Mathematical models of DNA-protein interactions

Candidate 1007177

Department of Mathematics
University of Oxford

This dissertation is submitted for the degree of
Master of Mathematics

September 2019

Table of contents

List of tables

Nomenclature

Acronyms / Abbreviations

BM	Brownian motion
FJC	Freely jointed chain
MD	Molecular dynamics
SDE	Stochastic differential equation
TBP	TATA-binding protein
TF	Transcription factor

Constants

k_B	Boltzmann constant	$\mathrm{m^2\,kg\,s^{-2}\,K^{-1}}$

Parameters

η	Solvent viscosity	$\mathrm{kg\,m^{-1}\,s^{-1}}$
γ	Drag force	N
λ_μ	Density of heat bath particles	$\mathrm{m^{-3}}$
μ	Ratio of diffusing particle mass to solvent particle mass	1
ω_μ^2	Second moment of heat bath velocity components	$\mathrm{m^2\,s^{-2}}$
σ	Radius of a bead on the Rouse chain	m
d_0	Initial radius between middle bead and protein	m
d_b	Radius of binding sphere	m

D_p	Diffusion coefficient of protein	$\mathrm{m^2\,s^{-1}}$
d_∞	Radius of reflective boundary	m
k_i	Spring constant associated with i^{th} spring	$\mathrm{N\,m^{-1}}$
N	Number of springs in Rouse model	1
R	Radius of particle in heat bath	m
r_I	Resolution increase radius	m
s	Resolution factor	1
T	Absolute temperature	K
Z	Zoom out factor	1

Sets

Ω	The computational domain
B	An orthonormal basis
S	The set of points on the surface of the diffusing particle

Variables

$\mathbf{q}_i(t)$	Position of i^{th} end bead at time t
$\mathbf{r}_i(t)$	Position of i^{th} bead at time t
$\mathbf{V}(t)$	Velocity of diffusing particle at time t
$\mathbf{v}_i(t)$	Velocity of i^{th} heat bath particle at time t
$\mathbf{w}_i$	Spring vector i
$\mathbf{X}(t)$	Position of diffusing particle at time t
$\mathbf{x}_i(t)$	Position of i^{th} heat bath particle at time t

Chapter 1

Introduction

In the past few years, there have been multiple attempts to use existing models of polymer chains as a coarse-grained model of DNA dynamics. For instance, one of the first such applications was the Rouse model: a simple bead-and-spring model of polymer dynamics. The benefit of using a bead-and-spring model is clear, since the double-helix structure of DNA is broadly speaking too complex to use in molecular dynamics (MD) or even mesoscopic regimes at a cellular level.

DNA filaments are found in the cell nucleus and contain the genetic code necessary for new proteins to be synthesised. The first step of this synthesis is a process known as transcription which is carried out by the protein RNA polymerase. However, before RNA polymerase can commence transcription, proteins known as transcription factors (TFs) must bind to cognate promoter regions on the DNA filaments. Exactly how these TFs find and bind to the correct part of the genome is a topic of ongoing investigation, but a 4-fold approach of diffusion, sliding along the chromosome, hopping over other TFs and jumping where the DNA folds has been proposed in the literature [11].

To aid future investigation into this process, we consider the first passage problem throughout this project. That is to say, we consider the time taken for a TF to bind to the DNA (given an initial displacement) and also the probability of this happening without the TF reaching the boundary of the domain (since this is a problem that is explored in the literature [10]). In this way, we are able to get a picture for how TFs in the nucleus are able to bind to the correct part of the genome quickly enough for our bodies to function correctly.

The aforementioned Rouse model treats the DNA as an ideal chain, with beads subjected only to forces from the harmonic springs attaching them to their neighbours, as well as some stochastic noise in the form of Brownian motion. Whilst computationally very efficient, a regime of this type can fail to provide the level of detail needed for an investigation of this type, as we shall see in Chapter 2.

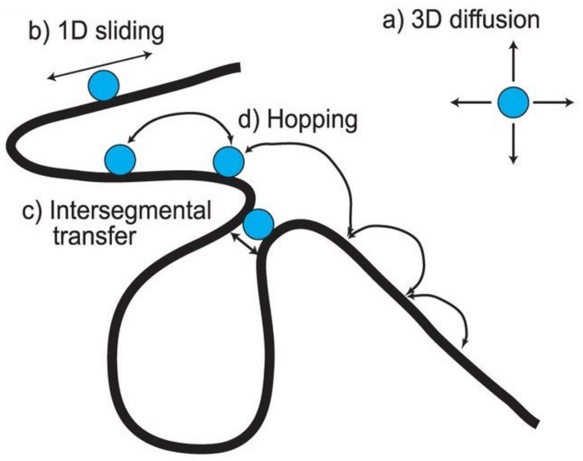

Fig. 1.1 The 4 modes of transcription factor motion [11]

Hence, the subsequent chapters in this project are centred around regimes with varying levels of resolution which can retain the level of detail offered by MD in a desired subdomain whilst preserving the efficiency offered by mesoscopic models. In Chapter 3, we present a model which changes resolution on-the-fly before introducing a theoretical heat bath model in Chapter 4 which forms the basis of the multi-resolution scheme which we introduce in Chapter 5. We conclude with a comparison of the different techniques and an in-depth discussion of potential avenues for future improvement.

1.1 Simple analytical model

Before we commence our study of more detailed models, we consider a very basic model with which we can make analytical predictions without need for simulation. We consider the DNA as an absorbing sphere, centred on the origin, with a larger concentric sphere acting as a reflecting boundary. The TF is able to diffuse in the space between the two spheres under Brownian motion with diffusion coefficient D, so its position $\mathbf{X}$ can be described by the discretised SDE:

$$\mathbf{X}(t+\Delta t) = \mathbf{X}(t) + \sqrt{2D\Delta t}\,\xi. \tag{1.1}$$

Due to the symmetry of the problem, we can reduce this to a 1-dimensional symmetric random walk with a reflecting boundary at $x = N$ and an absorbing boundary at $x = 0$, as shown in Figure 1.2.

We proceed by defining the first hitting time

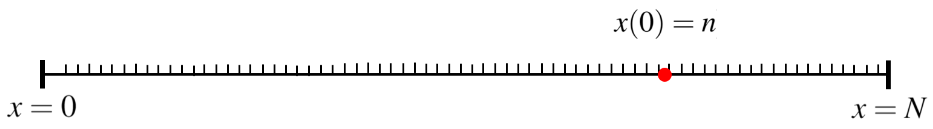

Fig. 1.2 The random walk in 1D

Definition 1.1.1. The first hitting time (or first passage time) of the 1D random walk above is defined as $h(n)$, where

$$h(n) = \mathbb{E}\left[inf\left\{t > 0|x(t) = 0\right\}|x(0) = n\right].$$

By conditioning on the first jump, we can derive the following difference equation for $h(n)$:

$$h(n+1) - 2h(n) + h(n-1) = -2, \tag{1.2}$$

With boundary conditions given by

$$h(0) = 0, \tag{1.3}$$

$$\left.\frac{dh(n)}{dn}\right|_{n=N} = 0. \tag{1.4}$$

In particular, the Neumann boundary condition given by Equation 1.4 is necessary to make the outer sphere's surface reflective.

The solution to Equation 1.2, subject to these boundary conditions, is given by

$$h(n) = 2Nn - n^2. \tag{1.5}$$

Now we relate N to the diffusion coefficient D in order to derive an estimate that is biologically tractable. If the length of the domain is ℓ and consists of N steps then the second moment of each jump is $\frac{\ell^2}{N^2}$. If the timescale of the random walk is the same as the timescale of Brownian motion in Equation 1.1, then we can relate this to D by:

$$N = \frac{\ell}{\sqrt{2D}}. \tag{1.6}$$

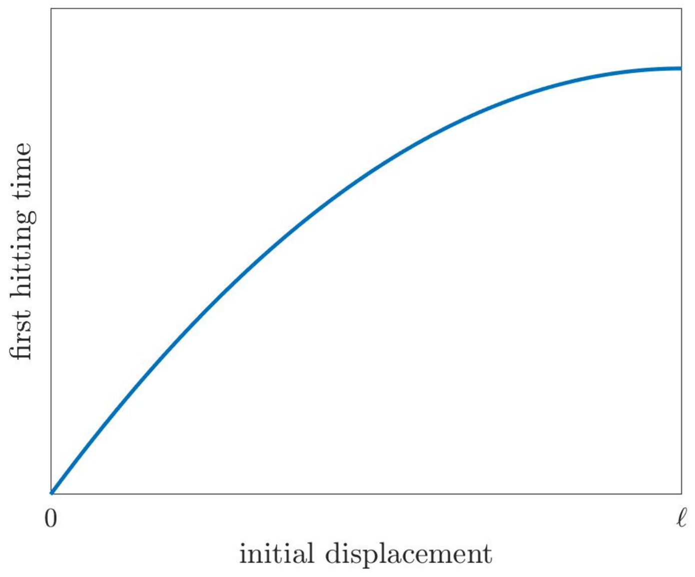

Fig. 1.3 The relationship between initial displacement and expected first hitting time given by the simple random walk model

Furthermore, if we write our initial position, n, in terms of the length of the domain and define the initial displacement as x_0, we can write our expected first passage time as

$$h(x_0) = \frac{x_0(2\ell - x_0)}{2D}. \tag{1.7}$$

In much the same way, we can use a random walk to solve the first passage probability problem. We find that the probability of successful binding occurring before the reflective boundary is hit, $p_{\text{bind}}(x_0)$ is equal to

$$p_{\text{bind}}(x_0) = \frac{\ell - x_0}{\ell}. \tag{1.8}$$

We shall compare these results to the results of our illustrative simulations in subsequent chapters.

Chapter 2

The Rouse Model

2.1 Model definition

The Rouse model is a simple model of the dynamics of an ideal chain, proposed by Prince E. Rouse in 1953, which describes the dynamics of short non-entangled polymer chains very successfully. It is a bead-and-spring model, so monomers are represented by spherical beads whilst the inter-connecting bonds are represented by harmonic springs. It will be instructive for our analysis to first formally define the concept of an *ideal chain*.

Definition 2.1.1. In an ideal chain model, every subsequent bead is placed uniformly at random on the sphere with radius b, centred on the previous bead. Consequently, the first two moments of each bond vector are

$$\langle (\mathbf{r}_{i+1} - \mathbf{r}_i) \rangle = 0 \tag{2.1}$$

$$\langle (\mathbf{r}_{i+1} - \mathbf{r}_i)^2 \rangle = b^2. \tag{2.2}$$

Furthermore, the bond vectors are all pairwise independent.

The dynamics of such a chain can then be described by a system of SDEs

Definition 2.1.2. Let $N > 1$ be a positive integer. The Rouse model consists of a system of $N+1$ identical beads of radius σ, with positions $\mathbf{r}_i$ (for $i = 1, 2, \ldots, N+1$), connected by N springs of stiffness k and Kuhn lengths b. Adjacent beads interact through a potential

$$U\left(|\mathbf{r}_{i\pm1} - \mathbf{r}_i|\right) = \frac{3}{2b^2} k_B T |\mathbf{r}_{i\pm1} - \mathbf{r}_i|^2,$$

where T is absolute temperature and k_B is Boltzmann's constant. By comparing this to the conventional potential for a harmonic spring, we can relate the spring stiffness to the Kuhn length by

$$k = \frac{3k_BT}{b^2}.$$

The positions of the beads evolve according to a system of SDEs

$$\gamma \mathrm{d}\mathbf{r}_i = k\left(\mathbf{r}_{i-1}(t) - 2\mathbf{r}_i(t) + \mathbf{r}_{i+1}(t)\right)\mathrm{d}t + \sqrt{2k_BT\gamma}\,\mathrm{d}\mathbf{W}_i, \tag{2.3}$$

where $\gamma = 6\pi\eta\sigma$ is the frictional drag coefficient given by Stokes' Theorem and $\mathrm{d}\mathbf{W}_i$ are independent Wiener processes.

