DIFFUSION MEDIATED SIGNALING: INFORMATION CAPACITY AND COARSE GRAINED REPRESENTATIONS

by

Matthew Thomas Garvey

Submitted in partial fulfillment of the requirements

For the degree of Master of Science

Thesis Adviser: Dr. Peter J. Thomas

Department of Mathematics

CASE WESTERN RESERVE UNIVERSITY

May, 2009

CASE WESTERN RESERVE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis/dissertation of

Matthew T. Garvey

candidate for the MS Applied Math degree *.

(signed) Peter J. Thomas
(chair of the committee)

Marshall Leitman

Robin Snyder

(date) December 15, 2008

*We also certify that written approval has been obtained for any proprietary material contained therein.

Contents

List of Figures

Acknowledgements

Thank you to all who have helped with this paper and my graduate study at CWRU. To name just a few: Deep gratitude and debt go to Dr. Peter Thomas, my advisor on the thesis, for all his help, prodding, and patience, as well as the use of the Computational Biomathematics Laboratory computers. To the committee members, Dr. Marshall Leitman and Dr. Robin Snyder, for reviewing the work and assisting along the way. To Dr. Stanislaw Szarek, my other advisor, in particular for suggesting the way to go for a second MS, of which this is a part. To my other professors, office mates, etc. at CWRU, whom I cannot enumerate or number in full, and to the semi-backless chair with exposed staples that shall serve as a representative example of daily office life. To my family and my dear friend John Banionis for support and constantly asking how the thesis was coming.

A thesis is always locked to its final form in haste; any errors I have failed to correct are my responsibility and not those of other proofreaders, be they confusingly worded mathematics or the (much more vexing in their minuteness) inevitable forgotten formatting changes, English usage, or plain typos.

Hugh: I'll get the dictionary.
Lisa: Why?
Hugh: You'll see when you get there: the word "stochastic".
Lisa: Pertaining to a process involving a randomly determined sequence of observations. (nervous laugh)
Both: (A beat, then embrace.)

—*The Simpsons*, episode #2F15

Abstract

Diffusion Mediated Signaling:

Information Capacity and Coarse Grained Representations

by

Matthew Thomas Garvey

Communication via diffusible chemical signals is ubiquitous within biology. We explore a model of a biochemical communications channel using a diffusible chemical signal transmitted across a volume. The received signal is attenuated by a combination of diffusion, decay, and counting noise. We find the response of the channel is well fit by an additive Gaussian noise model $y = \beta x + z$, where $x \in \mathbb{C}$ is the Fourier component of an arbitrary sinusoidal input at a frequency ω, $z \in \mathbb{C}$ is complex bivariate Gaussian noise with variance N, and $\beta \in \mathbb{C}$ and $N \in \mathbb{R}$ depend systematically on ω. We impose a natural constraint A on the input amplitude, and find the information capacity on a single frequency, and on several together by waterfilling. We also consider several different coarse grainings of the model and how the loss of detail affects apparent information capacity.

1 Introduction

Communication by means of diffusible chemical signals is ubiquitous within biology. Examples include quorum sensing within bacterial colonies [9] and triggering of developmental events and directed cell motion by spatial and temporal gradients of 3'-5'-cyclic adenosine monophosphate (cAMP) [6], localization of pathogens by white blood cells in the human body [7], and chemical signaling across the synaptic cleft separating an axon terminal from a postsynaptic dendritic spine [5].

Abstracting from these examples, we explore a few aspects of a diffusion-based signaling system, in which a passive diffusion process carries information through a medium connecting two locations. A motivating example is how varying concentrations of cyclic AMP may be used by cells such as *Dictyostelium discoideum* to trigger developmental events or movement [8]. Although we use an enclosed "cell" in the model we shall introduce, it represents an extracellular space in which one object (a synaptic release site localized to one point in space, for example) emits particles that are detected elsewhere by another object.

Our first goal is to estimate the information capacity of such a system. Information capacity is defined as the maximum of the mutual information between possible inputs and outputs in a certain physical information channel. In a classic model, the additive white Gaussian noise (AWGN) channel [3], inputs and outputs are realized as numbers in $\mathbb{R}$, and each transmission involves the sending of one signal which is received in a form corrupted

by additive Gaussian noise. In our model, the inputs for a single transmission are fixed-length time series comprising counts of particles injected into the system at a particular location. The outputs for a single transmission are corresponding time series representing the numbers of particles detected at another location. In general, the average number of particles injected is well above the number received, because the channel includes a linear decay process. Similarly, signaling patterns at lower frequencies are received more faithfully than those at higher frequencies because of attenuation from the decay and diffusion processes.

We define a mathematical model of a simplified diffusion process within an 8×16 lattice with fixed locations for particle injection and detection at opposite ends. We then use the results of simulations to compare the model to the AWGN channel and estimate the information capacity of the system.

Our second goal is motivated by the concept of coarse graining. Many biological models involve large numbers of nodes, which can increase the complexity of the model and the computational expense of simulations and analysis. Coarse graining is a means of representing a system with many nodes by another with fewer nodes. For example, grouping amino acids in large proteins into clusters of m amino acids allows for a reduction in computing time of m^3 and a reduction in memory requirement of m^2 in simulations of protein dynamics [2]. Here, we experiment with several ways of coarse graining our discrete diffusion lattice into 4-node clusters.

The act of coarse graining results in loss of information about the detailed configuration of a system. In many applications it is desirable to find a

coarse graining whose loss of information is as small as possible. Intuitively, one expects "useful" coarse grainings will somehow make a natural fit to a model's structure or dynamics; one may even hope that the process of finding a "good" coarse graining will provide insight into the function of whatever network is under investigation. Information theoretic performance measures such as the channel capacity offer a novel means of comparing alternative coarse grainings. We experiment on a small sample set of coarse grainings in order to see which work well. Our numerical experiments on the diffusion-based communications channel and its coarse-grained representations suggest a novel result: the more detail lost in the coarse graining, the higher the apparent information capacity of the coarse-grained channel.

In the remainder of the Introduction we review a mathematical model for transition matrices in Markov chains, and then the standard analysis for the capacity of a single AWGN channel and how it can be used in the multichannel case, following [3]. In Methods we detail our diffusion model and simulations, how they are adapted for coarse graining, and an approach to get some analytic and non-stochastic results using the transition and coarse-graining matrices. The Results section contains the analysis of the simulations and comparison to the Gaussian channel for both the original and coarse-grained models, then an adaptation of the Gaussian channel's information capacity model for one channel alone and multiple channels in parallel. This situation applies to our model by a standard frequency domain decomposition of the input signals. Finally, in Discussion we consider a few possible continuations of the project and present a few additional results of interest.

1.1 Transition Matrices

One way to consider a particle's movement is as a Markov chain with a transition matrix. The same transition matrix can also be used to look at behaviors of several particles at a time, for example to find a steady-state distribution.

In order to represent the transition probabilities, let w_t represent the index $w_t \in \{1, \cdots, N_{\text{nodes}}\}$ of a particle at time t. Define the transition matrix to be

$$\begin{aligned} \mathbf{P}_{ij} &= \Pr\{w_{t+1} = i | w_t = j\} \\ &= \begin{cases} p & (i,j) \text{ are neighbors;} \\ 1-4p & i=j \text{ is an internal node;} \\ 1-3p & i=j \text{ is an edge node;} \\ 1-2p & i=j \text{ is a corner node;} \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{1.1.1}$$

The parameter p of course cannot be greater than $\frac{1}{4}$. Let $\mathbf{n}_t$ be a column vector representing the location of a particle at time t. To represent a particle located at position $w_t = j$ at time t, $\mathbf{n}_t$ would contain a 1 in its jth entry and 0 elsewhere. $\mathbf{P}_{ij}$ is the probability that $w_{t+1} = i$ given that $w_t = j$. Then $\mathbf{n}_{t+1} = \mathbf{P}\mathbf{n}_t$ is a vector containing the probabilities of the particle's locations at the next timestep. Similarly, if a single particle takes position j at time t with probability $\mathbf{n}_t^{(j)}$, then it takes position i at time $t+1$ with probability $\mathbf{n}_{t+1}^{(i)} = \sum_j \mathbf{P}_{ij}\mathbf{n}_t^{(j)}$.

If we let $\mathbf{n}_t$ be more generalized, with a nonnegative integer for each

position i representing the total number of particles at that position at time t, then $\mathbf{P}\mathbf{n}_t$ shows the expected number of particles after one timestep. Rather than placing each particle into a new position, it "distributes" the particle into several positions, based on the transition probabilities. Invoking the central limit theorem[1], if the particle density is sufficiently high then the population vector obtained by the random movement of particles will be close to $\mathbf{P}\mathbf{n}_t$.

This interpretation of $\mathbf{P}$ allows us to obtain analytical results, for example via eigendecomposition. Because $\mathbf{P}$ is a symmetric matrix, it has real eigenvalues with orthogonal eigenvectors. Because the nodes of the lattice all intercommunicate, $\mathbf{P}$ has a unique maximal eigenvector with unit eigenvalue (from the Perron-Frobenius theorem [4]). The eigenvector corresponding to the unit eigenvalue, once normalized (so that it sums to 1), represents the steady state particle distribution of the system. $\mathbf{P}$ alone does not include the effects of particle injection or decay, so the steady state for our $\mathbf{P}$ is the uniform distribution on all nodes. Later we will modify the stochastic framework as well as the matrix analysis to take injection and decay into

[1]See *e.g.* [4]. Fix node j for consideration. Suppose node j initially holds n_0 particles and the nodes connected to j initially hold m_0 particles in total. After allowing one step of random particle movement, the number of particles at node j will be $n_0 - n_{\text{out}} + n_{\text{in}}$ where $n_{\text{out}} \sim \text{Binom}(n_0, p)$ and $n_{\text{in}} \sim \text{Binom}(m_0, p)$ are binomial random variables. The central limit theorem asserts that if $n_0 p$ and $m_0 p$ are sufficiently large, then the distributions will be approximately Gaussian; therefore the number of particles at node j is again approximately Gaussian and in particular its value will not deviate appreciably from the mean when the total number is large.

account.

Although the matrix $\mathbf{P}$ represents the transition probabilities of the model system, as a practical matter we numerically implement the Markov chain by direct simulation (as described in Methods). We do not actually create the matrix $\mathbf{P}$ explicitly for the stochastic simulations.

1.2 Information Capacity of a Gaussian Channel

In Results we will show numerically that the diffusion based communications channel is well approximated in the frequency domain by the classic additive white Gaussian noise (AWGN) channel [3]. Here we review the AWGN channel and show how to determine its capacity, following [3].

In the real-valued AWGN channel, the input signal x_t is transformed into a received signal y_t by the addition of Gaussian distributed white noise z_t. The corrupting noise is independent of the input signal, time, and history, and has equal power at all frequencies. We consider discrete time, with our focus on a single transmission at time t. Therefore, we refer to the input X, the noise Z, and the output $Y = X + Z$. We assume the mean noise $\mathbb{E}(Z) = 0$ and define the variance $\mathbb{V}(Z) = \mathbb{E}\left(Z - \mathbb{E}(Z)\right)^2 = N$ to be the noise variance or equivalently the noise power. We use $\mathbb{E}$ to denote the expectation of a random variable.

If the noise power N is zero, arbitrarily close inputs X_1, X_2, *et cetera* may be distinguished upon observing the corresponding outputs Y_1, Y_2, and the capacity is infinite. Intuitively, if the noise is nonzero, inputs must be spaced

out so that the noisy output can be distinguished with high probability. Similarly, if the inputs can be arbitrarily large, the capacity is infinite because spacing them out does not impose a limit on the variety of signals we can send reliably. In the real world, however, there is usually a constraint on the size of the inputs, such as the energy needed to represent them. With a constraint on input size and a nonzero noise power, only a finite set of inputs can be resolved with any accuracy, and the information capacity of a single transmission is finite.

Following Cover and Thomas' chapter *The Gaussian Channel* [3], we will calculate the information capacity of a single channel. A common constraint on input size is the power constraint, where the expected value of the square of the input may not exceed a limit P. For a set of inputs with mean 0 and equal probability of using any input, this corresponds to the variance of the input set. For the AWGN channel the noise is assumed to be Gaussian, typically because of the cumulative effect of many small random effects, with mean 0 and variance N. Whether this is true for the diffusion channel we will investigate empirically.

The information capacity C of a Gaussian channel is the maximum of the mutual information (I, defined below) between the input X and the noisy output Y over all possible input sets that satisfy the power constraint. Our use of X, Y, etc. may now be construed as connoting random variables. That is,

$$C = \max_{\mathbb{E}(X^2) \leq P} I(X;Y). \tag{1.2.1}$$

The mutual information I can be represented in terms of the entropy of X and Y. The entropy $h(X)$ of a random variable X is a measure of uncertainty of X. For a continuous random variable X with probability density function $f(x)$ the entropy is

$$h(X) = \mathbb{E}\left(\log(1/f(x))\right) = -\int_{f(x)>0} f(x)\log f(x)\,dx. \tag{1.2.2}$$

Given a joint density function $f(x,y)$ or a conditional density $f(x|y)$, we may also define the joint entropy

$$h(x,y) = -\int_{f(x,y)>0} f(x,y)\log(f(x,y))\,dx\,dy, \tag{1.2.3}$$

or the conditional entropy

$$h(x|y) = \int_y \left(\int_{f(x|y)>0} f(x|y)\log(f(x|y))\,dx\right) f(y)\,dy \tag{1.2.4}$$

The latter quantity is the (average) uncertainty of the input X upon observation of the output Y. The mutual information is the mean reduction in uncertainty about the input, given an observation of the output:

$$I(X;Y) = h(X) - h(X|Y) \tag{1.2.5}$$

Using the natural logarithm in the expression for the entropy yields mutual information measured in "nats". Conversion from "nats" to "bits" (binary digits) is accomplished by employing the logarithm in base two, or equivalently dividing measurements in nats by the natural log of two.

It is easily shown that the mutual information is symmetric in X and Y, and it is mathematically more convenient to work with the equivalent expression:

$$\begin{aligned} I(X;Y) &= h(Y) - h(Y|X) \\ &= h(Y) - h(X+Z|X) \\ &= h(Y) - h(Z|X) \\ &= h(Y) - h(Z) \end{aligned} \tag{1.2.6}$$

The last step follows because Z is independent of X.

To get to the entropy of Y we first look at its covariance, $\mathbb{E}Y^2$. Since X and Z are independent, and $\mathbb{E}Z = 0$, we may take the middle step of going from $2\mathbb{E}(XZ)$ to $2\mathbb{E}X\mathbb{E}Z$ to 0 in the expansion below. We should also assume that the maximum power P is used. Thus:

$$\mathbb{E}Y^2 = \mathbb{E}(X+Z)^2 = \mathbb{E}X^2 + \mathbb{E}Z^2 = P + N. \tag{1.2.7}$$

Theorem 9.6.5 in [3] notes that $h(Y) \leq \frac{1}{2}\log(2\pi e(P+N))$, with equality iff Y is normal with mean 0 and variance $P+N$. The entropy of Z is $\frac{1}{2}\log 2\pi eN$. We thereby arrive at the celebrated formula for the capacity of a Gaussian channel in terms of the signal-to-noise ratio P/N:

$$I(X;Y) \leq \frac{1}{2}\log(2\pi e(P+N)) - \frac{1}{2}\log(2\pi eN) = \frac{1}{2}\log\left(1 + \frac{P}{N}\right). \tag{1.2.8}$$

Equality, and thus the maximum, is reached when $Y \sim \mathcal{N}(0, N+P)$, and therefore when $X \sim \mathcal{N}(0, P)$. If we use logs in base 2, this is the information capacity C in bits per transmission.

A typical scenario involving additive models is when a channel carries information simultaneously on several nonoverlapping frequencies with independent noise. We therefore also want to consider the case where several

(k) AWGN channels are used at once, independently of each other except for a common power constraint. That is, each channel j ($j = 1, 2, \ldots k$) has a signal X_j and Gaussian noise Z_j drawn from $\mathcal{N}(0, N_j)$ that are added to make the output Y_j. The noise sources are again independent of each other, the X_js, and time and history. The common power constraint P remains, and the expected value of the sum of the squares of each channel's input (or alternatively the sum of the expected values of the squares of the inputs) must not exceed it. That is,

$$\mathbb{E} \sum_{j=1}^{k} X_j^2 \leq P. \tag{1.2.9}$$

How do we choose the input sets to maximize the information capacity?

As above, still following [3],

$$C = \max_{\mathbb{E}\sum X_j^2 \leq P} I(X_1, \ldots, X_k; Y_1, \ldots, Y_k). \tag{1.2.10}$$

Since each X_j is independent of Z_j, we can break down the right-hand side as before, so

$$\begin{aligned} & I(X_1, \ldots, X_k; Y_1, \ldots, Y_k) \\ = \; & h(Y_1, \ldots, Y_k) - h(Y_1, \ldots, Y_k | X_1, \ldots, X_k) \\ = \; & h(Y_1, \ldots, Y_k) - h(X_1 + Z_1, \ldots, X_k + Z_k | X_1, \ldots, X_k) \\ = \; & h(Y_1, \ldots, Y_k) - h(Z_1, \ldots, Z_k | X_1, \ldots, X_k) \\ = \; & h(Y_1, \ldots, Y_k) - h(Z_1, \ldots, Z_k). \end{aligned} \tag{1.2.11}$$

Because the noises are independent, their joint entropy is the sum of the marginal entropies. Without assuming the Y_js are independent, we can only

say that their sum gives an upper limit. Repeating the single-channel argument, we have

$$I \leq \sum_{j=1}^{k} h(Y_j) - h(Z_j) \leq \sum_{j=1}^{k} \frac{1}{2} \log\left(1 + \frac{P_j}{N_j}\right). \tag{1.2.12}$$

P_j now represents $\mathbb{E}X_j^2$, so then $\sum P_j = P$. Theorem 9.6.5 in [3] refers not just to one random variable but many, and we get equality for the right inequality in (1.2.12) iff

$$(X_1, X_2, \ldots, X_k) \sim \mathcal{N}\left(0, \begin{bmatrix} P_1 & 0 & \cdots & 0 \\ 0 & P_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & P_k \end{bmatrix}\right). \tag{1.2.13}$$

Equality for the left inequality in (1.2.12) follows upon assuming (1.2.13), which makes the X_js, and hence the Y_js, independent. In short, we will maximize capacity if each channel's inputs are normally distributed and independent. It remains to find the optimal values of the P_js.

We can use Lagrange multipliers to solve this problem. We maximize I subject to the constraint (1.2.9) by introducing an extra variable λ and finding extrema of

$$\sum \frac{1}{2} \log\left(1 + \frac{P_j}{N_j}\right) + \lambda((\sum P_j) - P), \tag{1.2.14}$$

which yields

$$\frac{1}{2 \ln 2} \frac{1}{P_j + N_j} + \lambda = 0. \tag{1.2.15}$$

Letting $\nu = \frac{-1}{\lambda 2 \ln 2}$, we see that

$$P_j + N_j = \nu, \tag{1.2.16}$$

or in other words, the power constraint plus noise variance for each channel should be equal. It may not be the case that a solution of this form exists, since $P_j \geq 0$; as shown in [3] one applies the Kuhn-Tucker conditions to show that using $P_j = (\nu - N_j)^+$ leads to the optimal distribution, where $(u)^+$ means $\max(u, 0)$. Consequently the capacity of the AWGN channel is

$$C = \sum_{j:N_j<\nu} \frac{1}{2} \log_2 \left(\frac{\nu}{N_j} \right) \tag{1.2.17}$$

where ν is chosen to be maximal given the constraint (1.2.9).

