EMERGING PRINCIPLES OF ECOLOGICAL NETWORK DYNAMICS

INNOVATIVE SYNTHESIS, MODELING, ANALYSIS, AND RESULTS

by

WILLIAM S. YACKINOUS

(Under the Direction of Bernard C. Patten)

ABSTRACT

The objective of this research is to characterize the dynamics of ecological networks. The work presented here takes a systems approach. It establishes a total system view of ecosystem behaviors and behavioral relationships before drilling down to details. The research is a blend of synthesis and analysis. We synthesize a "function-structure-process" framework that provides the high-level context for a full range of ecological system functions and their implementation. The implementation architecture of ecosystems is the *network*. Within this context, and based on an extensive review of the complex systems and network literature, we synthesize a view of ecological network dynamics. This view, in turn, is the basis for the central hypothesis of the research: ecological networks are ever-changing networks with propagation dynamics that are punctuated, fractal, and enabled by indirect effects. At this point, analysis becomes the focus of the work. We define, design, and develop an ecological network dynamics model to analyze and fully test the hypothesis. Our innovative modeling approach seeks to emulate features of real-world ecological networks. The approach does not make *a priori* assumptions about ecological network dynamics, but rather lets the dynamics develop as the model simulation runs. The model software development effort is substantial and includes not

only a comprehensive implementation of ecological network processes but also a full complement of analysis capabilities and graphics generation procedures. Model analysis results corroborate our hypothesis. We see that ecological networks exhibit fractal behavior in space and time. Network events have punctuated time series and power-law/fractal distributions. We see that ecological networks "flicker." (Network structure is not static and network flows are not steady state.) We see that *indirect effects* play a prominent role in ecological network dynamics. When observing the total behavioral picture, we can glimpse a general equivalence – a *universality* – in ecological network dynamics. We notice that these dynamics exhibit fundamentally the same form of behavioral statistics across spatial and temporal scales and even across processes. The same, apparently universal, principles seem to apply.

INDEX WORDS: ecological network dynamics, ecological network modeling, systems approach, complex systems, ecosystem functional framework, punctuated dynamics, fractal dynamics, indirect effects, universal principles

EMERGING PRINCIPLES OF ECOLOGICAL NETWORK DYNAMICS

INNOVATIVE SYNTHESIS, MODELING, ANALYSIS, AND RESULTS

by

WILLIAM S. YACKINOUS

B.S., Rutgers University, 1967

M.E.E., Stevens Institute of Technology, 1969

A Dissertation Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2010

EMERGING PRINCIPLES OF ECOLOGICAL NETWORK DYNAMICS

INNOVATIVE SYNTHESIS, MODELING, ANALYSIS, AND RESULTS

by

WILLIAM S. YACKINOUS

Major Professor:	Bernard C. Patten
Committee:	David K. Gattie John R. Schramski Ernest W. Tollner Stuart J. Whipple

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2010

TABLE OF CONTENTS

Page

LIST OF FIGURES vi

CHAPTER

1 INTRODUCTION 1

2 RESEARCH PERSPECTIVES 4

2.1 A Blend of Synthesis and Analysis 4

2.2 Network Thinking 8

2.3 "Flickering" Dynamics 9

2.4 Universality 10

3 FUNCTION-STRUCTURE-PROCESS FRAMEWORK 11

3.1 Understanding Function, Structure, and Process 12

3.2 A Framework for Ecological System Dynamics 16

3.3 New Perspective on Statics and Dynamics 22

4 A VIEW OF ECOLOGICAL NETWORK DYNAMICS 23

4.1 The Nature of Order in Complex Ecological Systems 23

4.2 Ecological Network Dynamics: Operational Viewpoint 30

4.3 Ecological Network Dynamics: Characteristics Viewpoint 31

4.4 Hypothesis 65

5 A NEW APPROACH TO MODELING ECOLOGICAL NETWORK DYNAMICS68

5.1 Model Requirements 68

5.2 Model Programming Design and Development 76

6 RESULTS 102

6.1 Operational Propagation Flow Results 102

6.2 Network Propagation Event Results 115

6.3 Path Length Results 120

6.4 Indirect Effects Results 126

6.5 Network Connectivity Results 131

7 SUMMARY AND CONCLUSIONS 142

REFERENCES 148

APPENDICES

A REVIEW OF NETWORK LITERATURE 156

B INTRODUCTION TO CELLULAR AUTOMATA MODELING 160

C GLOSSARY OF MATLAB VARIABLES BY CATEGORY 165

D DYNAMICS MODEL MASTER M-FILE 176

E DYNAMICS MODEL ANALYSIS M-FILE 196

F DYNAMICS MODEL GRAPHICS M-FILE 219

LIST OF FIGURES

Page

Figure 2.1: Combined Synthesis and Analysis 5

Figure 2.2: Reductionist Approach and Network Hierarchy 6

Figure 3.1: Function, Structure, and Process Relationships 15

Figure 3.2: Framework for Ecological System Dynamics 18

Figure 4.1: Weaver's Ranges of Complexity 26

Figure 4.2: Degree of Complexity 27

Figure 4.3: Degree of Order 27

Figure 4.4: The Domain of Attraction 28

Figure 4.5: Ecological Network Operation 31

Figure 4.6: Black Swan Region 33

Figure 4.7: Scale-Invariant/Fractal Behavior – Graphics 34

Figure 4.8: Human-Generated Spatial Fractals – Examples 39

Figure 4.9: Natural Spatial Fractal 39

Figure 4.10: Spatial Fractal Behavior in a Lattice Network 40

Figure 4.11: Real-world Complex Network 41

Figure 4.12: Example of Box Covering Procedure 42

Figure 4.13: Path Length Distributions of Real-world Fractal Complex Networks 43

Figure 4.14: Ecological Fluctuations Over Time 44

Figure 4.15: Frequency Spectra of Noise-like Time Signals 45

Figure 4.16: Self-Similarity of Temporal Fractals – Ising Model 49

Figure 4.17: Dynamical Equivalence across Time Frames 50

Figure 4.18: Heart Rate Time Series.....52

Figure 4.19: Heart Rate Frequency Spectra.....52

Figure 4.20: Healthy Heart Self-Similarity in Time 53

Figure 4.21: Empirical Data for Several Real Networks 58

Figure 4.22: Lévy Flight Probability Distribution 59

Figure 4.23: Macroevolution Critical Connectivity.....62

Figure 4.24: Network Percolation in an Undirected Random Network.....64

Figure 4.25: Percolation in a Two-Dimensional Lattice Network.....65

Figure 5.1: Neighborhoods for Node Interaction.....70

Figure 5.2: Estuary Aquatic Compartment Model.....72

Figure 5.3: Depiction of the Propagation Process 80

Figure 5.4: Path Length Calculations.....85

Figure 5.5: Riemann Zeta Function 95

Figure 6.1: Network Propagation Time Series 102

Figure 6.2: Explanation of Neighborhoods.....103

Figure 6.3: Network Propagation Flow (small propagation events)..... 104-105

Figure 6.4: Network Propagation Flow (mid-size propagation events).....106

Figure 6.5: Network Propagation Flow (large event; time step 152) 107

Figure 6.6: Network Propagation Flow (large event; time step 212) 107

Figure 6.7: Cumulative Propagation Flow (time step 100).....108

Figure 6.8: Cumulative Propagation Flow (time step 1000).....109

Figure 6.9: Cumulative Flow Value "Adjacency Matrix" (time step 100) 110
Figure 6.10: Cumulative Flow Value "Adjacency Matrix" (time step 1000) 111
Figure 6.11: Cumulative Node Input Value Grid 113
Figure 6.12: Cumulative Node Output Value Grid 114
Figure 6.13: Cumulative Node Stock Value Grid 115
Figure 6.14: Network Propagation Event Time Series and Distribution 116
Figure 6.15: Network Propagation Event Distribution Set 117
Figure 6.16: Network Propagation Event Frequency Spectrum 119
Figure 6.17: Path Length Time Series and Integrated Distribution 121
Figure 6.18: Integrated Path Length Distribution Set 122
Figure 6.19: Mean Path Length Time Series and Distribution Set 124
Figure 6.20: Indicator Individual-Time-Step Time Series 127
Figure 6.21: Indicator Cumulative Time Series 128
Figure 6.22: Cumulative Path Quantity Time Series 129
Figure 6.23: Indirect Path Quantity Time Series and Distribution 130
Figure 6.24: Direct Path Quantity Time Series and Distribution 131
Figure 6.25: Network Propagation Time Series with Selections 132
Figure 6.26: Node In-Degree Grid at Representative Time Steps 133
Figure 6.27: Node Combined-Degree Grid at Representative Time Steps 134
Figure 6.28: Node In-Degree Time Series and Distribution Set 135
Figure 6.29: Node Combined-Degree Time Series and Distribution Set 137
Figure 6.30: Node Mean Degree Time Series and Distribution Set 138
Figure 6.31: Network Connectivity Time Series Set 139

Figure B1: The Sandpile Metaphor.....161

Figure B2: The Sandpile Cellular Automaton Model.....162

Figure B3: Avalanche Event Distribution.....163

Figure B4: Avalanche Event Time Series.....164

CHAPTER 1

INTRODUCTION

Ecological systems take the form of networks. The broad purpose of my doctoral research is to increase understanding of ecological networks and the principles that characterize ecological network dynamics. To accomplish that, we employ a systems approach and proceed as follows. We create a functionality-driven framework that defines the ecological system function-structure-process landscape and guides our investigations. We hypothesize a view of ecological network dynamics principles based on the framework and the extant complexity and network literature. To test the hypothesis, we create a new approach for modeling ecological networks and exploring their dynamics. We develop software to implement the model. The model simulation results provide corroboration for the hypothesized principles of ecological network dynamics.

The research described in this dissertation is innovative and original in many respects. The full set of approaches and methods employed here has not before been applied to ecological systems. The work represents a "fresh look" at ecological network dynamics. I have begun with substantial experience from my own long-time career in systems engineering and with principles from systems theory, complexity theory, and network theory across many disciplines. I have assimilated, extended, and combined these resources in new ways to create this fresh look at ecological systems. That process is the working definition of innovation and creativity. Creativity is "*the ability to discover new relationships, to look at subjects from new perspectives,*

and to form new combinations from two or more concepts already in the mind." [1] Scientific discovery often arises from "picking up the stick from the other end." [2] The other end, in this case, is the systems engineer's perspective.

Here's a brief section-by-section description of the document. Chapter 2 discusses the systems engineering perspective and related perspectives that are reflected in this research. A function-structure-process (FSP) framework can provide a unifying context for exploring principles of ecological network dynamics. Chapter 3 describes the synthesis of this contextual framework. In Chapter 4, we synthesize a network-centric view of ecological system dynamics. We begin with a discussion of the nature of complex system self-organized order which leads to the notion of a *domain of attraction*. We then identify and define the characteristics of the ecological network dynamics associated with that attractor. In Chapter 5, we discuss our innovative approach to ecological network modeling. We model these networks (as well as their nodes) as discrete dynamic systems. We incorporate pertinent features of real-world ecological networks. We include analysis and graphics generation capabilities. We describe the model simulation software that we have developed to implement the model. Chapter 6 is all about results. Outputs of the model simulation runs are scrutinized and analyzed and results are generated. Important research conclusions are presented in Chapter 7.

There are six appendices. The literature review is the subject of Appendix A. Because ecological systems consistently take the form of complex networks, a comprehensive understanding of ecological systems requires an in-depth understanding of systems theory, complexity theory, and network theory. Accordingly, I have conducted an extensive review of

[1] J. R. Evans, *Creative Thinking in the Decision and Management Sciences*, South-Western Publishing, 1991.

[2] Herbert Butterfield, *The Origins of Modern Science*, Macmillan, 1960.

the classic to the most recent literature in these areas. That is evident throughout this dissertation. There are abundant excerpts from and references to the pertinent literature. In addition, I have documented my review of the network literature in a series of three papers on complex network structure and dynamics. Since the papers are unpublished, I have provided some brief introductory comments and a listing of the contents for each of the three papers in Appendix A. (The complete papers are available upon request.) In this research, our ecological network model is cellular automata based. Appendix B provides an introduction to cellular automata modeling along with a well-known example – the sandpile model. We have developed the software that implements the ecological network model using the MATLAB[3] programming environment and language. Appendix C contains a *glossary* of the software program variables. The glossary includes a set of naming conventions for the variables and a definition for each of the more than one hundred variables used in the model. Appendix D contains the software *master m-file* – which implements the model core process flow. Appendix E contains the m-file that implements the model *analysis activities*. Appendix F contains the m-file that implements the model *graphics generation* procedures. Each of the m-files is heavily commented to describe and document the software.

[3] MATLAB release R2009a, The MathWorks, Inc., February 12, 2009.

CHAPTER 2

RESEARCH PERSPECTIVES

This research fundamentally reflects a *systems engineering perspective*. My personal background and intentions are important motivating factors. During my career as a systems engineer at Bell Laboratories, I have learned a lot about the practice of systems engineering and the systems approach. Since the early 2000s, it has been my intention to apply this perspective and the associated skills and approaches to ecological systems. That is an underlying theme of my research and this dissertation.

Related perspectives are covered in the following subsections.

2.1 A Blend of Synthesis and Analysis

Let's start with some simple definitions.[4] The term *synthesis* comes from the Greek *syntithenai:* to put together. The term *analysis* comes from the Greek *analyein:* to break apart. In systems investigations, analysis alone is not enough.

Much doctoral research in the sciences, however, focuses on analysis and follows an essentially *reductionist approach*. The researcher identifies an area of interest; drills down to some narrow, specialized subset; and then investigates that subset in depth – all with the objective of discovery. The resulting discovery, however, is likely to be *isolated* in the sense that it is not adequately "connected" to a larger context. (We'll have a further explanation of the potential problems with the reductionist approach in a moment.)

My research takes a *systems approach* to scientific discovery. It does not focus solely on analysis, but rather is a blend of synthesis and analysis. The synthesis work provides the

[4] Merriam-Webster Online, http://www.merriam-webster.com/.

required higher level system "connections" and context. Research results are not isolated, but rather are appropriately *integrated* into a total system picture.

All systems efforts include both synthesis and analysis to some degree. The process is incremental, cyclic, and iterative as illustrated in Figure 2.1.

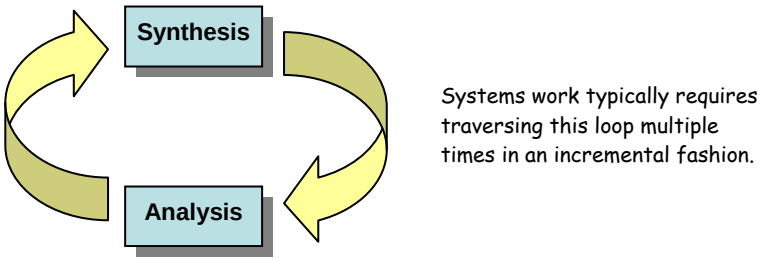


Figure 2.1 Combined Synthesis and Analysis

In the work described in this dissertation, synthesis is most prominent in:

- ✓ Creating a functionality-driven view of the function-structure-process landscape of ecological systems (Chapter 3).
- ✓ Creating a network-centric view of the principles of ecological system dynamics (Chapter 4).

Analysis is most prominent in:

- ✓ Developing an ecological network model to test our hypothesized view of ecosystem network dynamics (Chapter 5).
- ✓ Analyzing the model simulation output and generating results (Chapter 6).

Potential Problems with the Reductionist Approach

Reductionism is analysis-based: it breaks systems apart. The reductionist approach is predicated on "the belief that in every complex system the behavior of the whole can be understood entirely from the properties of its parts." [5] "The past three centuries of science have been predominantly reductionist, attempting to break complex systems into simple parts, and

[5] Fritjof Capra, *The Web of Life*, Anchor Books Doubleday, 1996.

those parts, in turn, into simpler parts." [6] "In recent years the practice of science has become increasingly reductionist in seeking to understand phenomena by detailed study of smaller and smaller components." [7]

The structure of ecological systems (and other complex systems) is a network – and a network can be represented as a hierarchy. Figure 2.2 depicts the reductionist approach in terms of a network hierarchy.

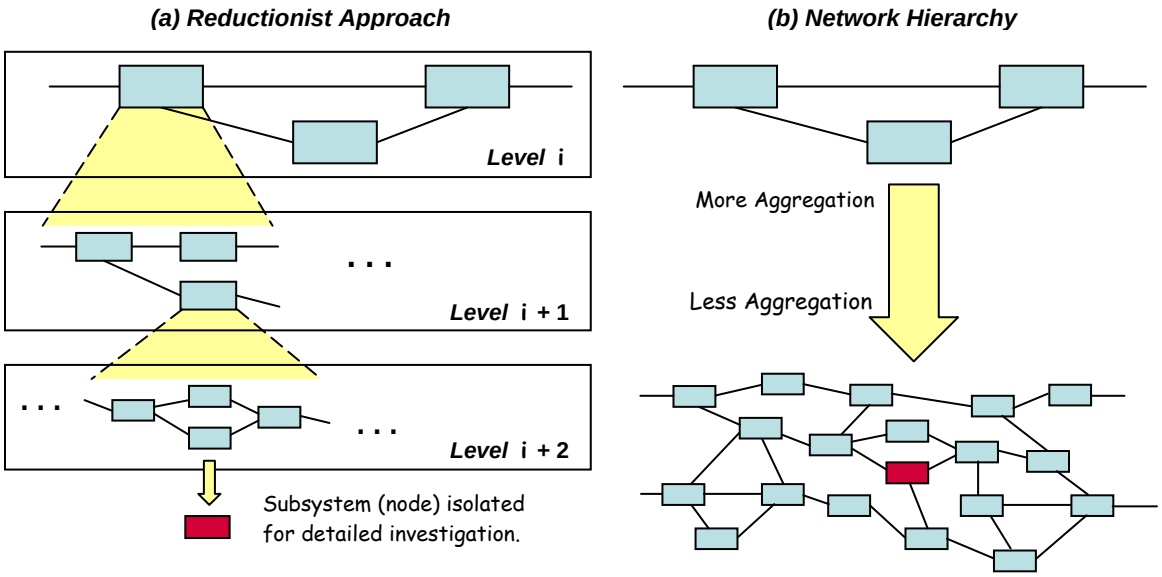


Figure 2.2 Reductionist Approach and Network Hierarchy

Part (a) of the figure shows the analysis-based reductionist approach of drilling down the system network hierarchy in order to conduct a detailed investigation of an increasingly specialized subsystem. At each level of the hierarchy, a subsystem (network node) is selected for further study, network connections (links) are broken, and the associated information typically is lost. "The reductionist likes to move from the top down, gaining precision of information about fragments as he descends, but losing information content about the larger orders he leaves

[6] Stuart Kauffman, *At Home in the Universe – The Search for Laws of Self-Organization and Complexity*, Oxford University Press, 1995.

