Accelerating large-scale simulations of cortical neuronal network development

Fumitaka Kawasaki

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Computing and Software Systems

University of Washington

2012

Committee:

Michael Stiber, Chair

Charles Jackels

Kelvin Sung

Munehiro Fukuda

Program Authorized to Offer Degree:
Computing and Software Systems

University of Washington

Abstract

Accelerating large-scale simulations
of cortical neuronal network development

Fumitaka Kawasaki

Chair of the Supervisory Committee:
Professor Michael Stiber
Computing and Software Systems

Cultured dissociated cortical cells grown into networks on multi-electrode arrays are used to investigate neuronal network development, activity, plasticity, response to stimuli, the effects of pharmacological agents, etc. We made computational models of such neuronal networks and studied the interplay of individual neuron activity, cell culture development, and network behavior. For small networks (100 neurons in a 10×10 arrangement), we concluded that our simulations' behaviors were dominated by their limited size. However, increasing network size required huge computational resources: for a single-threaded simulator, a 100×100 neuron simulation would take at least 2,000 hours (83 days). To tackle this problem, we ported the network simulator to the GPU. A first, naive implementation performed about 2.4 times faster than the single threaded simulator. By progressively modifying the simulator structure, we achieved about 23 times performance gain compared with the single threaded simulator, bringing large-scale simulations into the realm of feasibility. We executed a set of simulations of networks of 100×100 arrangements on GPU. We made statistical analyses of bursts generated by simulations, and found basic relationship between simulation parameters (independent variables), network structure (connectivity), and burst profiles (emergent properties).

TABLE OF CONTENTS

Page

List of Figures iii

List of Tables vi

Chapter 1: BIOLOGICAL BACKGROUND 1
1.1 Neurons and Synapses 1
1.2 Action Potential Generation 3

Chapter 2: COMPUTATIONAL MODEL 6
2.1 A Model of Dissociated Cortical Tissue 6
2.2 Neural Culture Simulator 12

Chapter 3: PRELIMINARY EXPERIMENTS 15
3.1 Computer Implementation 15
3.2 Analysis Methods 16
3.3 Results 17
3.4 Analysis 22

Chapter 4: IMPLEMENTATION FOR GPU 24
4.1 GPU Architecture and CUDA 24
4.2 Match between Algorithm Structure and GPU Architecture 26
4.3 Related Work 27
4.4 GPU Implementation 28
4.5 Performance Results 34
4.6 Performance Analysis 39

Chapter 5: EXPERIMENTS WITH LARGE NETWORKS 44
5.1 Computer Implementation 44
5.2 Analysis Methods 45
5.3 Results 64

Chapter 6: ANALYSIS 78
6.1 Influence of Simulation Parameters on Network Connectivity and Behavior . 78
6.2 Comparison to Previous Results 83

Chapter 7: CONCLUSION AND FUTURE WORK 90

Bibliography 92

LIST OF FIGURES

Figure Number — Page

1.1 The major types of neuron and its parts. . . . 2
1.2 Strucure of a chemical synapse. . . . 3
1.3 The phase of action potential. . . . 5

2.1 The equivalent circuit of the integrate-and fire model. . . . 7
2.2 Dependence of outgrowth/retraction on cell firing rate. . . . 10
2.3 Geometric relationship between two circles for computation of area of intersection. . . . 11
2.4 Simulator algorithm structure. . . . 12
2.5 Simulator architecture. . . . 13

3.1 Layouts used for different numbers of endogenously active and inhibitory neurons. . . . 16
3.2 Firing rates normalized relative to target rate plotted versus the two simulation parameters. . . . 17
3.3 Fractions of neurons with stable connectivity radii as a function of the two simulation parameters. . . . 18
3.4 Burstiness index versus simulation parameters for 20,000-25,000sec (A) and 25,000–30,000sec (B). . . . 18
3.5 Detailed simulation results for simulations with parameters. . . . 20
3.6 Effects of neurite outgrowth on bursting for simulations with parameters. . . . 21

4.1 CUDA C1060 GPU architecture. . . . 25
4.2 Run times of the naive GPU implementation vs. single thread. . . . 29
4.3 Structure of neuron's summation point. . . . 30
4.4 *ISimulation* interface that includes single-threaded, multi-threaded, and GPU implementations. . . . 32
4.5 Summary of speedups produced by different techniques. . . . 33

5.1 Layouts used for different numbers of endogenously active and inhibitory neurons. . . . 44

5.2 Evolution of burstiness index, radius, and firing rate for a simulation with parameters (0.1, 0.90). 46
5.3 Evolution of burstiness index, radius, and firing rate for a simulation with parameters (1.0, 0.90). 47
5.4 Evolution of burstiness index, radius, and firing rate for a simulation with parameters (1.9, 0.90). 48
5.5 Evolution of burstiness index, radius, and firing rate for a simulation with parameters (0.1, 0.98). 49
5.6 Evolution of burstiness index, radius, and firing rate for a simulation with parameters (1.0, 0.98). 50
5.7 Evolution of burstiness index, radius, and firing rate for a simulation with parameters (1.9, 0.98). 51
5.8 Example of burst patterns. Bursts around 10,800 sec (A) and 52,400 sec (B) with parameters (target rate, fraction excitatory cells) = (1.0, 0.98). 53
5.9 Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (0.1, 0.90) 54
5.10 Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (1.0, 0.90) 55
5.11 Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (1.9, 0.90) 56
5.12 Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (0.1, 0.98) 57
5.13 Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (1.0, 0.98) 58
5.14 Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (1.9, 0.98) 59
5.15 Distribution of spike counts (Hz) per neuron of all firing rate of entire simulation period with parameters. 62
5.16 Evolution of burst shape with parameters. 66
5.17 Evolution of mean burst height with parameters. 67
5.18 Evolution of mean burst width (blue) and mean peak position (red) with parameters 68
5.19 Evolution of mean spike counts per burst with parameters. 69
5.20 Evolution of burst counts with parameters. 70
5.21 Evolution of APNFR with parameters. 71
5.22 Distributions histograms of IBIs with parameter (0.1, 0.90) 72
5.23 Distributions histograms of IBIs with parameter (1.0, 0.90) 73

5.24 Distributions histograms of IBIs with parameter (1.9, 0.90) 74
5.25 Distributions histograms of IBIs with parameter (0.1, 0.98) 75
5.26 Distributions histograms of IBIs with parameter (1.0, 0.98) 76
5.27 Distributions histograms of IBIs with parameter (1.9, 0.98) 77

6.1 Evolution of APNFR with parameters. 82
6.2 Distribution histograms of IBIs between 50,000–60,000 sec and PDF plots. . . 89

LIST OF TABLES

Table Number Page

2.1 Neuron parameters. 8
2.2 Synapse parameters. 9
4.1 C1060 parameters. 26
4.2 Measured performance of three kernels. Basic data and cumulative execution time for each kernel function. 35
4.3 Measured performance of three kernels. Thread divergence data. 36
4.4 Measured performance of three kernels. 38
4.5 Measured performance of three kernels. 40
4.6 Percentage ratio of effective performance to peak performance. 42
5.1 The ratio of APNFR below 0.5 Hz per neuron ($\sum_{i=0}^{0.5} P_i$) to total firing. . . . 61
6.1 Mean burst height (Hz per neuron) and burst width (sec) between 38,000–39,000 sec (top values) and between 59,000–60,000 sec (bottom values). . . . 79
6.2 Coefficient of variation (CV) of IBI distribution during 30,000–39,000 sec (top values) and during 50,000–60,000 sec (bottom values). 80
6.3 Mean neurite radii of ordinary neurons at 39,000 sec (top values) and at 60,000 sec (bottom values) (excitatory, non-spontaneously active, interior). . 81
6.4 Standard deviation of neurite radii of ordinary neurons at 39,000 sec (top values) and at 60,000 sec (bottom values) (excitatory, non-spontaneously active, interior). 81
6.5 Range (the difference between the maximum and the minimum) of neurite radii of ordinary neurons at 39,000 sec (top values) and at 60,000 sec (bottom values) (excitatory, non-spontaneously active, interior). 81
6.6 GEV distribution parameters (including 95% confidence intervals for the parameters) between 50,000 and 60,000 sec simulation time of each simulation. 88

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to the Department of Computing & Software Systems for their extended long-term support and especially to Professor Michael Stiber for his vast reserve of patience and knowledge. This thesis would never have been completed without the encouragement and devotion of my family and friends.

Chapter 1

BIOLOGICAL BACKGROUND

The brain is an information processing system; however, the brain employs different data processing principles from those of digital computers: interaction of a huge number of simple processing elements, neurons, rather than independent contributions of more complex processing elements, processors, in parallel computing. Major factor of information processing in the brain is attributed to the interaction of neurons, not to a single neuron. Such capability that emerge from interaction of a huge number of neurons — *the emergent properties* — is the single most defining property of neural computation.

Creating a theory to understand mechanisms underlying brain functions is an important goal of neuroscience. However, the brain is the most complex information processing system in the known universe. Therefore, only experimental data, emergent properties, of the real brain give us cues to understand the mechanisms of the brain. This approach is different than that of traditional science, where a set of principles and rules induce a theory.

In contrast to the experimental domain, computational neuroscience tries to create hypotheses regarding how the brain operates. Such hypotheses are realized into models, evaluated analytically or numerically, and tested against experimental data. This is the way that computational neuroscience tackles the problem.

1.1 Neurons and Synapses

Though neurons are biological cells and have common features of other cells, they are specialized for signal processing. The shape of neurons vary considerably, and they are morphologically classified, but there are major structural features, a cell body and root-like extensions called neurites, which are further distinguished into the receiving fibers of

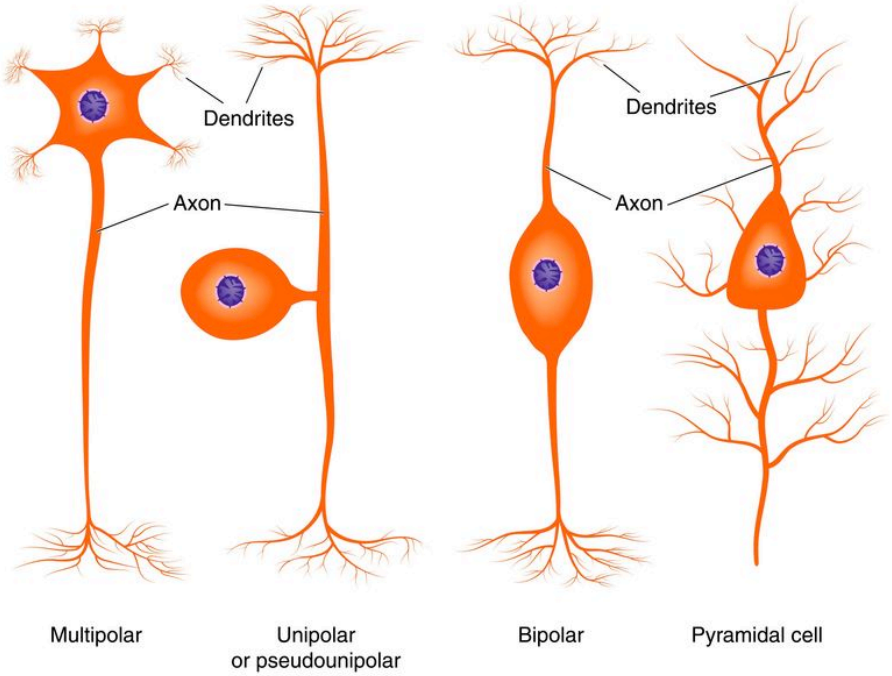

Figure 1.1: The major types of neuron and its parts.

neurons, dendrites, and one major outgoing trunk, an axon (Figure 1.1). Neurons receive signals from many other neurons contacting at specialized sites, called synapses. Synapses enable signals from a presynaptic neuron (the neuron sending the signal) to alter the state of a postsynaptic neuron (the neuron receiving the signal), and this can eventually trigger the generation of an electric pulse, or action potential, in the postsynaptic neuron. This is usually initiated at the initial segment of the axon, the axonhillock, and subsequently travels along the axon until it reaches synapses with other neurons.

Synapses consist of a specialized extension, the axon terminal, and specific receiving sites on dendrites. Axon terminals synthesize special chemicals called neurotransmitters, which are concentrated and stored in synaptic vesicles, and released by the arrival of an action potential. The released neurotransmitters drift across the synaptic cleft, a small gap between the axon terminal and the dendrite of the postsynaptic neuron, and regulate

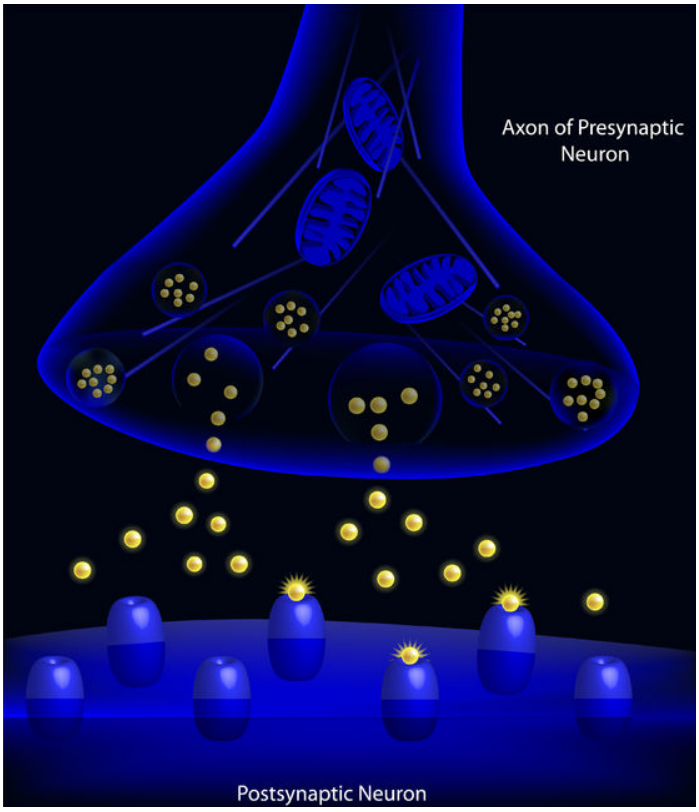

Figure 1.2: Strucure of a chemical synapse.

neurotransmitter-gated ion channels in postsynaptic dendrites to open and close, changing the membrane potential of the postsynaptic cell (Figure 1.2). The response in the membrane potential is called the postsynaptic potential (PSP). Some neurotransmitters increase the PSP to drive the postsynaptic neurons toward their excited state, whereas other neurotransmitters drive the PSP toward the resting potential. These processes (and the synapses) are said to be excitatory or inhibitory, respectively.

1.2 Action Potential Generation

The membrane potential, V_m, is defined as the difference between the electric potential within a cell and its surroundings. The origin of this potential difference comes from the different concentration of ions within and outside a cell, which is caused by the neuronal

membrane being selectively permeable to specific ions, so diffusion processes can increase the concentration difference. Eventually, the force generated by the concentration difference balances the electric force generated by ionic charges, and the neuron settles at an equilibrium state. For example, the equilibrium potential for potassium (K^+) in a typical neuron is a value around $-80mV$, and sodium (Na^+) is typically around $-65mV$. The selective permeability of cell membranes to certain ions is achieved via ion channels, which form pores that enable specific ions to enter or to leave cells. Sodium (Na^+), potassium (K^+), calcium (Ca^{2+}), and chloride (Cl^-) are common ions involved in such processes.

The change in the membrane potential can ultimately trigger the generation of an action potential, a spike, in compartments of the neuron with different voltage-sensitive ion channels, which open and close as a function of the voltage of the membrane. At least two types of voltage-dependent ion channels, sodium (Na^+) channels and potassium (K^+) channels, and ion pumps, which acts as a pump to move substances across the membrane, are involved in the minimal mechanisms for the generation of a spike [35] (Figure 1.3):

1. The resting potential is maintained by leakage channels through which potassium ions (K^+) can flow as a result of concentration differences between the inside and the surrounding fluid.

2. The cell is depolarized, a sharp increase of the membrane potential to the positive value, when voltage-gated sodium (Na^+) channels allow the influx of positively charged sodium ions (Na^+).

3. After a short time, sodium (Na^+) channels are blocked and voltage-gated potassium (K^+) channels open. This results in hyper polarization, undershooting the normal resting potential of the neuron.

4. Finally, the hyper polarization causes the inactivation of voltage-gated channels and a return to the resting potential.

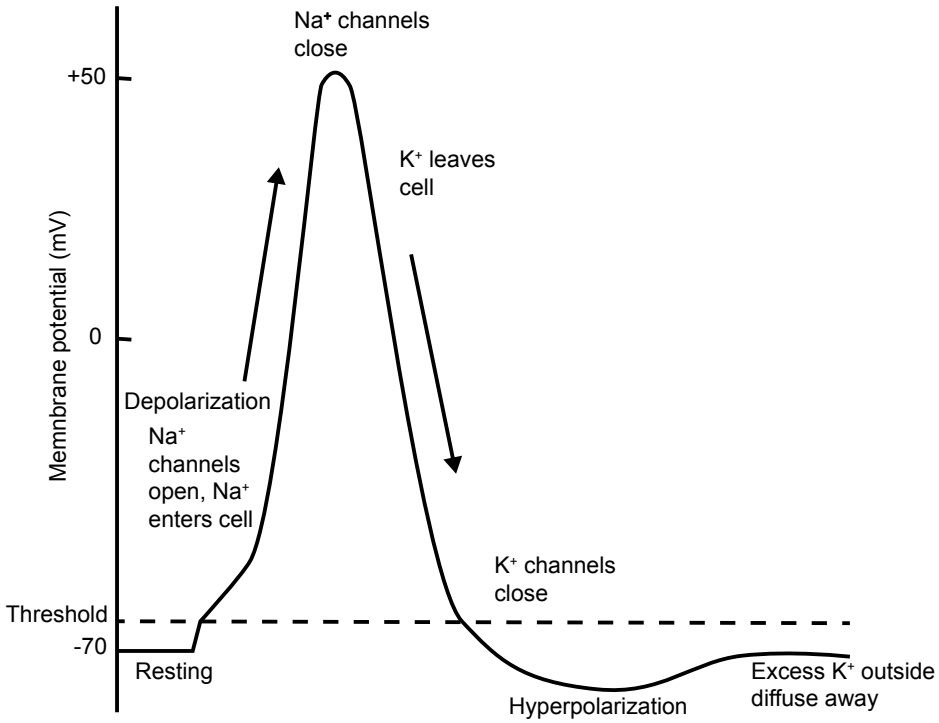

Figure 1.3: The phase of action potential.

5. Repeated generation of action potentials results in a repeated efflux of potassium (K^+) and influx of sodium (Na^+), eventually leading to the failure to generate action potentials. To cope with this problem, ion pumps can transfer specific ions against their concentration level.

Chapter 2

COMPUTATIONAL MODEL

2.1 A Model of Dissociated Cortical Tissue

How activities and connections of individual neurons contribute to the development and computation of cerebral cortex is one of the central questions in neuroscience. A powerful experimental approach for investigating these questions is growing networks of cultured dissociated cortical cells on multi-electrode arrays. Such preparations allow investigation of network development, activity, plasticity [13], responses to stimuli [12], the effects of pharmacological agents [6], etc. In such experiments, network behavior commonly converges to whole-culture pathological (in the sense that it does not occur *in vivo*) bursting that generally interferes with the experimental goals [29]. This bursting is interesting from both a theoretical point of view [37], as well as a clinical one. Understanding the mechanisms that underlie bursting could allow creation of more useful cell cultures and have medical applications [41]. Approaching the problem by comptational study, we were simulating a mathematical model of cell culture connectivity growth. This model includes descriptions of neuron, synapse (connection), and development (connection formation) dynamics.

We applied a simplified neuron model that neglects the details of underlying structure of neurons, such as ion-channel dynamics and concentration. This neuron model is the integrate-and-fire type and includes synaptic, constant bias, and noise currents [21]. The model approximates the dynamic integration of synaptic input, spike timing, and resetting after spikes. A neuron is modeled by using an electric circuit consisting of a parallel capacitor and resistor, where C_m, R_m, V_m, V_{rest}, and I represent the membrane capacitance, the membrane resistance, the membrane potential, the resting membrane potential, and injected currents respectively (Figure 2.1).

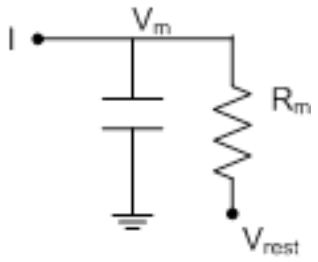

Figure 2.1: The equivalent circuit of the integrate-and fire model.

According to *Kirchoff's law*, the model is formalized:

$$C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} = \frac{1}{R_m}(V_{\mathrm{rest}} - V_m) + I_{\mathrm{syn}} + I_{\mathrm{inj}} + I_{\mathrm{noise}} \tag{2.1}$$

When V_m exceeds the threshold voltage V_{thresh}, it is reset to V_{rest} and held there for the length of the absolute refractory period, T_{refract}.

Synapses exhibit dynamics that include activity-dependent facilitation and depression [23, 36]. Their model has four state variables: three that govern the fraction of synaptic resources in particular states — x (recovered state), y (active state), and z (inactive state) — and one, u, that represents synaptic efficiency,

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{z}{\tau_{\mathrm{rec}}} - ux\delta(t - t_{\mathrm{sp}}) \tag{2.2}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{y}{\tau_I} + Ux\delta(t - t_{\mathrm{sp}}) \tag{2.3}$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = \frac{y}{\tau_I} - \frac{z}{\tau_{\mathrm{rec}}} \tag{2.4}$$

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{u}{\tau_{\mathrm{facil}}} + U(1 - u)\delta(t - t_{\mathrm{sp}}) \tag{2.5}$$

where $\delta(t - t_{\mathrm{sp}})$ is the unit impulse at time t_{sp}, the arriving spike time. The three time constants τ_I, τ_{rec}, and τ_{facil} govern inactivation after an arriving spike, recovery from inactivation, and facilitation after a spike, respectively. The synaptic current produced by an arriving spike is $I_{\mathrm{syn}} = Wy$, where W is the strength of the synaptic connection. See

Table 2.1: Neuron parameters.

Parameter	Model	Summary	Unit	Value
t_{m}	Neuron	The membrane time constant	sec	3.0×10^{-2}
C_{m}	Neuron	The membrane capacity	C	3.0×10^{-8}
R_{m}	Neuron	The membrane resistance	Ohm	1.0×10^{-6}
V_{resting}	Neuron	The resting membrane voltage	Volt	0.0
V_{thresh}	Neuron	The threshold voltage	Volt	$[13.565 \times 10^{-3}, 13.655 \times 10^{-3}]$ for spontaneously active neuron, 15.0×10^{-3} for other neuron
V_{reset}	Neuron	The voltage to reset Vm to after a spike	Volt	13.5×10^{-3}
I_{noise}	Neuron	The standard deviation of the noise to be added each integration time constant	Ampere	$[1.0 \times 10^{-9}, 1.5 \times 10^{-9}]$
V_{init}	Neuron	The initial condition for Vm at time t = 0	Volt	$[13.0 \times 10^{-3}, 13.5 \times 10^{-3}]$
T_{refract}	Neuron	The absolute refractory period	sec	3.0×10^{-3} for excitatory neuron, 2.0×10^{-3} for inhibitory neuron

table 2.1 and 2.2 for parameter values used [22].

Simulations were conducted by constructing networks with model neurons on a rectangular grid. Synaptic strength (connectivity), W, was determined dynamically by a model of neurite (cell input and output region) growth and synapse formation [40]. In this, a cell's region of connectivity is modeled as a circle with radius that changes at a rate inversely proportional to a sigmoidal function of cell firing rate [1]:

$$\frac{\mathrm{d}R_i}{\mathrm{d}t} = \rho G(F_i) \tag{2.6}$$

$$G(F_i) = 1 - \frac{2}{1 + \exp((\epsilon - F_i)/\beta)} \tag{2.7}$$

[1]This is a phenomenological model derived by a number of studies that demonstrated low level of electric activity (low firing rate) stimulated neurite outgrowth, and high level of electric activity (high firing rate) led to regression [4][7][31][9]. A theory, Ca^{2+} theory of neurite outgrowth [18][17], has been proposed to explain this phenomena, which states that changes in intracellular calcium concentrations regulate alterations in outgrowth as well as level of electrical activity.

Table 2.2: Synapse parameters. II—inhibitory to inhibitory, IE—inhibitory to exitatory, EI—exitatory to inhibitory, and EE—exitatory to exitatory synapses.

Parameter	Model	Summary	Unit	Value
I_{inject}	Neuron	The constant current to be injected into the LIF neuron	Ampere	13.5×10^{-9}
U	Synapse	The use parameter		II:0.32 IE:0.25 EI:0.05 EE:0.5
D	Synapse	The time constant of the depression	sec	II:0.144 IE:0.7 EI:0.125 EE:1.1
F	Synapse	The time constant of the facilitation	sec	II:0.06 IE:0.02 EI:1.2 EE:0.05
W	Synapse	The weight (scaling factor, strength, maximal amplitude)		II:19 IE:19 EI:60 EE:30
τ_{s}	Synapse	The synaptic time constant	msec	II:6 IE:6 EI:3 EE:3
delay	Synapse	The synaptic transmission delay	msec	II:0.8 IE:0.8 EI:0.8 EE:1.5

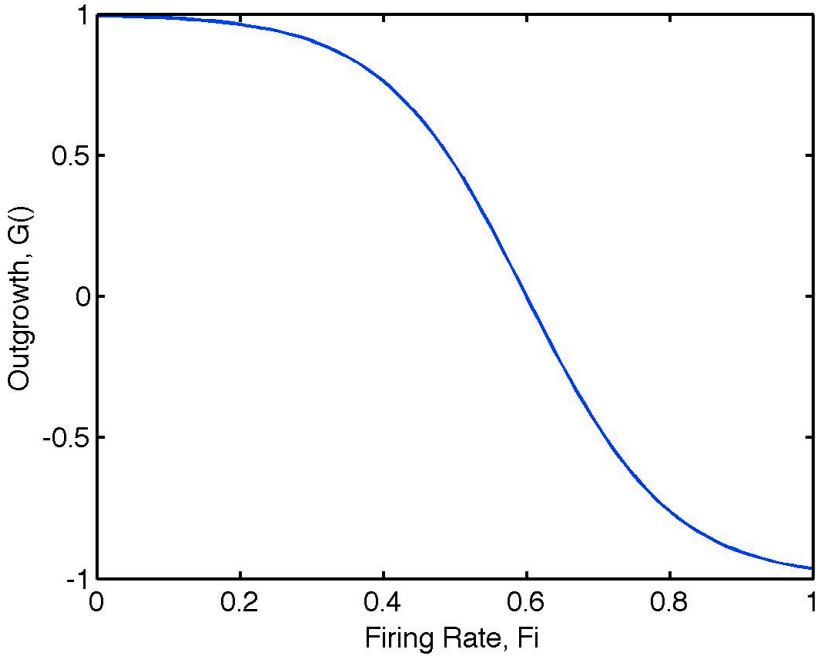

Figure 2.2: Dependence of outgrowth/retraction on cell firing rate.

where R_i is the radius of connectivity of neuron i, F_i is neuron i's firing rate (normalized to be in the range $[0, 1]$), ρ is an outgrowth rate constant, ϵ is a constant that sets the "null point" for outgrowth (the firing rate in spikes/sec that causes no outgrowth or retraction), and β determines the slope of $G(\cdot)$. Figure 2.2 shows $G(\cdot)$, where $\epsilon = 0.6$ and $\beta = 0.1$. One divergence in these simulations from strict modeling of the living preparation was that ρ was increased to reduce simulated development times from the weeks that the living preparation takes to 60,000s (approximately 16 simulated hours). Extensive analysis and simulation was performed to determine the maximum ρ ($\rho = 0.0001$) that would not interfere with network dynamics (the increased value of ρ was still orders of magnitude slower than the slowest of the neuron or synapse time constants, which were order of $10^{-2} \sim 10^{-3}$ sec).

Synaptic strengths were computed for all pairs of neurons that had overlapping connectivity regions as the area of their circle's overlap, as shown in Figure 2.3 and equations

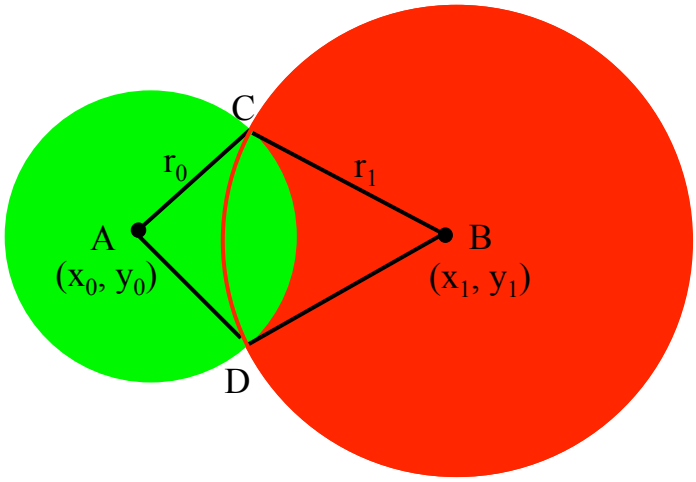

Figure 2.3: Geometric relationship between two circles for computation of area of intersection.

(2.8)– (2.14).

$$r_0^2 = r_1^2 + |AB|^2 - 2r_1|AB|\cos(\angle CBA) \tag{2.8}$$

$$\cos((\angle CBA) = \frac{r_1^2 + |AB|^2 - r_0^2}{2r_1|AB|} \tag{2.9}$$

$$\angle CBD = 2\angle CBA \tag{2.10}$$

$$\cos(\angle CAB) = \frac{r_0^2 + |AB|^2 - r_1^2}{2r_0|AB|} \tag{2.11}$$

$$\angle CAD = 2\angle CAB \tag{2.12}$$

$$w_{01} = \frac{1}{2}\angle CBD r_1^2 - \frac{1}{2}r_1^2\sin(\angle CBD) + \frac{1}{2}\angle CAD r_0^2 - \frac{1}{2}r_0^2\sin(\angle CAD) \tag{2.13}$$

$$w_{01} = w_{10} \tag{2.14}$$

The bulk of the neurons in the network were excitatory; a small number were chosen to be inhibitory. Similarly, most neurons were not spontaneously active, but a few had their firing threshold, V_{thresh}, lowered from 15 mV to 13.565 mV $\leq V_{\text{thresh}} \leq$ 13.655 mV to produce spontaneous firing at a rate of between 0.02 and 6 spikes/sec.