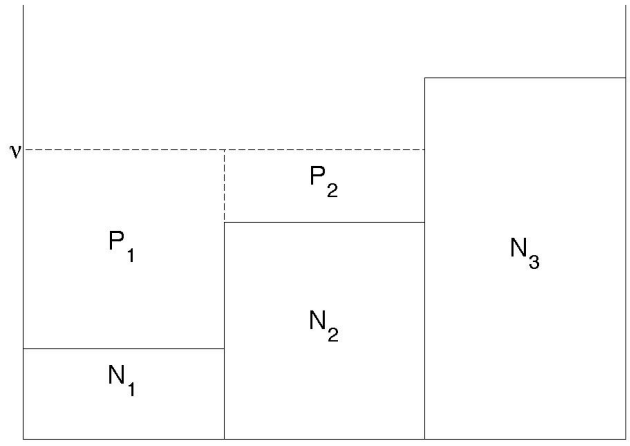

Figure 1: Waterfilling example. Given 3 AWGN channels with noise variance N_1, N_2, N_3 and constraint $\sum P_j \leq P$, the power is distributed to reach a common noise-plus-signal-power level ν in all channels that are used.

Finding ν can best be understood graphically (see Figure 1). This is often called a waterfilling model because of its similarity to pouring a fixed

quantity (P) of water into a tub with even-sized blocks of height N_j. The information capacity (in bits per transmission) of such a system is the sum of the capacities of all the channels using P_j and N_j.

We shall adapt both the single-channel information capacity formulas and the multi-channel waterfilling formula to our model. We explore and derive formulas for one interpretation, use of a disc in the Fourier transform's complex plane to encode a signal; we also touch briefly on another interpretation, in which we assume little faith in the detector's ability to synchronize with the source, restricting the signal to an amplitude only.

2 Methods

2.1 Simulation of a Diffusion Mediated Communications Channel

To approximate particle diffusion through a two-dimensional volume we introduce an 8×16 rectangular lattice. Particles perform a simulated discrete time, discrete space random walk, moving independently between neighboring grid points and having a constant probability per time step of removal. For definiteness, we number spatial coordinates as if indexing a transposed matrix (row and column, swapped) with $(1, 1)$ in the top left corner. In the simulation, each particle corresponds to a single (x, y) coordinate pair. Multiple particles may occupy the same grid point; we neglect possible crowding effects. Each grid point represents a subvolume $1/128$ the size of the total volume through which particles can diffuse. An injector adds some number of particles at every timestep to an injection point at coordinates $(1, 4)$, near the middle of the left wall, and a detector counts particles at $(16, 4)$ at the far side of the grid. See Figure 2 for an illustration of the grid with injection and detection points.

We think of the injector as a Poisson process with time-varying intensity

$$s'(t) = A + X \cos \omega t, \tag{2.1.1}$$

where we require $s'(t) \geq 0$, *i.e.* $|X| \leq A$. As a discrete-time approximation we inject a random number of particles given by a Poisson distribution with

mean $s'(t)$, *i.e.*

$$\Pr\{\#\text{ injected}(t) = k\} = \frac{s'(t)^k}{k!} e^{-s'(t)}. \tag{2.1.2}$$

We implicitly assume time units such that our time step $\Delta t = 1$.

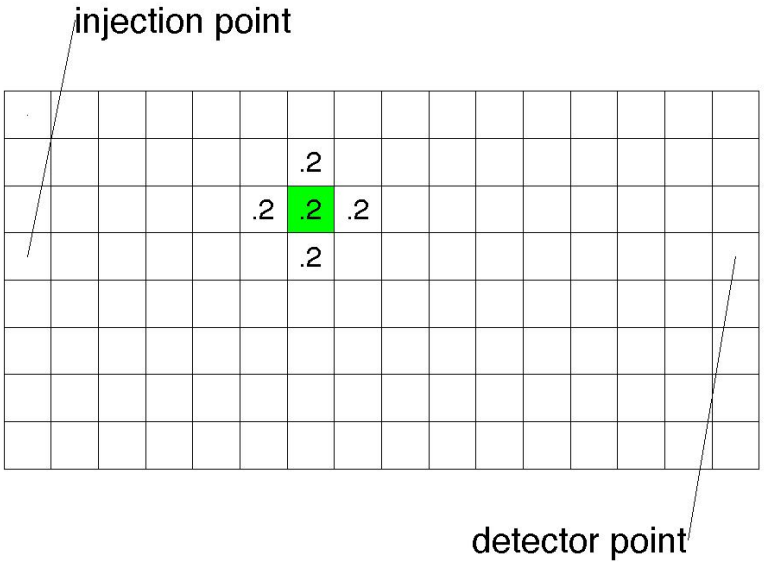

Figure 2: The particle grid. Particles perform a discrete-time discrete-space random walk on an 8×16 lattice, with injection at the left and detection at the right, and constant probability of removal per timestep. A particle in the green node can move to other nodes with specified probabilities. In this case the probability of moving to each nearest neighbor is $p = 0.2$.

The simulation works as follows. Particles are tracked in an unordered

list of (x, y) coordinates. We start with an empty grid, and at every timestep do the following:

1. Insert a number of particles at the injector location.
2. Move each particle either one step in a random direction, or zero steps.
3. Remove a randomly chosen set of particles from the list.
4. Record the number of particles at the detector location.
5. Plot the position of each particle (optional).

We accomplish particle movement by generating a list of random numbers from the uniform distribution on $[0, 1]$, one per particle, and adjusting the particles' coordinates based on those numbers. Let $p \leq 0.25$ represent the probability of a given particle moving to an adjacent grid point on a given timestep. The intervals $[0, p]$, $(p, 2p]$, $(2p, 3p]$, $(3p, 4p]$, $(4p, 1]$ correspond to right, left, down, up, and "stay". We maintain the integrity of the "walls" by keeping a particle in place if its random movement would place it off the grid. This implementation changes slightly in the generalization for coarse-graining (see Section 6.1 for code). Particle decay is done similarly: a new list of random numbers is generated, and particles corresponding to numbers less than the decay parameter α are removed.

The discrete-time Poisson process representing particle injection into the volume is accomplished by a MATLAB command such as:

```
s(t) = poissrnd(max([0,(1+amp*cos(w*t)')*a]))
```

Here `w` (for ω) is a column vector of several frequencies, `amp` is a row vector of fractional amplitudes corresponding to each frequency, `a` is the overall amplitude and the mean number of particles injected, and `poissrnd` is a Poisson random number generator which uses our cosine or sum of cosines as its parameter. That parameter is truncated at 0 in the code because only nonnegative arguments are accepted; however, it is undesirable for the injector to tend to dip below 0 for very long, because that introduces artifacts into the power spectrum, complicating the subsequent analysis. It is up to the user to ensure that `amp` is carefully chosen to keep the argument of `poissrnd` nonnegative; we do so for our simulations. Using a vector representation allows linear combination of input signals at multiple frequencies. Below, "injector amplitude" will refer to components of `amp` or `amp*a`, the amplitudes of the cosines, not to the mean injection rate `a`.

Simulations begin with an empty grid, but we do not want to consider the initial transient period, during which an average particle distribution is being built up, in the analysis. After recording the time series of the number of particles at the detector, we discard the first 1024 of 3072 readings and keep only the last 2048. This is adequate time to exclude the transient from our analysis (see Figure 3).

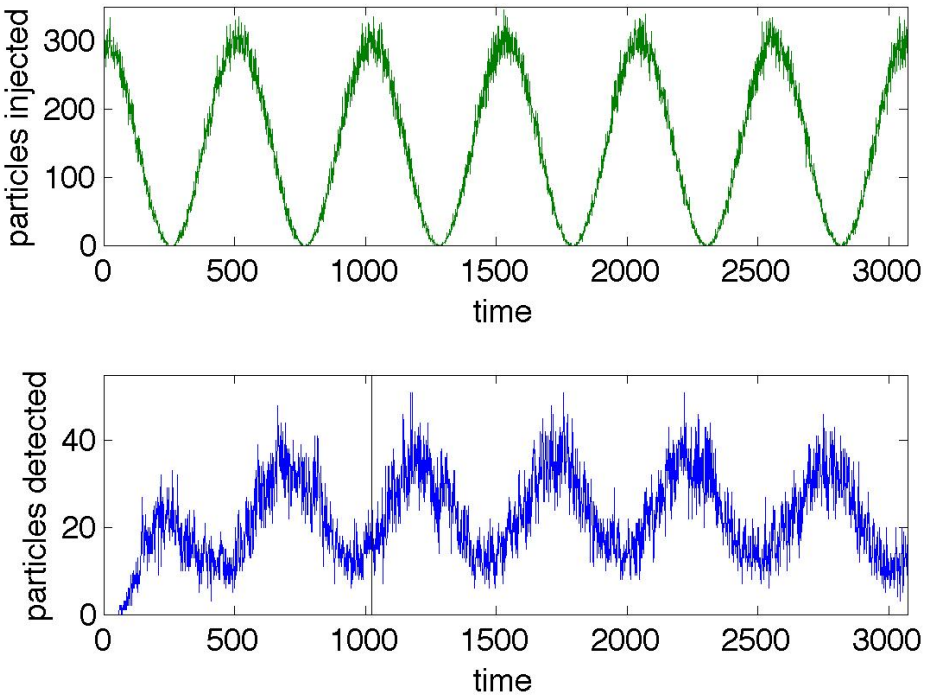

Figure 3: Example of diffusion channel input and output on a single trial. Input used a mean injection rate $A = 150$ and amplitude 150 at a single frequency $f = \frac{1}{512}$ (*i.e.* $\omega = 2\pi/512$). The number of particles injected (top trace, green) is a Poisson random variable with mean determined by the input rate. The number of particles at the counter (lower trace, blue) reflects the input frequency with a phase shift. Note decay of the transient over the first 1024 timesteps (separated by a vertical line), which are discarded before performing frequency analysis.

2.2 Spectral Analysis

In our simulations we set the mean injection rate to be $A = 150$, diffusion parameter to be $p = 0.2$, and decay rate to be $\alpha = 0.01$.[2] Because we keep only a time series of length 2048 for analysis, the argument ω is taken to be an integer multiple of $\frac{2\pi}{2048}$, giving a base frequency of $\frac{1}{2048}$ (units of inverse time). Frequency by itself will be denoted f, with $\omega = 2\pi f$. Because the simulation is time-intensive, in practice we only sample frequencies $\frac{4k}{2048}$ for $k = 1, 2, 3, 4$. This range includes frequencies that transmit information well and some that transmit almost none. Detailed analysis will be done on these frequencies, but later we will use deterministic simulations to extrapolate results at intermediate frequencies.

Our simulation generates a time series, $r(t)$, giving the number of particles at the counter location at each of $N = 2048$ time points t. We assume N is even. We use MATLAB's `fft` (fast Fourier transform) function to estimate the power spectrum. Given a column vector $\mathbf{u}$ representing samples of a time signal, it returns the discrete Fourier transform $\mathbf{v}$ as a column vector of the same length. Each entry is a complex number that, in polar coordinates, will represent the magnitude and phase of a component wave of a different frequency. The power spectrum is $\mathbf{v}$ multiplied elementwise by its conjugate, divided by the vector length (in MATLAB, `v.*conj(v)/N`).

We will find it more useful to use one of two more direct interpretations.

[2] These choices are chosen because they give numerically manageable quantities and simulation times. Varying them slightly should result in predictable changes; testing exactly how the results depend on choice of parameters is left for future work.

The vectors `2v/N` and `2*abs(v)/N` are respectively the complex representation and the amplitudes of the component waves.[3] At nonzero, non-Nyquist frequencies, for a real-valued time series, the power is equally split between frequencies f and $-f$. Each of those frequencies and its negative have conjugate values in both interpretations of the FT, so that the total value at a given positive frequency is the double. For both we must be consistent, considering only the positive frequencies and ignoring the negatives, after this doubling. Though we do not use the power spectrum, the amplitude version of the Fourier transform described above is proportional to the square root of it.

For a vector of length N, and $i = 1, 2, \ldots, \frac{N}{2}$, the ith entry $\mathbf{v}_i$ in the `fft` output corresponds to frequency $\frac{i-1}{N}$. The next entry, just after the halfway point, corresponds to the Nyquist frequency ($\frac{1}{2}$), and the rest are the complex conjugates in reverse order, referring to the negative frequencies (for example, the Nth is the conjugate of the 2nd, the $N-1$th that of the 3rd, etc.). Before running `fft`, we subtract the mean of $\mathbf{u}$ from $\mathbf{u}$, in order to suppress the spike at $f = 0$. MATLAB's function `fft` corresponds to the standard Fourier transform; for example, if we start with the vector $\mathbf{u} = a \cos \frac{2\pi t}{512}$, where $t = (1, 2, \ldots, 2048)$, we get a spike of amplitude a at

[3]This doubling is correct for the frequencies we attribute significance to. The exceptions are $f = 0$ and $f = \frac{1}{2}$ (the Nyquist frequency). Because we subtract the mean, the value at $f = 0$ is just 0; for our simulations we never approach the Nyquist frequency. Similarly, the frequencies above $\frac{1}{2}$ should not even be considered after the adjustment. A total doubling of the vector is for computational convenience when the interpretation is well-known.

precisely $\frac{1}{512}$, with zero power everywhere else (except the conjugate spike at $\frac{-1}{512}$, which we ignore since we've rolled it into the positive frequency).

For this project, although the cell's input and output are $s(t)$ and $r(t)$, they are not used directly in considering information capacity. A time series of 2048 real numbers corresponds reversibly to its discrete Fourier transform, which is complex but with redundancy so that it has 1023 independent complex numbers and 2 independent real numbers (at 0 and the Nyquist frequency), hence 2048 independent real values. Since most of those numbers will be close to 0 in the frequency domain of both input and output, deviating only by small amounts independent of the input, we may redefine the input and output X and Y of the model to be the complex amplitude of their Fourier transforms at whichever frequency we are considering. When multiple frequencies are used simultaneously, X_j and Y_j represent the complex amplitude of the corresponding input and output Fourier components for each. Thus, by defining the input and output by their Fourier transforms, we are not omitting any information, just transforming it into a more useful form.

Each combination of one or more input frequencies was repeated over 30 or 100 trials in order to estimate both the mean and variance of the resulting power spectrum of the resulting time series $r(t)$.

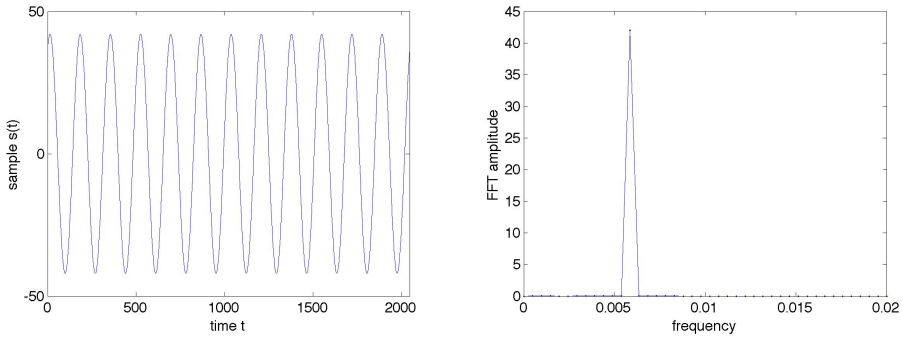

Figure 4: Example of a Fourier transform of $42\cos(2\pi\frac{3}{512}t - .5)$, $t = 0, 1, \ldots, 2047$. Left: time series. Right: amplitude of Fourier transform vs. frequency at low frequencies. There is a single spike of amplitude 42 at $f = \frac{3}{512}$ and zero values elsewhere.

2.3 Deterministic Simulations

The mean number of particles evolves according to a deterministic dynamics governed by the same transition matrix $\mathbf{P}$ as the stochastic simulation. We performed direct simulations of the corresponding deterministic system in order to compare results at the four frequencies chosen for stochastic simulations as well as for rapid generation of interpolating results at intermediate frequencies. We ran deterministic versions of the simulation for all the variations in injector frequencies and amplitudes, and obtained Fourier transforms matching the means of the corresponding stochastic simulations, as we will see in Results. Obviously, however, the deterministic simulations do not directly shed light on the nature of the noise.

We used the same approach with a modified transition matrix (as de-

scribed below) to study the coarse grainings in conjunction with the stochastic simulations.

2.4 Coarse Grained Channel Simulations

In addition to the basic simulation model, we implemented several coarse-grained versions as follows:

- *Chunky:* Aggregate blocks of 2×2 adjacent nodes to form a coarsened 4×8 grid.
- *Vertical:* Aggregate blocks of 4×1 adjacent nodes to form a coarsened 2×16 grid.
- *Horizontal:* Aggregate blocks of 1×4 adjacent nodes to form a coarsened 8×4 grid.
- *Random:* Aggregate randomly chosen groups of four (typically unconnected) nodes to form an irregular coarsened network.

In addition, we implemented an extremely coarse representation in which two blocks of 8×8 nodes were aggregated into a two-node network. One node contained the injection site and the other contained the counter, giving a two-state Markov communications model.

The original simulation exploited the regularity of the 8×16 grid to simplify the code. In order to handle arbitrary transition matrices we adapted the code to employ a pair of matrices to define the coarse graining, one containing a list of nodes adjacent to each node, the other containing a

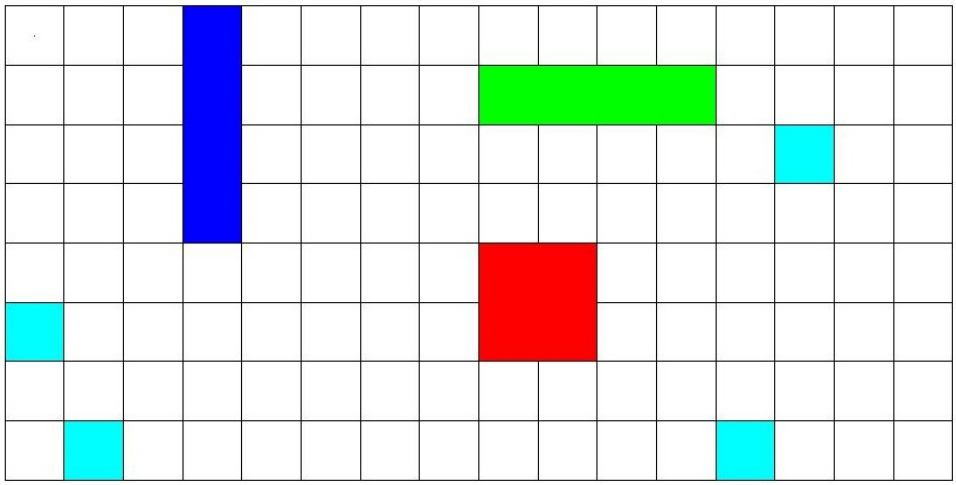

Figure 5: Graphical representation of the coarse grainings. Each of the red (chunky), blue (vertical), and green (horizontal) groups of 4 nodes stands for one coarse-grained node which is tiled 32 times. The cyan nodes form one of the random coarse-grained nodes.

3	5	11	14	13	26	2	10	10	22	14	26	6	13	21	15
11	15	13	10	24	21	8	17	22	20	7	7	13	1	29	18
25	12	32	24	21	19	31	9	26	14	2	16	16	28	20	32
4	2	12	29	22	29	23	12	4	31	14	22	30	31	27	32
6	21	9	16	9	6	19	24	26	17	31	15	25	18	8	30
28	3	3	27	12	23	29	32	20	5	25	19	11	27	8	6
10	17	18	3	4	24	11	17	30	9	20	5	4	8	5	1
15	28	1	30	2	7	16	7	18	1	19	27	28	23	23	25

Figure 6: Cluster numbers used for random coarse graining. The particle entry is in cluster 4, and the detector is in cluster 32.

probability distribution for movement to those nodes. See Section 6.1 (Code) for implementation details.

Each coarse-grained matrix model above requires a transition matrix. The original method can be expressed as a transition matrix, $\mathbf{P}$, which is square and has both dimensions equal to the number of grid points (128).