[7] Eugene Odum and Gary Barrett, *Fundamentals of Ecology* Fifth Edition, Thomson Brooks/Cole, 2005.

behind." [8] Part (b) of the figure suggests that the number of broken network connections can be substantial. We see that the top-level view of the network is highly aggregated. Each time we move one level down the hierarchy we de-aggregate, i.e., we decompose each composite network node into its component network. As we proceed down, level-by-level, the network view rapidly expands and the number of network nodes and links rapidly increases. At the bottom level shown in the figure, we isolate a node for detailed investigation. Not only do we have a few broken direct connections, but also a very large number of broken indirect connections. The resulting information loss can be substantial.[9]

Here are two additional comments about the reductionist problems of isolation and information loss from noted authors. Odum and Barrett[10] say that reductionism is "specialization in isolation." Bar-Yam[11] states: "Indeed, one of the main difficulties in answering questions or solving problems … is that we think the problem is in the parts, when it is really in the relationships between them." The relationships/interactions among the parts – the connections – make all the difference.

We have highlighted potential problems with the reductionist approach. Reductionism, of course, has been a valuable tool for hundreds of years and continues to be a valuable tool. The key, in my opinion, is to use it prudently in the context of a holistic systems approach.

There is yet another problem. Look again at the bottom of Figure 2.2, part (b). The reductionist approach can leave us with a "node-centric" view of the system under investigation.

---

[8] Paul Weiss, a pioneer in general systems theory.

[9] The information loss that results from broken connections in reductionist ecological investigation can be quantified. Please see Yackinous, *Reductionism and Information Loss in Ecological Investigation*, March 25, 2009 (unpublished; available from the author).

[10] Eugene Odum and Gary Barrett, *Fundamentals of Ecology* Fifth Edition, Thomson Brooks/Cole, 2005.

[11] Yaneer Bar-Yam, *Making Things Work – Solving Complex Problems in a Complex World*, NECSI Knowledge Press, 2004.

System behavior, however, is primarily determined by network characteristics, not node characteristics. Networks trump nodes. We'll discuss that next.

2.2 Network Thinking

Networks occupy a central role in my research. The structure of an ecological system is a network – which consists of nodes (structural elements) and links (node structural connections). What most determines the behavior of the system? Is it the individual nodes or is it the interplay of the network as a whole? Reductionist scientific research is conducted by drilling down and isolating a system component (node) for detailed analysis. It is assumed that understanding a system is all about understanding the nodes. That is incorrect. It is not all about the nodes. It is about the network. To understand a complex system, *network thinking* is required.

Network thinking trumps node thinking. Many respected investigators and authors agree. Herbert Simon[12] declares that system behavior is determined more by the organization and relationships of the system components than by the detailed properties of each of the components. Solé and Goodwin[13] say: "Interactions, not individuals, are the key ingredients of behavioral complexity." Tamas Vicsek[14] suggests: "The laws that describe the behavior of a complex system are qualitatively different from those that govern its units." Paul Krugman[15] relates that Philip Anderson – a Bell Labs scientist, a Nobel laureate in physics, and sometimes called the "father" of complexity theory – believes that system collective behavior cannot be determined from individual unit behavior.

---

[12] Herbert Simon, *The Sciences of the Artificial*, MIT Press, 1996.

[13] Ricard Solé and Brian Goodwin, *Signs of Life – How Complexity Pervades Biology*, Basic Books, 2000.

[14] Tamas Vicsek, *Complexity: The Bigger Picture*, Nature, July 2002.

[15] Paul Krugman, *The Self-Organizing Economy*, Blackwell Publishers, 1996.

Because system behavior depends primarily on network dynamics and not on node details, we might ask whether, under some circumstances, nodes can be replaced without substantially changing network behavior. The answer seems to be yes. Ervin Laszlo[16] has observed that a range of complex "people" systems – e.g., consumers, investors, teams, societies, cultures – change individual "parts" all the time yet seem to maintain essentially the same properties. John Holland[17] has said (in the context of system diversity): "If we remove one kind of agent [node] from the system, creating a 'hole,' the system typically responds with a cascade of adaptations resulting in a new agent that 'fills the hole'." … "The ecosystem interactions are largely re-created, although the agents are quite different." And the network maintains coherence. In a complex adaptive system, "a pattern of interactions disturbed by the extinction of component agents often reasserts itself, though the new agents may differ in detail from the old."

2.3 "Flickering" Dynamics

Traditionally, much network modeling and analysis assumes a static network structure, steady-state network flows, and (sometimes) linear system behavior.[18] We, on the other hand, do not make these *a priori* assumptions about network dynamics. Our objective is to model and analyze ecological networks in order to determine the nature of their dynamics. We envision an ever-changing network structure, non steady-state flows, and nonlinear system behavior.

---

[16] Ervin Laszlo, *The Systems View of the World*, G. Braziller, 1972.

[17] John Holland, *Hidden Order – How Adaptation Builds Complexity*, Helix Books, 1996.

[18] All system models incorporate simplifying assumptions. This is necessary to reduce a complex real-world situation to something that is tractable and amenable to, e.g., mathematical or numerical analysis. In her recent book (M. Mitchell, *Complexity: A Guided Tour*, Oxford University Press, 2009), Melanie Mitchell discusses the well-known and pertinent quote by statistician George E. P. Box: "All theoretical models are wrong, but some are useful."

Odum and Barrett say "There are no equilibriums [in ecological systems], but there are *pulsing balances*." [19] We see these "pulsing" dynamics in our work here. Ecological network events have power-law distributions and punctuated time series. Individual events can be small or large. They can be gradual or abrupt. We see not steady-state behavior, but fluctuating dynamics in space and time. Ecological networks "flicker." [20]

2.4 Universality

We will see that ecological network behavior often is not dependent on scale. It is scale invariant. It is self-similar at all scales. The phrase *self-similar at all scales* is the definition of *fractal* behavior.[21] [We'll have much more to say about fractals in Chapter 4. The topic is very important to our understanding of the dynamics of ecological networks.]

Many ecological processes exhibit this self-similarity. A given process will exhibit fundamentally the same behavioral statistics across spatial scales and across temporal scales. An even broader type of self-similarity can be observed. Ecological systems exhibit the same *form* of dynamics *across processes*. There seems to be a general equivalence – a *universality* – in ecological network dynamics. The same, apparently universal, principles apply across space, time, and process.

---

[19] Eugene Odum and Gary Barrett, *Fundamentals of Ecology* Fifth Edition, Thomson Brooks/Cole, 2005.

[20] Here's the definition from Merriam-Webster Online (http://www.merriam-webster.com/).
flicker – **1**. to move irregularly or unsteadily **:** flutter, **2**. to burn or shine fitfully or with a fluctuating light.

[21] Solé et al, *Criticality and Scaling in Evolutionary Ecology*, Trends in Ecology and Evolution, April 1999.

CHAPTER 3

FUNCTION-STRUCTURE-PROCESS FRAMEWORK

A function-structure-process (FSP) framework can provide a unifying context for exploring principles of ecological network dynamics. The framework can be the vehicle for comprehensive and disciplined identification of ecological system functions, the structure (architecture) upon which the system functions are implemented, and the processes that operate within that architecture to deliver the functions. We have begun the synthesis of such a framework and, at this point in time, have achieved a high-level view of the framework. Since my current research deals with general principles of ecological network dynamics, the high-level view is sufficient for the work at hand.[22] Further, more detailed development of the framework would require follow-on work by me and/or interested others.

This section describes the synthesis of the contextual framework. We begin with fundamentals and build from there.

Read any ecology article, attend any ecology lecture, or read any ecology book. You're likely to see/hear something about *function*, *structure*, and/or *process*. These terms are very frequently used, but very rarely defined. Often the only sense you get is that structure has something to do with physical form while function and process have something to do with activity. Clarity is sorely lacking. In Section 3.1, we clearly define, describe, and discuss these concepts and how they are related. We indicate which of them is primary – and which are

[22] The high-level view, furthermore, is perhaps the most crucial. As noted in Chapter 2, drilling down to narrow, specialized investigations in a reductionist fashion without an understanding and awareness of the total system picture can yield faulty results due to broken connections, information loss, and analysis in isolation.

secondary (derived). We describe how they interact with each other and with the ecosystem environment. In Section 3.2, we see that this set of concepts leads us to a framework for ecological system dynamics – a framework that can help guide us toward a comprehensive understanding of ecological systems.

3.1 Understanding Function, Structure, and Process

3.1.1 Primary vs. Secondary

3.1.1.1 Function Is Primary

Flux (or we might also say movement or activity) is primary. Flux – from the Latin *fluere*, to flow – can be defined as a continuous moving on or passing by; a continued flow.[23] As the John Travolta character in the movie *Phenomenon* says: "Everything is on its way to somewhere."

David Bohm[24] has proposed that the primary characteristic of reality is this flux/ movement/activity. He says that reality is "a set of forms in an underlying universal movement." Bohm provides some historical background: "The notion that reality is to be understood as [movement] is an ancient one, going back at least to Heraclitus, who said that everything flows. In more modern times, Whitehead was the first to give this notion a systematic and extensive development." [Alfred North Whitehead (in the twentieth century) developed *The Philosophy of Organism* – the conviction that reality is composed principally of flows rather than objects.] Throughout Bohm's book, "the central underlying theme [is] the unbroken wholeness of the

[23] Merriam-Webster Online, http://www.merriam-webster.com/.

[24] David Bohm, *Wholeness and the Implicate Order,* Ark Paperbacks, 1983. (I note that Bohm, along with many other investigators, uses the term *process* loosely – and without defining it. Bohm, at times, implies that process is synonymous with the underlying flux. My view is that process is a *derived* expression of this flux.)

totality of existence as an undivided flowing movement without borders." "Not only is everything changing, but all *is* flux."

Flux is primary. Function is the concept we use to describe system flux or activity. System "*functioning* means *showing activity*." "Ecosystem functioning reflects the collective life activities of plants, animals, and microbes." [25] *Function is primary*.

3.1.1.2 Structure and Process Are Secondary

How is system function achieved? The answer is – via system *structure* and *process*. *Structure* represents a particular derived system implementation that can realize system function – via particular derived active mechanisms (which depend on this structure) called *process*. Structure is derived from function – it is a derived platform upon which derived processes execute.

Several respected investigators and authors support this perspective. Yaneer Bar-Yam[26] notes that "the function of the system dictates its structure." A very well-known design principle makes the same statement: "form ever follows function." Architect Louis Sullivan coined that phrase in 1896.[27] The general principle has been applied across system design disciplines (building architecture, engineering, many others) for years. David Bohm[28] has some interesting insights. He observes that modern physics seems to say that elementary particles such as electrons, protons, and neutrons (i.e., structure) are primary – and that physical phenomena are abstractions of the motions of the particles. But the particles can be created, transformed,

---

[25] These two quotes are from Shahid Naeem, Frank B. Golley, et al, *Biodiversity and Ecosystem Functioning*, Ecological Society of America, Issues in Ecology, Number 4, Fall 1999 (http://www.epa.gov/watertrain/pdf/issue4.pdf).

[26] Yaneer Bar-Yam, *Making Things Work – Solving Complex Problems in a Complex World*, NECSI Knowledge Press, 2004.

[27] Louis Sullivan, *The Tall Office Building Artistically Considered*, Lippincott's Magazine, March 1896.

[28] David Bohm, *Wholeness and the Implicate Order,* Ark Paperbacks, 1983.

annihilated – they are transitory – and so must be derived or "abstracted from some deeper level of movement."

Structure and process are derived from function. *Structure and process are transitory and secondary*.

3.1.2 Definitions and Relationships

3.1.2.1 Definitions

Function Definition

Function describes system activity – *what the system does*. The word function comes from the Latin *fungi*, to perform. Examples of ecological system functions performed in operational (as opposed to developmental) time frames include production, consumption, decomposition, and storage. Function is a primary characteristic of any system. For teleological systems, function is the *raison d'etre* (reason or justification for existence) of the system. For ecological systems (non-teleological systems) the reason for existence, or whether there is a reason, is not known.

Structure Definition

Structure is derived from function and specifies system physical form. Merriam-Webster[29] defines structure as "the arrangement of particles or parts" in a system. Ecological systems consistently take the form of networks. Network structure is comprised of nodes (structural elements) and links (node structural connections).

Process Definition

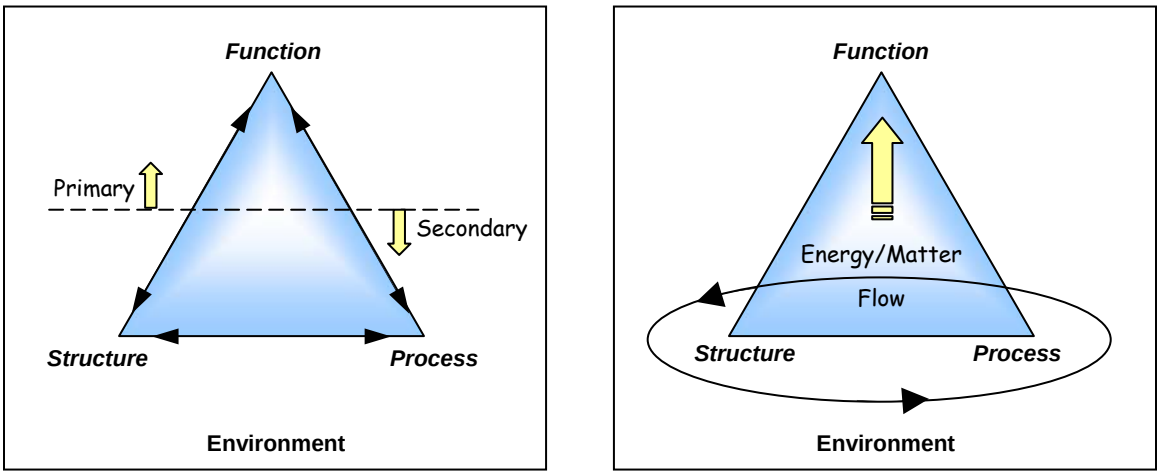
Process is derived from function and depends on structure. Processes are active flows which execute on the platform of structure to deliver function. The word process comes from the

[29] Merriam-Webster Online, http://www.merriam-webster.com/.

Latin *processus* and *procedere* – meaning procedure. Merriam-Webster's definition of process includes "something going on" and "a series of actions" that "lead toward a particular result."

3.1.2.2 Relationships

Figure 3.1 illustrates the relationships among function, structure, and process.



A. Primary/Secondary Relationship *B. Flow Relationship*

Figure 3.1 Function, Structure, and Process Relationships

Part A depicts the primary/secondary relationship that we have been discussing. Both structure and process are derived from function and both the function-to-structure relationship and the function-to-process relationship are one-to-many. Process further depends on structure and, to achieve any given function, the structure-to-process relationship is also one-to-many. All of the relationships are in the context of environment. Part B of the figure depicts the flow relationship in ecological systems. Process is flow through structure that delivers function in the context of environment. In ecological systems, the flow "currency" is energy/matter. Function is the outcome of the process/structure interaction – or as stated in the book *Philosophy and Design*[30] – "process and structure co-produce function in the context of environment."

[30] *Philosophy and Design – From Engineering to Architecture*, edited by Vermaas et al, Part 2, chapter by Kristo Miettinen, Springer Netherlands, 2008.

Consider an ecological system compartment model. The system structure depicted in such a (network) model supports processes that deliver functions. The following multi-part description of process applies. A *process* is:

- a (sub)system input/output relationship that realizes a function.
- a dynamic "thread" through system compartments and connections.
- a "compound flow" that encompasses some set of system compartments, system adjacency "simple flows," and system inputs and outputs.

In set-theoretic notation, the definition can be expressed as:

$$\text{Process} \equiv \left\{ z_i , x_i , f_{ij} , y_i \quad \textit{where } i, j \in 1,2,\cdots,n \quad \textit{such that } \text{a system function is achieved} \right\}$$

In the expression, the z's are system inputs, the x's are state variables (representing compartments), the f 's are flows (over connections), the y's are system outputs, i and j are indices in the range 1 to n, and n is the number of system compartments. A process, therefore, is a sequentially ordered k-tuple of system inputs, compartments, flows, and outputs such that a system function is achieved.

We see that process is an end-to-end flow path across system (model) structure often driven by input(s) from the environment and producing output(s) to the environment. A function is the net result of the occurrence of the process.

3.2 A Framework for Ecological System Dynamics

3.2.1 What, How, and Why

There are two quintessential questions that must be addressed in order to *understand* an ecological (natural) system or to understand and *build* a human-generated (artificial) system.

What does the system do? *How* does the system do it? Yaneer Bar-Yam[31] agrees and echoes the sentiment when he says: "Scientists look at something and want to understand what it does, and how it does it." The "what" is described by the system's *functions*. The "how" is described by the system's implementation, i.e., the implemented *structure* and implemented *processes* that realize the system functions.

There is also a third question: Why? This question can be viewed as teleological – having to do with the purpose of the system. When building an artificial (human-generated) system, we can answer this question. If we didn't know why we were building a system, we wouldn't build it. In a commercial/industrial setting, for example, it is the system stakeholders that provide the answer. For an ecological system, on the other hand, we are not in a position to answer the "why" question – or even to know (scientifically) if there is an answer.

To understand an ecological system, at best we can begin to address and answer the "what" (function) question and the "how" (structure and process) question.

