On Interacting Particles in 1D and 2D

Joshua DM Hellier

Doctor of Philosophy
The University of Edinburgh
July 2018

Abstract

Interface growth, and in particular the prediction of its rate, has long been a tough problem in statistical physics. In this thesis, I will outline my personal take on the matter, and will showcase a possible approach to it consisting of constructing a microscopic model on a lattice and using this to parametrise a large-scale model of the phenomenon. I will then discuss how to do this with multiple interacting particle species in play.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Parts of this work have been published in .

(*Joshua DM Hellier, July 2018*)

Acknowledgements

Insert people you want to thank here.

Contents

List of Figures

List of Tables

Chapter 1

Preliminary Work, Background and Motivation

Here we need to talk about the intent of the project.

1.1 The TiO_2/Ti Interface System

A description of the initial problem upon which the project was based.

1.2 Initial Attempts to Model the TiO_2/Ti Interface System

1.2.1 Molecular Dynamics

Need to explain why issues with using MD, and why I eventually decided it was not a useful technique for this problem.

1.3 Large-Scale Models of the Ti/O/Nb Interacting System

I had a think about various methods I could use to tackle the system in question, and decided that the approach would would be most likely to bear fruit would be a continuum-modelled bulk PDE system with appropriate boundary conditions between phases.

1.3.1 Proposed Linear System

Simplest possible model, and why it failed.

1.3.2 Attempts to create a Suitable Nonlinear System

Talk about why nonlinearity is necessary (as in, it just spits out the previous system again), and the difficulties of parametrising it.

1.3.3 Parametrisation from a Microscopic Model

Talk about the Dresden conference and what I learned from it.

1.4 The Sticky Particle Model

1.4.1 Model Motivation

As in, why this is a good start in 1d.

1.4.2 Model Definition

1.4.3 Model Properties

Including Detailed Balance, symmetry, "locality". Also mention that it is a Markov process.

1.4.4 Relation to Existing Literature

1.4.5 Generalisation to Higher Dimensions

Including a proof of detailed balance in arbitrary dimensions (on square lattice).

1.5 Implications of Initial Work for the PhD Direction

1.5.1 Why the Change of Direction?

Essentially, why trying to solve this particular problem is actually kind of silly, and why having a better theory of driven lattice flows would be more useful.

1.5.2 Why Investigate Flow in the SPM?

Talk about how boundary-condition-induced flow on systems that would otherwise obey detailed balance hasn't really been done before. Bring it around to the question: "Can we have interesting dynamics in a model whose bulk motion is symmetric and obeys detailed balance?"

Chapter 2

Transition Rate Matrix Analysis

Now that we have MFT predictions about the relationship between density difference and current in the SPM, it would be good to try to investigate their validity. In Chapter 3 we will use Monte-Carlo methods to do this in 1d and 2d, but in this chapter we will restrict our attention to 1d.

2.1 The Transition Rate Operator for the SPM

The SPM is an autonomous continuous-time Markov Process, which describes continual transitions between states with transition rates depending only upon the current state. As such, if we call the total space of states Ξ then the probability distribution $P : \Xi \times \mathbb{R} \to \mathbb{R}$ should obey a **master equation**

$$\frac{\partial P(\xi, t)}{\partial t} = \mathcal{A}P(\xi, t), \tag{2.1}$$

where $\mathcal{A} : \Xi \to \Xi$ is the **transition rate operator** or TRO. Note that I am going to be using column vectors for probabilities rather than row vectors (as many in the probability community do) because it is what I am used to. Parametrising $\mathcal{A}$ via

$$(\mathcal{A}f)(u) = \int_\Xi \mathrm{d}\xi \; \sigma(u, \xi) f(\xi) \tag{2.2}$$

puts it in a more familiar, transport equation-style notation:

$$\frac{\partial P(\xi, t)}{\partial t} = \int_\Xi \mathrm{d}u \; \sigma(\xi, u) P(u, t) \tag{2.3}$$

We demand that σ satisfies

$$\forall \xi \in \Xi,\ \sigma(\xi, \xi) \leq 0 \tag{2.4}$$

and

$$\forall \xi_1, \xi_2 \in \Xi : \xi_1 \neq \xi_2,\ \sigma(\xi_1, \xi_2) \geq 0, \tag{2.5}$$

as well as the constraint

$$\forall u \in \Xi,\ \int_{\Xi} \mathrm{d}\xi\ \sigma(\xi, u) = 0. \tag{2.6}$$

The last constraint implies that

$$\int_{\Xi} \mathrm{d}\xi \frac{\partial P(\xi, t)}{\partial t} = \int_{\Xi} \mathrm{d}u\ P(u, t) \left[\int_{\Xi} \mathrm{d}\xi\ \sigma(\xi, u) \right] = 0 \tag{2.7}$$

regardless of the structure of P, which is our probability conservation equation.

The formal (forward-time) solution to Eq. 2.1 is given by

$$P(\xi, t) = e^{(t-t_0)\mathcal{A}} P_0, \tag{2.8}$$

where t_0 is some initial time, P_0 is the starting distribution, and the operator exponential is defined by its Taylor expansion, which should converge fine for bounded $\mathcal{A}$, satisfied for the finite-system SPM. As $e^{(t-t_0)\mathcal{A}}$ and $\mathcal{A}$ share eigenvectors, we see that the eigenstructure of $\mathcal{A}$ is something well worth investigating, as it should give us information about the time-evolution of the system. An important thing to point out is that $\mathcal{A}$ does not in general have orthogonal eigenvectors because it is not in general symmetric, and so we cannot normally diagonalise it using orthogonal transformations. This means that modes do not "decouple" in the way that states do in the Schrödinger equation, and we instead have to deal with an **adjoint system**.

Luckily, when it comes to the analysis of steady states, there are a few results that can help us. If we consider the operator $\mathcal{G}_T = e^{T\mathcal{A}}$ (in other words, the **propagator** for a period of time T), it is pretty easy to see that this is a standard Markov Operator, as we are essentially reversing the limiting process we would perform in order to define a continuous time Markov process as a limit of a discrete time one. If we assume that Ξ is finite and $\mathcal{G}_T$ is irreducible then as a Markov operator it must have a a unique eigenvector with corresponding eigenvalue 1. All other eigenvalues must have modulus between 0 and 1. Therefore, $\mathcal{A}$ must

share that same unique eigenvector with associated eigenvalue 0, and its other eigenvectors must have negative real part. In other words, the system must have a single steady state probability distribution, which it always relaxes towards exponentially quickly, with a rate determined by the nonzero eigenvalue of $\mathcal{A}$ with real part closest to 0.

Of course, for such finite-state systems (such as the SPM on a finite domain) the integrals become sums, and σ a matrix, Q. In such systems, we can arrange to have some labelling scheme which uniquely relates system states to natural numbers, and therefore relates states to basis vectors in a vector space. In the SPM, a site is either full or empty, which means that there is a natural mapping between states and natural numbers based upon binary representation; a string of 1s and 0s can be associated with a natural number as well as a configuration of particles and vacancies.

2.1.1 A Small Worked Example: Closed System

As a concrete example, let us consider the SPM on a cyclic domain of length 3. There are 2^3 possible combinations, and so 8 possible states: 000, 001, and so forth. The transition rate matrix describing this system is:

$$Q = \begin{bmatrix} 0 & & & & & & & \\ & -2 & 1 & & 1 & & & \\ & 1 & -2 & & 1 & & & \\ & & & -2\lambda & & \lambda & \lambda & \\ & 1 & 1 & & -2 & & & \\ & & & \lambda & & -2\lambda & \lambda & \\ & & & \lambda & & \lambda & -2\lambda & \\ & & & & & & & 0 \end{bmatrix}, \tag{2.9}$$

where we have omitted most of the zero entries for clarity. An alert observer will note that Q is reducible, and so by permuting the basis vectors we can rearrange

the matrix into block form:

$$Q' = \begin{bmatrix} -2 & 1 & 1 & & & & & \\ 1 & -2 & 1 & & & & & \\ 1 & 1 & -2 & & & & & \\ & & & -2\lambda & \lambda & \lambda & & \\ & & & \lambda & -2\lambda & \lambda & & \\ & & & \lambda & -2\lambda & \lambda & & \\ & & & & & & 0 & \\ & & & & & & & 0 \end{bmatrix}. \tag{2.10}$$

According to our recipe, to learn about the solutions to Eq. 2.1 in this case, we need to know about the eigendecomposition of Q. The block structure of Q' means that there are 4 distinct parts of the state space, between which there are no transitions; this partitioning corresponds to the fact that particle number is conserved on the ring. Two of these sectors correspond to the full and empty states, and their dynamics are completely trivial, in the sense that the state space is 1d and there are no dynamics, as the particles/vacancies have nowhere to go! The remaining sectors correspond to the situation where there is one particle or one vacancy. The matrix is symmetric, meaning that its eigenvalues are real, and we find that both nontrivial blocks can be diagonalised to form a multiple of

$$\begin{bmatrix} 0 & & \\ & -3 & \\ & & -3 \end{bmatrix}, \tag{2.11}$$

with eigenvectors $[1,1,1]^{\mathrm{T}}$, $[-1,0,1]^{\mathrm{T}}$ and $[-1,1,0]^{\mathrm{T}}$ respectively, the latter two forming a degenerate eigenspace.

In this particular example then, we find that there 4 steady states:

- All slots full,
- All slots empty,
- One particle present, with equal chance to be in any particular position,
- The same but with a vacancy instead of a particle.

Whilst the first 2 cases are trivial (one-dimensional probability spaces), in the other two cases we relax towards the steady state with rates 3 and 3λ respectively.

In this example, the TRM was actually symmetric, and so all the eigenvalues were real. It was also highly reducible, due to the strict constraints imposed by the particle conservation law. When we have boundary conditions which permit the creation and destruction of particles, that will change; we will consider that situation now.

2.2 Forming the TRM for Systems with Dirichlet Boundary Conditions

2.2.1 Dirichlet Boundary Conditions

In Chap. **??** we made an MFT for the SPM in 1-dimension, and when investigating steady states used Dirichlet boundary conditions when we needed them. In particular, we had a system of length L and sought solutions in which the system density was pinned at ρ_0 at $x = 0$ and ρ_L at $x = L$.

We would like to do a similar thing for the non-MFT SPM. An exact analogue of the situation does not exist, as occupation number is not defined *a priori* in a Markov process, but merely emerges as a result of the rate prescribed.The closest imitation to it we can get is by allowing particles to be created and destroyed in boundary regions at either end of a chain, and then try to set these rates so that the time-averaged occupation probability in the end sites are ρ_0 and ρ_L respectively. Note that this is a little more involved than simply loading and unloading particles as one does in ASEP, as

- loading and unloading really doesn't simulate the boundary condition we are looking for, and
- we actually need to consider a two-site boundary layer attached to each end, because the internal dynamics of the particles depend upon their immediate environment.

To simulate a boundary which is attached to a reservoir with occupation ρ, we allow particles to appear in empty boundary sites with rate $B_0\sqrt{\frac{\rho}{1-\rho}}$ and to disappear from full ones with rate $B_0\sqrt{\frac{1-\rho}{\rho}}$, for some positive B_0. If we switch off all other dynamics and consider this in isolation, we would have a bunch of

decoupled two-state systems. Writing a TRM for this with the first basis vector being the empty state and the second the full one, we get

$$Q = \begin{bmatrix} -B_0\sqrt{\frac{\rho}{1-\rho}} & B_0\sqrt{\frac{1-\rho}{\rho}} \\ B_0\sqrt{\frac{\rho}{1-\rho}} & -B_0\sqrt{\frac{1-\rho}{\rho}} \end{bmatrix}. \tag{2.12}$$

There is of course a zero eigenvalue, which corresponds to a stationary distribution $[1-\rho, \rho]^{\mathrm{T}}$, precisely as desired.