To define a "coarse-grained" representation of the original N-node system as a system with $M < N$ nodes, we introduce an $M \times N$ matrix $\mathbf{W}$. $\mathbf{W}_{ij}$ is 1 if original point $j \in \{1, \cdots, N\}$ is assigned to coarse-grained node $i \in \{1, \cdots, M\}$ and 0 otherwise. Let $\mathbf{n}_t$ be an N-dimensional vector representing the number of particles at each node in the fine-grained representation. Then $\tilde{\mathbf{n}}_t = \mathbf{W}\mathbf{n}_t$ represents the number of particles at each coarse-grained node. A single particle at node j would be represented by a vector $\mathbf{n}_t = [0, \cdots, 0, 1, 0, \cdots, 0] \in \mathbb{R}^N$ where the 1 is in the jth component. The coarse-

grained representation of this state would be $\mathbf{W}\mathbf{n}_t = [0, \cdots, 0, 1, 0, \cdots, 0] \in \mathbb{R}^M$ with the 1 in the location corresponding to whichever coarse-grained node includes fine-grained node j.

Coarse graining is an irreversible process, but may be approximately reversed by suitable choice of an $N \times M$ matrix $\mathbf{U}$. If $\mathbf{n} \in \mathbb{R}^N$ represents a fine-grained description of the system then $\tilde{\mathbf{n}} \in \mathbb{R}^M$, with $\tilde{\mathbf{n}} = \mathbf{W}\mathbf{n}$, represents the corresponding coarse grained description. We require that the pair $\mathbf{U}$ and $\mathbf{W}$ satisfy the following *first consistency criterion*: mapping a coarse grained state into a fine grained state and back again should leave the coarse grained description unchanged. That is, $\forall \tilde{\mathbf{n}} \in \mathbb{R}^M$,

$$\mathbf{W}\mathbf{U}\tilde{\mathbf{n}} = \tilde{\mathbf{n}}. \tag{2.4.1}$$

Equivalently,

$$\mathbf{W}\mathbf{U} = \mathbf{I}_M, \tag{2.4.2}$$

the $M \times M$ identity matrix.

In general, mapping from the fine-grained representation to the coarse-grained representation and back again will *not* leave the detailed description invariant. Nevertheless we may impose a *second consistency criterion.* Given a particular choice of the one-step $N \times N$ transition matrix $\mathbf{P}$, the unique steady-state distribution vector π satisfies $\mathbf{P}\pi = \pi$. We require that this fine-grained steady-state distribution remain preserved by mapping from $\mathbb{R}^N$ to $\mathbb{R}^M$ and back:

$$\mathbf{U}\mathbf{W}\pi = \pi \tag{2.4.3}$$

Together, conditions (2.4.2) and (2.4.3) may not fully constrain the choice of $\mathbf{U}$. For our spatially uniform grid the steady state is the uniform distribution for both the fine grained and each coarse grained representations, by construction. In this case we may choose $\mathbf{U}$ to be the *pseudoinverse* of $\mathbf{W}$,

$$\mathbf{W}^{+} = \mathbf{W}^{*}(\mathbf{W}\mathbf{W}^{*})^{-1} \tag{2.4.4}$$

where $\mathbf{W}^{*}$ denotes the adjoint matrix. The pseudoinverse is the unique solution to the matrix equation $\mathbf{WU} = \mathbf{I}$ that minimizes $\sum_{ij} \mathbf{U}_{ij}^{2}$. It also gives the least-squares solution to an underdetermined linear equation $\mathbf{Wx} = \mathbf{b}$ when $\mathbf{W}$ is rectangular as in our case.[4] Given the choice of $\mathbf{U} = \mathbf{W}^{+}$ satisfying both consistency criteria, it remains to choose an $M \times M$ matrix to represent a set of one-step transition probabilities on the coarse-grained space. Because the probability of transition between coarse-grained nodes depends, at least in the short run, on the fine-grained node of entry, the true dynamics as seen through the coarse-grained system is no longer Markovian. Nevertheless we will define a Markov process on the coarse-grained space to approximate the transitions present in the fine-grained description. The coarse-grained transition matrix $\widetilde{\mathbf{P}}$ must satisfy our *third consistency criterion*: that the steady states of the fine- and coarse-grained descriptions should be consistent. That is, a vector π stationary under $\mathbf{P}$ should be mapped to a coarse grained vector

[4] As a simple example consider a 4-to-2 coarse graining given by the matrix $\mathbf{W} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$, for which we have the pseudoinverse $\mathbf{W}^{+} = \begin{pmatrix} 1/2 & 0 \\ 1/2 & 0 \\ 0 & 1/2 \\ 0 & 1/2 \end{pmatrix}$.

$\widetilde{\pi} = \mathbf{W}\pi$ stationary under $\widetilde{\mathbf{P}}$. This criterion is satisfied by the coarse-grained transition matrix

$$\widetilde{\mathbf{P}}_{kl} = \sum_i \sum_j \mathbf{W}_{ki}\mathbf{P}_{ij}\mathbf{W}^+_{jl} \tag{2.4.5}$$

where $\widetilde{\mathbf{P}}_{kl}$ is the probability of a particle moving from l to k in the coarse-grained system. This formula can be thought of as un-coarse-graining the particle at l to several fine-grained positions j, finding the probabilities of moving to other fine-grained positions i, and coarse-graining those to see which wind up at coarse-grained node k. Equivalently, we define $\widetilde{\mathbf{P}} = \mathbf{WPW}^+$. Because (invoking equation (2.4.3)) $\mathbf{W}^+\mathbf{W}\pi = \pi$ we may write

$$\widetilde{\mathbf{P}}\widetilde{\pi} = (\mathbf{WPW}^+)(\mathbf{W}\pi) = \mathbf{WP}\pi = \mathbf{W}\pi = \widetilde{\pi} \tag{2.4.6}$$

which establishes that $\widetilde{\pi}$ is stationary under $\widetilde{\mathbf{P}}$ whenever π is stationary under $\mathbf{P}$ and $\widetilde{\pi} = \mathbf{W}\pi$.

For the purpose of computation we coded the transition probabilities by hand rather than explicitly constructing the matrices $\mathbf{W}$ and $\mathbf{P}$. The probability of moving to a certain coarse-grained node i is the average of the original probabilities of each point j within it (0 or p) moving to a point within i. For example, in the (2×2) coarse graining, the probability of moving to the node below is $(0 + 0 + p + p)/4 = p/2$.

The pseudoinverse approach we use to make transitions for the coarse-grainings is exact in the case of a uniform distribution of particles within each coarse-grained node. In our simulation, there will be on average more particles on the left than on the right of any coarse-grained node, and slightly more in the vertical center than at the top or bottom. The net effect of

coarse graining is to "graduate" particles prematurely toward the counter in the more horizontal coarse grainings, especially the one with only two nodes. Given a known fine-grained steady state, one could replace $\mathbf{W}^+$ with a different pseudoinverse that reflected it (as in consistency condition $\mathbf{UW}\pi = \pi$) for the nonuniform steady state π. Further discussion of how the pseudoinverse affects things is found later in this section. However, we leave detailed consideration of pseudoinverses corresponding to nonuniform steady states for future work.

The "received signal" in the coarse-grained system must be interpreted with care. In the fine-grained system, the received signal $R(t)$ is $r(t)$, the time series of particle counts at the counter node. In the coarse-grained system let $\tilde{r}(t)$ be the time series of particles at the coarse-grained node containing the counter. For a coarse graining in which the counter is aggregated into a coarse node as one of N/M fine nodes, there are (at least) three reasonable but nonequivalent ways to define the received signal $\tilde{R}(t)$ for the coarse-grained system.

1. $\tilde{R}(t) = \tilde{r}(t)$
2. $\tilde{R}(t) = \frac{M}{N}\tilde{r}(t)$
3. $\tilde{R}(t) \sim \text{Binom}\left(\tilde{r}(t), \frac{M}{N}\right)$

Alternative 1 amounts to expanding the counter's size by a factor of N/M to count all particles in the coarse-grained node. Inasmuch as we wish to compare the relationship between the input signal $s(t)$ and the received signal

$R(t)$ in the coarse- and fine-grained systems respectively, 1 would seem to give an unfair comparison. In the analysis, we will include both 2 and 3 as possible interpretations of the received signal. Alternative 2 amounts to applying the pseudoinverse matrix $\mathbf{W}^+$ consistent with the uniform steady state to produce an approximate fine-grained time series $\hat{R}(t) = \hat{r}(t)$ for the received signal. Alternative 3 assigns an equal chance of being counted to each particle in the coarse-grained node containing the counter. Hence the received signal at time t is taken to be a random variable drawn from a binomial distribution based on $\tilde{r}(t)$ attempts with probability M/N. While this approach may seem the most realistic interpretation, it also adds an additional source of noise due to the binomial counting process. In the following sections, Alternative 3 will be referred to as the "probabilistic" counting method.

2.5 Steady States under Coarse Graining

In the introduction, we saw that finding the steady state of a model is as easy as finding eigenpairs of the transition matrix. However, because we have particle injection and decay, we must adapt that approach. The sinusoidal components of the injection signal average out over observation times commensurate with their frequencies. Therefore to find the mean steady state distribution we consider only the constant mean injection rate. First we replace $\mathbf{P}$ by $\mathbf{Q} = (1 - \alpha)\mathbf{P}$ for ease of notation. If $\mathbf{n}_t$ is a vector representing a population of many particles, then

$$\mathbf{n}_{t+1} = \mathbf{Q}\mathbf{n}_t + \mathbf{s}, \tag{2.5.1}$$

where $\mathbf{s}$ is a vector with all zeroes except for the constant injection rate (150) in the row corresponding to the injection node. This corresponds to particle injection after movement and decay, instead of before, which uses $\mathbf{Qs}$ above. However, the two solutions only differ by $\mathbf{s}$, so we use this for simplicity. Then the steady state $\mathbf{n}_\infty$ is where $\mathbf{n}_t = \mathbf{n}_{t+1}$, so with a little rearranging,

$$(\mathbf{I} - \mathbf{Q})\mathbf{n}_\infty = \mathbf{s}, \tag{2.5.2}$$

$$\mathbf{n}_\infty = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{s}, \tag{2.5.3}$$

which has a unique solution because the eigenvalues of $\mathbf{Q}$ are all $\leq (1 - \alpha)$.

In the coarse-grained case, since $(1 - \alpha)\widetilde{\mathbf{P}} = (1 - \alpha)\mathbf{WPW}^+ = \mathbf{W}(1 - \alpha)\mathbf{PW}^+$, we can substitute $\widetilde{\mathbf{Q}}$ as the transition-plus-decay matrix. Then with $\tilde{\mathbf{s}} = \mathbf{Ws}$,

$$\tilde{\mathbf{n}}_{t+1} = \widetilde{\mathbf{Q}}\tilde{\mathbf{n}}_t + \tilde{\mathbf{s}}, \tag{2.5.4}$$

with similar steady-state solution $\tilde{\mathbf{n}}_\infty = (\mathbf{I} - \widetilde{\mathbf{Q}})^{-1}\tilde{\mathbf{s}}$. Note $\tilde{\mathbf{n}}_\infty$ refers only to the steady state of a coarse-grained system, not necessarily the coarse graining of $\mathbf{n}_\infty$ (because of our choice of $\mathbf{U}$ and particle injection and decay). How does this compare to the fine-grained model?

If we compare $\tilde{\mathbf{n}}_\infty = (\mathbf{I} - \mathbf{WQW}^+)^{-1}\mathbf{Ws}$, the coarse-grained steady state, to the coarse-grained version of the fine-grained steady state, $\mathbf{Wn}_\infty = \mathbf{W}(\mathbf{I} - \mathbf{Q})^{-1}\mathbf{s}$, we may note that $(\mathbf{I} - \mathbf{WQW}^+)^{-1} = (\mathbf{WIW}^+ - \mathbf{WQW}^+)^{-1} = (\mathbf{W}(\mathbf{I} - \mathbf{Q})\mathbf{W}^+)^{-1}$. If $\mathbf{W}$ were actually invertible (a trivial case, involving a renumbering of nodes at best), then $\mathbf{W}^+ = \mathbf{W}^{-1}$, and $(\mathbf{W}(\mathbf{I} - \mathbf{Q})\mathbf{W}^+)^{-1} = \mathbf{W}(\mathbf{I} - \mathbf{Q})^{-1}\mathbf{W}^{-1}$, so that $\tilde{\mathbf{n}}_\infty = (\mathbf{I} - \mathbf{WQW}^+)^{-1}\mathbf{Ws} = \mathbf{W}(\mathbf{I} - \mathbf{Q})^{-1}\mathbf{s} = \mathbf{Wn}_\infty$, and the two would be equal. The degree to which $(\mathbf{W}(\mathbf{I} - \mathbf{Q})\mathbf{W}^+)^{-1}$ differs

from $\mathbf{W}(\mathbf{I}-\mathbf{Q})^{-1}\mathbf{W}^{+}$ makes up most of the difference when $\mathbf{W}$ is nontrivial, depending on $\mathbf{W}$ and $\mathbf{Q}$.

We find that the steady states of the coarse grainings generally spread things out more evenly (compared to the coarse grainings of the original steady state). This effect reflects graduation of particles too quickly toward the counter as described above. See Figure 8.

Figure 7 shows a visualization of the original model's steady state for reference; in Figure 8 we see them for the more orderly coarse grainings, along with the original steady state coarse-grained for comparison. All use the calculation matching injection before movement. Note the flattening effect of coarse graining first. The only exception is the horizontal one, which is close to equal anyway. The random coarse graining (not shown) has a very even steady-state distribution, with only the injector node being significantly higher, and the populations falling off slightly by minimum distance to it; but it is hard to visualize. Coarse graining the original steady state to match makes it very erratic. The difference for the 2-node model (not shown) is similar to the rest.

The average behavior of the system calculated from the steady state of the deterministic model provides a useful check on the time-varying stochastic simulations. The mean number of particles at the detector at steady state should be close to the stochastic system's mean; if that is true over the final 2048 timesteps, as it was for our models, the first 1024 steps were indeed sufficient to reach equilibrium.

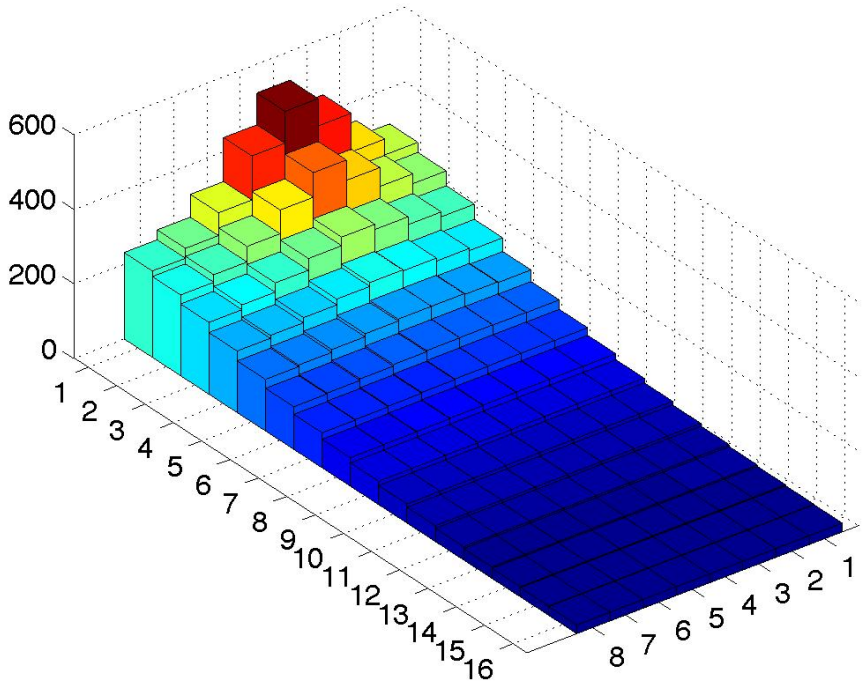

Figure 7: Steady state concentration of the original model as a function of position, as viewed from the bottom right corner. The injector at (1,4) is the peak; the counter is at (16,4). Heights represent number of particles. Color corresponds to height (population) for ease of visualization.

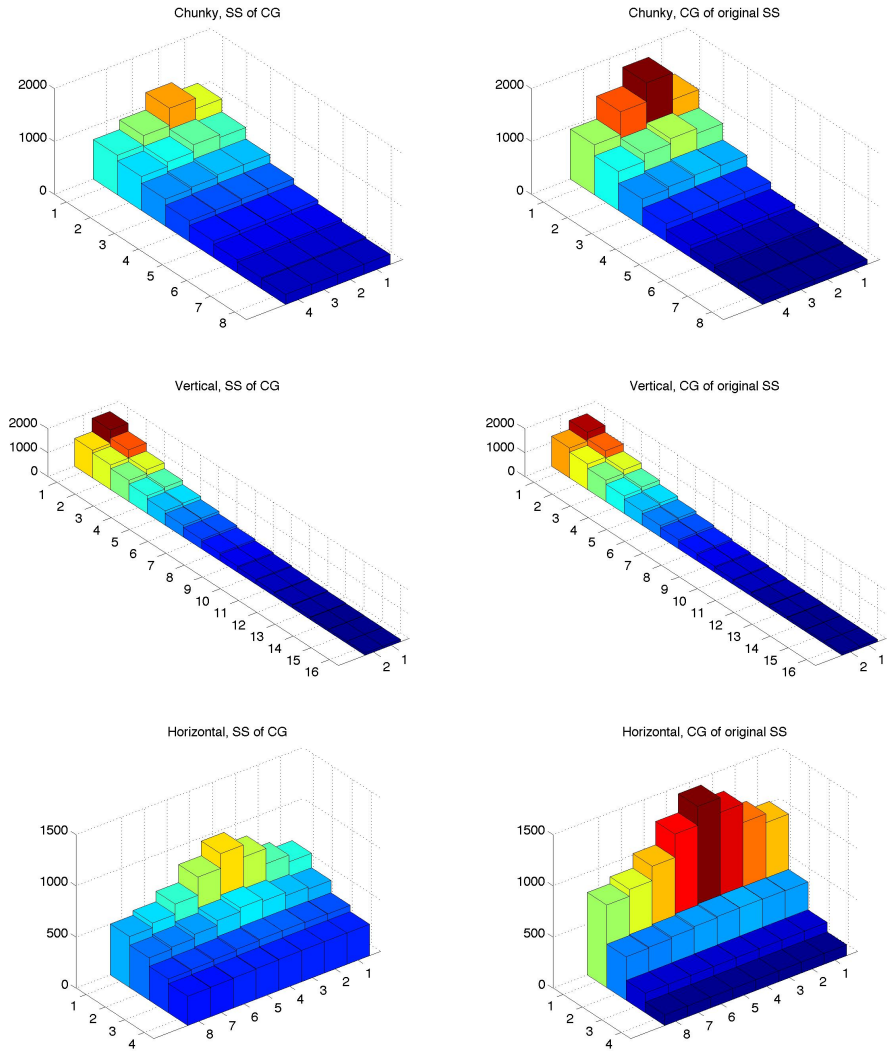

Figure 8: Steady states of the coarse grainings. Left: their steady states ($\tilde{\mathbf{n}}_\infty$). Right: coarse grainings of the original model's steady state ($\mathbf{W}\mathbf{n}_\infty$). The format is the same as Figure 7, and color mapping is consistent in each pair.

2.6 Information Capacity

Finally, for both the original model and the coarse-grained versions we wish to determine the information capacity of the cell viewed as a communications channel.

We require some adaptation of the Gaussian channel model's formulas to do so. As we will see in Results, there will be attenuation of the original signal obeying a directly proportional relationship between the sent and received amplitudes; the received signal will be at the same frequency; there will be a somewhat predictable amount of phase shift; and we will define noise for our model, which will be approximately Gaussian and independent in $\mathbb{C}$ though not in amplitude-and-angle representation. We must reconcile the expectation of a power constraint, which might allow for a direct use of the Gaussian channel formulas, with an amplitude constraint for single-channel information capacity and waterfilling. For waterfilling, we will see whether signals sent simultaneously on different frequencies (channels) may be treated as independent channels or if they influence each other.