2.2 Chain statistics

2.2.1 Self diffusion constant

The *self diffusion constant* is an extension of the traditional definition of a diffusion constant to the centre of mass of the polymer chain, $\mathbf{r}_G(t)$, which is defined as it typically is for multi-particle systems in classical mechanics as

$$\mathbf{r}_G(t) = \frac{1}{N+1}\sum_{n=1}^{N+1} \mathbf{r}_n(t). \tag{2.4}$$

By summing the system of SDEs in Equation 2.3, the time evolution of the centre of mass can be described by the SDE

$$\mathrm{d}\mathbf{r}_G = \frac{1}{N+1}\sqrt{\frac{2(N+1)k_BT}{\gamma}}\mathrm{d}\mathbf{W}, \tag{2.5}$$

where we have used the property that the sum of independent Wiener processes is another Wiener process with variance equal to the sum of the original processes' variances. Equation 2.5 is simply geometric Brownian motion with no drift, and so we can equate the coefficient of the Wiener process to the typical coefficient of $\sqrt{2D}$ to define the self diffusion constant D_G as

$$\begin{aligned} D_G &= \frac{k_B T}{(N+1)\gamma} \\ &= \frac{k_B T}{6\pi\eta\sigma(N+1)}. \end{aligned} \tag{2.6}$$

2.2.2 End-to-end distance

A second important statistic for our chain is the expected magnitude of the end-to-end vector $\mathbf{R} = \mathbf{r}_{N+1} - \mathbf{r}_1$, defined by

$$\rho = \langle \mathbf{R}^2 \rangle^{1/2} \tag{2.7}$$

If the dynamics described by the Rouse model are to be compatible with the definition of an ideal chain, they should preserve the expected value of ρ predicted by the ideal chain model. If the chain is long enough, we can apply the central limit theorem to the sum of the bond vectors

$$\begin{aligned} \mathbf{R} = \mathbf{r}_{N+1} - \mathbf{r}_1 &= \sum_{i=1}^{N} (\mathbf{r}_{i+1} - \mathbf{r}_i) \\ &\sim \mathscr{N}(0, Nb^2), \end{aligned}$$

enabling us to evaluate ρ as

$$\rho = \langle \mathbf{R}^2 \rangle^{1/2} = \sqrt{Nb^2} = b\sqrt{N}. \tag{2.8}$$

Now returning our attention to the system of SDEs in Equation 2.3, we define the R-dimensional vector of bonds as

$$\vec{\mathbf{y}}(t) = [\mathbf{r}_2(t) - \mathbf{r}_1(t), \mathbf{r}_2(t) - \mathbf{r}_2(t), \ldots, \mathbf{r}_{N+1}(t) - \mathbf{r}_N(t)]^T .$$

By subtracting adjacent SDEs, we can then rewrite the system in matrix form as $\mathrm{d}\vec{\mathbf{y}} = A\vec{\mathbf{y}}\mathrm{d}t + B\mathrm{d}\vec{\mathbf{W}}$, where $A \in \mathbb{R}^{N\times N}$ is the matrix given by

$$A = \begin{pmatrix} -\frac{2k}{\gamma} & \frac{k}{\gamma} & 0 & 0 & \dots & 0 \\ \frac{k}{\gamma} & -\frac{2k}{\gamma} & \frac{k}{\gamma} & 0 & \dots & 0 \\ 0 & \frac{k}{\gamma} & -\frac{2k}{\gamma} & \frac{k}{\gamma} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -\frac{2k}{\gamma} \end{pmatrix},$$

$B \in \mathbb{R}^{N\times N}$ is the diagonal matrix given by

$$B = \begin{pmatrix} 2\sqrt{\frac{k_BT}{\gamma}} & 0 & 0 & \dots & 0 \\ 0 & 2\sqrt{\frac{k_BT}{\gamma}} & 0 & \dots & 0 \\ 0 & 0 & 2\sqrt{\frac{k_BT}{\gamma}} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 2\sqrt{\frac{k_BT}{\gamma}} \end{pmatrix},$$

and $d\vec{\mathbf{W}}$ is an N-dimensional vector of independent 3-dimensional Wiener processes. We define the stationary covariance matrix by

$$C = \lim_{t\to\infty} \langle \vec{\mathbf{y}}\, \vec{\mathbf{y}}^T \rangle,$$

and, borrowing from linear control theory [7], we can use the well-established result that C is the unique solution to the Lyapunov equation $AC + CA^T + BB^T = 0$. Since B is diagonal and A is symmetric, we can rewrite this as $2AC = -BB^T$ which we can easily solve to find that C is a diagonal matrix with diagonal entries c, satisfying

$$\begin{aligned} \frac{-2kc}{\gamma} &= \frac{-4k_BT}{\gamma} \\ \implies \qquad c &= \frac{2k_BT}{k} \\ &= b^2. \end{aligned} \tag{2.9}$$

This implies that the bond vectors $\mathbf{r}_{i+1} - \mathbf{r}_i$ are pairwise independent with long-time variance equal to b^2. Hence, it follows that the long-time limit of the rms end-to-end distance given by the dynamics of the Rouse model is equal to

$$\rho_\infty = \sqrt{N \times b^2} = b\sqrt{N} = \rho, \tag{2.10}$$

as we had hoped.

Having defined these two quantities, we shall show that they are as predicted in our simulations and shall use this as a way of demonstrating the efficacy of our next model in Chapter 3.

2.3 Simulation

We now return our focus to the first passage time problem discussed in Chapter 1 and look to improve on our initial estimate given by the random walk with reflective boundary. To do so, we use the Rouse model we have just defined. We initialize our chain by placing the first bead on the origin and iteratively placing every subsequent bead uniformly at random on the sphere of radius b, centred on the previous bead. This is referred to as the freely jointed chain (FJC) model in the literature. We can express our update rule in computational form using the Euler-Maruyama method as

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \frac{k}{\gamma}\left(\mathbf{r}_{i-1}(t) - 2\mathbf{r}_i(t) + \mathbf{r}_{i+1}(t)\right)\Delta t + \sqrt{\frac{2k_BT\Delta t}{\gamma}}\xi_i. \tag{2.11}$$

Meanwhile, we allow the protein to move under simple Brownian motion

$$\mathbf{X}(t+\Delta t) = \mathbf{X}(t) + \sqrt{2D_p\Delta t}\xi, \tag{2.12}$$

and in order for our problem to retain biological relevance, we assume that the protein in question is the TATA-binding protein (TBP). TBP is a transcription factor that binds the promoter region of DNA in order to initiate the formation of the transcription preinitiation complex - a vital step in positioning RNA polymerase II at the gene transcription start site. TBP has a saddle shape that facilitates its docking to the DNA, as can be seen in Figure 2.1. However, for the purpose of this project, we consider the protein in a low level of detail and simply treat it as a sphere of radius 25 Å or 2.5 nm in our simulations. As well as this, we use $k_B = 1.4 \times 10^{-23}\ \mathrm{m^2\,kg\,s^{-2}\,K^{-1}}$, $T = 300$ K and $\eta = 0.001\ \mathrm{kg\,m^{-1}\,s^{-1}}$ as values of the Boltzmann constant, absolute temperature and the viscosity of water respectively.

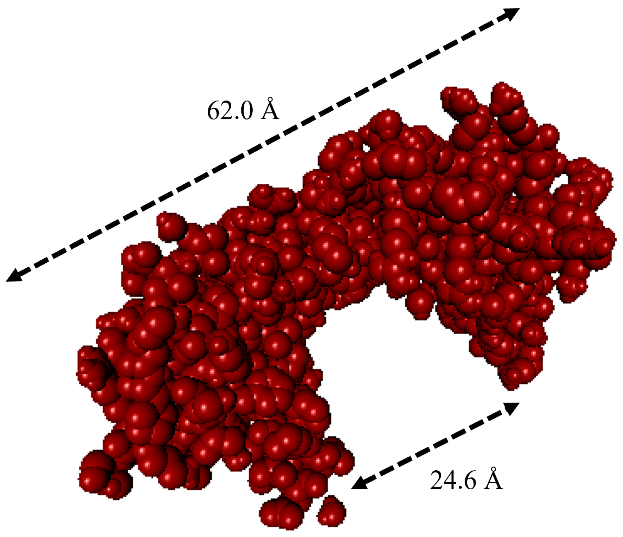

Fig. 2.1 VMD visualization of a TBP with annotations showing key dimensions. PDB file taken from [6].

At each timestep, after updating the particles' positions, if the minimum distance between the protein and a bead on the Rouse chain is less than the binding radius d_b, we assume that binding successfully occurs with probability 1 and we stop the simulation and record the time taken. Conversely, if the minimum distance is more than or equal to the reflective boundary radius d_∞, we backtrack and impose reflection on this boundary instead.

We carry out 1,000 simulations for each value in a range of starting separations d_0 and record both the average time taken for successful binding to occur and whether or not binding would have occurred without a reflective boundary. We now examine the results of our simulations.

2.4 Results

Returning briefly to the analytical model we discussed in Section 1.1, we can now assign values to the parameters we used. For the diffusion coefficient D, it makes sense to use the sum of the protein's diffusion coefficient D_p and the self-diffusion coefficient of the Rouse chain D_G, since we consider the DNA to be fixed on the origin so the relative movement of the protein should be the sum of the movement if they were both free to move. Furthermore, we use the binding radius d_b as the radius of the adsorbing sphere representing the DNA. This means that the length of the 1-dimensional domain for the random walk model is $\ell = d_\infty - d_b$ and we can use our analytical estimate for the values of $x_0 = d_0 - d_b$ to predict the first passage time in terms of d_0 as

Table 2.1 Parameter selection for the Rouse DNA binding model

Symbol	**Name**	**Value**	**Justification**
N	Number of springs	900	To match the literature
σ	Bead radius	1.2 nm	Chosen in the literature
b	Kuhn length	60 nm	Chosen in the literature
d_b	Binding radius	10 nm	Arbitrary
d_∞	Radius of reflective boundary	1 µm	Approximate average nucleolus radius
d_0	Initial radius	$1 \times 10^{-2+\frac{j}{4}}$ µm $j = 0, 1, \dots, 8$	Gives range between d_b and d_∞
D_p	Protein diffusion coefficient	89 µm^2 s^{-1}	Estimated using TBP radius and Stokes-Einstein relation
Δt	Timestep	10 ps	Scaled with D_p to match $D_p \times \Delta t$ in the literature

$$h(d_0) = \frac{(d_0 - d_b)(2d_\infty - d_b - d_0)}{2(D_p + D_G)}. \tag{2.13}$$

In Figure 2.2, we compare the results of our simulations to Equation 2.13 and see that, whilst the shape of the curve is similar between the two, the analytical model predicts a smaller first passage time than we observed across the whole range of values of d_0. We discuss this further in our analysis in Chapter 6.

As expected, the observed macroscopic properties of the DNA filament align with our analytical estimates made earlier in the chapter. As we turn our attention to improving the efficiency of the Rouse model, we shall keep in mind that the preservation of these macroscopic properties is a good way to check that we are maintaining the underlying mechanics of the original model.

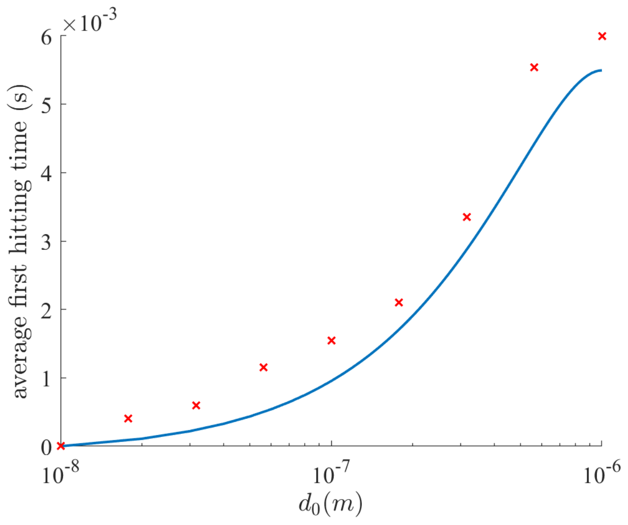

Fig. 2.2 Comparison between the results of our simulations and the prediction of the analytical model from Chapter 1 for the first passage time problem

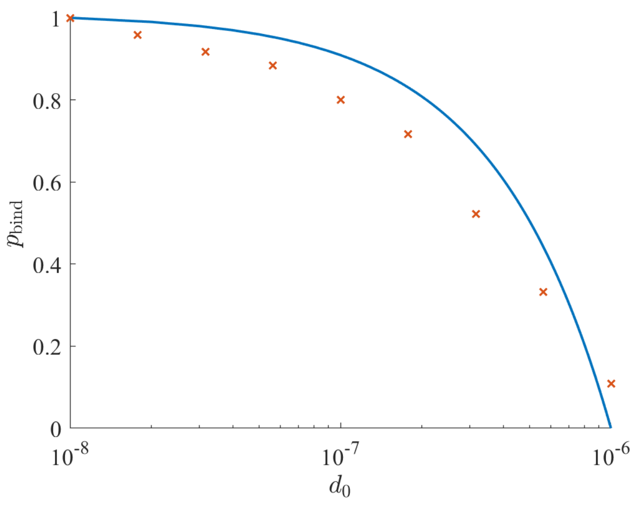

Fig. 2.3 Comparison between the results of our simulations and the prediction of the analytical model from Chapter 1 for the first passage probability problem