In order to simulate attaching a reservoir with particle density ρ, we apply the creation and annihilation rates above to the outermost two sites in our lattice. The outermost site only performs these operations. The inner boundary site undergoes these creation and annihilation processes, as well as the normal dynamics of the SPM; thus, particles will move in and out of it as normal, and when the outermost site's occupation is required to determine the transition rate for a particle moving inward, that information is available. One could possibly eliminate the need for the outermost boundary site by averaging over occupations, however we have chosen not to do this for consistency with out Monte-Carlo calculations in Chap. 3.

Observant readers will notice that by using these creation and annihilation rates we are left with a free variable, B_0. This controls the ratio of the creation and annihilation rates to the internal dynamical rates 1 and λ. In general when we're trying to simulate an adjacent reservoir we want the boundary motion to be "fast" compared to the internal dynamics, and so B_0 may be regarded as a regularisation parameter; any choice of B_0 should be good so long as the creation and annihilation rates sufficiently dominate both 1 and λ. In practise, in our larger-scale calculations we used

$$B_0 = b(1+\lambda), \tag{2.13}$$

with b set to, 100 or 1000.

2.2.2 Another Small Worked Example: Open System

Using full boundary conditions, the smallest system we can consider consists of adjacent boundary layers, in which particles pass directly from the inner layer of one boundary to the inner layer of the other boundary. Unfortunately the state

space of such a system is $2^4 = 16$-dimensional, and so it would be cumbersome to consider such a system as an example here.

However, in the special case in which one boundary has density $\rho_0 = 1$ and the other has $\rho_L = 0$, the outer boundary layers become nondynamical, and the system simplifies considerably; particles can only enter at the full end with rate λ, and can only leave at the empty end. Therefore, we can consider a system consisting of single boundary layers attached to a single internal slot without needing to consider a state space larger than $2^3 = 8$. Using the same state-labelling convention as in Sec. 2.1.1, the TRM for this new open system is

$$Q = \begin{bmatrix} -\lambda & 1 & & & & & & \\ & -\lambda-2 & 1 & & & & & \\ & 1 & -\lambda-2 & 1 & \lambda & & & \\ & & & -2\lambda-1 & & \lambda & & \\ \lambda & & 1 & & -\lambda & 1 & & \\ & \lambda & & \lambda & & -\lambda-2 & \lambda & \\ & & \lambda & & & 1 & -\lambda & \lambda \\ & & & \lambda & & & & -\lambda \end{bmatrix}. \tag{2.14}$$

This time the matrix is not symmetric, and there does not exist a basis permutation which puts the TRM into block form. If we calculate the characteristic polynomial $p(q) = |Q - qI|$, we find that

$$\begin{aligned} p(q) = q\left[q^7 + (9\lambda+7)q^6 + \left(34\lambda^2+53\lambda+17\right)q^5 + \left(71\lambda^3+165\lambda^2+103\lambda+17\right)q^4 + \right. \\ + \left(89\lambda^4+274\lambda^3+247\lambda^2+76\lambda+6\right)q^3 + \left(67\lambda^5+256\lambda^4+294\lambda^3+126\lambda^2+17\lambda\right)q^2 \\ \left. + \left(28\lambda^6+127\lambda^5+171\lambda^4+91\lambda^3+15\lambda^2\right)q + \left(5\lambda^7+26\lambda^6+37\lambda^5+24\lambda^4+4\lambda^3\right)\right]. \end{aligned} \tag{2.15}$$

Of course, the roots of $p(q)$ are the eigenvalues of Q. Clearly one of the eigenvalues is $q = 0$, and the others could be found by finding the root of the 7^{th} order polynomial, which we will not be doing because it is extremely tedious and besides the point of this example. The 0-eigenvector, and therefore the steady-state probability distribution over the possible configurations, can be computed with

a little care, and turns out to be

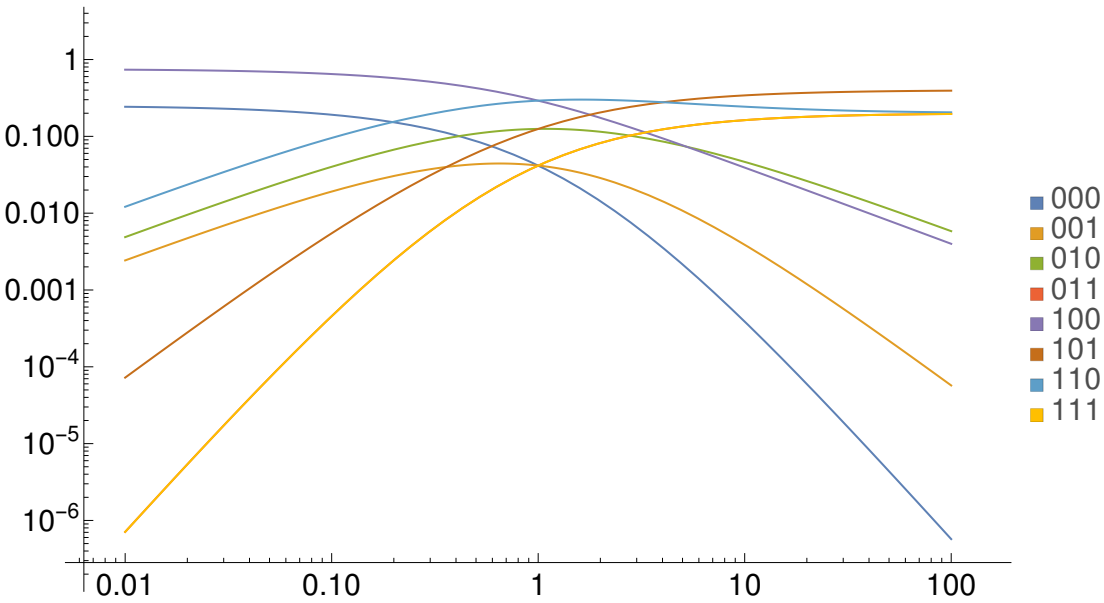
$$q_0 = D^{-1} \begin{bmatrix} 3\lambda+1, \lambda(3\lambda+1) \\ \lambda(\lambda+2)(3\lambda+1) \\ \lambda^3(\lambda+3) \\ (\lambda+3)(2\lambda(\lambda+2)+1) \\ \lambda^2(\lambda+3)(2\lambda+1) \\ \lambda(\lambda(\lambda(\lambda+8)+14)+5) \\ \lambda^3(\lambda+3) \end{bmatrix}, \tag{2.16}$$

where

$$D = \lambda(\lambda(\lambda(5\lambda+26)+37)+24)+4. \tag{2.17}$$

The variation of the occupation probabilities with λ is displayed in Fig. 2.1. Notice that at extremely small λ the system either empties or a single particle

Figure 2.1 *The variation of the steady-state configuration probability distribution with λ for our open system. The coloured curves correspond to specific configurations, as indicated in the legend.*



gets stuck to the full boundary, essentially preventing further flow by blocking it. Meanwhile, at large λ there are a few fairly popular states, with the most dominant being one with alternating particles and vacancies, as one might expect. The steady-state current from left to right in this system can be calculated analytically once we have the steady state probability distribution, and turns

out to be

$$J = \lambda \left[\frac{1}{4} + \frac{(1-\lambda)(3+\lambda)\lambda^2}{4 + \lambda\left(24 + \lambda\left[37 + \lambda\left(26+5\right)\right]\right)} \right]. \tag{2.18}$$

The second term in the bracket is generally dominated by the first, so we can say to an excellent approximation that $J \sim \frac{1}{4}\lambda$. Thus, at least for this tiny example with very extreme boundary conditions, there does not appear to be any change in the scaling of the current with λ.

2.2.3 Formation of the TRM in Sparse Format

As we have seen, it is possible to find the eigendecomposition of the TRM analytically for extremely small systems with special boundary conditions. However, we'd really like to see what happens for somewhat larger systems, with more diverse boundary conditions. Thus, let us try to develop a method for the numerical analysis of TRMs of arbitrary size.

The important thing to note about transition rate matrices for local lattice models such as the SPM is that whilst the TRM itself grows very aggressively with system size, the TRM is generally extremely sparse. The state space dimension grows as 2^L, and the TRM dimension therefore grows as $2^L \times 2^L$. However, a state containing N particles only has transitions to

- states which differ from the current state by one particle move, of which there are $\mathcal{O}(N)$, and
- states which differ from the current state by a single particle creation or annihilation, of which there are $\mathcal{O}(4)$.

Thus, as $N \leq L$ the number of nonzero entries in the matrix is $\mathcal{O}(2^L L)$, which is not a particularly tight bound. Therefore, the overall density of the TRM is $\mathcal{O}(2^{-L} L)$. Given that there already exist a great many efficient numerical routines for sparse linear algebra operations, there exists the possibility that we could use this to solve the SPM on a finite domain for small systems "exactly" (or at least, up to some nominated numerical tolerance).

To make use of this, we need to assemble the TRM in a suitable sparse format. We have written a Python code which does this. The script itself is stored ¡decide how precisely¿, but the gist of the algorithm is to simply run through all possible states, document all the transitions they can perform, and store the resulting

entries in a sparse matrix element by element. During construction, the matrix should be stored in coordinate list format, i.e. a list of elements of the form $(\text{row}, \text{column}, \text{value})$, as this is trivial to update as we only inspect each matrix entry once. We can then convert the matrix to a compressed format such as CSC (Compressed Sparse Column) or CSR (Compressed Sparse Row). We used CSC, but in hindsight CSR would probably have been a better choice as it tends to make matrix-vector multiplication a little more efficient. Once we have the TRM in this format, it is ready for sparse linear algebra operations. Note that these whilst these operations generally happen "in place", this part of the process is the memory-intensive bit, as of course the memory usage scales with the number of nonzero matrix elements, which is $\mathcal{O}(2^L L)$. In terms of actual numbers, we found that 4G of memory was more than adequate to solve a system of 16 sites in total; as the memory required to represent the TRM is the main limiting factor in this kind of calculation, we would recommend computing on machines with large working memories, e.g. DiRAC.

2.3 The Eigenspectrum of the TRM

2.3.1 The Computation of the TRM Eigenspectrum

Once we have the TRM Q in CSC or CSR format, we can then use sparse linear algebra. In our code we called the Python routine `scipy.sparse.linalg.eigs` upon it, which itself is a wrapper for C codes which find eigenpairs according to desired criteria; precisely which algorithm to used is determined during runtime, and it may try different methods if it doesn't initially succeed.

In our computations, we typically performed two types of calculation. In the first we merely sought to find the steady state, so which we requested only the eigenvector x_0 associated to the eigenvalue q_0 with smallest absolute value, which should always be numerically zero. We requested that the eigenpair be found to a relative accuracy ϵ accuracy of 1 part in 10^{12}, which amounts to saying that

$$\frac{\|Qx_0 - q_0 x_0\|}{\|x_0\|} \leq 10^{-12} = \epsilon \tag{2.19}$$

where $\|\cdot\|$ is some reasonable subordinate matrix norm (in our case, the 1 norm). Because we requested the eigenvalue closest to 0, `eigs` used the shift-

invert method ¡find article¿, leading to greater accuracy in the computation of eigenvalues near to 0 which is exactly what we wanted. For the other type of computation, we instead requested the k eigenpairs with largest real part, which most likely provoked the code to use a Implicitly Restarted Arnoldi Method (IRAM). This is not as accurate for computing the steady state, but yields vastly more accurate results when computing the other eigenpairs compared to the first method.