In the simulations, we used an injector function with mean 150 and amplitude of 0, 50, 100, or 150 at various frequencies (or combinations of them). To obtain a spectrum for analysis, we take the `fft` of the detector's time series as described above; the relationship of the amplitude and phase of the FT entries to the input's own amplitudes and phases at the injected frequencies gives us a signal-to-noise ratio. As we will see, amplitude and phase at the driven frequencies is influenced, but at other frequencies is uncorrelated.

3 Results

3.1 Main Model Results and Comparison to AWGN Channel

First we examine the Fourier transform. This is useful in amplitude-only form, such as for plotting against frequency to visualize the spectrum; it is also useful in the complex plane, such as for comparing results from different trials or parameters at particular frequencies.

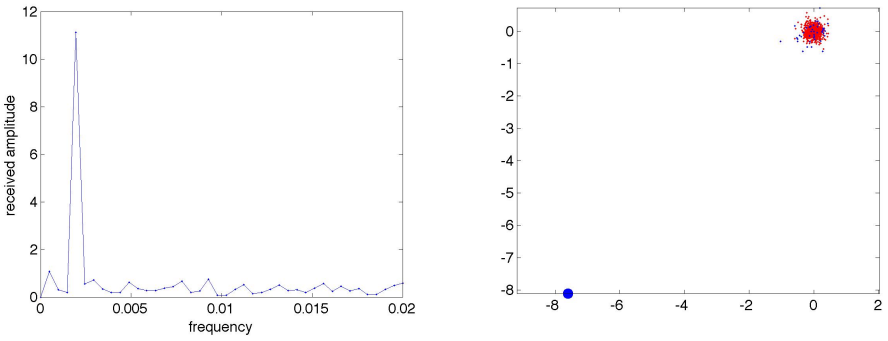

Figure 9: Fourier transform example of received signal, using full-amplitude injector (150) at $f = \frac{1}{512}$ (about .00195 on the horizontal axis, left). Left: amplitude vs. frequency. Right: FT in the complex plane, with blue points corresponding to those on the left and red points representing the rest of the spectrum (the higher frequencies not seen at left). Note the lone point near (-8,-8) corresponding to the peak in the left plot (enlarged for visibility). The time series for this data is found in Figure 3.

Figure 9 shows the low-frequency end of a spectrum of the received signal,

as an illustrative example. The high point represents the amplitude of the received signal at the same frequency the injector was using, $\frac{1}{512}$. At all other frequencies, the amplitude is quite low, indicating the noisy particle movement of the diffusion process. The bulk of the power is contained in the original frequency, consistent with a linear signal transmission model.

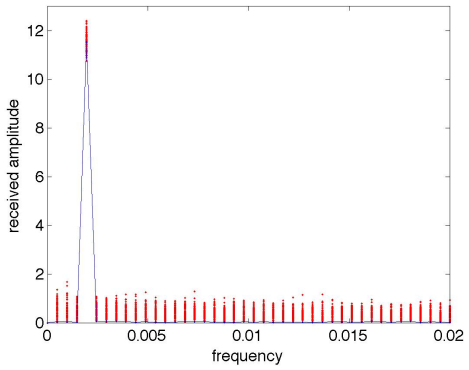

Figure 10: Same as Figure 9, but showing the amplitude of the mean of 100 runs. The red dots indicate individual results.

The single-frequency simulations ran with cosine amplitudes of 0, 50, 100, and 150 at the first four multiples of the base frequency $\frac{4}{2048}$. The received signal amplitude scaled linearly with the injector amplitude, and the spikes decreased as the injector frequency increased in a way we shall examine shortly.

Figure 11 shows the first property, using the first four multiples of the base frequency at four injector amplitudes. Here we plot the means of the amplitudes from 100 runs. Error bars reflect the standard deviations. The relationship is approximately linear. In Figure 12 we see the same data,

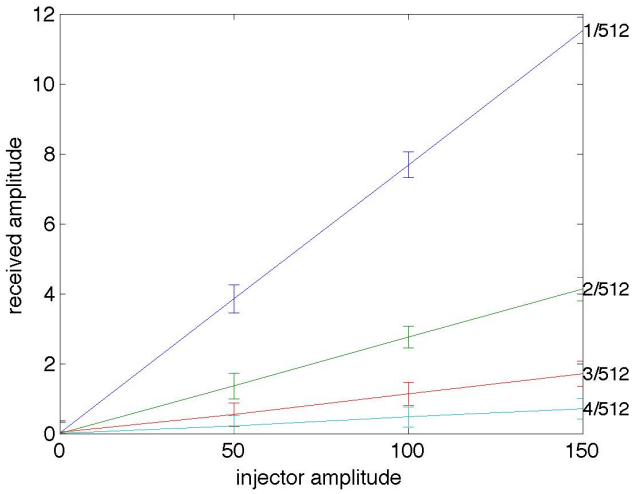

Figure 11: Amplitudes of mean peaks (100 runs each), plotted vs. injector amplitude. Each line represents one frequency. Error bars indicate the standard deviation.

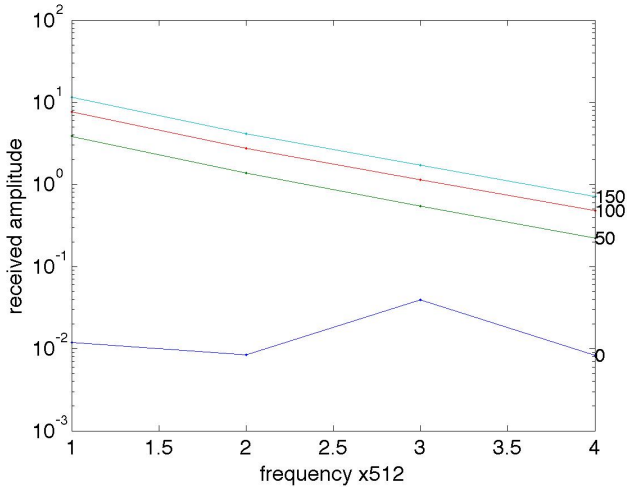

Figure 12: Amplitudes of mean peaks (100 runs each), plotted vs. frequency. Each line represents one amplitude.

rearranged to show how the received signal amplitude changes with injector frequency. The vertical axis is log-scaled to illustrate the trend. At nonzero injector amplitudes, the peaks fall off nearly exponentially. At zero injection amplitude, that is, a constant injection rate, the mean is very close to 0 everywhere, so the jagged line is only a result of the scale.

The waterfilling construction for the channel capacity requires that linear superposition of input frequencies be transmitted as linear superposition in the output. To test this, we ran simulations which superimposed several cosines of different frequencies in the injector function. As seen in Figure 13, the transmission preserves linear superposition of the multiple frequency components. Closer examination shows that the peaks are additive under superposition, and the background amplitudes at other frequencies are about 0 in mean. For individual runs, each frequency component varies similarly to the single-injector-frequency runs.

As described in the introduction, the AWGN model assumes each frequency component is perturbed by additive Gaussian noise with mean 0, independent of other frequencies and the signal itself. The background noise found in all frequencies is the same whether or not a signal is present. To what extent can we represent the perturbation at the encoded frequencies as additive Gaussian noise?

To address this question, it is most helpful to evaluate the distributions of the encoded frequencies' Fourier transform entries in the complex plane. Figure 14 shows all 100 trials for each injector amplitude at each frequency, with their own means and the results of the deterministic simulations marked.

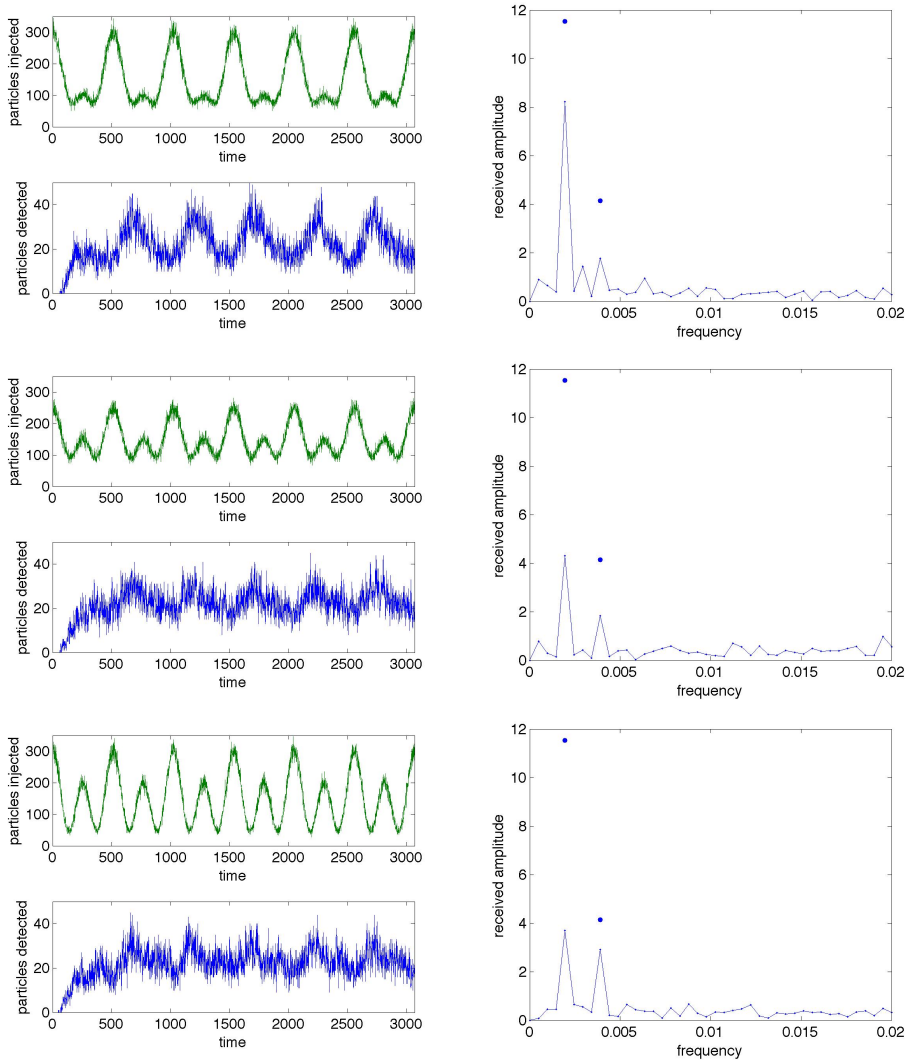

Figure 13: Combining different frequencies. Left: sample runs of injected signal (green) and received signal (blue). Right: corresponding FT amplitude spectra of output for those samples. The extra blue points represent the means from the full-strength injectors for comparison. Top to bottom: $(\frac{2}{3}, \frac{1}{3})$, $(\frac{1}{3}, \frac{1}{3})$, $(\frac{1}{3}, \frac{2}{3})$ times the base amplitude of the first two frequencies $(\frac{1}{512}, \frac{2}{512})$.

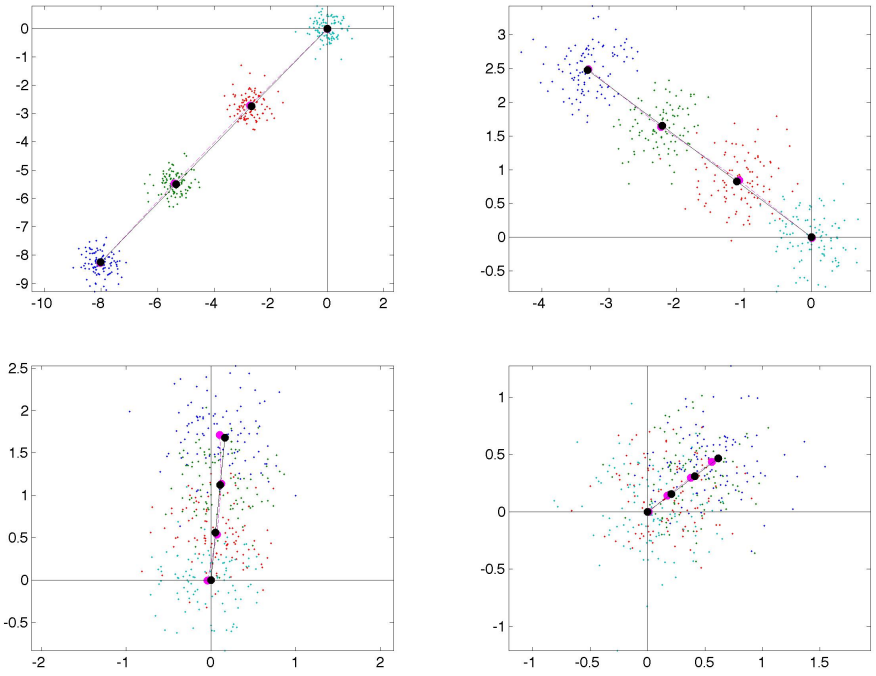

Figure 14: Diffusion noise appears effectively Gaussian and additive. Complex Fourier representation of channel output at the input frequency, showing 100 runs of each type of injector with different amplitudes of injection rate modulation (blue: amplitude 150; green: 100, red: 50; cyan: 0). Large magenta points show the sample mean of each cluster; black points show the mean expected based on the deterministic simulation. Top left: $f = \frac{1}{512}$. Top right: $f = \frac{2}{512}$. Bottom left: $f = \frac{3}{512}$. Bottom right: $f = \frac{4}{512}$.

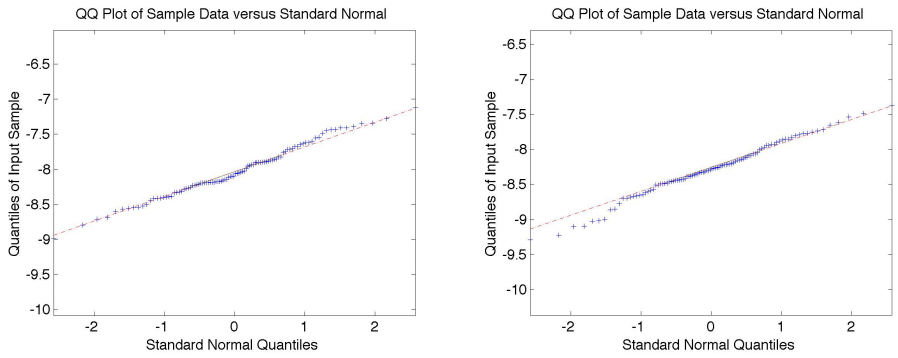

Figure 15: Sample `qqplot`s (quantile-quantile plot) of the real and imaginary parts of the FT for one set of 100 runs. We can claim that the distributions are approximately Gaussian because the plots are close to the diagonal lines.

The means match the deterministic version well. In both cases, not only are the amplitudes spread out in a linear fashion, but the phase (angle) is the same, independent of the amplitude but characteristic only of the frequency.

Concerning the noise, the actual distributions appear to be normal and isotropic. Figure 15 shows a sample of a pair of `qqplot`s, corresponding to the real and imaginary parts of the transforms of the 100 trials using an injector at $f = \frac{1}{512}$ and `amp` $= 1$. If the plots are close to the center line, the distribution is approximately normal, and we see that that is the case.

Using methods described below, we find the variances (in one dimension) of these distributions, as well as of the many unseen ones from unused frequencies (all of which have complex scatter plots resembling the 0-amplitude distributions above). Figure 16 contains two visualizations of these variances. On the left, we see the variances of the 16 distributions from Figure 14, plotted against injector amplitude. Note that the variance shows no

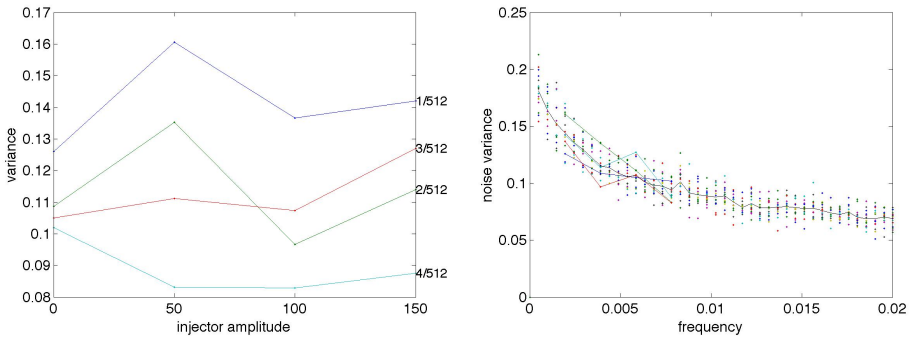

Figure 16: Variance of FT for each setup (100 runs each). Left: plotted vs. amplitude, one line per frequency. Right: plotted vs. driving frequency, including variances at other frequencies. The 16 points from the left are connected by lines representing injection amplitude and are quite haphazard. The means of these variances are represented by the long line. Both plots indicate that noise variance depends on injection frequency and not amplitude.

correlation with the amplitude; *i.e.*, for each frequency (channel), the noise is independent of the signal. However, the variance does decrease slightly as the frequency increases, and this relationship will make different N_is in the multichannel model. On the right, we have all computed variances vs. frequency. The usual 16 points are highlighted, connected by amplitude to show the independence. The means are connected as well on the longer line.

We see that the variance is a little higher at low frequencies. While it bends the truth to call it white noise, it will not matter for our purposes, especially when we use the waterfilling method. How the noise fits in will be seen when we work on information capacity.

3.2 Capacity of the Diffusion Channel at a Single Sending Frequency

Before estimating the capacity of the diffusion-based communications channel for general inputs combining multiple frequencies, we require an analytic approximation for the capacity of a single channel. The numerical results in section 3.1 show that for a single frequency ω_k a (complex) Fourier input $x_k \in \mathbb{C}$ is converted to an output with Fourier component $y_k \in \mathbb{C}$ at the same frequency given by

$$y_k = \beta_k x_k + z_k \tag{3.2.1}$$

where $\beta_k \in \mathbb{C}$ captures the attenuation and phase lag of the transmitted signal, and $z_k \in \mathbb{C}$ is a standard complex Gaussian random variable, *i.e.*, its mean is zero and its density in the complex plane is a circularly symmetric

bivariate Gaussian with variance N_k along both the real and imaginary axes. Moreover, the channel satisfies linear superposition of multiple frequencies. The noise variance ($2N_k$), the attenuation $|\beta_k|$, and lag $\mathrm{Arg}[\beta_k]$ all vary with ω_k.

The complicating factor for this channel is that the natural constraint on the ensemble of input signals $x \in \mathbb{C}$ is not a mean power constraint but rather a constraint on the support of the probability distribution $f_x(x)$. For a mean sending rate A, we require $|x| < A$ so that the injected signal $A + \Re[xe^{i\omega t}] \geq 0$ at all times. As described above, the capacity at this frequency alone is given by the supremum of $I(x;y)$ over all distributions $f_x(x)$,

$$I(x;y) = h(y) - h(y|x). \tag{3.2.2}$$

Assuming $y = \beta x + z$, then as in the AWGN case the conditional entropy, $h(y|x) = h(y - \beta x|x) = h(z|x) = h(z)$, is a constant independent of the distribution $f_x(x)$.

To find the largest value of I therefore requires maximizing

$$h(y) = -\int_{y\in\mathbb{C}} f_y(y) \log(f_y(y))\, dy \tag{3.2.3}$$

with

$$f_y(y) = \int_{|x|<A} g_N(|y - \alpha x|) f_x(x)\, dx \tag{3.2.4}$$

where g_N is the bivariate Gaussian with variance $2N$. The supremum is taken over all distributions $f_x(x)$. Being unable to find a closed-form solution of this nonlinear optimization problem, we explored several approximations.