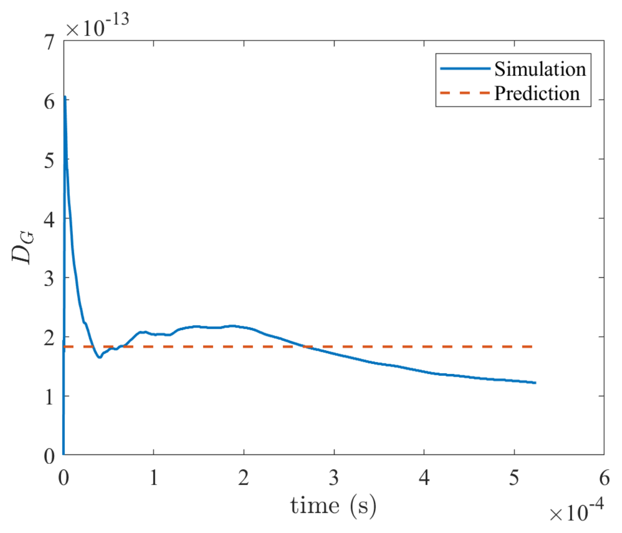

Fig. 2.4 Comparison between the observed self-diffusion constant and the expected self-diffusion constant for one instance of our simulation

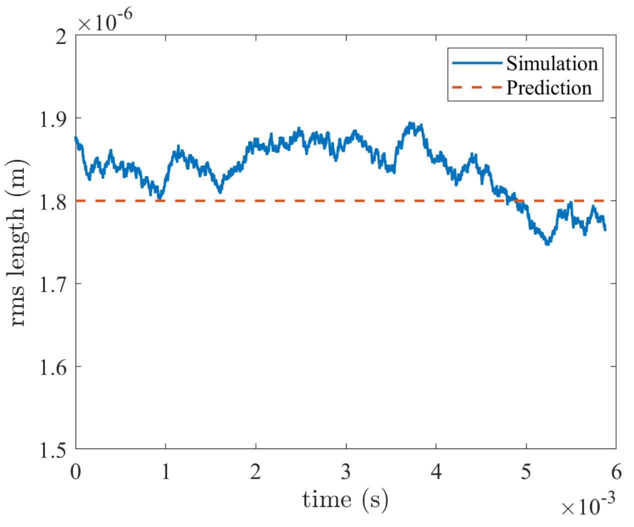

Fig. 2.5 Comparison between the observed RMS length and the expected length for one instance of our simulation

Chapter 3

Adaptive resolution model

The Rouse model presented in the previous chapter has its strengths. In particular, it is conceptually simple and requires only a few lines of code. However, in order to avoid the protein taking excessively large jumps (i.e. jumping from one side of the chain to the other without any binding registering), we have to take a very small timestep of $\Delta t = 1 \times 10^{-11}$ s. This makes the Rouse model extremely slow to run, which is an obvious deterrent for most investigators.

We can clearly do better. Although a small timestep is certainly needed when the DNA and protein are close together, do we need one when they are far apart? Furthermore, do we really need to keep track of so many beads at every timestep?

The answer to both of these questions is "no" and this has been demonstrated by the introduction of schemes that vary temporal and spatial resolution on the fly which have been explored in the literature [10] and which we now explore.

3.1 Model definition

The underlying principle of the adaptive resolution model (henceforth referred to as AdRes for short) is that we can start out with fewer, larger beads which we update using bigger timesteps until the protein becomes sufficiently close, at which point we increase resolution by an integer resolution factor s. More specifically, we divide the chain up into N sections, each corresponding to one spring of Kuhn length b, with an associated bead radius of σ and timestep s^4. When the protein gets near, we increase the spatial resolution in sections of the chain that are sufficiently near by replacing the section's spring with s^2 springs of Kuhn length b/s and associated bead radius of σ/s^2 and timestep 1. These relationships are chosen in the literature to preserve key chain statistics like self diffusion constant and rms end-to-end distance in both levels of resolution.

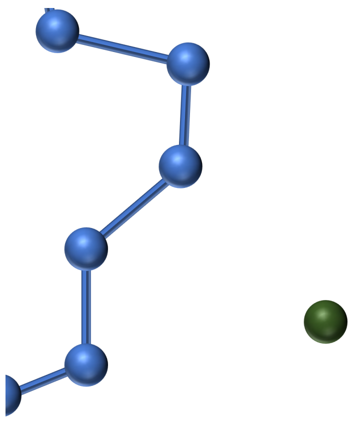
Fig. 3.1 Rouse chain in region of low resolution

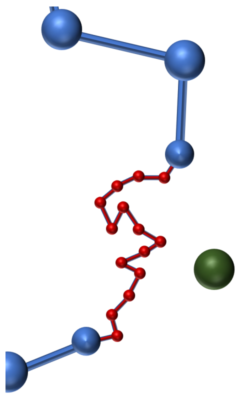
Fig. 3.2 Rouse chain in region of high resolution

Unlike our simple Rouse model, we now have to keep track of the timestep, bead radius, Kuhn length and resolution in every section, so define a set of vectors called the *springs vectors* which we denote by $\mathbf{w}_i$, where each vector corresponds to one spring

$$w_{i,1} = \text{timestep} = \begin{cases} s^4 & \text{in low resolution} \\ 1 & \text{in high resolution} \end{cases}$$

$$w_{i,2} = \text{Kuhn length} = \begin{cases} sb & \text{in low resolution} \\ b & \text{in high resolution} \end{cases}$$

$$w_{i,3} = \text{radius} = \begin{cases} s^s\sigma & \text{in low resolution} \\ \sigma & \text{in high resolution} \end{cases}$$

$$w_{i,4} = \text{resolution} = \begin{cases} 0 & \text{in low resolution} \\ 1 & \text{in high resolution} \end{cases}$$

We assume that a bead has radius equal to the average of the radii associated with the two springs either side of it. We use this detail to calculate the drag term γ_i at each step. We also assume that the first and last beads have radius $(\sigma + w_{1,3})/2$ and $(\sigma + w_{N(t),3})/2$ respectively, where $N(t)$ is the current number of springs.

We also keep track of the original $N+1$ beads in a separate set of vectors $\mathbf{q}_i$, which we call the boundary bead vectors.

3.2 Using multiple timescales

Using different timescales for different springs means that we have to consider the timescale of each adjacent spring when updating the position of a bead. We summarise the method for doing this at the nth step of our simulations below. We set $N(t) = N$ and $\mathbf{r}_i(t)$ by $\mathbf{r}_i^t$ for ease of notation.

Algorithm 1: Updating the DNA chain over one timestep

```
if $w_{1,1}|n$ then
    Put $\mathbf{r}_1^{n\Delta t} = \mathbf{r}_1^{(n-1)\Delta t} + \frac{k_1}{\gamma_1}\left(\mathbf{r}_2^{(n-w_{1,1})\Delta t} - \mathbf{r}_1^{(n-w_{1,1})\Delta t}\right) w_{1,1}\Delta t + \sqrt{\frac{2k_BT}{\gamma_1} w_{1,1}\Delta t}\,\xi_1$
else
    Put $\mathbf{r}_1^{n\Delta t} = \mathbf{r}_1^{(n-1)\Delta t}$
if $w_{N-1,1}|n$ then
    Put $\mathbf{r}_N^{n\Delta t} =$
    $\mathbf{r}_N^{(n-1)\Delta t} + \frac{k_{N-1}}{\gamma_{N-1}}\left(\mathbf{r}_{N-1}^{(n-w_{N-1,1})\Delta t} - \mathbf{r}_N^{(n-w_{N-1,1})\Delta t}\right) w_{N-1,1}\Delta t + \sqrt{\frac{2k_BT}{\gamma_{N-1}} w_{N-1,1}\Delta t}\,\xi_N$
else
    Put $\mathbf{r}_N^{n\Delta t} = \mathbf{r}_N^{(n-1)\Delta t}$
for $i = 1, 2, \ldots, N-2$ do
    if $\min\{w_{i,1}, w_{i+1,1}\}|n$ then
        Put $\mathbf{r}_i^{n\Delta t} = \mathbf{r}_i^{(n-1)\Delta t} + \sqrt{\frac{2k_BT}{\gamma_i} \min\{w_{i,1}, w_{i+1,1}\}\Delta t}\,\xi_i$
        if $w_{i,1}|n$ then
            Put $\mathbf{r}_i^{n\Delta t} = \mathbf{r}_i^{n\Delta t} + \frac{k_i}{\gamma_i}\left(\mathbf{r}_{i-1}^{(i-wi,1)\Delta t)} - \mathbf{r}_i^{(i-wi,1)\Delta t)}\right) w_{i,1}\Delta t$
        if $w_{i+1,1}|n$ then
            Put $\mathbf{r}_i^{n\Delta t} = \mathbf{r}_i^{n\Delta t} + \frac{k_{i+1}}{\gamma_i}\left(\mathbf{r}_{i+1}^{(i-wi+1,1)\Delta t)} - \mathbf{r}_i^{(i-wi+1,1)\Delta t)}\right) w_{i+1,1}\Delta t$
    else
        Put $\mathbf{r}_i^{n\Delta t} = \mathbf{r}_i^{(n-1)\Delta t}$
```

3.3 Introducing and removing beads

If the diffusing protein becomes less than r_I away from a boundary bead, we increase the resolution in that region by introducing $s^2 - 1$ new springs between $\mathbf{r}_A$ and $\mathbf{r}_B$ such that all beads between $\mathbf{r}_A$ and $\mathbf{r}_B$ lie a distance of d' apart, where $d' = |\mathbf{r}_B - \mathbf{r}_A|/s$. To introduce new beads, we sample from the probability function of acceptable chains using the Metropolis-

Hastings algorithm. This is described in detail in [10] so we do not explore it fully here as the technique is a well-established Monte Carlo method.

Equivalently, when the protein moves away from the DNA filament, we zoom out so as to increase computational efficiency again by returning to a low resolution regime. We define the threshold for this via the zoom out factor Z where we zoom out if the distance between the boundary beads and the protein becomes greater than Zr_I.

3.4 Simulation and parameters

We conduct 1,000 simulations using Algorithm 2 for the same range of initial displacements d_0 as we had for the simple Rouse model. For our new parameters, we use the following values

Table 3.1 Parameter selection for AdRes model

Symbol	Name	Value
s	Resolution factor	3
Z	Zoom out factor	5
r_I	Resolution increase radius	0.5 µm

Algorithm 2: Summary of AdRes algorithm

```
Update all $\mathbf{r}_i$ according to Algorithm 1
for $i = 1, 2, \ldots, N$ do
    if Region i is in low resolution then
        if $\min_{j=i,i+1} |\mathbf{q}_j - \mathbf{X}| < r_I$ then
            Introduce new beads and springs in high temporal resolution using
              Metropolis-Hastings algorithm
    if Region i is in high resolution then
        if $\min_{j=i,i+1} |\mathbf{q}_j - \mathbf{X}| > Zr_I$ then
            Revert to low resolution by deleting beads and springs
Output: First passage time and probability
```

3.5 Results

The results of our simulations are shown in Figure 3.3 and Figure 3.4. We find that, on average, our simulations show a 3.4-times improvement in speed on an AMD FX(tm)-4350 Quad-Core Processor from the simple Rouse model. Furthermore, the observed self-diffusion constant is unaffected by the variation of resolution and is still consistent with the expected result from the analysis in Chapter 2.