2.3.2 The Structure of the TRM Eigenspectrum

Using the code listed at ¡place¿, we can compute the eigenspectrum of an SPM system with boundary densities connected at both ends. Such a computation requires the following parameters to be specified:

- $\rho_0 \in (0, 1)$, the density of the reservoir connected to the left end of the domain.
- $\rho_L \in (0, 1)$, the density of the reservoir connected to the right end of the domain.
- $L \in \mathbb{N}$, the system size. The way we have defined things in the code, we do not count the two sets of two particles representing the boundary; thus, $L = 1$ actually refers to a system which contains 5 lattice sites, of which 4 are busy doing boundary duties.
- $b > 0$, the variable which controls the separation of timescales between the flickering motion on the boundaries and the internal motion within the bulk of the system. This should be set to something large, and compared to other values of it to ensure that it is working as a regularisation parameter (i.e. large changes to b have minimal impact on the relevant internal dynamics of the system, even if they have a great impact on the boundary dynamics).
- $\lambda > 0$, the internal anomalous movement rate in the SPM.
- ϵ, the relative accuracy the calculation aims for in the sense of Eq. 2.19.
- $1 \leq k < 2^{L+4}$, the number of eigenvalues to compute.

For consistency we kept L, b, ϵ and k constant during runs of calculations. This leaves ρ_0, ρ_L and λ to be varied. In terms of what to vary and how to display

the data, we decided to allow the spectrum to depend upon only one variable, and picked λ to be that variable. We generally chose $\rho_0 = 0.6$ and $\rho_L = 0.4$ in order to study a system in which, at least for $\lambda = 1$, the density is middling and a current flows. We then computed the resulting eigenspectrum in a couple different ways. First, we performed a relatively small calculation, in terms of the number of eigenvalues demanded and the number of λ used. We altered the values of L and b between runs, so that we can see what impact they have on the eigenspectrum. We then performed a very large calculation, in order to get a good look at the eigenstructure as a whole. In Chap. **??** our MFT suggested that a transition might occur around $\lambda = \frac{1}{4}$, so we should look out for odd behaviour in that regime.

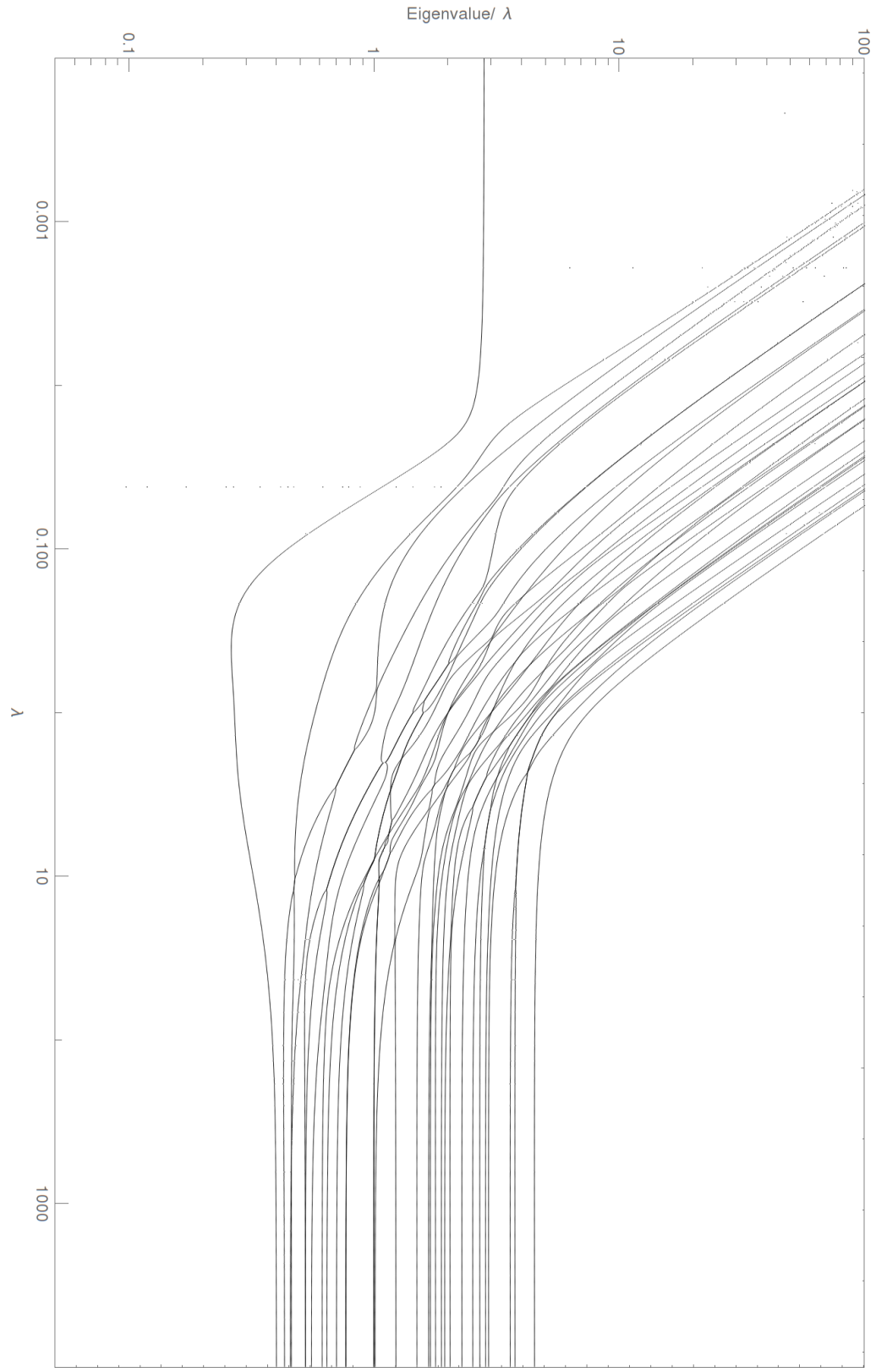
Small Calculations

First, we performed a calculation with $L = 5$, $b = 1000$ in which we computed the negative real part of the 32 eigenvalues with real part closest to zero for a wide range of λ with fixed boundary conditions. The resulting eigenspectrum is displayed in Fig. 2.2. Of course, for such a system we expect that all eigenvalues have negative real part, as we discussed in 2.1, so we have chosen to plot the negative real part divided by λ as a function of λ; the zero eigenvalue, which we expect to exist and is generally found to exist in numerical terms, has been ignored, as its magnitude is simply an artefact of the numerical calculation and is therefore meaningless except possibly as a check on the numerics. The decision to divide by λ was taken because the eigenvalue closest to 0 (which dominates approach to equilibrium) mostly scales as $\mathcal{O}(\lambda)$, so plotting without division uses graph space poorly.

There are a several things to note about this spectrum:

- It would appear that, for all λ, there is a single eigenvalue which is closest to 0 and undergoes no crossing as λ varies. At both extremes of λ, it clearly scales as $\mathcal{O}(\lambda)$, but is unusually low in an intermediate regime of $\lambda \in (0.03, 10)$, reaching a minimum at $\lambda \sim 0.3$. This is important, as the eigenvalue with smallest negative real part is the one that controls the approach to equilibrium; thus, we can see that around 0.3 the system should taken unusually long to relax from an arbitrary prepared state to equilibrium.

Figure 2.2 *The TRM eigenspectrum, computed for a system with* $\rho_0 = 0.6$, $\rho_L = 0.4$, $L = 5$, $b = 1000$.



- All eigenvalues tend to scale as $\mathcal{O}(\lambda)$ as $\lambda \to \infty$.
- Eigenvalues other than the bottom one tend to scale as $\mathcal{O}(1)$ at small λ. Thus, whilst the overall approach to equilibrium occurs with rate $\mathcal{O}(\lambda)$, there are still plenty of (probably more localised) processes occurring over much quicker timescales, whilst the whole system is sluggish to equilibrate.
- The eigenvalues tend to retain their order at the extremes of λ, but they frequently cross, merge and separate again in the intermediate regime. This suggests that something complicated is occurring; an analytic solution to this dynamical model would need to describe these crossings in detail, as they are important as eigenvalue crossing implies the appearance of degenerate eigenspaces in the decay modes which the associated eigenvectors refer to. This makes it look rather unfeasible that such an analytic solution could be constructed in the first place.

Of course, this is only the 32 lowest eigenvalues; there could be different behaviour in the higher ones. Before we perform a larger calculation however, let us turn our attention to the dependence of the eigenspectrum upon b and L, which we have studied by repeating our calculation with different values for those parameters, as shown in Fig 2.3. Again, there's a lot going on in this image, so let's break it down.

- Firstly, let us note that keeping L constant and switching b between 100 and 1000 seems to have very little impact on the eigenspectrum, as we can see by the fact that the points corresponding to the same system sizes are more or less lying on top of each other unless we delve into the low-λ limit of the $L = 5$ system. In the same limit, we see that the discrepancy is vastly reduced in the $L = 10$ system, suggesting that it is less bad for large systems, and is in some sense a small-size effect. Regardless, it would appear to be the case that adjusting b doesn't affect these low-lying eigenvalues very much, so it seems to be playing its role as a regularisation parameter as intended. Thus, we're simply going to use $b = 1000$ from now on, unless explicitly specified otherwise.
- The bottom eigenvalue, which controls relaxation to equilibrium, is generally several orders of magnitude lower for the $L = 10$ system than for the $L = 5$ one; this effect is particularly extreme in the limit of small λ. This makes sense; the larger system should take much longer to equilibrate than

Figure 2.3 *The lower TRM eigenspectrum, computed for a system with* $\rho_0 = 0.6$, $\rho_L = 0.4$, *with all combinations of* $L = \{5, 10\}$, $b = \{100, 1000\}$. *Missing computations, visible via the vertical gaps in the data, are due to computational issues, rather than being numerically meaningful.*