3.2.2 Developing the Framework

So let's address the *what* and *how* questions in a systematic way. Let's develop a framework that addresses the questions; embodies the concepts of *function*, *structure*, and *process*; and therefore can serve as a basis for a comprehensive understanding of ecological systems. A high-level view of the resulting framework is provided in Figure 3.2.

[31] Yaneer Bar-Yam, *Making Things Work – Solving Complex Problems in a Complex World*, NECSI Knowledge Press, 2004.

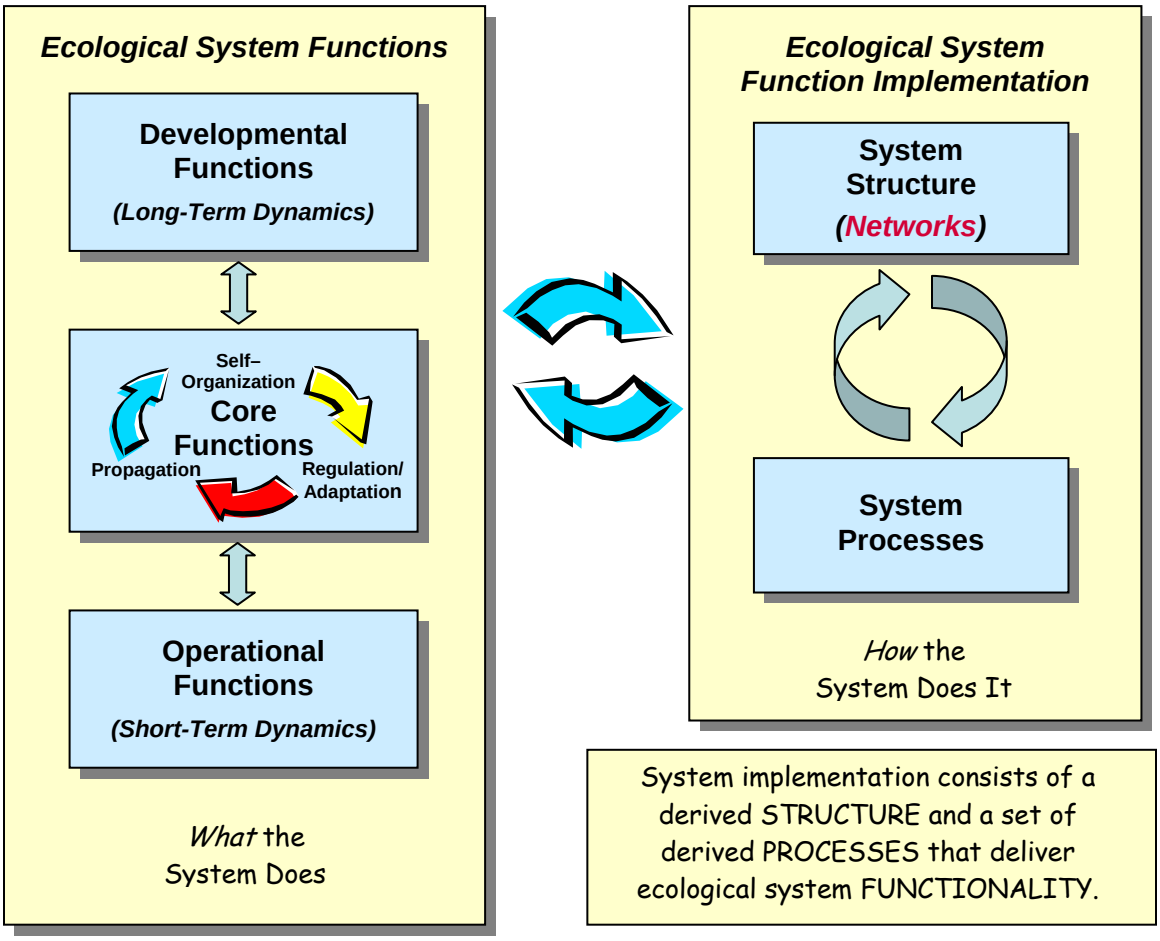


Figure 3.2 Framework for Ecological System Dynamics

As shown, the two major partitions of the framework address *what* and *how*, respectively. The left-hand partition covers "what the system does", i.e., the system *functions*. The right-hand partition covers "how the system does it" – the system implementation, i.e., the implemented *structure* and implemented *processes* that realize the system functions.

Let's first consider the left-hand ecological system function partition. John Holland[32] favors a two-tier model of complex adaptive systems – an upper tier that represents the "slow dynamics" of long-term system development and a lower tier that represents the "fast dynamics"

[32] John Holland, *Hidden Order – How Adaptation Builds Complexity*, Helix Books, 1996.

of short-term system operation. David Bohm[33] has said that to understand a system, we must understand the way "in which it forms, maintains itself, and ultimately dissolves." As shown in Figure 3.2, we take Bohm's and Holland's guidance and partition ecological system functions into *developmental functions* (long-term dynamics) and *operational functions* (short-term dynamics). The developmental functions involve *forming and dissolving*. The operational functions involve *maintaining* – and apply when the long-term dynamics can be assumed to be negligible. Ecological system developmental functions[34] include evolution (microevolution and macroevolution) and succession. Ecological system operational functions include production, consumption, and decomposition.

There is a third function partition shown in Figure 3.2 which is extremely important. This partition consists of a small interacting set of *core functions* that span the slow and fast dynamics regimes. I believe that an understanding of these core functions of self-organization, regulation/adaptation, and propagation is crucial for a comprehensive understanding of ecological systems. *Self-organization* [35] is perhaps the key underlying enabler of all ecosystem functionality. It is my view that an ecological system's *regulation/adaptation* to its environment and *propagation* of energy/matter are involved in or contribute to essentially all developmental and operational functions.[36] *Ecological systems (networks) self-organize in order to adapt and propagate in support of operational and developmental functions.*

---

[33] David Bohm, *Wholeness and the Implicate Order,* Ark Paperbacks, 1983.

[34] Bill Yackinous, *Evolution and Universal Development Concepts*, March 2009 discusses developmental functions and models.

[35] See Bill Yackinous, *Thermodynamic Entropy and Order in Ecological Systems*, March 2009 for a discussion of self-organization principles and characteristics.

[36] See Bill Yackinous, *Control Aspects of Ecological Systems*, March 2009 for a discussion of adaptation and propagation in system control.

Next let's consider the right-hand side of Figure 3.2 – ecological system function implementation, i.e., the implemented structure and implemented processes that realize the system functions. *System structure* takes the form of *network* architecture. *System processes* are active flows which execute on network structure to deliver ecological system function.

Networks: Complex ecological systems consistently take the form of networks. A comprehensive understanding of ecological systems, therefore, requires an in-depth understanding of complex networks.[37] Networks occupy a central role in my research.

System Processes: There are exceptions, but in general end-to-end processes tend to get neglected. In my extensive survey of the classic and recent network literature, for example, I found that most investigation to date in the scientific community has focused on the *structure* of networks. Investigation of end-to-end *processes* on networks lags way behind. This type of behavior is typical of many western scientific pursuits. In our reductionist research, we drill down and isolate a system component (node) for detailed analysis. In so doing we break network connections, sever system end-to-end processes, and lose valuable information.[38] Process concepts are also under-utilized in traditional ecological modeling. "Process" is employed primarily in a fragmented pair-wise sense. Process and process equations are used mostly to explain and calculate compartment-to-compartment pair-wise flows of the model currency. *System processes* must be given new respect if we are to achieve a comprehensive understanding of ecological systems.

This framework has and will continue to provide guidance to me in my work and in my efforts to better understand ecological systems. Other investigators are certainly welcome to use

[37] See the Yackinous series of three papers on complex network structure and dynamics (Appendix A).

[38] See Bill Yackinous, *Reductionism and Information Loss in Ecological Investigation*, March 2009.

the framework – and to offer suggestions for its elaboration and improvement. It is still a "work in progress."

3.2.3 The Three Core Functions of the FSP Framework

Self-organization, in a sense, is the underlying enabling function that makes all other functions possible. I have researched self-organization extensively in my literature review[39] and I have written about it.[40] In Chapter 4, we'll discuss self-organization and see that it plays an important role in our synthesis of a view of ecological network dynamics.

Regulation/adaptation is the system control function.[41] The need for this function in ecosystems is profound: it is required for system survival. In ecological systems, regulation/adaptation is *homeorhetic* – it maintains "pulsing states within limits." "There are no equilibriums [in ecological systems], but there are *pulsing balances*." [42] The homeorhetic control mechanisms are diffuse. Patten and Odum made that case in 1981. They explained that the control mechanisms "are all the factors, processes and interactions ... which serve to mediate the movement or transformation of energy-matter." [43] The interplay of material cycles and energy flows generates a self-controlling *homeorhesis* that allows an ecological system to adapt and respond to system perturbations. (As we shall see, these behaviors are fully consistent with the view of ecological network dynamics developed in Chapter 4.) The interactions and interplay inherent in regulation/adaptation are provided by the propagation function.

---

[39] See for example Bak 1996, Kauffman 1995, Krugman 1996, Müller 1996, Solé and Bascompte 2006.

[40] See Bill Yackinous, *Thermodynamic Entropy and Order in Ecological Systems*, March 2009.

[41] See Bill Yackinous, *Control Aspects of Ecological Systems*, March 2009 for a detailed discussion of this function.

[42] These two quotes are from Eugene Odum and Gary Barrett, *Fundamentals of Ecology* Fifth Edition, Thomson Brooks/Cole, 2005.

[43] Patten and Odum, *The Cybernetic Nature of Ecosystems*, The American Naturalist, Volume 118, 1981.

Propagation is the fundamental and essential means of interaction among nodes in an ecological network and between the network and its environment. In my research, the propagation function becomes the focus. In the remainder of this dissertation document, propagation is the centerpiece of much of the synthesis, modeling, and analysis activities.

3.3 New Perspective on Statics and Dynamics

In the traditional perspective, structure is often treated as a static property of a system. Function and process yield the system dynamics. I suggest a new perspective in which all three are dynamic. Bohm[44] supports the notion that structure is dynamic: All systems "are in a continual movement of growth and evolution of structure … ." Certainly in longer-term developmental time frames, system structure changes with time. Even in shorter-term operational time frames, system structure – represented as a network – is not static. Network node state changes continually occur and network links continually become active and inactive as required by the system functions/processes and the system environment. The network "flickers." Everything is changing all the time. Everything is dynamic. As David Bohm has said: "All *is* flux."

[44] David Bohm, *Wholeness and the Implicate* Order, Ark Paperbacks, 1983.

CHAPTER 4

A VIEW OF ECOLOGICAL NETWORK DYNAMICS

The core functions of the FSP framework provide the context for defining the *attractor* [45] of complex ecological network dynamics and for synthesizing a view of those dynamics.

We begin with a discussion of the nature of complex system self-organized order which leads to a definition of the *domain of attraction*. We then synthesize a view of the ecological network dynamics associated with that attractor. We take both an *operational perspective* and a *characteristics perspective* on the dynamics. The synthesis is solidly based on our comprehensive literature review.

4.1 The Nature of Order in Complex Ecological Systems

4.1.1 Self-Organization

Self-organization is a fundamental core function of complex ecological systems. Stuart Kauffman[46] has explored this function extensively and seeks its general principles. He says that order "arises naturally and spontaneously because of these principles of self-organization – laws of complexity that we are just beginning to understand." "Self-organization … may be the ultimate wellspring of [the] dynamical order" that underlies the origin, development, and operation of living systems. Several other prominent authors and researchers have explored the subject as well – in the context of a variety of both living systems and nonliving systems. We'll provide a sampling in the following paragraphs.

---

[45] We will refer to this attractor as the *domain of attraction* for ecological network dynamics.

[46] Stuart Kauffman, *At Home in the Universe – The Search for Laws of Self-Organization and Complexity,* Oxford University Press, 1995.

Fritjof Capra[47] claims that philosopher Immanuel Kant (in 1790) was "the first to use the term 'self-organization' to define the nature of living organisms." Capra says the hallmark of self-organization is "the striking emergence of new structures and new forms of behavior." The work of Ilya Prigogine (1977 Nobel laureate in chemistry) demonstrates that. "The first, and perhaps most influential, detailed description of self-organizing systems was the theory of 'dissipative structures' by the Russian-born chemist and physicist Ilya Prigogine" developed in the 1970s. Prigogine's vehicle for investigation included Bénard cells. These are hexagonal convection cells that appear spontaneously when a thin layer of liquid is heated from below. It is a nonequilibrium (temperature nonuniformity) threshold phenomenon – that exhibits spontaneous self-organization. Prigogine also studied the BZ (Belousov-Zhabotinskii) chemical reaction – in which a combination of appropriate chemicals plus energy yields a reaction solution whose color oscillates from red to blue to red at regular time intervals. It's a self-organized chemical clock. Note that both of these phenomena require an input of energy that is dissipated by the system. Prigogine introduced the term *dissipative structures* to describe the principle that "in open systems dissipation becomes a source of order."

Odum and Barrett[48] weigh in on the subject of self-organization. They cite the work of Prigogine as well as the work of Bob Ulanowicz. "A major key to ecosystem development is the concept of *self-organization*, based on Prigogine's theory of non-equilibrium thermodynamics. Self-organization can be defined as the process whereby complex systems consisting of many parts tend to organize to achieve some sort of stable, pulsing state in the absence of external interference. ... Self-organized ecosystems can only be maintained by a constant flow of energy

[47] Fritjof Capra, *The Web of Life*, Anchor Books Doubleday, 1996.

[48] Eugene Odum and Gary Barrett, *Fundamentals of Ecology* Fifth Edition, Thomson Brooks/Cole, 2005.

through them; therefore, they are not in thermodynamic equilibrium. ... Ulanowicz[49] used the term ascendency for the tendency for self-organizing, dissipative systems to develop complexity of biomass and network flows over time."

Paul Krugman[50] (2008 Nobel laureate in economics – and a professor at Princeton University and a columnist for the NY Times) discusses the importance of the concept of self-organization – across disciplines. "The most provocative claim of the prophets of complexity is that complex systems often exhibit spontaneous properties of self-organization." "I believe that the ideas of self-organization theory can add substantially to our understanding of the economy." "In the last few years the concept of *self-organizing systems* ... has become an increasingly influential idea that links together researchers in many fields, from artificial intelligence to chemistry, from evolution to geology."

4.1.2 The Domain of Attraction

Nonlinear dynamics theory[51] defines three types of *attractors* for the dynamics of complex systems – point attractors, limit cycle or periodic attractors, and strange attractors. These attractor types were derived based on systems of relatively low complexity. In my view, the reason for working with low complexity systems is that these are the systems that are amenable to mathematical and numerical analyses. The question then arises: Does this attractor classification scheme apply equally well to very complex systems such as ecological systems? My research findings suggest that it does not – at least not entirely. Conventional nonlinear dynamics theory seems to be incomplete. Stephen Wolfram's work on cellular automata[52] seems

[49] For example, in Robert Ulanowicz, *Ecology, the Ascendent Perspective*, Columbia University Press, 1997.

[50] Paul Krugman, *The Self-Organizing Economy*, Blackwell Publishers, 1996.

[51] See Yackinous, *Fundamentals of Nonlinear Dynamics*, November 27, 2007.

[52] See, for example, Schiff, *Cellular Automata*, Wiley-Interscience, 2008.

to support this point of view. Cellular automata (CA) are discrete dynamical systems. Wolfram has defined four classes of CA. Classes I, II, and III have dynamics that correspond to the three known classes of attractors, i.e., point attractors, periodic attractors, and strange attractors, respectively. Class IV cellular automata dynamics, on the other hand, are not well-understood and they do not correspond to a conventional attractor type. Class IV CA seem to be much more complex systems. Another question then arises: What is the attractor(s) for very complex systems? One objective of my research is to begin to address that question. I have coined a term and refer to the attractor of the dynamics of self-organized complex ecological networks as the *domain of attraction*. A high-level narrative rationale for this domain follows.

Warren Weaver's[53] *ranges of complexity* have been influential over the years and are a useful way of thinking about complexity. Figure 4.1 illustrates the ranges.

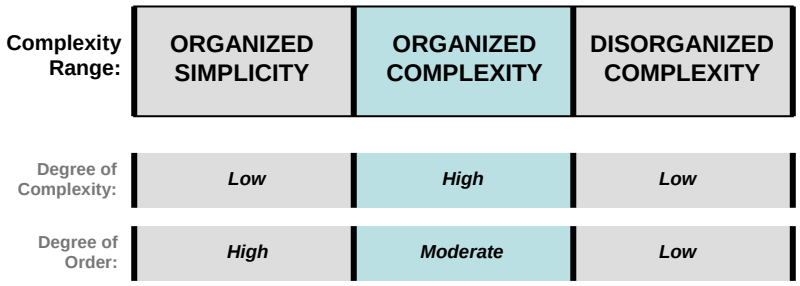


Figure 4.1 Weaver's Ranges of Complexity

Let's make some additions to Figure 4.1 to illustrate degree of complexity and order – as a way of introducing the concept of *domain of attraction*. Figure 4.2 depicts degree of complexity vs. Weaver's ranges.

[53] Warren Weaver, *Science and Complexity*, American Scientist, 1948.

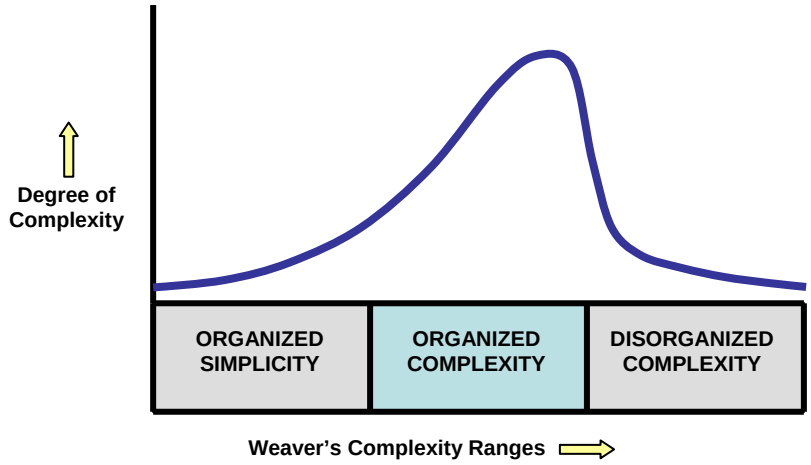


Figure 4.2 Degree of Complexity

Solé and Goodwin[54] offer support for this depiction: "A properly defined complexity measure … should reach its maximum at some intermediate levels between the order of a perfect crystal and the disorder of a gas." … "The point of maximal complexity is sharply defined. This is where complexity lives."