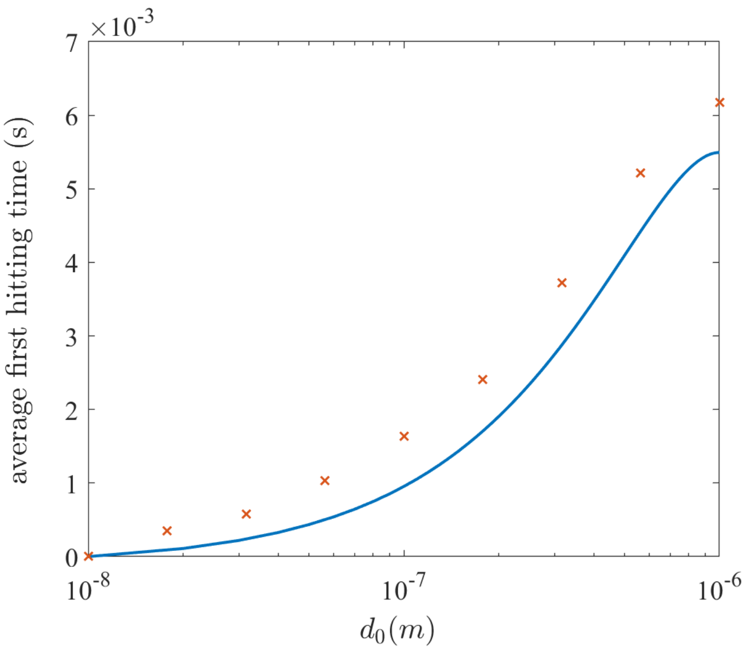

Fig. 3.3 AdRes first passage time problem

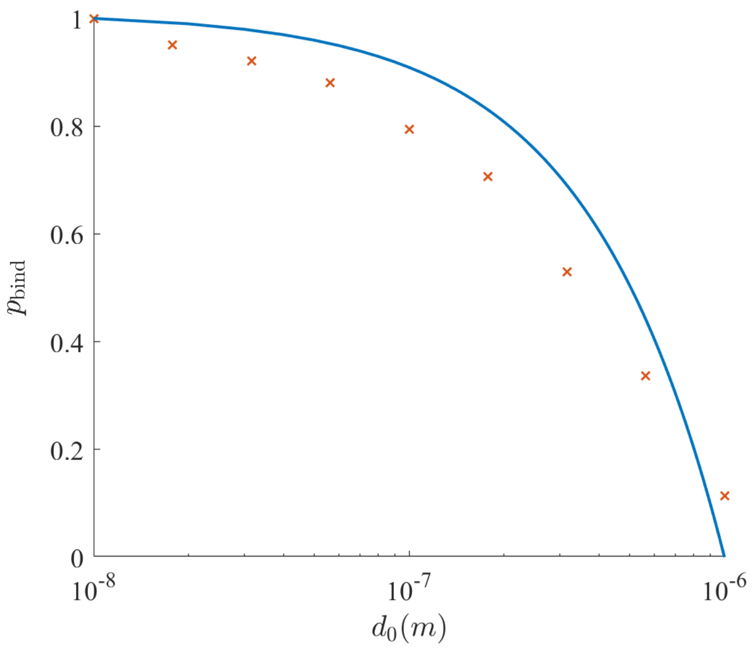

Fig. 3.4 AdRes first passage probability problem

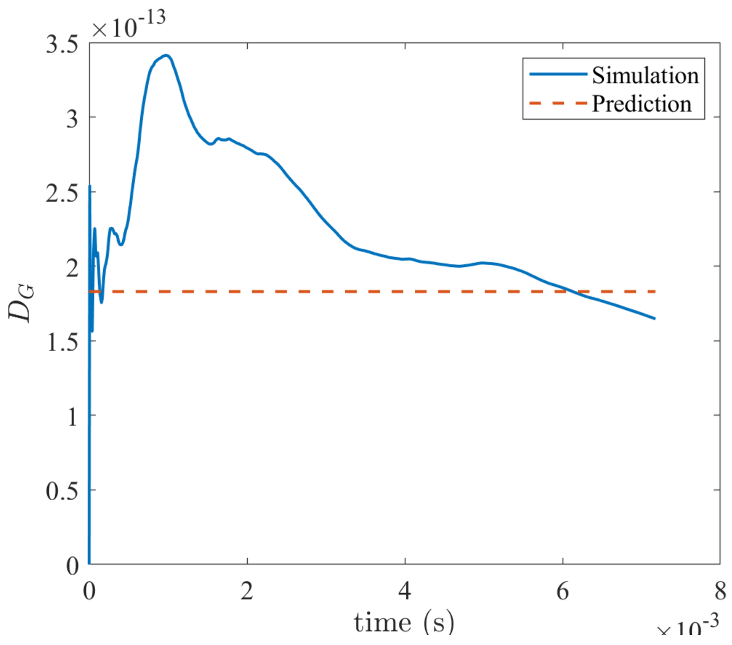

Fig. 3.5 Comparison between observed self-diffusion constant and prediction from Chapter 2 in one instance of our simulations

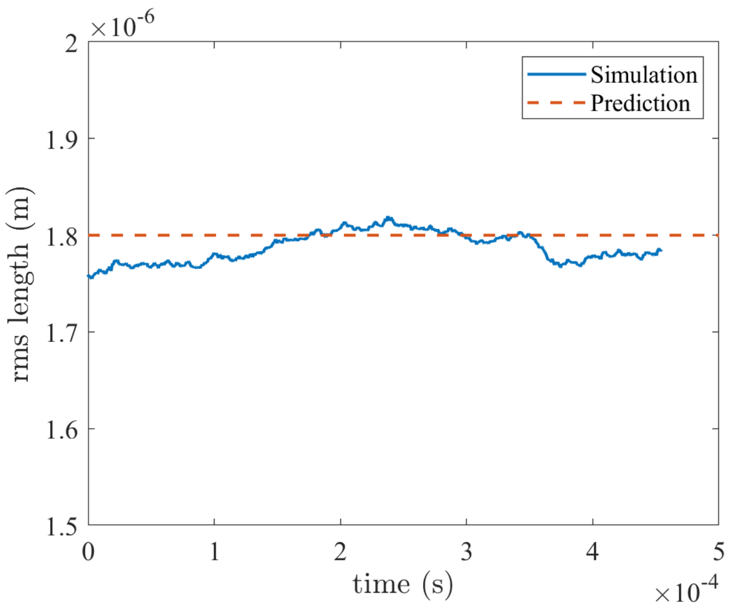

Fig. 3.6 Comparison between observed RMS length and prediction from Chapter 2 in one instance of our simulations

Chapter 4

The hard sphere heat bath

In this chapter, we present a theoretical molecular dynamics model which considers only extremely short-range interactions between a diffusing particle and its surrounding solvent through fully elastic collisions. This theory will form the basis of our final model of DNA-protein interactions in Chapter 5. In the literature, this interaction is often referred to as the hard-sphere potential [1] and has a lot of overlap with the model of an ideal gas that has been popular since Bernoulli's pioneering 1738 work, *Hydrodynamica*.

4.1 Model definition

We consider our diffusing particle to be a sphere of radius R and mass M, immersed in a solvent represented by many identical smaller point particles (i.e. no volume exclusion effects occur between the solvent), each of mass m. We denote the ratio of the mass of the diffusing particle to the mass of the heat bath particles by $\mu := \frac{M}{m}$. This will be a critical quantity which we shall seek to maximise in order to improve the physical relevance of our model. The solvent particles are distributed with density, λ_μ, defined as

$$\lambda_\mu = \frac{3}{8R^2}\sqrt{\frac{(\mu+1)\gamma}{2\pi MD}}, \tag{4.1}$$

where $\gamma = 6\pi\eta R$ is the drag force given by Stokes' Law. Intuitively, this means that for a more viscous solvent (i.e. with higher η) we have a greater density of heat bath particles.

To initlialize the solvent particles in our computational domain, Ω (assumed here to be a cube of length L), we sample from a Poisson distribution with mean $\lambda_\mu|\Omega|$ to get the initial number of particles $N(0)$. We then initialize the position of each particle uniformly in $[0,L]^3$ and remove any particles that lie inside the diffusing sphere, so

$$\mathbf{x}_i(0) \sim U(0,L)^3 \tag{4.2}$$

We wish to randomize the initial direction of velocity of each heat bath particle. However, we wish to control the second moment of their velocities in order to maintain a constant temperature in the domain. As a result, we sample each component of their velocities from a Gaussian distribution with mean 0 and variance equal to

$$\langle \mathbf{v}_{i,x}^2 \rangle = \langle \mathbf{v}_{i,y}^2 \rangle = \langle \mathbf{v}_{i,z}^2 \rangle = \omega_\mu^2 = (\mu+1)D\frac{\gamma}{M}. \tag{4.3}$$

The choice for the second moment of velocities given by Equation 4.3 means that the average kinetic energy of the solvent particles is

$$\begin{aligned} E_{KE} &= 3 \times \frac{1}{2} \times m \times (\mu+1) \times \frac{k_B T}{6\pi\eta R} \times \frac{6\pi\eta R}{M} \\ &= \frac{3}{2}\left(\frac{1}{\mu}+1\right) k_B T \\ &\longrightarrow \frac{3}{2} k_B T \qquad \text{as } \mu \to \infty \end{aligned}$$

where we have used the Stokes-Einstein relation in the first line to rewrite our diffusion coefficient [3]. This is the average kinetic energy predicted by the equipartition theory for a particle at temperature T. Hence, in the large μ limit we shall be considering, this choice of $\langle \mathbf{v}_i^2 \rangle$ gives us the desired physical conditions. Once we have initialized the solvent, we set the diffusing particle at rest in the centre of the computational domain.

We use an event-driven algorithm by iteratively finding the time to the next collision between the diffusing particle and a heat bath particle and updating the system at this time. We show later that this approach offers significant improvements in the computational efficiency of this model.

For each solvent particle, the time to the next collision is the smallest root of the quadratic

$$\begin{gathered} \mathbf{V} - \mathbf{v}_i = \mathbf{a} \\ \mathbf{X} - \mathbf{x}_i = \mathbf{b} \\ \tau_i^2\, \mathbf{a}\cdot\mathbf{a} + 2\tau_i\, \mathbf{a}\cdot\mathbf{b} + \mathbf{b}\cdot\mathbf{b} - R^2 = 0. \end{gathered} \tag{4.4}$$

If the two particles are moving apart and will never collide, $\mathbf{a}\cdot\mathbf{b}$ is positive and we do not evaluate the roots of Equation 4.4. Furthermore, if the particles are moving towards one another but are on course to miss, the discriminant is negative and again we ignore these solutions. For all remaining particles, we get strictly positive, real roots. We take the smaller of the two roots since this corresponds to the time that the solvent particle enters the diffusing particle, whilst the larger root corresponds to the time it would exit the interior of the larger particle.

Setting the timestep as $\Delta t = \min_{i=1,\ldots,N(t)} \tau_i$, we update all particles' positions according to their free-flight equations

$$\mathbf{X}(t+\Delta t) = \mathbf{X}(t) + \mathbf{V}(t)\Delta t \tag{4.5}$$

$$\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \mathbf{v}_i(t)\Delta t \tag{4.6}$$

$$i = 1,\ldots,N(t).$$

Particles' velocities remain unchanged, except for the diffusing particle and the solvent particle it collides with. Assuming our collisions are fully elastic, we use the conservation of total linear momentum and kinetic energy to derive the post-collision velocities of the solvent and diffusing particle ($\tilde{\mathbf{v}}$ and $\tilde{\mathbf{V}}$ respectively) in terms of the particles' normal and tangential velocities prior to the collision

$$\tilde{\mathbf{v}} = \mathbf{v}_t + \frac{1-\mu}{\mu+1}\mathbf{v}_n + \frac{2\mu}{\mu+1}\mathbf{V}_n, \tag{4.7}$$

$$\tilde{\mathbf{V}} = \mathbf{V}_t + \frac{\mu-1}{\mu+1}\mathbf{V}_n + \frac{2}{\mu+1}\mathbf{v}_n, \tag{4.8}$$

where the normal and tangential velocities are illustrated in Figure 4.1.

4.2 Co-moving frame

After each collision, we reposition the boundary of Ω (the *frame*) so that the diffusing particle is always at the centre of the domain. The reasons for doing this will become clear once we start using this model for simulations. We define the frame velocity to be

$$\mathbf{V}_f = \frac{\mathbf{X}(t+\Delta t) - \mathbf{X}(t)}{\Delta t}. \tag{4.9}$$

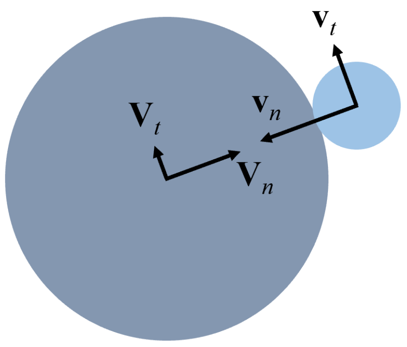

Fig. 4.1 Diagram showing the tangential and normal components of the pre-collision velocities of the colliding particles (not to scale).

4.3 Boundary conditions

The final piece of the jigsaw is to determine suitable boundary conditions for our domain. Multiple options exist, including periodic boundary conditions or reflective boundaries. We wish to use our simulation to increase the resolution in a subdomain of the nucleus, so it makes sense for us to consider our cubic domain Ω as subdomain of a much larger domain which we treat as all of $\mathbb{R}^3$. With this in mind, we can imagine solvent particles to be distributed with density λ_μ in the space around each surface of the cube.

We must consider each surface of the cube separately, since the movement of the frame means that we must introduce more heat bath particles across the surfaces in which direction the frame has moved.

Let us consider our domain at an arbitrary time during our simulation. Without loss of generality, we let Ω be the region $[0,L] \times [0,L] \times [0,L]$. We consider how many particles to introduce across the surface at $x = 0$.