First we considered the analogous optimization problem for real-valued x and y with the support of x limited to the compact interval $0 \leq x \leq a$. In this case, the optimization problem is to maximize

$$h(y) = -\int_{y\in\mathbb{R}} f_y(y)\log(f_y(y))\,dy \tag{3.2.5}$$

with

$$f_y(y) = \int_{|x|\in[0,a]} g(y-x)f_x(x)\,dx \tag{3.2.6}$$

subject to the constraints

$$f_x(x) \geq 0, \qquad \int_{x\in[0,a]} f_x(x)\,dx = 1. \tag{3.2.7}$$

For this problem we arbitrarily set the Gaussian to be $g(u) = \exp[-u^2]/\sqrt{\pi}$, and explore the change in h as a function of the scale a. If we choose the uniform distribution $f_x(x) = 1/a$, the resulting distribution for y ranges between a Gaussian when a is small and a plateau with rounded shoulders for large a. The entropy of f_y will be maximized when this posterior distribution is as "close" to a uniform distribution as possible. Rather than solve the full nonlinear optimization problem for f_x we try to gain some intuition by examining convex combinations of the form

$$f_x(x) = (1-\gamma)\frac{1}{a} + \frac{\gamma}{2}(\delta(x) + \delta(x-a)) \tag{3.2.8}$$

for $0 \leq \gamma \leq 1$. By shifting weight to the ends of the interval we can increase the entropy of f_y. Figures 17 and 18 show the entropy $h(y)$ for different values of the convex combination parameter γ. Note that the entropy of the y-distribution generated by optimal convex combination of the two basis functions differs from the entropy of the y-distribution generated by the

uniform Ansatz only slightly once $a > 10$. Since the uniform distribution will be easiest to work with later on, and our significant results satisfy this condition, we use the approximation of a uniformly-distributed input set.

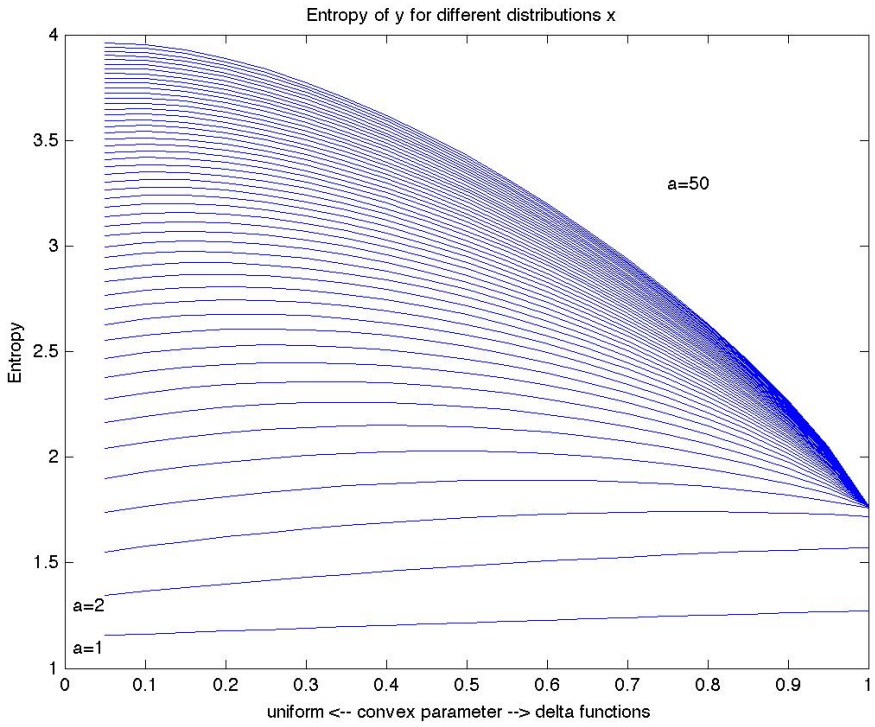

Figure 17: Entropy from 1D uniform distribution + delta functions. Entropy (in nats) was calculated numerically for a family of distribution functions on the interval $[0, a]$ of the form $f_x(x) = (1-\gamma)\frac{1}{a} + \frac{\gamma}{2}(\delta(x) + \delta(x-a))$ for $0 \leq \gamma \leq 1$. When $\gamma = 0$ we have the uniform distribution, $x \sim \text{Unif}[0, a]$. When $\gamma = 1$ we have a pair of delta functions at $x = 0, a$.

As noted, we examined two ways of looking at the channel. Of primary interest is the use of a disc in the Fourier transform's complex plane constrained

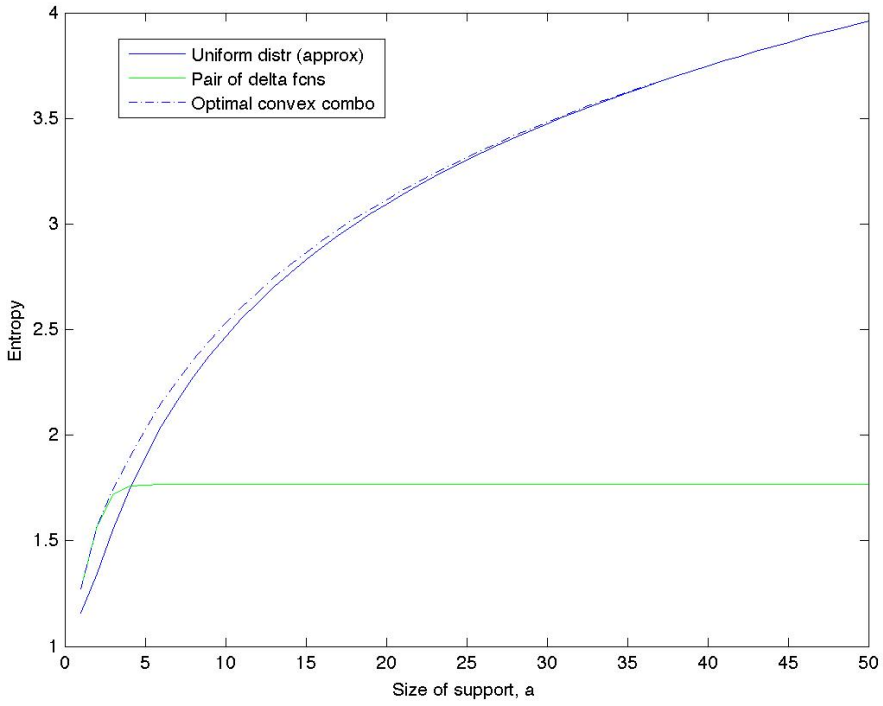

Figure 18: Entropy $h(y)$ in nats for $f_y = f_x * g$ for f_x taken to be the uniform distribution on $[0, a]$ (solid blue line), a pair of delta functions at $x = 0, x = a$ (solid green line), and the optimal convex combination of the two (dotted line), versus the size of the support region, a. Once the support parameter a is above 10, the uniform distribution and the optimal convex combination of uniform plus delta functions for f_x give nearly the same entropy $h(y)$. Here the Gaussian $g(x)$ has width $\sqrt{\mathbb{E}(x^2)} = 1/\sqrt{2}$. The "signal-to-noise ratio" analogous to $a/\sqrt{N}$ (introduced later) corresponds to $a/\sqrt{2}$ (for this temporary a).

by amplitude for containing the input ensemble; we shall also touch on the one-dimensional "amplitude-only" version of this, ignoring phase shifts.

Returning to the complex case, we will approximate the convolution (3.2.4) as follows. We assume that as in the case of the mean power constraint waterfilling construction, most of the capacity in the multifrequency channel will be carried in frequencies at which the noise is relatively small compared to the signal. In the diffusion channel case, let

$$a = |\beta|A \tag{3.2.9}$$

represent the amplitude of the maximum signal after attenuation by the channel, ignoring noise (or considered as the mean). Assuming $a \gg \sqrt{N}$, the integrand in (3.2.4) will be close to the uniform distribution on the disc $|y| < a$, except within approximately the range $|y| \approx (a \pm \sqrt{N})$. In this region the profile of f_y as a function of $r = |y|$ will be close to the convolution of a Gaussian with a step function, involving the standard error function erf:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt \tag{3.2.10}$$

with a correction of order a^{-1} due to the curvature of the boundary:

$$f_y(r) = \frac{1}{\pi a^2} \left(\frac{1 - \mathrm{erf}((r-a)/\sqrt{N})}{2} + O(a^{-1}) \right) \tag{3.2.11}$$

Given this approximation for f_y we investigated the entropy (3.2.3) via straightforward numerical integration. For $N = 0$ we know the entropy: the uniform distribution on the disc of radius a has entropy $h_0 = \log_2(\pi a^2)$. An empirical fit to first order in a and N shows that the correction due to

small amounts of noise takes the form

$$h_y(a,N) \approx \log_2(\pi a^2) + K_1 \frac{\sqrt{N}}{a} + O\left(\frac{\sqrt{N}}{a}\right)^2. \tag{3.2.12}$$

The constant K_1 obtained empirically is $K_1 \approx 1.8873$ bits (or $K_1 \approx 1.3082$ nats if we calculate entropy using the natural logarithm). The mutual information then takes the form

$$\begin{aligned} I(x;y) &= h(y) - h(y|x) \\ &\approx \log_2(\pi a^2) + K_1(\sqrt{N}/a) + O\left(\sqrt{N}/a\right)^2 - \log_2(2\pi e N) \\ &= 2\log_2(a/\sqrt{N}) + K_1(\sqrt{N}/a) - \log_2(2e) + O(\sqrt{N}/a)^2 \\ &= 2\log_2\left(\frac{a}{\sqrt{N}} + K_1\frac{\ln 2}{2}\right) - \log_2(2e) + O(\sqrt{N}/a)^2 \\ &= 2\log_2\left(\frac{a}{\sqrt{N}} + K\right) - \log_2(2e) + O(\sqrt{N}/a)^2 \end{aligned} \tag{3.2.13}$$

where we obtain the last equality from the Taylor expansion

$$\log_2(x+\epsilon) = \log_2(x) + \epsilon/(x\ln(2)) + O(\epsilon^2). \tag{3.2.14}$$

The unitless constant K in (3.2.13) is approximately $K \approx 0.6541$. Note that by definition the mutual information I cannot be negative. The asymptotic expression (3.2.13) requires $a/\sqrt{N} > \sqrt{2e} - K \approx 1.68$. Because this is an asymptotic expression valid in the limit of small values of $\sqrt{N}/a$ (or large values of $a/\sqrt{N}$) this condition is always satisfied for the frequencies of interest.

Expression 3.2.12 includes several assumptions:

1. The noise is small in the sense that $\sqrt{N}/a \ll 1$.

2. For given values of a and N, the maximal entropy of the distribution $f_y^*(y)$ is close to the entropy of the distribution $f_y(y)$ resulting from convolution of a Gaussian of width $\sqrt{N}$ with the uniform distribution on the disc of radius a, at least for the relevant frequencies.

3. The expression obtained for the entropy is a good approximation of the true entropy for $f_y(y)$.

3.3 Adapting Information Capacity and Waterfilling

As suggested by the results in the previous section we will assume that for a combination of inputs with complex Fourier components at several frequencies $\{\omega_i\}$, the channel affords linear superposition with independent noise added to each frequency component:

$$s(t) = \Re[A + \sum_i x_i e^{i\omega_i t}] \tag{3.3.1}$$

$$r(t) = \Re[\bar{r} + \sum_k y_i e^{i\omega_i t}] + O(1) \tag{3.3.2}$$

$$y_k = \beta_i x_i + z_i \tag{3.3.3}$$

where $s(t)$ is the input sending rate, $r(t)$ is the number of particles counted at the receiver (the term $O(1)$ denotes the discrepancy between having counts restricted to integers and having real-valued approximate values), and $\bar{r}$ is the mean number counted at steady state. Note that $O(1)$ is *small* compared to the typical size of $r(t)$ for the parameter ranges we consider.

The natural constraint on the system is

$$\sum |A_i| \leq A \tag{3.3.4}$$

where $|x_i| \leq A_i$ for each frequency, the A_is being those individual frequencies' constraints which we seek to solve to maximize capacity. The constraints guarantee a nonnegative sending rate $s(t)$. Given this constraint and the (approximate, empirically derived) form of the single frequency capacity (3.2.13) we will obtain a waterfilling formula analogous to (1.2.17).

From (3.2.13) the multi-channel mutual information is approximately

$$I(x,y) = \sum_i \left(2\log_2\left(a_i/\sqrt{N_i} + K\right) - \log_2(2e) + O(\sqrt{N_i}/a_i)^2\right) \tag{3.3.5}$$

with $a_i = |\beta_i A_i|$. This approximation is valid for all i for which $\sqrt{N_i}$ is sufficiently smaller than a_i. We wish to maximize I given the constraint (3.3.4) by maximizing

$$I(x,y) + \lambda(A - \sum_i |A_i|). \tag{3.3.6}$$

The solution leads to a waterfilling distribution of *amplitudes* rather than *power*:

$$|A_i| = \left(\nu - K\frac{\sqrt{N_i}}{|\beta_i|}\right)^+ \tag{3.3.7}$$

where the constant ν is chosen to be maximal given the constraint (3.3.4), and $(u)^+ = \max(0,u)$ as before. For the frequencies at which the signal-to-noise ratio is too low, *i.e.*, $\nu < K\sqrt{N_i}/|\beta_i|$, the asymptotic expression (3.2.13) is no longer valid. For these frequencies we set the amplitude of the

sending signal to $|A_i| = 0$ and also set to zero the corresponding terms in calculating the channel capacity:

$$\begin{aligned} C &= \sum_{i:K\sqrt{N_i}/|\beta_i|<\nu} 2\log_2\left(\frac{|\beta_i A_i|}{\sqrt{N_i}} + K\right) - \log_2(2e) \\ &= \sum_{i:K\sqrt{N_i}/|\beta_i|<\nu} 2\log_2\left(\frac{|\beta_i|}{\sqrt{N_i}}\nu\right) - \log_2(2e) \end{aligned} \tag{3.3.8}$$

for the constant K.

This means that finding the optimal distribution is equivalent to waterfilling! Instead of the solutions having equal levels of power plus noise variance, they have equal levels of essentially the sums of their square roots (in the sense that the power P, or constraint on input variance, from the AWGN channel is proportional to the square of our amplitude constraint).

Two problems arise. If the optimal distribution uses 0 amplitude in some of the noisier frequencies (channels), this method will not find that distribution. In addition, since the information capacity function does not intercept the origin, no matter how the formula is rearranged there is always a nonzero constant term that does not drop out in the derivative. (It is true that we never seek to use an A_i that makes $a_i/\sqrt{N_i}$ so low that the approximation is invalid, but the optimization method denotes unused channels that way.) Therefore, for example, a distribution using 3 frequencies may be better for us than an "optimal" distribution using 4 if the constant is close enough to the gain. The answer to both problems is to repeat the process several times, each time leaving out the (next) noisiest frequency; the solutions provide a basis for comparison. This solves the first problem essentially by considering

boundary conditions (like [3]'s explanation of waterfilling); for the second problem, since the optimization is broken down into groups with a fixed number of frequencies, the subtraction of some constant number of bits does drop out in the derivatives.

3.4 Coarse Graining Results

Now we move on to the coarse-grained versions of the simulation. The major difference in actually running them was the time involved; because of the matrix lookups involved and other factors, each trial took about two to three times as long as the standard 8×16 simulation. Consequently, not as many trials could be run for smoothing the results (30 instead of 100), but that is still adequate to compare with the original grid results.

For each coarse graining the amplitude of the received signal falls away linearly as the amplitude does, and approximately exponentially as the frequency increases. Superposition of cosines results in the correct power at each frequency. Distributions of the relevant Fourier transform points are approximately Gaussian and independent of the amplitude. The means match the deterministic versions. What changed primarily were the actual amplitudes of the received signal and the rate at which they dropped off with increases in frequency. Using probabilistic particle counting affected mostly the noise, but not the amplitudes of the means in any noticeable way.

Figures 19 to 21 contain a visual summary of the coarse-grained models' results, along with the original, highlighting the differences among them.

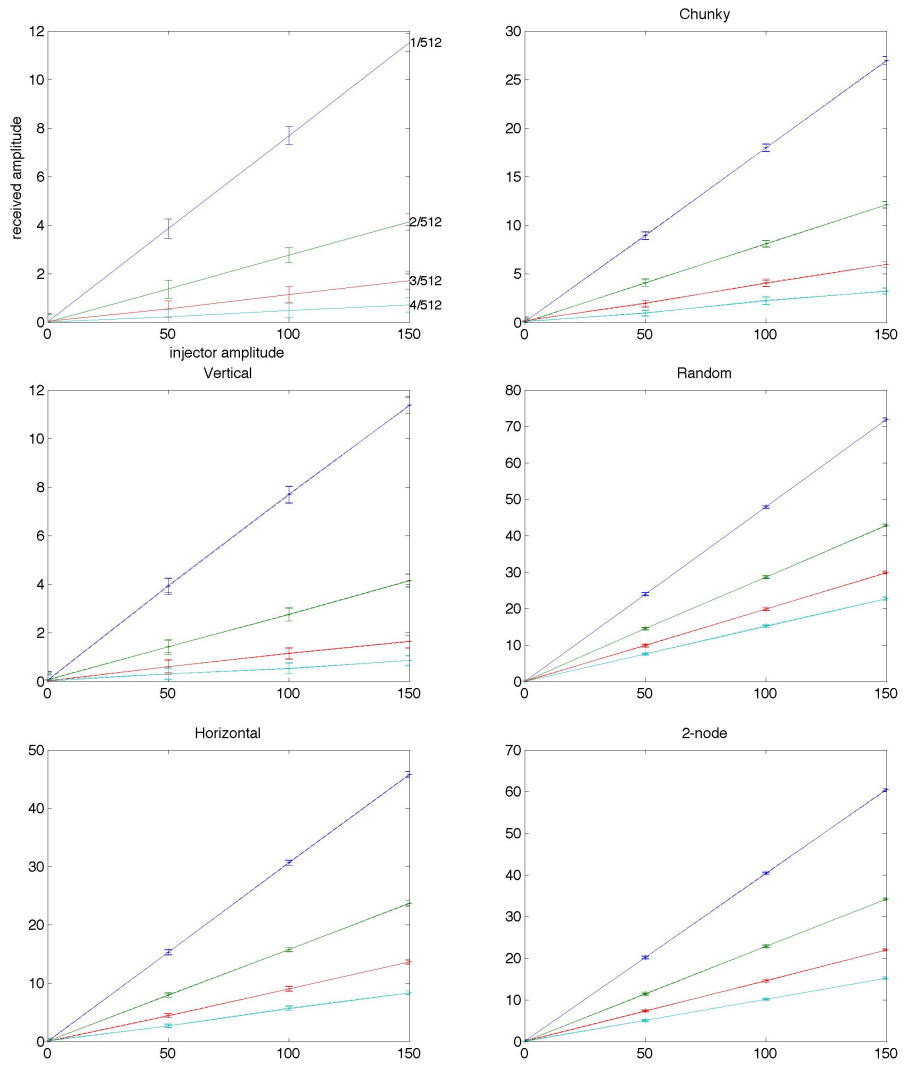

Figure 19: Coarse grainings, showing direct proportion to injector amplitude. Each model is received amplitude vs. injected amplitude. Probabilistic versions are plotted with dotted lines and +s. Error bars representing standard deviations are included for both; at this scale, the difference in them between counting methods is difficult to detect.