Consistent with Solé and Goodwin's pronouncement, Figure 4.3 depicts degree of order vs. Weaver's ranges.

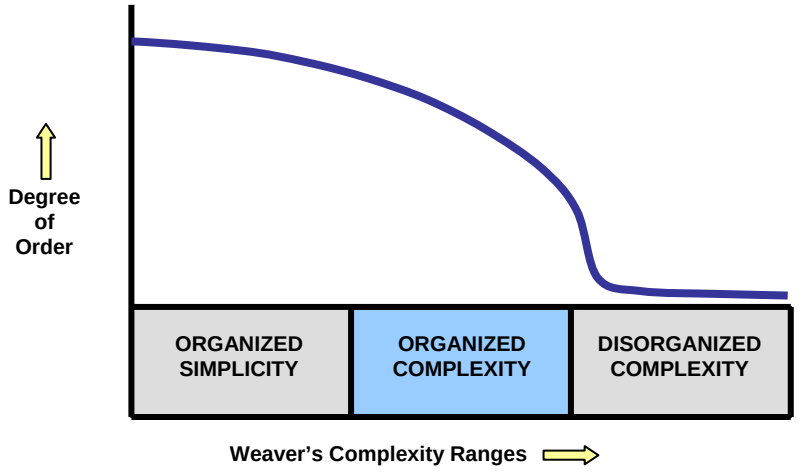


Figure 4.3 Degree of Order

[54] Solé and Goodwin, *Signs of Life – How Complexity Pervades Biology*, Basic Books, 2000.

Highly complex systems seem to be poised in an agile, responsive domain between "too ordered" and "too disordered" behavior.

Let's now combine these notions of "just right" (flexible) order and high complexity – and define what I call the *domain of attraction*. See Figure 4.4.

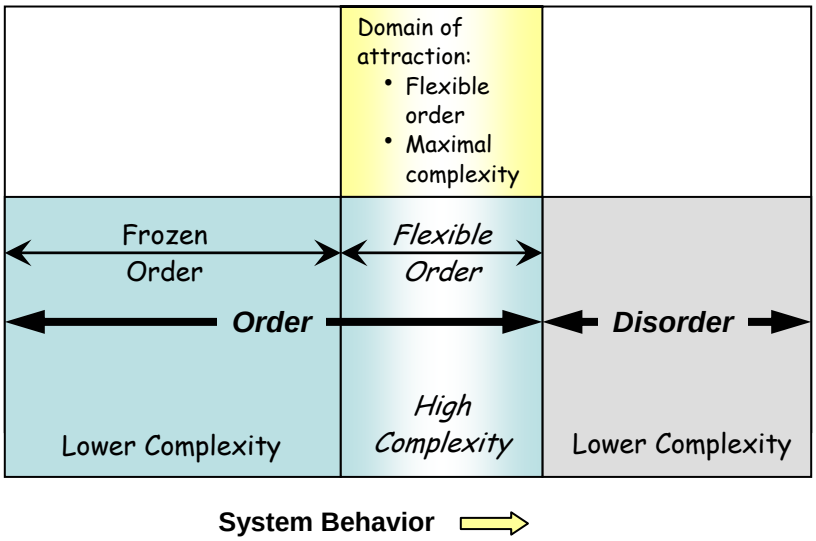


Figure 4.4 The Domain of Attraction

The domain of attraction is characterized by *maximum complexity* and *flexible order* –conditions that appear to help optimize system sustainability and survivability. This domain is where well-functioning ecological systems reside. The edge-of-order location is a "good place to be … where, on average, we all do best." [55] Stuart Kauffman[56] offers many important insights regarding the edge-of-order location:

- "Life evolves toward a regime that is poised between order and chaos."
- "Life exists at the edge of chaos" – near a phase transition, in the language of physics.

[55] Stuart Kauffman quoted in Bak, *How Nature Works*, Springer-Verlag, 1996.

[56] Stuart Kauffman, *At Home in the Universe – The Search for Laws of Self-Organization and Complexity*, Oxford University Press, 1995. Kauffman uses the term "chaos." In my view the conventional, non-scientific use of that term has implications that are inaccurate in the context of this investigation. I prefer the term "disorder."

- "It is a lovely hypothesis, with considerable supporting data, that genomic systems lie in the ordered regime near the phase transition to chaos." These systems are not too rigidly ordered, not unordered, but just right.
- To cope with a complex environment, networks must not be too rigid – they need order *and* flexibility. "How do … networks achieve both stability and flexibility? The new and very interesting hypothesis is that networks may accomplish this by achieving a kind of poised state balanced on the edge of chaos."
- "Just between [order and chaos], just near this phase transition, just at the edge of chaos, the most complex behaviors can occur – orderly enough to ensure stability, yet full of flexibility and surprise. Indeed, this is what we mean by complexity."
- "Perhaps networks just at the phase transition, just poised between order and chaos, are best able to carry out ordered yet flexible behaviors."
- "The transitional region between order and chaos [is] where complex behavior thrives."

Kauffman posits that perhaps this edge-of-order location, "ordered and stable, but still flexible, will emerge as a kind of universal feature of complex adaptive systems in biology and beyond."

Per Bak[57] makes insightful observations about what system complexity is and is not. Bak says: "A picture of nature in 'balance' often prevails." This is not true – even though it can seem so in the time scale of human perception. "How can there be evolution if things are in balance? Systems in balance or equilibrium, by definition, do not go anywhere." Behavior of complex systems cannot be explained in a context of balanced equilibrium.

Bak says further that systems in chaos are not highly complex. Not much useful happens in the disordered regime. Here, system behavior exhibits randomness; system fluctuations have

[57] Per Bak, *How Nature Works – The Science of Self-Organized Criticality*, Springer-Verlag, 1996.

a white noise spectrum; there is no contingency, no memory, no correlation with past events. Bak observes that: "Chaos signals [time series] have a white noise spectrum, not 1/f." … "However, precisely at the 'critical' point where the transition to chaos occurs, there is complex behavior, with a 1/f-like signal. The complex state is at the border between predictable periodic behavior and unpredictable chaos." [We'll cover 1/f signals in Section 4.3.2.2.]

There is ample support for the idea that highly complex self-organized systems can reside in the *domain of attraction*. Next, we want to synthesize a view of the ecological network dynamics associated with this domain. We take both an *operational viewpoint* and a *characteristics viewpoint* on the dynamics.

4.2 Ecological Network Dynamics: Operational Viewpoint

How do ecological networks operate in the domain of attraction? A description of our hypothesized view follows.

Reality is dynamic. There are families of natural system networks in play at all times. The network nodes are available for interaction, and any given node can participate in multiple networks. In response to some stimulus – say an input of energy or biomass – connections between nodes become active and begin formation of a network. The ecological network can evolve to become a critically connected network with high complexity and flexible order. The nodes of the network operate and interact locally and/or globally as necessary until the "processing" of the stimulus is completed. If the system input is removed, the network connections eventually become inactive once more. *Network connections are not persistent – they are ever-changing – active when needed and inactive when not needed. The network "flickers."* The above scenario is repeated over and over again, in space and in time, as required by the real environment.

Ecological network operation is illustrated graphically in Figure 4.5.

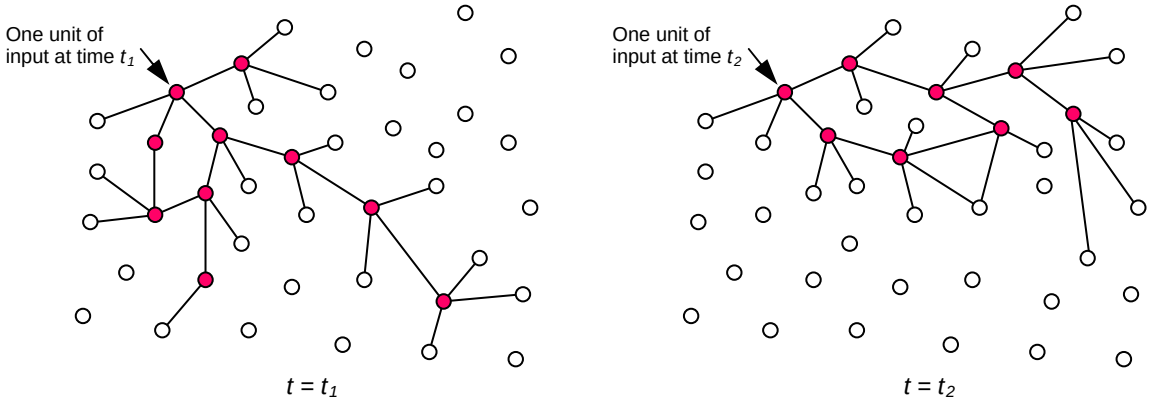


Figure 4.5 Ecological Network Operation

The figure displays two different moderate-sized propagation events that occur at two different instants of time. At time t_1 a unit of input is applied to a randomly selected input node. The input causes that node to propagate (and turn red or "light up" as depicted in the figure). Propagation results in flows to nearby nodes. Some of those downstream nodes then propagate to other nodes, and so on. The process continues until no further propagation flow occurs. We see that node interaction begins locally, but can extend globally. As a result of this propagation event, the states of all the involved nodes have changed. At the next instant (time t_2), we have a different set of network initial conditions, potentially a different input node, and therefore a different propagation event. Different nodes will "light up." The network *flickers*. (A good visual metaphor would be blinking lights on a holiday tree.)

4.3 Ecological Network Dynamics: Characteristics Viewpoint

How can we statistically characterize the ecological network dynamics associated with the domain of attraction – and with the above operational behavior? We identify and define the characteristics here.

4.3.1 Punctuated Dynamics

Geologist/anthropologist Charles Lyell [1797–1875] formulated the philosophy of uniformitarianism or gradualism which claimed that smooth gradual processes were at work in natural systems. A small cause yielded a small effect. All things could be explained by linear extrapolation. Darwin apparently accepted this view – he stated that evolution is smooth and gradual; he even denied the existence of evolutionary mass extinctions. [This is according to Per Bak.[58]]

Gould and Eldredge disagreed and instead offered a "punctuated equilibrium" explanation (e.g., in their classic 1977 paper[59]). "Punctuated equilibrium is the idea that evolution occurs in spurts instead of following the slow, but steady path that Darwin suggested. Long periods of stasis with little activity in terms of extinctions or emergence of new species are interrupted by intermittent bursts of activity." [Quote is from Bak.]

My view is that punctuated dynamics – not gradualism – are at work in ecological systems. I suggest that the dynamics of complex ecological networks exhibit such behavior. The time series of ecological system events are punctuated and the event probability distribution follows a power law. There is a wide distribution of events – from the ordinary (gradual and expected) to the extreme (abrupt and unexpected).

Black Swans

"Black Swans" are extreme events – *unlikely*, *unexpected*, *high impact* events. They reside at the "long tails" of power-law/fractal event distributions. The Black Swan metaphor, perhaps mentioned first in John Stuart Mill, *A System of Logic*, 1860, has been used many times since. It is currently most associated with Karl Popper and his work on The Problem of

[58] Ibid.

[59] Gould and Eldredge, *Punctuated Equilibria: The Tempo and Mode of Evolution Reconsidered*, Paleobiology, Spring 1977.

Induction (I see only white swans – therefore all swans are white – except when they are not – etc.). There is a recent book on the subject by Nassim Taleb.[60]

A pictorial illustration of the black swan region of a power-law event distribution is provided in Figure 4.6.

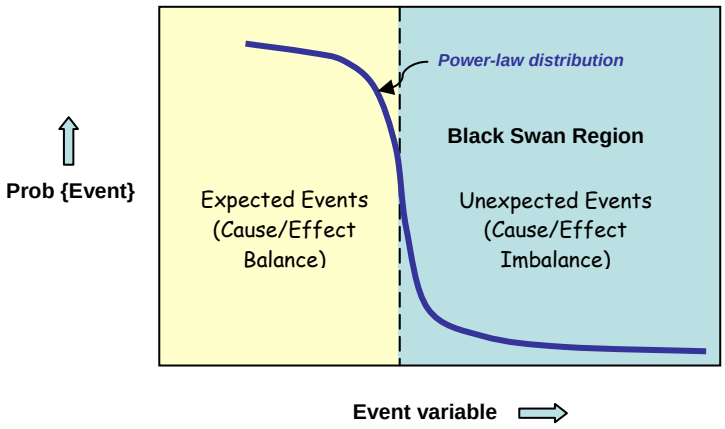


Figure 4.6 Black Swan Region

This punctuated dynamics perspective represents a challenge for many of us. Linear explanations are pervasive in our thinking. Acceptance of nonlinear, punctuated behavior is a challenge.

4.3.2 Fractal Behavior

In my research work, I am finding that complex system behavior in the domain of attraction is generally not dependent on scale. Rather, it is scale invariant; it is self-similar at all scales. The phrase *self-similar at all scales* is the definition of *fractal* behavior.[61] Euclidian geometry is human-made, but fractal geometry is the geometry of nature. Fractal behavior is central to our understanding of the dynamics of ecological networks. Fractals are very

[60] Nassim Taleb, *The Black Swan*, Random House, 2007.
[61] Solé et al, *Criticality and Scaling in Evolutionary Ecology*, Trends in Ecology and Evolution, April 1999.

widespread in nature – not only spatially but also temporally. This section addresses these important ideas.

Complex ecological networks exhibit a wide range of system responses (a power-law/scale-invariant/fractal distribution of responses). The responses occur within well-defined statistical laws. What are the well-defined statistical laws – graphically and mathematically? Figure 4.7 presents a graphical view of scale-invariant/fractal behavior.

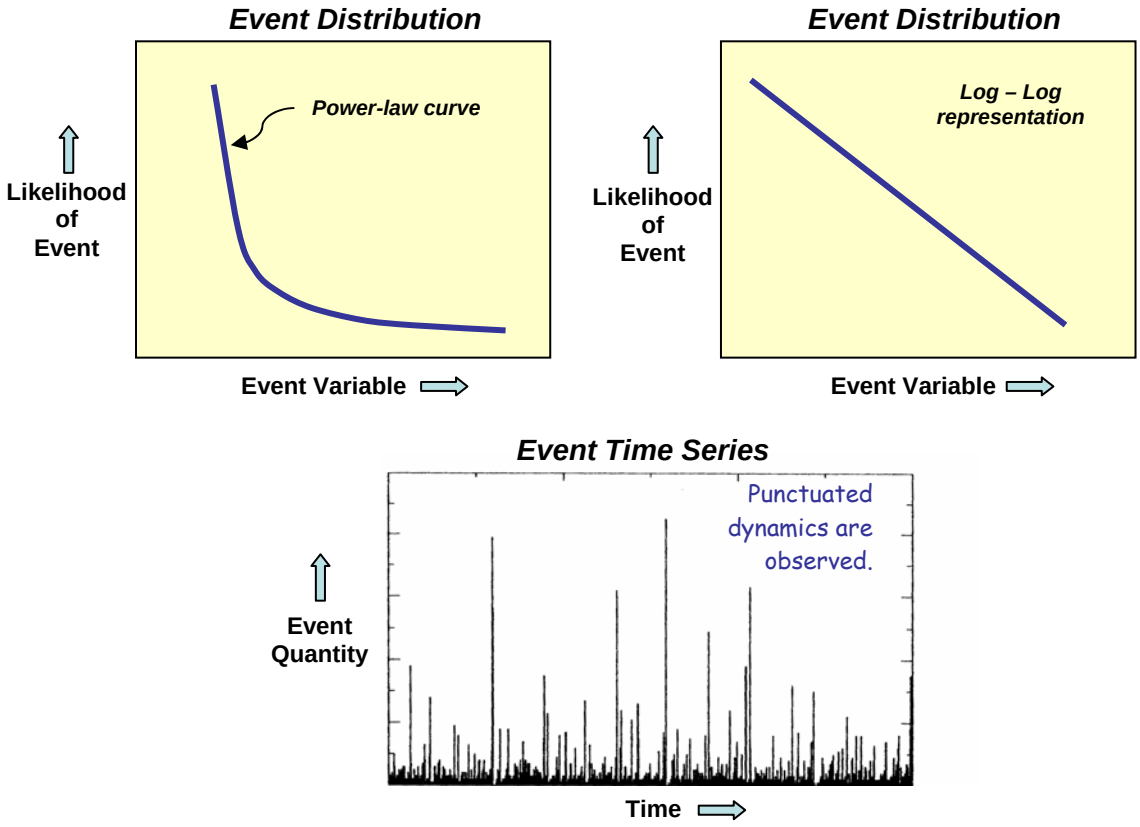


Figure 4.7 Scale-Invariant/Fractal Behavior – Graphics

The upper left panel of the figure shows that a fractal event distribution follows a power-law curve.[62] The upper right panel indicates that, in log-log coordinates, the event distribution becomes a straight line. Examples of *events* include: network formation events, network

[62] Those who early noticed, studied, and wrote about power-law behavior include Pareto (1890s); Yule (1920s); Zipf (1930s); Herbert Simon (1950s); and Benoit Mandelbrot (1990s). More recently, many complex system investigators have joined this group.

propagation events, … , earthquakes, rain, and heart rate. Examples of *event variables* include: node degree, event size, spatial dimension, temporal duration, and frequency content. The event distributions show that there is a wide spectrum of possible events. "Small" events are more likely, but "large" events can occur and do occur. Small events are expected; large events should be expected as well. The lower panel of Figure 4.7 illustrates the form of a fractal time series (event fluctuations over time). The dynamics are clearly punctuated. We'll focus on temporal fractal behavior in Section 4.3.2.2.