In order to introduce a new particle with x-coordinate $\tilde{x}$, it must have sufficient velocity in the x direction to have travelled from $(-\infty, 0)$ to $\tilde{x} \in (0, \infty)$ and to have overtaken the frame, so to speak. Thus, the particle's x velocity, v_x, must be at least $V_{f,x} + \frac{\tilde{x}}{\Delta t}$, where $V_{f,x}$ is the x component of the frame's velocity.

With this in mind, the expected density, $g(\tilde{x})$, of new particles with x-coordinate $\tilde{x}$ can be evaluated as

$$\begin{aligned} g(\tilde{x}) &= \int_{-\infty}^{\infty} H\left(v_x\Delta t - \tilde{x} - V_{f,x}\Delta t\right) \frac{\lambda_\mu}{\omega_\mu\sqrt{2\pi}} \exp\left(-\frac{v_x^2}{2\omega_\mu^2}\right) \mathrm{d}v_x \\ &= \frac{\lambda_\mu}{2} \operatorname{erfc}\left(\frac{\tilde{x} + V_{f,x}\Delta t}{\sqrt{2}\omega_\mu\Delta t}\right), \end{aligned} \tag{4.10}$$

Integrating Equation 4.10 over all possible new coordinates in domain $(x^*, \infty) \times [y^*, y^* + L] \times [z^*, z^* + L]$ gives us the expected number of particles, m_{new}, that have entered our domain across this face as

$$\begin{aligned} m_{\text{new}} &= \frac{\lambda_\mu L^2}{2} \int_0^\infty g(\tilde{x})\mathrm{d}\tilde{x} \\ &= \lambda_\mu L^2 \Delta t \left(\frac{\omega_\mu}{\sqrt{2\pi}} \exp\left[-\frac{V_{f,x}^2}{2\omega_\mu^2}\right] - \frac{V_{f,x}}{2} \operatorname{erfc}\left[\frac{V_{f,x}}{\omega_\mu\sqrt{2}}\right] \right). \end{aligned} \tag{4.11}$$

To implement this in our simulations, we generate the number of particles to introduce from a Poisson distribution with mean m_{new}. For each new particle, we then initialize its x coordinate by sampling from the distribution $\sqrt{\pi}\operatorname{erfc}(\chi)$ using the algorithm in Appendix A and setting

$$x_{\text{new}} = \sqrt{2}\omega_\mu \Delta t \chi.$$

The particle's y and z coordinates are sampled uniformly in (0,L). Finally, we need to initialize the new particle's velocity. We wish for the velocities to still be distributed according to a normal distribution with variance ω_μ^2 but need the particle's x component of velocity to be sufficient to allow it to have travelled from outside the domain to inside it, being careful to consider the fact that the frame has also moved in the previous timestep. In other words, we need

$$v_x \geq \frac{x_{\text{new}}}{\Delta t} + V_{f,x} = \sqrt{2}\omega_\mu \chi + V_{f,x} \tag{4.12}$$

Hence, we sample from a truncated normal distribution $\zeta \sim H(\xi - \sqrt{2}\omega_\mu\chi - V_{f,x})\xi$ (where $\xi \sim \mathcal{N}(0,1)$ using the acceptance-rejection algorithm in Appendix A and set $v_x = \omega_\mu \zeta$. The particle's y and z velocities are sampled from a standard normal distribution with mean 0 and variance ω_μ^2.

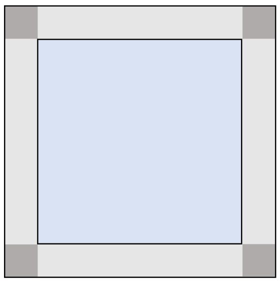

Fig. 4.2 2D illustration of overcounting problems. Areas in light grey are double counted, whilst areas in dark grey are triple counted.

4.4 Overcounting

When using these boundary conditions, we must be wary of the potential for overcounting by considering particles on edges of the cube twice and particles on vertices 3 times, as shown in Figure 4.2. This is more of a significant issue when using a small value of L. However, we can easily overcome these issues by ensuring that we only accept particles from these overcounted regions in 1 of every 2 or 3 instances (respectively).

We adjust for this once we have sampled the velocity and position of a new particle by defining an acceptance probability

$$p_{acc}(\mathbf{x},\mathbf{v}) = \begin{cases} 1 & \mathbf{x}-\mathbf{v}\Delta t \in \mathscr{Y}_1 \\ 1/2 & \mathbf{x}-\mathbf{v}\Delta t \in \mathscr{Y}_2 \\ 1/3 & \mathbf{x}-\mathbf{v}\Delta t \in \mathscr{Y}_3, \end{cases} \tag{4.13}$$

where $\mathscr{Y}_i$ is the region where precisely i coordinates are in [-L/2, L/2]. We then simply generate a uniform random variable in [0,1] and accept the new particle if it is less than p_{acc}

4.5 Convergence to Langevin description

Our main motivation behind introducing this theoretical model is that it converges to the Langevin description

$$X_i(t+\mathrm{d}t) = X_i(t) + V_i(t)\mathrm{d}t \tag{4.14}$$

$$V_i(t+\mathrm{d}t) = V_i(t) - \beta V_i(t)\mathrm{d}t + \beta\sqrt{2D}\,\mathrm{d}W_i, \quad i = 1,2,3, \tag{4.15}$$

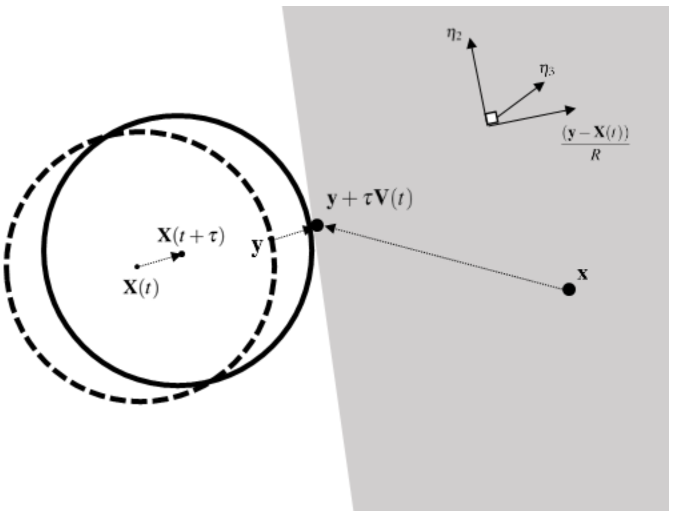

Fig. 4.3 Illustration of collision along with associated vectors

in the limit $\mu \to \infty$. This enables us to easily transition between the two models and thus vary the resolution in parts of the domain.

In order to prove this convergence, we consider describing the motion of the diffusing particle according to an SDE

$$V_i(t+\Delta t) = V_i(t) + \alpha_i \Delta t + \beta_i \sqrt{\Delta t}\,\xi_i, \tag{4.16}$$

where ξ_i are independent standard normal random variables for $i = 1,2,3$. We will show that this is equivalent to Equation 4.15.

A sensible way to go about finding α_i, β_i would be to evaluate the first two moments of the change in velocity and so we attempt to do this by introducing a probability density function $\psi_i(\mathbf{y})$ for the distribution of the change in the i-th component of velocity of the diffusing particle caused by collisions around the surface point $\mathbf{y}$ in an interval of length Δt. This means that the expected contribution of collisions in the time interval $(t,\, t+\Delta t)$ over the surface area $(\mathbf{y},\, \mathbf{y}+\mathrm{d}\mathbf{y})$ to the total change in the i-th velocity component is given by $\psi_i(\mathbf{y})\mathrm{d}\mathbf{y}$.

We consider what happens when a collision occurs at point $\mathbf{y}$, as illustrated in Figure Figure 4.3. We assume that the collision occurs at time $t+\tau$, where $\tau < \Delta t$. The heavy particle, moving with velocity $\mathbf{V}(\mathbf{t})$, collides with a heat bath particle which was previously at $\mathbf{x}$. We can express the velocity of the heat bath particle in terms of these variables as

$$\mathbf{v} = \frac{\mathbf{y} + \tau \mathbf{V}(t) - \mathbf{x}}{\tau} = \mathbf{V}(t) + \frac{(\mathbf{y} - \mathbf{x})}{\tau}. \tag{4.17}$$

We define the collision axis as the vector going through the centre of the sphere to the point of collision. It has normalized equation

$$\kappa = \frac{\mathbf{y} - \mathbf{X}(t)}{R}. \tag{4.18}$$

Using these equations and Equation 4.8, we can write the change in velocity of the diffusing particle as

$$\begin{aligned}
\tilde{\mathbf{V}} - \mathbf{V} &= \frac{2}{\mu + 1} [\mathbf{v}_n - \mathbf{V}_n] \\
&= \frac{2}{\mu + 1} [(\mathbf{v} - \mathbf{V}) \cdot \kappa] \, \kappa \\
&= \frac{2}{\mu + 1} \left[\frac{(\mathbf{y} - \mathbf{x})}{\tau} \cdot \frac{(\mathbf{y} - \mathbf{X}(t))}{R} \right] \frac{(\mathbf{y} - \mathbf{X}(t))}{R}.
\end{aligned}$$

We now consider which values $\mathbf{x}$ can take. In order to collide with the sphere as in Figure 4.3, the heat bath particle must have come from the half-space to the right hand side of the collision, represented by the shaded area. In order to simplify the algebra, we choose an orthonomal basis for this half-space containing the collision axis and two other arbitrary vectors η_1, η_2

$$B = \{\kappa, \eta_2, \eta_3\}.$$

We can then write $\mathbf{x}$ in terms of B as

$$\mathbf{x} = \mathbf{y} + \tau \mathbf{V}(t) + c_1 \tau \kappa + c_2 \tau \eta_2 + c_3 \tau \eta_3,$$

where $c_1 > 0$, $c_2 \in \mathbb{R}$, $c_3 \in \mathbb{R}$ are scalar coefficients. Substituting this into (4.19) yields

$$\tilde{\mathbf{V}} - \mathbf{V} = -\frac{2}{(\mu + 1)} (c_1 + \mathbf{V}(t) \cdot \kappa) \, \kappa, \tag{4.19}$$

where properties of orthonormal vectors are used to make the simplification.

In order to estimate the probability distribution function, we use the law of total expectation

$$\begin{aligned}\psi(\mathbf{y}) &= \mathbb{E}\left[\text{ change in velocity by all collisions at } \mathbf{y} \text{ in } (t, t+\Delta t)\right] \\ &= \int_t \int_{\mathbf{x}} \mathbb{E}\left[\text{ change}\ldots \mid \text{solvent came from } \mathbf{x}\right] \times \mathbb{P}\left(\text{solvent came from } \mathbf{x}\right) \mathrm{d}\mathbf{x}\, \mathrm{d}t.\end{aligned}$$

Using Equation 4.19 and the properties of the heat bath discussed earlier in the chapter, the j^{th} component of the pdf can be written as

$$\begin{aligned}\psi_j(\mathbf{y}) &= -\frac{2\lambda_\mu \kappa_j}{\omega_\mu^3\,(\mu+1)\,(2\pi)^{3/2}} \int_0^\infty \int_{-\infty}^\infty \int_{-\infty}^\infty \int_0^{\Delta t} (c_1 + \mathbf{V}(t)\cdot\kappa)^2 \times \\ &\qquad \exp\left[-\frac{c_1^2}{2\omega_\mu^2}\right] \exp\left[-\frac{c_2^2}{2\omega_\mu^2}\right] \exp\left[-\frac{c_3^2}{2\omega_\mu^2}\right] \mathrm{d}\tau\, \mathrm{d}c_3 \mathrm{d}c_2 \mathrm{d}c_1 \\ &= -\frac{2\lambda_\mu \kappa_j \Delta t}{\omega_\mu(\mu+1)\sqrt{2\pi}} \int_0^\infty (c_1 + \mathbf{V}(t)\cdot\kappa)^2 e^{-\frac{c_1^2}{2\omega_\mu^2}} \mathrm{d}c_1 \\ &= -\frac{2\lambda_\mu \kappa_j \Delta t}{(\mu+1)\sqrt{\pi}} \int_0^\infty \left(u\sqrt{2}\omega_\mu + \mathbf{V}(t)\cdot\kappa\right)^2 e^{-u^2} \mathrm{d}u. \end{aligned} \tag{4.20}$$