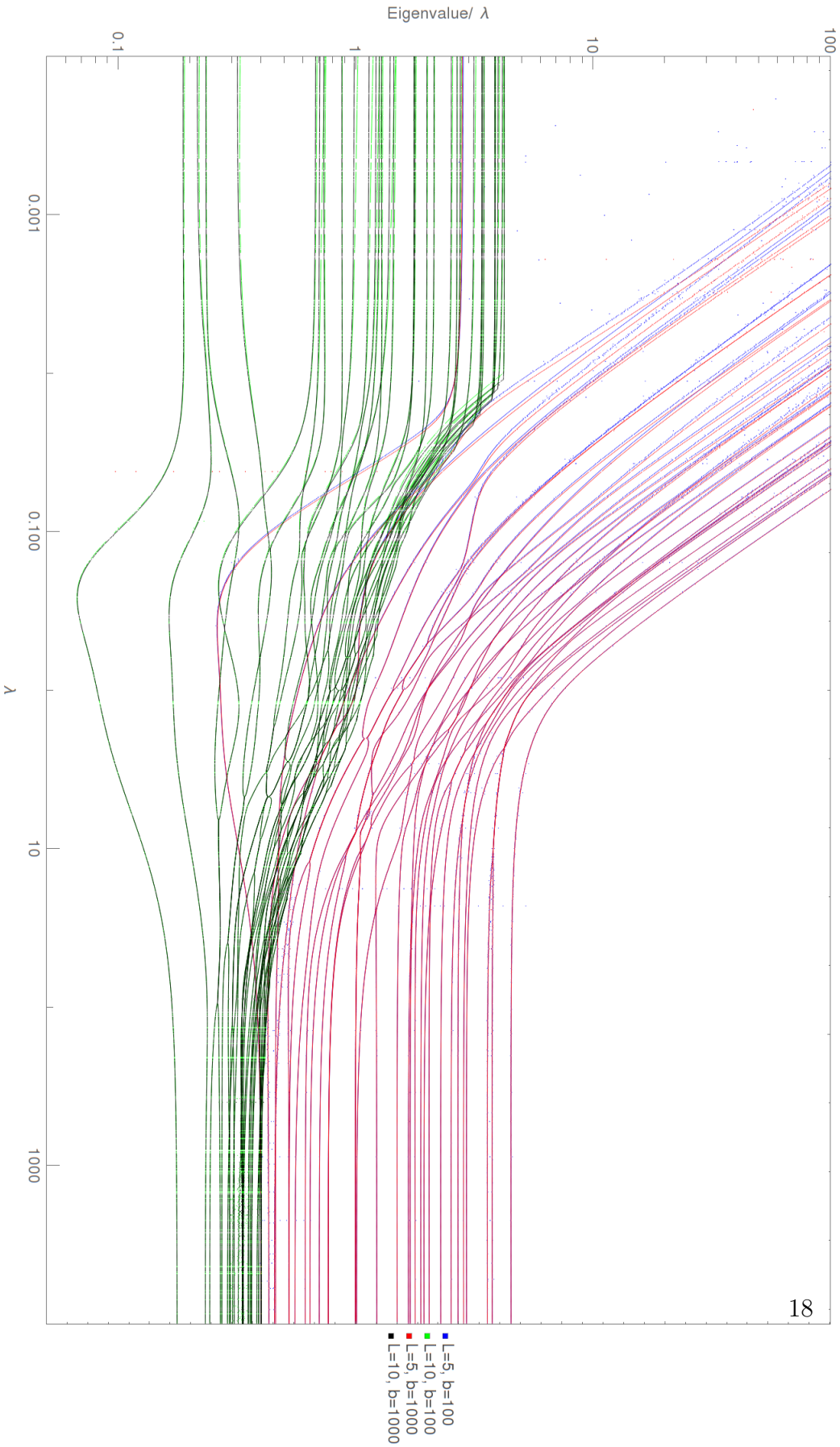


Table 2.1 *A table of the number of slow modes occurring at low λ for given L.*

L	5	6	7	8	9	10	11	12	13
Bandwidth	1	3	6	11	20	36	64	113	199

the smaller one, as so many more fluctuations need to occur to make it change its global state. This is something we should remember for later, as it suggests that equilibration time might scale pretty aggressively with system size, which is bad news for our attempts to simulate the system using Monte-Carlo methods (Chap. 3).

- The big, obvious difference between the spectra of different system size is the behaviour at small λ. For $L = 5$ there is only 1 eigenvalue which scales as $\mathcal{O}(\lambda)$ as $\lambda \to 0$, whereas for $L = 10$, it would appear that they all do!

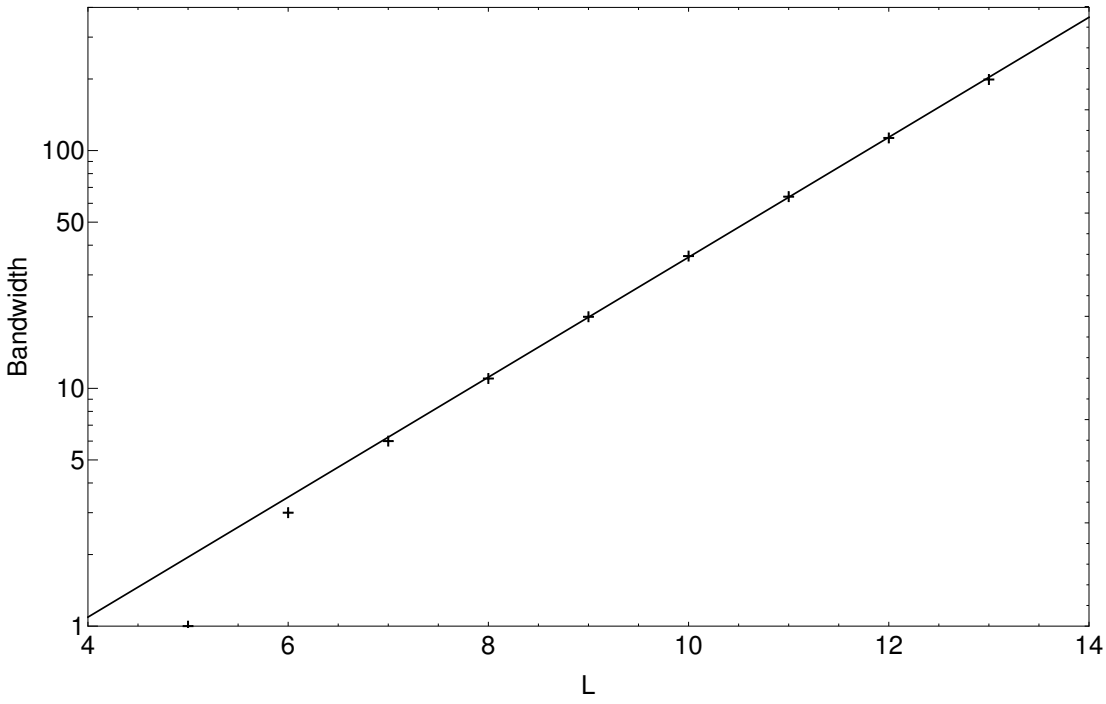
The last point begs the following question: how does the number of eigenvalues corresponding to "slow modes" at small-λ scale with the system size? This is important, because it determines how many of the available decay modes would tend to persist for any meaningful amount of time during the relaxation towards equilibrium. We will attempt to address this in the next subsection.

The Scaling of the Number of Slow Modes at Low λ

To find out how the number of nonzero eigenvalues in the $\mathcal{O}(\lambda)$-scaling band (as observed in Fig. 2.3) scales with system size, we can simply fix λ to be sufficiently low (say, $\lambda = 0.0001$)and repeat computations of the low-lying eigenspectrum for different L. Here, due to the computational limitations we are up against, we only compute up to $L = 13$. The number of slow modes as a function of L is displayed in Tab. 2.1. If we plot the number of slow modes as a function of system size on a plot with a logarithmic y-axis, as we have done in Fig. 2.4, it is rather obvious that the bandwidth scales exponentially with system size. However, the number of slow modes seems to scale as $\mathcal{O}(2^{\sim 0.84L})$, whereas the number of possible states, and therefore the total number of eigenvalues of the TRM, scales as 2^L. Thus, one can see that although the number of slow modes grows very aggressively with the system size, it would appear to eventually become outpaced by the growth in the total number of eigenvalues; in this way, we can say that the slow band carries increasingly less of the total eigenvalue density as system size becomes

large.

Figure 2.4 *A plot of the scaling of the number of slow modes in the eigenspectrum of the TRM of an SPM system of size L. The trendline displayed has equation* Bandwidth $= 2^{a(L+b)}$*, with* $a = 0.84$*,* $b = -3.9$*.*



The Upper Part of the Spectrum

By varying L and b and analysing the lower (more physically relevant) part of the TRM eigenspectrum, we concluded that b has very little impact and essentially acts as a regularisation parameter, whilst L, the system size, has a large impact, particularly on the approach to equilibrium. However, both b and L contribute to the actual numbers contained within the TRM, so it stands to reason that they must impact the eigenspectrum in some way. Therefore, we have done an additional check, by varying b and L and performing a calculation in which the 64 eigenvalues with largest negative real part were calculated. The negative real parts of these eigenvalues are displayed in Fig. 2.5. We see that now the tables are indeed turned. Altering L does have a small effect on the upper band, as we can see by how the red/black and blue/green points lie on top of each other. On this graph, we see that for the $L = 5$ datasets there are apparently two bands,

whilst for $L = 10$ there is only one; however, recall that we have produced this by plotting the negative real parts of the 64 eigenvalues with largest negative real part; thus, as the $L = 10$ systems have $2^5 = 32$ times as many eigenvalues in total, it makes sense that we may not have finished the very top band in the $L = 10$ case yet whilst we have moved on to the second band in the $L = 5$ case; thus, I think we can conclude that system size has little effect on the top of the eigenspectrum.

The value of b, however, has a huge impact, as it determines the overall magnitude of these larger eigenvalues. The eigenmodes associated with them almost certainly relate to fluctuations at the boundary, and then the ensuing joint fluctuations travelling a little into the bulk. As these are boundary effects, it again makes sense that L has little impact. The value of b directly controls the speed of the fluctuations at the boundary, thus the large shift in magnitude observed should be expected.

Large Calculation

Now that we are more or less satisfied that the value of b doesn't matter so long as it's quite large, let's perform a similar calculation (sweeping through λ with constant boundary conditions), but this time compute a lot of eigenvalues in order to get a general overview of the TRM. Plotted in Fig. 2.6 are the negative real components of the 1024 eigenvalues with smallest negative real part, for the TRM of an SPM system of length $L = 8$ and regularisation parameter $b = 1000$. We say that this is the "full" eigenspectrum not because we have computed all of the eigenvalues (we have not, as there are $2^{12} = 4096$ in total), but because the eigenvalues with very large negative real part seem to converge upon each other, forming the hyperbola-shaped entity towards the top of the graph. We believe that this is because the relevant dynamical modes correspond to extremely rapid changes in state due to the boundary conditions.

This large computation continues the themes we have seen in the small ones:

- For large λ there is a thick band of eigenvalues spread over a couple of orders of magnitude which scale as $\mathcal{O}(\lambda)$.
- For intermediate λ the bottom eigenvalue becomes unusually small, whilst the higher ones in the thick band are constantly crossing over each other.

Figure 2.5 *The upper TRM eigenspectrum, computed for a system with* $\rho_0 = 0.6$, $\rho_L = 0.4$, *with all combinations of* $L = \{5, 10\}$, $b = \{100, 1000\}$.