Next, here's a brief overview of scale-invariant/fractal mathematics. Newman[63] states that scale-invariant/fractal "refers to any functional form $f(x)$ that remains unchanged to within a multiplicative factor under a rescaling of the independent variable x. In effect this means power-law forms, since these are the only solutions to $f(ax) = bf(x)$." Any scale – within the power-law interval of applicability – will exhibit statistically the same behavior. Solé et al[64] provide a similar development: Consider the distribution function, $N(s)$. The variable s could be spatial or temporal. $N(s)$ is said to follow a power law if

$$N(s) = C\, s^{-\gamma}$$

Here C is a constant and γ is a given exponent, often called the scaling exponent. The reason why these laws are characteristic of scale-invariant/fractal behavior is that they are the only functions displaying invariance under scale change. If we look at a larger or smaller scale, that is, if we take

$$s' = a\, s$$

where a is a multiplicative factor, it is not difficult to see that

[63] Newman, *The Structure and Function of Complex Networks*, SIAM Review, May 2003.

[64] Solé et al, *Criticality and Scaling in Evolutionary Ecology*, Trends in Ecology and Evolution, April 1999.

$$N(s') = C' N(s)$$

or, in other words, a change of scale does not modify the basic statistical behavior. We have self-similar behavior at all scales.

Showing that a power-law event distribution is a straight line in log-log coordinates is simple. Taking the logarithm of both sides of the power-law equation, we obtain

$$\log N(s) = \log C - \gamma \log s$$

In log-log space this, of course, is the equation of a straight line with slope equal to the negative of the scaling exponent.

Before moving on to a more detailed treatment of fractals in space and time, I'd like to provide additional perspectives on these subjects from the literature.

From Solé et al[65]:

- "The patterns displayed by many natural systems do not allow for a simple description using Euclidean geometry: they present scale-invariance; that is, no characteristic length measure can be obtained from them. Therefore, when observed at different resolutions, they display the same pattern. This is the case of river networks and mountains, tree branching and blood vessels or forest spatial structures. Even at the molecular level, fractals can be observed: if we analyse the linear distribution of nucleotides in a DNA chain, a self-similar pattern can also be detected."
- "Fluctuations in ecological systems are known to involve a wide range of spatial and temporal scales, often displaying self-similar (fractal) properties."
- "Fractals are widespread in nature and have features that look the same when there is a change in scale [spatial or temporal]: they are called 'self-similar'. In biology, self-similar

[65] Ibid.

patterns are known to occur [in space] at many levels. But fractals are also present in time: the fluctuations of a given quantity can appear the same when observed at different temporal resolutions. This is the case for heartbeat intervals, epidemics in small islands, breeding bird populations or the fossil record."

From Solé and Bascompte[66]:

"A striking, widespread feature of many complex systems is that some of their properties are reproduced at different scales in such a way that we perceive the same patterns when looking at different subparts of the same system. This property, known as scale-invariance, is widespread in many systems under nonequilibrium conditions. This is the case for ecological systems, where flows of energy enter into the system and are dissipated at different, interconnected scales [see local-to-global dynamics – Section 4.3.3]. The origin of such fractal patterns, named after the pioneering work by Benoit Mandelbrot[67], is a fundamental problem in many areas of science. Actually, empirical evidence has been mounting in support of the unexpected possibility that many different systems arising in disparate disciplines such as physics, biology, and economy may share some intriguingly similar scale invariant features."

We will now focus on spatial scale-invariant/fractal behavior and then temporal scale-invariant/fractal behavior in the next two subsections, respectively.

4.3.2.1 Fractal Behavior in Space

In this subsection, we'll look at examples of human-generated spatial fractals and then natural spatial fractals. Following that, we'll discuss spatial fractals in networks.

---

[66] Solé and Bascompte, *Self-Organization in Complex Ecosystems*, Princeton University Press, 2006.

[67] Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman, 1990.

4.3.2.1.1 Human-Generated Spatial Fractals and Natural Spatial Fractals

Note that human-generated spatial fractals can appear *exactly* the same at different scales. Spatial fractals in nature, on the other hand, are usually so only in the statistical sense – i.e., their features are *statistically*, rather than *exactly*, the same when observed and measured at different scales. [The same is true for temporal fractals.]

A good way to explain some spatial fractals is via examples. First we look at examples of human-generated spatial fractals. A mechanism[68] for generating fractal objects takes the form of a recursive process in which self-similar objects of different size are repeated. The dimensional relationship of such a spatial fractal can be written as:

$$g = r^d$$

where g = the number of smaller self-similar structural elements in the next larger self-similar element, r = the ratio of the characteristic measure of the larger to the smaller element, and d = the fractal dimension. For this process, $d \neq$ an integer and $d <$ the geometric dimensionality. The self-similar objects have a power-law size distribution $N(s)$:

$$N(s) = C\, s^{-d}$$

where s = object size, d = fractal dimension, and C is a constant.

[68] The sources for this development are Csermely, *Weak Links – Stabilizers of Complex Systems,* 2006 and Solé and Bascompte, *Self-Organization in Complex Ecosystems*, 2006.

Figure 4.8 provides examples of the so-called Sierpinski Gasket and Koch Curve.

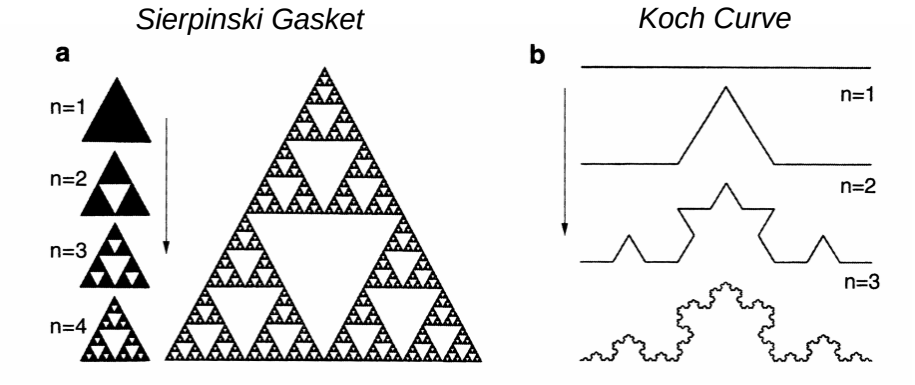


Figure 4.8 Human-Generated Spatial Fractals – Examples

In the figure, parameter *n* is the iteration number. In each of the two examples, the last form shown is a more fully developed view of the fractal – after several more iterations.

For the *Sierpinski Gasket*: $g = r^{d} \Rightarrow 3 = 2^{d} \Rightarrow d \approx 1.58$

For the *Koch Curve*: $g = r^{d} \Rightarrow 4 = 3^{d} \Rightarrow d \approx 1.26$

Next let's look at examples of natural spatial fractals. A very well-known spatial fractal in nature is shown in Figure 4.9.[69]

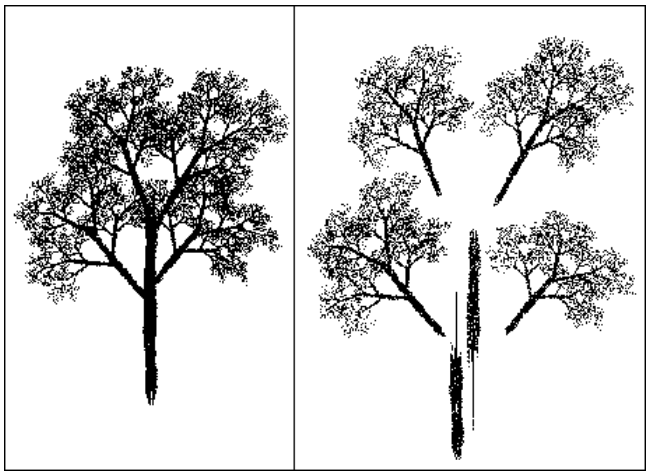


Figure 4.9 Natural Spatial Fractal

[69] *Fractal Geometry*, Yale University at http://classes.yale.edu/fractals/.

The entire tree, the four major quadrants, and each quadrant's major branches, minor branches, and twigs – all are self-similar with a power law size distribution.

Some three-dimensional natural fractals have the characteristic of maximum surface area for a given volume. This characteristic is very beneficial for filtering and purification processes. You will not be surprised, therefore, to hear that human lungs and wetlands are two more examples of natural structures that are rich in fractals.

4.3.2.1.2 Spatial Fractals in Networks

For a simple two-dimensional lattice network at percolation,[70] we can visually observe spatial fractal behavior. See the following diagram.

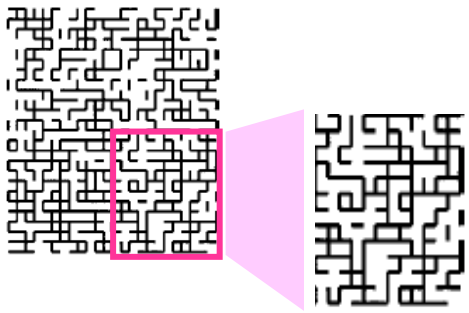


Figure 4.10 Spatial Fractal Behavior in a Lattice Network

The network spatial configuration is statistically self-similar at all length scales.

For more complex network configurations, it is not so easy to show spatial fractal behavior. For many real-world complex networks, spatial fractal characteristics cannot be observed from a network diagram (Figure 4.11).

[70] Network percolation is described in Section 4.3.6.

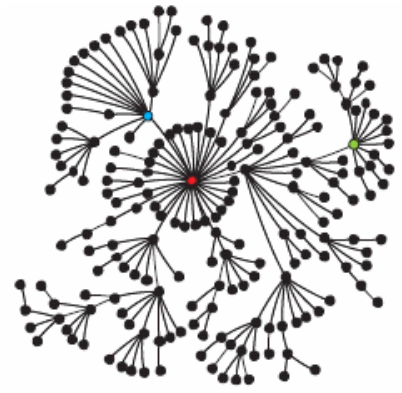


Figure 4.11 Real-world Complex Network

An important reason is that the spatial parameters defined for networks such as the one illustrated in the figure do not correspond to physical dimensions, e.g.:

- ✓ *Path length* is not a physical length – rather it is the number of links traversed from node *i* to node *j*.
- ✓ *Shortest path* from node *i* to node *j* is the path length with the smallest number of links.
- ✓ *Characteristic path length* is the average of all shortest paths in the network.
- ✓ *Network diameter* is the longest of all the shortest paths in the network.

Several methods, however, have been developed to determine the "length" fractal dimension of complex networks that exhibit power-law/fractal behavior. One approach is the "box covering" method.

Description of the *box covering* method[71, 72]: For a given network and a box size ℓ_B, a box is defined as a set of nodes for which the shortest path length ℓ_{ij} between any two nodes *i* and *j* in the box is smaller than ℓ_B. We cover the entire network with such boxes. The minimum number of boxes required to cover the network is denoted by N_B. Note that

[71] Song, Gallos, Havlin, and Makse, *How To Calculate the Fractal Dimension of a Complex Network*, Journal of Statistical Mechanics, March 2007.

[72] Song, Havlin, and Makse, *Self-Similarity of Complex Networks*, Nature, January 2005.

When $\ell_B = 1$, $N_B = N$ *where* $N =$ *the number of nodes in the network*

When $\ell_B \geq \ell_B^{\max}$, $N_B = 1$ *where* $\ell_B^{\max} =$ *the network diameter* $+ 1$

Figure 4.12 shows an example of the box covering procedure for a very simple network. For each value of the box size ℓ_B, we search for the number of boxes needed to tile the entire network such that each box contains nodes separated by a distance $\ell < \ell_B$.

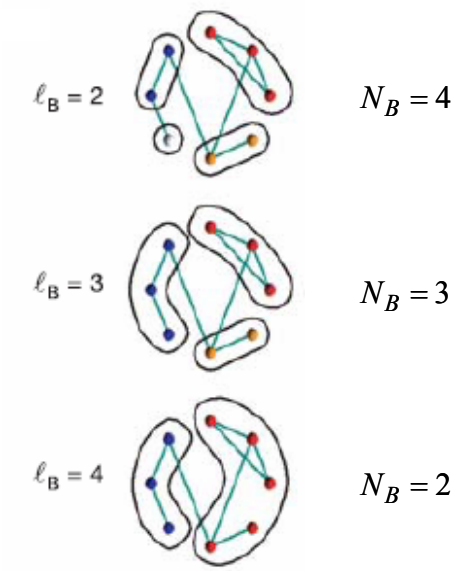


Figure 4.12 Example of Box Covering Procedure

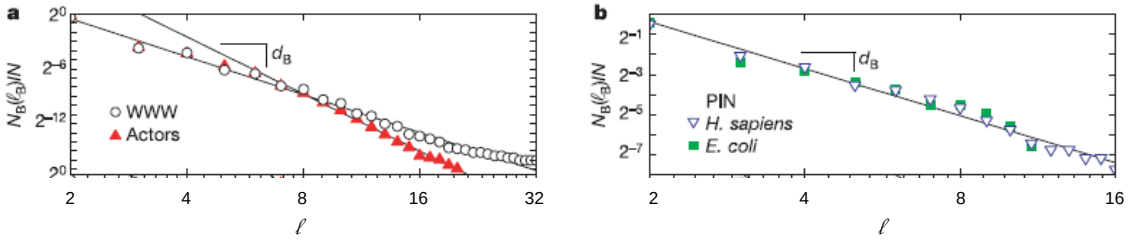
If the network is fractal, it can be shown that

$N_B \sim \ell_B^{-d_B}$

where $d_B =$ the network fractal dimension

$N_B(\ell_B)$ is considered a measure of the distribution of path lengths in the network. (More precisely, it is a measure of the distribution of shortest path lengths in the network.)

Plots of path length distributions using data from several types of real-world fractal complex networks are provided in Figure 4.13.[73] Values of the fractal dimension, d_B, are calculated.



WWW: a part of the WWW composed of 325,729 web pages that are connected if there is a URL link from one page to another.
d_B = 4.1
Actors: a social network where the nodes are 392,340 actors linked if they were cast together in at least one film.
d_B = 6.3

PIN (protein interaction network): the biological networks of protein–protein interactions found in Homo sapiens (946 proteins) and Escherichia coli (429 proteins) linked if there is a physical binding between them.
Homo sapiens d_B = 2.3
Escherichia coli d_B = 2.3

Figure 4.13 Path Length Distributions of Real-world Fractal Complex Networks

The distributions in the figure plot N_B/N, where N = the number of nodes in the network, vs. ℓ_B. Since these are log-log plots, the straight lines fit to the data represent power laws:

$$\frac{N_B(\ell_B)}{N} \sim \ell_B^{-d_B}$$

4.3.2.2 Fractal Behavior in Time

4.3.2.2.1 Principles of Temporal Fractal Behavior

At the beginning of Section 4.3.2, we presented a graphical view of scale-invariant/fractal behavior [Figure 4.7]. The lower panel of that figure [repeated here as Figure 4.14] illustrates the form of a scale-invariant/fractal event time series (ecological fluctuations over time).[74] A

[73] Ibid.

[74] Solé and Goodwin, *Signs of Life – How Complexity Pervades Biology*, Basic Books, 2000.

wide range of events (from very small to very large) and punctuated dynamics are evident. We will now explore temporal fractal behavior in more detail.

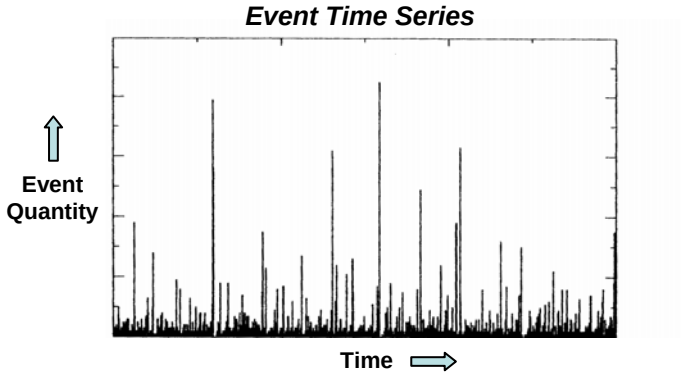


Figure 4.14 Ecological Fluctuations Over Time

Let's start by posing several questions. Does the apparently noise-like time series of Figure 4.14 exhibit scale-invariant/fractal behavior in time? To show that, can we derive an ecological fluctuation distribution function that takes the form of a power law? What is the significance of fractal behavior in time? We will answer these questions and more in the following paragraphs.

First, we'll derive an ecological fluctuation distribution function and show that it takes the form of a power law. Time interval and frequency are inversely related. We know from signal processing theory that the relevant distribution function for a fluctuating time series is the so-called frequency spectrum – which indicates the contribution of each frequency to the overall time series. We can transform a time domain representation (time series) into a frequency domain representation (frequency spectrum) via the Fourier transform, defined by:

$$X(f) = \int_{-\infty}^{+\infty} x(t)\, e^{-i\,2\,\pi f t}\, dt \qquad \textit{where}$$

f *is frequency*

t *is time*

X *is a function of frequency*

x *is a function of time*

The resulting frequency spectrum of a fluctuating noise-like signal has one of a family of forms depicted in the graph of Figure 4.15.[75]

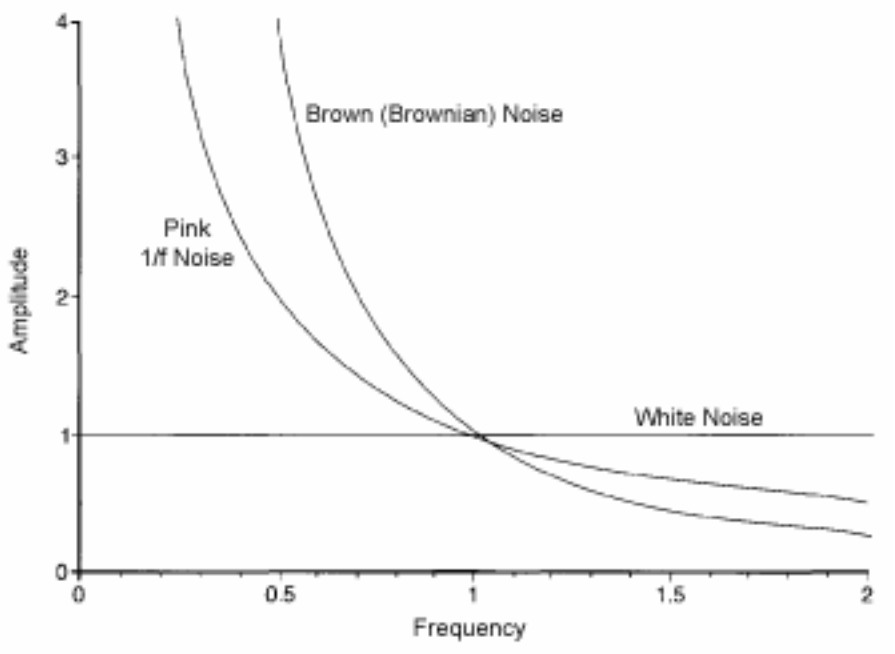


Figure 4.15 Frequency Spectra of Noise-like Time Signals

The family of frequency spectra shown in the graph can be expressed as:

$P(f) = C\, f^{-\beta}$ *where*

$P(f)$ *is the amplitude of the frequency contribution*

f *is frequency*

β *is the spectral* exponent

C *is a* constant

[75] This graph has been adapted from Halley 1996 and Csermely 2006.