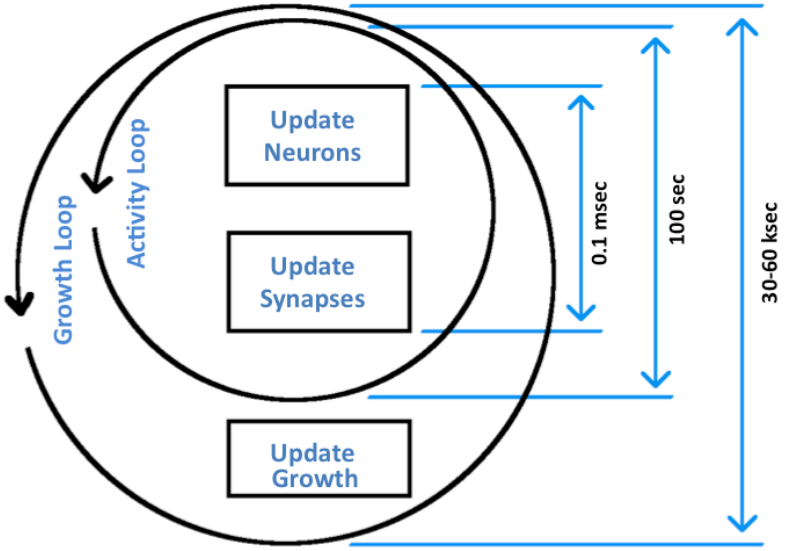

Figure 2.4: Simulator algorithm structure. A 0.1ms time step included neuron and synapse updates and was repeated 1,000,000 times to produce a 100s activity epoch. Activity epochs, separated by growth updates, were repeated to produce 30,000–60,000s simulations.

2.2 Neural Culture Simulator

As shown in figure 2.4, each simulation proceeded as a sequence of 100 second segments (activity epochs) with 0.1 millisecond time step, during which the connectivity was kept constant. For each time step, states of neurons and synapses were updated, according to equations (2.1) and (2.2)–(2.5), and neurons' spikes history (neuron number and time of fire) was logged. At the end of each epoch, the average firing rate of each neuron during the epoch was calculated based on the spikes history, and it was used to adjust its neurite outgrowth, according to equation (2.6), for the next. Simulations were 300–600 activity epochs long (30,000–60,000 seconds). All neurons' spikes history and radii history were recorded for later analysis.

Before GPU implementation, the simulator supported parallelization using OpenMP, in addition to a single-threaded version. This architecture is illustrated in figure 2.5, where *ISimulation* interfaces with underlying different implementations. Neuron and synapse up-

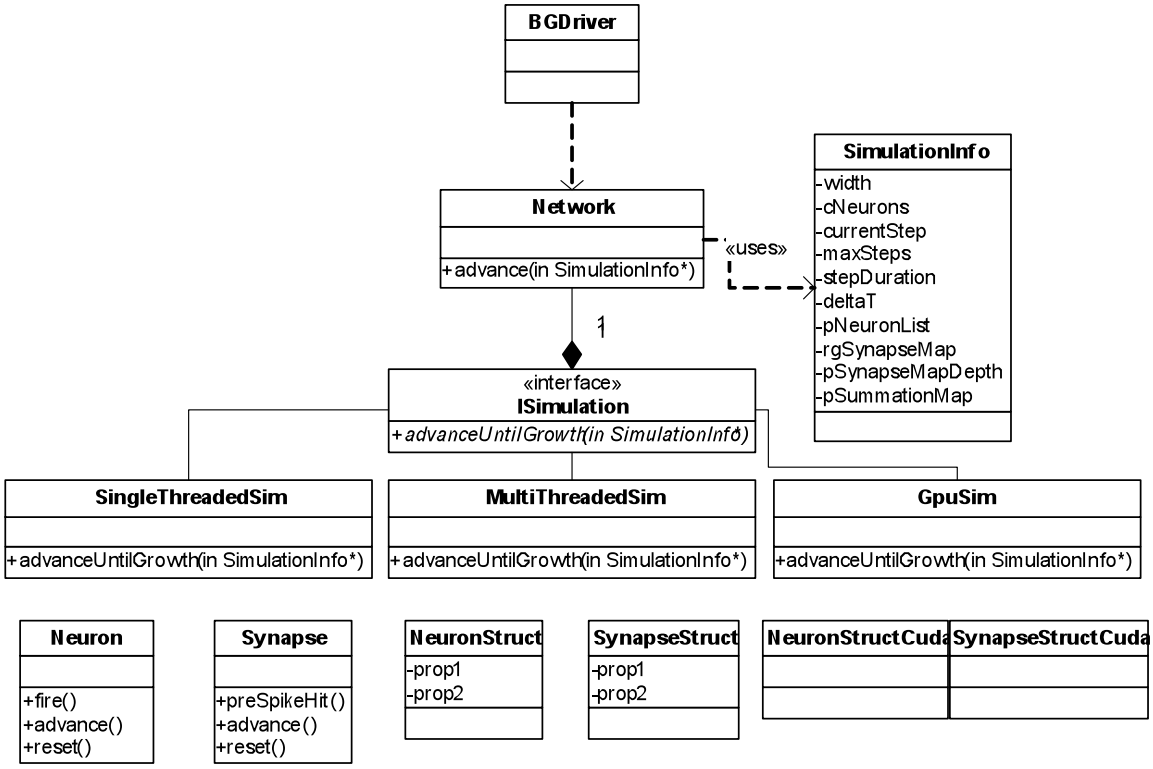

Figure 2.5: Simulator architecture. An abstract interface (*ISimulation*) was used to allow swapping of different underlying simulator implementations.

dates of an activity epoch were performed by *advanceUntilGrowth()* function, and *SingleThreadedSim*, *MultiThreadedSim* and *GpuSim* classes contain single-threaded, OpenMP, and GPU version of the function respectively.

Initial simulations consisted of networks of 100 neurons in a 10×10 arrangement, with each simulation taking around 20 hours on computers with 2-3 GHz microprocessors. Preliminary results from these simulations indicated that behavior was dominated by small network effects. Larger simulations, say 100×100 networks, would help distinguish between the effects of network size and inherent behavior. However, increasing network size has computational consequences that must be addressed: in the single-threaded form, a 60,000 seconds simulation of a 100×100 network would take at least 2,000 hours (83 days). From our preliminary research, we realized that parallelization using OpenMP could not feasibly lead to large enough speedups to make such simulations practical, and thus we turned to a GPU implementation.

Chapter 3

PRELIMINARY EXPERIMENTS

This chapter represents the results of preliminary experiments of a computational study of the interplay of individual neuron activity, cell culture development, and the network behavior with small networks (100 neurons in a 10×10 arrangement), which were experimented in 2007 [32].

3.1 Computer Implementation

We used CSIM (A Neural Circuit SIMulator) version 1.1 for the simulations [26]. For performance reasons, the original, Matlab-integrated code was pared down to a small core that was linked to a stand-alone C++ program to run on Linux, Windows, and Macintosh computers.

Simulations consisted of networks of 100 neurons in a 10×10 arrangement. Each simulation proceeded as a sequence of 100 second segments (epoch), during each of which the connectivity was kept constant. The average firing rate of each neuron during the preceding segment was used to adjust its neurite outgrowth, according to equation (2.6), for the next. Simulations were 300–600 segments long (30,000–60,000 seconds); while this represents a great speeding up of neurite outgrowth compared to the living preparation, numerical investigation indicated that this did not introduce instability in the simulation.

Initial simulations were performed with randomly chosen endogenously active and inhibitory neurons, but this produced enormous variations in results. It was hypothesized that this was due to the small size of the network and edge effects, and so the set of standardized layouts in Figure 3.1 was chosen to maximize spacing among these cells and reduce edge effects. This produced more consistent simulation results.

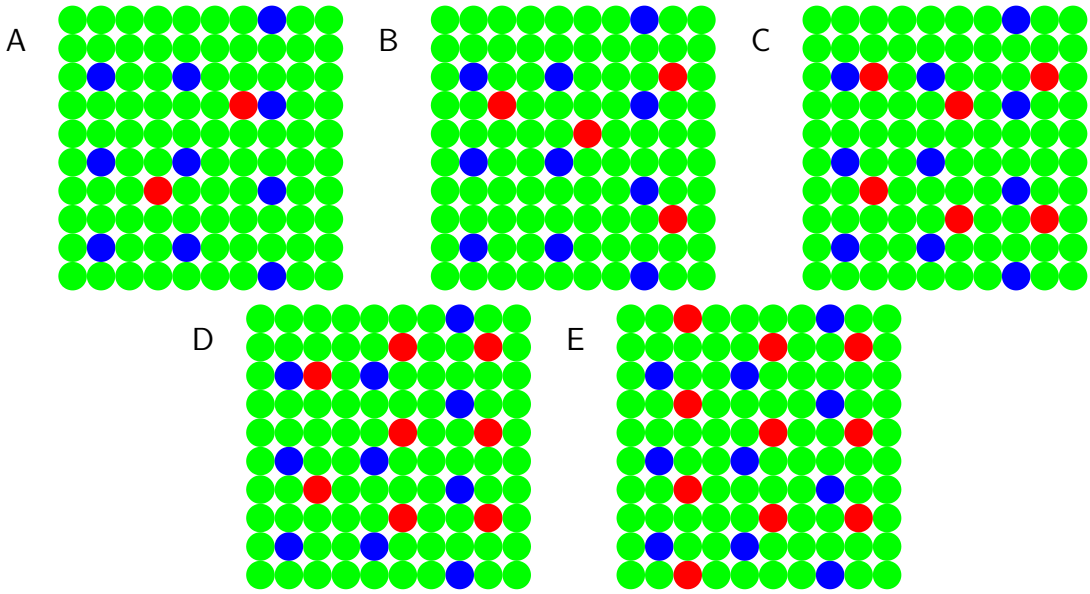

Figure 3.1: Layouts used for different numbers of endogenously active (EA, blue dots) and inhibitory (INH, red dots) neurons. (A) 10EA, 2INH, (B) 10EA, 4INH, (C) 10EA, 6INH, (D) 10EA, 8INH, and (E) 10EA, 10INH.

3.2 Analysis Methods

We choose two parameters (fraction of excitatory neurons and target firing rate) and observed network behaviors as these were varied. As metrics of the neural network behaviors, we calculated and plotted:

neuron firing rate To get an overview of the effects of the two parameters, average firing rate was computed for two windows of time: 20,000–25,000sec and 25,000–30,000sec. The ratio of average rate to target rate was plotted against the two parameters.

fraction of neurons with stable connectivity radii A neuron's radius of connectivity was judged to have stabilized at a constant value if it changed by less than around 15% during the final 10,000 seconds of the simulation. The fraction of neurons with stable radii was plotted versus the two simulation parameters.

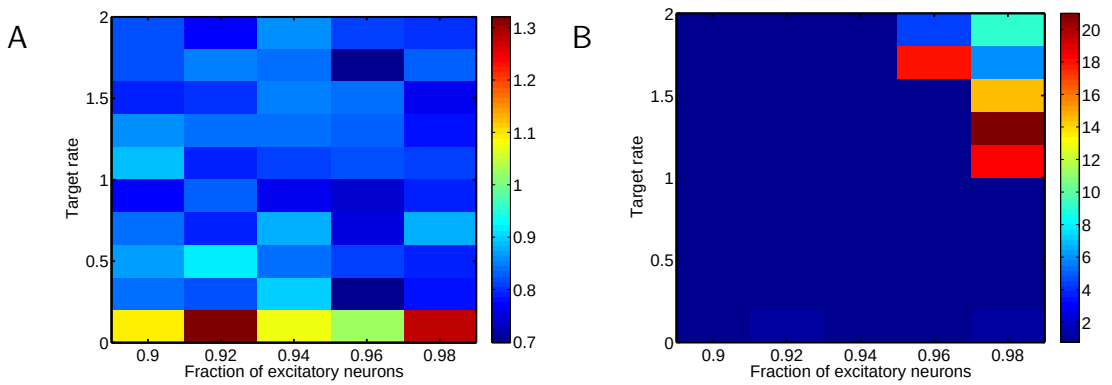

Figure 3.2: Firing rates normalized relative to target rate, for 20,000–25,000sec (A) and 25,000-30,000sec (B), plotted versus the two simulation parameters.

burstiness index A *burstiness index* [41] was computed by first calculating a spike count vs. time histogram for the entire network during the whole simulation period [1]. The fraction, f_{15}, of the total number of spikes contained by the 15% most populous bins was then normalized to produce the burstiness index, BI, as BI $= (f_{15} - 0.15)/0.85$. BI was plotted against simulation parameters.

In addition, detailed examination of single simulations involved plots of individual neuron's connectivity radii and firing rate versus time.

3.3 Results

Based on preliminary investigations of the effects of different model parameters on network behavior evolution, an initial set of 50 simulations was performed with target rates in the range 0.1–1.9 (inclusive, in 10 steps) and fraction of excitatory neurons 0.9–0.98 (inclusive, in five steps). Figure 3.2 shows normalized firing rates for two time ranges. These show that the lowest target rates were met early on (A). However, only the simulations with higher

[1]The simulation was 60,000 sec-long ,which 5 min window was used to calculate a burstiness index, where there were 300 1-sec-long time bins, and counted the number of spikes (entire network) in each bin. If a recording was burst dominated, a burstiness index would be close to 1. The burstiness index can be used to qualify the level of bursting in the assumption that bursts did not occupy $>$ 45 1-sec-long bins (15%) in a 5 min recording.

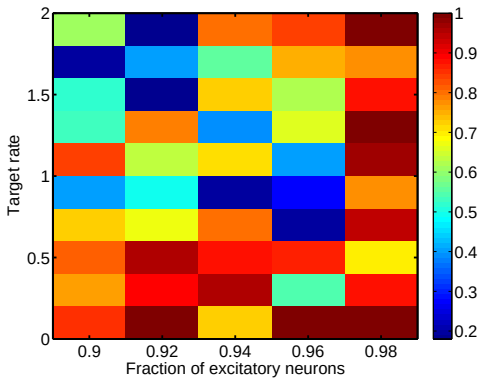

Figure 3.3: Fractions of neurons with stable connectivity radii as a function of the two simulation parameters.

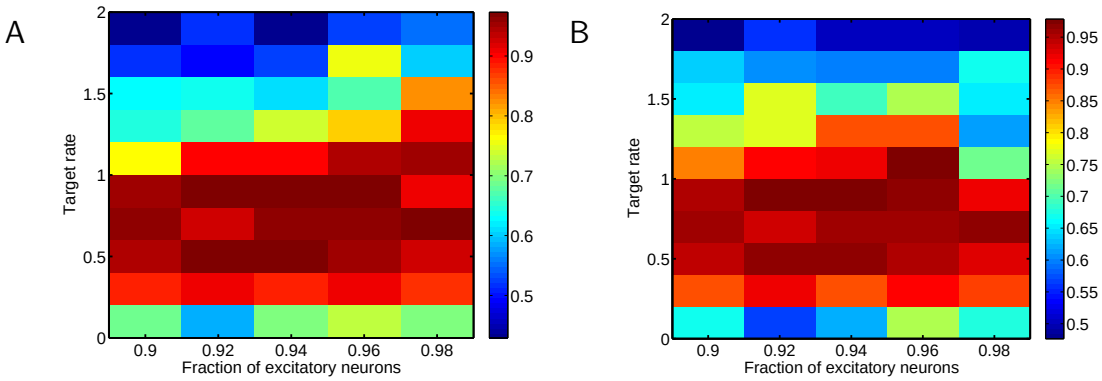

Figure 3.4: Burstiness index versus simulation parameters for 20,000-25,000sec (A) and 25,000–30,000sec (B).

target rates and fewer inhibitory neurons showed the great increase in firing rate that might be associated with bursting (B). Longer (60,000sec) simulations exhibited some bursting type of behaviors with the fraction of bursting as low as 0.9 for the higher target rates.

Figure 3.3 shows that a rather undemanding criterion for outgrowth stabilization was easily met by a majority of neurons in most simulations. More detailed examination indicated significant edge effects, with edge neurons and their neighbors having larger connectivity radii that are less likely to have stabilized.

Figure 3.4 indicates that the fraction of excitatory neurons in this range has only a

modest effect on burstiness index, and that moderate target rates produce the highest BI values. This apparent conflict with the previous observation of bursting at higher target rates can be explained by examining the detailed behavior of individual simulations.

Figure 3.5 shows detailed information for simulations with five sets of parameters: (target rate, fraction excitatory cells) = (0.1, 0.9), (0.1, 0.98), (0.9, 0.94), (1.9, 0.9), and (1.9, 0.98). These include the parameter extremes and a central value. In these cases, 60,000sec simulations were performed. Note that some of the simulations that weren't bursting at 30,000sec were bursting shortly thereafter, as evidenced by the (0.9, 0.94) one.

Nevertheless, this confirms that the low BI values for low target rates corresponds to non-bursting behaviors (and that the connectivity radii have not stabilized for cells that are not inhibitory or spontaneously active). Low BI values for high target rates are a possibly result of the very broad or frequent bursts. For the bursting behaviors, connectivity radii have stabilized, excepting small variations during bursting. In all simulations, connectivity radii for edge neurons are larger than others, due to them having fewer neighbors. Inhibitory neurons had moderate radii, while spontaneously active neurons had a wide range of different connectivities, likely due to the variability in their firing thresholds.

In either bursting or non-bursting behaviors, spontaneously active neurons tended to be the most active. This is not surprising, as their lowered thresholds would make them more excitable. The next higher firing rates belonged to the inhibitory cells, then non-edge cells, then edge cells.

In the results in figure 3.5, it seems possible that the mechanism for burst initiation and termination is the variation in connectivity. Figure 3.6 presents simulations in which connectivity was fixed either during bursts (A, C) or in between bursts (B, D). For the lower target rate/lower fraction of excitatory cells simulation, it does indeed seem that bursting is controlled by connectivity. However, for a higher rate/higher fraction of excitatory cells, bursting can continue even in the absence of variation in connectivity.

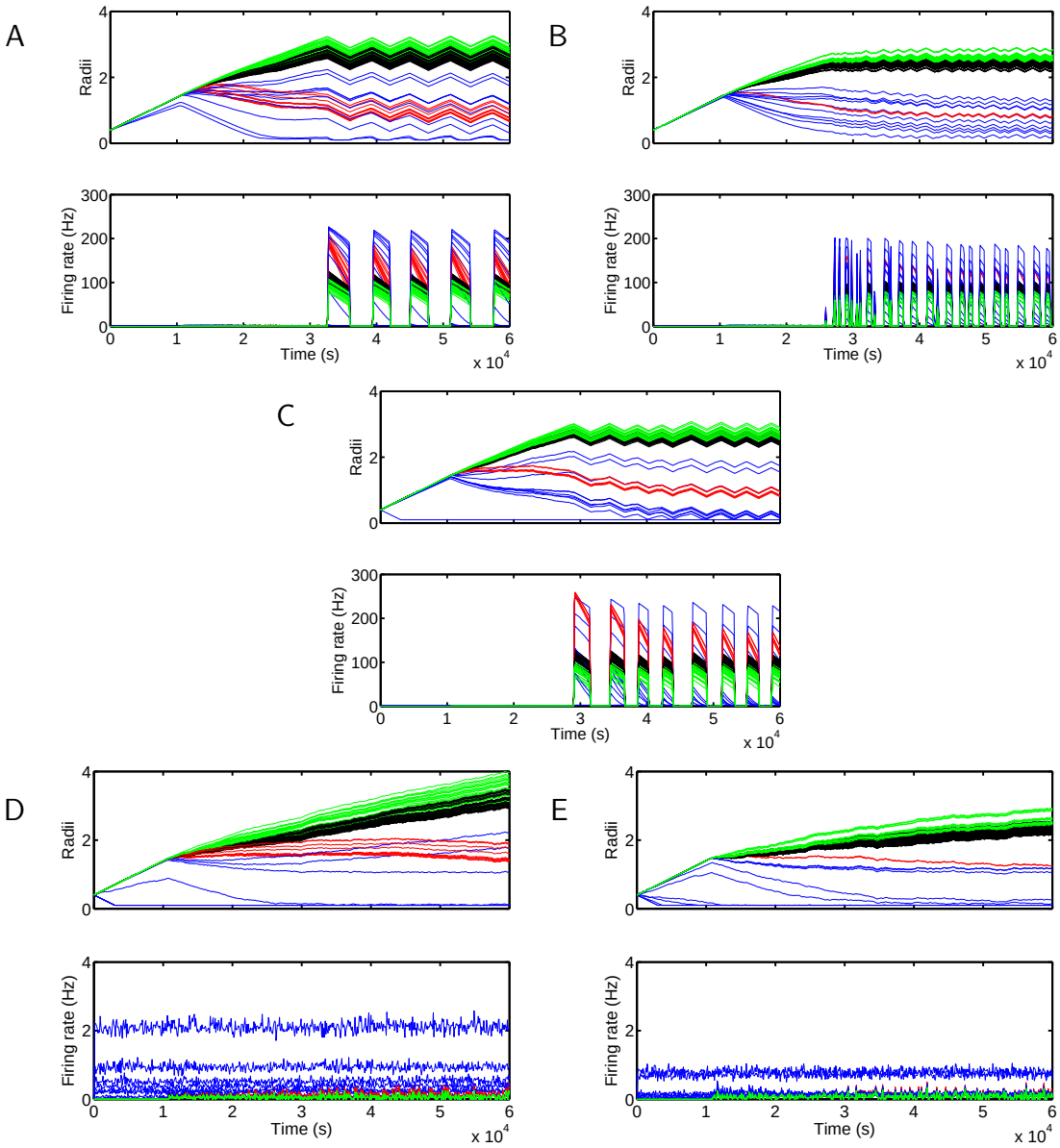

Figure 3.5: Detailed simulation results for simulations with parameters (target rate, fraction excitatory cells) of (1.9, 0.9) (A), (1.9, 0.98) (B), (0.9, 0.94) (C), (0.1, 0.9) (D), and (0.1, 0.98) (E). In all graphs, data for edge neurons are green, non-edge (and non-inhibitory, non-spontaneously active) neurons are black, inhibitory neurons are red, and spontaneously active neurons are blue.

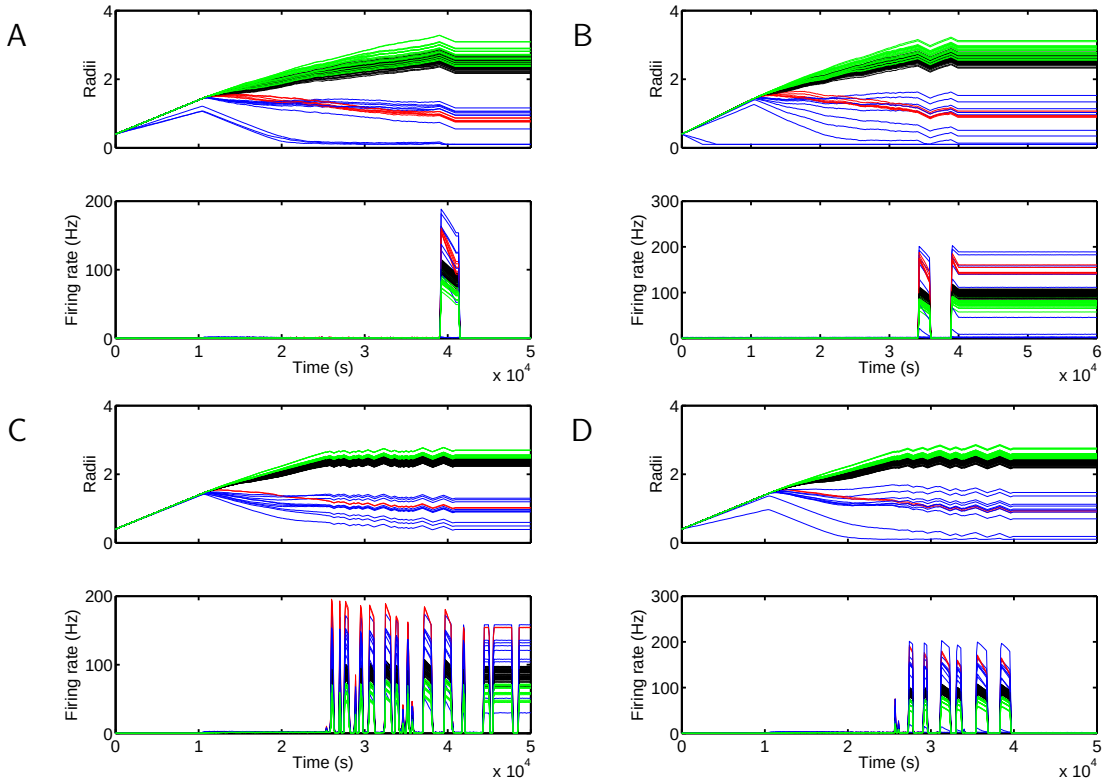

Figure 3.6: Effects of neurite outgrowth on bursting for simulations with parameters (target rate, fraction excitatory cells) of (0.9, 0.94) (A, B) and (1.9, 0.98) (C, D). Growth was "frozen" either during (A, C) or between (B, D) bursts.

3.4 Analysis

Bursting occurred with sufficiently small inhibition and high target firing rate. One might expect the latter to produce greater connectivity for every neuron (cells requiring more and stronger inputs to fire more rapidly), which in turn would be the mechanism for whole-culture bursting. However, in the low-target-rate, non-bursting simulations, such as figure 3.5(D, E), non-inhibitory, non-spontaneously active cells grow large connectivity radii. Spontaneously active cells, on the other hand, tend to have large connectivity radii in the bursting simulations. Presumably, lowering these cells autonomous firing rate would result in bursting at lower target rates.

In previous investigations of bursting with randomly connected networks [37], the model synapses' depression and facilitation were, neglecting the influence of noise, the mechanism underlying burst initiation and termination. Our preliminary results indicate that this is possibly the case under certain circumstances, but possibly not all. For some regions of parameter space, it may be the case that the mechanism is a hysteresis effect involving changing connectivity radii.

There are a number of possible reasons for this difference:

- There are a number of parameters that were set arbitrarily or not fully explored. For example, biologically realistic preparations might have a fraction of excitatory cells as low as 0.75 [37]. More significantly, in the current simulations connection weights were merely areas of overlap of connectivity circles, scaled down into the nA range, with no differential scaling based on type of synapse (i.e., inhibitory vs. excitatory).

- In the current simulations, only spontaneously active cells had any parameter variability; all other cells of a given type (inhibitory or excitatory) had identical parameters.

- However, the overriding issue here is the small network size, and we view any other changes as unlikely to have a significant effect on network activity unless this is

changed. Currently, edge effects are great (edge neurons' connectivity radii are always the greatest of all neurons and 36/100 of the cells are edge neurons) and there are small numbers of inhibitory and spontaneously active cells. The final networks are almost completely connected, even with final radii of only around 3. If the network size were increased to, say, 100x100, it seems the impact on the final connectivity radii would be minimal, but then each neuron would only connect to less than 10% of the network.

Increasing network size will have computational consequences that must be addressed: in its current form, a 60,000sec simulation of a 100x100 network would take at least 2,000 hours (83 days). The solution to this will be a parallel version of the simulator planned to take advantage of a UW Bothell computing grid, eventually with hundreds of computers.

There are also fundamental differences between the connectivity patterns generated by this model (perhaps most similar to radial basis functions) and many other models of cortex or recurrent networks (in which either network topology is irrelevant or a power law-type distribution is used that produces mostly local connections with a few long-range ones). It will be instructive to investigate the detailed correlation structure of inter- and intra-burst neuron firing.

Chapter 4

IMPLEMENTATION FOR GPU

This chapter presents the experience and results to implement the simulator into a GPU [19].

4.1 GPU Architecture and CUDA

CUDA (Computer Unified Device Architecture) is a parallel computing architecture developed by NVIDIA, which enables GPU programming through a C-like programming language. In the CUDA model, a code executed on GPU is written using C for CUDA and called by host CPU as a function. Such a function is called a kernel. When the kernel function is called, the CUDA runtime generates a large number of threads to exploit data parallelism. All threads that are generated by a kernel function are collectively called a grid. A grid consists of blocks, which are the highest unit of CUDA scheduling, and the blocks are assigned to streaming multiprocessors (SMs) and executed in any order relative to each other. Once a block is assigned to a SM, it is further divided into 32-threads units called warps. The warp is the unit of thread scheduling in SMs, where every thread of a warp executes the same instruction. NVIDIA describes such an execution model as Single Instruction Multiple Thread (SIMT), a variant on SIMD [20].

The SM contains Stream Processors (SPs), arithmetic units (SFU — a Special Function Unit, and FMAD or DPU — a 64 bit Fused Multiply-Add Units), and other private resources such as shared memory and registers (Figure 4.1). The NVIDIA Tesla C1060 board that we used has 30 SMs and 4 GB global memory, which is shared between SMs. Table 4.1 shows the hardware parameters for the Tesla C1060 board [1].

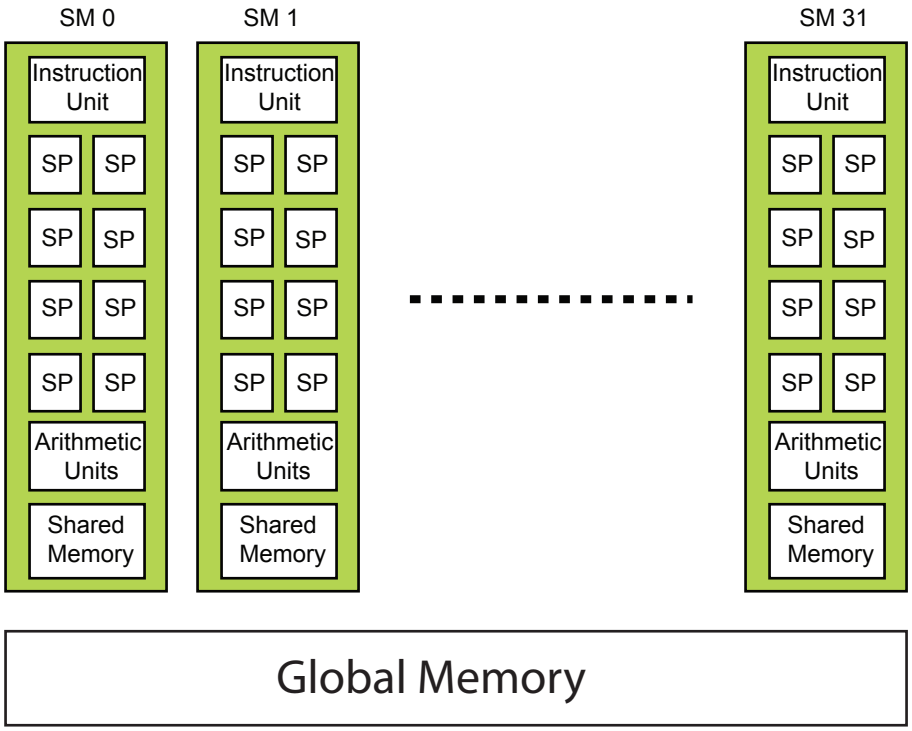

Figure 4.1: CUDA C1060 GPU architecture.

Table 4.1: C1060 parameters.