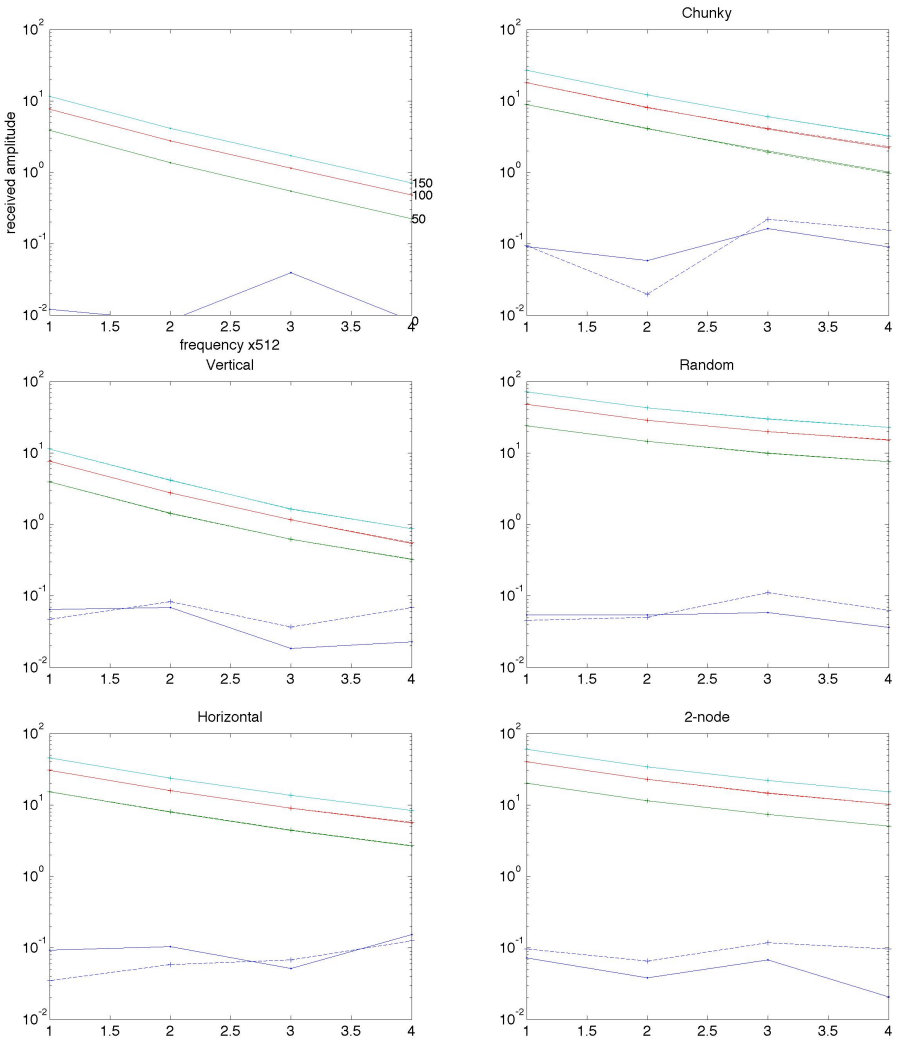

Figure 20: Coarse grainings, showing exponential decay of received amplitude as frequency increases. Each model is received amplitude vs. injector frequency. The scales are identical. Probabilistic versions are plotted with dotted lines and +s.

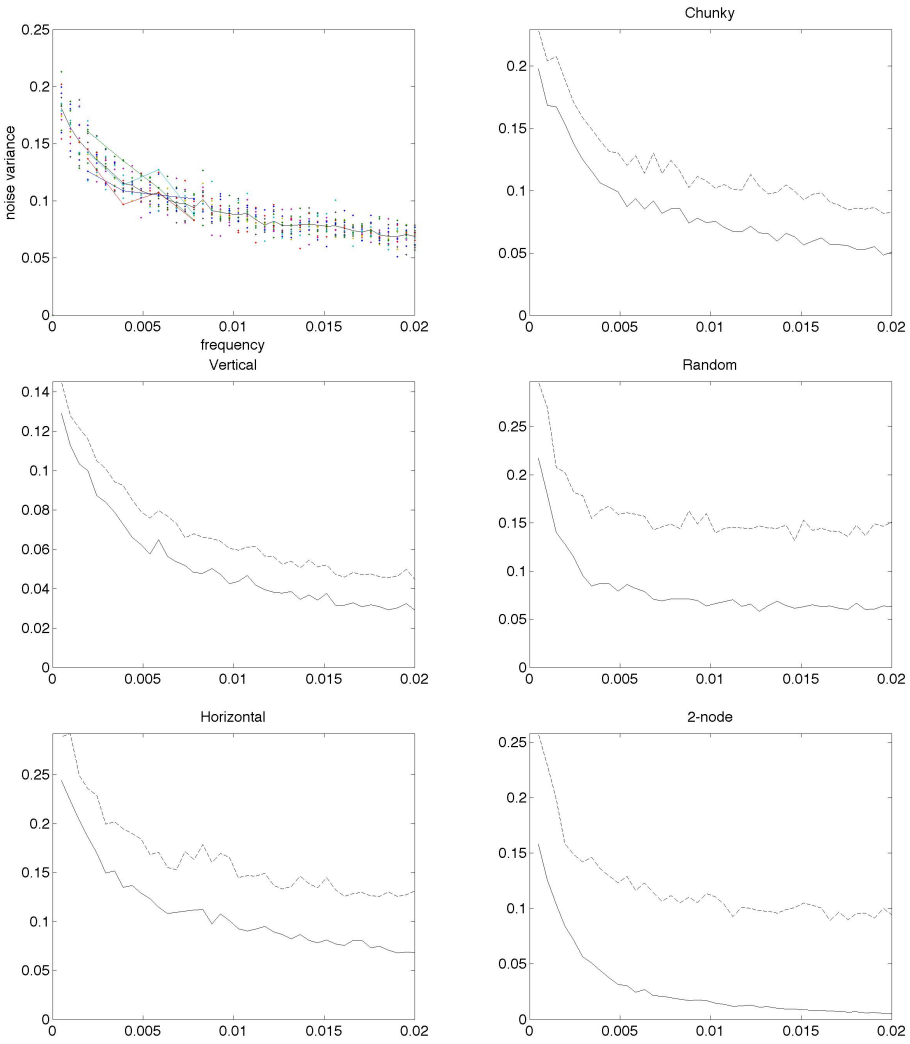

Figure 21: Coarse grainings, showing (mean) noise variance vs. frequency. Note how those for the probabilistic counting (dotted lines) are higher, especially the 2-node coarse graining.

Figure 19 shows the amplitudes of the means of the received signals of various injector types vs. injector amplitude. All are linear, but note the different scales. Surprisingly, the random coarse graining is the highest, with the 2-node grid not far behind. After those two, the horizontal coarse graining and the chunky one continue the decrease, with the vertical coarse graining lowest and in fact matching the original model very closely. This order seems to suggest that horizontal distance from the injector to the detector is instrumental in determining the fraction of particles getting across. (In the random one, there are several very probable routes of just 2 steps from injector to detector, something which was generally true of all the candidate random coarse grainings.)

Figure 20 shows the same data vs. frequency again, with a log-scaled vertical axis to highlight that the amplitude decreases approximately exponentially with frequency. These are all on the same scale. Note, however, that the rate of the drop is much lower (the lines are closer to horizontal) for the random and 2-cell coarse grainings, with a similar increase in the decay rate corresponding to the distance to the injector. In both cases (overall amplitude and exponential decay with frequency), this dependence on number of steps to the other side is notable, especially since the probabilities of movement are adjusted for each coarse graining. Perhaps it is actually the length of the coarse-grained nodes that plays this role; we noted in Section 2 that horizontally longer clusters tended to reduce the bias of the steady state to the left. With more particles to transmit information with, and less time to have to travel and get shuffled around, the increasing frequency does

not affect it as much.

In both figures, note also that the differences between counting 1/4 (or $\frac{1}{64}$ in one case) the particles in the detector node, shown with dots and solid lines, and counting them probabilistically, shown with +s and dotted lines, are negligible. That is, overall (in the mean of 30 runs) the results don't change much. (In fact, even at full size they are hard to tell apart on the graphs.) However, when we look at the variances of the distributions, the extra randomness inserted by the probabilistic counting plays a bigger part, as seen below.

The amplitudes and their relationships are not useful by themselves; it is only in tandem with the variances that we can calculate the information capacity. An analog to Figure 16 (left) is not supplied, because the jagged lines make for difficult comparison; still, the reader may be assured again that the variances are independent of the injector amplitude. Figure 21, instead, shows the means of the variances, highlighting the familiar decreasing variance spectrum, as well as the large jumps in variance when probabilistic counting is used. In particular, the 2-cell coarse graining, with a cluster size of 64 instead of 4, finds a much higher jump in variance overall.

In all cases, the vertical coarse graining is closest to the original model.

3.5 Numerical Capacities for Original and Coarse-Grained Models

The single-channel and waterfilling calculation for the information capacity developed in section 3.3 can be repeated for each of the coarse grainings. We summarize the results in the tables below. The first lists the maximum mean received amplitude a and noise variance N for each frequency and model, along with the channel capacity C in bits as described above.

The second table contains waterfilling results, with amplitude distributions and capacities listed for cases in which four, three, or two channels are active under the constraint condition. The distributions are listed with the lowest (least noisy) frequency first, and are presented as fractions of the original amplitude constraint (for example, .25 instead of $37.25 = .25 \cdot 150$). Blank spaces and parenthetical entries indicate when adding the next noisiest channel does not improve the capacity. An entry is blank when the noisiest channel's new constraints are too low and would result in its contribution to the capacity being less than zero, were it not excluded. Numbers in parentheses represent a slightly different condition: that of the optimal distribution over n channels not resulting in enough improvement over $n-1$ channels to overcome the per-channel offset. Finally, the highest capacities per model are underlined for convenience.

Frequency		1/512	2/512	3/512	4/512
Original	a	11.538	4.137	1.717	0.711
	N	0.144	0.115	0.107	0.094
	C	7.468	4.922	2.685	0.697
Chunky	a	26.964	12.120	6.000	3.238
	N	0.154	0.106	0.094	0.086
	C	9.792	8.043	6.236	4.660
C. prob.	a	26.984	12.117	6.004	3.233
	N	0.189	0.140	0.128	0.124
	C	9.496	7.651	5.804	4.149
Vertical	a	11.375	4.146	1.660	0.872
	N	0.100	0.723	0.065	0.048
	C	7.949	5.564	3.243	1.979
V. prob.	a	11.393	4.172	1.641	0.869
	N	0.116	0.092	0.080	0.068
	C	7.741	5.250	2.943	1.551
Horizontal	a	45.822	23.788	13.647	8.363
	N	0.186	0.135	0.114	0.112
	C	11.036	9.620	8.272	6.921
H. prob.	a	45.841	23.703	13.657	8.345
	N	0.236	0.195	0.171	0.163
	C	10.700	9.086	7.708	6.383
Random	a	71.926	42.924	29.880	22.777
	N	0.128	0.088	0.082	0.071
	C	12.867	11.933	10.982	10.415
R. prob.	a	71.935	42.843	29.949	22.779
	N	0.203	0.163	0.159	0.149
	C	12.209	11.034	10.046	9.358
2-node	a	60.550	34.255	22.030	15.243
	N	0.084	0.044	0.025	0.020
	C	12.973	12.264	11.842	11.111
2 prob.	a	60.481	34.208	22.041	15.247
	N	0.159	0.136	0.116	0.112
	C	12.063	10.648	9.616	8.621

# of channels	4	3	2	1
Original	(proportions)		.5161 .4839	
	(bits)		8.5961	7.4684
Chunky	(.2704 .2623 .2465 .2208)	.3440 .3359 .3201	.5040 .4960	
	(13.6969)	14.8979	13.9121	9.7922
C. prob.	(.2747 .2651 .2463 .2139)	.3460 .3364 .3176	.5048 .4952	
	(12.2260)	13.8263	13.2338	9.4964
Vertical		(.3688 .3445 .2867)	.5122 .4878	
		(8.0902)	9.6805	7.9490
V. prob.			.5140 .4860	
			9.1765	7.7411
Horizontal	.2585 .2546 .2484 .2385	.3380 .3341 .3279	.5020 .4980	
	20.3388	19.6016	16.7021	11.0356
H. prob.	.2607 .2555 .2479 .2360	.3394 .3341 .3265	.5026 .4974	
	18.4627	18.2031	15.8402	10.7001
Random	.2522 .2509 .2491 .2478	.3348 .3335 .3317	.5006 .4994	
	30.3811	26.3518	20.8217	12.8669
R. prob.	.2534 .2513 .2488 .2464	.3356 .3335 .3310	.5010 .4990	
	26.9019	23.8871	19.2723	12.2088
2-node	.2513 .2504 .2498 .2485	.3341 .3333 .3326	.5004 .4996	
	32.3431	27.6371	21.2575	12.9729
2 prob.	.2546 .2519 .2488 .2446	.3362 .3334 .3304	.5014 .4986	
	25.2519	22.9396	18.7437	12.0630

The waterfilling results are notable for a few reasons. Note that the original model and the chunky and vertical coarse grainings have fewer than 4 channels used (some only 2!). As illustrated in Figure 1, under standard AWGN waterfilling with a power constraint it may happen that some channels get no power; in the present case, some channels get no amplitude.

Finally, these results are summarized graphically in Figure 22, which shows capacities for each model at each of the 4 individual frequencies as well as the waterfilling solution. Colors help visualize the height of each bar. Subtle differences in the shades of blue across the lower 32 are present; the waterfilling solutions are particularly marked, with the six models using all four channels shown in red to light green, the two using three channels in a sea-foam green, and those using just two channels in the same blues as the rest.

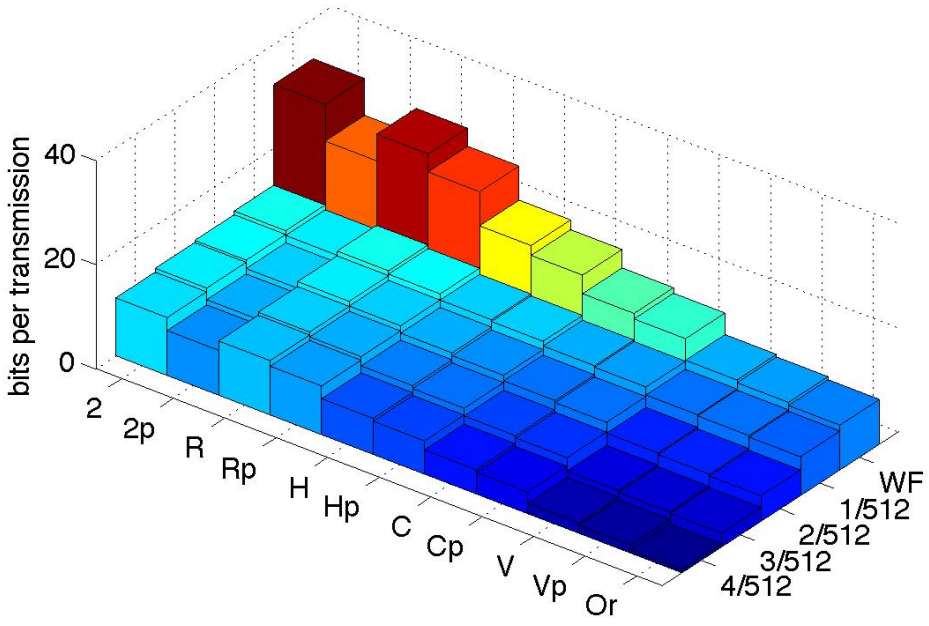

Figure 22: Single-channel capacities at each frequency and multichannel capacity with waterfilling for each coarse-graining scheme. Or=Original, Vp = Vertical (probabilistic), V=Vertical, C=Chunky, H=Horizontal, R=Random, 2=two-node division. See text for notes on the color.

4 Discussion

The exponential drop in the received signal amplitude at a driven frequency and the shift in the phase of the received signal as a function of frequency are typical behavior of a damped driven system that linearly attenuates its inputs. In future work it should be possible to extend the calculations herein to arbitrary geometries and different diffusion and decay constants via analytic solutions of the forced diffusion equation with appropriate parameters. This direction of inquiry lay beyond the scope of the present thesis.

To explore trends a little further numerically, we ran the deterministic simulation at full amplitude on intermediate frequencies, $\frac{1}{2048}$ to $\frac{16}{2048}$, for a total of 16 instead of 4 ($\frac{4}{2048}$, $\frac{8}{2048}$, $\frac{12}{2048}$, $\frac{16}{2048}$). One thing we can do with the received amplitudes at more frequencies is a finer-scale capacity and waterfilling approximation. We have noise measurements for the lowest 16 frequencies ($\frac{1}{2048}, \ldots, \frac{16}{2048}$) from the random simulations, and the deterministic simulations provide approximate amplitudes for them for the original model. Figure 23 shows some results we may look at: $\frac{a}{\sqrt{N}}$ vs. frequency resembles exponential decay, since both a and N approximately do. If we use those in the capacity formula from Section 3 (Results), we see that the individual-frequency capacity decreases approximately linearly with frequency; this should not be surprising, since it's close to the log of an exponential. The appearance of exponential decay and linearity is unlikely to persist beyond this frequency range, however. Finally, we consider the waterfilling case. The optimization solution uses values proportional to $\sqrt{N}/a$, which are simply the reciprocals

of the first plot (in the bottom of the figure). In the same way that one resembles exponential decay in this frequency range, its reciprocal resembles exponential growth. If we are able to represent the noise curve analytically, it may help in the following way. As the length of the transmission period T increases, the lowest possible encoding frequency and the interval between frequencies decreases. Interpolation of the noise levels is useful for any finite number of frequencies that can fit into this range; however, being able to use a standard function for integration would provide an easy way to approximate the waterfilling result in the limit as $T \to \infty$. Such a direction may be the subject of future work.

That the coarse-grained versions of the model all seem to have better information capacity is a surprise at first. The intuitive idea of a coarse graining is that one loses information. However, we lose fine detail, which is not the same as the "information" in information capacity. The 2-node system, for example, has one of the highest capacities despite losing almost all detail; a 1-node system would transmit the injector's time series perfectly and without delay (save for a slight roughness from the Poisson random generator), but could hardly be called a faithful coarse graining. That the vertical one is very close to the original model in all respects seems significant: it seems clear that the horizontal dimension, and the distance across the cell, is the most important, and this is a coarse graining that loses little important detail.

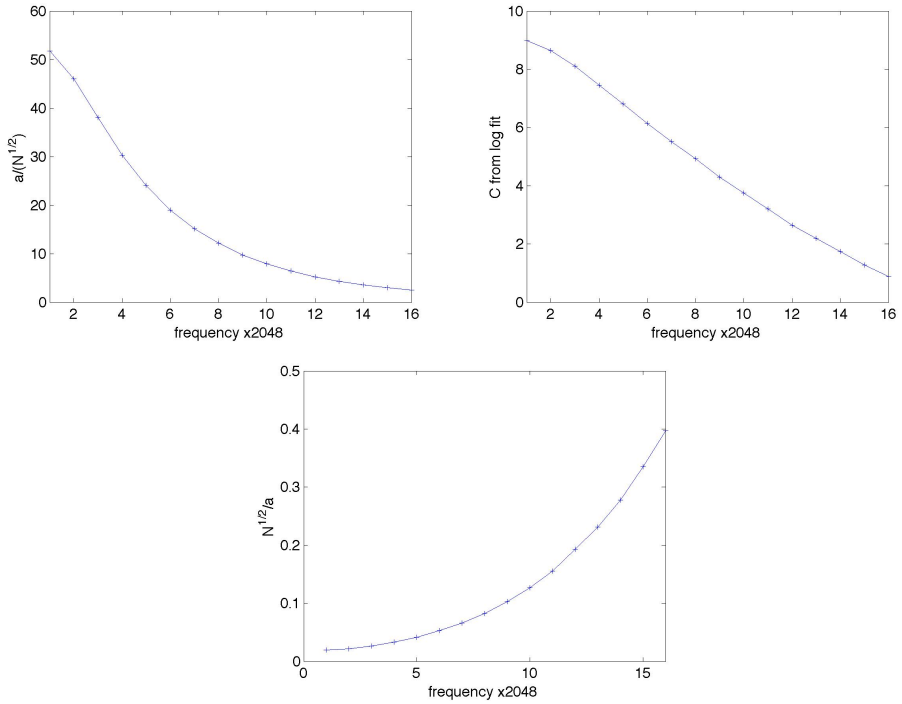

Figure 23: Linearly interpolated channel behavior based on deterministic simulation. We performed (fast) deterministic simulations at integer multiples of the base frequency (1/2048), for multiples from 1 to 16. These simulations filled in the frequencies omitted from the (slow) stochastic simulation sets, and confirmed agreement between the deterministic and mean stochastic behavior for those frequencies at which both were tested (see also Figure 14). Top left: signal-to-noise ratio $a/\sqrt{N}$ versus frequency. Top right: approximate capacity, obtained from the logarithmic fitting procedure (see Results) using a from the deterministic simulations and N from the stochastic simulations, versus frequency. Bottom: effective noise amplitude spectrum to be used for waterfilling calculation (compare Figure 1).