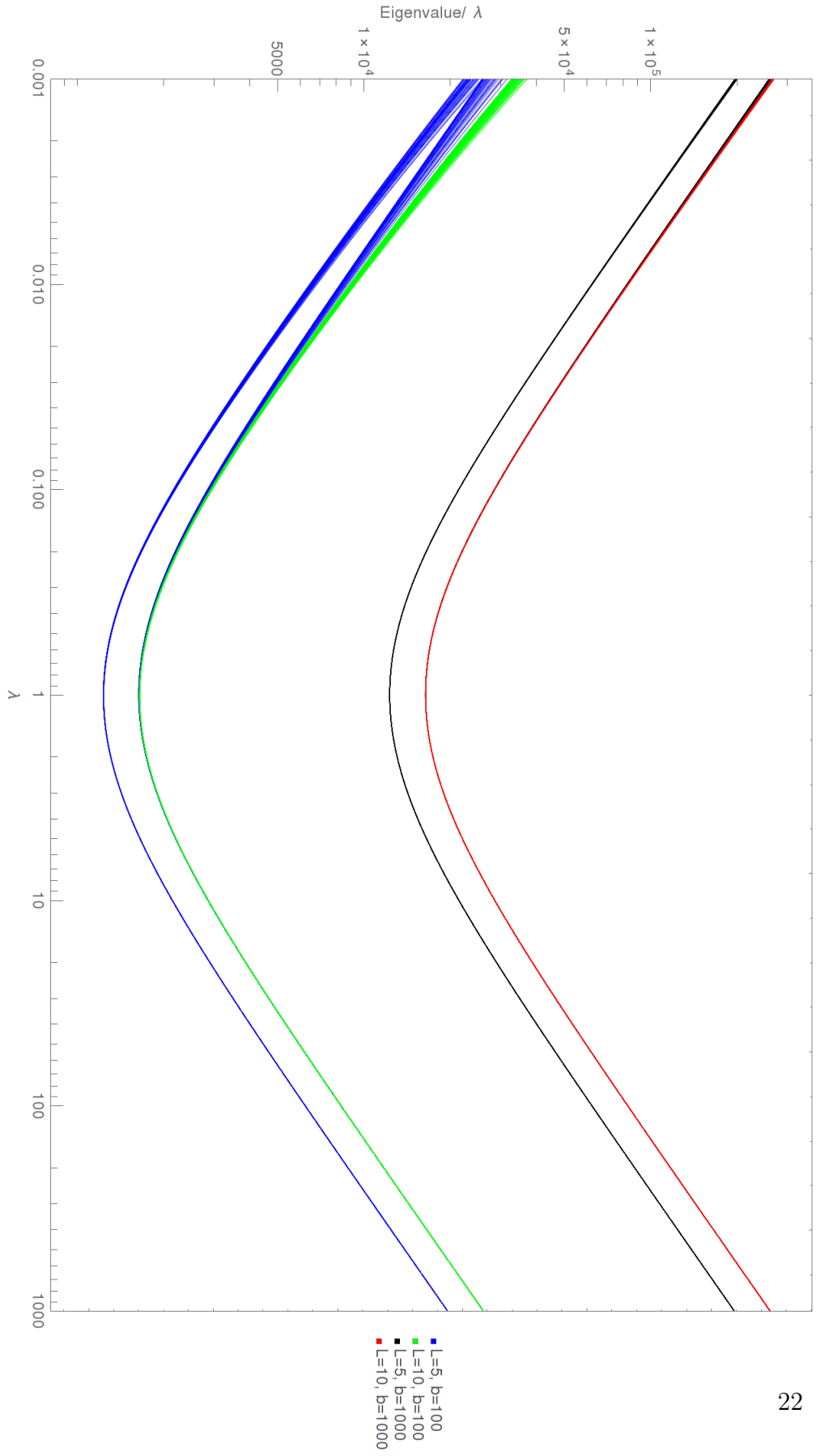
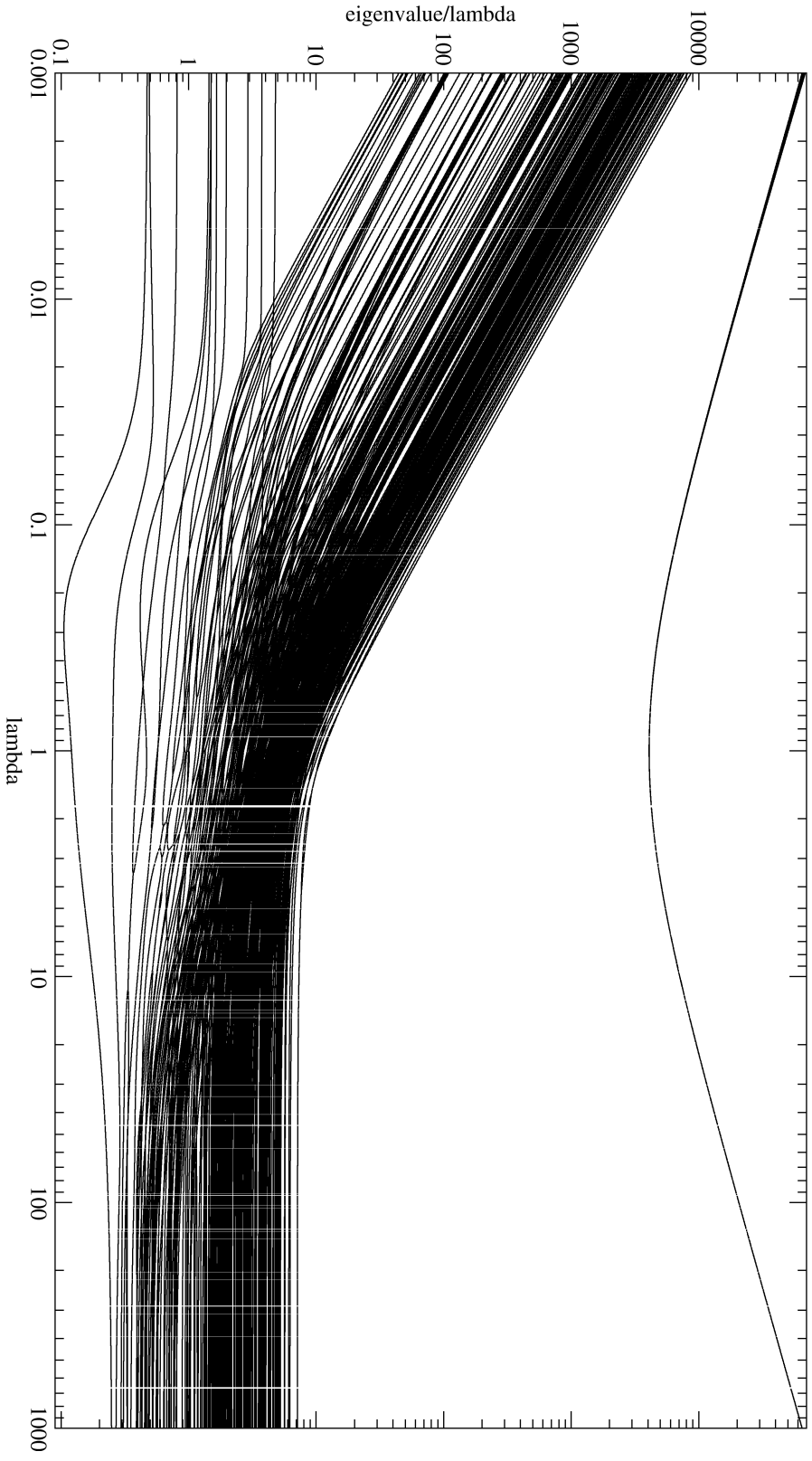


Figure 2.6 *The "full" TRM eigenspectrum, computed for a system with* $\rho_0 = 0.6$, $\rho_L = 0.4$, *with* $L = 8$, $b = 1000$. *Missing computations, visible via the vertical gaps in the data, are due to computational issues, rather than being numerically meaningful.*



- As we go to small λ a thin band of $\mathcal{O}(\lambda)$-scaling eigenvalues split off from the main sequence, which scales as $\mathcal{O}(1)$.
- For all λ there is the hyperbola-shaped band of eigenvalues with highly negative real part, which are many orders of magnitude different to the main sequence; thus, their dynamics operate over an extremely short timescale, and they are almost certainly attributable to the flickering motion at the boundaries.

2.3.3 Current and Density in the Steady State

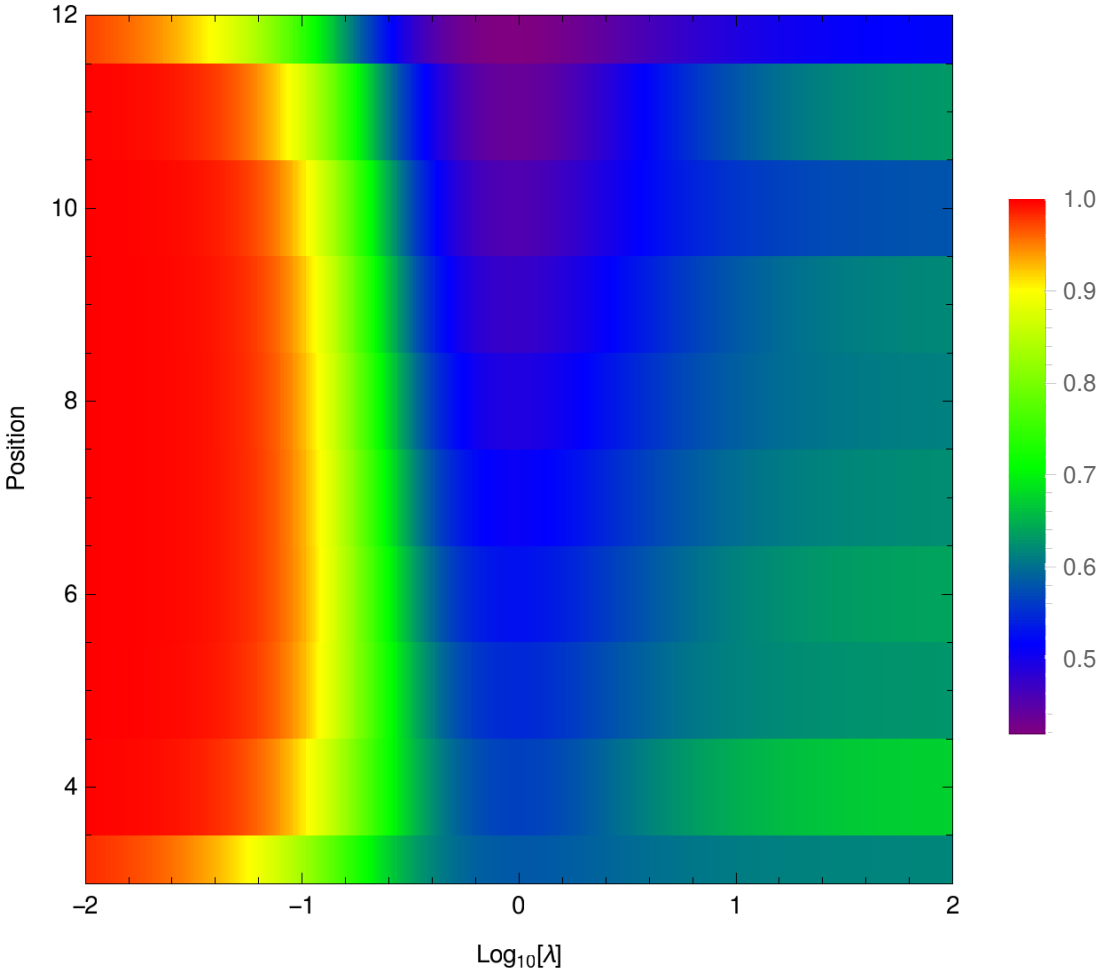
The eigenvectors of the TRM correspond to decay modes, and in general contain both positive and negative components; thus, they do not in general correspond to a particular distribution, only to a time-dependent part of a sum of vectors on their way to equilibrium. The exception to this is the 0-eigenvalue eigenvector, which **does** correspond to a physical state, if properly normalised.

Given a vector corresponding to a normalised probability distribution, we can extract the mean occupation of its i^{th} site by taking the inner product of the distribution vector with a vector which has component 1 on states in which the i^{th} sites is occupied and 0 otherwise. Thus, we can construct an operator P : $\Xi \to \mathbb{R}^{L+4}$ which can tell us the density profile of the system given its ground state, which we can have found using the linear algebraic methods discussed in Sec 2.3.1. We can create a similar operator $J : \Xi \to \mathbb{R}^{L+3}$ which gives the current; however, if used on a steady state, it simply gives a constant result internal to the system, due the the existing constraint that particle number is conserved. We can of course extract the homogeneous current through the system by using this operator and then simply keeping one of the current components.

TRM-Computed Density Profile

By way of example, we computed the density profile for a system with $L = 10$, $b = 1000$, again sweeping through a wide range of scales of λ as in the previous section. We kept the boundaries constant with our usual $\rho_0 = 0.6$, $\rho_L = 0.4$ configuration. A density plot of the data is shown in Fig. 2.7. Looking at $\lambda = 1$, we see that there is a roughly constant gradient for the density profile between the boundaries, which makes sense as this is the simple exclusion limit, so it's

Figure 2.7 *A coloured plot showing the variation of the density profile with λ in steady state for a system with boundary conditions ($\rho_0 = 0.6, \rho_L = 0.4$) of size $L = 10$. Note that the indices of the internal sites are* 3-12 *inclusive, as* 1-2 *and* 13-14 *are reserved for the boundaries, whose densities are already prescribed and therefore aren't plotted here.*



essentially just normal diffusion. As we go to smaller λ, we see that the system in general is quite full. Presumably this is because material has been sucked into the system due to the attraction implied by the small λ; this indicates that the boundaries used are a bit odd in this limit, as the system might not naturally form a homogeneous phase for such a λ and could instead wish to form a mixture of lower and higher density clumps, we will discuss this at greater length in Chap. 3.