This can be split into three different integrals which can be evaluated to give

$$\begin{aligned}\psi_j(\mathbf{y}) &= -\frac{2\lambda_\mu \kappa_j \Delta t}{(\mu+1)\sqrt{\pi}} \left[\frac{\omega_\mu^2\sqrt{\pi}}{2} + \omega_\mu(\mathbf{V}(t)\cdot\kappa)\sqrt{2} + \frac{(\mathbf{V}(t)\cdot\kappa)^2\sqrt{\pi}}{2}\right] \\ &= -\frac{\lambda_\mu \omega_\mu^2 \Delta t \kappa_j}{\mu+1} - \frac{2\sqrt{2}\lambda_\mu \omega_\mu \Delta t \kappa_j}{(\mu+1)\sqrt{\pi}} \mathbf{V}(t)\cdot\kappa - \frac{\lambda_\mu \Delta t \kappa_j}{\mu+1} \left(\mathbf{V}(t)\cdot\kappa\right)^2. \end{aligned} \tag{4.21}$$

We now revert our attention to Equation 4.16. By taking expectations of each side, the expected change in the j^{th} component of the velocity over the timestep Δt is equal to $\alpha_j \Delta t$. Hence, we can relate our pdf to α_j by integrating Equation 4.21 over the whole surface of the diffusing particle, defined as the set S, where

$$S = \left\{\mathbf{y} \mid \|\mathbf{y} - \mathbf{X}(t)\|^2 = R^2\right\}. \tag{4.22}$$

In other words

$$\begin{aligned}\alpha_j &= \frac{1}{\Delta t}\int_S \psi_j(\mathbf{y})\mathrm{d}\mathbf{y}\\ &= -\frac{\gamma}{M}V_j(t).\end{aligned} \tag{4.23}$$

It is a bit more onerous to evaluate β_j. We can see from Equation 4.16 that it is equivalent to the expectation of the square of the change in velocity of the diffusing particle in the period $(t, t+\Delta t)$. To find this, we can square Equation 4.19 and integrate as we did before to get

$$\psi_j^2(\mathbf{y}) = \frac{4\lambda_\mu \kappa_j^2 \Delta t}{(\mu+1)^2\omega_\mu\sqrt{2\pi}}\int_0^\infty (c_1 + \mathbf{V}(t)\cdot\kappa)^3 \exp\left[-\frac{c_1^2}{2\omega_\mu^2}\right]\mathrm{d}c_1. \tag{4.24}$$

Here we can look ahead and observe that $\int_S \kappa_j^n \mathrm{d}\mathbf{y} = 0$ for all odd n, so we are only interested in the terms with even powers of κ in Equation 4.24. Furthermore, we discard terms of $O(\mathbf{V}^2)$ as we assume that they are very small in the limit $\mu \to \infty$. This leaves us with a single term that we can use to write

$$\begin{aligned}\beta_j^2 &= \frac{1}{\Delta t}\int_S \psi_j^2(\mathbf{y})\mathrm{d}\mathbf{y}\\ &= \frac{4\lambda_\mu}{(\mu+1)^2\omega_\mu\sqrt{2\pi}}\int_S\int_0^\infty c_1^3\kappa_j^2 \exp\left[-\frac{c_1^2}{2\omega_\mu^2}\right]\mathrm{d}c_1\mathrm{d}\mathbf{y}.\end{aligned} \tag{4.25}$$

We introduce spherical polar coordinates defined by

$$\begin{aligned}y_1 - X_1(t) &= R\cos\gamma\sin\theta,\\ y_2 - X_2(t) &= R\sin\gamma\sin\theta,\\ y_3 - X_3(t) &= R\cos\theta,\\ \gamma \in [0,2\pi],\ \ &\theta \in [0,\pi],\end{aligned}$$

enabling us to evaluate Equation 4.25 with ease by

$$\beta_1^2 = \frac{8\lambda_\mu \omega_\mu^3}{(\mu+1)^2\sqrt{2\pi}} \int_0^\pi \int_0^{2\pi} R^2 \cos^2\gamma \sin^3\theta \;\, \mathrm{d}\gamma\, \mathrm{d}\theta,$$
$$\beta_2^2 = \frac{8\lambda_\mu \omega_\mu^3}{(\mu+1)^2\sqrt{2\pi}} \int_0^\pi \int_0^{2\pi} R^2 \sin^2\gamma \sin^3\theta \;\, \mathrm{d}\gamma\, \mathrm{d}\theta,$$
$$\beta_3^2 = \frac{8\lambda_\mu \omega_\mu^3}{(\mu+1)^2\sqrt{2\pi}} \int_0^\pi \int_0^{2\pi} R^2 \cos^2\theta \sin\theta \;\, \mathrm{d}\gamma\, \mathrm{d}\theta.$$

We can then evaluate these three integrals and substitute in the definitions of the constants ω_μ and λ_μ to show that

$$\beta_j = \frac{\gamma}{M}\sqrt{2D}, \qquad \forall j \in \{1,2,3\}, \tag{4.26}$$

in keeping with the Langevin description. Thus we have shown that Equation 4.16 is equivalent to the Langevin description given by Equation 4.14-4.15 in the limit $\mu \to \infty$. $\square$

4.5.1 Illustrative figures

We conclude this chapter by including some figures generated using the event-driven hard sphere model which show that our boundary conditions are correct:

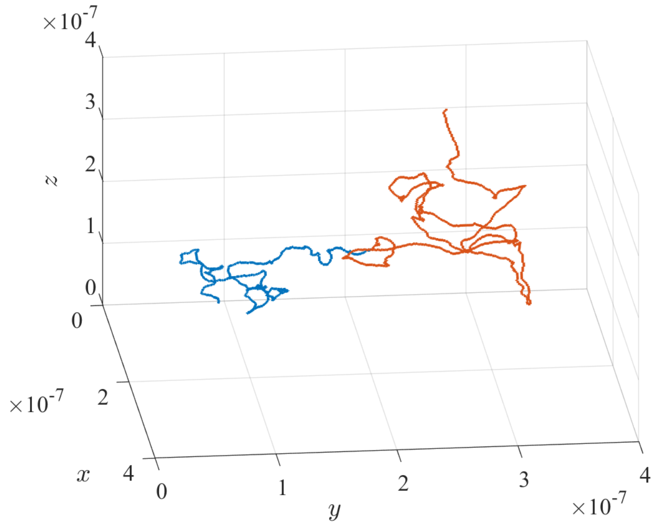

Fig. 4.4 Here we can see two trajectories. In blue is a trajectory from a model using the event-driven hard sphere theoretical heat bath. In red is a trajectory using the Langevin description.

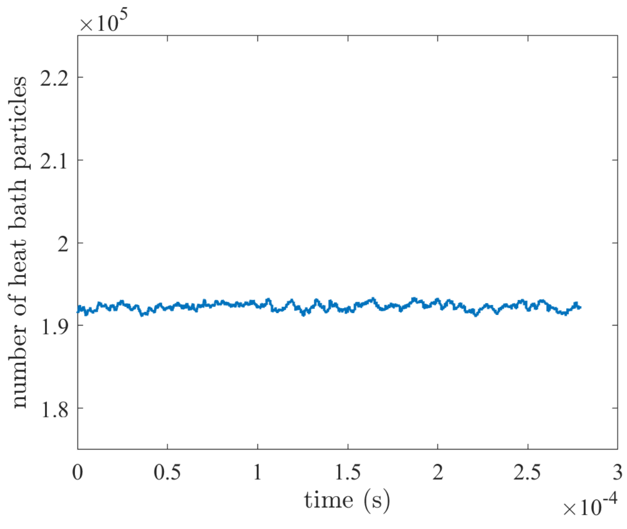

Fig. 4.5 Shows the number of heat bath particles in the cubic domain during the simulation. The fact that it is constant shows that we are introducing the right number of particles at each step.

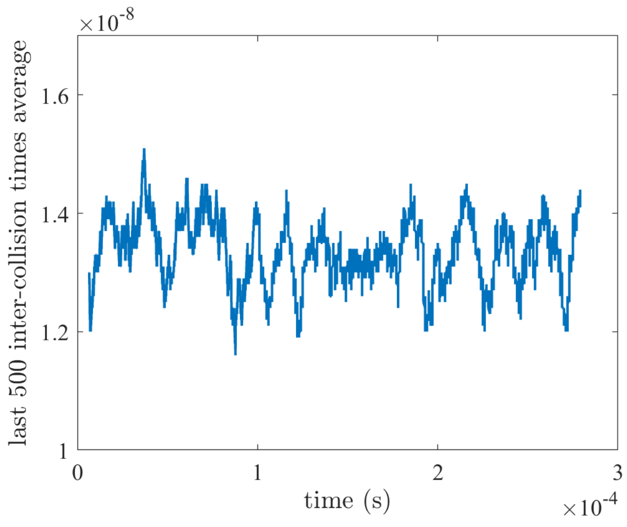

Fig. 4.6 Shows that we are also introducing them with the right velocities because the moving average of the inter-collision times does not change significantly during the simulation.

Chapter 5

Multi-resolution model

5.1 Model description

Now we finally turn to the primary novel contribution of this project. As we saw in Chapter 1, the Rouse model - whilst simple in theory - can prove to be a computationally expensive way of investigating the first passage problem. To improve upon the efficiency of the simple Rouse model, we propose a new multi-resolution approach which divides the computational domain into two regions: one in which we use coarser Langevin dynamics and one in which we use detailed molecular dynamics.

Clearly the region in which we wish to have the highest level of resolution is the area immediately surrounding the protein, since any successful binding must occur in this space. Thus, we immerse the diffusing protein in an event-driven hard sphere heat bath of the type described in Chapter 3. In the rest of the domain (i.e. the DNA chain), we use an adapted version of the Rouse model defined in Chapter 2 whereby we replace Brownian dynamics with Langevin dynamics

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \dot{\mathbf{r}}_i(t)\Delta t, \tag{5.1}$$

$$\dot{\mathbf{r}}_i(t+\Delta t) = \dot{\mathbf{r}}_i(t) - \frac{\gamma}{m_b}\dot{\mathbf{r}}_i(t)\Delta t + \frac{\gamma}{m_b}\sqrt{\frac{2k_B T\Delta t}{\gamma}}\xi + \frac{k}{m_b}\left[\mathbf{r}_{i-1}(t) - 2\mathbf{r}_i(t) + \mathbf{r}_{i+1}(t)\right]\Delta t, \tag{5.2}$$

for $i = 1, 2, \ldots, N+1$

where we have assigned an identical mass m_b to each bead such that $(R+1)m_b$ is equal to the mass of an average chromosome (estimated at 1.62×10^{-16} kg). This conversion from Brownian dynamics to Langevin dynamics for our DNA chain has minimal impact because

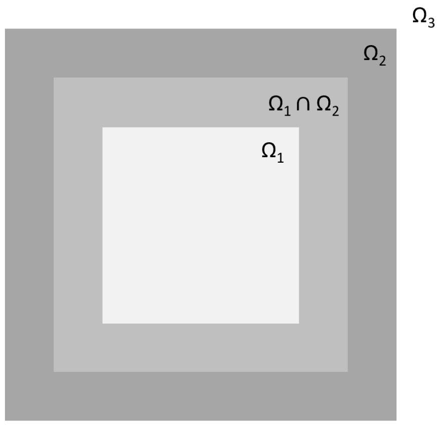

Fig. 5.1 2D illustration of computational domain

the ratio γ/m_b is so large that we can essentially discard terms of order 1 in Equation 5.2 and recover the Brownian definition from Chapter 2.

We update the Rouse chain after every 1,000 collisions in the heat bath, since this corresponds to a similar timestep to the one used in the AdRes model when it is in low resolution. When a particular bead gets near to the protein, it is able to enter the heat bath via an intermediate region, increasing the level of resolution for that particular bead. In this way, we are essentially able to 'skip over' the uninteresting part of the simulation where the DNA and the protein are far apart and focus our computational efforts on giving a good level of resolution in the final step. We now look at how exactly we combine the two levels of resolution.

5.2 Coupling Langevin dynamics and MD

In order to couple Langevin dynamics with the molecular dynamics model presented in the previous chapter, we must divide up our domain into 3 different regions: Ω_1, Ω_2 and Ω_3. In Ω_1, we have a protein centred in a heat bath where it obeys the hard sphere molecular dynamics model. We denote the space in which a bead could intersect the boundary of Ω_1 by Ω_2, as shown in Figure 5.1. The remainder of the computational domain is denoted by Ω_3.