The behavior here, therefore, follows a power law – it is scale-invariant/fractal. [The mathematical description above is completely analogous to the spatial scale-invariant/fractal description.] Because the expression for $P(f)$ can, of course, be written as

$$P(f) \approx \frac{1}{f^{\beta}}$$

these dynamics are often called "1/f noise." Different values of the spectral exponent correspond to different "colors" of noise:

$\beta = 0 \quad \Rightarrow \quad$ *white noise*
$0 < \beta < 2 \quad \Rightarrow \quad$ *pink noise*
$\beta = 2 \quad \Rightarrow \quad$ *brown noise*
$\beta > 2 \quad \Rightarrow \quad$ *black noise*

The white and pink colors are assigned by analogy with the visible light spectrum. White light contains equal amounts of all visible light frequencies. By analogy, white noise has a flat spectral density, containing equal amounts of all frequencies. As you can observe from the graph, the pink noise spectrum emphasizes the lower frequencies. By analogy with the visible spectrum, where lower frequencies are red, pink noise is "reddened" relative to white noise – hence it is pink. Brown noise (from Brownian motion) has a spectrum that corresponds to that of a signal doing a random walk.

With respect to complex system behavior, pink 1/f noise is the most interesting and most important. It represents the dynamics of *ecological fluctuations*. Csermely[76] says: "Pink noise is encountered in a wide variety of systems … and is suggested to be a characteristic feature of system complexity." Halley[77] says: "There are good reasons to believe that the structure of environmental fluctuation is well described by a phenomenon called 1/f-noise." … "Recent

[76] Csermely, *Weak Links – Stabilizers of Complex Systems*, Springer, 2006.

[77] Halley, *Ecology, Evolution and 1/f-noise*, Trends in Ecology and Evolution, January 1996.

analyses of data, results of models, and examination of basic 1/f-noise properties suggest that pink 1/f-noise, which lies midway between white noise and the random walk, might be the best null model of environmental variation [fluctuations in time]." Environmental fluctuation "processes behave in an essentially fractal way, having statistical self-similarity on all scales."

Let's discuss further the significance of fractal behavior in time. For ecological fluctuations, there is a *wide range of possible events*. Halley[78] (again) says: "Ecologists expect both rare and common events to be important. The diversity of a desert ecosystem, for example, will be influenced by numerous small changes each day. Some rare events, such as desert storms, will have longer-lasting influence. [Pink] 1/f-noise is a way of describing these kinds of events."

As depicted in the pink 1/f noise frequency spectrum [Figure 4.15], events with low frequency content (gradual changes in time) are more likely than events with high frequency content (abrupt changes in time), but both can occur and do occur. In ecosystems, gradual behavior occurs more often and *is expected* – but abrupt behavior also occurs and *should be expected*. [Recall the *black swans*.] It is well-known from signal processing theory that any fluctuating signal in time can be represented by a sum of sine waves. The frequency of each of these component sine waves is equal to the inverse of its period – the time it takes for one complete oscillation:

$$f = \frac{1}{T} \qquad \textit{where}$$

f *is frequency*

T *is the period*

[78] Ibid.

In a fluctuating signal, low-frequency content means larger time periods and more slowly changing dynamics. High-frequency content means smaller time periods and more rapidly changing dynamics.

Another very significant aspect of scale-invariant/fractal behavior in time is *self-similarity at all scales*. For systems that are fractal in time, we can show analytically that time behavior is self-similar at all time scales. Since frequency behavior of such systems follows a power law and is scale-invariant, and since $f = 1/T$, then time interval T also follows a power law and is scale-invariant:

$$P(f) = C\, f^{-\beta}$$
$$f = \frac{1}{T}$$
$$P(T) = C \left(\frac{1}{T}\right)^{-\beta}$$
$$P(T) = C\, T^{\beta}$$

If we now look at a larger or smaller time scale, that is, if we take

$$T' = b\, T$$

where b is a multiplicative factor, it is not difficult to see that

$$P(T') = C' P(T)$$

or, in other words, a change of scale does not modify the basic statistical behavior. We have fractal self-similar behavior at all time scales.

Figure 4.16[79] illustrates time self-similarity for Ising model magnetization fluctuations.

[79] Solé and Goodwin, *Signs of Life – How Complexity Pervades Biology*, Basic Books, 2000.

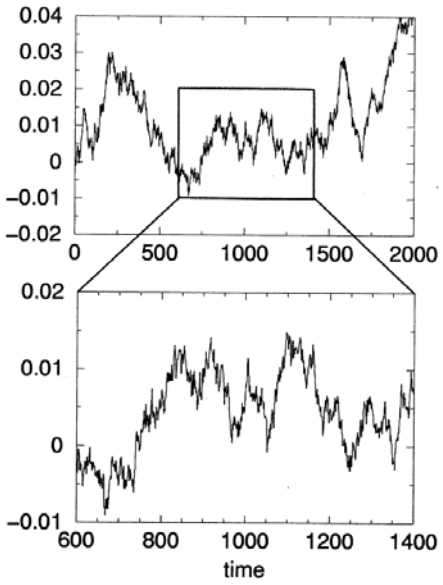


Figure 4.16 Self-Similarity of Temporal Fractals – Ising Model

In an analogous manner, ecological phenomena exhibit self-similarity at all time scales.

A broader type of self-similarity can also be observed. Ecological systems exhibit the same *form* of dynamics *across phenomena* and across time (and space). There seems to be a universality – an equivalence – in ecological network dynamics. We discuss dynamical equivalence across time in the following paragraphs.

Dynamical Equivalence across Time

Ecological system networks exhibit the same *form* of dynamics in very different time frames. In this sense, shorter-term operational dynamics match longer-term developmental dynamics. There is a general *dynamical equivalence* across time frames. The same, apparently universal, principles apply.

Consider an ecological network of species. Different phenomena (functions/processes) occur in different time frames. We can characterize the functions/processes in terms of *events* that occur. In short-term operational time frames we have, for example, biomass/energy propagation events. In long-term developmental time frames we have, for example,

macroevolution events, i.e., extinction events and speciation events. We can describe system dynamics therefore, in any time frame, in terms of:

- Event distribution (likelihood of event vs. event variable).
- Event time series (event quantity vs. time).

This representation of ecological system dynamics has the same form across all time frames. It is in this sense that we have dynamical equivalence across time. Figure 4.17 illustrates the time series behavior of the network of species for short-term and long-term time frames.

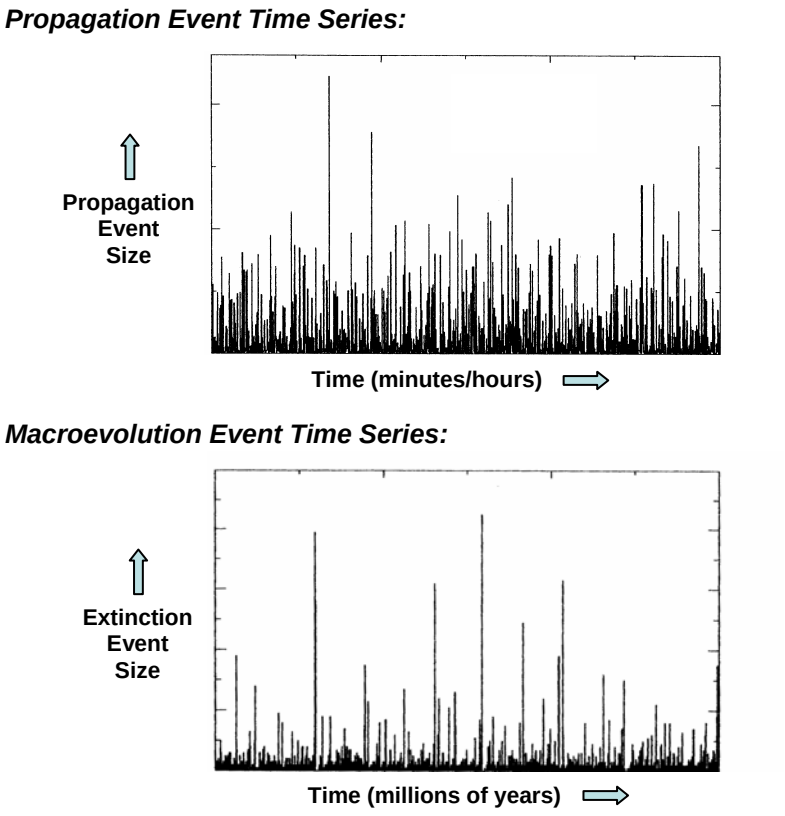


Figure 4.17 Dynamical Equivalence across Time Frames

The behavioral dynamics of the network have the same form across these vastly different time frames. The upper graph shows ecological fluctuations in a short time interval (short-term dynamics). The lower graph shows ecological fluctuations in a very long time interval (very long-term dynamics). The two graphs reflect the same statistical principles.

4.3.2.2.2 Heart Rate Dynamics: An Example of Temporal Fractal Behavior

The sources for this example are Solé and Goodwin[80] and Ivanov, Goldberger, et al.[81]

Health "is revealed as a subtle emergent property of the dynamic complexity of living organisms" [Solé and Goodwin]. Health is the "normal" condition and a dynamic attractor for living systems. Can we identify characteristics of the "healthy state"?

Consider heart function. In healthy human beings, heart rate is not steady. It shows significant spontaneous variation. As part of the body's physiological network, a healthy circulatory subsystem is responsive to what is going on in the body's other subsystems and in the body's environment. Healthy heart function has characteristics of agility and responsiveness. We will see that healthy heart dynamics exhibit scale-invariant/fractal behavior in time.

A group at Beth Israel Hospital in Boston, under the direction of Ary Goldberger, has been investigating heart dynamics with the tools available from nonlinear analysis and complexity theory. They have considered both healthy individuals and those with various types of cardiac arrhythmias. Time series of instantaneous heart rate for a healthy subject and for two individuals with cardiac disease show important differences. See Figure 4.18.

[80] Ibid.

[81] Ivanov, Goldberger, et al, *Scaling Behavior of Heartbeat Intervals obtained by Wavelet-Based Time-Series Analysis*, Nature, Volume 383, 1996.

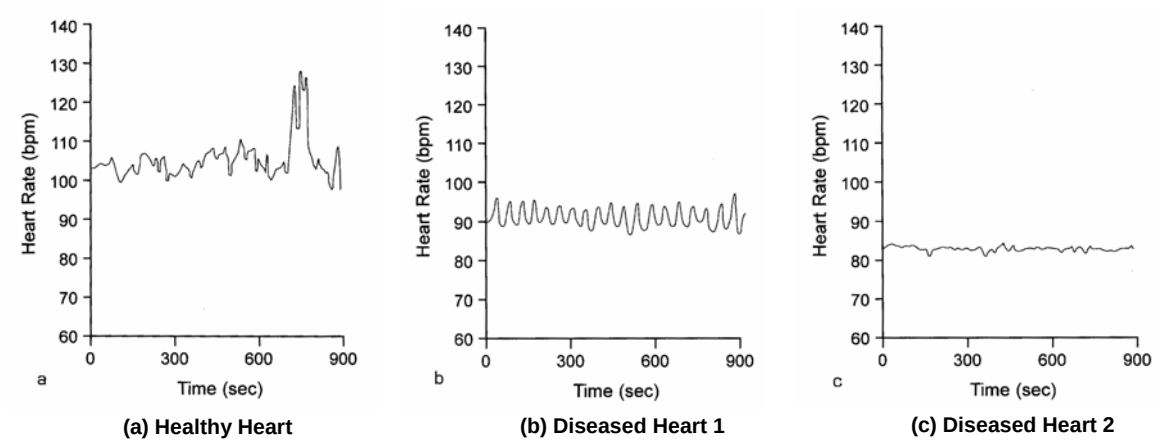


Figure 4.18 Heart Rate Time Series

The healthy heart exhibits a wide range of variability and responsiveness (punctuated dynamics). The diseased heart behavior is much more orderly: the heart rate of one shows an orderly oscillation and the other is nearly constant. The diseased hearts are not responding effectively to their environments. "Here, too much order is a sign of danger" [Solé and Goodwin].

Fourier analysis was applied to the three time series – to yield a frequency spectrum for each of the cases. See Figure 4.19.

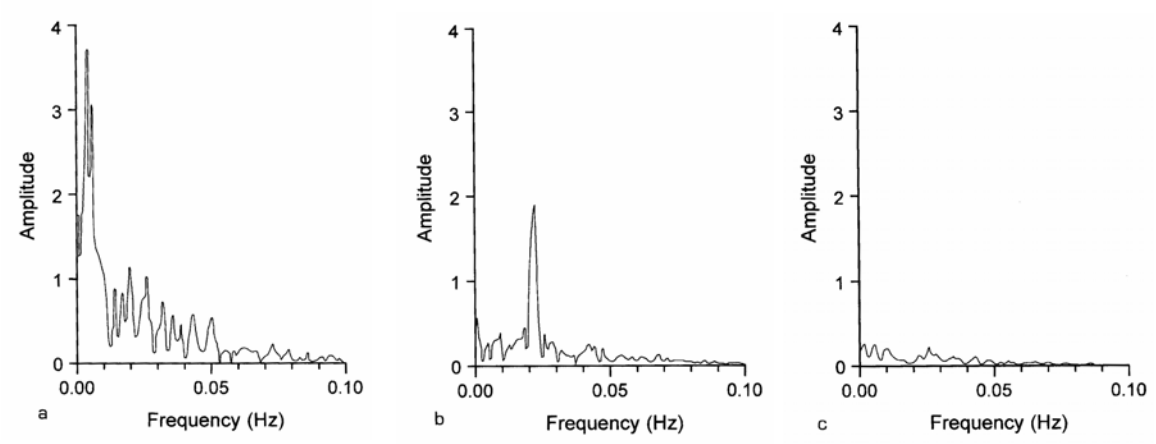


Figure 4.19 Heart Rate Frequency Spectra

The healthy heart result suggests a pink *1/f noise* spectrum -- a broad range of frequencies with emphasis on the lower frequencies. This spectrum suggests temporal scale-invariant/fractal dynamics. The diseased heart cases, as expected, do not exhibit such behavior.

To illustrate the healthy heart self-similarity in time, Goldberger examined different time scales as shown in Figure 4.20.

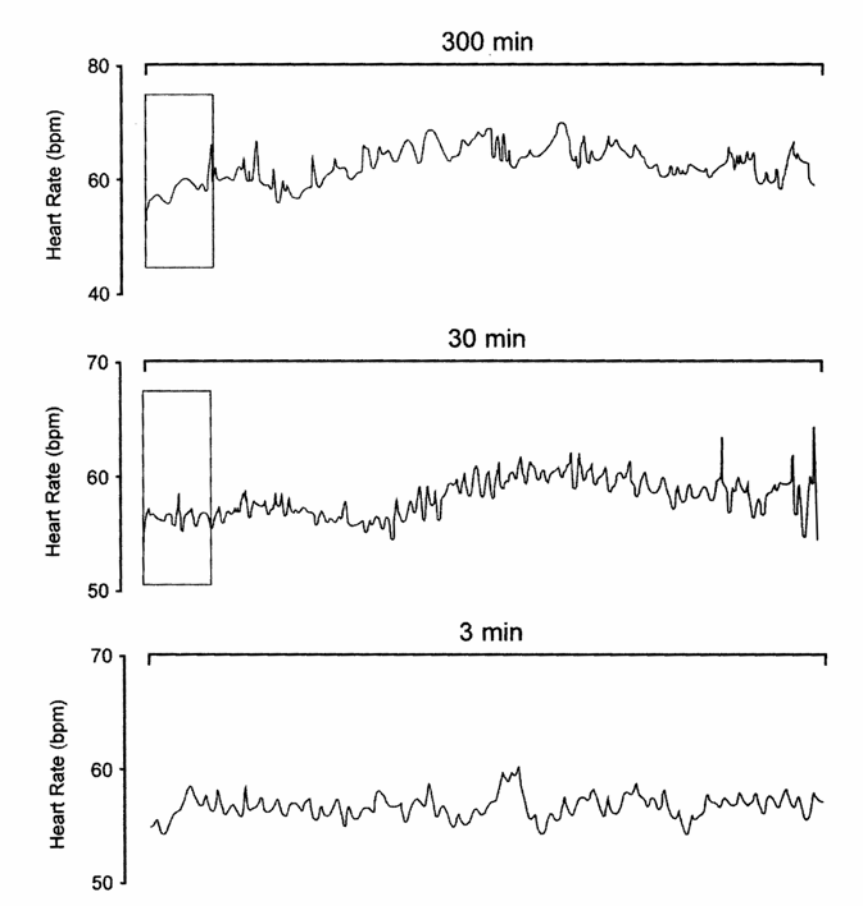


Figure 4.20 Healthy Heart Self-Similarity in Time

The one-tenth boxed area of an upper time series plot is expanded in the next lower time series plot. The resulting three time series at three different time scales are statistically self-similar. At each scale, we see similar fluctuations.

4.3.3 Local-to-Global Dynamics

Local-to-global dynamics, another important characteristic of ecological networks, are consistent with scale-invariant/fractal behavior. Our discussion here has two parts: (a) "small-world" networks as facilitators of the dynamics and (b) the role of system regulation/adaptation and propagation.