Our theory about why this occurs is as follows. At large λ, we see that the density profile defaults to $\rho \sim \frac{2}{3}$ plus oscillations on the boundaries which decay as we go towards the interior. This is suspiciously close to the critical density of the MFT (Chap. **??**) which permits a maximal flow; therefore, we suggest that the system

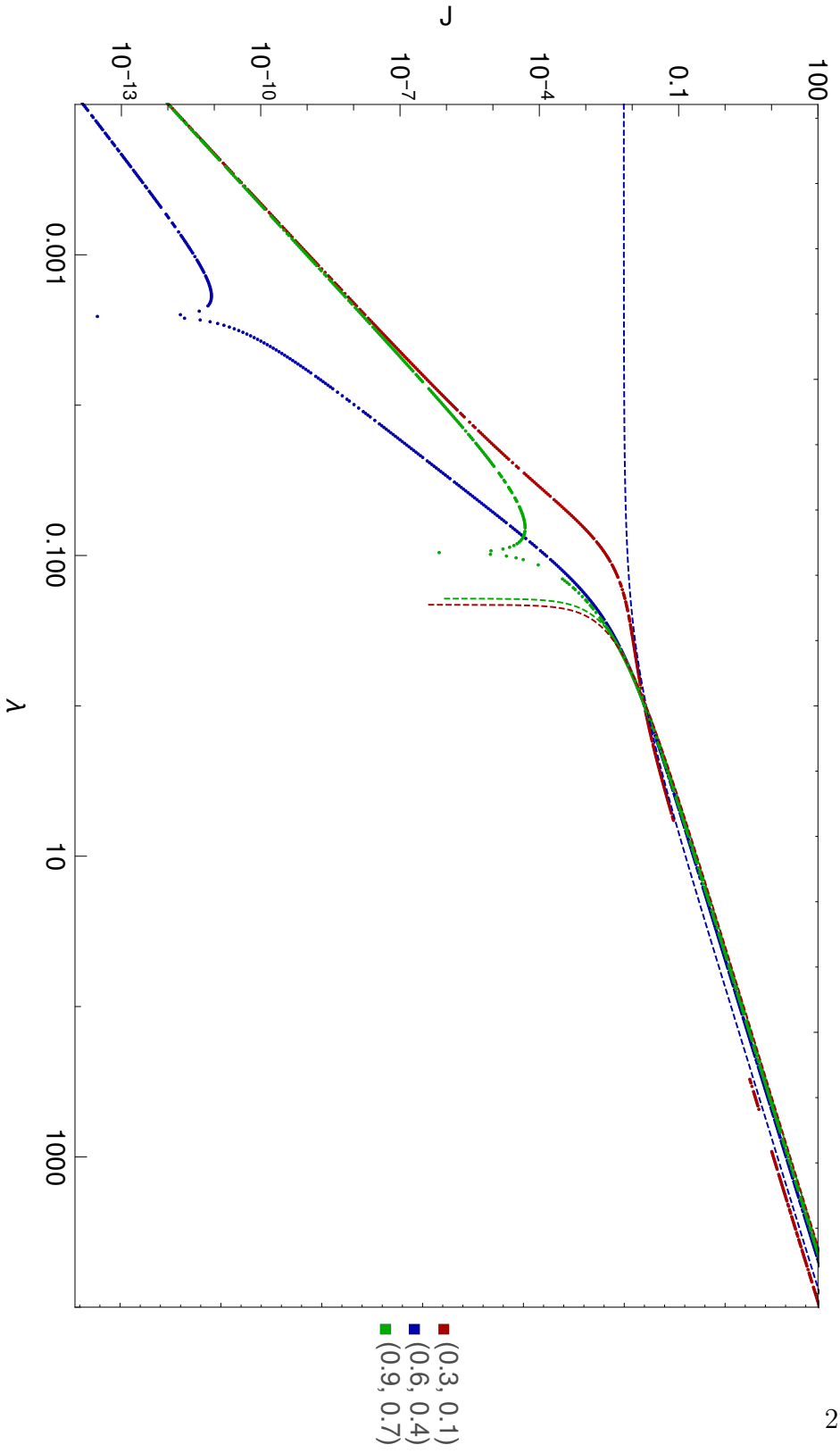
self-organises at high λ to favour configurations that permit high current. It is fairly simple to see why such a configuration would have bulk configuration $\frac{2}{3}$; in such a configuration, any particular particle should on average be in contact with an adjacent particle, with a free space to move into, so the dynamics of the system should be dominated by λ rather than 1. If the density were higher, some particles don't have spaces to move into, so that will slow things down; if the density were lower, particles aren't receiving the speed benefit of being a neighbour. If a boundary fills or empties the system away from $\rho = \frac{2}{3}$, that boundary will have less impact on the local density as it cannot adjust it as quickly as it can around $\frac{2}{3}$, and so the particles from the rest of the system will tend to fill or empty the boundary region. Thus, the only density which permits steady flow is $\rho =\sim \frac{2}{3}$ in the bulk, with relatively rapid matching at the boundaries. In this way, a bulk density of $\frac{2}{3}$ is naturally selected by virtue of its stability. This also explains the oscillations observed at the boundaries; in alternating sites, particles will tend to linger in sites where they are alone, rapidly leaving when they are prompted by the arrival of a new particle in the space next to them.

TRM-Computed Current

Whilst we computed the density profile, we also measured the steady state current. We also performed the same computation for different boundary conditions, and the results are displayed in Fig.2.8, along with the corresponding MFT prediction. Note that some of these calculations failed, in the sense that we were expecting the output to always put the 0-eigenvalue and its associated eigenvector in the first output slot, whereas in reality this did not necessarily happen, even though it had always been the case during testing. in the failed case, we simply haven't plotted any data, although luckily this did not occur in critical places, mainly just for the $(0.3, 0.1)$ calculations for $\lambda > 0$. There's a lot going on here, so let's break it down.

- At large λ, we see that the scaling ($\mathcal{O}(\lambda)$) is the same for both the MFT and the TRM numerics. The actual values predicted by the MFT and the TRM calculations differ slightly, but not by a great deal until we have $\lambda < 1$. It is to be expected that there is a discrepancy, as the MFT is in the continuum-limit whereas the TRM system is only of size $L = 10$. Furthermore, there should be a discrepancy anyway as the TRM system is not mean-field; it just doesn't seem to be so relevant in this regime.

Figure 2.8 *A logarithmic plot showing the variation of current with λ in steady state for a system with boundary conditions (ρ_0, ρ_L) as indicated with system size $L = 10$. The current measurement is oriented so that positive current corresponds to flow from high density to low (the normal case). The relevant MFT prediction is shown via a dashed line in each case.*



- At $\lambda = 1$, both MFT and TRM calculations all coincide, because the boundary conditions all differ by the same amount and $\lambda = 1$ is the trivial case of simple exclusion.

- As we explore very small values of λ, we find that the MFT and the TRM have very little resemblance. For boundary conditions $(0.3, 0.1)$ and $(0.9, 0.7)$ the MFT predicts that the flow should start running backwards or at the very least be pinned at zero, whilst for $(0.6, 0.4)$ it suggests that the current should tend to a constant as $\lambda \to 0$. Instead of the flow crashing to zero, the TRM results show a more gradual reduction of the current. Measurement by comparison with a trendline shows that the current varies as $\mathcal{O}(\lambda^3)$ as λ becomes small.

- There is an intermediate regime in which the dependence of the current upon λ transitions between $\mathcal{O}(\lambda^3)$ and $\mathcal{O}(\lambda)$. This is the situation for $\lambda \in (0.03, 0.5)$.

- There is also some slightly odd behaviour on display in terms of the apparent "poles" in the current for the $(0.6, 0.4)$ and $(0.9, 0.7)$-boundaried systems. This could be due to a close-approach of eigenvalues, causing the algorithm to pick the wrong one under certain circumstances. We have filtered the results by imposing the inequality $\|q_0\| \leq \lambda K$ for $K = 10^{-9}$ with q_0 being the "zero" eigenvalue, but some results may have slipped through this net. We think this is the most likely explanation we have to hand.

So, it seems that whilst we don't see a transition to zero flow as predicted by the MFT, we do see a big change in the way the flow depends upon λ as we pass between the large-λ to small-λ regimes.

TRM-Computed Diffusion Coefficient

It is also possible for us to compute an approximation to the effective diffusion coefficient. Recall that the diffusion coefficient D should satisfy

$$\mathbf{J} = -D\nabla\rho; \tag{2.20}$$

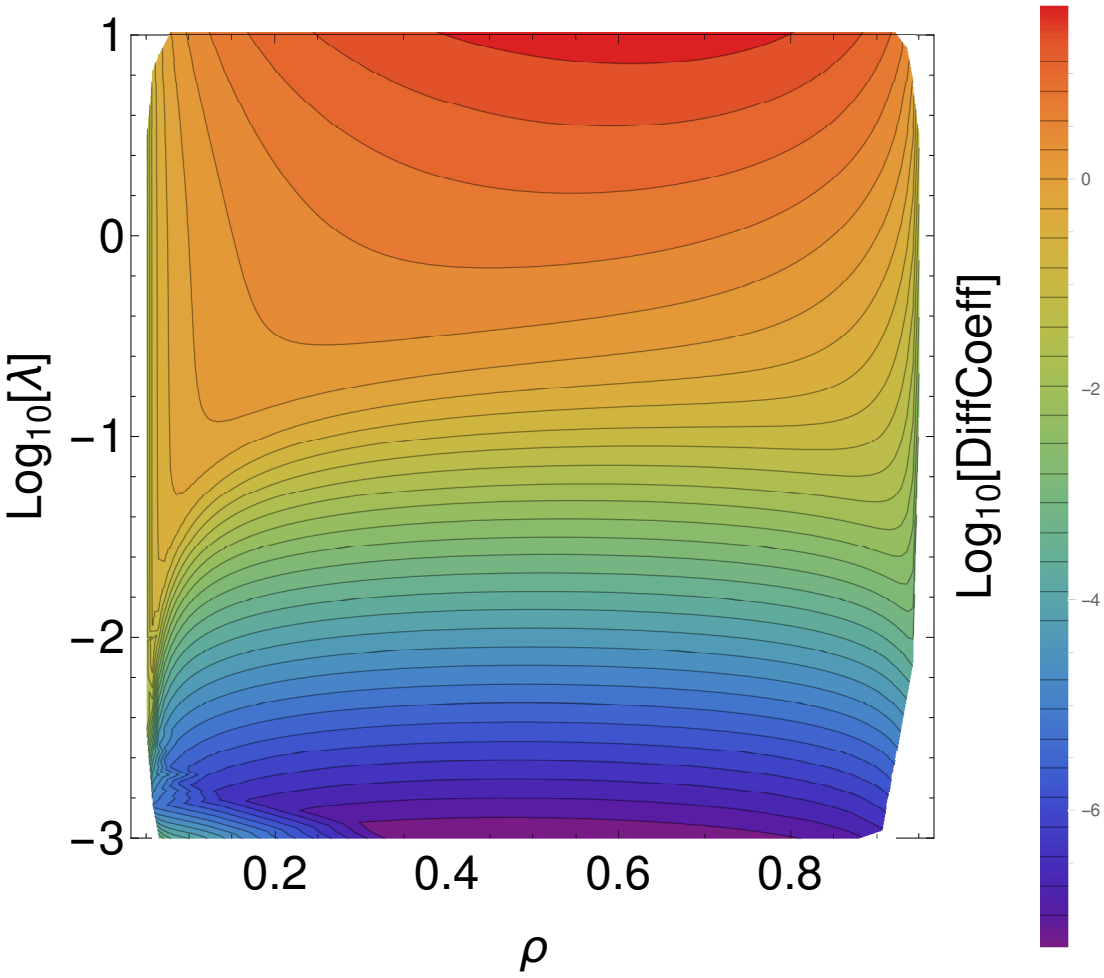
thus, we should be able to compute the diffusion coefficient using the TRM by computing the current J with boundary conditions $(\rho_0, \rho_L) = (\rho + \frac{1}{2}\delta\rho, \rho - \frac{1}{2}\delta\rho)$,

and then evaluating the quantity

$$D(\rho, \lambda) = \frac{L}{\delta\rho} J. \tag{2.21}$$

We performed such a calculation, in which we picked $\delta\rho$ to be 0.001 and $L = 10$. We tabulated D for a range of λ and ρ, and a contour plot of the data is displayed in Fig. 2.9. In general, for given λ we see that the variation with ρ is usually

Figure 2.9 *A coloured plot showing the variation of the diffusion coefficient with λ and ρ in steady state for a system of size $L = 10$.*



pretty mild, with the more extreme diffusion coefficients tending to occur for intermediate ρ. The exception to this occurs primarily at small-λ, low density, where the diffusion coefficient is unusually high. This is presumably because the density of particles is so low that the small value of λ has little impact on the diffusion coefficient. There is also some odd behaviour, in particular for extremely low λ and low ρ. We probably shouldn't trust those the results. As $\lambda \to 0$, the

space of vectors with eigenvalue zero switches from being 1-dimensional to being very high-dimensional, because any state containing adjacent particles becomes a steady state (and in fact an absorbing state). Thus, we should expect that at some as we go to smaller and smaller λ our calculations should start behaving badly because the lower nonzero eigenvalues become numerically indistinguishable from the actual zeros, and that is probably what is happening here.