# of SPs/SM	8
# of SFUs/SM	2
# of DPUs/SM	1
Registers/SM	16 K
Shared memory/SM	16 KB
Maximum active warps/SM	32
Maximum active threads/SM	1024
Maximum active blocks/SM	8
Maximum threads/block	512
Threads/warp	32
Constant cache	8 KB
Texture cache	6–8 KB
# of SMs	30
Core clock	1.296 GHz
Total constant memory	64 KB
Global memory	4 GB
Memory clock	800 MHz
Memory latency	400–600 clocks

4.2 Match between Algorithm Structure and GPU Architecture

As described in figure 2.4, the simulation algorithm consisted of updates to neuron and synapse state that occurred every 0.1ms time step. For a 100×100 network, the number of synapses, determined by neuron connectivity radius (R_i), was much larger than the 10,000 neurons. Preliminary simulations indicated that the common equilibrium value of neuron radius would yield a ratio between the number of neurons and the number of synapses of $1 : 46.4$. Therefore, the states of 10,000 neurons and about 464,000 synapses were required to be updated every time step, using the exponential Euler method for numerical integration of equations (2.1)–(2.5). Consequently, a 60,000s simulation time of a 100×100 arrangement represents 6×10^8 time steps, or approximately 2.8×10^{14} state updates.

The massive parallelism of a GPU seemed to suit the inherent parallelism of the simulation algorithm very well. In the single threaded implementation, the calculation for each neuron and synapse was done by two inner loops, one for neurons and one for synapses

(Figure 2.4). A calculation for each neuron or synapse is independent of other neurons or synapses, so those calculations can be done in parallel. Therefore, the naive conversion of the loops into GPU kernel functions — *advanceNeuronDevice()* and *advanceSynapseDevice()* — was straightforward. Each thread of the kernel functions implemented one iteration of the original loops of the single threaded C++ code. Within the kernels, each thread could identify its neuron or synapse using its *blockIdx* and *threadIdx* values [1].

4.3 Related Work

Earlier works on large-scale Spiking Neural Network (SNN) simulations were mostly conducted on distributed computers, or on dedicated hardware architectures. Though such approaches gave us better performance, limited programmability and high-cost were drawbacks. However, the advent of GPU computing ameliorated the situation with its inherit massively parallel computing capability, a specific computational libraries such as CUDA, and lower cost.

Tiesel et al. [34], created a SNN simulator for integrate-and-fire neurons without axonal delays or synaptic learning. Instead of using CUDA architecture, they used the OpenGL graphics API to exploit the GPU hardware. They gave qualitative and quantitative measures to compare two implementations (CPU and GPU), and showed the GPU implementation outperformed the CPU version.

Nageswaran et al. [25], developed a SNN simulator that ran on a single GPU utilizing the CUDA architecture. Their model includes Izhikevich spiking neurons, detailed models of synaptic plasticity and variable axonal delay. They provide a high-level programming interface similar to the PyNN (a simulator-independent language for building neural network models). Performance testing indicated that the GPU implementation (on NVIDIA GTX-280 with 1 GB of memory) achieved a speed-up of 26 over a CPU version for the simulation

[1]The *blockIdx* and *threadIdx* are CUDA predefined variable that can be accessed in kernel functions. These variables return the coordinates of the thread so that each thread is able to determine the area of data to work on.

of 100K neurons with 50 Million synaptic connections.

Taha et al. [11], created a simulator for image recognition for the modeling for both Hodgkin-Huxley and Izhikevich neuron models utilizing the CUDA API. Performance testing showed the fastest GPGPU, the Tesla S1070, provided a speedup of 5.6 and 84.4 over CPU versions, a qudcore 2.67 GHz Xeon processor, for the Izhikevich and the Hodgkin-Huxley models, respectively.

4.4 GPU Implementation

We implemented the simulator on a GPU and performed simulations with 10,000 neurons and 464,000 synapses in a 100×100 arrangement. We performed one 100s activity epoch, including growth update at its conclusion, on each revision of the GPU implementation, which includes the whole execution path and allows us to evaluate the performance of each component. We set all starting neuron radii to 2.0 because that was a common equilibrium value of neuron radius in previous simulations. This allowed for evaluation of simulator performance at that state. Simulations were run on a 2.8GHz AMD Phenom II X4 920 processor with 512KB L2 cache and an NVIDIA Tesla C1060 device. Execution times of kernel functions were determined using GPU timers. The CUDA event API call *cudaEventRecord()* was placed before and after each kernel function to record time stamps and the *cudaEventElapsedTime()* API call returned elapsed time. Elapsed time was accumulated under iteration and resulted in cumulative execution times.

Referring to the simulator architecture described in figure 2.5, we implemented a *GpuSim* sub-class of *ISimulation* with a GPU version of the *advancedUntilGrowth()* method, where the process of one activity epoch was performed. For each time step, the state update of each neuron and each synapse was performed by the kernel functions *advanceNeuronDevice()* and *advanceSynapseDevice()* concurrently, in contrast to their sequential nature in the singe-threaded version. Figure 4.2 compares execution times of the single-threaded version and GPU version. As seen in the figure, this naive GPU implementation achieved a speedup about 2.4 times against the single thread version — not too dissimilar to the multi-core

Figure 4.2: Run times of the naive GPU implementation vs. single thread.

OpenMP version.

One inherent bottleneck in the model's structure lies in the convergence of synapse input onto each individual neuron. This occurs at each neuron's *summation point*, where synapse responses are summed to produce a net input for the connected neuron (Figure 4.3). These summation points might be written into by more than one synapse, modified simultaneously by *advanceSynapseDevice()* kernel function. Therefore, we used atomic operations — read-modify-write without interruption from other threads — to avoid conflict. However, these atomic operations were very expensive, so the *advanceSynapseDevice()* kernel function occupied the majority of the execution time. To eliminate the atomic locks, the code was modified as follows:

1. The summation operation was separated from the kernel function *advanceSynapseDevice()*.

2. An *inverse map* was created. This was a table indexed by neuron number that mapped to the synapses (by synapse ID) that provided input to that neuron.

3. A new kernel function, *calcSummationMap()*, was implemented. One thread of this executed for each neuron, using the inverse map to sum up the responses of the

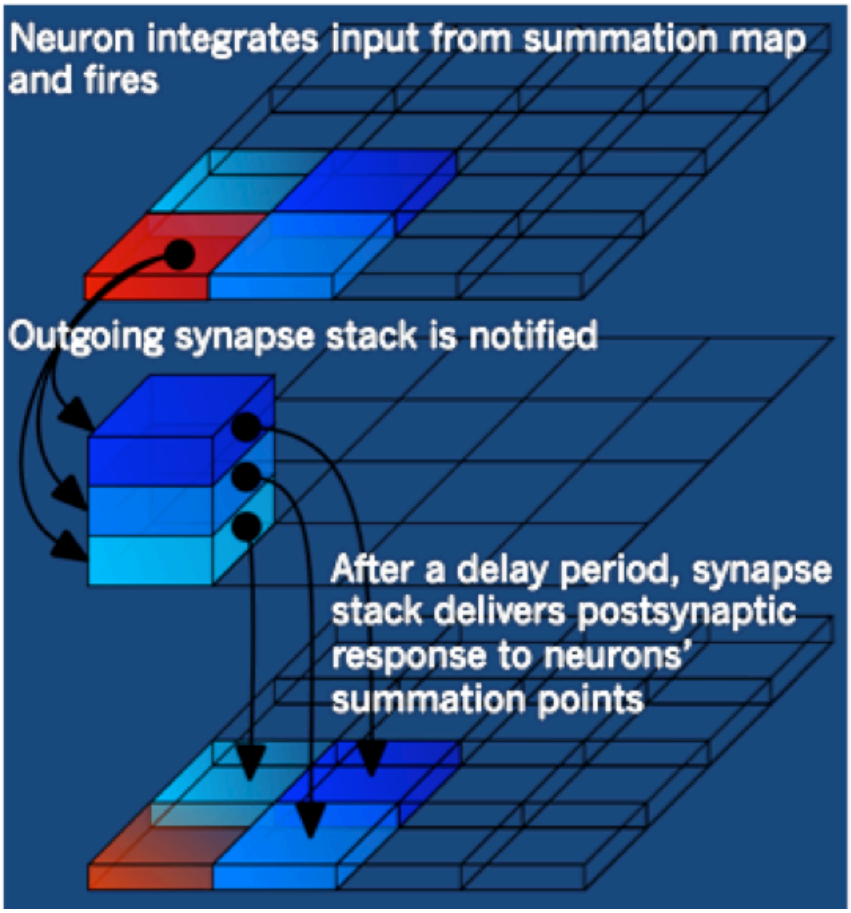

Figure 4.3: Structure of neuron's summation point.

synapses that provided input to that neuron.

Next, we applied various performance-enhancement techniques to the kernel functions, which included:

1. Reduced global memory access by using registers.

2. Combined kernel functions. The naive implementation mirrored the single-threaded version by using device functions, called from the GPU kernels, that matched subroutines to handle such things as spike arrivals to synapses and neuron spike generation.

These were merged into the *advanceNeuronDevice()* function to eliminate the significant overhead associated with the function calls.

3. Used constant memory. Synapse constants, U, D, F, were moved into constant memory.

4. Each synapse object had a delay queue: a queue to store incoming spike events scheduled to arrive in the future, with its own pointer to indicate current position in the queue. We made the delay queue pointer global to all synapses; therefore, each synapse didn't need to update the pointer.

5. The neuron and synapse data were originally stored as arrays of *LifNeuron_struct* and *DynamicSpikingSynapse_struct*, respectively. This mirrored the object-oriented nature of the single-threaded simulator. However, this configuration prevented favorable global memory access patterns because the GPU hardware achieves high global memory access efficiency when all threads in a warp access consecutive global memory locations (coalesced memory access). To correct this, the arrays of neuron and the synapse data structures were reorganized into global neuron and synapse structures containing multiple, homogeneous arrays of data.

6. Neuron network connectivity update is done once each 100 seconds of simulated time (once each activity epoch). Therefore, the ratio of connectivity update to simulation time step is $1 : 1,000,000$ (remember that the simulation time step is 0.1ms). However, the execution time of adjusting synapses, which is the major portion of *updateNetwork()*, still took about 9.4% of total execution time. This was because updating was done by a function on the host CPU, so there was room for performance improvement by moving that process to the GPU. We refactored the simulator architecture so that the *ISimulation* interface included an *updateNetwork()* method, and implemented a GPU version of this method (Figure 4.4). Updating connectivity on the GPU also

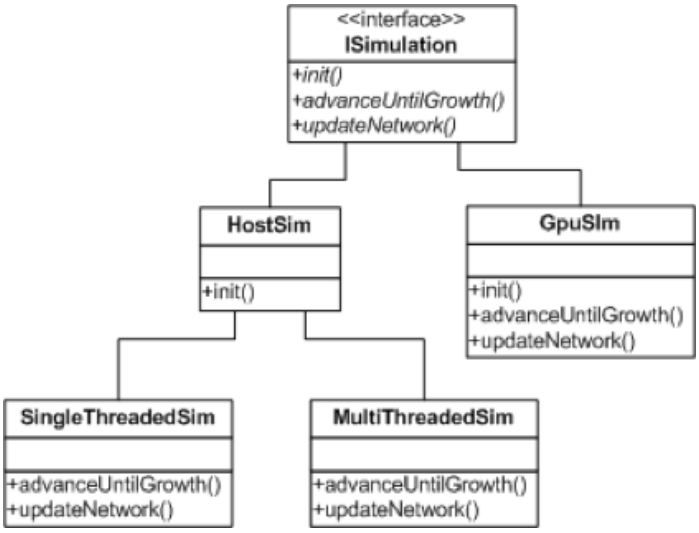

Figure 4.4: *ISimulation* interface that includes single-threaded, multi-threaded, and GPU implementations.

eliminated the need to copy neuron and synapse data between host and GPU memory each activity epoch. As a result, the only communication necessary between host and GPU was initialization, kernel sequencing, once-per-epoch communication of results from GPU to host, and final communication of simulation state from GPU to host (to enable simulation resumption at a later time).

7. A random number generator was used to produce each neuron's noise current, I_{noise}. Because a set of random numbers is needed for each time step, random number generation occupied about 55% of the total execution time. Random numbers were originally generated by a single thread on the host CPU; we implemented a GPU version of the random number generator, using a Mersenne Twister algorithm [24].

The successive improvements brought by these optimizations are shown in figure 4.5. The final combination of performance tuning techniques produced an approximately 23 times speedup compared with the single threaded version. The final simulator version runs almost entirely on the GPU, requiring 4–5 days to complete a 60,000s simulation.

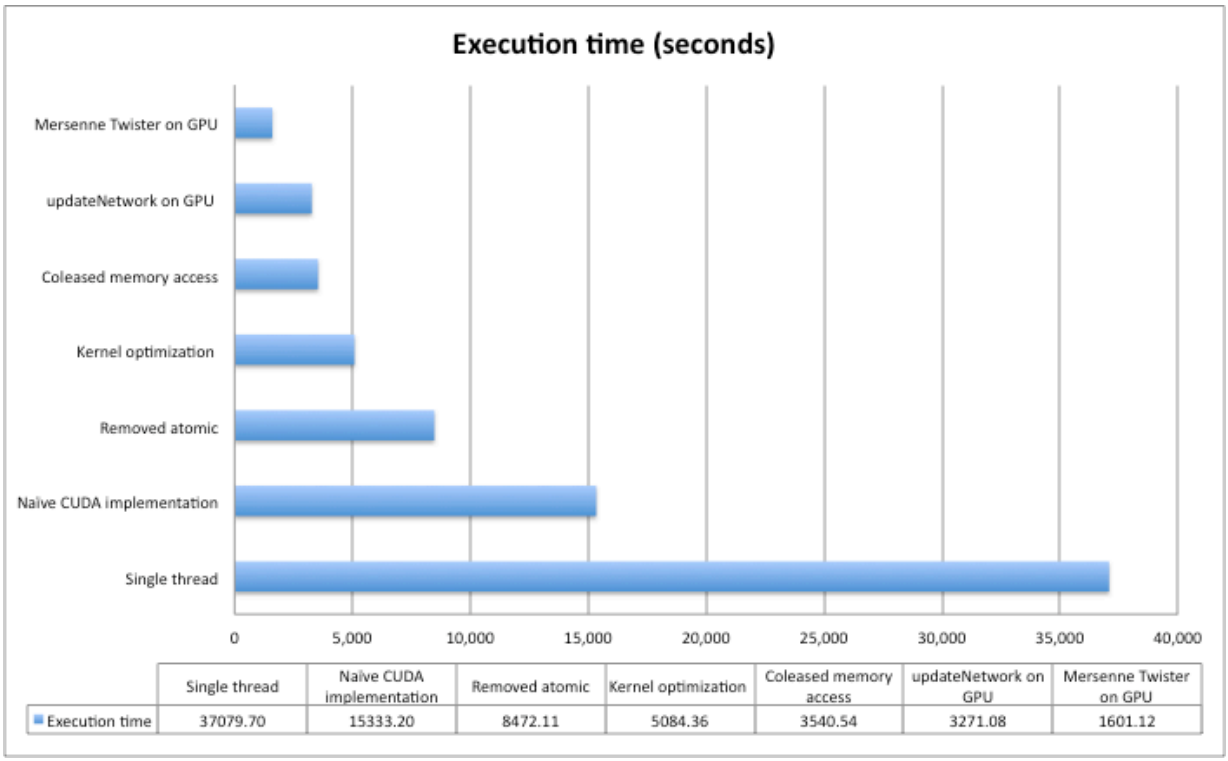

Figure 4.5: Summary of speedups produced by different techniques. Execution times of a single activity epoch, plus network connectivity update, are shown in comparison to the single-threaded CPU version (bottom). From bottom to top, times are for the naive GPU implementation and successively optimized versions.

4.5 Performance Results

Three of the kernels comprising the final GPU implementation account for most of its execution time: *advanceNeuronDevice()*, *advanceSynapseDevice()*, and *calcSummationMap()*. Detailed performance data was collected and analyzed for these to enable a more precise understanding of which limitations were inherent in the algorithmic structure of this type of simulator, which were peculiar to the particular simulator, and which might be amenable to further optimization. Table 4.2 presents basic configuration information and execution time for these three major kernel functions. We also examined *multiprocessor occupancy*, *thread divergence*, *effective memory bandwidth*, and *effective MIPS*.

4.5.1 Multiprocessor Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. The number of active warps per streaming multiprocessor (SM) is limited by several CUDA hardware resources, such as register availability [2]. Table 4.2 indicates occupancy for the three major kernels. Because of their larger use of register memory, *advanceNeuronDevice()* and *advanceSynapseDevice()* were not able to achieve 100% occupancy.

4.5.2 Thread Divergence

Thread divergence occurs when threads in the same warp follow different paths of control flow, such may occur in if-then-else or for-loop constructs. When divergent, parts of a warp take different execution paths sequentially — the threads for one pathway are executed (until the divergent paths rejoin), then those for the next, etc., until all pathways have been processed. This has an important effect on performance. Table 4.3 summarizes thread divergence data for the three major GPU kernels.

In table 4.3, the divergent paths 1, 2, and 3 of the *advanceNeuronDevice()* kernel function correspond to the three neuron states (refractory, firing, and normal), respectively. Each

Table 4.2: Measured performance of three kernels. Basic data and cumulative execution time for each kernel function.

Kernel function	***AdvanceNeuronDevice***	***AdvanceSynapseDevice***	***calcSummationMap***
Threads per block	256	256	256
Registers per thread	28	30	15
Physical max warps/SM	32	32	32
# of warps/SM	16	16	32
Occupancy	50%	50%	100%
# of iteration	1,000,000	1,000,000	1,000,000
# of threads	10,000	464,108	10,000
Cumulative execution time (ms)	162,631	635,624	529,130

Table 4.3: Measured performance of three kernels. Thread divergence data.

Kernel function	***AdvanceNeuronDevice***	***AdvanceSynapseDevice***	***calcSummationMap***
# of paths	3	2	1
Probability of divergent path 1	0.058	0.002	1.000
Probability of divergent path 2	0.002	0.998	0.000
Probability of divergent path 3	0.940	0.000	0.000

probability was calculated as follows.

From simulation data, the average firing rate of a single neuron was 19.4085Hz; in other words, on average each neuron fired that many times per second. An excitatory neuron's refractory period was 3ms; an inhibitory neuron's was 2ms. The percentages of excitatory and inhibitory neurons was 98% and 2%, respectively. Therefore the average refractory period (for each time it fired) was 2.98 milliseconds. Firing events themselves took one simulation step, 0.1ms. Thus, the average probabilities of a thread taking each of the three paths was:

$$P(\text{refractory}) = p_1 = (19.4085)(2.98)/1,000 \approx 0.058 \tag{4.1}$$

$$P(\text{firing}) = p_2 = 19.4085/10,000 \approx 0.002 \tag{4.2}$$

$$P(\text{normal}) = p_3 = 1 - (0.058 + 0.002) = 0.94 \tag{4.3}$$

The divergence paths 1 and 2 of the *advanceSynapseDevice()* kernel function correspond to the states of a synapse (firing and normal), respectively, and each probability was calculated from the average firing rate in a manner similar to that for the neurons. The *calcSummationMap()* kernel was not divergent.

4.5.3 Effective Memory Bandwidth

Table 4.4 shows the number of read/written bytes and effective bandwidth of each kernel function. The average number of bytes read/written, $\overline{B}_{\text{rw}}$, was calculated as:

$$\overline{B}_{\text{rw}} = \sum_{i=1}^{D} p_i B_{\text{rw,i}} \tag{4.4}$$

where, p_i was the probability of divergence path i, D the total number of divergent paths for that kernel, and $B_{\text{rw,i}}$ the number of bytes read/write in divergent path i. The effective bandwidth was:

$$B_e = \frac{\overline{B}_{\text{rw}} n_{\text{ite}} n_{\text{thread}}}{t_{\text{cum}}} \tag{4.5}$$

Table 4.4: Measured performance of three kernels. Number of bytes read and written is show for each kernel's divergent pathways; probabilities of each pathway from table 4.3 was used to compute average number of bytes and effective memory bandwidth.

Kernel functions	***AdvanceNeuronDevice***	***AdvanceSynapseDevice***	***calcSummationMap***
# bytes read in path 1	36	113	588
# bytes written in path 1	16	44	8
# bytes read in path 2	73	25	0
# bytes written in path 2	37	12	0
# bytes read in path 3	80	0	0
# bytes written in path 3	16	0	0
Average # read/written bytes	93	37	596
Effective bandwidth (GB/s)	5.7	27.2	11.3

where, n_{ite} was the number of iterations, n_{thread} was the number of threads, and t_{cum} was the cumulative execution time.

4.5.4 Effective MIPS

Table 4.5 shows the number of instructions and effective MIPs of each kernel function. The average number of instructions, $\overline{I}$, was calculated in a manner analogous to that for $\overline{B}_{\text{rw}}$ in equation (4.4). The effective MIPS, $MIPS_e$, for each kernel was calculated as:

$$MIPS_e = \frac{\overline{I} n_{\text{ite}} n_{\text{thread}}}{t_{\text{cum}}} \tag{4.6}$$

4.6 Performance Analysis

Theoretical peak performance of a device is a combination of memory bandwidth and computing resources. The Tesla C1060 is theoretically capable of 933.12GFLOPs (single precision) of processing performance and 102.4GB/s bandwidth, which can be calculated from its hardware specifications [1]. For example, the Tesla C1060 uses DDR (double data rate) RAM with a memory clock rate of 800 MHz and a 512-bit wide memory interface, so the peak theoretical memory bandwidth is:

$$B_p = \frac{800 \times 10^6 \times \left(\frac{512}{8}\right) \times 2}{10^9} = 102.4\text{GB/s} \tag{4.7}$$

The maximum throughput of a streaming processor (SP) is one instruction per clock when the pipeline of the SP is full [27]. Therefore, the peak theoretical MIPS is:

$$MIPS_p = \frac{n_{\text{mp}} n_{\text{sp}} r}{10^6} = \frac{30 \times 8 \times 1.296}{10^6} = 311,040\text{MIPS} \tag{4.8}$$

where n_{mp} is the number of multiprocessors, n_{sp} is the number of streaming processors per multiprocessor, and r is the clock rate in GHz.

Because the C1060 card can dual-issue a multiply concurrent with a multiply-add, giving

Table 4.5: Measured performance of three kernels. Number of instructions calculated for each divergent pathway, average number of instructions executed by each thread of the three major kernels, and effective MIPS achieved by each kernel.

Kernel function	***AdvanceNeuronDevice***	***AdvanceSynapseDevice***	***calcSummationMap***
# instructions in path 1	57	472	963
# instructions in path 2	166	70	0
# instructions in path 3	92	0	0
Average # instructions	90.1	70.8	963.0
Effective MIPS	5,541.3	51,698.3	18,199.7

a 50% boost to the theoretical max, the peak theoretical FLOPS (single precision) speed is:

$$FLOPS_p = \frac{2 \times 1.5 \times MIPS_p}{10^3} = 932.12\text{GFLOPS} \tag{4.9}$$

The ratio of effective MIPS to theoretical peak MIPS in table 4.6 reflects the utilization efficiency of the streaming multiprocessor (SM) execution units. When all the execution units in the CUDA device are fully utilized, the device can achieve its peak performance. There are some factors that limit execution unit utilization efficiency:

1. There need to be enough instructions between memory accesses, and enough warps issuing those instructions, to hide the long memory access latency, and therefore to maximize the utilization of execution units. When this is not the case, execution units sit idle while awaiting memory access completion. Therefore, the global memory to instruction cycle ratio measures the degree of memory access frequency. A lower ratio will result in higher utilization efficiency.

2. Occupancy determines the number of warps per streaming multiprocessor (SM), and therefore the number of threads that can issue instructions "simultaneously". Low occupancy decreases utilization efficiency.

3. Thread divergence, where the execution of threads in a warp takes different paths sequentially, lowers utilization efficiency.

4. The kernel functions *advanceNeuronDevice()* and *calcSummationMap()* don't utilize all streaming multiprocessors (SMs) due to network size. The kernel function *advanceNeuronDevice()* only uses 20 streaming multiprocessors (SMs) out of 30 SMs (2/3 utilization ratio), and the kernel function *calcSummationMap()* only uses 10

Table 4.6: Percentage ratio of effective performance to peak performance.

Kernel function	B_e/B_p	$MIPS_e/MIPS_p$
AdvanceNeuronDevice	5.6%	1.8%
AdvanceSynapseDevice	26.6%	16.6%
calcSummationMap	11.0%	5.9%

streaming multiprocessors (1/3 utilization ratio). [2]

5. Furthermore, there are some other factors that seem to lower actual utilization efficiency more than our analysis above:

   (a) We generated PTX files and counted the number of instructions in the assembly level code. However, the PTX code is not a sufficiently precise representation of the actual machine instructions executed on the GPU, as it does not precisely correspond to its actual machine instructions.

   (b) Some instructions are executed by special function units (SFUs) or double-precision units (DPUs), which are lower throughput devices than SPs.

A number of these conclusions are relevant to any simulation of this type (networks of simple interacting components with behavior simulated via iteration in time). In particular, we can consider occupancy, instruction mix, and serial dependency.

The neuron and synapse models simulated here are not terribly complex when compared to other neural simulations. In particular, the leaky integrator neuron model is just about the simplest that one might use. This would seem to suggest that large numbers

[2]The number of SMs used by a kernel function is obtained by the following formula:

$$\text{(Warps per block)} = \text{(Threads per block)}/\text{(Threads per warp)} \tag{4.10}$$
$$\text{(Blocks per SM)} = \text{(Warps per SM)}/\text{(Warps per block)} \tag{4.11}$$
$$\text{(\# of blocks)} = \text{(\# of threads)}/\text{(Threads per block)} \tag{4.12}$$
$$\text{(\# of SMs)} = \text{(\# of blocks)}/\text{(Blocks per SM)} \tag{4.13}$$

Threads per block, *warps per SM*, and *# of threads* of each kernel function are obtained by table 4.2, and *threads per warp* is obtained by C1060 parameters table 4.1.

of neurons could be simulated at once, thereby increasing the number of threads and, as a result, increasing utilization efficiency. However, despite this relative model simplicity, register usage by *advanceNeuronDevice()* and *advanceSynapseDevice()* still precludes 100% occupancy due to resource limits.

The simplicity of these models leads to a low ratio of computation to control and communications. This prevents full utilization of highly pipelined SP execution units and causes SPs to sit idle while threads wait for memory accesses to complete. One common approach to dealing with this would be to make each thread responsible for more than one neuron or synapse. However, this would increase register usage by each thread, resulting in decreased occupancy. Reduction in number of threads would also decrease occupancy. Therefore, it seems likely that this would be a self-defeating approach.

The computational complexity of this class of algorithms derives both from the large number of objects (neurons, synapses) to be simulated and the large number of iterations (here hundreds of millions) to be performed. The former is amenable to parallelization speedup. However, because neural simulations stereotypically involve very simple computational elements, the latter imposes an upper limit on overall performance.

To sum up, despite convenient libraries that enable rapid development of GPU-enabled code, detailed knowledge of GPU architecture is still essential for achieving maximum performance. One important concept is utilization efficiency: making all execution units as busy as possible, and many performance-improving techniques of GPU programming are related to this idea. As seen in the table 4.6, the utilization efficiency of each kernel function varies widely. It is heavily dependent on the nature of the function, such that memory bound functions may show lower utilization efficiency. However, understanding the detailed architecture of the GPU and focusing on utilization efficiency does lead to distinctly better solutions.

Chapter 5

EXPERIMENTS WITH LARGE NETWORKS

This chapter presents the results of a computational study of the interplay of individual neuron activity, cell culture development, and network behavior in larger networks (10,000 neurons in 100×100 arrangements) with the simulator executing on a GPU.

5.1 Computer Implementation

We used the GPU version of the simulator described in the chapter "IMPLEMENTATION FOR GPU". Simulations were run on NVIDIA Tesla C1060 or Tesla M2050 devices.

Simulations consisted of networks of 10,000 neurons in 100×100 arrangements simulated for 600 epochs (60,000 sec).

We used the set of standardized layouts of endogenously active and inhibitory neurons presented in Figure 5.1, repeating them to fill the 100×100 arrangements.

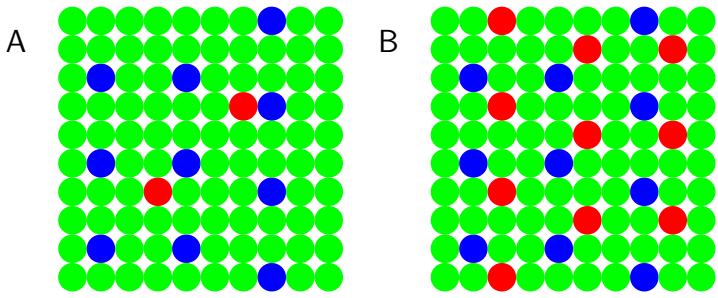

Figure 5.1: Layouts used for different numbers of endogenously active (EA, blue dots) and inhibitory (INH, red dots) neurons. (A) 10%EA, 2%INH, (B) 10%EA, 10%INH.

5.2 Analysis Methods

5.2.1 Preliminary Analysis

As in the case of the preliminary experiments, we chose two parameters (fraction of excitatory neurons and target firing rate) and observed network behaviors as these were varied. We ran 60,000 second simulations with the set of parameters (target rate, fraction excitatory cells) = (0.1, 0.9), (1.0, 0.9), (1.9, 0.9), (0.1, 0.98), (1.0, 0.98), and (1.9, 0.98). [1] We began with the same analysis methods as those of the preliminary experiments. Figures 5.2–5.7 show results of those simulations.

As seen in those results, final firing rates (B–F, lower plots) converged on the target rates except for spontaneously active neurons with firing rates above the target rate. Those neurons did not slow down, but slower firing spontaneously active neurons' firing rates were sped up. Connectivity radii for edge and corner neurons (B and D lower plots, respectively) were larger than others, probably due to them having fewer neighbors. Inhibitory neurons (E, lower plots) had moderate radii, while spontaneously active neurons (C, lower plots) had a wide range of different connectivities, likely due to the higher variability in their firing thresholds. These results could be expected from the results in preliminary experiments. However, none of these simulations seemed to exhibit any of the bursting behaviors observed in the preliminary experiments (Figure 3.5) — a big difference from the results of small networks.

Though bursting behaviors were not apparent in these plots, plots of burstiness index (A in the figures) (which used 1 sec long bins as described in the previous section) suggested that firing patterns were not stationary. Therefore, we changed the scale of time resolution from 100 sec time bins (1 epoch) to 10 msec time bins to investigate detailed firing patterns. We accumulated spikes of all neurons in each 10 msec time bin and recorded the total network

[1] Some simulations could not complete 60,000 sec simulation period. In these simulations, the number of synapses per neuron or the maximum firing rate per neuron exceeded the maximum size that could be allocated in global memory of GPU, so simulations aborted.