When we introduced the hard sphere heat bath, we noted that we assume that the surrounding space is identically distributed with heat bath particles which we do not keep track of. This is important to remember when considering a bead with its centre in Ω_2 (i.e. part of the bead is also in the region Ω_1, because our failure to keep track of collisions outside

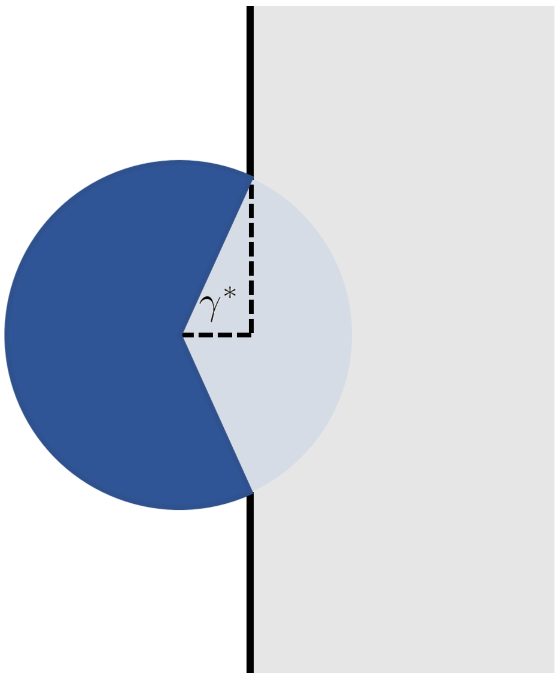

Fig. 5.2 A particle crossing the boundary into the heat bath

Ω_1 would lead to a bead being pushed away when it tries to enter the heat bath. Consequently, we apply a correction to the velocity of the bead in the form of a discretized SDE

$$\dot{r}_k(t+\Delta t) = \dot{\tilde{r}}_k(t+\Delta t) + \alpha_k \Delta t + \beta_k \sqrt{\Delta t}\,\xi_k, \qquad k = 1,2,3, \tag{5.3}$$

where $\dot{\tilde{\mathbf{r}}}(t+\Delta t)$ is the post-collision velocity of the bead after considering only collisions with heat bath particles from Ω_1. Note, our bead subscripts here denote vector components rather than bead index. We do not include bead index in the remainder of our analysis for simplicity.

To evaluate α_k, β_k, we proceed in a similar manner to when we proved that the hard sphere model converges to the Langevin description. However, whereas we previously integrated over the whole surface S of the sphere, in this case we only integrate over a subset of the surface $\tilde{S}$, defined as

$$\tilde{S} = \{\mathbf{y} \in \Omega_2 : |\mathbf{y} - \mathbf{r}(t)|^2 = \sigma^2\}. \tag{5.4}$$

We can parametrize this surface using an adapted set of spherical polar coordinates. For example, if consider centering Ω_1 at $[L/2, L/2, L/2]$ and focus on the face of the cube at

$x=0$, we can define the angle γ^* as shown in Figure 5.2 where $\gamma^* = \cos^{-1}\left(-\frac{r_1}{\sigma}\right)$, and thus $\mathbf{y} \in \tilde{S}$ is equivalent to

$$\begin{aligned} y_1 &= r_1(t) + \sigma\cos\gamma && (5.5)\\ y_2 &= r_2(t) + \sigma\cos\theta\sin\gamma && (5.6)\\ y_3 &= r_3(t) + \sigma\sin\theta\sin\gamma && (5.7)\\ \gamma &\in [\gamma^*, \pi], \;\; \theta \in [0, 2\pi]. && (5.8)\end{aligned}$$

Hence, we have that

$$\alpha_k = \frac{1}{\Delta t}\int_{\tilde{S}} \psi_k(\mathbf{y})\mathrm{d}\mathbf{y}, \tag{5.9}$$

where ψ_k is the pdf we defined in the previous chapter. We have to be careful because we are no longer integrating over closed curves so do not get quite as simple an integral as before. Proceeding with caution, we can find that

$$\begin{aligned}\alpha_1 &= -\frac{4\lambda_\mu\omega_\mu\dot{r}_1(t)}{(\mu+1)\sigma^2\sqrt{2\pi}}\int_0^{2\pi}\int_{\gamma^*}^{\pi}\sigma^4\cos^2\gamma\sin\gamma\mathrm{d}\gamma\mathrm{d}\theta - \frac{\lambda_\mu\omega_\mu^2}{(\mu+1)\sigma}\int_0^{2\pi}\int_{\gamma^*}^{\pi}\sigma^3\cos\gamma\sin\gamma\mathrm{d}\gamma\mathrm{d}\theta\\ &= -\frac{8\lambda_\mu\omega_\mu\sigma^4\pi\dot{r}_1(t)}{3(\mu+1)\sigma^2\sqrt{2\pi}}\left(1-\frac{r_1^3}{\sigma^3}\right) + \frac{\pi\sigma^3\omega_\mu^2\lambda_\mu}{(\mu+1)\sigma}\left(1-\frac{r_1^2}{\sigma^2}\right)\\ &= -\frac{\gamma}{2m_p}\dot{r}_{k,1}(t)\left(1-\frac{r_1^3}{\sigma^3}\right) + \frac{3\gamma\sqrt{\pi}\omega_\mu}{8m_p\sqrt{2}}\left(1-\frac{r_1^2}{\sigma^2}\right), \qquad (5.10)\end{aligned}$$

and can proceed similarly for other values of i and to evaluate β_i. We find that for a bead intersecting the surface at $x=0$, the right velocity correction terms are

$$\begin{aligned}\alpha_1 &= -h_1(r_1)\gamma\dot{r}_1 + h_0(r_1), && && (5.11)\\ \alpha_k &= -h_k(r_1)\gamma\dot{r}_k(t), && \text{for } k=2,3 && (5.12)\\ \beta_k &= \sqrt{h_k(r_1)}\gamma\sqrt{2D} && \text{for } k=1,2,3, && (5.13)\end{aligned}$$

where $h_k : [-\sigma, \sigma] \to \mathbb{R}$ satisfy

$$h_0(r_1) = \frac{3\gamma\omega_\mu\sqrt{\pi}}{8\sqrt{2}m_b}\left(1 - \frac{r_1^2}{\sigma^2}\right) \tag{5.14}$$

$$h_1(r_1) = \frac{1}{2m_b}\left(1 - \frac{r_1^3}{\sigma^3}\right) \tag{5.15}$$

$$h_2(r_1) = h_3(r_1) = \frac{1}{4m_b}\left(2 - 3\frac{r_1}{\sigma} + \frac{r_1^3}{\sigma^3}\right). \tag{5.16}$$

As a sanity check, we can substitute $r_1 = -\sigma$ to find

$$h_0(-\sigma) = 0, \quad h_k(-\sigma) = 1 \quad \text{for } k = 1,2,3,$$

which confirms that this velocity correction converges to the Langevin description (Equation 5.1) on the boundary between Ω_3 and Ω_2. Thus, we have formulated the conditions for an intermediate region which blends Langevin dynamics and MD together.

5.3 Simulation details

All that is left to do is explain how we bring this rather complicated model together in one iteration of the algorithm.

Algorithm 3: Multi-Resolution Algorithm (MultiRes)

```
Input: FJC-initialized chain, protein at initial distance $d_0$ from middle bead
while $\min|\mathbf{r}_i - \mathbf{X}| > d_r$ do
    if all $\mathbf{r}_i \notin \Omega_1 \cup \Omega_2$ then
        Update position of protein according to event-driven heat bath
        Update position of DNA chain every 1,000 collisions according to
          Equation 5.1-5.2
    else
        Update all particles in heat bath, considering that next collision could now be
          with a bead
        Apply velocity corrections to beads in $\Omega_2$
        Update other beads $\notin \Omega_1 \cup \Omega_2$ every 1,000 collisions according to
          Equation 5.1-5.2
        If new $\min|\mathbf{r}_i - \mathbf{X}| < d_r \implies$ binding successful
Output: Total time and whether or not protein was reflected
```

5.4 Parameter choice

Just as we reduced the resolution of the Rouse chain in Chapter 3, we use a resolution factor of 2 to reduce the chain to 225 beads

We use the resolution factor from Chapter 3 to scale our parameters from the simple Rouse model. We choose a resolution factor of 2, meaning that we end up with 225 beads of radius 4.8 nm, joined by springs of Kuhn length 120 nm.

Our choices for the parameters defined in the Chapter 4 are summarised in Table 5.1.

Symbol	**Name**	**Value**	**Justification**
L	Length of cubic domain	100 nm	Arbitrary
μ	Ratio of particle mass to solvent mass	$1,000$	Large enough to provide good accuracy
M	Mass of protein	6.26×10^{-26} kg	The mass of the canonical TBP protein chain
λ_μ	Solvent density	6.96×10^{26}	Calculated according to Equation 4.1
R	Diffusing particle radius	2.5 nm	From Chapter 2
ω_μ	Second moment of heat bath velocity	$6.7 \times 10^{8}\ \mathrm{m\,s^{-1}}$	Calculated according to Equation 4.3

Table 5.1 Parameter selection for MultiRes model

5.5 Results

The results of 1,000 simulations are shown in Figure 5.3-5.4. Our heat bath contains 69,649,299 heat bath particles, on average, which creates a fairly heavy computational load. Consequently, we actually find this multi-resolution model to be 5.3 times slower on an AMD FX(tm)-4350 Quad-Core Processor than the simple Rouse model, which is a disappointing finding. We consider the implications of this further in Chapter 6.

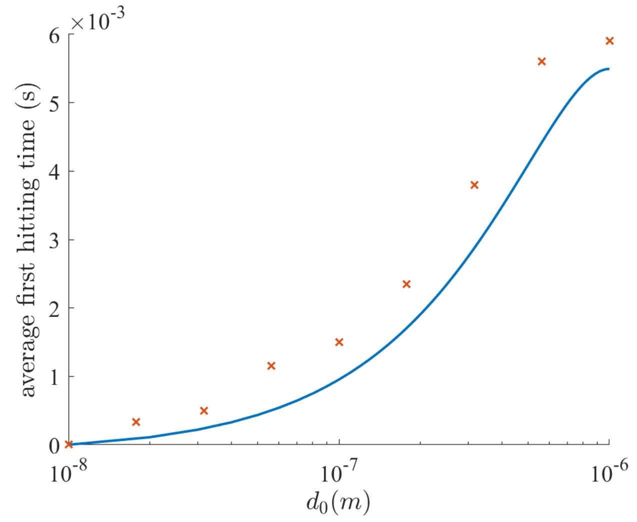

Fig. 5.3 Comparison between the results of our simulations and the prediction of the analytical model from Chapter 1 for the first passage time problem

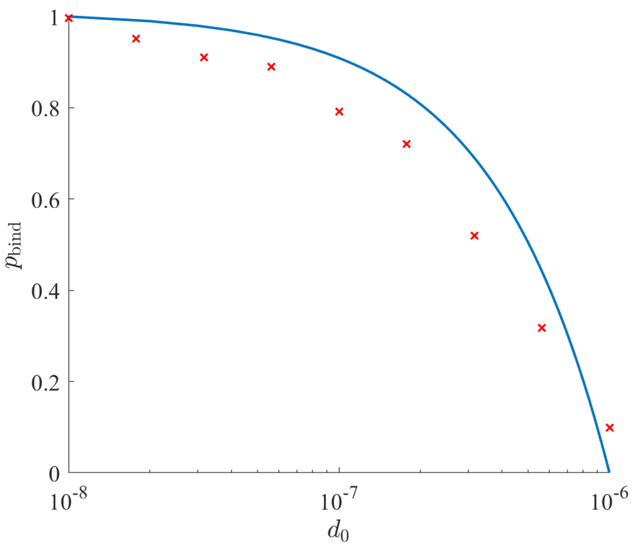

Fig. 5.4 Comparison between the results of our simulations and the prediction of the analytical model from Chapter 1 for the first passage probability problem

Chapter 6

Discussion and conclusion

6.1 Conclusion

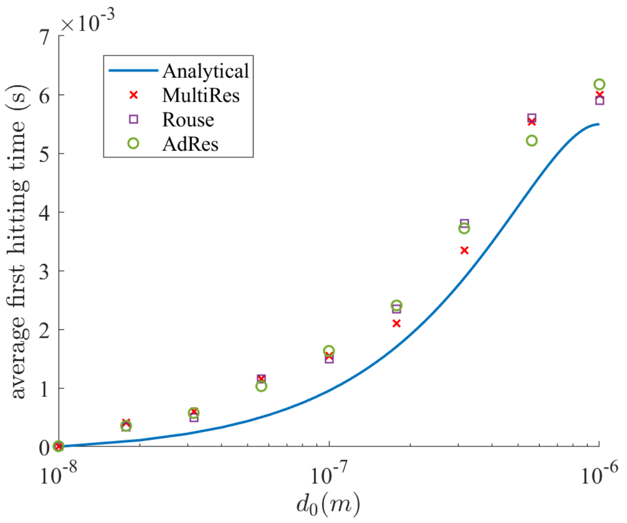

Fig. 6.1 Comparison between the results of our simulations and the prediction of the analytical model from Chapter 1 for the first passage time problem

The results of these three different models for DNA-protein interactions are summarised in Figure 6.1-6.2, compared to the analytical result from Chapter 1. There are two clear takeaways from the results. Firstly, the analytical model is an overly-simplified picture of what is really going on and cannot capture the true story. This is shown by the sizeable difference between the predictions of the analytical model and all three other models.