The trend in terms of λ is that the gradient of the diffusion coefficient as it varies with λ and ρ is generally low for large λ and high for small λ. In light of our results in Fig. 2.8, this makes sense as the power law seems to change as we switch between low and high λ. This suggests that an order parameter of the form

$$\chi(\rho, \lambda) = \left(\frac{\partial \ln D}{\partial \ln \lambda} \right)_{\rho} \tag{2.22}$$

should reveal the power-law structure. We can compute this from our existing data in Fig. 2.9, and it is displayed in Fig. 2.10. As you can see, this order parameter does seem to nicely partition (ρ, λ) space into two components, divided by a region of extremely rapid change. The order parameter χ could be regarded as some kind of susceptibility. An issue here is that it is hard to see how χ's apparent transition will vary with system size, as $L = 10$ is a very small system, and transitions only become sharp in the limit of large systems. We could of course use it in a Monte-Carlo calculation in a larger system, but then we have the problem that it's difficult to take meaningful derivatives of noisy data. We will leave it, then, as a curiosity.

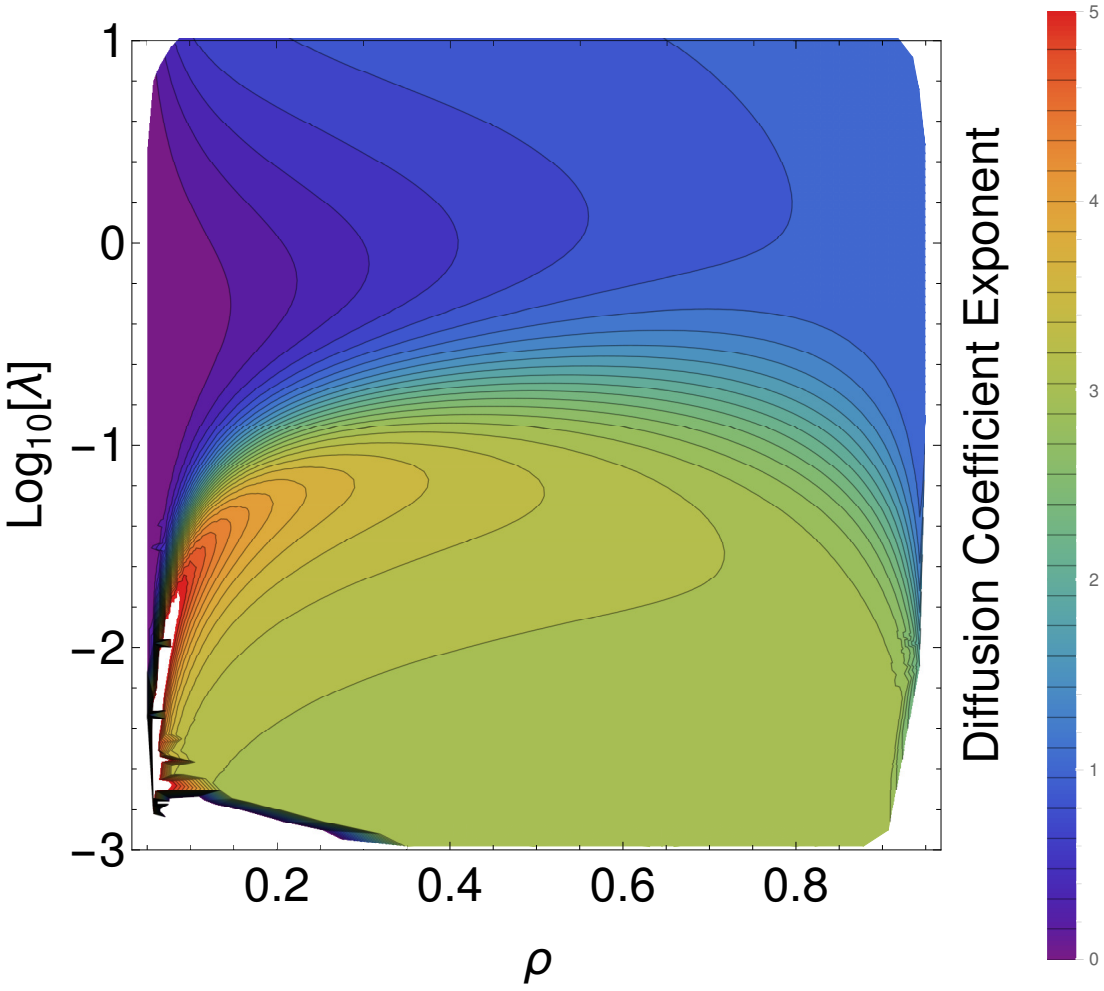
2.4 Time-Dependent Properties of Small SPM Systems

Although we have mostly concentrated on calculating steady state properties of the SPM, it is also possible to calculate some dynamical properties.

2.4.1 The Relaxation Time for the SPM

As we have seen, the eigenvalue with least negative real part effectively controls the rate at which a generically-prepared system equilibrates, by acting as a lower

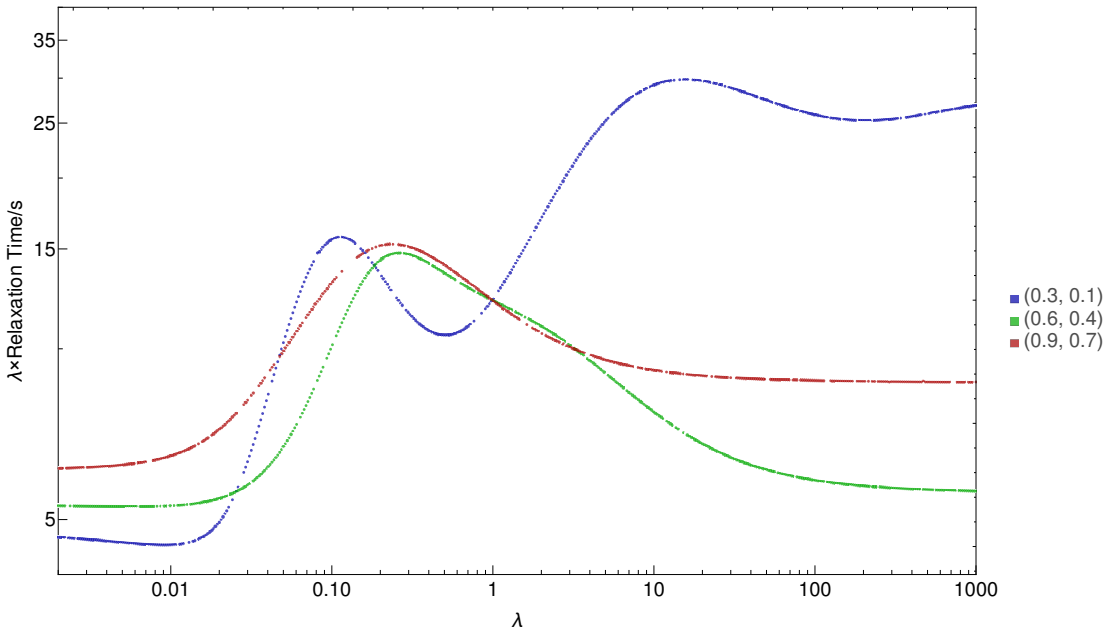
Figure 2.10 *A coloured plot showing the variation of the order parameter χ with λ and ρ in steady state for a system of size $L = 10$.*



bound on the rate of relaxation towards equilibrium. We have computed this relaxation time for our three standard boundary conditions in Fig. 2.11. Notice first that we have plotted the relaxation time multiplied by λ, as the relaxation time generally scales as $\mathcal{O}(\lambda^{-1})$. One can see why we made this choice by looking at the extremes, where the lines are generally quite flat.

Observe that the relaxation times for large λ are generally a little higher than for low λ once the $\mathcal{O}(\lambda^{-1})$ dependence is taken into account. In particular, the system with low densities at the boundary is much slower to relax to equilibrium than the system with high boundary densities, which in turn is slow compared to the system with medium densities. This is presumably because the system is attempting to reach the maximal flow density we observed, $\rho = \frac{2}{3}$, and it finds it more awkward to do that the further away the furthest boundary is from $\frac{2}{3}$.

Figure 2.11 *A plot showing the dependence of the relaxation time upon λ for a system of size $L = 10$ with boundary densities $(0.3, 0.1)$, $(0.6, 0.4)$ and $(0.9, 0.7)$.*



For very low λ, the difference between boundary configurations isn't so great, and we believe that the differences between the boundaries occur for the same reason as for the high-λ situation, only now the system is trying to fill instead of hold at $\rho = \frac{2}{3}$. For intermediate λ, things are less clear. At $\lambda = 1$, all systems equilibrate at the same rate; other than that, behaviour varies pretty wildly. Equilibration time is generally on the high side for $\lambda \in (0.03, 1)$, which is where the current looks like its undergoing some kind of transition, so this increase in relative equilibration time could be interpreted as an accumulation of fluctuations in that regime.

2.4.2 Time-Evolution of States

Recall from Sec. 2.1 that the Master Equation (Eq. 2.1) has a formal solution given by Eq. 2.8, in which we premultiply the initial state by a matrix exponential of the TRM. Throughout this chapter we have been making use of the fact that the TRM is sparse in order to perform our computations. Thus, it would seem that we couldn't investigate the time evolution of systems, as e^A is in general not sparse even if A is sparse, and thus we would instantly run out of memory if we

tried to compute anything.

However, the premultiplication of a vector by a sparse matrix can be performed *in place*, and this is implemented by Python's routine `scipy.sparse.linalg.expm_multiply`. Therefore we have been able to calculate the time-evolution of arbitrary initial distributions. By way of an example we prepared three systems with initial uniform distributions, in which all possible configurations are equally likely, and then monitored how their spatial occupations varied as they relaxed towards equilibrium (which should always occur, as discussed in Sec. 2.1). The results are displayed in Fig. 2.12. In the plots we have included the innermost site in the boundary layer; notice that it switches to the "correct" mean occupation almost instantly, which is exactly what is supposed to happen as it represents a highly responsive reservoir. The system then gradually makes its ways toward equilibrium, starting at the boundaries and working inward, and the characteristic timescale over which this occurs seems to be in line with our results from 2.4.1.