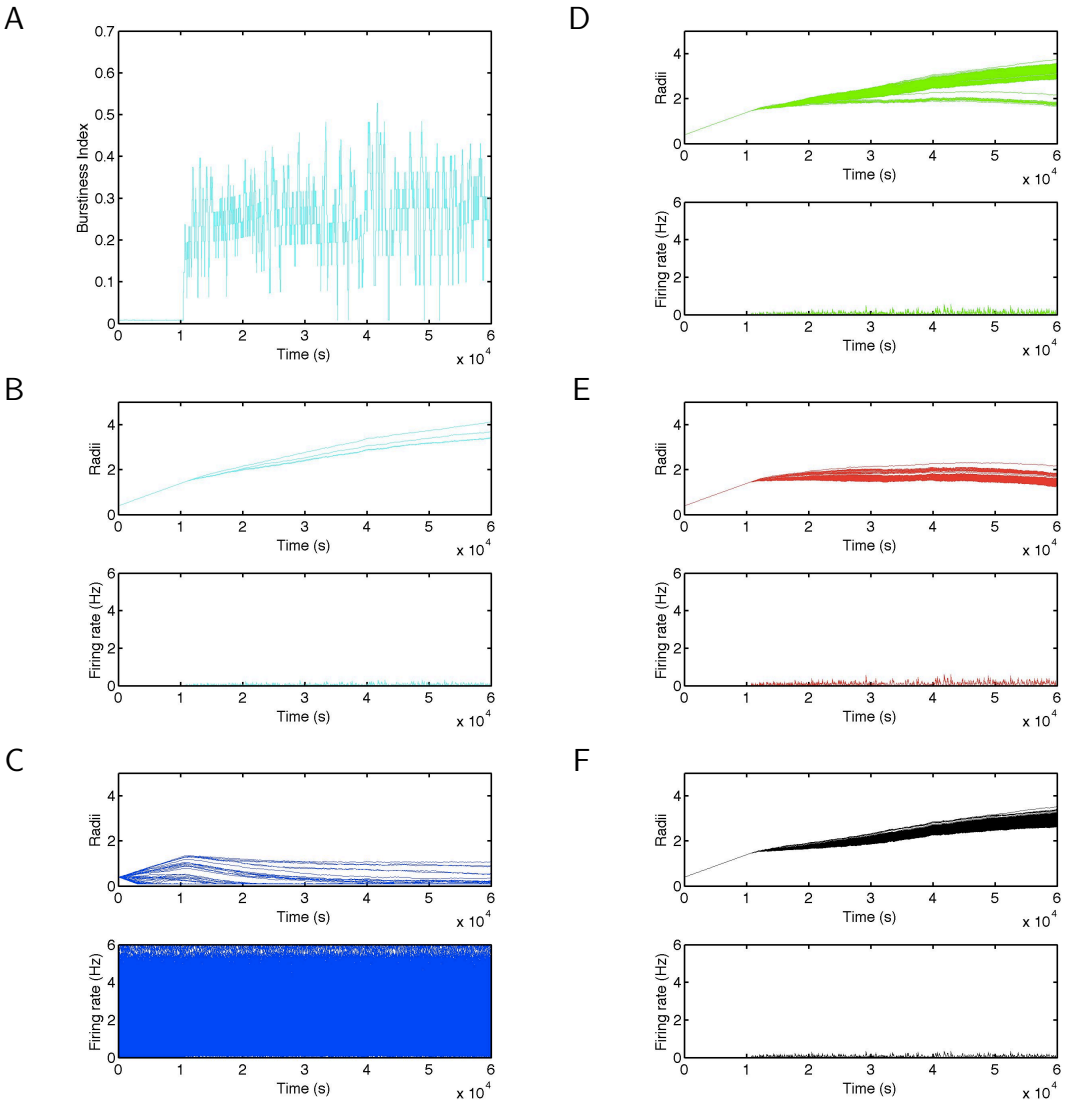

Figure 5.2: Evolution of burstiness index (A), radius, and firing rate for a simulation with parameters (target rate, fraction excitatory cells) of (0.1, 0.90) (B for the four corner neurons), (C for spontaneously active neurons), (D for edge neurons), (E for inhibitory neurons), and (F for excitatory, non-spontaneously active, interior, neurons).

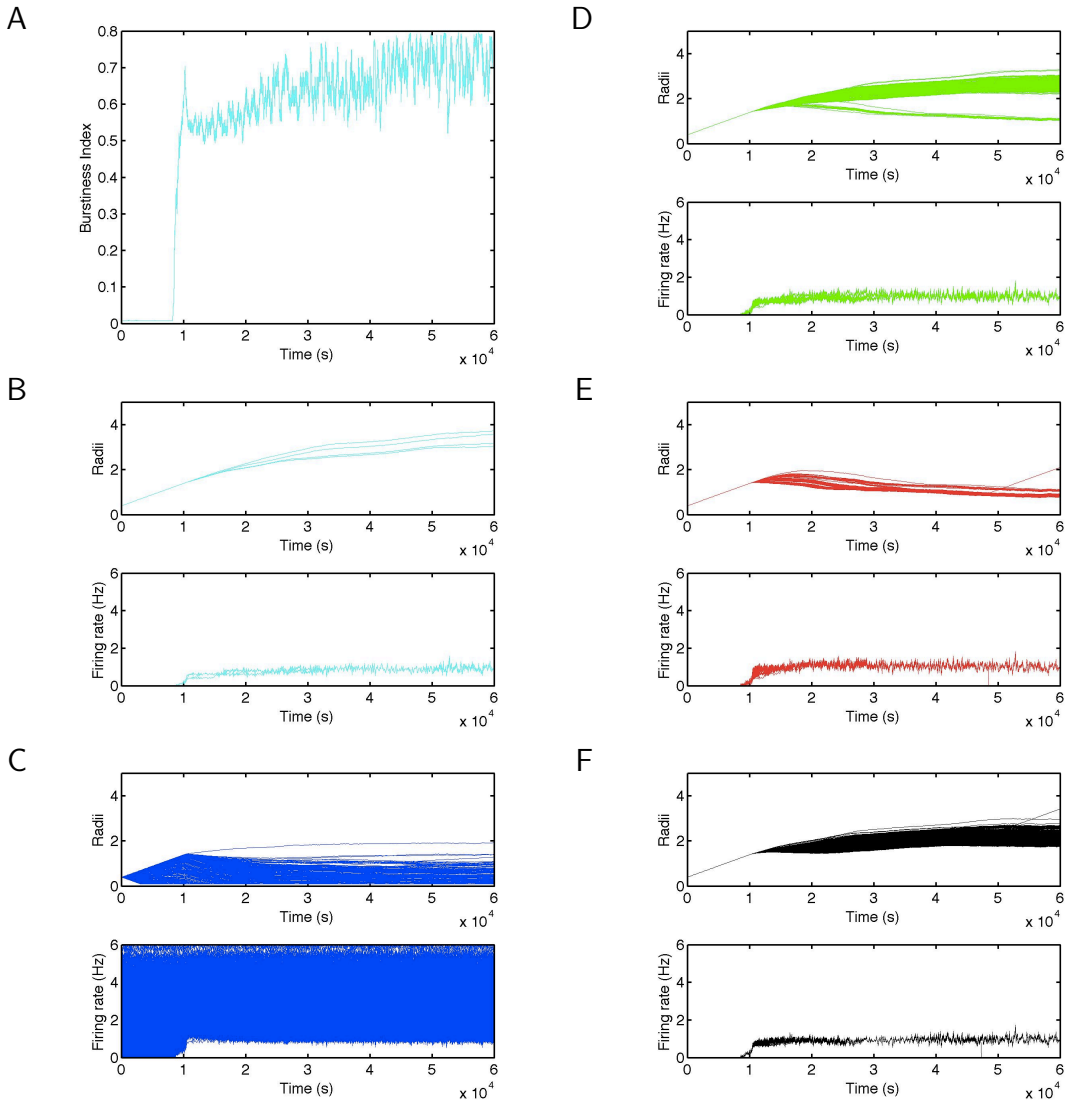

Figure 5.3: Evolution of burstiness index (A), radius, and firing rate for a simulation with parameters (target rate, fraction excitatory cells) of (1.0, 0.90) (B for the four corner neurons), (C for spontaneously active neurons), (D for edge neurons), (E for inhibitory neurons), and (F for excitatory, non-spontaneously active, interior, neurons).

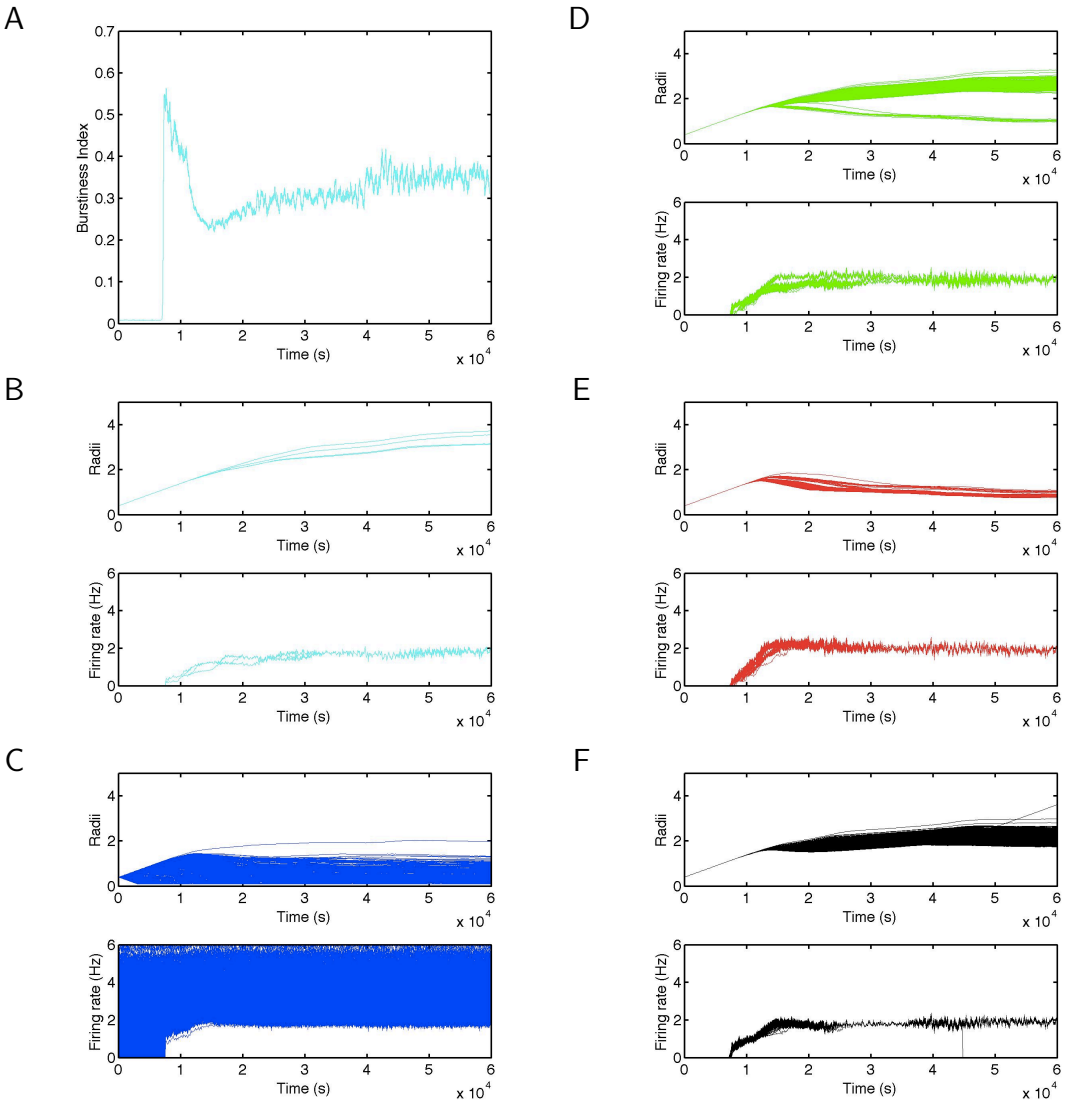

Figure 5.4: Evolution of burstiness index (A), radius, and firing rate for a simulation with parameters (target rate, fraction excitatory cells) of (1.9, 0.90) (B for the four corner neurons), (C for spontaneously active neurons), (D for edge neurons), (E for inhibitory neurons), and (F for excitatory, non-spontaneously active, interior, neurons).

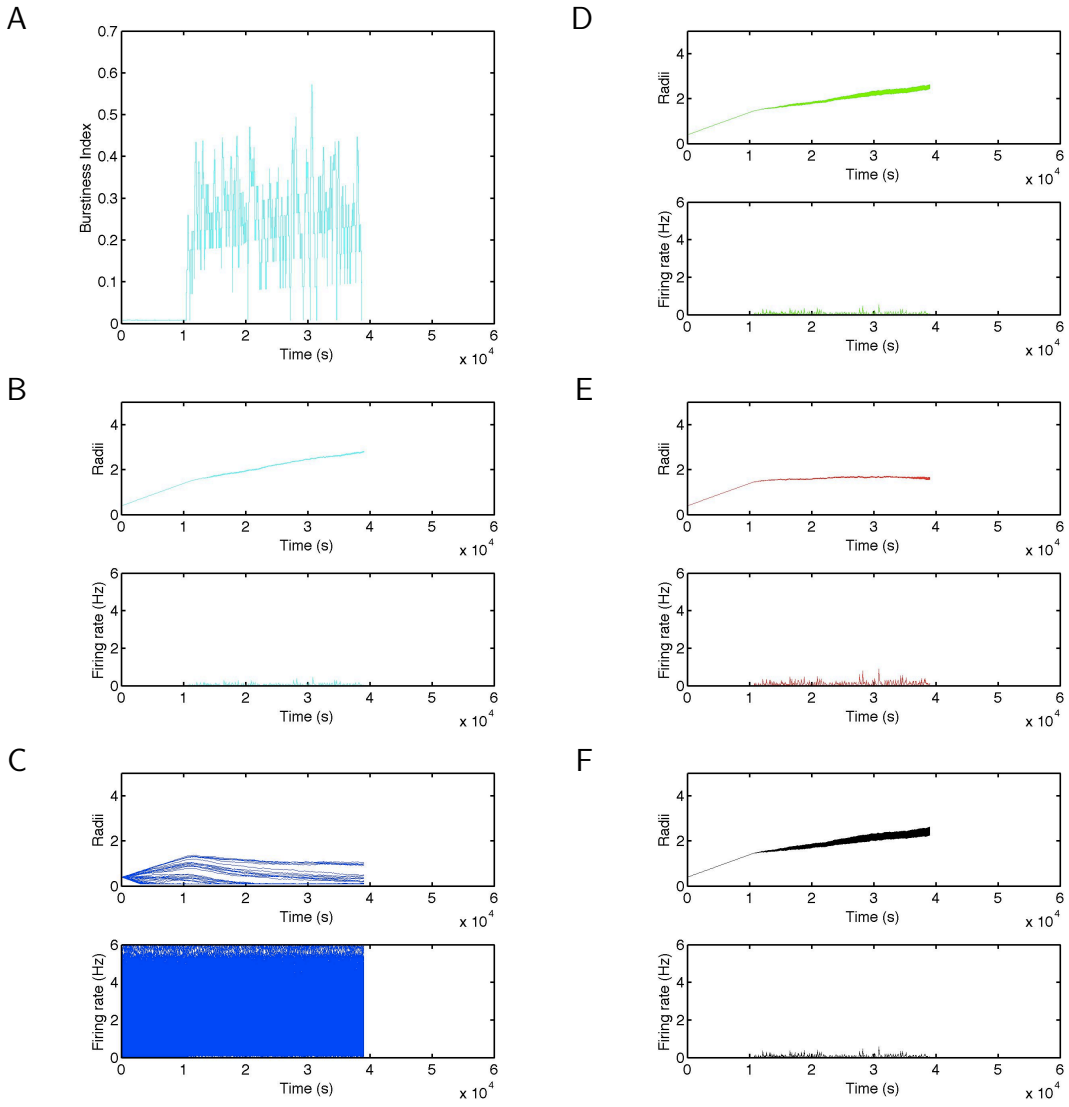

Figure 5.5: Evolution of burstiness index (A), radius, and firing rate for a simulation with parameters (target rate, fraction excitatory cells) of (0.1, 0.98) (B for the four corner neurons), (C for spontaneously active neurons), (D for edge neurons), (E for inhibitory neurons), and (F for excitatory, non-spontaneously active, interior, neurons).

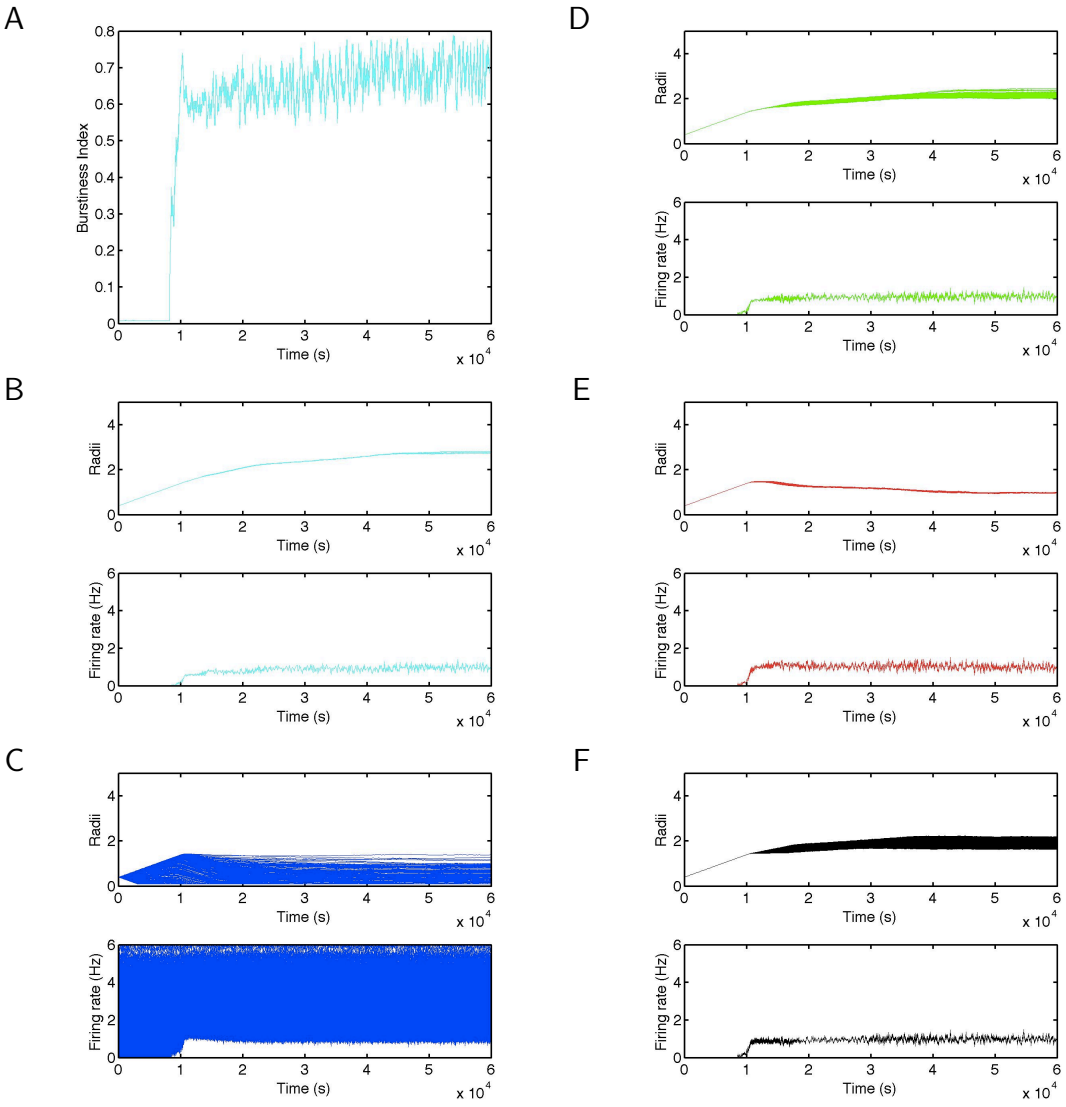

Figure 5.6: Evolution of burstiness index (A), radius, and firing rate for a simulation with parameters (target rate, fraction excitatory cells) of (1.0, 0.98) (B for the four corner neurons), (C for spontaneously active neurons), (D for edge neurons), (E for inhibitory neurons), and (F for excitatory, non-spontaneously active, interior, neurons).

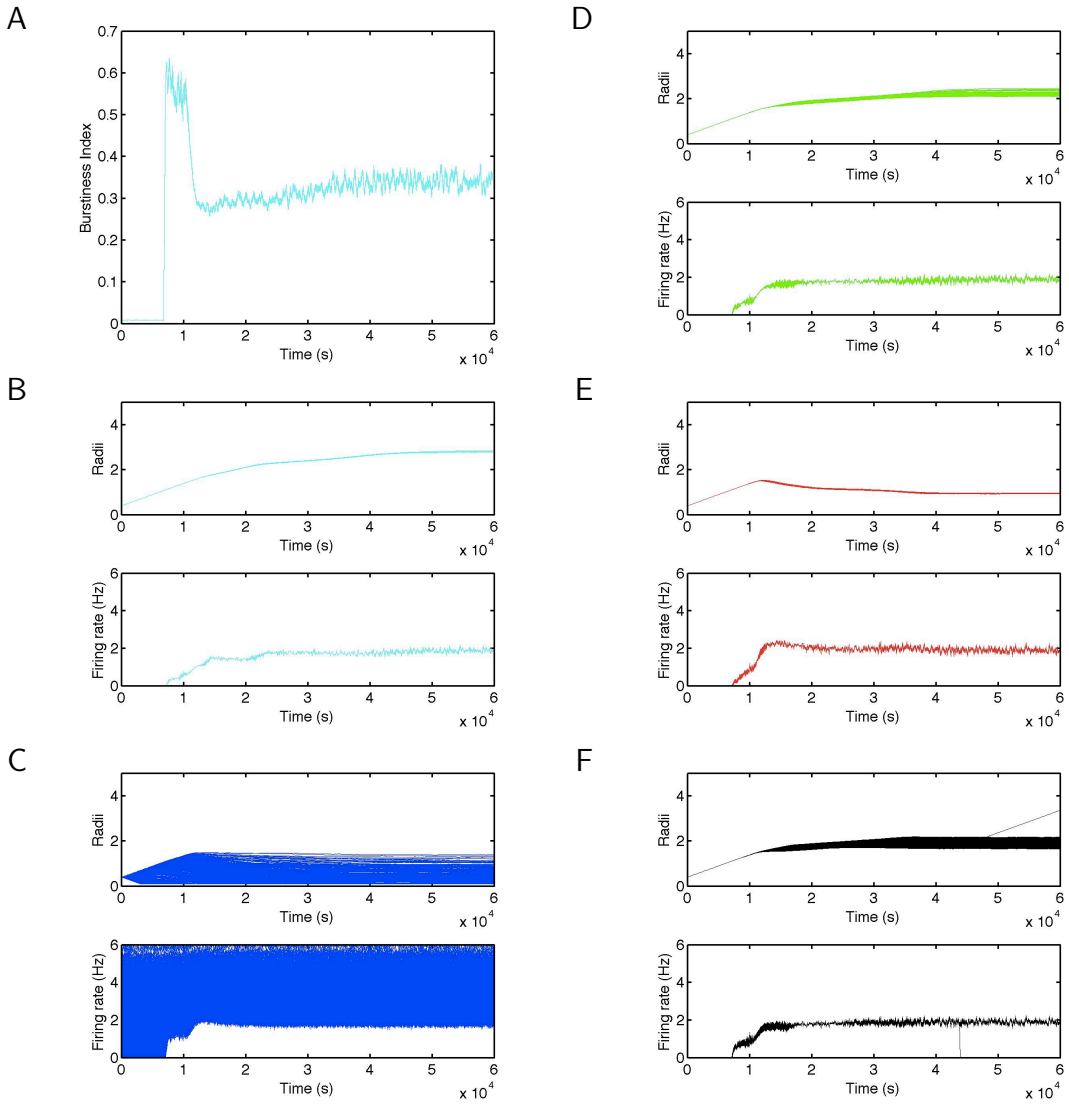

Figure 5.7: Evolution of burstiness index (A), radius, and firing rate for a simulation with parameters (target rate, fraction excitatory cells) of (1.9, 0.98) (B for the four corner neurons), (C for spontaneously active neurons), (D for edge neurons), (E for inhibitory neurons), and (F for excitatory, non-spontaneously active, interior, neurons).

spike counts (we did not record each individual neuron's spike history in 10 msec bins because of memory constraint — it would require about 240 GB to store 10,000 neurons' spike histories for a 60,000 sec simulation using 10 msec bins).

We plotted average per neuron firing rate (APNFR) versus time. APNFR was spike count of the network per second (Hz) normalized per neuron. It was derived by the formula: (spike counts in 10ms of the network)/(0.01 $*$ (number of neurons)). The following was a summary of what we observed from the plot.

- APNFR stayed at the spontaneous rate (0.21 Hz per neuron) until neurons grew to connect to each other. The mean firing rate of spontaneously active neurons was about 2.1 Hz per neuron, so APNFR when only these neurons were firing was about 0.21 Hz per neuron (remember that 10% of neurons were spontaneously active).

- Once neurons were connected, non-spontaneously firing neurons started firing so the overall firing rate increased.

- However, spiking was not uniformly distributed along time. Neurons tended to fire around the same time, producing bursts and inter-burst periods.

- In the inter-burst period, the firing rate stayed around the spontaneous firing rate.

- APNFR evolved during each burst, increasing more or less smoothly from the background rate to a peak, then decreasing back to the inter-burst behavior.

- The shape of the burst and the length of inter-burst period varied, depending on the simulation parameters, and evolving during each simulation. Figure 5.8 illustrates this burst evolution during one simulation.

To distinguish the bursts observed here from the bursts in small networks, we called these bursts *micro-bursts*.

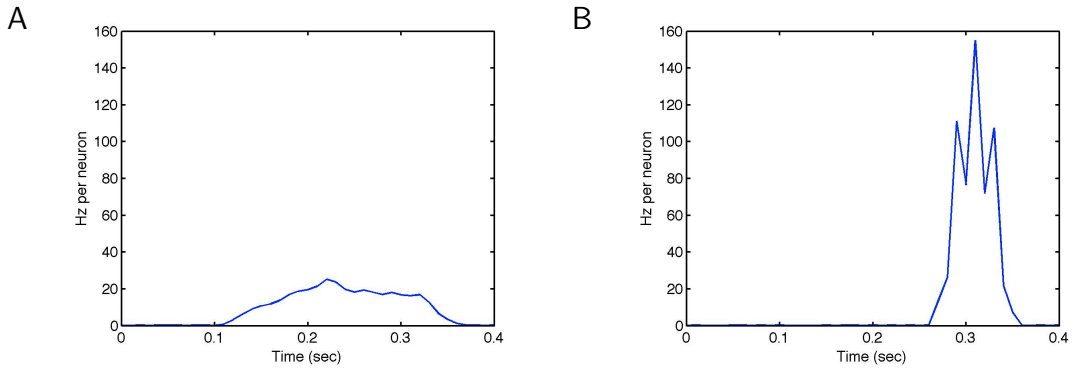

Figure 5.8: Example of burst patterns. Bursts around 10,800 sec (A) and 52,400 sec (B) with parameters (target rate, fraction excitatory cells) = (1.0, 0.98).

5.2.2 Profile of Spontaneous Firing

For large networks, we focused on the micro-bursts. To distinguish the micro-bursts from the background firing (spontaneous firing), we first investigated the pattern of the background firing. We made histograms of spike counts. The histograms (Figure 5.9–Figure 5.14, A, C, and E) show the distribution of spike counts in 10 msec bins during particular periods (5,000–5,100 sec (A) when there were no bursts, 53,000–53,100 sec (B), and 59,900–60,000 sec (C) simulation time, the last 100 sec of simulation). The spike counts of the histograms (x-axes) were normalized APNFR (Hz per neuron) so that we could use the same metrics regardless of the size of time bins and size of network. APNFR above 0.5 Hz per neuron was cut off in these histograms. The y-axes of these histograms indicate the probabilities that data fell into values specified in x-axes. The bin size of the histogram was 0.01 Hz per neuron.

We also made normal probability plots of spike counts (Figure 5.13–Figure 5.14, B, D, and F) using the *normplot(X)* function of MATLAB to graphically assess whether the data in X was consistent with a normal distribution. The line in the plot connects the 25th and 75th percentiles in the data X (a linear fit of the sample order statistics), and it is extrapolated out to the ends of the sample to help evaluate the linearity of the data. The

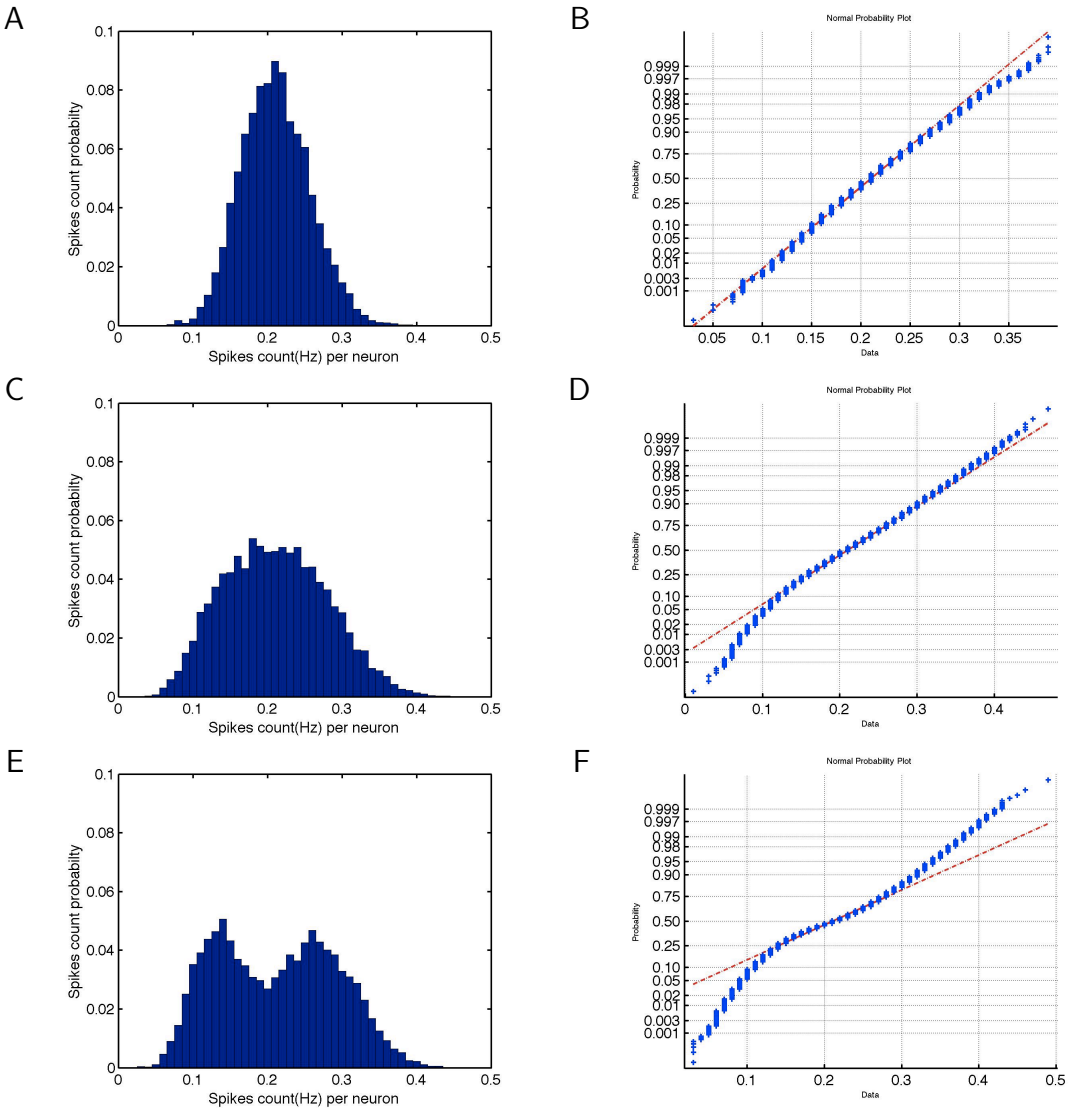

Figure 5.9: Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (target rate, fraction excitatory cells) = (0.1, 0.90). Simulation times were 5,000–5,100 sec (A)(B) when no micro-bursting ($\mu = 0.2107$, $\sigma = 0.0462$, $\sum_{i=0}^{0.5} P_i = 1.0$), 53,000–53,100 sec (C)(D)($\mu = 0.2106$, $\sigma = 0.0692$, $\sum_{i=0}^{0.5} P_i = 0.9983$), and 59,900–60,000 sec (E)(F)($\mu = 0.2116$, $\sigma = 0.0804$, $\sum_{i=0}^{0.5} P_i = 0.9983$). The unit of x-axis was APNFR (Hz per neuron), which was calculated by $(spikes\ count\ in\ 10\ ms\ of\ the\ network)/(0.01 * (number\ of\ neurons))$. Bin size was 0.01Hz per neuron.