The second clear conclusion is that the three models are more or less consistent with one another at both of the problems considered. This second conclusion is not altogether surprising. Indeed, it is exactly what we had hoped to find when starting with the simple

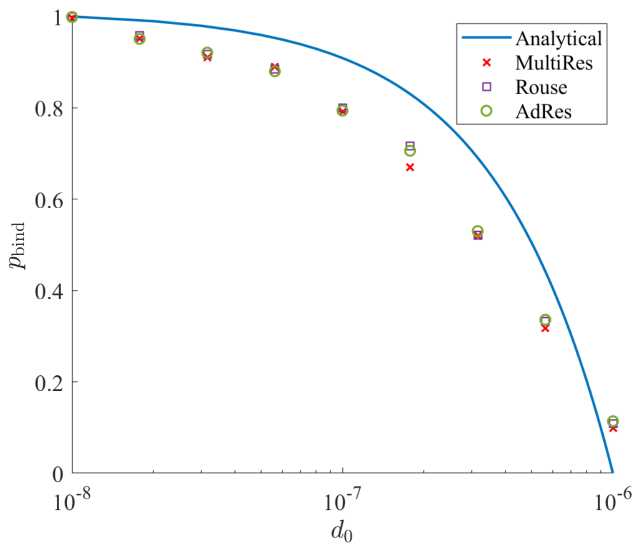

Fig. 6.2 Comparison between the results of our simulations and the prediction of the analytical model from Chapter 1 for the first passage probability problem

Rouse model in Chapter 2 and trying to improve upon it whilst maintaining the accuracy of the original model. The implication of this is that an investigator can freely choose between all three models without having to compromise on accuracy of results.

An investigator whose primary focus was efficiency and who wanted to consider a coarse picture of DNA-protein dynamics should clearly opt for the AdRes model since it is 3.4 times quicker than either of the other two. However, that is not to say that the novel contribution of this project - the MultiRes model - is totally useless. The MD level of detail around the protein provides considerable versatility for further research in areas such as docking where the orientation of the protein must be considered. In this sense, the MultiRes model has a great deal of potential advantages over the AdRes model.

6.2 Discussion

In terms of the biological accuracy of our model, a brief comparison to biological literature suggests that there are a number of areas in which our analysis is flawed. For instance, we have assumed that our protein and filament are immersed in a solvent with the same viscosity as water (1 cP). However, this first stage of transcription actually takes place in the nucleoplasm which is reported as having a viscosity of around 10 times that of water. This could prove to be a serious issue for the MultiRes model in particular as a tenfold increase in solvent density would leave us needing to keep track of ten times as many heat bath particles.

Since we already track just under 70,000,000, any further increase would likely prove to be prohibitive for the memory of most computers.

For further direction, the contributions of this project could be expanded upon by comparison to other preeminent techniques for first passage problem analysis. First passage kinetic Monte Carlo [8] and Green's function reaction dynamics methods [12] are two examples of approaches that have been taken in the literature, so a comparison to the results obtained using these methods, along with the computational efficiency, would be a good place to start.

References

[1] Allen, M. P. and Tildesley, D. J. (2017). *Computer simulation of liquids*. Oxford university press.

[2] Devroye, L. (1986). Sample-based non-uniform random variate generation. In *Proceedings of the 18th conference on Winter simulation*, pages 260–265. ACM.

[3] Einstein, A. (1906). A new determination of molecular dimensions. *Ann. Phys.*, 19:289–306.

[4] Erban, R. (2014). From molecular dynamics to brownian dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 470(2167):20140036.

[5] Hastings, C. (1955). *Approximations for Digital Computers*. Princeton University Press, Princeton, NJ, USA.

[6] Juo, Z., Chiu, T., Leiberman, P., Baikalov, I., Berk, A., and Dickerson, R. (1996). Tata binding protein (tbp)/dna complex.

[7] Kwakernaak, H. and Sivan, R. (1972). *Linear optimal control systems*, volume 1. Wiley-interscience New York.

[8] Oppelstrup, T., Bulatov, V. V., Donev, A., Kalos, M. H., Gilmer, G. H., and Sadigh, B. (2009). First-passage kinetic monte carlo method. *Physical Review E*, 80(6):066701.

[9] Robert, C. P. (1995). Simulation of truncated normal variables. *Statistics and computing*, 5(2):121–125.

[10] Rolls, E., Togashi, Y., and Erban, R. (2017). Varying the resolution of the rouse model on temporal and spatial scales: Application to multiscale modeling of dna dynamics. *Multiscale Modeling & Simulation*, 15(4):1672–1693.

[11] Schmidt, H. G., Sewitz, S., Andrews, S. S., and Lipkow, K. (2014). An integrated model of transcription factor diffusion shows the importance of intersegmental transfer and quaternary protein structure for target site finding. *PLOS one*, 9(10):e108575.

[12] van Zon, J. S. and Ten Wolde, P. R. (2005). Green's-function reaction dynamics: a particle-based approach for simulating biochemical networks in time and space. *The Journal of chemical physics*, 123(23):234910.

Appendix A

Random number generation

Here we present two acceptance-rejection algorithms which we use throughout our simulations for sampling from more complicated distributions. Both algorithms are based on the following result from [2], presented without proof:

Lemma A.0.1. Let $h(x)$ and $g(x)$ be two densities such that $h(x) \leq Mg(x)$ for all x in the support of $h(x)$. Then the random variable x given as the output of the following algorithm

1. Generate $z \sim g(z)$
2. Generate $u \sim \mathscr{U}(0,1)$. If $u \leq h(z)/Mg(z)$, take $x = z$. Else, repeat

is distributed according to $h(x)$

A.1 Complementary error function

To sample numbers from the probability distribution of the normalized complementary error function

$$h(x) = \sqrt{\pi}\,\mathrm{erfc}(x), \tag{A.1}$$

we use acceptance-rejection sampling with $g(x) = \exp(-x/a_1)/a_1$, i.e. the pdf of an exponential distribution with mean a_1.

We assume that the ratio of the two densities is bounded above

$$\frac{h(x)}{g(x)} = \sqrt{\pi}a_1\,\mathrm{erfc}(x)\exp\left(\frac{x}{a_1}\right) \leq M(a_1), \tag{A.2}$$

where M is some real number that depends on our choice of parameter a_1. If we further define a second parameter, $a_2 = \sqrt{\pi}a_1/M$ then we can write the acceptance-rejection algorithm using Lemma A.0.1 as

- Generate a random number $\eta_1 \sim \mathscr{U}(0,1)$
- Compute an exponentially distributed random number η_2 by setting $\eta_2 = a_1 \log(\eta_1)$
- Generate a random number $\eta_3 \sim \mathscr{U}(0,1)$
- If $\eta_3 < a_2 \operatorname{erfc}(\eta_2) \exp(\eta_2/a_1)$, choose η_2 as a sample from Equation A.1. Otherwise, repeat algorithm.

Table A.1 Algorithm for sampling from complementary error function.

The obvious follow-up question is: what impact do our values of a_1 and a_2 have and how should we choose them accordingly? Obviously we are keen to have as high an acceptance probability as possible. We can evaluate this acceptance probability by considering all possible values of η_2

$$\begin{aligned}
p_{\text{acceptance}} &= \int_0^\infty \mathbb{P}(\eta_3 < a_2 \operatorname{erfc}(\eta_2) \exp(\eta_2/a_1), \eta_2 = y)\, \mathrm{d}y \\
&= \int_0^\infty a_2 \operatorname{erfc}(y) \exp(y/a_1) \times \frac{\exp(-y/a_1)}{a_1} \mathrm{d}y \\
&= \frac{a_2}{a_1\sqrt{\pi}} \qquad \text{(A.3)} \\
&= \frac{1}{M(a_1)}. \qquad \text{(A.4)}
\end{aligned}$$

Therefore, to maximise $p_{\text{acceptance}}$, we must minimise M over all potential choices of a_1. To do this, we proceed numerically and use a good approximation for $\operatorname{erfc}(x)$, taken from Hastings' 1955 work *Approximations for Digital Computers*, which represents the function in terms of a rational polynomial and a Gaussian to $\sim 10^{-5}$ accuracy [5]. By differentiating the left side of Equation A.2, one can find that the maximum occurs where

$$\operatorname{erfc}(x) = \frac{2e^{-x^2}a_1}{\sqrt{\pi}},$$

which we can reduce when applying the approximation into a cubic polynomial

$$0.3480242t - 0.0958798t^2 + 0.7478556t^3 - \frac{2a_1}{\sqrt{\pi}} = 0,$$

where $t = 1/(1+0.47047x)$. We can solve this to find the x-coordinate of the maximum of Equation A.2) in terms of a_1 and use this to evaluate $M(a_1)$. As can clearly be seen in Figure A.1, a local minimum of M exists for some small positive value of a_1. This can be evaluated numerically to be at $a_1 = 0.5316$, which gives $M = 1.158$, $a_2 = 0.8136$ and an acceptance probability of 86.3%. We use these parameters in the algorithm in Table A.1 throughout our simulations and it is pleasing to note that they are the same as the values used in the literature [4].

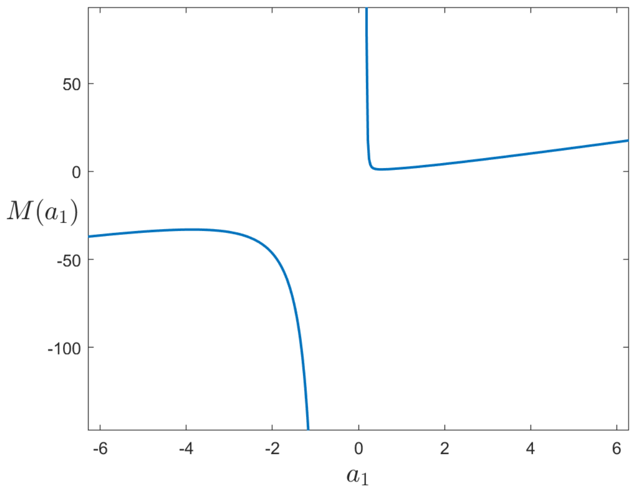

Fig. A.1 A graph of M against a_1, showing the minimum point.

A.2 Truncated normal distribution

We use another acceptance-rejection algorithm for sampling from the truncated normal distribution proposed in the literature [9]. The underlying principle of the algorithm is that the translated exponential distribution is similar in shape to the truncated normal distribution whilst always being greater than it, making it a suitable choice for the density $g(x)$ in the context of Lemma A.0.1, particularly if we choose it to have mean a, where

$$a = \frac{A + \sqrt{A^2+4}}{2} \tag{A.5}$$

- Generate a random number $\eta_1 \sim \mathscr{U}(0,1)$
- Compute a translated exponentially distributed random number η_2 by setting $\eta_2 = A - \log(\eta_1)/a$
- Generate a random number $\eta_3 \sim \mathscr{U}(0,1)$
- If $\eta_3 < \exp(-(\eta_2 - a)^2/2)$, choose η_2 as a sample from Equation A.6. Otherwise, repeat algorithm.

Table A.2 Algorithm for sampling from truncated normal distribution.

and A is the value at which our Gaussian distribution is truncated.

The density we wish to sample from, $h(x)$, is defined on the interval (A, ∞) as

$$h(x) = \frac{\sqrt{2}}{\sqrt{\pi}\,\mathrm{erfc}\left(A/\sqrt{2}\right)} \exp\left(-\frac{x^2}{2}\right), \qquad \text{for } x \geq A, \tag{A.6}$$

where the factor of $\mathrm{erfc}(A/\sqrt{2})/2$ is a normalizing factor that can be derived by integrating a standard Gaussian over the new domain.

By dividing and completing the square, one can rearrange to find

$$\begin{aligned} \frac{h(x)}{g(x)} &= \frac{\sqrt{2}\exp(-Aa/2)}{a\sqrt{\pi}\,\mathrm{erfc}\left(A/\sqrt{2}\right)} \exp\left(-\frac{(x-a)^2}{2}\right) \\ &\leq \frac{\sqrt{2}\exp(-Aa/2)}{a\sqrt{\pi}\,\mathrm{erfc}\left(A/\sqrt{2}\right)} := M \end{aligned} \tag{A.7}$$

Thus, $h(z)/Mg(z) = \exp\left(-(z-a)^2/2\right)$, and so using Lemma A.0.1 we know that the algorithm in Table A.2 must produce random variables x distributed according to Equation A.6.

Appendix B

Illustrative code availability

Illustrative code written in Fortran 90 for all three models introduced in this project can be found on an anonymous GitHub page at www.github.com/dna-protein-interaction-dissertation.