4.3.3.1 Small-World Networks as Facilitators of Local-to-Global Dynamics

Many real-world complex network systems, including ecological systems, are "small-world" network systems – i.e., they possess *high clustering* and *short characteristic path lengths*.[82] These small-world attributes facilitate local-to-global dynamics. The high clustering attribute is extremely effective for local processing of stimuli. The short path length attribute provides efficient propagation channels between distant parts of the system, thereby supporting any dynamical process taking place on the network that requires global propagation and processing. In ecological systems, these capabilities are essential to facilitate effective handling of stimuli from the environment. In response to some stimulus – say an input of energy, biomass, or information – an ecological network processes locally and/or connects globally as appropriate until the "processing" of the stimulus is completed. Solé and Bascompte[83] say it this way: In ecological systems, "flows of energy [or biomass, etc.] enter into the system and are dissipated at different, interconnected scales."

4.3.3.2 The Role of System Regulation/Adaptation and Propagation

Regulation/adaptation and propagation are two key core functions of complex ecological systems. The need for *regulation/adaptation* is profound – it is required for system survival. *Propagation* is the fundamental and essential means of interaction among nodes in an ecological

[82] Yackinous, *The Structure of Complex Networks*, November 5, 2007.

[83] Solé and Bascompte, *Self-Organization in Complex Ecosystems*, Princeton Univ Press, 2006.

network and between the network and its environment. The two functions work together to achieve, in a local-to-global manner, "*life as a relaxation phenomenon*." [84]

Life as a Relaxation Phenomenon

Ecological system regulation/adaptation and propagation can be viewed as the means of handling perturbations that impinge upon an ecological network. To survive, every living system must respond (adapt) effectively to stimuli from its environment. The stimulus can be considered a source of tension, and the response as a relaxation. When an ecological network receives a stimulus (e.g., energy, biomass, information) from its environment, the input is often propagated and dissipated locally – yielding local relaxation in the system. When some stimulus cannot be dissipated locally, tension persists. As the stimulus inputs continue, tension gradually increases. Local tensions can accumulate and may develop to a point where global propagation (of the energy/matter/information) suddenly occurs – yielding global relaxation in the system. The set of local and global system relaxation events exhibit scale-invariant/fractal behavior in space and in time. Any event can have small or large extent and duration – as well as anything in between.

Peter Csermely says that living system functions "cannot be performed in the absence of widespread network communication." They cannot be performed without local-to-global dynamics.

Local network processing can spawn global processing. Global network processing can potentially change the state of any involved network node – which can affect subsequent local processing. Global behavior can spawn local behavior. We thereby have local-to-global-to-local dynamics.

[84] This view is inspired by Peter Csermely, *Weak Links*, Springer, 2006.

4.3.4 Indirect Effects

Due to the work of Dr. B. C. Patten and colleagues at the University of Georgia and elsewhere, the dominance of indirect effects in ecological networks is widely recognized. Indirect effects are an important characteristic of the ecological network dynamics principles being developed here.

Indirect effects are exhibited via ecological network propagation events. A propagation event is often a *cascade* that can involve many network nodes and propagation paths – many of which are indirect paths. I hypothesize that propagation event dynamics are punctuated, fractal, and have local-to-global reach. I expect indirect effects dynamics to be consistent with propagation event and path length dynamics. In particular:

- Indirect effects with shorter indirect propagation paths are more likely, but indirect effects with longer propagation paths can occur and do occur.
- Indirect effects dynamics are punctuated rather than continuous.
- Indirect effects exhibit fractal behavior.
- Indirect effects support local-to-global processing.

Indirect effects dynamics obviously have a lot to do with propagation path length dynamics. Ecological network structure and the ecological processes that operate on that structure can inform us about path length dynamics and, therefore, indirect effects dynamics.

Consider network structure. As noted in the previous subsection, ecological networks have scale-invariant, small-world network structure.[85] Such networks have high clustering and relatively low characteristic path length (typically less than 6 or so). Local connections are via direct paths (path length = 1) or short indirect paths (path length > 1). Global connections are

[85] Yackinous, *The Structure of Complex Networks*, November 5, 2007.

mostly via indirect paths of short to moderate length. Ecological network structure is consistent with shorter, rather than longer, propagation path lengths.

The table[86] of Figure 4.21 provides empirical evidence of the low characteristic path length and high clustering associated with real-world ecological networks. Focus on the two food web entries in the figure (the shaded entries). The table indicates network size (number of nodes), the characteristic path length ℓ, and the clustering coefficient C for each network. For comparison we have also included the characteristic path length ℓ_{random} and clustering coefficient C_{random} of a comparable random network (i.e., a random network having the same number of nodes and links as the real network). In my paper on network structure [see the footnote], I explained that a low characteristic path length is one that is close to that of a comparable random network and a high clustering coefficient is one that is much higher than that of a comparable random network. These effects can be clearly seen in the table of Figure 4.21.

[86] The table has been adapted from Albert and Barabási, *Statistical Mechanics of Complex Networks*, 2002 and is described in Yackinous, *The Structure of Complex Networks*, November 5, 2007.

Network	**Size** (#nodes)	ℓ	ℓ_{random}	C	C_{random}
World Wide Web	153127	3.1	3.35	0.1078	0.00023
Los Alamos Lab co-authorship	52909	5.9	4.79	0.43	1.8×10^{-4}
Medical co-authorship	1520251	4.6	4.91	0.066	1.1×10^{-5}
Physics co-authorship	56627	4.0	2.12	0.726	0.003
Neuroscience co-authorship	209293	6	5.01	0.76	5.5×10^{-5}
E. coli substrate graph	282	2.9	3.04	0.32	0.026
E. coli reaction graph	315	2.62	1.98	0.59	0.09
Ythan estuary food web	134	2.43	2.26	0.22	0.06
Silwood Park food web	154	3.40	3.23	0.15	0.03
C. Elegans	282	2.65	2.25	0.28	0.05
Electric power grid	4941	18.7	12.4	0.08	0.005

Figure 4.21 Empirical Data for Several Real Networks

Next consider two real-world ecological processes that operate on that network structure – and what they tell us about path length and indirect effects dynamics. Both of the following examples suggest that, although paths of all lengths can and do occur, shorter path lengths are more likely. They suggest that path length dynamics, and therefore indirect effects dynamics, exhibit power-law/fractal behavior.

Lévy Flights

Here's some corroborating evidence that propagation path lengths have a power-law/fractal distribution.[87] In foraging, what is the best statistical strategy to adopt in order to

[87] Vishwanathan et al, *Optimizing the Success of Random Searches*, Nature, October 1999.

search efficiently for "target sites"? Analysis shows that search efficiency depends on the probability distribution of flight lengths (or flight times) taken by a forager. Results indicate that a long-tailed power-law distribution of flight lengths, corresponding to Lévy flight motion, represents an optimal strategy. Lévy flights are characterized by the following distribution function:

$$P(\ell) \approx \ell^{-\mu}$$
where $\ell = the\ flight\ length$
and $\mu = 2$

This power-law distribution is depicted in the log-log plot of Figure 4.22. These results have been verified experimentally.

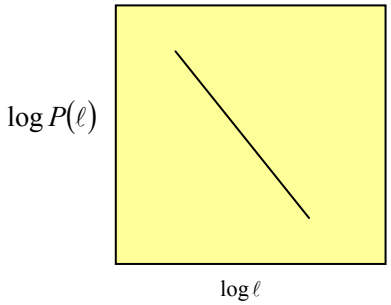


Figure 4.22 Lévy Flight Probability Distribution

Ant Colonies Find and Use Shortest Routes

Here is evidence that when multiple paths (with differing path lengths) exist between pairs of nodes in an ecological network, the paths with the shortest lengths are more likely to be used.

Ant colonies solve the "shortest route" problem (aka the Operations Research traveling salesman problem) when traveling to/from a food source.[88] When foraging for food, an ant secretes a chemical called *pheromone* to mark its trail. Shorter routes are traveled in less time

[88] See Schiff, *Cellular Automata*, Wiley-Interscience, 2008.

and get more pheromone deposited more quickly than longer routes. Other ants prefer to travel a trail richer in pheromones, so the shorter routes get reinforced. After a brief transient period, this process finds the shortest route – which all ants will then use. Following this same process, an ant colony will find the shortest route from the nest to each food source as well as the shortest route between food sources – that is, the ant colony solves the OR shortest route/traveling salesman problem.

This is no small feat. To visit each of the destination sites (food sites), and then return to the origin site (nest) – there are effectively $(n - 1)! / 2$ possible routes. With, say, 15 sites – there are over 43 billion possible routes.

A cellular automata model has been developed by Dorigo and Gambardella[89] that mimics the behavior of ants and achieves near-optimal to optimal solutions to the traveling salesman problem. This cellular automata model is as good as or better than competing Operations Research solution approaches. [My modeling approach is cellular automata based. See Chapter 5.]

4.3.5 Critical Connectivity

Critical connectivity is another characteristic of ecological network dynamics. Ecological networks and other complex system networks can reach *critical connectivity* in the domain of attraction. This connectivity is sparse connectivity. Ecological networks are *sparsely connected networks* – with just enough connections to provide full or nearly full node-to-node access via direct and (mostly) indirect paths.

[89] Dorigo and Gambardella, *Ant Colonies for the Traveling Salesman Problem*, Biosystems, Volume 43, 1997.

Stuart Kauffman has conducted several modeling and analysis efforts in this area of investigation using Boolean network models. We'll summarize his work in the following paragraphs.

Boolean Network Models

Stuart Kauffman[90] has modeled and analyzed large, complex, randomly constructed Boolean (2 states per node) directed networks. His "NK model" analysis considers networks with n nodes and k inputs per node (k is the node mean in-degree). Sample networks with given n and k values are assembled – and the behavior of those networks is studied. For large n, the results indicate:

$k = 1$ ⇨ nothing interesting happens ("too orderly").

$k = n$ ⇨ massively chaotic – small perturbations cause massive changes in behavior.

$k = 4$ or 5 ⇨ chaotic behavior.

$k = 2$ ⇨ flexible "order arises, sudden and stunning."

For these networks, a node mean in-degree value of 2 represents critical connectivity. With large n and $k = 2$, the network *looks* like a "scrambled jumble" but it *behaves* in an orderly fashion. "These networks are not *too* orderly. Unlike the $k = 1$ network, they are not frozen like a rock, but are capable of complex behaviors." Kauffman says that his results "apply to networks of all sorts." This "is almost certainly merely the harbinger of similar emergent order in whole varieties of complex systems."

A related effort is Stuart Kauffman's NKC macroevolution model[91] – where n is the number of traits of a species, k is the number of other traits a given trait is affected by, and c is the number of couplings between species. Some results from Kauffman's modeling/analysis are

[90] Stuart Kauffman, *At Home in the Universe – The Search for Laws of Self-Organization and Complexity*, Oxford University Press, 1995.

[91] Kauffman's work is described in Solé and Goodwin, *Signs of Life*, Basic Books, 2000.

depicted in Figure 4.23. [Note that, in the figure, the ordinate "Connectivity" is related to the number of species connections – which is not the typical definition of connectivity.]

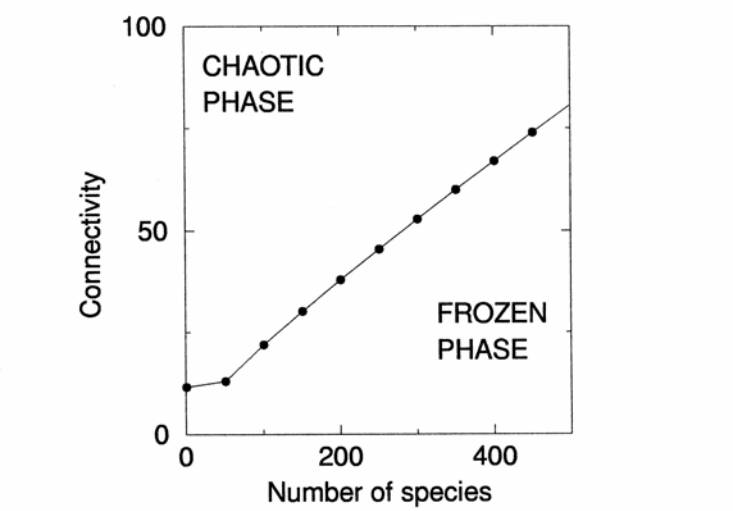


Figure 4.23 Macroevolution Critical Connectivity

For low connectivity, nothing much happens (little or no activity). For high connectivity, nothing useful happens. There is much disordered activity – Red Queen dynamics all the time. At a critical connectivity, there are coevolutionary events of all sizes – that follow a power law – that match observed behavior in the fossil record. Macroevolution activity apparently occurs in the context of a species network at critical connectivity.

Kauffman says further that "astonishingly simple rules, or constraints, suffice to ensure that unexpected and profound dynamical order emerges spontaneously." "If the network is 'sparsely connected' [e.g., $k = 2$], then the system exhibits stunning order." "Our intuitions about the requirements for order have, I contend, been wrong for millennia. We do not need careful construction; we do not require crafting. We require only that extremely complex webs of interacting elements are sparsely coupled."

4.3.6 Network Percolation

Critical connectivity of ecological networks and other complex networks is achieved in a phase transition phenomenon known as *network percolation*. As the number of links increase, a network rather abruptly transitions from fragmented to connected. When the number of links is small, a network is likely to be fragmented into many small clusters of nodes. As links increase, the clusters grow, at first by connecting to isolated nodes and later by coalescing with other clusters. A phase transition occurs when many clusters crosslink spontaneously to form a single *giant cluster*. The network transitions from a state in which most clusters are small to a state in which all or nearly all nodes are joined together in the single giant cluster. Note that network percolation is abrupt – a threshold phenomenon – separating two well-defined phases. In the subcritical phase, we have short-range (local) interactions. In the critical phase, we add long-range (global) interactions.

Traditional percolation theory was originally studied in physics and statistical mechanics. It was first applied to lattices and later to random graphs. The reason for this, in part, is that these two cases are the ones amenable to mathematical and numerical analyses. In the following paragraphs, therefore, we look at a random network example and a lattice network example, respectively. It is expected that the general principles noted there will be useful for networks of all sorts – including ecological networks.

Newman[92] depicts network growth and transition behavior for an undirected random network in Figure 4.24. [Note that, in the figure – and in much of the network literature – "clusters of nodes" are termed "components."]

[92] Newman, *The Structure and Function of Complex Networks*, SIAM Review, May 2003.

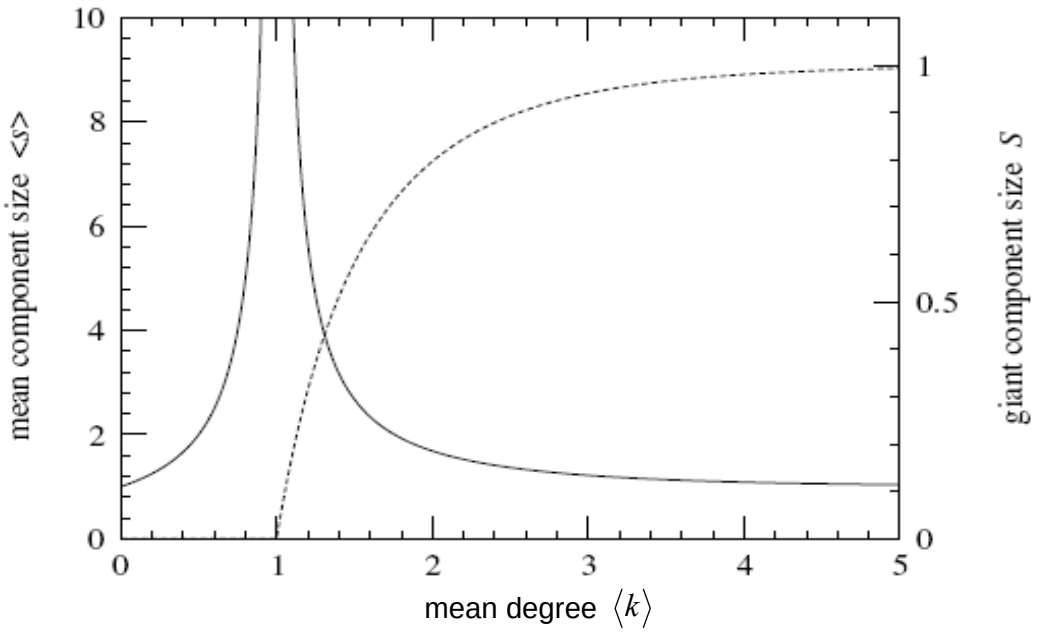


Figure 4.24 Network Percolation in an Undirected Random Network

We define $< s >$ as the mean component size, S as the fraction of the graph occupied by the giant component, and $< k >$ as the mean degree. We plot the mean component size (solid line), excluding the giant component if there is one, and the giant component size (dotted line), for a Poisson random graph. The phase transition occurs at $< k > = 1$. This is also the point at which $< s >$ diverges. In the figure, you can see the dramatic growth of mean component size as we approach the transition – and then the collapse as the giant component appears. Adding more links after the transition occurs changes the giant component size very little – although the additional links could significantly change network behavior from order to disorder (see Figure 4.23).

The critical value of k in the Boolean analysis of the previous subsection is $k = 2$, while in the percolation analysis of this subsection it is $k = 1$. The factor of 2 difference is explained by the fact that in the Boolean analysis the network is *directed* and in the percolation analysis the

network is *undirected*. Each adjacent connection in an undirected network equals two adjacent connections in a directed network.

Next consider a lattice network analysis from Albert and Barabási.[93] Figure 4.25 shows two snapshots of a regular lattice whose edges (links) are present with probability p and absent with probability 1 - p. For small p, only a few edges are present and only small clusters of nodes connected by edges can form. At larger values of p – specifically at critical probability p_c , called the *percolation threshold* – a percolating cluster of nodes connected by edges appears.