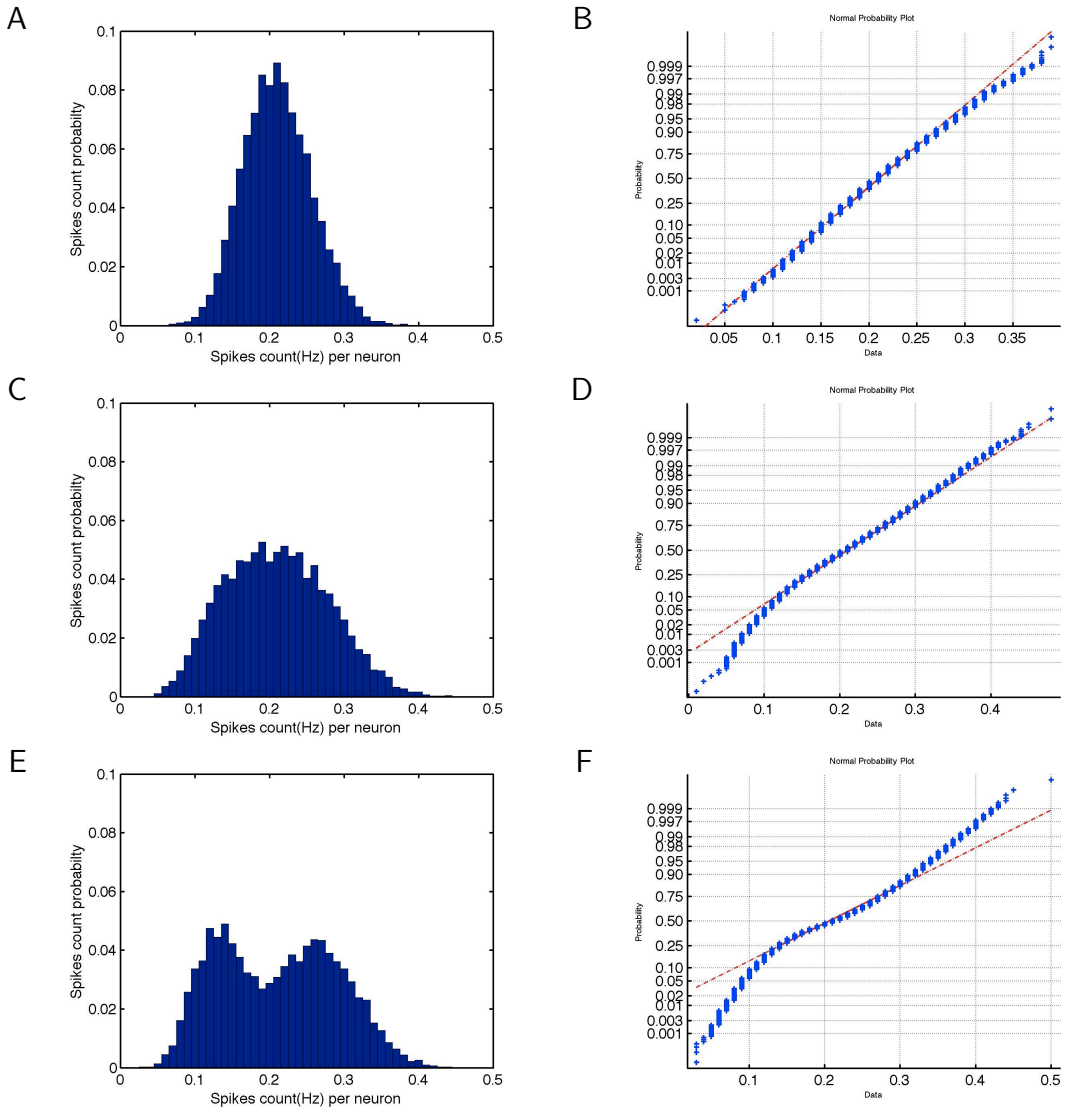

Figure 5.10: Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (target rate, fraction excitatory cells) = (1.0, 0.90). Simulation times were 5,000–5,100 sec (A)(B) when no micro-bursting ($\mu = 0.2103$, $\sigma = 0.0461$, $\sum_{i=0}^{0.5} P_i = 1.0$), 53,000–53,100 sec (C)(D)($\mu = 0.2098$, $\sigma = 0.0692$, $\sum_{i=0}^{0.5} P_i = 0.9719$), and 59,900–60,000 sec (E)(F)($\mu = 0.2106$, $\sigma = 0.0801$, $\sum_{i=0}^{0.5} P_i = 0.9772$). The unit of x-axis was APNFR (Hz per neuron), which was calculated by $(spikes\ count\ in\ 10\ ms\ of\ the\ network)/(0.01 * (number\ of\ neurons))$. Bin size was 0.01Hz per neuron.

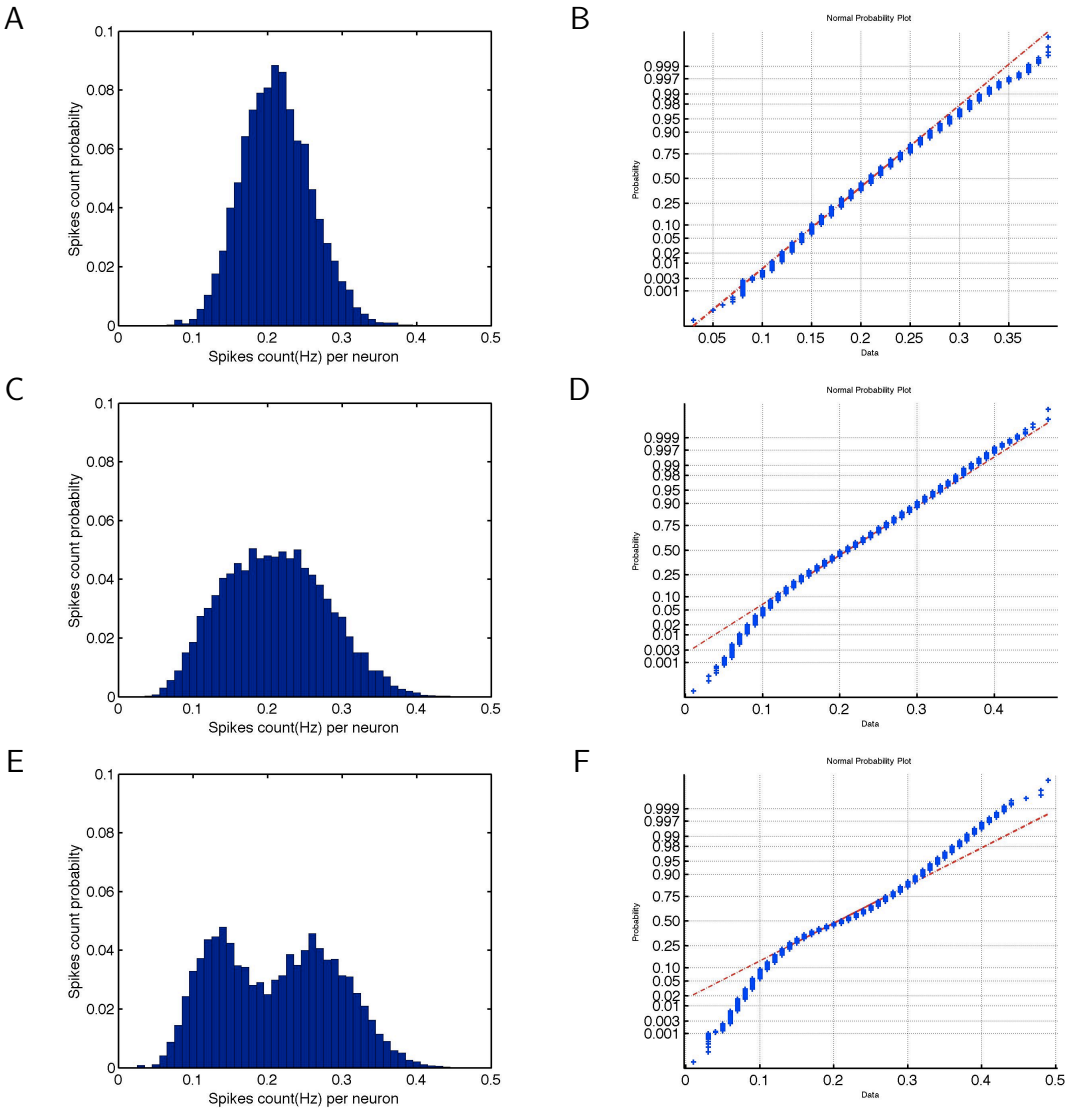

Figure 5.11: Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (target rate, fraction excitatory cells) = (1.9, 0.90). Simulation times were 5,000–5,100 sec (A)(B) when no micro-bursting ($\mu = 0.2121$, $\sigma = 0.0465$, $\sum_{i=0}^{0.5} P_i = 1.0$), 53,000–53,100 sec (C)(D)($\mu = 0.2102$, $\sigma = 0.0693$, $\sum_{i=0}^{0.5} P_i = 0.9566$), and 59,900–60,000 sec (E)(F)($\mu = 0.2112$, $\sigma = 0.0806$, $\sum_{i=0}^{0.5} P_i = 0.9554$). The unit of x-axis was APNFR (Hz per neuron), which was calculated by $(spikes\ count\ in\ 10\ ms\ of\ the\ network)/(0.01 * (number\ of\ neurons))$. Bin size was 0.01Hz per neuron.

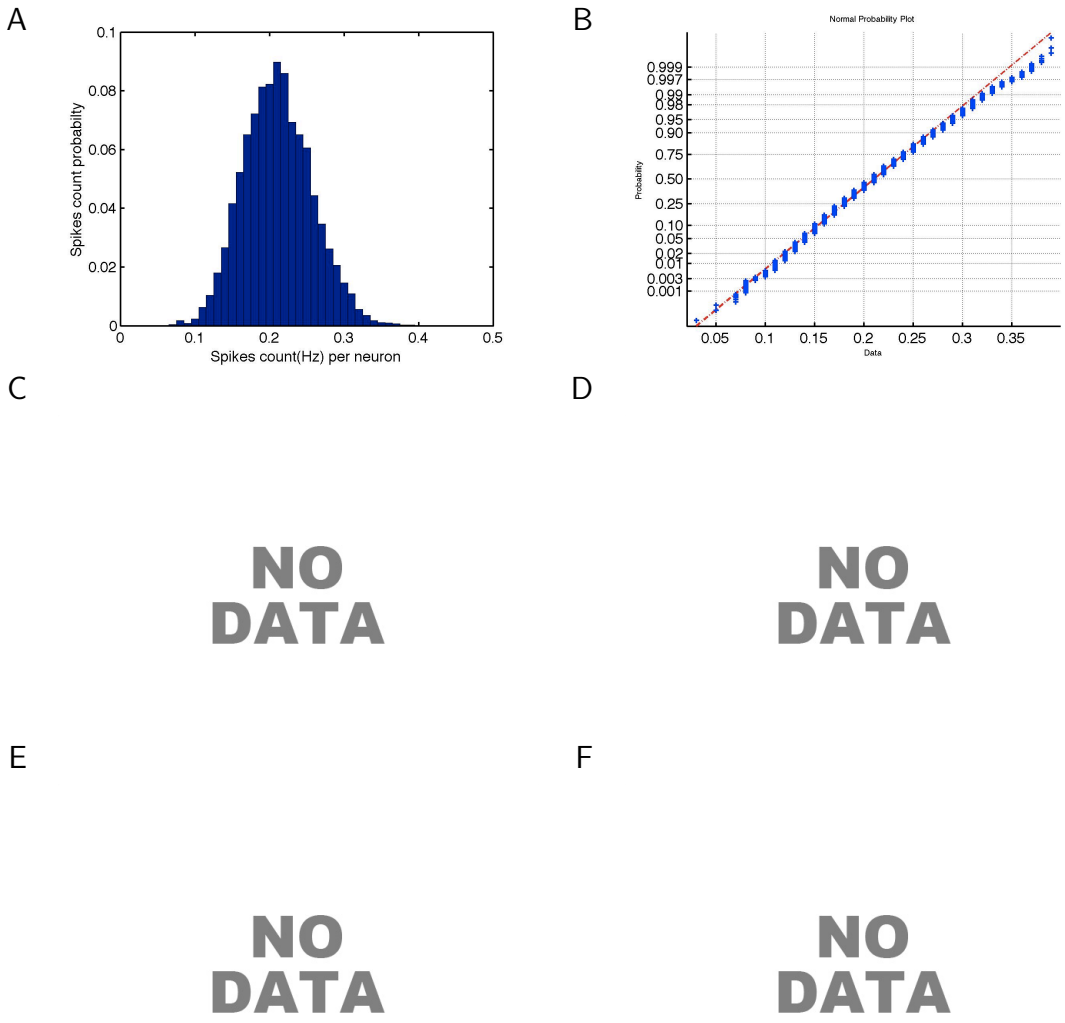

Figure 5.12: Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (target rate, fraction excitatory cells) = (0.1, 0.98). Simulation times were 5,000–5,100 sec (A)(B) when no micro-bursting ($\mu = 0.2107$, $\sigma = 0.0462$, $\sum_{i=0}^{0.5} P_i = 1.0$). The unit of x-axis was APNFR (Hz per neuron), which was calculated by $(spikes\ count\ in\ 10\ ms\ of\ the\ network)/(0.01 * (number\ of\ neurons))$. Bin size was 0.01Hz per neuron.

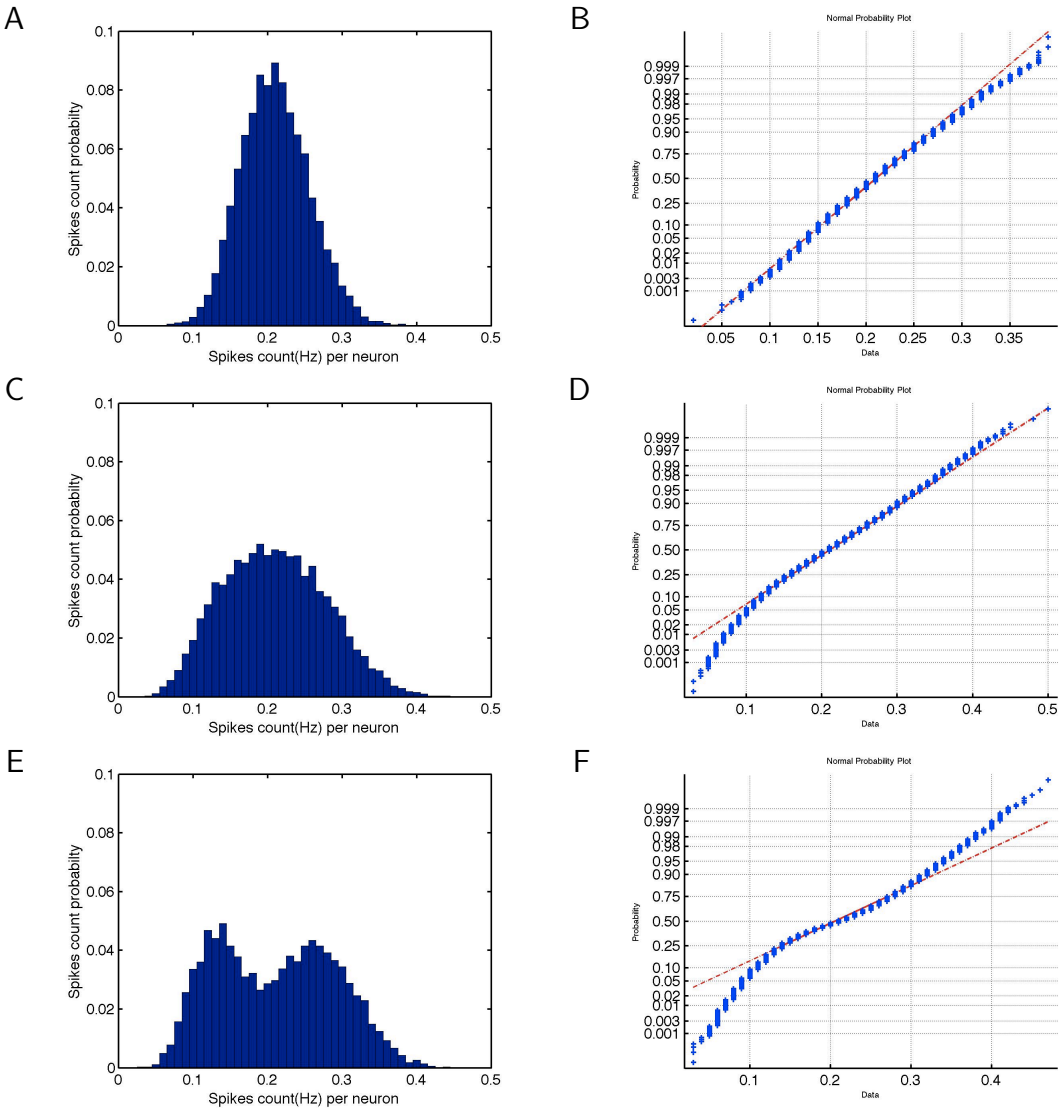

Figure 5.13: Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (target rate, fraction excitatory cells) = (1.0, 0.98). Simulation times were 5,000–5,100 sec (A)(B) when no micro-bursting ($\mu = 0.2103$, $\sigma = 0.0461$, $\sum_{i=0}^{0.5} P_i = 1.0$), 53,000–53,100 sec (C)(D)($\mu = 0.2099$, $\sigma = 0.0693$, $\sum_{i=0}^{0.5} P_i = 0.9699$), and 59,900–60,000 sec (E)(F)($\mu = 0.2108$, $\sigma = 0.0802$, $\sum_{i=0}^{0.5} P_i = 0.9703$). The unit of x-axis was APNFR (Hz per neuron), which was calculated by $(spikes\ count\ in\ 10\ ms\ of\ the\ network)/(0.01 * (number\ of\ neurons))$. Bin size was 0.01Hz per neuron.

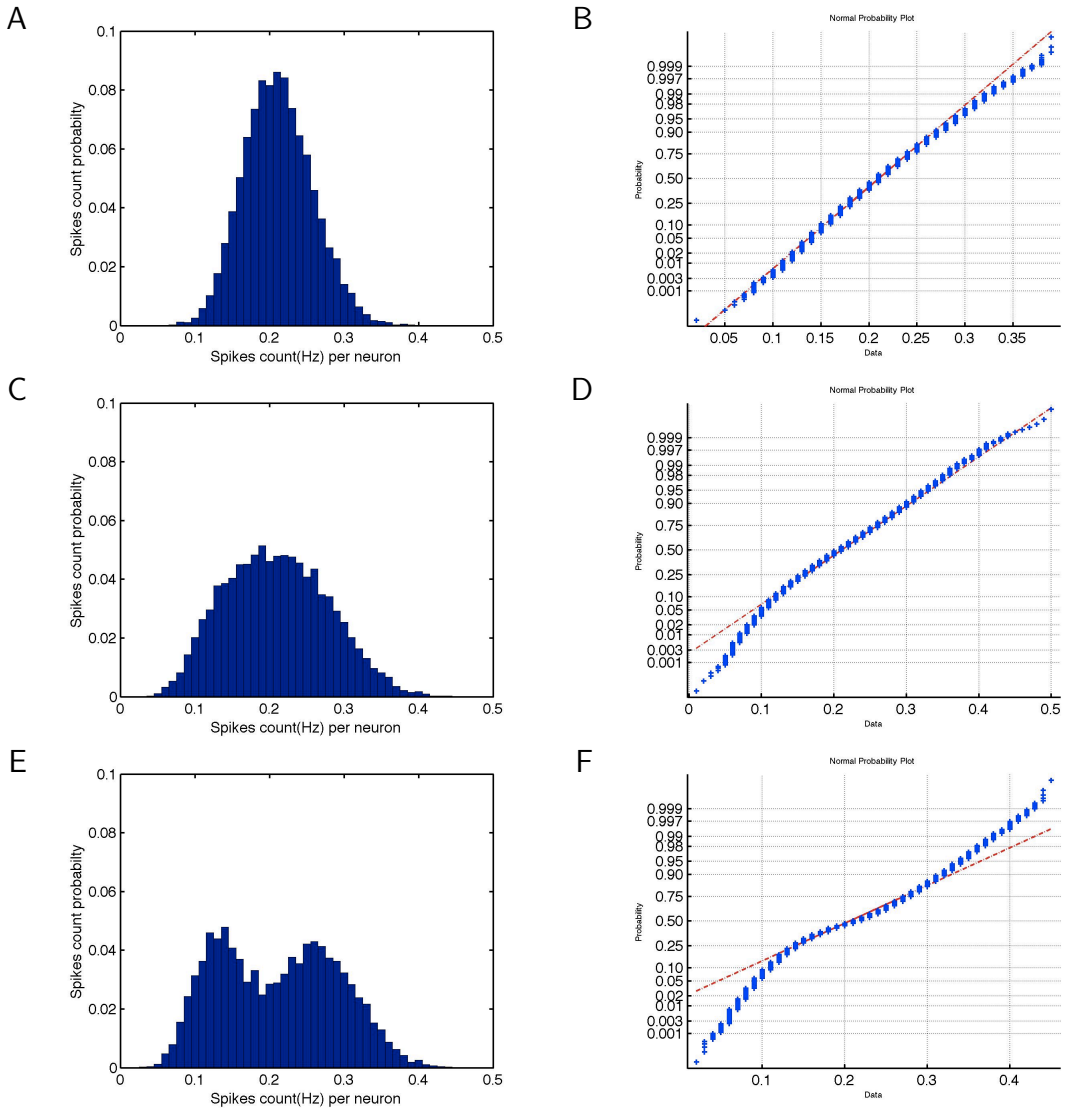

Figure 5.14: Distributions histograms of spike counts (Hz) per neuron of spontaneous firing and normal probability plots of them with parameter (target rate, fraction excitatory cells) = (1.9, 0.98). Simulation times were 5,000–5,100 sec (A)(B) when no micro-bursting ($\mu = 0.2114$, $\sigma = 0.0464$, $\sum_{i=0}^{0.5} P_i = 1.0$), 53,000–53,100 sec (C)(D)($\mu = 0.2097$, $\sigma = 0.06943$, $\sum_{i=0}^{0.5} P_i = 0.9475$), and 59,900–60,000 sec (E)(F)($\mu = 0.2108$, $\sigma = 0.0803$, $\sum_{i=0}^{0.5} P_i = 0.9516$). The unit of x-axis was APNFR (Hz per neuron), which was calculated by $(spikes\ count\ in\ 10\ ms\ of\ the\ network)/(0.01 * (number\ of\ neurons))$. Bin size was 0.01Hz per neuron.

x-axis of the plot is the value of data (APNFR), and the y-axis values are probabilities from zero to one, but the scale is not linear. The distance between tick marks on the y-axis matches the distance between the quantiles of a normal distribution. The plus signs plot the empirical probability versus the data value for each point in the data. In a normal probability plot, if all the data points fall near the line, an assumption of normality is reasonable. Otherwise, the points will curve away from the line, and an assumption of normality is not justified [33].

Figure 5.9–Figure 5.14 A show the distributions of spike counts when no micro-bursting was observed; therefore, those were the distributions of the spontaneous firing. According to the normal probability plots (Figure 5.9–Figure 5.14, B), these distributions were slightly skewed to higher values, but well fitted by a normal distribution. The distributions during 53,000–53,100 sec simulation time were slightly skewed to lower values (Figure 5.9–Figure 5.14, C and D), but it seemed to be normally distributed. The distributions during 59,900–60,000 sec simulation time (Figure 5.9–Figure 5.14, E and F) indicate a bi-modal distribution. These distribution patterns were affected by network development, but not affected by varying simulation parameters (target rate and fraction excitatory cells). These results suggested that distributions of background firing might be affected by changes of inter-connections between neurons, but the underlying mechanism causing these changes is an open question at this time. The ratio of APNFR below 0.5 Hz per neuron ($\sum_{i=0}^{0.5} P_i$) to total firing was determined by target rate, which meant that lower target rate induced higher occupancy of spontaneous firing (less bursting), and higher target rate induced lower occupancy of spontaneous firing (more bursting) (Table 5.1).

Figure 5.15 shows the distributions of spike counts in 10 msec bins of all firing rate of entire simulation period (0–60,000 sec). The spike counts of the plots (x-axes) were normalized APNFR (Hz per neuron), and y-axes indicate the number of samples within bin size of 0.01 Hz per neuron. Both axes were logarithmic scales. From those plots, we could observe two kinds of distribution patterns, those of which seemed to be originated from

Table 5.1: The ratio of APNFR below 0.5 Hz per neuron ($\sum_{i=0}^{0.5} P_i$) to total firing.

Parameters (target rate, fraction excitatory cells)	$\sum_{i=0}^{0.5} P_i$ (5,000–5,100 sec)	$\sum_{i=0}^{0.5} P_i$ (53,000–53,100 sec)	$\sum_{i=0}^{0.5} P_i$ (59,900–60,000 sec)
(0.1, 0.90)	1.0	0.9983	0.9983
(1.0, 0.90)	1.0	0.9719	0.9772
(1.9, 0.90)	1.0	0.9566	0.9554
(0.1, 0.98)	1.0	—	—
(1.0, 0.98)	1.0	0.9699	0.9703
(1.9, 0.98)	1.0	0.9475	0.9516

spontaneous firing and micro-bursting respectively. The distribution pattern of spontaneous firing was gradually shifted to the distribution pattern of micro-bursting within 0.3–0.5 Hz per neuron. This is a result of the fact that all micro-bursts exhibited firing rates that varied smoothly from the background (spontaneous) rate to the peak burst rate. Target rate determined the heights of the distribution pattern of micro-bursting; that is, lower target rate induced lower heights of the distribution, and higher target rate induced higher heights of the distribution. Peaks of the distribution of micro-bursting existed within 20–40 Hz per neuron, and it was more remarkable when fractions excitatory cells were higher.

5.2.3 Burst Profiles

From the results above, we could identify micro-bursts. First, we needed to define a threshold value that could distinguish micro-bursts from background firing. The threshold value should be high enough to filter out all background firing, but should be able to identify micro-bursts in the early simulation period, when heights of micro-bursts were usually lower. According to Figure 5.15, background firing and micro-bursts overlapped around 0.3–0.5 Hz per neuron, which meant that there was no threshold value that cleanly distinguish micro-bursts from background firing. So, we defined a threshold value that could remove almost all the background firing, but it was low enough to identify micro-bursts of early simulation period. Considering these factors, we identified the threshold value as 0.5 Hz per neuron

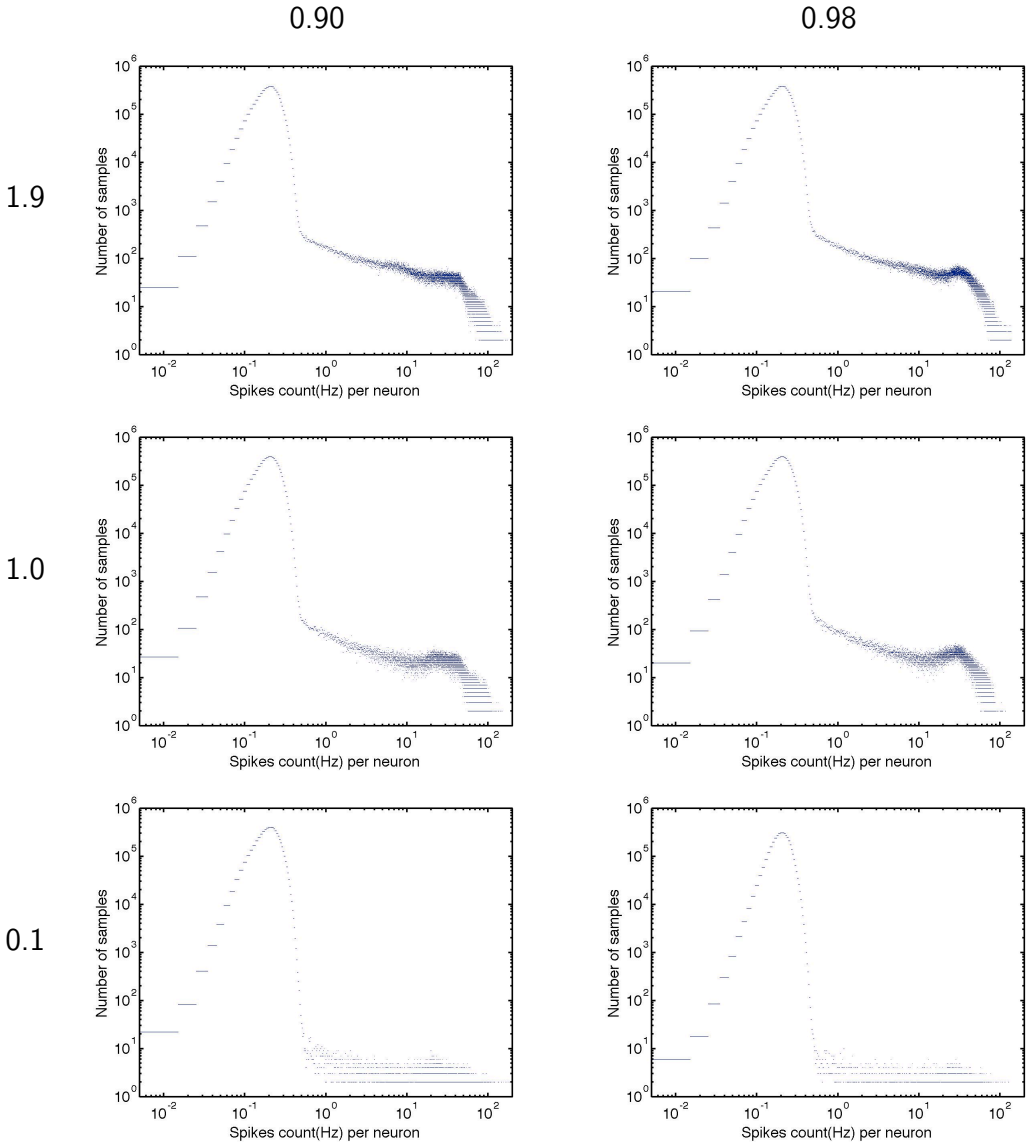

Figure 5.15: Distribution of spike counts (Hz) per neuron of all firing rate of entire simulation period with parameters (target rate, fraction excitatory cells) = (0.1, 0.90), (1.0, 0.90), (1.9, 0.90), (0.1, 0.98), (1.0, 0.98), and (1.9, 0.98). The unit of x-axis was average per neuron firing rate (Hz per neuron), and y-axis indicated the number of samples within bin size of 0.01 Hz per neuron. These axes were logarithmic scales.