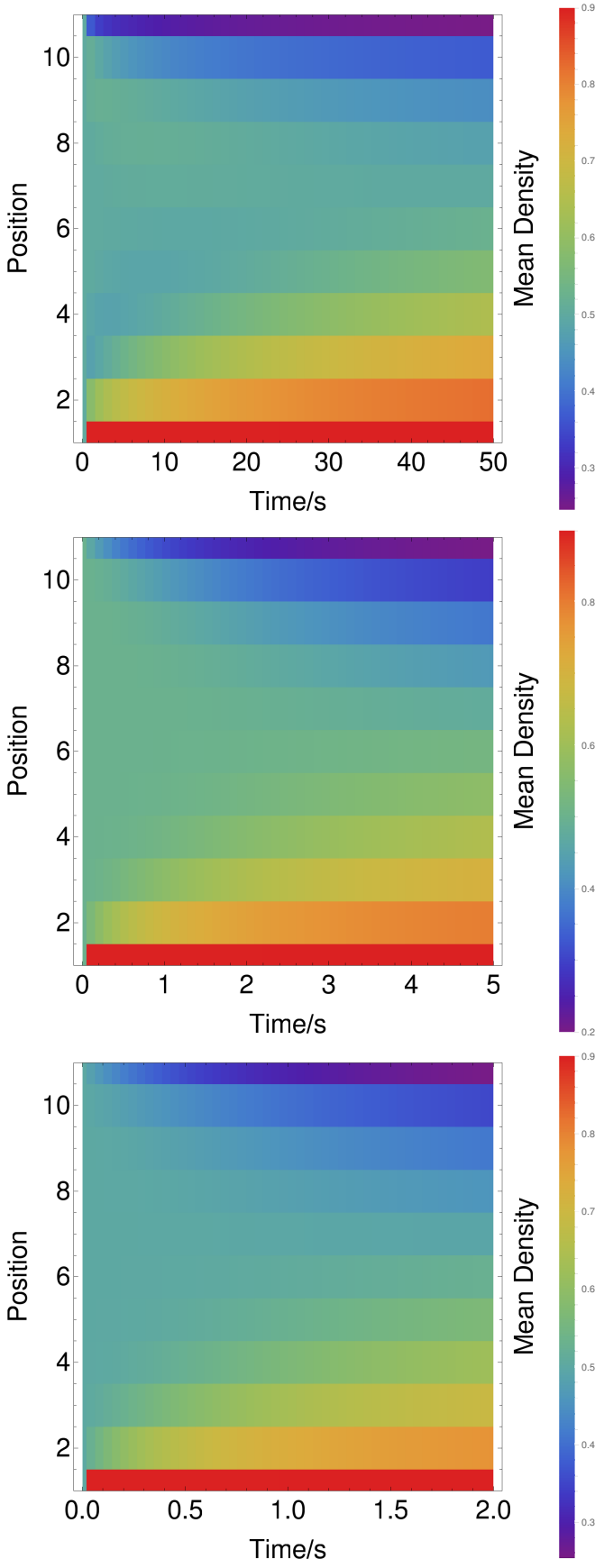
Of course, this is more of a demonstration of what this method could achieve than actually useful results. We are quite sure that with a little effort one could use it to calculate time-dependent spatial correlation functions, for example, but we simply worked out how to perform TRM calculations too late in the project to be able to use it to full effect.

2.5 Conclusions

Numerically approximating the eigenpairs of the transition rate matrix is a convenient alternative to the use of Monte Carlo methods for the computation of the properties of a Markovian statistical mechanics system. When using the TRM, we sacrifice system size for accuracy, as the memory requirement for TRM calculations scales exponentially with system size, whilst similar Monte Carlo calculations scale linearly. However, the TRM method avoids the sampling issues which often emerge when attempting Monte Carlo simulations, and can yield the relevant statistics with relative ease once the matrix computations are completed. Furthermore, we can also investigate time-dependent properties at our convenience, which can be awkward in Monte-Carlo calculations as there we don't usually work with normal time.

The main reason we performed these calculations, however, is to investigate

Figure 2.12 *Plots of the time-evolution of the density profiles of systems prepared with uniform distributions. These systems are of size* $L = 10$*, with boundary densities* $(0.9, 0.1)$*,* $b_0 = 100.0$ *and* $\lambda = \{0.1, 1.0, 2.0\}$*, going from top to bottom, respectively. The total time intervals have been adjusted to suit the equilibration time.*



the current flowing due to the boundary conditions. We found that the MFT prediction that the currents drops to zero or becomes negative below a critical λ (depending upon boundary conditions) does not seem to be correct; however, we have found that the scaling of the current does seem to undergo some kind of transition, from $J \propto \lambda$ for high λ to $J \propto \lambda^3$. This is something which we can investigate further using Monte-Carlo methods on larger-scale systems, and we will do this in the next chapter.

Chapter 3

Monte-Carlo Simulations of the SPM

We now have numerical results for SPM systems using TRM analysis; however, this only allows us to study relatively small systems. In order to study larger ones, we have used Monte-Carlo methods. In this chapter, we will discuss the methods we used, the results they yielded and their meaning, with particular emphasis on what they tell us about the suspected transition between low and high-λ behaviours.

3.1 Numerical Simulations of Continuous-Time Markov Processes

3.1.1 Purpose of Monte-Carlo Methods

We should first really describe what we mean by a Monte-Carlo method. In essence, Monte-Carlo methods refer to numerical routines in which we attempt to characterise an unknown distribution by using pseudorandom numbers in order to produce sample data which is hopefully faithful to the original distribution, at least in terms of the statistics we are trying to calculate. A good example of a commonly-used Monte-Carlo method in Physics is the Metropolis-Hastings algorithm, which in its original for is used to calculate statistics for equilibrium statistical mechanics systems.

In our situation, we wish to be able to mimic a continuous-time discrete-state Markov process. As we saw in Chap. **??**, the state space for a TRM system of size L scales as $\mathcal{O}(2^L)$; thus we quickly run out of size if we try to consider exactly probability distributions, which correspond to vectors in $\mathbb{R}^{L^2}$. We can, however, store individual configurations, which only occupy $\mathcal{O}(L)$ space. Therefore, we need to find a way to produce trajectories through the discrete state space which sample the actual space of system trajectories well enough to allow us to access the statistics we want. Of course, there isn't a unique "best" way to do this. We have considered two contrasting methods, which differ primarily in the way in which they convert the original continuous time into discrete steps which we can use in an algorithm.

3.1.2 Evenly-Spaced Timesteps

3.1.3 The N-Fold Way, or Gillespie Algorithm

3.2 Implementation of Monte Carlo Methods

3.2.1 Our Implementation of a Metropolis-Hastings

3.2.2 `KMCLib`

Talk about how it works, why I picked it over other implementations.

3.2.3 Running our Monte-Carlo Calculations

How calculations are managed day-to-day.

3.3 1D Calculation Results

3.3.1 Calculational Choices

3.3.2 Flow Patterns

3.3.3 Current

3.3.4 Density

3.3.5 Diffusion Coefficient

3.4 2D Calculation Results

3.4.1 Calculational Choices

3.4.2 Results

3.5 Conclusions

Chapter 4

Conclusions

Need to summarise the key results of the research here, and give an overview.

Appendix A

Code Listings

A.1 1d Ising Correlation Functions

This `Python` script computes the probability of a site being occupied l lattice spacings away from an occupied site. It requires the system size L and the number of particles N as inputs. The output is saved in a file called `corrFnResults.m`, which is formatted so that it may be used by `Mathematica`.

```
import copy
import sys

def configMake(L, N, prevList, totList):
    if L==1:
        endList = [copy.deepcopy(prevList), N]
        totList.append(unfold(endList))
        return [N]
    if N==0:
       return configMake(L-1, 0, [copy.deepcopy(prevList), 0], totList)
    if L==N:
        return configMake(L-1, N-1, [copy.deepcopy(prevList), 1], totList)
    return [configMake(L-1, N, [copy.deepcopy(prevList), 0], totList),
    configMake(L-1, N-1, [copy.deepcopy(prevList), 1], totList)]

def adjSum(candList):
    listLen = len(candList)
    total = 0
    for index in range(0, listLen):
        total += candList[index-1]*candList[index]
    return total

def unfold(candList):
    if isinstance(candList, list):
        if len(candList)==2:
            return unfold(candList[0])+unfold(candList[1])
```

```
        if len(candList)==1:
            return candList
        if len(candList)==0:
            return []
    return [candList]

def listCollate(candList):
    maxItem = 0
    for index in candList:
        if index > maxItem:
            maxItem = index
    outPut = []
    for size in range(0, maxItem+1):
        numCounts = 0
        for index in candList:
            if index == size:
                numCounts += 1
        outPut.append((size, numCounts))
    return outPut

def genCorrFn(L, N):
    totList = []
    allStates = configMake(L, N, [], totList)
    restStates = []
    weightList = []
    maxAdj = 0
    for state in totList:
        if state[0]==1:
            restStates.append((state, adjSum(state)))
            if restStates[-1][1]>maxAdj:
                maxAdj = restStates[-1][1]
            weightList.append(restStates[-1][1])
    partFnList = listCollate(weightList)
    print(partFnList)
    partitionFn = "("
    for pair in partFnList:
        partitionFn += str(pair[1])+"␣Exp["+str(pair[0]-maxAdj)+"b]␣+␣"
    partitionFn += "0)"
    print(partitionFn)
    finalOut = "{"
    for shift in range(0, L-L/2):
        tempList = []
        for config in restStates:
            if config[0][shift] == 1:
                tempList.append(config[1])
        stateDist = listCollate(tempList)
        outSum = "{"+str(shift)+",␣("
        for pair in stateDist:
            outSum += str(pair[1])+"␣Exp["+str(pair[0]-maxAdj)+"b]␣+␣"
        outSum += "0)/"+partitionFn+"}"
        finalOut += outSum
        if shift != L-L/2-1:
            finalOut += ",␣"
    finalOut+="}"
    return finalOut

L = int(sys.argv[1])
```

```
with open("corrFnResults.m", 'w') as f:
    f.write("{")
    for n in range(2, L-2):
        f.write("{"+str(n)+"/"+str(L)+",␣"+genCorrFn(L, n)+"},␣")
    f.write(genCorrFn(L, L-2) + "}")
```

A.2 n-Dimensional Continuum-Limit MFT

This Mathematica script computes the current which flows between two adjacent sites (offset in the e_1 direction) in the MFT of the n-dimensional SPM; due to symmetry, this tells us what happens in an arbitrary direction. In this case n is set to 3, but it still works if changed to any positive number.

```
n = 3;
i = 1;
zero = 0*UnitVector[n, 1];
e[i_] := UnitVector[n, i];
Hess = Table[
   Piecewise[{{d2p[j, i], j > i}}, d2p[i, j]], {i, 1, n}, {j, 1, n}];
Jacob = Table[dp[i], {i, 1, n}];
p[x_] := p0 + Jacob.x + 1/2 x.(Hess.x);
rightJ = 1/
    t0 (1 - p[1/2 a e[i]]) p[-(1/2) a e[i]] (1 -
     z p[-(3/2) a e[i]]) Product[
    Piecewise[{{(1 - z p[-a e[j] - 1/2 a e[i]]) (1 -
          z p[a e[j] - 1/2 a e[i]]), j != i}}, 1], {j, 1, n}];
leftJ = 1/t0 (1 - p[-(1/2) a e[i]]) p[
    1/2 a e[i]] (1 - z p[3/2 a e[i]]) Product[
    Piecewise[{{(1 - z p[-a e[j] + 1/2 a e[i]]) (1 -
          z p[a e[j] + 1/2 a e[i]]), j != i}}, 1], {j, 1, n}];
fullJ = rightJ - leftJ + O[a]^3;
FullSimplify[fullJ]
```

Bibliography