_{react} * \frac{1}{1-\alpha'} * \alpha \tag{6.16}$$

Now we note that $\frac{\alpha}{1-\alpha'} = 1$, and $\frac{\alpha'}{1-\alpha'} = \frac{k_1'}{k_1}$, and simplify:

$$\Delta t_{correct}) = \Delta t_{fail} * \frac{k_1'}{k_1} + \Delta t_{react} \tag{6.17}$$

And thus we arrive at the (full) form for k_{eff}:

$$k_{eff} = \frac{1}{\Delta t_{correct}} * \frac{1}{z} \tag{6.18}$$

This requires only a single assumption: that the reaction is dominated by the bimolecular step, and thus can be described by equation 6.3. We note that this derivation can be generalized for multiple possible stop states [2].