(according to Figure 5.9–Figure 5.14 B, D, and F, the probability of APNFR of 0.5 Hz per neuron or above of the background firing was less than 0.001). This value was high enough to distinguish micro-bursts from the background firing, and would not disturb the data of micro-bursts during the early simulation period (see Figure 5.8, A, as an example of such a micro-burst).

Once we identified micro-bursts, we could characterize them. Bursts could be characterized by intra-burst parameters (intra-burst profile) and inter-burst parameters (inter-burst profile).

intra-burst profile The shape of a micro-burst was described using width, peak height, peak position, and spike count per burst. The width was the duration of a micro-burst. The peak height was the maximum firing rate within a micro-burst, and the peak position was time from burst start to peak bin. The spike count per burst was the number of spikes in the burst normalized per neuron. To explore burst evolution during each simulation, we divided 60,000sec simulations into windows of 1,000 sec (10 epochs), calculated the mean values of burst profiles, and plotted them versus simulation time.

inter-burst profile Inter burst intervals (IBIs) were calculated as the temporal distances between peaks of neighboring micro-bursts. We made histograms of IBIs of 5 simulation periods (10,000–20,000 sec, 20,000–30,000 sec, 30,000–40,000 sec, 40,000-50,000 sec, and 50,000–60,000 sec). We also computed the number of bursts per each 1,000 sec (10 epochs) and plotted this burst count, scaled as bursts per second, versus simulation time.

In addition to intra-burst and inter-burst profiles, we plotted the evolution of the APNFR.

5.3 Results

A set of 6 simulations was performed with target rates (0.1, 1.0 and 1.9) and fraction of excitatory neurons (0.90 and 0.98).

5.3.1 Intra Burst Profiles

Figure 5.16–Figure 5.19 show intra-burst profile statistics. Figure 5.16 shows the overview of the evolution of bursts shape in 3D plots. Burst shape plot was the mean burst shape in a 1,000 sec time window; this was plotted along simulation time. As networks grew, network bursts were observed from about 7,000-10,000 sec simulation time. Networks with higher target rate tended to start bursting earlier. The widths of early network bursts were typically longer and heights were lower, gradually evolving into narrower and more intense with longer rising phase in general. Network bursts with lower target rate ended up narrower and more intense (Figure 5.17–Figure 5.18). However, the number of spikes included per burst was not affected by varying simulation parameters (Figure 5.19), which meant that the inverse relationship between burst width and height compensated for each other to produce the same number of spikes within a burst. In Figure 5.19, the plots of lowest target rate (0.1) exhibited great variability along time, the cause of which seemed to be fewer number of sampling data (fewer number of bursts) when target rate was low. Other than for the lowest target rate (0.1), the evolution of spikes per burst was similar for networks with the same fraction of excitatory cells — linearly doubling the target rate had little effect.

Figure 5.21 shows evolutions of APNFR with varying simulation parameters. The APNFR converged on around the target rates except for the case of target rate being 0.1, in which firing of spontaneously active neurons had bigger influence to increase the average firing rate. Figure 5.20 shows evolution of burst counts with varying simulation parameters. Networks with lower target rates generated lower numbers of bursts, and networks with higher target rates generated higher number of bursts. Figure 5.20 C (target rate,

fraction excitatory cells, 1.9, 0.90) exhibits an overshoot of bursting at the early period of development.

5.3.2 Inter Burst Profiles

Figure 5.22–Figure 5.27 show distributions of inter burst intervals (IBIs). Networks with lower target rate generated longer IBIs, and variability of them was wider. The shapes of most IBIs distributions were skewed (higher occurrence of short IBIs than long ones). The variability of IBIs distribution when target rate was 0.1 was maximum and showed multi-modal distributions (Figure 5.22), whereas the variability of IBIs distribution when target rate was 1.9 was minimum, where changes of distribution along entire simulation period was also stable (Figure 5.27). Figure 5.23 and Figure 5.26, where simulation parameters were (target rate, fraction excitatory cells) = (1.0, 0.90) and (1.0, 0.98), show increasing both mean and standard deviation of IBIs distributions as networks developed. At the same target rate, lower fraction of excitatory cells tended to show wider variability of IBIs distribution.

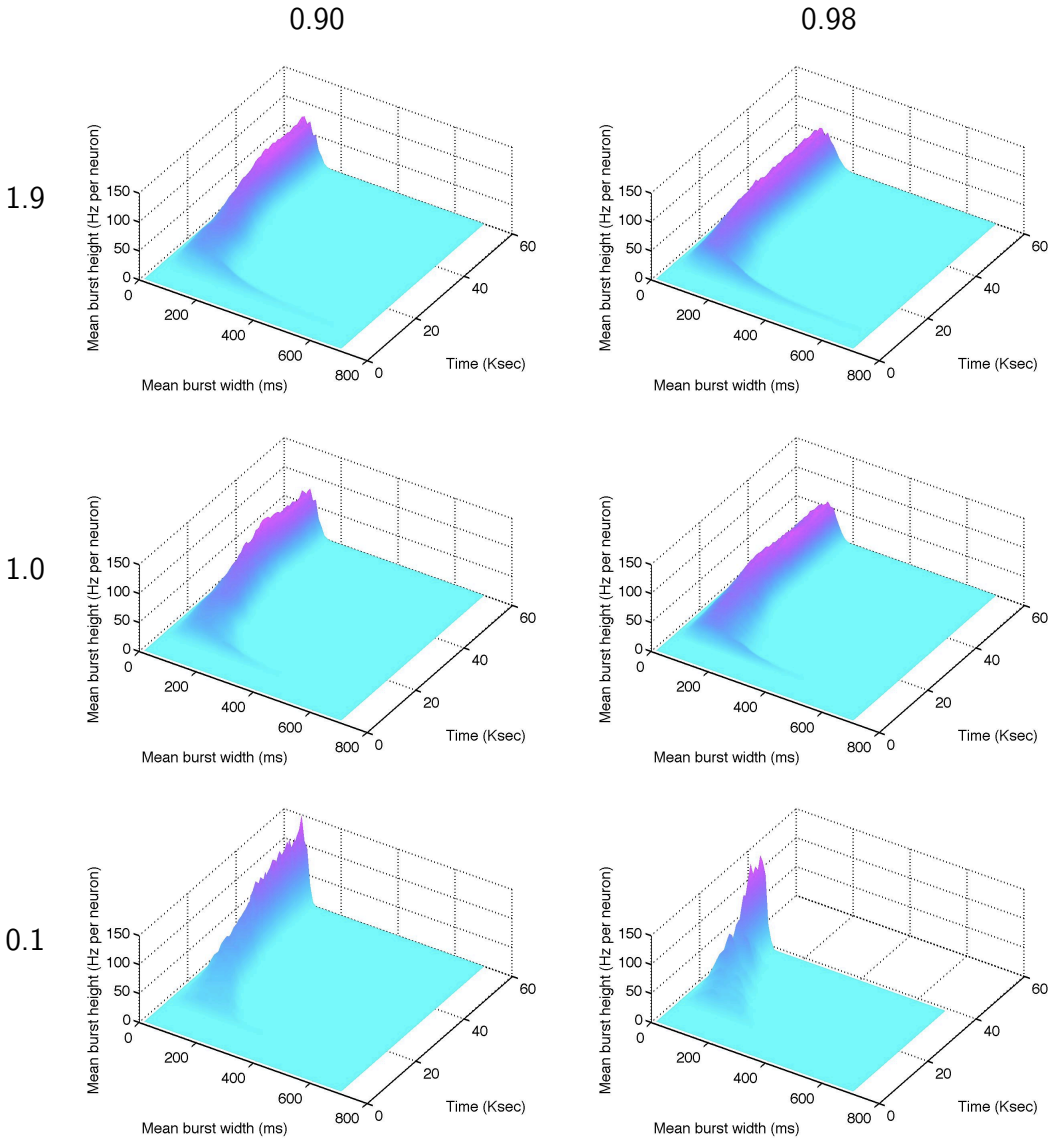

Figure 5.16: Evolution of burst shape with parameters (target rate, fraction excitatory cells) = (0.1, 0.90), (1.0, 0.90), (1.9, 0.90), (0.1, 0.98), (1.0, 0.98), and (1.9, 0.98).

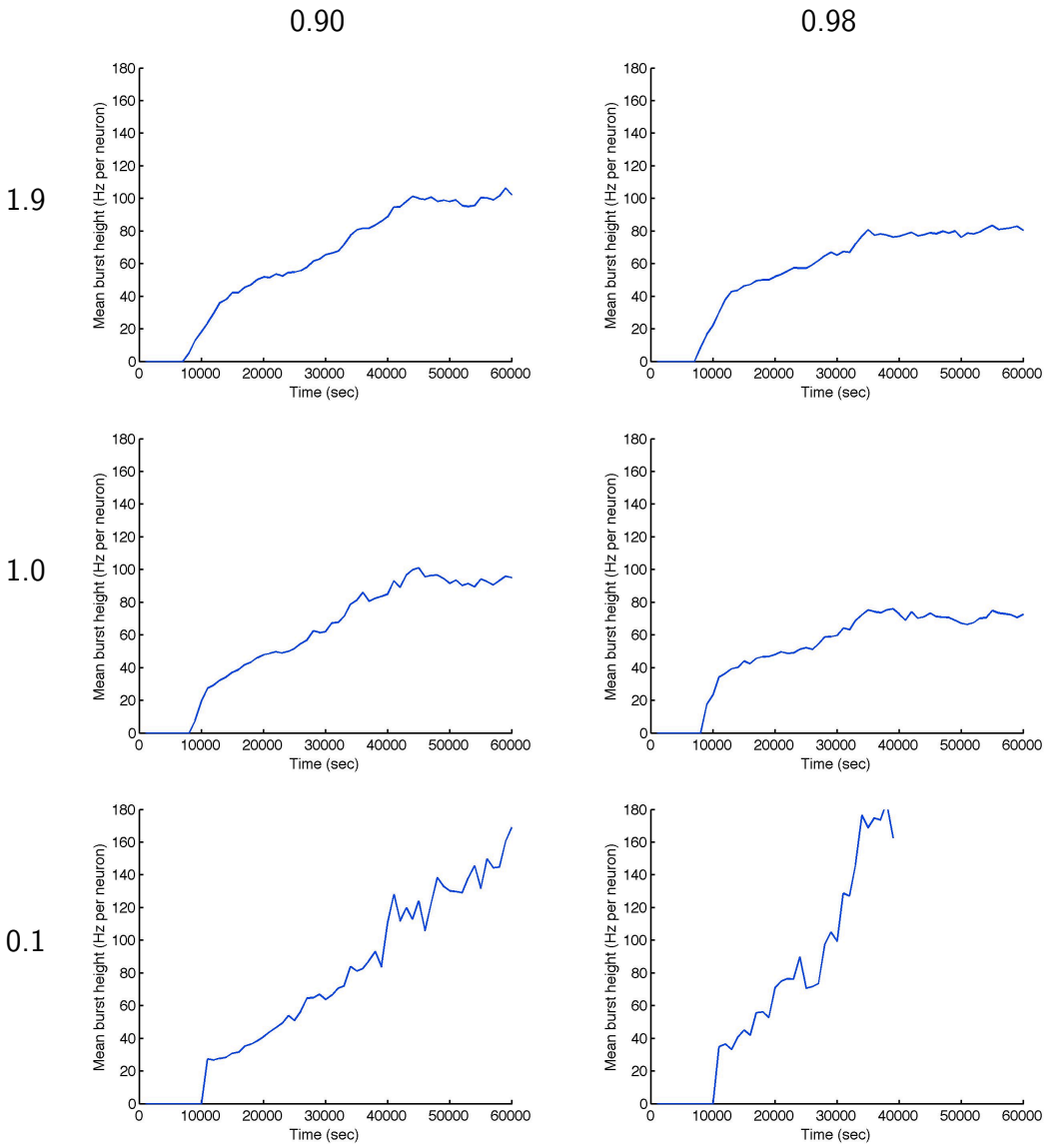

Figure 5.17: Evolution of mean burst height with parameters (target rate, fraction excitatory cells) = (0.1, 0.90), (1.0, 0.90), (1.9, 0.90), (0.1, 0.98), (1.0, 0.98), and (1.9, 0.98).

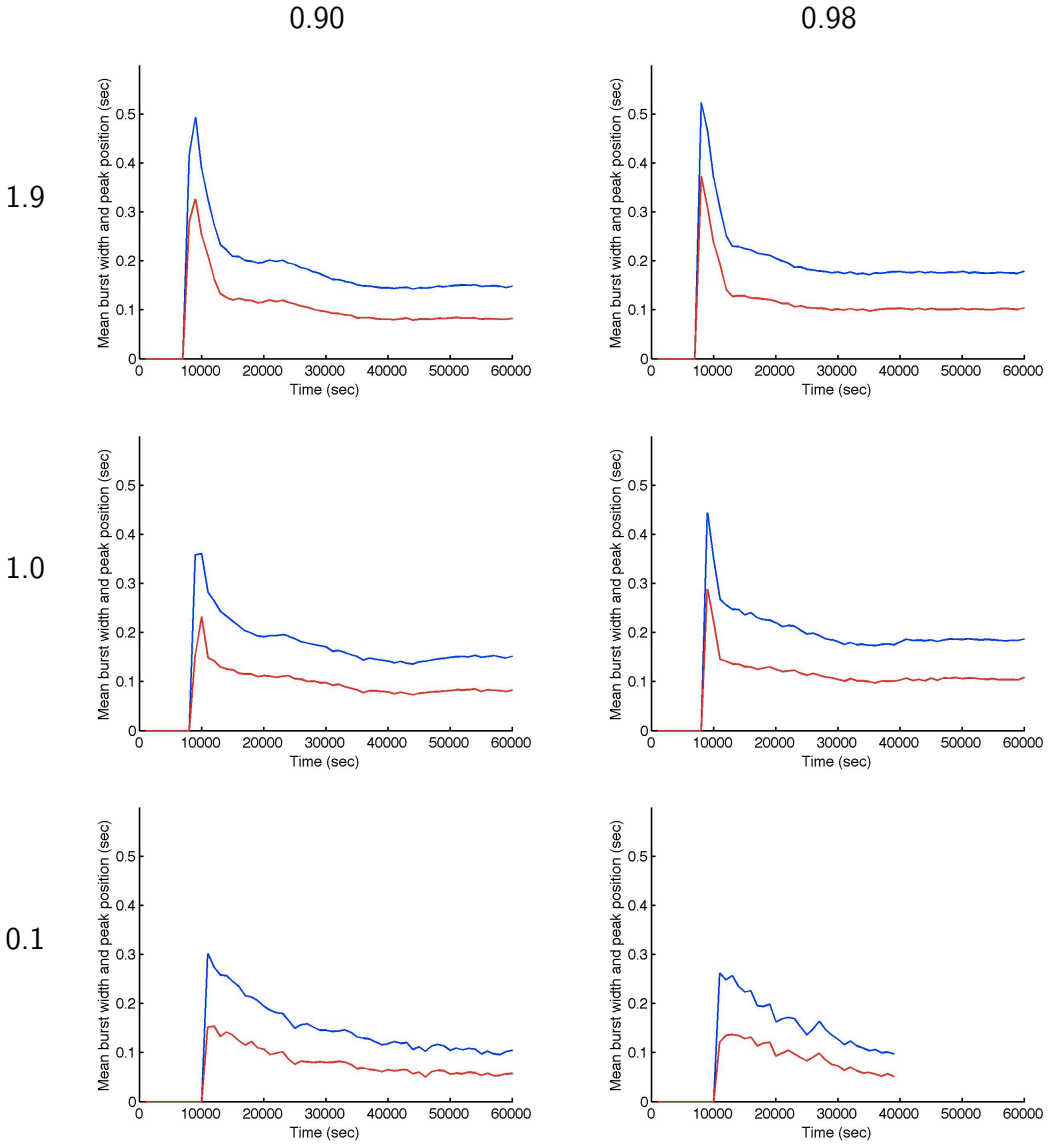

Figure 5.18: Evolution of mean burst width (blue) and mean peak position (red) with parameters (target rate, fraction excitatory cells) = (0.1, 0.90), (1.0, 0.90), (1.9, 0.90), (0.1, 0.98), (1.0, 0.98), and (1.9, 0.98).

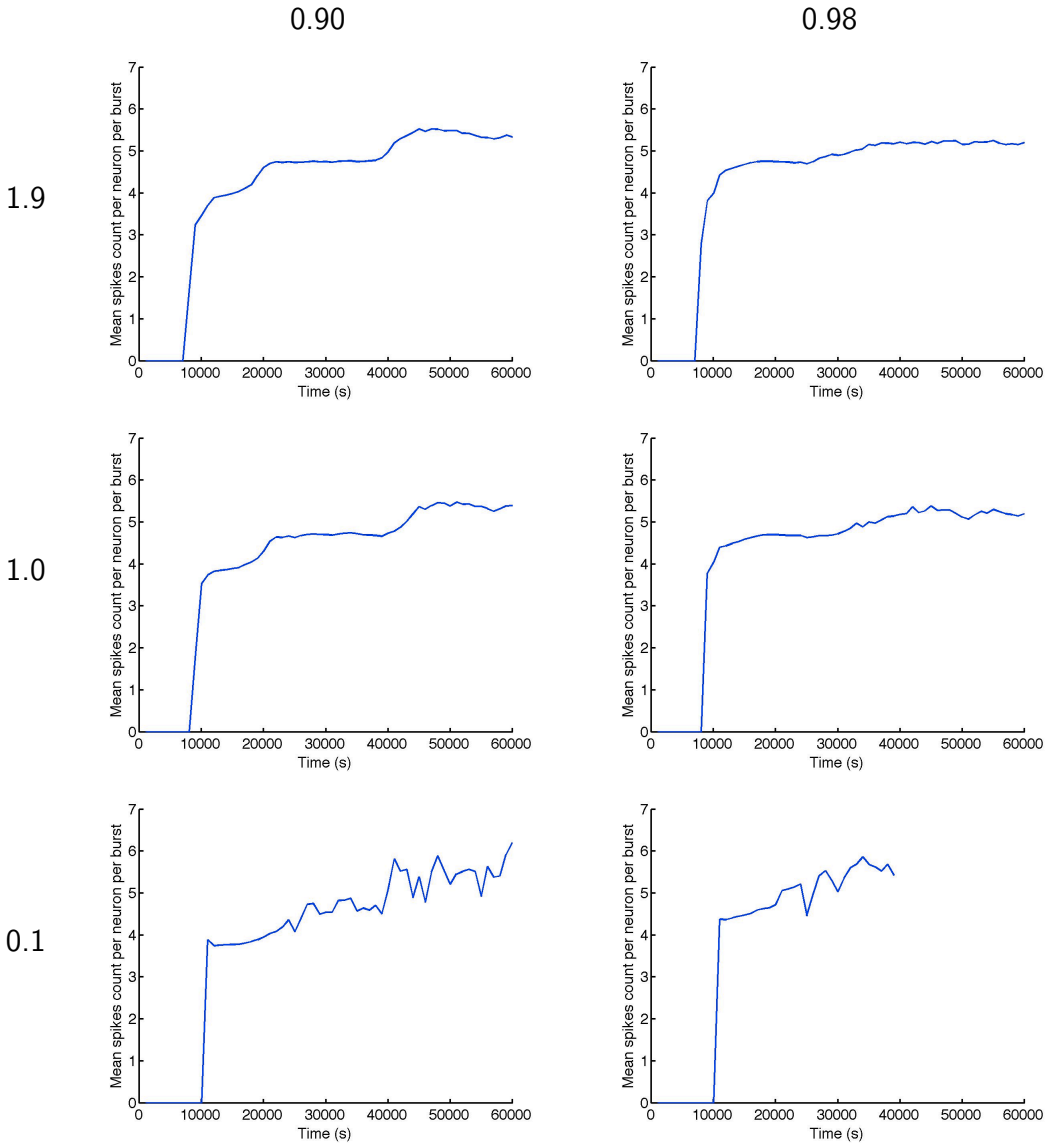

Figure 5.19: Evolution of mean spike counts per burst with parameters (target rate, fraction excitatory cells) = (0.1, 0.90), (1.0, 0.90), (1.9, 0.90), (0.1, 0.98), (1.0, 0.98), and (1.9, 0.98).

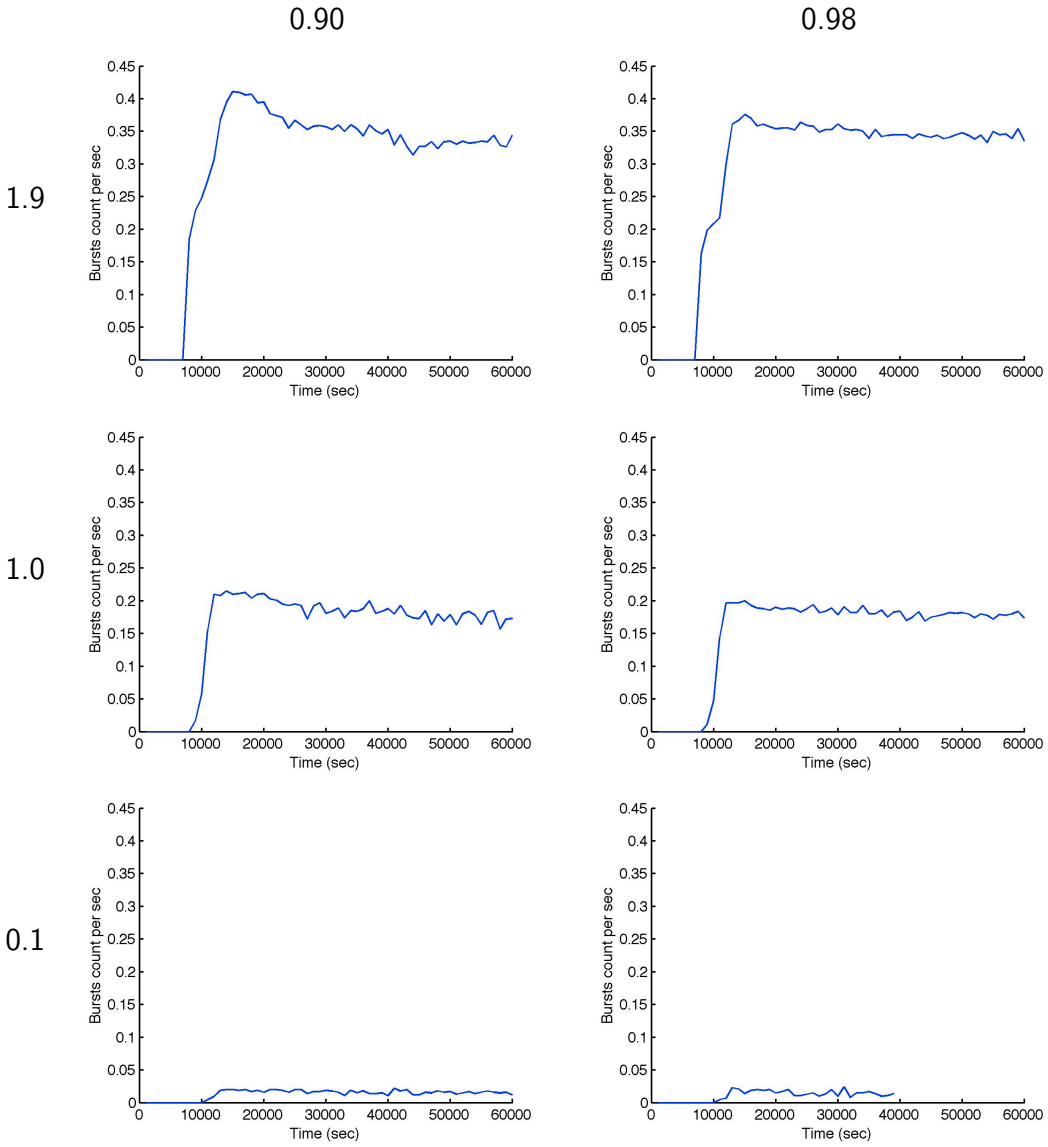

Figure 5.20: Evolution of burst counts with parameters (target rate, fraction excitatory cells) = (0.1, 0.90), (1.0, 0.90), (1.9, 0.90), (0.1, 0.98), (1.0, 0.98), and (1.9, 0.98).

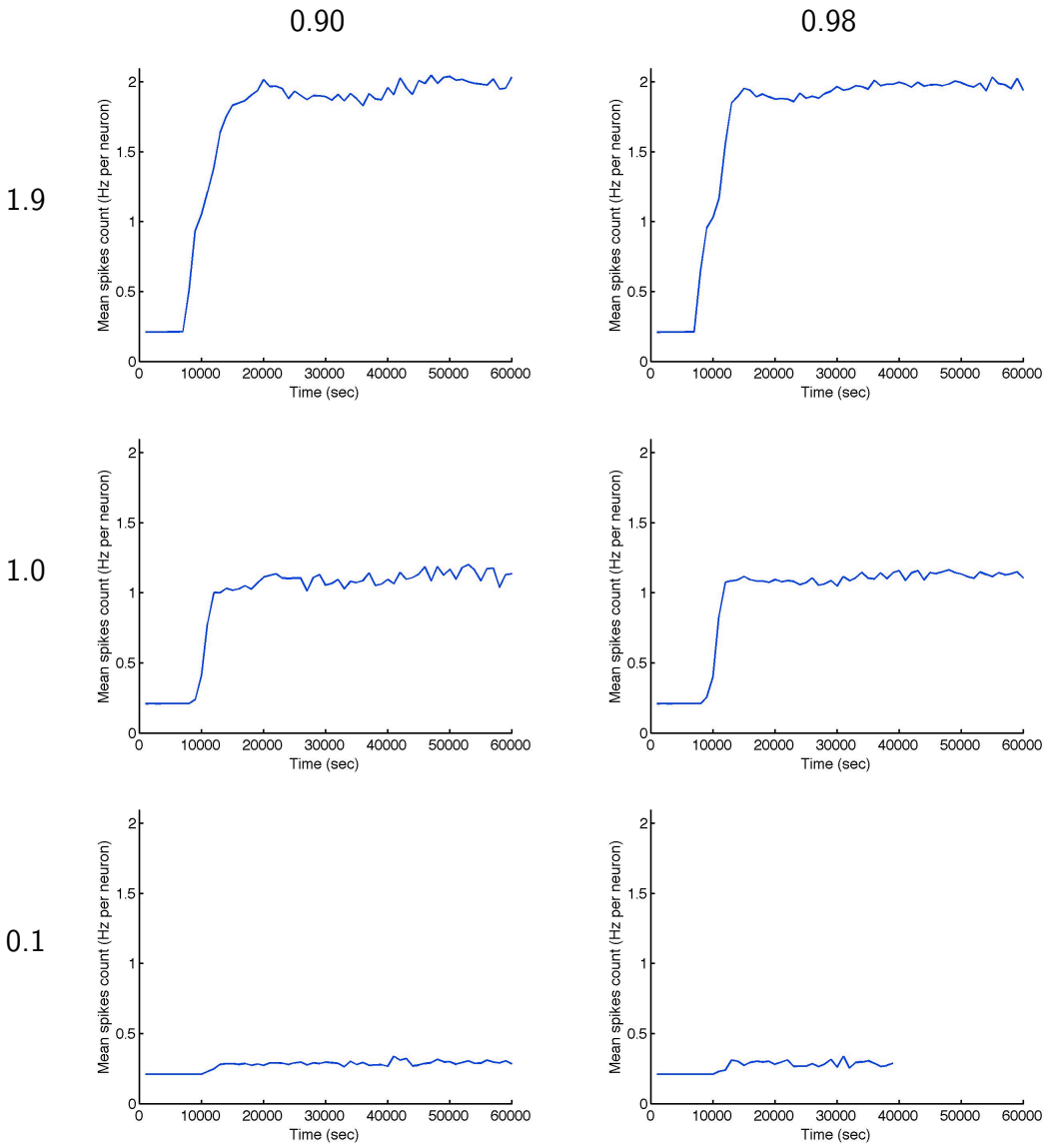

Figure 5.21: Evolution of APNFR with parameters (target rate, fraction excitatory cells) = (0.1, 0.90), (1.0, 0.90), (1.9, 0.90), (0.1, 0.98), (1.0, 0.98), and (1.9, 0.98).