4.1 Capacity in a System without Synchronization

It is an assumption of our model that phase shifts in the input sinusoids can be determined from the output because both input and output are on the same clock. However, if we cannot assume that the detector knows the phase of the original signal, then we cannot use this component, and information can only be sent as the (unsigned) amplitude of the sinusoids.[5]

An analog to the approach used for the disc of inputs in the complex plane may be used, with the same assumption of uniform distribution, integration of the error function, and log fit, but now applied to an interval on the real line. The fact that amplitudes do not average out correctly near 0 will be ignored for this discussion. A full discussion is omitted for brevity, but the results are very similar: capacity is of the form $\log_2(K\frac{a}{\sqrt{N}} + L)$ for some constants K and L, which is roughly half the capacity using the complex disc, differing mainly by an additive constant.

There is a useful geometric interpretation of this result. Imagine a disc in the complex plane full of 2^C evenly spaced circles representing discrete inputs and their ranges of influence (ignoring for now the scaling and rotating effect of β), so that the disc is tessellated with many circles or squares or hexagons. If we wish to restrict the input to a single line, individual discrete inputs would have the same spacing, but there could geometrically only be a number of them proportional to the *square root* of the disc's area, leading to

[5] In the multi-frequency case, we might only apply this restriction to a single frequency, which could then synchronize the rest. Our aim in this section is just to characterize the change in single-frequency information capacity.

a capacity of about $\frac{1}{2}C + \log_2(K)$ if there are $K\sqrt{2^C}$ inputs, for some other constant K. The constant depends mostly on the manner of packing.

Waterfilling may be applied for this interpretation as well, and since the capacity is approximately logarithmic the waterfilling takes the same form as for the full two dimensional case. If it is exactly half the original capacity (plus a constant), the solutions use the same proportions of amplitude, because the term optimized is half the original. The same checks for optimality must be used, but the results of that process can differ significantly because of the change in offsets. For our model and its coarse grainings under this interpretation, the number of channels used is generally reduced; overall this is due to the halved single-channel capacities.

4.2 Brief Comparison to AWGN Capacity Formula

We present these notes out of general interest, but qualitatively for brevity.

The classical information capacity formula for an AWGN channel with power constraint P and noise variance N is $\frac{1}{2}\log_2(\frac{P}{N}+1)$. If we consider this in two dimensions, the capacity doubles exactly to $\log_2(\frac{P}{N}+1) = 2\log_2(\sqrt{\frac{P}{N}+1})$. Our formula is $2\log_2(\frac{a}{\sqrt{N}}+K)-\log_2(2e)$. We have seen that the noise in both models is comparable. The variance of a uniform distribution is proportional to the square of its size (a here); with that in mind, the two information capacity formulas are seen to be fairly close in spirit, though differing in the placement of square root.

In addition, how might the AWGN capacity be used as an approximation

to ours? The original formulas were dependent mostly on the entropy of the output. In the AWGN case, the input and noise are both distributed normally, thus so is the output; in ours, the input is uniform but the noise is normal. If $a/\sqrt{N}$ is very low, the noise overshadows the signal, and the output approaches a normal distribution as $a/\sqrt{N}$ goes to 0. If it is high, on the other hand, the output approaches strict uniformity. (Recall that this same reason was given as justification for the uniform distribution to lead to a reasonable approximation for capacity.) In that case, since the entropy of a uniform distribution can be calculated easily, and differs by a constant from the entropy of a normal distribution with equal variance, we might alternatively use the AWGN formula, minus a constant, for capacity of our system when $a/\sqrt{N}$ is high enough.

4.3 Future Work and Other Notes

Most parameters were not varied at all once the simulations began to work. Possible future areas of exploration include asking how well the signals are transmitted when the diffusion probability is changed (for example to a more stagnant .1 or .05, or up to .25 so particles never sit still) or how the transmission is affected by non-uniform probabilities (for example favoring left-right movement over up-down movement, or strongly favoring movement to the right).

The grid size was not changed at all; would there be significant changes if the ratio of dimensions was altered, or the shape became non-rectangular,

perhaps even three-dimensional, to mimic a real space even better? Some of these variations have been found in the coarse-grained models, though even then in limited amounts, and not isolated (the first coarse graining used .1 probabilities in a 4-by-8 grid, but not one change by itself). We may wish to try a non-rectangular lattice as well; would something hexagonal, for example, help approximate continuous-space particle movement better and change the results?

Another possible area for future work is the elimination of counted particles. Since the particles move independently, once one is detected, it offers no further information [1], yet it continues to be counted again and again as it hovers around the detector space until it eventually decays. It would be interesting to see whether making the counter an absorbing node increased the capacity or not.

As mentioned in Section 2, the pseudoinverse used for coarse graining transition probabilities is not tuned for the actual particle distributions found in the simulation. A better pseudoinverse might allow a more faithful representation of the cell model, while still reducing dimensionality.

A future area for analysis might be the integer granularity inherent in the particle counts. In a system with a more continuous measure of the signal-propagation device, scaling should not matter; here, clearly, if everything were divided by two, starting with the particle injection rate, we would expect lower relative information capacity (and thus higher proportional noise to signal). An area where this can be seen is in the fourth image in Figure 14. The black dots represent the deterministic results, which do not rely

on integer particle counts, and when the signal is small they are noticeably different from the magenta means from the simulations. We would like to know how exactly the discrete particle counts affect results.

5 Conclusions

We have seen how information can be transmitted by particle diffusion in several versions of a simple model. We have obtained estimates of the channel capacity of a diffusion mediated signaling model by examining the stochastic response of the channel to distinct driving input frequencies. Regarding the secretion or injection of diffusing particles at one location as the "input" to the channel and the counting or detection of particles at another location as the "output" of the channel, we have constructed a well defined communications channel that may contribute to developing an understanding of biological communication systems. Through computer simulation we have observed that the input-output relationship of our model system, when viewed in the frequency domain, is quantitatively similar to that of the additive white Gaussian noise (AWGN) channel, a well understood system in communications engineering. By combining simulations and analysis we have obtained approximate expressions for the channel capacity that are analogous to the well known logarithmic signal-to-noise ratio for the AWGN channel capacity, despite the fact that our new channel has a qualitatively different kind of constraint on the ensemble of input signals. Our approximation results in a logarithmic capacity function, which aids greatly in analysis and in adapting the classical AWGN waterfilling approach for our model.

In addition we have examined the effect of replacing a fine grained lattice model of diffusion with a coarse grained version of the same model. Surprisingly, the more information about the microscopic state of the channel

that is "lost" in the coarse graining process, the higher the apparent channel capacity of the resulting channel. We also noted that the coarse grainings having the highest impact on the apparent channel capacity were those that most reduced the shortest path length between the injection and counting sites. A fuller explanation of this observation awaits future work.

6 Appendix

6.1 Matlab Code

Following are some scripts to aid in the simulations described in this paper. Some comments indicate how to set parameters. Variable names are generally different from their use elsewhere in the paper.

Coarse grainings are identified by number or letter throughout, and usually a variable `cg` must be set: 1 (A/chunky), 2 (B/vertical), 3 (C/random), 4 (D/2-node), 5 (E/horizontal).

Otherwise, for all three versions of the simulation, parameters are contained in the code itself. `T` represents the period used, or the reciprocal of the base frequency (the way the code is written, the period is actually `2*T`); `w` is a vector of frequencies to use based on that period. Both may be adjusted if desired. The duration of the simulation and the amount of it that is used for analysis is hard-coded, but references to numbers in the vicinity of 1024, 2048, and 3072 represent timesteps. The function `r` and its parameter `a` may be adjusted, along with transition probability `p` and decay rate `decay`.

The most useful things to adjust are a short list. `Amp` is a matrix with each row representing a fraction amplitude set to use for the injector; it has as many columns as the size of `w`, with each one corresponding to one frequency in `w`. The code provided uses 16 rows that give the results found in this paper. `imax` is simply the number of times to run each amplitude set; it is set to 100 for the original model and 30 for the coarse grainings (and should stay at 1 for the deterministic version).

After any desired adjustments are made to any of the three scripts, the code may be run. When the simulation is complete, save the desired output data before running the script again. The most important are `Pzz` (the transform), `seenvtotal` (each column represents one detector timeseries), and `rtotal` (each column represents one injector timeseries). These are described below.

Afterward, to find means and standard deviations for the stochastic versions, be sure to separate out by `Amp` type. Code like this (written for 2048 timesteps, 16 amplitude sets, and 30 runs of each) works. Remember that standard deviation in one direction is based on half the complex variance, so in that use, divide by $\sqrt{2}$.

```
Pzzm=zeros(2048,16);
Pzzs=zeros(2048,16);
for i=1:16;
 Pzzm(:,i)=mean(Pzz(:,i*30-29:i*30),2);
 Pzzs(:,i)=std(Pzz(:,i*30-29:i*30),1,2);
end
```

Data can be located easily. Except for the mean and standard deviation, each column represents one run, with each group of `imax` representing one amplitude set. (For mean and standard deviation, there is one column per amplitude set.) In the Fourier transform data, the first row corresponds to $f = 0$, and should be all 0s; the next half correspond to each next frequency, as outlined in Methods. (See the note of caution there.) In the simulations as used in this paper, frequencies $f = \frac{1}{512}, \frac{2}{512}, \frac{3}{512}, \frac{4}{512}$ correspond to rows 5, 9, 13, and 17.

6.1.1 Stochastic Simulations (Original Model)

Some parameters to adjust: `Amp` is amplitude fractions for the injector (currently for the first four multiples of $f = 1/512$), `imax` is the number of times to run the simulation per amplitude set.

```
%Run this for the main simulation.

%Most easily adjusted parameters are found first:

Amp = [1,0,0,0;0,1,0,0;0,0,1,0;0,0,0,1;
       2/3,0,0,0; 0,2/3,0,0; 0,0,2/3,0; 0,0,0,2/3;
       1/3,0,0,0; 0,1/3,0,0; 0,0,1/3,0; 0,0,0,1/3;
       0,0,0,0;
       2/3,1/3,0,0; 1/3,2/3,0,0; 1/3,1/3,0,0];
%Amp contains the list of amplitude vectors to use on a particular run.
%Each one is run a certain number of times before moving to the next. In
%this case, it assumes a frequency vector of length 4, but that can be
%changed in conjunction with code below. Note that these are fractions of
%the base amplitude.


typemax=size(Amp);
typemax=typemax(1); %simply to grab the number of amplitude vectors

imax=100;
%imax is the number of times to run the simulation for each amplitude type.


%More setup/initialization, not too much to vary, though.

t=0; %discrete time step
A=[]; %position array (n rows by 2 cols)
Pzz=[];
rtotal=[];
rvec=[];
seenvtotal=[];
%These four (along with some later) just keep records of all the runs.
%Pzz contains the power spectra, one column per run;
%rvec (along with seenv below) keeps track of the injector (detector) for
%the duration of one run;
%rtotal and seenvtotal contain each run's rvec and seenv.
```

```
%probability of moving in some direction (so 1-4p is staying put)
p=.2; %be sure it's less than .25!

%decay rate
decay=.01;

%grid size = 16 x 8 points with entry at 1,4 and eye at 16,4
gridx=16;
gridy=8;
entry=[1,4];
eye=[16,4];


seenv=[]; %see above


T=256;
%period to make base frequency; NOTE that it is later doubled (sorry!) (The
%doubled value should divide the number of timesteps used in the FFT.)

w=(0:(T/2))*2*pi/(2*T);
w = w(2:5);
%Note the 2T that represents the true base period. w(0) is 0, so w(2:5)
%gives the first four useful frequencies.


%The actual simulation begins:

for type=(1:typemax)
for iter=(1:imax)
amp = Amp(type,:); %amp now has one row of the user-defined Amp above


    %reinitializations of everything that gets replaced per run
    t=0;seenv=[];A=[];
    rvec=[];


a=150; %admittedly hidden down here, this is the base amplitude
b=150*ones(size(w))'; %ignore this

r = inline('poissrnd(max([0,(1+amp*cos(w*t)'')*a]))','t','a','w','amp');
%The main function that does everything. amp and w are rows, so a transpose
%is needed (in the string, '' is needed to represent '), then the actual
%value of the function is truncated at 0 if nececssary, and the result
```

```
%passed to the Poisson random number generator.


while(t<=3071) %A little more hard-coding;
               %t starts at 0 and we want 3072 steps

    count=max([0,round(r(t,a,w,amp))]); %just to be sure we
                                        %get a positive int
    for i=1:count
        A=[A;entry]; %for each particle we want to inject,
                     %add it to the list at the entry point
    end

    rvec=[rvec;count]; %keep track of the number of particles injected


    %diffuse
    D=rand(size(A(:,1))); %generate a list of random numbers in [0,1],
                          %one per particle
    %for 0<r<=p add 1 to x
    %for p<r<=2p subtract 1 from x
    %for 2p<r<=3p add 1 to y
    %for 3p<r<=4p subtract 1 from y
    %else stay put
    A(:,1)=min(gridx*ones(size(A(:,1))),A(:,1)+          (D<=p));
    A(:,1)=max(ones(size(A(:,1))),      A(:,1)-  ((p<D)&(D<=2*p)));
    A(:,2)=min(gridy*ones(size(A(:,2))),A(:,2)+((2*p<D)&(D<=3*p)));
    A(:,2)=max(ones(size(A(:,2))),      A(:,2)-((3*p<D)&(D<=4*p)));

    %decay
    D=rand(size(A(:,1))); %see above, we just want a new random list
    A=A(find(D>decay),:); %retains only those particles whose random
                          %numbers are above decay rate

    %how many eye sees
    seen=sum((A(:,1)==eye(1))&(A(:,2)==eye(2)));

    switch seen
        otherwise
            seenv=[seenv;t,seen];
    end
    %switch functionality was used earlier, but now, we just build up seenv
    %as a record of the detector for the entire run
```

```
    t=t+1;
end %end while(<=3071)


%Once the 3072 steps are done, some analysis and record-keeping:

%subtract the mean so there's no spike of the FFT at frequency 0
svminusmean=seenv(:,2)-mean(seenv(1025:3072,2));
Z=fft(svminusmean(1025:3072,:),2048);
Pzz = [Pzz,2*Z/2048]; %see thesis text for doubling explanation

    rtotal=[rtotal,rvec];
    seenvtotal=[seenvtotal,seenv(:,2)]; %more archiving

iter %to let me keep tabs on the progress of the whole batch

end %end for iter=(1:imax)

amp %also to let me keep tabs

end %end for type=(1:typemax)
```

6.1.2 Stochastic Simulations (Coarse Grainings)

The variable `cg` needs to be set first, as noted above. See below for instructions on generating probabilistic counts.

Some parameters to adjust (as before): `Amp` is amplitude fractions for the injector (currently for the first four multiples of $f = 1/512$), `imax` is the number of times to run the simulation per amplitude set.