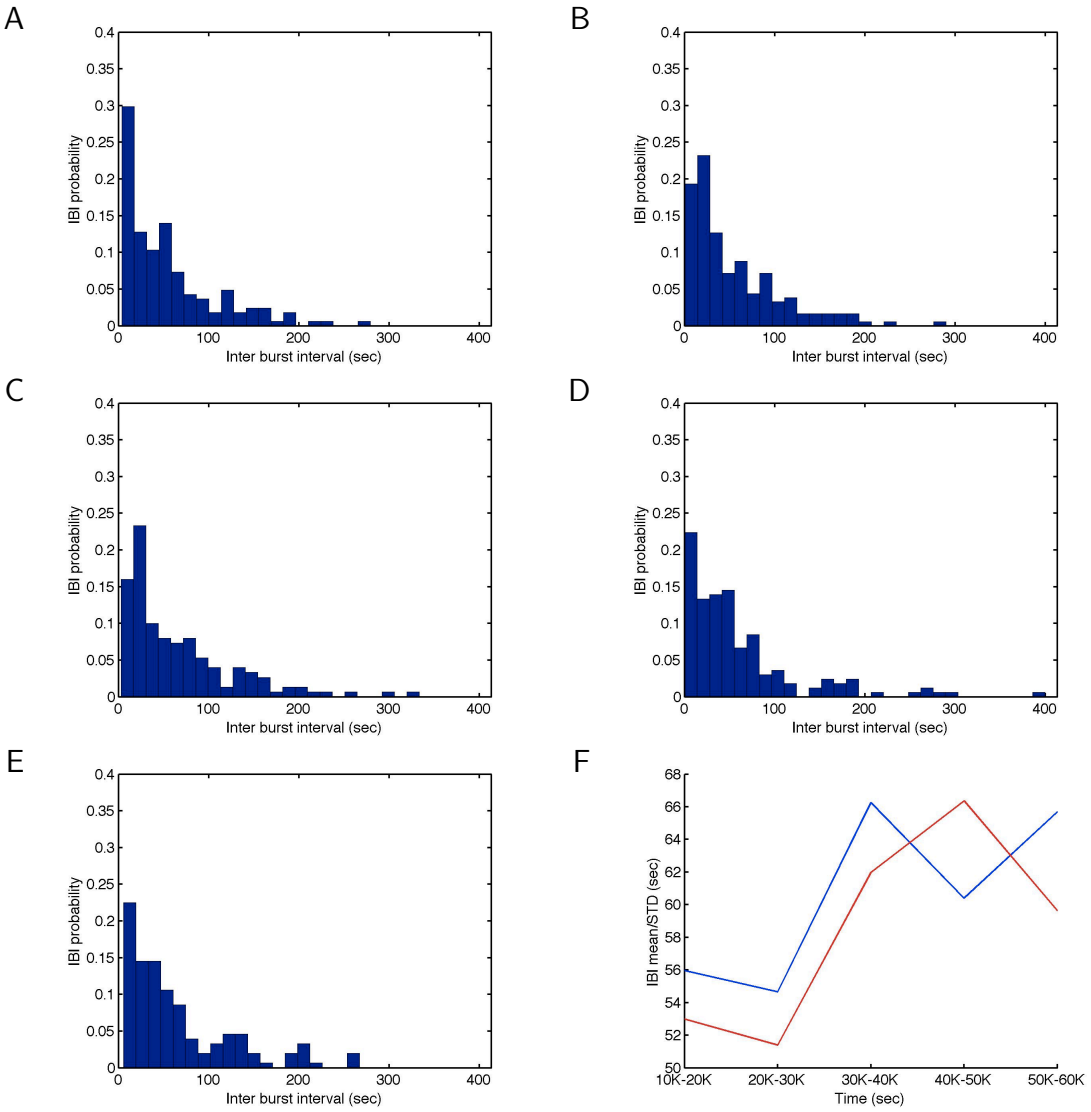

Figure 5.22: Distributions histograms of IBIs between 10,000—20,000 sec (A)(μ = 55.96, σ = 52.98, max = 282.58, min = 3.75), 20,000—30,000 sec (B)(μ = 54.66, σ = 51.39, max = 292, min = 0.76), 30,000—40,000 sec (C)(μ = 66.27, σ = 61.96, max = 339.25, min = 2.96), 40,000—50,000 sec (D)(μ = 60.40, σ = 66.36, max = 412.25, min = 0.18), 50,000—60,000 sec (E) (μ = 65.68, σ = 59.63, max = 267.07, min = 5.42) simulation times with parameter (target rate, fraction excitatory cells) = (0.1, 0.90). Evolution of mean (blue) and STD (red) (F).

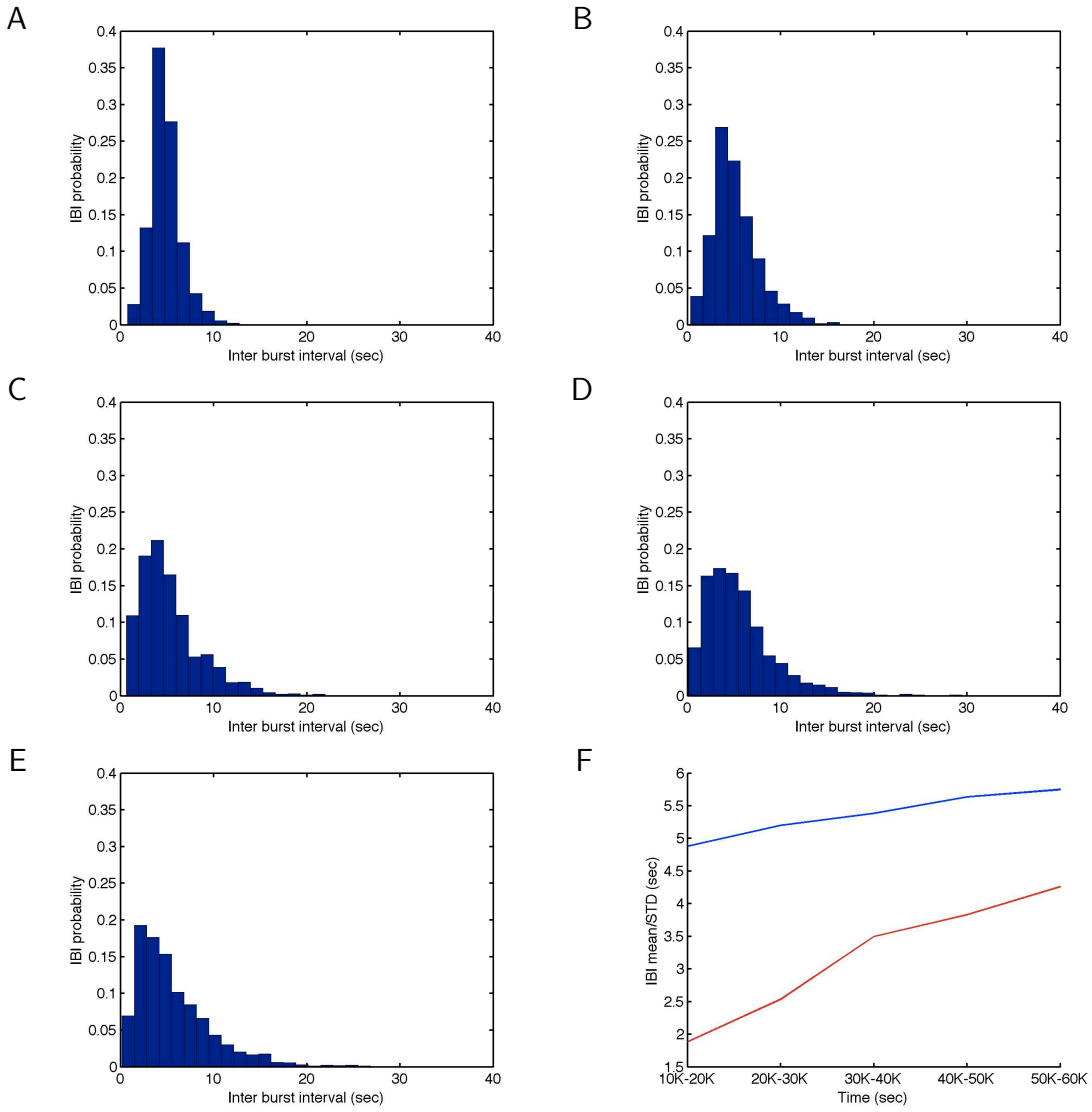

Figure 5.23: Distributions histograms of IBIs between 10,000—20,000 sec (A)($\mu = 4.88$, $\sigma = 1.88$, $max = 38.08$, $min = 0.75$), 20,000—30,000 sec (B)($\mu = 5.20$, $\sigma = 2.54$, $max = 21.42$, $min = 0.33$), 30,000—40,000 sec (C)($\mu = 5.38$, $\sigma = 3.50$, $max = 31.54$, $min = 0.62$), 40,000—50,000 sec (D)($\mu = 5.63$, $\sigma = 3.83$, $max = 29.62$, $min = 0.10$), 50,000—60,000 sec (E) ($\mu = 5.75$, $\sigma = 4.26$, $max = 39.02$, $min = 0.13$) simulation times with parameter (target rate, fraction excitatory cells) = (1.0, 0.90). Evolution of mean (blue) and STD (red) (F).

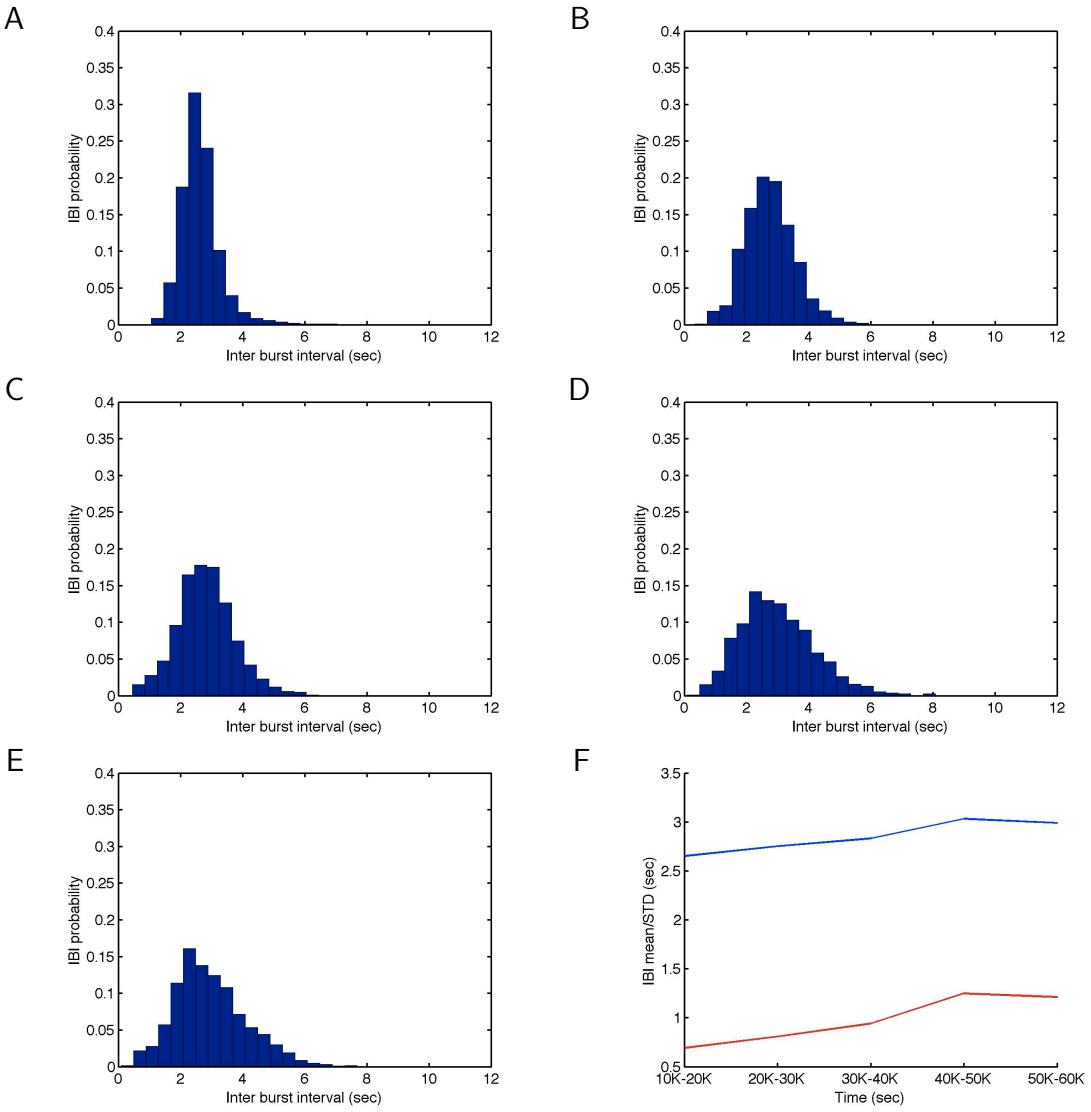

Figure 5.24: Distributions histograms of IBIs between 10,000—20,000 sec (A)($\mu = 2.65$, $\sigma = 0.69$, $max = 8.28$, $min = 0.25$), 20,000—30,000 sec (B)($\mu = 2.75$, $\sigma = 0.81$, $max = 6.71$, $min = 0.33$), 30,000—40,000 sec (C)($\mu = 2.83$, $\sigma = 0.94$, $max = 9.53$, $min = 0.45$), 40,000—50,000 sec (D)($\mu = 3.03$, $\sigma = 1.25$, $max = 9.84$, $min = 0.09$), 50,000—60,000 sec (E) ($\mu = 2.99$, $\sigma = 1.21$, $max = 11.09$, $min = 0.09$) simulation times with parameter (target rate, fraction excitatory cells) = (1.9, 0.90). Evolution of mean (blue) and STD (red) (F).

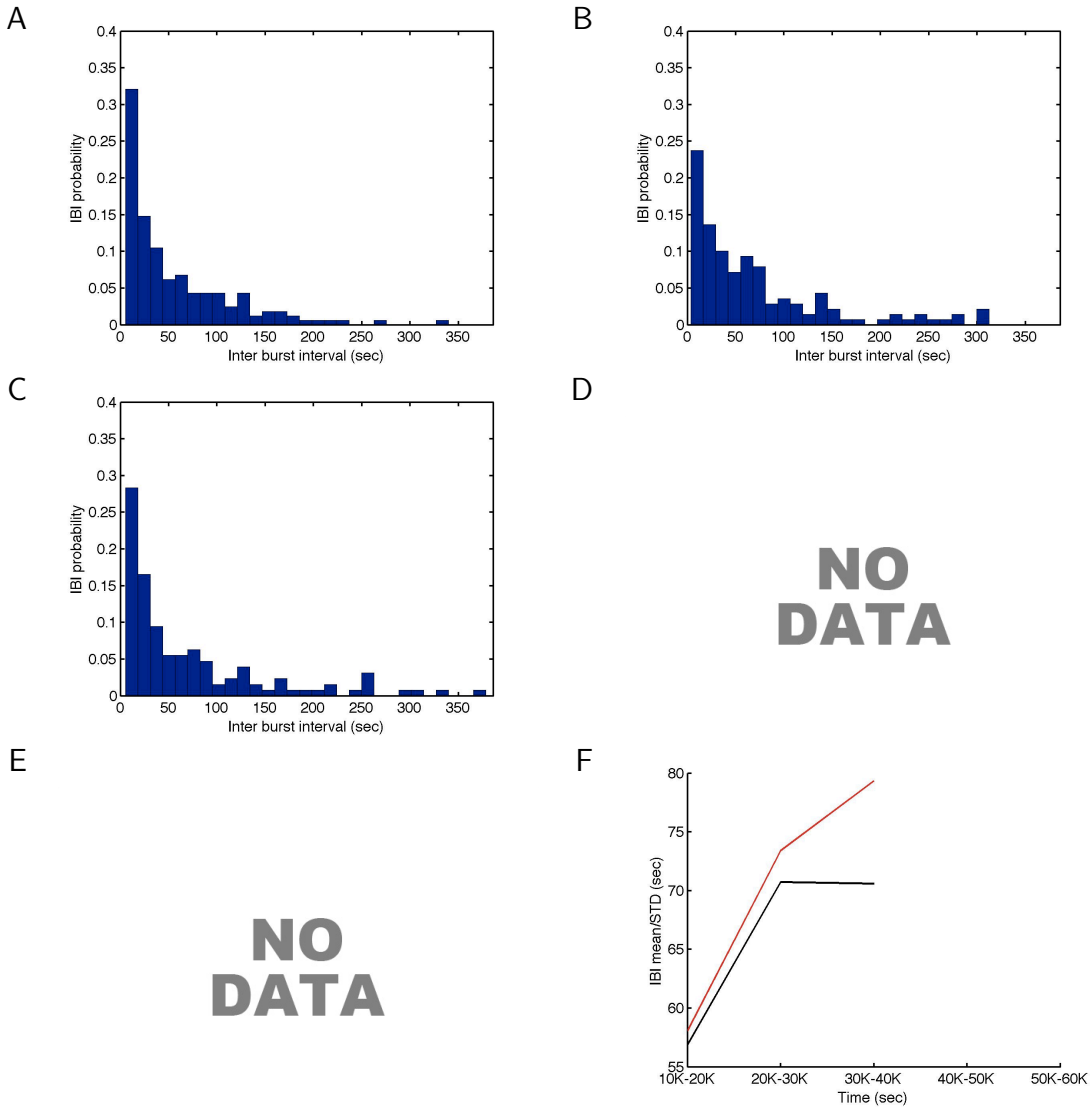

Figure 5.25: Distributions histograms of IBIs between 10,000—20,000 sec (A)($\mu = 5.30$, $\sigma = 2.13$, $max = 38.08$, $min = 0.18$), 20,000—30,000 sec (B)($\mu = 5.37$, $\sigma = 2.46$, $max = 19.18$, $min = 0.33$), 30,000—39,000 sec (C)($\mu = 5.45$, $\sigma = 2.86$, $max = 24.42$, $min = 0.21$), 40,000—50,000 sec (D)($\mu = 5.63$, $\sigma = 3.55$, $max = 30.82$, $min = 0.42$), 50,000—60,000 sec (E) ($\mu = 5.62$, $\sigma = 3.58$, $max = 28.77$, $min = 0.54$) simulation times with parameter (target rate, fraction excitatory cells) = (0.1, 0.98). Evolution of mean (blue) and STD (red) (F).

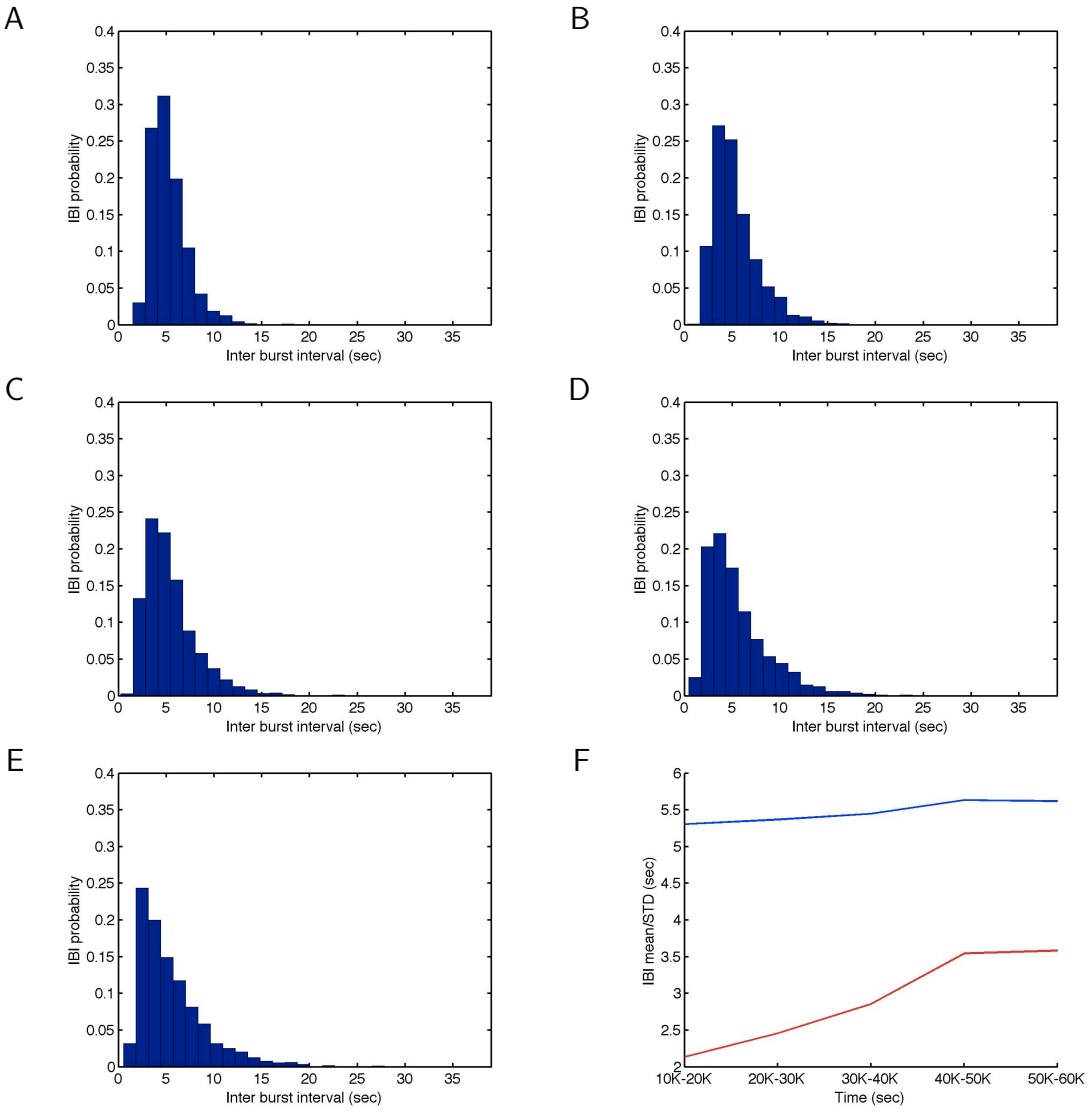

Figure 5.26: Distributions histograms of IBIs between 10,000—20,000 sec (A)($\mu = 5.30$, $\sigma = 2.13$, $max = 38.08$, $min = 0.18$), 20,000—30,000 sec (B)($\mu = 5.37$, $\sigma = 2.46$, $max = 19.18$, $min = 0.33$), 30,000—40,000 sec (C)($\mu = 5.45$, $\sigma = 2.86$, $max = 24.42$, $min = 0.21$), 40,000—50,000 sec (D)($\mu = 5.63$, $\sigma = 3.55$, $max = 30.82$, $min = 0.42$), 50,000—60,000 sec (E) ($\mu = 5.62$, $\sigma = 3.58$, $max = 28.77$, $min = 0.54$) simulation times with parameter (target rate, fraction excitatory cells) = (1.0, 0.98). Evolution of mean (blue) and STD (red) (F).

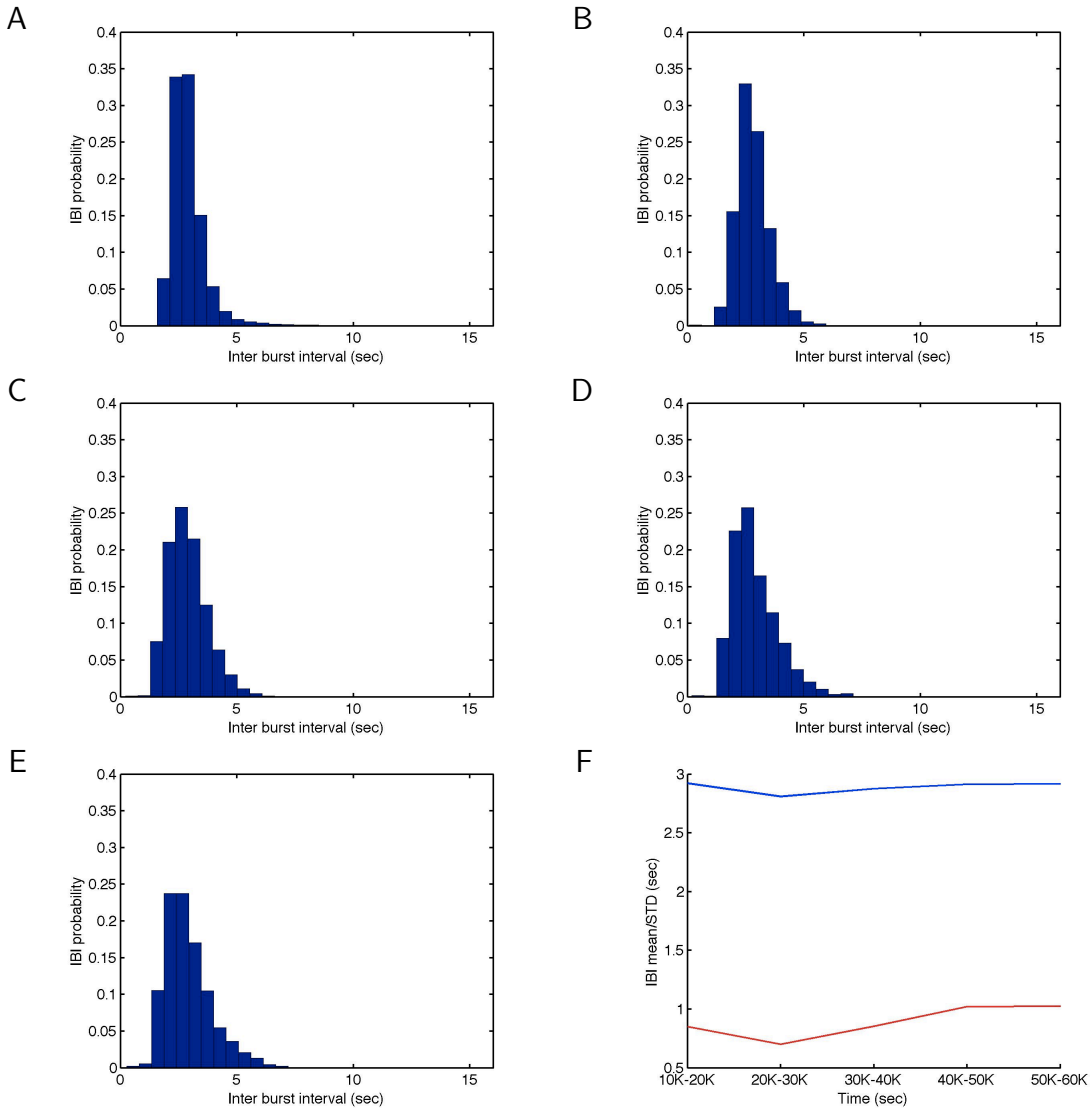

Figure 5.27: Distributions histograms of IBIs between 10,000—20,000 sec (A)(μ = 2.922, σ = 0.8518, max = 15.17, min = 1.57), 20,000—30,000 sec (B)(μ = 2.8096, σ = 0.7018, max = 6.09, min = 0.8), 30,000—40,000 sec (C)(μ = 2.8762, σ = 0.8535, max = 7.53, min = 0.21), 40,000—50,000 sec (D)(μ = 2.9136, σ = 1.0212, max = 9.84, min = 0.18), 50,000—60,000 sec (E) (μ = 2.917, σ = 1.0242, max = 9.16, min = 0.26) simulation times with parameter (target rate, fraction excitatory cells) = (1.9, 0.98). Evolution of mean (blue) and STD (red) (F).

Chapter 6

ANALYSIS

6.1 Influence of Simulation Parameters on Network Connectivity and Behavior

Intra-burst profiles and inter-bursts profiles are emergent properties of network development, which seem to be a manifestation of underlying network structure (connectivity). Here, we analyze the relationship between simulation parameters and burst profiles first. Then, we look at the effects of parameters on network connectivity. Finally, we will show an overview of the relationship between simulation parameters (independent variables), network structure (connectivity), and burst profiles (emergent properties).

6.1.1 Intra-burst Profiles

For intra-burst profiles (burst shape), target rate had a major influence; lower target rate (0.1) induced narrower and more intense bursts, whereas higher target rates (1.0 and 1.9) induced wider and less intense bursts; the difference between target rates 1.0 and 1.9 was not remarkable (Figures 5.17–5.18 and Table 6.1). The fraction of excitatory cells also had an effect on the intra-burst profiles in those figures: lower fraction of excitatory cells (more inhibittory cells) tended to induce narrower and more intense bursts. However, Table 6.1 shows one exceptional result, where simulation parameter was (target rate, fraction excitatory cells) = (0.1, 0.98); that exhibited much more intense and narrower bursts between 38,000 and 39,000 sec simulation time. Interestingly, number of spikes per burst did not appear to depend on these simulation parameters (Figures 5.19).

Table 6.1: Mean burst height (Hz per neuron) and burst width (sec) between 38,000–39,000 sec (top values) and between 59,000–60,000 sec (bottom values).

target rate / fraction excitatory cells	0.90	0.98
1.9	86.0271, 0.1452	76.3293, 0.1777
1.0	83.6281, 0.1439	76.1085, 0.1754
0.1	83.6420, 0.1160	162.6593, 0.0971
1.9	102.1598, 0.1485	80.5281, 0.1788
1.0	95.1390, 0.1515	72.7651, 0.1867
0.1	169.1908, 0.1050	—

6.1.2 Inter-burst Profiles

For inter-burst profiles (IBI distribution), target rate determined the average IBI: lower target rate induced longer IBIs (fewer bursts per unit time), and higher target rate induced shorter IBIs (more bursts) (Figure 5.20) so that APNFR converged on the target rate (Figure 5.21). Both target rate and fraction of excitatory cells had an effect on the regularity of IBIs; lower target rate or lower fraction of excitatory cells induced irregular bursting, whereas higher target rate or higher fraction of excitatory cells induced more regular bursting (Figures 5.22–5.27, F). Table 6.2 summarizes coefficient of variation (CV, σ/μ) of IBI distribution between 30,000 and 39,000 simulation time, and between 50,000 and 60,000 simulation time; for each target rate, decreasing the fraction of excitatory cells increased CV (increasing irregularity). The table also shows the strong effect of decreasing target rate. Similar to the intra-burst profiles, it exhibited a larger CV for higher fraction of excitatory cells, where simulation parameter was (target rate, fraction excitatory cells) = (0.1, 0.98), and this was an exception to the overall pattern.

6.1.3 Connecting Patterns

In Figures 5.2–5.7, one notices that simulation parameters affected the connectivity pattern of neurons in networks as follows: target rate affected the neurite outgrowth patterns, and fraction of excitatory cells affected the variation of neurite radii respectively. These

Table 6.2: Coefficient of variation (CV) of IBI distribution during 30,000–39,000 sec (top values) and during 50,000–60,000 sec (bottom values).

target rate / fraction excitatory cells	0.90	0.98
1.9	0.3228	0.2964
1.0	0.6491	0.5178
0.1	0.9786	1.1235
1.9	0.4052	0.3511
1.0	0.7407	0.6378
0.1	0.9078	—

figures show that neurite outgrowth rate independent of target rate; for the lowest target rate (0.1), outgrowth continued until larger final radii were reached. Therefore, the final neurite radii of networks with the lowest target rate (0.1) were largest (greater connectivity). Table 6.3—Table 6.5 summarize mean values, standard deviation (STD), and range (the difference between the maximum and the minimum) of neurite radii at 39,000 sec and 60,000 sec of ordinary neurons (excitatory, non-spontaneously active, interior) for each parameter. Table 6.3 indicates that the final neurite radii of networks with lower fraction of excitatory cells (more inhibitory cells) were also larger. Table 6.4 indicates that networks with higher fraction of excitatory cells or higher target rate induced wider variation of final neurite radii (bigger STD). However, Table 6.5 reveals that networks with lower fraction of excitatory cells or higher target rate induced wider range of final neurite radii: increasing fraction of excitatory cells increased the STD, but decreased the range of final neurite radii. Figure 6.1 shows distributions of final neurite radii. As seen in the figure, distributions of lower fraction of excitatory cells show wider variabilities with some extreme values, whereas distributions of higher fraction of excitatory cells show bimodal distributions, which might produced the apparent bigger STDs.