See additional comments in the original model's code.

```
Amp = [1,0,0,0;0,1,0,0;0,0,1,0;0,0,0,1;
       2/3,0,0,0; 0,2/3,0,0; 0,0,2/3,0; 0,0,0,2/3;
       1/3,0,0,0; 0,1/3,0,0; 0,0,1/3,0; 0,0,0,1/3;
       0,0,0,0;
       2/3,1/3,0,0; 1/3,2/3,0,0; 1/3,1/3,0,0];

typemax=size(Amp);
typemax=typemax(1);
```

```
imax=30;

t=0;
A=[];
Pzz=[];
rtotal=[];
rvec=[];
seenvtotal=[];

%coarse...B = destinations, C = prob distr

if (cg==1)

 B = [2,9,1,0,0;
      1,3,10,2,0;
      2,4,11,3,0;
      3,5,12,4,0;
      4,6,13,5,0;
      5,7,14,6,0;
      6,8,15,7,0;
      7,16,8,0,0;
      1,10,17,9,0;
      2,9,11,18,10;
      3,10,12,19,11;
      4,11,13,20,12;
      5,12,14,21,13;
      6,13,15,22,14;
      7,14,16,23,15;
      8,15,24,16,0;
      9,18,25,17,0;
      10,17,19,26,18;
      11,18,20,27,19;
      12,19,21,28,20;
      13,20,22,29,21;
      14,21,23,30,22;
      15,22,24,31,23;
      16,23,32,24,0;
      17,26,25,0,0;
      18,25,27,26,0;
      19,26,28,27,0;
      20,27,29,28,0;
      21,28,30,29,0;
      22,29,31,30,0;
      23,30,32,31,0;
      24,31,32,0,0]
```

```
C= [.1,.2,1,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,1,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,.4,1;
    .1,.2,.3,1,1;
    .1,.2,1,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,.3,1,1;
    .1,.2,1,1,1]

entry = 9;
eye=16;


elseif (cg==2)

    B = [2,17,1,0;
        1,3,18,2;
        2,4,19,3;
        3,5,20,4;
        4,6,21,5;
```

```
5,7,22,6;
6,8,23,7;
7,9,24,8;
8,10,25,9;
9,11,26,10;
10,12,27,11;
11,13,28,12;
12,14,29,13;
13,15,30,14;
14,16,31,15;
15,32,16,0;
18,1,17,0;
17,19,2,18;
18,20,3,19;
19,21,4,20;
20,22,5,21;
21,23,6,22;
22,24,7,23;
23,25,8,24;
24,26,9,25;
25,27,10,26;
26,28,11,27;
27,29,12,28;
28,30,13,29;
29,31,14,30;
30,32,15,31;
31,16,32,0]

C=[.2,.25,1,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.4,.45,1;
    .2,.25,1,1;
    .2,.25,1,1;
```

```
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.4,.45,1;
            .2,.25,1,1]

    entry=1;
    eye=16;

elseif (cg==3)

    %generated by my ti-92, then sorted, and grouped by hand...
    %sanity check to follow
    B = [5,6,9,13,18,19,25,28,29,30,1,0,0,0,0;
        4,7,8,10,12,14,16,21,26,30,2,0,0,0,0;
        4,5,9,11,17,18,21,27,28,30,3,0,0,0,0;
        2,3,5,6,8,11,12,24,25,26,28,31,4,0,0;

        1,3,4,8,9,11,15,17,19,20,23,25,27,5,0;
        1,4,8,9,13,19,21,23,26,28,29,30,6,0,0;
        2,13,14,16,17,18,20,24,26,7,0,0,0,0,0;
        2,4,5,6,17,18,21,23,27,30,31,8,0,0,0;

        1,3,5,6,12,16,17,20,21,22,26,30,31,9,0;
        2,13,14,15,17,22,24,28,10,0,0,0,0,0,0;
        3,4,5,13,14,15,16,17,19,24,25,27,29,11,0;
        2,4,9,15,23,24,25,27,29,32,12,0,0,0,0;

        1,6,7,10,11,14,15,16,21,24,26,32,13,0,0;
        2,7,10,11,13,20,22,26,31,14,0,0,0,0,0;
        5,10,11,12,13,18,19,21,22,25,28,31,15,0,0;
        2,7,9,11,13,22,27,28,29,30,16,0,0,0,0;

        3,5,7,8,9,10,11,18,22,26,28,30,31,32,17;
        1,3,7,8,15,17,25,27,29,30,31,32,18,0,0;
```

```
1,5,6,11,15,20,21,23,24,25,27,29,31,19,0;
5,7,9,14,19,22,25,26,27,28,29,30,32,20,0;

2,3,6,8,9,13,15,19,22,24,26,29,21,0,0;
9,10,14,15,16,17,20,21,26,29,30,22,0,0,0;
5,6,8,12,19,24,25,28,29,31,23,0,0,0,0;
4,7,10,11,12,13,19,21,23,26,29,32,24,0,0;

1,4,5,11,12,15,18,19,20,23,30,31,25,0,0;
2,4,6,7,9,13,14,17,20,21,22,24,26,0,0;
3,5,8,11,12,16,18,19,20,28,31,32,27,0,0;
1,3,4,6,10,15,16,17,20,23,27,31,28,0,0;

1,6,11,12,16,18,19,20,21,22,23,24,32,29,0;
1,2,3,6,8,9,16,17,18,20,22,25,31,32,30;
4,8,9,14,15,17,18,19,23,25,27,28,30,31,0;
12,13,17,18,20,24,27,29,30,32,0,0,0,0,0]

C=[.05,.1,.15,.25,.35,.4,.45,.55,.6,.65,1,1,1,1,1;
    .1,.2,.25,.3,.4,.5,.55,.6,.65,.7,1,1,1,1,1;
    .05,.1,.15,.2,.25,.35,.4,.5,.55,.6,1,1,1,1,1;
    .1,.15,.2,.25,.3,.35,.45,.5,.55,.65,.7,.75,1,1,1;

    .05,.1,.15,.25,.3,.35,.4,.45,.5,.6,.65,.7,.75,1,1;
    .05,.1,.15,.2,.3,.35,.4,.45,.5,.55,.6,.65,1,1,1;
    .1,.15,.2,.35,.4,.45,.5,.55,.6,1,1,1,1,1,1;
    .05,.1,.2,.25,.3,.35,.4,.45,.6,.65,.7,1,1,1,1;

    .05,.1,.15,.2,.35,.45,.5,.55,.6,.65,.7,.75,.8,1,1;
    .05,.1,.15,.2,.3,.4,.5,.55,1,1,1,1,1,1,1;
    .05,.1,.15,.2,.25,.3,.35,.4,.45,.5,.6,.65,.7,1,1;
    .1,.2,.35,.4,.5,.55,.6,.65,.7,.8,1,1,1,1,1;

    .1,.2,.25,.3,.35,.4,.45,.5,.55,.6,.65,.7,1,1,1;
    .1,.15,.2,.25,.3,.35,.45,.55,.7,1,1,1,1,1,1;
    .05,.1,.15,.2,.25,.3,.35,.4,.45,.5,.55,.6,1,1,1;
    .05,.2,.3,.35,.4,.45,.5,.55,.6,.65,1,1,1,1,1;

    .05,.1,.15,.2,.25,.35,.4,.45,.5,.55,.6,.65,.75,.8,1;
    .1,.2,.25,.3,.35,.4,.45,.5,.55,.6,.65,.7,1,1,1;
    .05,.1,.15,.2,.25,.3,.4,.45,.5,.55,.6,.7,.75,1,1;
    .1,.15,.2,.25,.3,.4,.45,.5,.55,.6,.65,.7,.8,1,1;

    .05,.1,.15,.2,.25,.3,.35,.45,.5,.65,.7,.75,1,1,1;
    .05,.15,.25,.3,.35,.4,.5,.55,.6,.7,.75,1,1,1,1;
```

```
            .05,.1,.15,.25,.3,.35,.4,.45,.55,.6,1,1,1,1,1;
            .05,.1,.2,.25,.3,.35,.4,.55,.6,.65,.7,.8,1,1,1;

            .05,.1,.15,.25,.3,.35,.4,.45,.5,.55,.6,.65,1,1,1;
            .05,.15,.2,.25,.3,.35,.45,.5,.55,.6,.65,.7,1,1,1;
            .1,.15,.3,.35,.4,.45,.5,.55,.6,.65,.7,.75,1,1,1;
            .1,.15,.2,.25,.3,.35,.4,.45,.5,.55,.6,.65,1,1,1;

            .05,.1,.15,.2,.25,.3,.4,.45,.5,.6,.7,.75,.8,1,1;
            .05,.1,.15,.2,.25,.3,.35,.4,.45,.5,.55,.6,.65,.7,1;
            .05,.1,.15,.3,.35,.45,.5,.55,.6,.65,.7,.75,.8,1,1;
            .1,.15,.2,.25,.35,.45,.5,.55,.6,1,1,1,1,1,1]

            %to verify things are ok, two things:
            sum(sum((C-[zeros(32,1),C(:,1:14)]).*B)) %magic numbers
                                                     %i'm afraid

  %           C - also check for weird rows
  %           (sort each row somehow, subtract
  %           from C, see if all are 0
            %it's going to be harder to do it that way, so i'll just:
            max(C-[zeros(32,1),C(:,1:14)])
            min(C-[zeros(32,1),C(:,1:14)])
            %should weed out any weird things like 5 instead of .5 that
            %the sum wouldn't catch

     entry=4;
     eye=32;

 elseif (cg==4)


B=[2,1;
    1,2]
C=[.025,1;
    .025,1]
entry=1;
eye=2;


 elseif (cg==5)

B = [2,9,1,0,0;
    1,3,10,2,0;
    2,4,11,3,0;
```

```
    3,5,12,4,0;
    4,6,13,5,0;
    5,7,14,6,0;
    6,8,15,7,0;
    7,16,8,0,0; %end col 1
    10,1,17,9,0;
    9,11,2,18,10;
    10,12,3,19,11;
    11,13,4,20,12;
    12,14,5,21,13;
    13,15,6,22,14;
    14,16,7,23,15;
    15,8,24,16,0; %end col 2
    18,9,25,17,0;
    17,19,10,26,18;
    18,20,11,27,19;
    19,21,12,28,20;
    20,22,13,29,21;
    21,23,14,30,22;
    22,24,15,31,23;
    23,14,32,24,0; %end col 3
    26,17,25,0,0;
    25,27,18,26,0;
    26,28,19,27,0;
    27,29,20,28,0;
    28,30,21,29,0;
    29,31,22,30,0;
    30,32,23,31,0;
    31,24,32,0,0]

C = [.2,.25,1,1,1;
    .2,.4,.45,1,1;
    .2,.4,.45,1,1;
    .2,.4,.45,1,1;
    .2,.4,.45,1,1;
    .2,.4,.45,1,1;
    .2,.4,.45,1,1;
    .2,.25,1,1,1;

    .2,.25,.3,1,1;
    .2,.4,.45,.5,1;
    .2,.4,.45,.5,1;
    .2,.4,.45,.5,1;
    .2,.4,.45,.5,1;
    .2,.4,.45,.5,1;
```

```
    .2,.4,.45,.5,1;
    .2,.25,.3,1,1;

    .2,.25,.3,1,1;
    .2,.4,.45,.5,1;
    .2,.4,.45,.5,1;
    .2,.4,.45,.5,1;
    .2,.4,.45,.5,1;
    .2,.4,.45,.5,1;
    .2,.4,.45,.5,1;
    .2,.25,.3,1,1;

    .2,.25,1,1,1;
    .2,.4,.45,1,1;
    .2,.4,.45,1,1;
    .2,.4,.45,1,1;
    .2,.4,.45,1,1;
    .2,.4,.45,1,1;
    .2,.4,.45,1,1;
    .2,.25,1,1,1]

entry=4;
eye=28;



 end

decay=.01;
seenv=[];


T=256; %remember that the actual period is double this
w=(0:(T/2))*2*pi/(2*T);
w = w(2:5);

for type=(1:typemax)
for iter=(1:imax)
amp = Amp(type,:);

    t=0;seenv=[];A=[];
    rvec=[];

a=150;
b=150*ones(size(w))'; %ignore this
```

```
r = inline('poissrnd(max([0,(1+amp*cos(w*t)'')*a]))','t','a','w','amp');

while(t<=3071)
    count=max([0,round(r(t,a,w,amp))]);
    for i=1:count
        A=[A;entry];
    end

    rvec=[rvec;count];

    %diffuse
    D=rand(size(A(:,1))); %how can i just get the number of rows easily
    for n=(1:size(A(:,1)));
        A(n) = B(A(n), min(find(C(A(n),:)>D(n))));
    end;

    %decay
    D=rand(size(A(:,1)));
    A=A(find(D>decay),:);

if (cg==4) %need to divide by 64
    seen = sum(A==eye)/64; %64 CG nodes per node
else
    seen = sum(A==eye)/4; %4 CG nodes per node
end

    switch seen
        otherwise
            seenv=[seenv;t,seen];
    end
    t=t+1;
end

svminusmean=seenv(:,2)-mean(seenv(1025:3072,2));
Z=fft(svminusmean(1025:3072,:),2048);

Pzz = [Pzz,2*Z/2048]; %see thesis text for doubling explanation
    rtotal=[rtotal,rvec];
    seenvtotal=[seenvtotal,seenv(:,2)];

iter
end %end iter
amp
end %end amps
```

To do probabilistic counts, apply a method like this. Here 3072 and 480 are based on the size of `seenvtotal`; the 4 and .25 must be changed to 64 and 1/64 for the 2-node version.

```
seenvtotalp=seenvtotal;
for i=(1:3072);
 for j=(1:480);
  seen=4*seenvtotal(i,j);
  seenvtotalp(i,j)=sum(rand(seen,1)<.25);
 end;
end;
```

To obtain the Fourier transform, which is usually done in the script, this code works. (It's a little nicer than doing it at each iteration as well.) Note that the mean is not subtracted before performing the transform, so the first row is set to 0 manually here. Again, 1025 and 3072 are based on the timesteps used, and the division by 1024 stands for the division by 2048 (number of timesteps) and multiplication by 2 (see Methods).

```
Pzzp=fft(seenvtotalp(1025:3072,:))/1024; Pzzp(1,:)=0;
```

6.1.3 Deterministic Simulations (All Models)

Before running the script, be sure the matrices are set by running the following. It looks odd but gets the job done without using too much space.

```
p=.2;
P=diag(ones(128,1)*(1-4*p))+diag(ones(112,1)*p,16)+diag(ones(112,1)*p,-16);
%main diagonal (stay still), prob going up,        prob of going down
 %add up/down for top/bottom rows into main diagonal
 P(1:16,1:16) = P(1:16,1:16) + diag(ones(16,1)*p);
 P(113:128,113:128) = P(113:128,113:128) + diag(ones(16,1)*p);

 %doing the left-right motion is more difficult
 %for each block representing a cell row i want to add a tridiagonal matrix
```

```
 %with p above and below, and p at each end of the main diagonal

 for i=1:16:113
P(i:i+15,i:i+15)=P(i:i+15,i:i+15)+diag(ones(15,1)*p,1)+diag(ones(15,1)*p,-1);
     P(i,i)=P(i,i)+p;
     P(i+15,i+15)=P(i+15,i+15)+p;
 end


%making transition matrices W for all 5 coarse grainings
%for each, make kind of a pattern, build one area, then repeat (except C)

%WA (2x2 blocks, row=1 to 8, 9 to 16, etc) - inject 9, eye 16
WA=zeros(32,128);
for i=1:8
    j=i*2-1;
    WA(i,j:j+1) = [1,1];
    WA(i,j+16:j+17) = [1,1];
end
%makes a single block
WA(9:16,33:64)=WA(1:8,1:32);
WA(17:32,65:128)=WA(1:16,1:64);
%duplicates it to the rest


%WB (vertical chunks, row = 1 to 16, 17 to 32) - inject 1, eye 16
WB=zeros(32,128);
%make first half, which is 4 diagonals
for i=0:3
    WB(1:16,16*i+1:16*(i+1))=diag(ones(16,1));
end
WB(17:32,65:128)=WB(1:16,1:64);

%WC (random) - oh god (inject 4, eye 32)
WC=zeros(32,128);
%i guess i'll just have to through the list and add 128 1s
WC(3,1)=1;
WC(5,2)=1;
WC(11,3)=1;
WC(14,4)=1;
WC(13,5)=1;
WC(26,6)=1;
WC(2,7)=1;
WC(10,8)=1;
WC(10,9)=1;
```

```
WC(22,10)=1;
WC(14,11)=1;
WC(26,12)=1;
WC(6,13)=1;
WC(13,14)=1;
WC(21,15)=1;
WC(15,16)=1;
WC(11,17)=1;
WC(15,18)=1;
WC(13,19)=1;
WC(10,20)=1;
WC(24,21)=1;
WC(21,22)=1;
WC(8,23)=1;
WC(17,24)=1;
WC(22,25)=1;
WC(20,26)=1;
WC(7,27)=1;
WC(7,28)=1;
WC(13,29)=1;
WC(1,30)=1;
WC(29,31)=1;
WC(18,32)=1;
WC(25,33)=1;
WC(12,34)=1;
WC(32,35)=1;
WC(24,36)=1;
WC(21,37)=1;
WC(19,38)=1;
WC(31,39)=1;
WC(9,40)=1;
WC(26,41)=1;
WC(14,42)=1;
WC(2,43)=1;
WC(16,44)=1;
WC(16,45)=1;
WC(28,46)=1;
WC(20,47)=1;
WC(32,48)=1;
WC(4,49)=1;
WC(2,50)=1;
WC(12,51)=1;
WC(29,52)=1;
WC(22,53)=1;
WC(29,54)=1;
```

```
WC(23,55)=1;
WC(12,56)=1;
WC(4,57)=1;
WC(31,58)=1;
WC(14,59)=1;
WC(22,60)=1;
WC(30,61)=1;
WC(31,62)=1;
WC(27,63)=1;
WC(32,64)=1;
WC(6,65)=1;
WC(21,66)=1;
WC(9,67)=1;
WC(16,68)=1;
WC(9,69)=1;
WC(6,70)=1;
WC(19,71)=1;
WC(24,72)=1;
WC(26,73)=1;
WC(17,74)=1;
WC(31,75)=1;
WC(15,76)=1;
WC(25,77)=1;
WC(18,78)=1;
WC(8,79)=1;
WC(30,80)=1;
WC(28,81)=1;
WC(3,82)=1;
WC(3,83)=1;
WC(27,84)=1;
WC(12,85)=1;
WC(23,86)=1;
WC(29,87)=1;
WC(32,88)=1;
WC(20,89)=1;
WC(5,90)=1;
WC(25,91)=1;
WC(19,92)=1;
WC(11,93)=1;
WC(27,94)=1;
WC(8,95)=1;
WC(6,96)=1;
WC(10,97)=1;
WC(17,98)=1;
WC(18,99)=1;
```

```
WC(3,100)=1;
WC(4,101)=1;
WC(24,102)=1;
WC(11,103)=1;
WC(17,104)=1;
WC(30,105)=1;
WC(9,106)=1;
WC(20,107)=1;
WC(5,108)=1;
WC(4,109)=1;
WC(8,110)=1;
WC(5,111)=1;
WC(1,112)=1;
WC(15,113)=1;
WC(28,114)=1;
WC(1,115)=1;
WC(30,116)=1;
WC(2,117)=1;
WC(7,118)=1;
WC(16,119)=1;
WC(7,120)=1;
WC(18,121)=1;
WC(1,122)=1;
WC(19,123)=1;
WC(27,124)=1;
WC(28,125)=1;
WC(23,126)=1;
WC(23,127)=1;
WC(25,128)=1;

%WD (two-cell, except that node 1 has 1 to 8, 17 to 24, etc.)
WD=zeros(2,128);
WD(1:2,1:16)=[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0;
              0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1];
WD(:,17:32)=WD(:,1:16);
WD(:,33:64)=WD(:,1:32);
WD(:,65:128)=WD(:,1:64);

%WE (horizontal chunks, but COLUMNS = 1 to 8, etc.) - inject 4, eye 28)
WE=zeros(32,128);
for i=1:8
    j=i*16-15;
    WE(i,j:j+3)=[1,1,1,1];
end
WE(9:16,5:128)=WE(1:8,1:124);
```

```
WE(17:32,9:128)=WE(1:16,1:120);


%creating actual transition matrices now:
sP=sparse(P);
WPA=WA*P*pinv(WA);
WPB=WB*P*pinv(WB);
WPC=WC*P*pinv(WC);
WPD=WD*P*pinv(WD);
WPE=WE*P*pinv(WE);
```

The variable `cg` needs to be set first, as noted above. 0 may be used for the original model.

As before, `Amp` is amplitude fractions for the injector (currently for the first four multiples of $f = 1/512$). Since this script was used for additional frequencies, this would be the place to adjust `T` and `w` for a period of 2048 and more frequencies; `Amp` needs as many columns as the length of `w` as well.

See additional comments in the original model's code.

```
switch(cg)
    case 0
        trans=sP; %sparse P
        entry=49;
        eye=64;
    case 1
        trans=WPA;
        entry=9;
        eye=16;
    case 2
        trans=WPB;
        entry=1;
        eye=16;
    case 3
        trans=WPC;
        entry=4;
        eye=32;
    case 4
        trans=WPD;
        entry=1;
```

```
        eye=2;
    case 5
        trans=WPE;
        entry=4;
        eye=28;
end

%NOTE: 1 runthrough, NO POISSON, NO ROUNDING

Amp = [1,0,0,0;0,1,0,0;0,0,1,0;0,0,0,1;
       2/3,0,0,0; 0,2/3,0,0; 0,0,2/3,0; 0,0,0,2/3;
       1/3,0,0,0; 0,1/3,0,0; 0,0,1/3,0; 0,0,0,1/3;
       0,0,0,0;
       2/3,1/3,0,0; 1/3,2/3,0,0; 1/3,1/3,0,0];

typemax=size(Amp);
typemax=typemax(1);
imax=1; %only needs to be run once

t=0;
%instead of position array, vector
xt=zeros(size(trans,1));

Pzz=[]
rtotal=[];
rvec=[];
seenvtotal=[];

decay=.01;

seenv=[];

T=256; %NOTE this is still half the actual period
w=(0:(T/2))*2*pi/(2*T);
w = w(2:5);

for type=(1:typemax)
for iter=(1:imax)
amp = Amp(type,:);

    t=0;seenv=[];xt=zeros(size(xt));
    rvec=[];

a=150;
b=150*ones(size(w))'; %ignore this
```

```
r = inline('max([0,(1+amp*cos(w*t)'')*a])','t','a','w','amp');

while(t<=3071)

    count=max([0,r(t,a,w,amp)]);
    xt(entry)=xt(entry)+count;

    rvec=[rvec;count];

    %instead of diffuse and decay...
    xt=(1-decay)*trans*xt;

    %and instead of below...
    seen=xt(eye)*(size(trans,1)/128);

    switch seen
        otherwise
            seenv=[seenv;t,seen];
    end

    t=t+1;
end

svminusmean=seenv(:,2)-mean(seenv(1025:3072,2));
Z=fft(svminusmean(1025:3072,:),2048);

Pzz = [Pzz,2*Z/2048]; %2 is for neg freq
    rtotal=[rtotal,rvec];
    seenvtotal=[seenvtotal,seenv(:,2)];

iter
end %end iter
amp
end %end type of amps to use
```

References

[1] H.C. Berg and E.M. Purcell. Physics of chemoreception. *Biophys J*, 20(2):193–219, Nov 1977.

[2] Chakra Chennubhotla and Ivet Bahar. Signal propagation in proteins and relation to equilibrium fluctuations. *PLoS Comput Biol*, 3(9):1716–1726, Sep 2007.

[3] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[4] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford University Press, Oxford, 3rd ed edition, 2001.

[5] M B Kennedy. Signal-processing machines at the postsynaptic density. *Science*, 290(5492):750–754, Oct 27 2000.

[6] C A Parent and P N Devreotes. A cell's sense of direction. *Science*, 284(5415):765–770, Apr 30 1999.

[7] P Rickert, O D Weiner, F Wang, H R Bourne, and G Servant. Leukocytes navigate by compass: roles of PI3Kgamma and its lipid products. *Trends Cell Biol*, 10(11):466–473, Nov 2000.

[8] Peter J. Thomas, Donald J. Spencer, Sierra K. Hampton, Peter Park, and Joseph P. Zurkus. The diffusion-limited biochemical signal-relay channel.

In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[9] Christopher M Waters and Bonnie L Bassler. Quorum sensing: cell-to-cell communication in bacteria. *Annu Rev Cell Dev Biol*, 21:319–346, 2005.