6.1.4 Relationship between Connecting Patterns and Bursting Behavior

The neurite outgrowth patterns and the evolution of mean burst height resemble each other in the sense that they continue to grow for a longer time at the lowest target rate (0.1)

Table 6.3: Mean neurite radii of ordinary neurons at 39,000 sec (top values) and at 60,000 sec (bottom values) (excitatory, non-spontaneously active, interior).

target rate / fraction excitatory cells	0.90	0.98
1.9	2.1137	1.8296
1.0	2.0857	1.8247
0.1	2.3833	2.3627
1.9	2.0990	1.8334
1.0	2.1235	1.8229
0.1	2.8485	—

Table 6.4: Standard deviation of neurite radii of ordinary neurons at 39,000 sec (top values) and at 60,000 sec (bottom values) (excitatory, non-spontaneously active, interior).

target rate / fraction excitatory cells	0.90	0.98
1.9	0.1029	0.1319
1.0	0.1073	0.1480
0.1	0.0687	0.1029
1.9	0.1305	0.1287
1.0	0.1299	0.1388
0.1	0.0897	—

Table 6.5: Range (the difference between the maximum and the minimum) of neurite radii of ordinary neurons at 39,000 sec (top values) and at 60,000 sec (bottom values) (excitatory, non-spontaneously active, interior).

target rate / fraction excitatory cells	0.90	0.98
1.9	0.8698	0.5229
1.0	0.9389	0.5793
0.1	0.5712	0.3644
1.9	1.8765	1.7086
1.0	1.6707	0.5646
0.1	0.8986	—

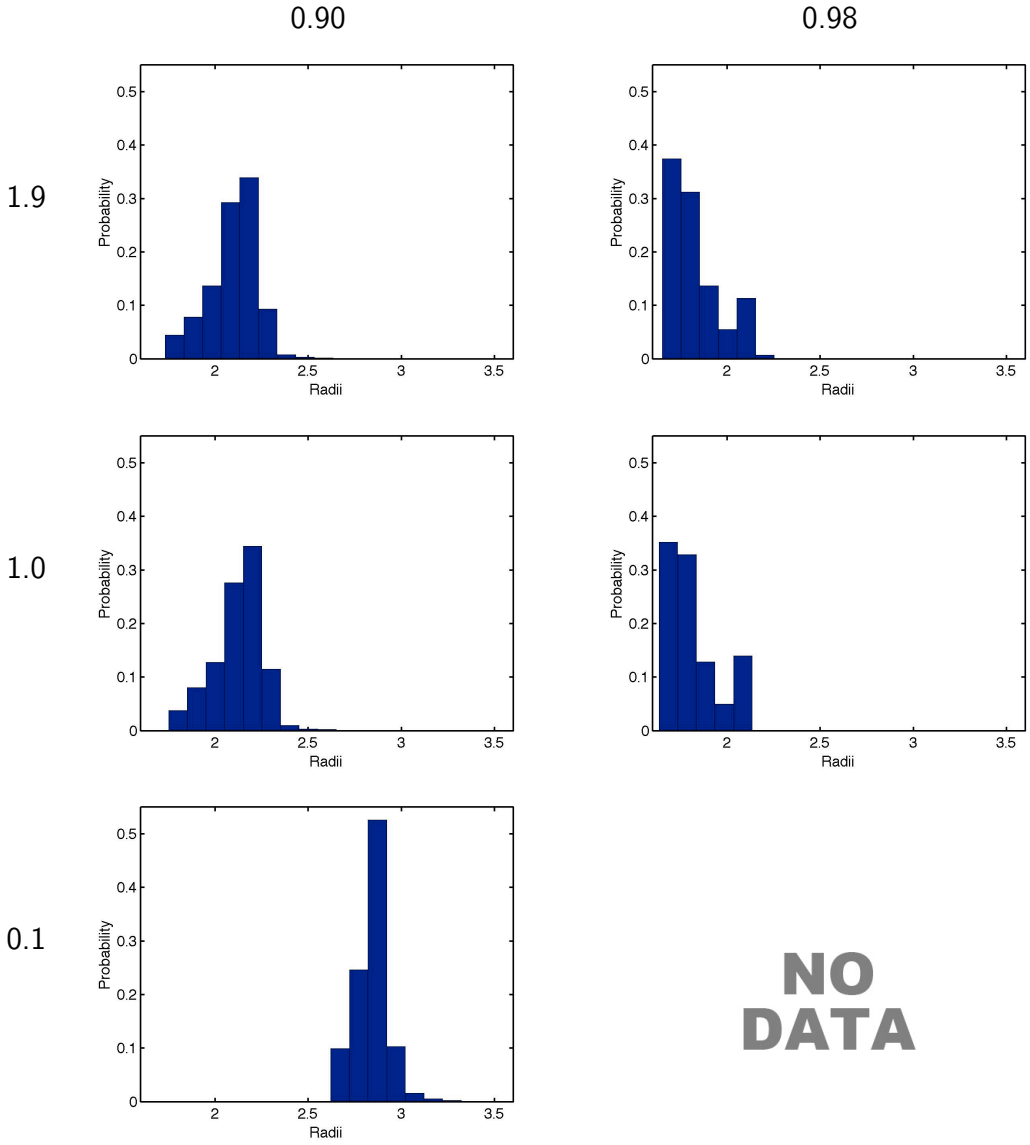

Figure 6.1: Distribution histograms of neurite radii at 60,000 sec with parameters (target rate, fraction excitatory cells) = (0.1, 0.90), (1.0, 0.90), (1.9, 0.90), (0.1, 0.98), (1.0, 0.98), and (1.9, 0.98).

(Figure 5.17). Because burst heights are inversely proportional to burst widths, the neurite outgrowth patterns and the evolution of mean burst width are also interrelated (Figure 5.18): as connectivity radii increased, burst widths decreased. Variabiity of neurite radii might be related to the irregularity of IBIs. Here, we could assume that the connectivity pattern of neurons in networks ended up having an influence on burst profiles.

In these simulations, bursting is a phenomenon that emerges from the interactions of a huge number of neurons and synapses; therefore, larger connection radii may generate greater temporal synchronization. That could explain the relationship between size of neurites and burst shape; that is, bigger neurites made stronger connections between neurons, and eventually this generated stronger synchronized behavior, narrower and more intense bursts. Also, we can hypothesize that homogeneity of connectivity had an effect on regularity of IBIs; that is, homogeneous connections between neurons generated regular bursting, and heterogeneous connections between neurons generated irregular bursting.

However, this hypothesis has some drawbacks: (1) Exceptional values of intra-bursts profiles and inter-bursts profiles, where simulation parameter was (target rate, fraction excitatory cells) = (0.1, 0.98), could not be explained, even though the connectivity pattern took the form of the expected; (2) The irregularity of IBIs and the variability of neurite radii were positive relationship with the same target rate; however, target rate had stronger effect on the irregularity of IBIs; that is, as target rate decreased, the variability of neurite was also decreased, but the irregularity of IBIs increased.

6.2 Comparison to Previous Results

6.2.1 Activity Dependent Neurite Outgrowth Model

There have been a number of detailed studies of activity dependent neurite outgrowth models, where both neuronal morphology and behavior in networks were investigated. For one morphological study, the effect of inhibition on the development of connectivity was investigated, in which simulations of networks with 32 excitatory cells and 4 inhibitory cells

were performed, varying the inhibitory-to-excitatory synaptic strength [40]. Simulations showed a transient overproduction in total network connectivity (overshoot). Stronger inhibition and more regular spatial distribution of inhibitory cells increased this overshoot. The strength and the distribution pattern of inhibitory cells also had effects on connectivity at equilibrium; that is, when inhibition was strong, excitatory connectivity levels in the stable network were higher, and connectivity from excitatory-to-excitatory cells was higher at equilibrium when inhibitory cells were clustered. However, if inhibition was too strong, the network would increase its connectivity indefinitely. Furthermore, higher inhibition played an important role in increasing variability among individual cells with respect to the developmental course of their field size and firing behavior.

For the behavioral study, behaviors for both outgrowth and electric activity in a purely excitatory network composed of cells with different neurite outgrowth properties were investigated [39]. Simulations were performed on relatively small sized networks such as 16 cells placed on a grid, in which the spatial distribution of the cells and the distribution of the outgrowth properties were varied. The general emergent properties of such a network were overshoot and oscillations of outgrowth and electrical activity on the timescale of neurite outgrowth. Though such periodic behaviors were observed in networks with identical outgrowth properties, the spatial distribution of the cells and distribution of the outgrowth property affected the kinds of complex periodic behaviors observed, both in terms of electrical activity and connectivity of individual cells.

Several of these results are consistent with ours: (1) The neurite fields of inhibitory cells tended to become smaller; (2) Network with strong inhibition (lower fraction of excitatory cells) had larger final neurite radii; (3) The variation of the final neurite radii in our simulation could be explained by the fact that inhibition generates variability among cells; and (4) The periodic behavior, whose timescale was neurite outgrowth, reminded us of the bursting patterns in small networks (10x10). One of the essentials for its occurrence was a hysteresis relationship between the slow variable (growth rate) and the average electrical

activity in the network [3, 5]. In the activity-dependent neurite outgrowth model, hysteresis was present in the relationship between average connection strength and average membrane potential [39]. However, such a periodic behavior was not observed in large networks. The cause of abolition of hysteresis in our large networks has not yet been addressed. Also, the transient overproduction of connections (overshoot) was not obvious in our results (some overshoot were observed in inhibitory cells).

6.2.2 Cultured Dissociated Cortical Cells

Many studies of investigating network burst firing patterns in living cultured dissociated cortical cells on multi-electrode arrays have been done. Experiments using dissociated rat cortex cells cultured on planar multi-electrode plates demonstrated age dependent network burst patterns in average firing rate profiles of total network activity and of individual sites were recorded at different ages [29, 28]. In general, these experiments showed that: (1) Spontaneous action potential discharges began about the 2nd week in vitro (WIV); (2) The initially short and slightly skewed bursts lasting on the order of 1-2 s evolved, during the 3rd WIV, into long-lasting bursts of about 6 s with almost symmetrical firing rate profiles; (3) At later ages (about 4th WIV), network bursts tightened up to about 200 msec. Bursts at this age were characterized by highly synchronized onsets, reaching peak firing levels within less than 60 ms — this pattern persisted for the rest of the culture period; and (4) Throughout the recording period, active sites showed highly persistent temporal relationships within network bursts.

The remarkable difference between these and our results was the evolution of initially short bursts into ones with prolonged duration during the 3rd WIV (our model generated initially wide, low bursts that gradually developed into narrow and more intense bursts). In the living preparation, neocortical cell cultures pass through a period of delayed development of synaptic inhibition relative to excitation [30, 38]. Increasing inhibitory feedback later on might have the effect of prolonging the bursts [28]. Since our model applied the same

growth rate to excitatory and inhibitory cells, this might be one of the factors that made the difference. The burst durations at later ages (about 4th WIV) were about 200 msec, which were in line with our results. However, the maximum firing rate within bursts were about 5-6 spikes for all 60 electrodes within a 10 msec time bin, which corresponds to 8.3-10 Hz per neuron (under the assumption that one electrode monitored one neuron's activity), one order of magnitude less than our results (Figure 5.17).

6.2.3 Inter-burst Intervals in Spontaneous Activity

Analysis of inter-burst profiles in spontaneous activity of cortical neuronal culture was reported in a study in which experimental values were compared to model simulations [8]. In the models, the number of neurons in networks varied between 500 and 5,000. From experimental data, all available recordings were divided into four groups according to burst count means (multimodal distribution), each of which yielded smooth histograms described by fitted GEV (generalized extreme value) distributions. The results obtained from small networks, or larger networks with homogeneous synaptic strength, could not reproduce results from in vitro experiments. Alternatively, noise-driven network models that included a set of intense neurons with stronger connections to their targets yielded realistic IBIs.

The GEV distribution is a family of continuous probability distributions developed with extreme value theory to combine type I (Gumbel), type II (Frechet) and type III (Weibull) extreme value distributions. Distributions whose tails decrease exponentially, such as normal distributions, lead to the type I. Distributions whose tails decrease as a polynomial, such as Students t, lead to the type II. Distributions whose tails are finite, such as the beta, lead to the type III. The GEV distribution is often used to model the extreme value among a large set of independent, identically distributed random values representing measurements or observations. The GEV distribution is characterized by three parameters: shape parameter (ξ), scale parameter (σ), and location parameter (μ). The ξ parameter determines the form of GEV distribution: $\xi = 0$, type I; $\xi > 0$, type II; and $\xi < 0$, type III [33]. Though IBIs of

living preparations yielded GEV distributions (type II) [8], the underling mechanism is not known at this time.

We used the MATLAB *mle()* function to get maximum likelihood estimates (MLEs) for the parameters of GEV distributions using the results of our simulations. Table 6.6 shows GEV distribution parameters (including 95% confidence intervals for the parameters) derived by the *mle()* function from data of IBIs distribution between 50,000 and 60,000 sec simulation time of each simulation. Then, we used MATLAB *gevpdf()* (generalized extreme value probability density function) to plot probability density function of the GEV distribution with parameters returned by the *mle()* function overlapping distribution histograms of IBIs (50,000–60,000 sec simulation time, bin size is 1 sec) to visually evaluate the distribution pattern of our results (Figure 6.2).

As shown in Figure 6.2, IBIs distributions were well fit by GEV (the IBIs distributions of lowest target rate exhibited great variability probably due to fewer number of sampling data for each bin). From GEV distribution parameters (Table 6.6), we observed the following: (1) Target rate had major effects on these three parameters — lower target rate induced higher ξ value (longer tail), higher σ value (more irregular), and higher μ value (longer IBI); (2) Fraction of excitatory cells also had an effect on σ — lower fraction of excitatory cells (higher inhibition) induced higher σ value (more irregular); and (3) The effect of fraction of excitatory cells on other two parameters (ξ and μ) were not statistically significant. The values of these parameters were in the range of those values from living preparations [8].

Table 6.6: GEV distribution parameters (including 95% confidence intervals for the parameters) between 50,000 and 60,000 sec simulation time of each simulation.

(target rate, fraction excitatory cells)	ξ	σ	μ
(1.9, 0.90)	-0.0741 (-0.0926, -0.557)	1.0366 (1.0100, 1.0639)	2.4695 (2.4310, 2.5079)
(1.0, 0.90)	0.2628 (0.2147, 0.3109)	2.4257 (2.3165, 2.5400)	3.6092 (3.4749, 3.7436)
(0.1, 0.90)	0.5401 (0.3064, 0.7739)	27.6732 (22.7352, 33.6837)	31.1553 (25.5226, 36.7880
(1.9, 0.98)	0.0063 (-0.0141, 0.0268)	0.7824 (0.7621, 0.8033)	2.4605 (2.4317, 2.4893)
(1.0, 0.98)	0.2879 (0.2427, 0.3330)	1.9881 (1.8996, 2.0806)	3.7917 (3.6844, 3.8990)

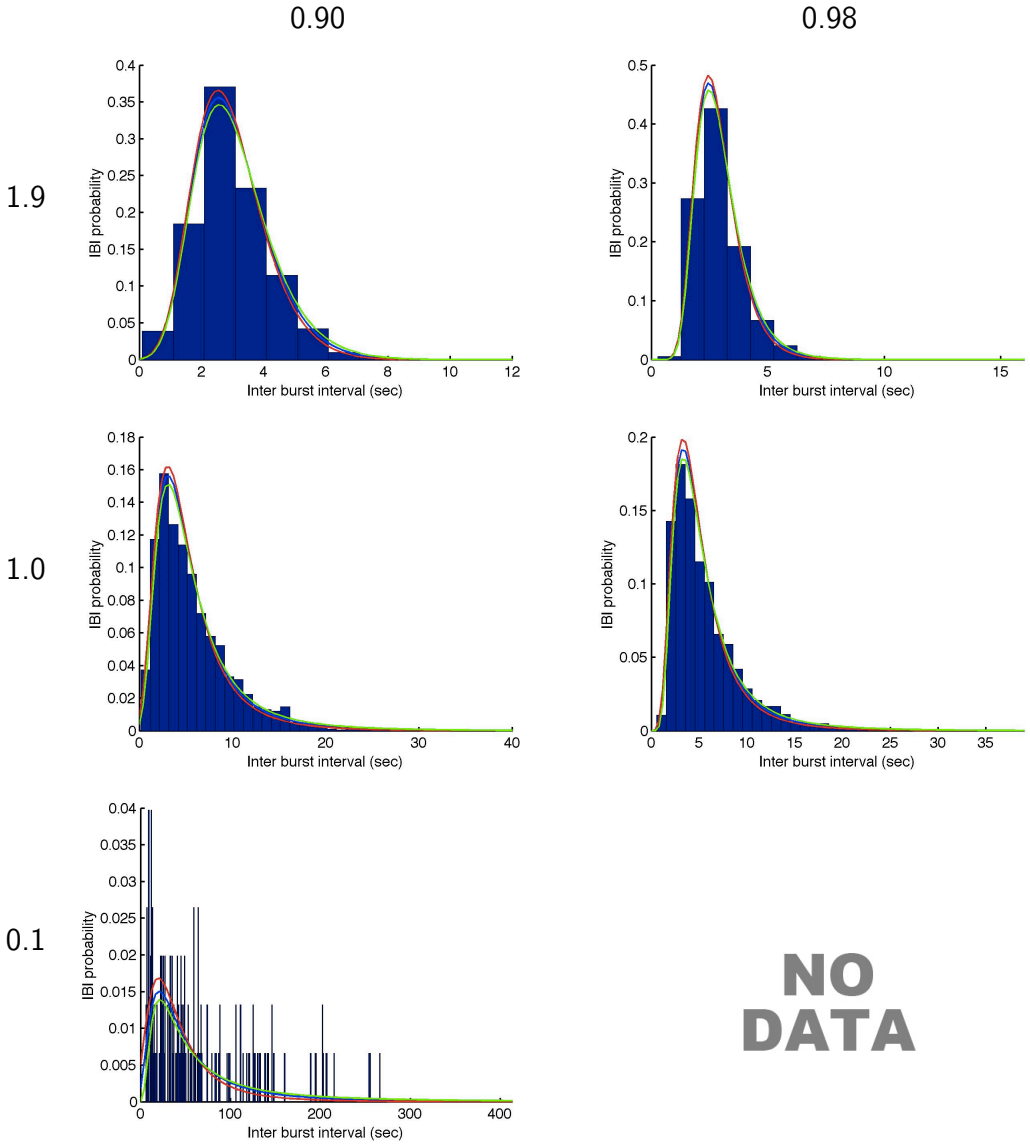

Figure 6.2: Distribution histograms of IBIs between 50,000–60,000 sec and PDF plots (including 95% confidence intervals, green and red) with parameters (target rate, fraction excitatory cells) = (0.1, 0.90), (1.0, 0.90), (1.9, 0.90), (0.1, 0.98), (1.0, 0.98), and (1.9, 0.98).

Chapter 7

CONCLUSION AND FUTURE WORK

We have demonstrated that our simulation model can generate bursts, and burst profiles were controlled by simulation parameters (target rate and fraction of excitatory cells). We found that burst profiles and the connectivity patterns (size of neurites and variability of them) of networks were interrelated. However, we also realized limitations of the current simulation model and the method: our simulation model and the combination of parameters could only reproduce the subset of bursts patterns of living preparations. To tackle the limitation, we need to address the following: (1) Parameter space — We only investigated the 2-dimensional parameter space consisting of target rate and fraction of excitatory cells (we also explored only a limited sample of parameter values). However, some properties were not affected by these parameters; for example, the number of spike counts included per burst (Figure 5.19) was not affected by varying simulation parameters; (2) Growth parameters — We used the same values of growth parameters, outgrowth rate constant (ρ) and null point for outgrowth (ϵ), regardless of type of neuron. However, some empirical observations reveal that the critical level of electrical activity above which the neurite of a cell retract (ϵ) is in fact different for different classes of neurons [10, 16, 15]; (3) Clustering — We used the set of standardized networks (Figure 5.1), repeating them to fill the 100x100 arrangements. However, this method could not construct network level heterogeneous structures. For example, the human cortex show homogeneous connectivity, with additional strongly heterogeneous projections from one area to another [14]. Because complex periodic behaviors were the emergence of underlying complexity of the spatial distribution of the cells and their properties [39], the homogeneity of the network of our model had limitation.

In addition to the limitation of current model and method, we also have left undone,

mostly due to time constraints. Firstly, we only did limited quantitative analysis of the dependency of bursting on parameters, due to insufficient data. If we had more data, we might be able to predict burst profiles from new set of parameters. Secondly, we did not investigate spatio-temporal firing patterns in the network, where individual neuron firing activity was recorded along simulation time. This spatio-temporal analysis is common in investigating spontaneous firing in living preparations, and it can provide more information of underlying network structure. There were CPU memory limitations that prevented us from saving all neuron activity records, but we could have chosen a subset of neurons and recorded their activities. Finally, we did not investigate effects when network size was changed.

Based on the model and results, we will pursue deeper understanding of behaviors of neuronal networks. We need to reconsider the way of investigation, and, therefore, we need better tools: (1) Memory limitations — A GPU has 3–4 GB global memory. For performance reasons, the global memory space where all neurons and synapses data were stored was allocated at initialization and kept throughout the simulation. Therefore, the number of neurons and synapses were restricted by the size of global memory; (2) Performance — The GPU performance is restricted by processing performance (MIPS) and memory bandwidth. However, we cannot have excessive expectations that improvement of GPU architecture achieves remarkable performance improvement. One practicable solution is to utilize multi-GPUs for executing a single simulation. Solving these issues presents technical challenges, but it will allow us to handle bigger networks. To add new features is another challenge. One of such features is an ability of simulations with external stimuli during development, through which we can observe effects on network structure, burst abolition, and ability to perform computation tasks upon maturity.

BIBLIOGRAPHY

[1] Tesla C1060 computing processor board specification. Technical Report BD-04111-001_v06, NVIDIA Corporation, 2010.

[2] *CUDA C Best Practice Guide*. NVIDIA Corporation, 2011.

[3] A. Babloyanz and A. Destexhe. Mapping of spatiotemporal acivity of networks into chaotic dynamics: thalamocortical networks. *Artificial Neural Networks*, pages 139–144, 1991.

[4] C.S. Cohan and S.B. Kater. Suppression of neurite elongation and growth cone motility by electrical activity. *Science*, (232):1638–1640, 1986.

[5] A. Destexhe and P. Gaspard. Bursting oscillations from a homoclininc tangency in a time delay system. *Physics Letters*, 173:386–39, 1993.

[6] Diego Echevarría and Klaus Albus. Activity-dependent development of spontaneous bioelectric activity in organotypic cultures of rat occipital cortex. *Developmental Brain Research*, 123:151–64, 2000.

[7] R.D. Fields, E.A. Neale, and P.G. Nelson. Effects of patterned electrical activity on neurite outgrowth from mouse neurons. *The ournal of Neuro Science*, (10):2950–2964, September 1990.

[8] T. Gritsun, J. le Feber, J. Stegenga, and W. L. C. Rutten. Experimental analysis and computational modeling of inerburst intervals in spontaneous activity of cortical neuronal culture. *biol. cybern*, 105:197–210, 2011.

[9] S. Grumbacher-Reinert and J. Nicholls. Influence of subtrate on retraction of neurites following electrical activity of leech retzius cells in culture. *The Journal of Experimental Biology*, (167):1–14, 1992.

[10] P. B. Guthrie, M. P. Mattson, L. R. Mills, and S. B. Kater. Calcium homeostasis in moluscan and mammmalian neuron: neuron-selective set-point of calcium test concentration. *Soc. Neurosc. Abstr.*, (14):582, 1988.

[11] Bing Han and Tarek M. Taha. Acceleration of spiking neural network based pattern recognition on nvidia graphics processors. *APPLIED OPTICS*, 49(10):B83–B91, April 2010.

[12] Yasuhiko Jimbo, Akio Kawana, Peitro Parodi, and Vincent Torre. The dynamics of a neuronal culture of dissociated cortical neurons of neonatal rats. *Biological Cybernetics*, 83:1–20, 2000.

[13] Yasuhiko Jimbo, Hugh P. C. Robinson, and Akio Kawana. Strengthening of synchronized activity by tetanic stimulation in cortical cultures: Application of planar electrode arrays. *IEEE Transactions on Biomedical Engineering*, 45(11):1297–1304, November 1998.

[14] V. K. Jirsa and J. A. S. Kelso. Spatiotemporal pattern formation in neural systems with heterogeneous connection topologies. *Physical Review E.*, 62(6), December 2000.

[15] S. B. Kater, P. B. Guthrie, and L. R. Mills. Integration by the neuronal growth cone: a continuum from neuroplasticity to neuropathology. In P. D. Coleman, G. A. Higgins, and C. H. Phelps, editors, *Molecular and Cellular Mechanisms of Neuronal Plasticity in Normal Aging and Alzheimer's Disease*, volume 86 of *Progress in Brain Research*, pages 117–128. Elsevier, 1990.

[16] S. B. Kater, M. P. Mattson, C. Cohan, and J. Connor. Calcium regulation of the neuronal growth cone. *Trends in Neurosc.*, (11):315—321.

[17] S.B. Kater, P.B. Guthrie, and L.R. Mills. Integration by the neuronal growth cone: a continuum from neuroplasticity to neuropathology. 1990.

[18] S.B. Kater, M.P. Mattson, C.S. Cohan, and J. Connor. Calcium regulation of the neuronal growth cone. *Trends in Neurosciences*, (11):315–321, 1988.

[19] Fumitaka Kawasaki and Michael Stiber. Accelerating large-scale simulations of cortical neuronal network development. Technical Report UWB-CSS-12-01, University of Washington Bothell, Coputing & Software Systems Program, 18115 Campus Way NE, February 2012.

[20] David B. Kirk and Wen mei W. Hwu. *Programming Massively Parallel Processors.* Elsevier Inc., Burlington, 2010.

[21] L.F.Abbot. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, Vol. 50(Nos. 5/6):pp. 303–304, 1999.

[22] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–60, November 2002.

[23] Henry Markram, Yun Wang, and Misha Tsodyks. Differential signaling via the same axon of neocortical pyramidal neurons. *Proceedings of the National Academy of Sciences of the USA*, 95:5323–8, April 1998.

[24] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS) - Special issue on uniform random number generation*, Volume 8(Issue 1), January 1998.

[25] Jayram Moorkanikara Nageswaran, Nikil Dutt, Jeffrey L. Krichmar, Alex Nicolau, and Alexander V. Veidenbaum. A configurable simulation environment for the efficient simulation of large-scale siking neural networks on graphics processors. *Neural Netwroks*, 22(5-6):791–800, 2009.

[26] Thomas Natschläger, Henry Markram, and Wolfgang Maass. Computer models and analysis tools for neural microcircuits. In R. Kötter, editor, *A Practical Guide to Neuroscience Databases and Associated Tools*, chapter 9. Kluver Academic Publishers, Boston, 2002.

[27] M. Papadopoulon, M. Sadooghi-Alvandi, and Henry Wong. Micro-benchmarking the GT200 GPU. Technical report, Computer Group, ECE, University of Toronto, 2009.

[28] J. Van Pelt, M.A. Corner, P.S. Wolters, W.L.C. Rutten, and G.J.A. Ramakers. Long-term characterization of firing dynamics of spontaneous bursts in cultured neural netwroks. *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, 51(11), NOVEMBER 2004.

[29] J. Van Pelt, M.A. Corner, P.S. Wolters, W.L.C. Rutten, and G.J.A. Ramakers. Longterm stability and developmental changes in spontaneous network burst firing patterns in dissociated rat cerebral cortex cell cultures on multielectrode arrays. *Neuroscience Letters*, 361:86–89, 2004.

[30] G. J. A. Remarkers, H. van Galen, M. G. P. Feenstra, M. A. Corner, and G. J. Boer. Activity-dependent plasticity of inhibitory and excitatory amino acid transmitter systems in cultured rat cerebral cortex. *Int. J. Devel. Nerosci*, 12:611–621, 1994.

[31] K. Schilling, M.H. Dickinson, J.A. Connor, and J.I. Morgan. Electrical activity in cerebellar cultures determines purkinje cell dendritic growth patterns. *Neuron*, (7):891–902, 1991.

[32] Michael Stiber, Fumitaka Kawasaki, and Dongming Xu. A model of dissociated cortical tissue. In *Neural Coding 2007*, pages 24–7, Montevideo, Uruguay, November 2007.

[33] http://www.mathworks.com/.

[34] Jan-Phillip Tiesel and Anthony S. Maida. Using parallel gpu architecture for simulation of planar i/f networks. In *Proceedings of International Joint Conference on Neural Networks*, pages 3118–3223, Atlanta, GA, June 2009. IEEE.

[35] Thomas P. Trappenberg. *Fundamentals of Computational Neuroscience Second Edition*. OXFORD UNIVERSITY PRESS, New York, 2010.

[36] Misha Tsodyks, Klaus Pawelzik, and Henry Markram. Neural networks with dynamic synapses. *Neural Computation*, 10:821–35, 1998.

[37] Misha Tsodyks, Asher Uziel, and Henry Markram. Synchrony generation in recurrent networks with frequency-dependent synapses. *The Journal of Neuroscience*, 20(RC50):1–5, 2000.

[38] F. van Huizen, H. J. Romijin, and A. M. M. C. Habets. Synaptogenesis in rat cerebral cortex is affected during chonic blockade of spontaneous bioelectric activity by tetrodotoxin. *Dev. Brain Res.*, 19:67–80, 1985.

[39] A. Van Ooyen and J. Van Pelt. Complex periodic behaviour in a neural network model with activity-dependent neurite outgrowth. *J. thor. Biol.*, 179:229–242, 1996.

[40] A. Van Ooyen, J. Van Pelt, and M.A. Corner. Implications of activity dependent neurite outgrowth for neuronal morphology and network development. *J. thor. Biol.*, 172:63–82, 1995.

[41] Daniel A. Wagenaar, Radhika Madhavan, Jerome Pine, and Steve M. Potter. Controlling bursting in cortical cultures with closed-loop multi-electrode stimulation. *The Journal of Neuroscience*, 25(3):680–8, January 2005.