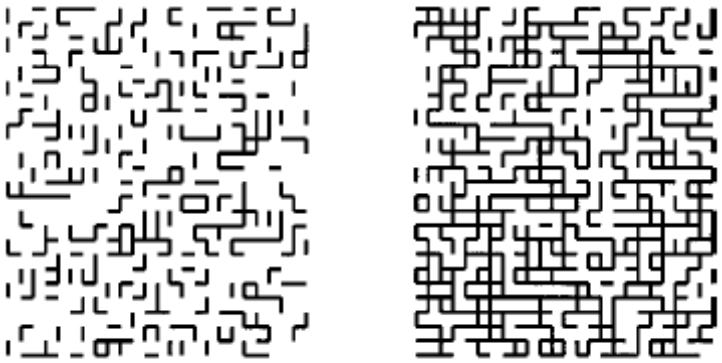


Figure 4.25 Percolation in a Two-Dimensional Lattice Network

For this illustration of percolation, nodes are placed on a 25x25 square lattice. In the snapshot on the left, which is below the percolation threshold, the connected nodes form isolated clusters. In the snapshot on the right, which is slightly above the percolation threshold (and slightly above critical connectivity), the network percolates.

4.4 Hypothesis

We have synthesized an operational view and a characteristics view of ecological network dynamics. The operational view shows that ecological networks are "flickering"

[93] Albert and Barabási, *Statistical Mechanics of Complex Networks*, Reviews of Modern Physics, January 2002.

networks. They continually change with time. The characteristics view shows that ecological networks possess the following behavioral characteristics:

- Punctuated behavior
- Fractal behavior in space and time
- Local-to-global propagation
- Dominance of indirect effects

We hypothesize that these two views comprehensively describe the behavioral dynamics of ecological networks. We must now test and corroborate (or falsify) this hypothesis. We have developed a sophisticated model (Chapter 5) for that purpose.

To accomplish the hypothesis testing, we proceed as follows. We model the operation of a propagating ecological network. We proceed to collect operational data and perform analysis in five areas as outlined here:

- Network operational propagation flow
  - ✓ *Construct node-and-link flow diagrams at selected model time steps.*
  - ✓ *Compile input, output, stock, and flow value histories.*
- Propagation events
  - ✓ *Determine propagation event size for each model time step.*
  - ✓ *Construct propagation event time series.*
  - ✓ *Construct propagation event distributions.*
- Path Length
  - ✓ *Calculate path lengths.*
  - ✓ *Construct path length time series.*
  - ✓ *Construct path length distributions.*

- Indirect Effects
  - ✓ *Use path length data to perform indirect effects analysis.*
  - ✓ *Demonstrate dominance of indirect effects.*
- Network Connectivity
  - ✓ *Construct node degree distributions.*
  - ✓ *Examine network connection densities and coverage.*

The modeling and analysis results will test our hypothesis. The network node-and-link flow diagrams and the input, output, stock, and flow value histories will provide direct visual evidence of network operation and test the operational view. They will also provide some evidence of network behavioral characteristics. The remaining four categories of analyses will fully test the characteristics view. Specifically, the propagation event and path length statistics will test for the presence of punctuated, fractal, and local-to-global behaviors. Indirect effects analysis will determine whether indirect effects are dominant and also whether they are punctuated, fractal, and enablers of local-to-global propagation. Network connectivity analysis will test for node degree fractal behavior. We will also be able to examine related network connection traits. Our hypothesis, therefore, will be fully and comprehensively tested.

The requirements, design, and development of the ecological network dynamics model are covered in Chapter 5. The resulting model software includes the modeling of the operation of propagating ecological networks, the required analysis activities, and the needed graphics generation capabilities.

CHAPTER 5

A NEW APPROACH TO MODELING ECOLOGICAL NETWORK DYNAMICS

5.1 Model Requirements

Our objective is to model complex real-world ecological networks as discrete dynamic systems in order to observe and analyze their behavior at each model simulation time step as well as their composite behavior over all model simulation time steps.

5.1.1 Model Features

Model feature summary list:

- Cellular automata based
- Spatial as well as relational node interactions
- Real-world network structure characteristics
- Preferential attachment capabilities
- Various node stock and node propagation rules
- Aspects of both aggregate and individual-based modeling
- Dynamic network model is linked to an underlying ecological network compartment model

Discussion of features:

The model (actually a family of models) is cellular automata based. Cellular automata provide an excellent metaphor for complex ecological networks. The cells represent the network nodes and the cell interactions represent the network links. A cellular automaton can be described as an array of interacting *finite state machines*. In the network context, each finite-state-machine node is a discrete dynamic system. The total network (array), therefore, can be a

very complex discrete dynamic system. Note that we do not adhere strictly to all definitional restrictions of cellular automata. For example, we do not require that all cells be identical and that they all be updated simultaneously at each model time step.

Much traditional network modeling (ecological and otherwise) does not include physical space and length considerations – rather only relational length considerations (i.e., the path length between any two nodes is defined as the *number of links* traversed). Since our cellular automata based models are defined on a physical grid, we can explicitly represent both spatial and relational aspects of networks.

We design the network model to reflect the structure characteristics of real-world ecological networks. The cellular automata structure can most straightforwardly model lattice networks. Such networks have a *high clustering coefficient* [i.e., a node's neighbors are likely to also be neighbors of each other] and high *characteristic path length* [defined as the average shortest path length taken over all pairs of nodes in a network]. Most real-world ecological (and other) networks, however, are not lattice networks – they are scale-invariant (with respect to node degree) small-world networks with high clustering and *low* characteristic path length. The high clustering is consistent with the modularity found in many real networks. The low characteristic path length facilitates the efficient global (network-wide) interaction required by real networks. To achieve these real-world structure features in our network model, we augment the usual cellular automaton neighborhoods – following the lead of Watts and Strogatz[94] and Newman and Watts[95]. For node interaction, we use local von Neumann neighborhoods, local extended von Neumann neighborhoods, and a global neighborhood concept. (The authors just

[94] Watts and Strogatz, *Collective Dynamics of 'Small-World' Networks*, Nature, June 1998.

[95] Newman and Watts, *Scaling and Percolation in the Small-World Network Model*, Physical Review E, December 1999.

noted use the term ”short cuts” to describe such global access.) Figure 5.1 illustrates these neighborhoods for node interaction.[96]

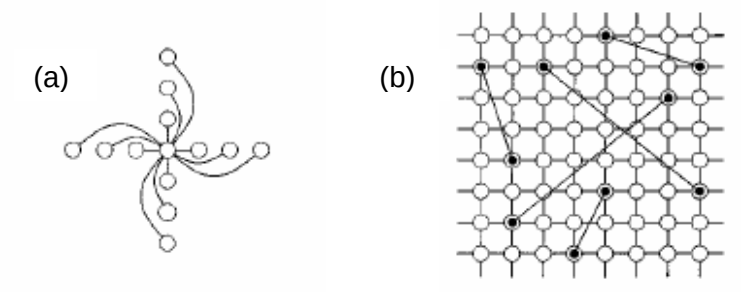


Figure 5.1 Neighborhoods for Node Interaction

Part (a) of the figure illustrates three types of von Neumann neighborhoods – with grid spacing ranges from the center node of 1, 2, and 3, respectively. We will refer to the range = 1 nearest-neighbor case as a *local neighborhood* and the range = 2 next-nearest-neighbor case as a *local extended neighborhood*. We will not use the range = 3 case. Part (b) of the figure illustrates some node-to-node *global neighborhood* connections.

Albert and Barabási[97] were probably the first to identify “preferential attachment” as a mechanism that dynamically generates scale-invariant small-world complex networks. The process produces the “hub” nodes often seen in many types of real networks. In ecological networks, *keystone species* can be regarded as hubs. In our modeling, we use compartment and node preferential attachment in both local neighborhoods and global neighborhoods. This should help the model achieve the scale-invariant degree distribution – and the high clustering and low characteristic path length – observed in real-world complex networks.

We can implement various node stock and node propagation rules. Our models are discrete dynamic models. The node stock and node propagation rules are therefore discrete. A

[96] The figure is adapted from Newman and Watts 1999.

[97] Albert and Barabási, *Statistical Mechanics of Complex Networks*, Reviews of Modern Physics, January 2002.

node will accumulate stock in unit increments until a threshold (thd) is reached. At that point, the node will propagate a unit flow to each of a number of destinations based on criteria defined and specified in the model (including neighborhoods and preference as discussed above) until the total propagation flow quantity (npfq, where npfq ≤ thd) is exhausted. The values of thd and npfq can be varied. Note that stock and flow rules and values are kept quite simple in our model. Stock values can be 0, 1, 2, … , thd. All individual flows are unit (value = 1) flows.

We include aspects of both aggregate and individual-based[98] modeling. Typical ecological network compartment models (stock and flow models) are aggregate models. We model the aggregate compartments – and individual nodes within the compartments. We observe and analyze both aggregate and individual-based behavior.

Our dynamic ecological network model is linked to an underlying ecological network compartment model. Any pre-existing compartment model is a candidate for this. The compartment model diagram, adjacency matrix, input vector, and output vector are particularly relevant. For our model development effort, we have selected a particular compartment model – and we describe it in the next subsection.

5.1.2 Underlying Ecological Network Compartment Model

Our dynamic ecological network model is linked to a pre-existing underlying ecological network compartment model. The compartment model that we use is an estuary aquatic model that was developed by Chip Small, Nicole Gottdenker, and Bill Yackinous in 2005 as a course project in ECOL 8580 – Dr. B. C. Patten's course in systems ecology at the University of Georgia. The compartment model diagram, adjacency matrix, input vector, and output vector are provided here.

[98] Individual-based modeling is the term often used in Ecology. Agent-based modeling seems to be the more broadly used term – across disciplines.

The compartment model diagram is provided in Figure 5.2.

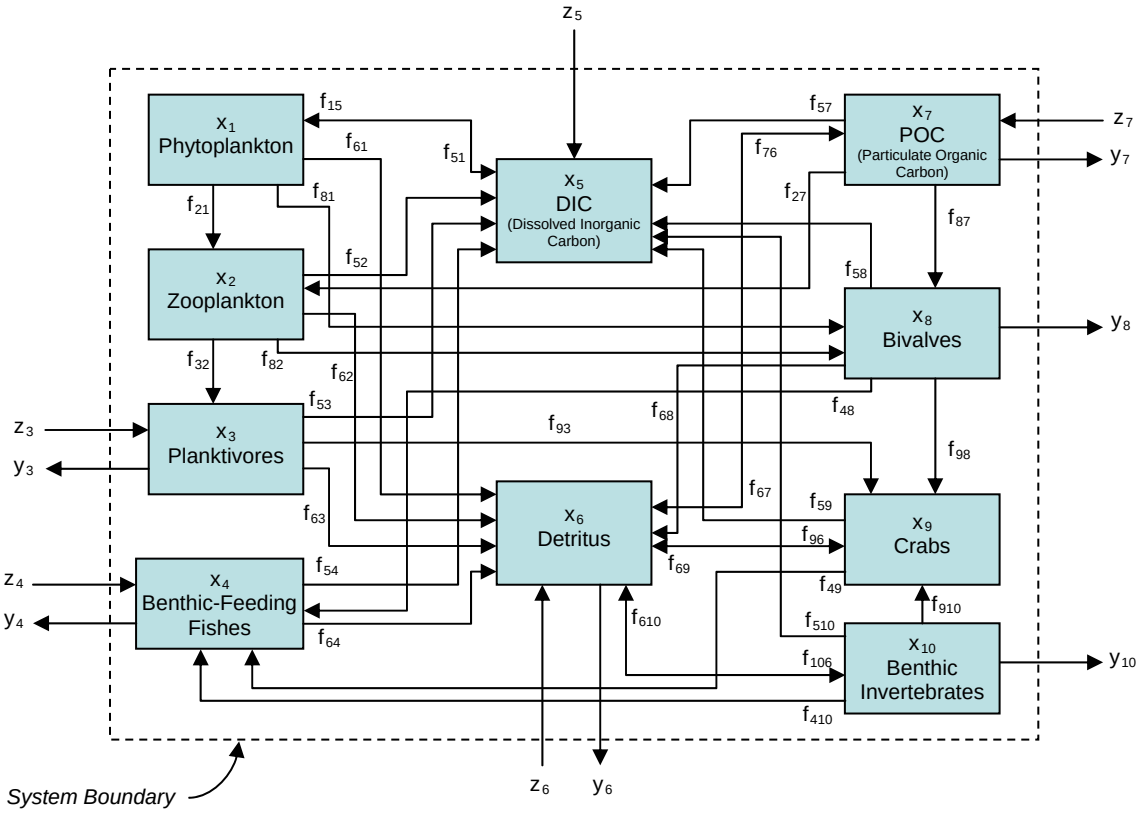


Figure 5.2 Estuary Aquatic Compartment Model

The adjacency matrix, input vector, and output vector follow.

Adjacency Matrix:

$$\mathbf{A} = (a_{ij}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Input Vector:

$$\mathbf{Z} = (z_j) = \begin{bmatrix} 0 \\ 0 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Output Vector:

$$\mathbf{Y} = (y_i) = \begin{bmatrix} 0 & 0 & y_3 & y_4 & 0 & y_6 & y_7 & y_8 & 0 & y_{10} \end{bmatrix}$$

5.1.2.1 Compartment-to-Node Transition

We model the dynamics of individual nodes within the compartments – and we maintain a view of the dynamics of the aggregate compartments as well. The node network contains 100 nodes and there is an average of 10 nodes per compartment. The transition from network compartments to network nodes includes some important connection-based considerations. Network connection density (also called connectance) is given by

$$Network\ density = \frac{\langle k \rangle}{n-1} \cong \frac{\langle k \rangle}{n}$$

where

$\langle k \rangle$ = *mean vertex* (*compartment or node*) *degree*

n = *number of vertices in the network*

In our case, the number of vertices is being increased by a factor of 10. To keep the network density constant, the mean vertex (node) degree must also *increase* by a factor of 10. That would occur in the following scenario: if compartment C1 connects to compartment C2, then every node in C1 connects to every node in C2. On the other hand, if we want to keep mean vertex degree constant, network density would *decrease* by a factor of 10. The most realistic situation may lie somewhere between these two extremes. In a real-world ecological network, we would not expect every individual in C1 to interact with every individual in C2. Initially at least, we will hold network density constant – and see how it goes.

5.1.2.2 Predetermined/Candidate/Operational Adjacency Matrices

There are also considerations regarding the interpretation of adjacency matrices in the transition from conventional compartment modeling to the network dynamics modeling presented here. In conventional compartment modeling, direct propagation links are determined *a priori* and are represented in the adjacency matrix. The network dynamics modeling presented here differs. The operational propagation links among nodes are not determined *a priori* – they

are part of what is *being modeled* and are known only *a posteriori*. In this approach, the usual adjacency matrix represents *candidate* propagation links. In addition to candidate adjacency matrices, later we introduce the concept of *operational* adjacency matrices – which represent realized propagation links.

The values of network density and mean node degree associated with a candidate adjacency matrix will, in general, differ from the values associated with operational adjacency matrices. Candidate adjacency matrix values serve as an upper bound for operational adjacency matrix values.

5.1.3 Analysis Requirements

A key objective of this effort is to observe and analyze model behavior at each model simulation time step as well as composite behavior over all simulation time steps.

Model analysis activities:

Generate network node-and-link propagation diagrams – diagrams that apply to an individual time step and diagrams that are cumulative over time steps – and display them for selected time steps.

Determine network cumulative flow values. The flows shall be graphically depicted in network propagation diagrams displayed at selected time steps.

Track and record network input, output, and stock values at each time step – and plot and display cumulative values over time.

Calculate and plot network propagation event data. Network propagation event size is equal to the number of nodes involved in propagation – at a given time step. Develop and plot the event time series (event size vs. time). Check for punctuated dynamics. Develop and plot the event distribution (number of events vs. event size) using both normal coordinates and log-log coordinates. Check for power-law/fractal behavior. Take

the discrete Fourier transform (DFT) of the network propagation event time series to obtain its frequency spectrum. Check for "*1/f noise*" behavior and temporal fractal behavior.

Calculate network propagation path lengths. Plot the path length time series (path length vs. time). Plot path length distributions. Does propagation path length exhibit punctuated dynamics and power-law/fractal behavior?

Use path length data to calculate network indirect effects and direct effects. Plot the results. Check for punctuated behavior and dominance of indirect effects.

Analyze network connectivity. Develop and plot network node degree distributions. Do they exhibit degree scale invariance (fractal behavior)? Calculate network node mean degree and network connection density for each time step. At any given time step, does the network achieve critical connectivity and percolate? At what value of node mean degree? [The theoretical value for a directed random network is node mean degree = 2.] What is the size of the resulting giant component, i.e., the proportion of network nodes linked together?

5.1.4 Intermezzo

From the above model requirements, it should be clear that we endeavor to rigorously define, develop, and analyze a realistic dynamic model of complex ecological networks. We are not, however, attempting to describe detailed behavior of a particular real-world ecological network. (Recall, for example, that our relatively simple input, stock, and flow rules and values are not intended to match any specific ecosystem network.) We are developing a new approach here that differs in some respects from traditional ecological network compartment modeling and subsequent network environ analysis.

My intention is to provide a new perspective on the behavioral dynamics of ecological networks. The focus of my modeling effort is to determine common behaviors and general characteristics of the dynamics across a spectrum of ecological networks.

The compartment modeling/network environ analysis perspective and my behavioral dynamics modeling/analysis perspective are complementary: each can contribute to the understanding of the other. The first perspective employs steady-state/linear analysis that effectively models the mean-value behavior of an ecological network. The second perspective does not utilize steady-state/linear assumptions but rather allows the pulsing, punctuated dynamics often observed in ecological networks to develop. The results from each perspective should help in interpreting the results from the other.

[As follow-on work, I will consider investigating the use of EcoNet[99] (post stock and flow) to do network environ analysis on my model – at least the path length and indirect effects portion – and compare that with my results. Note, however, that individual-based models and aggregate models may or may not yield similar results. There is a NetLogo model (the Wolf Sheep Predation model) that illustrates the case of dissimilar results.[100]]

5.2 Model Programming Design and Development

The MATLAB R2009a programming environment[101] is being used to develop the ecological network propagation dynamics model.

---

[99] Kazanci and Tollner, *EcoNet, A New Software for Ecological Model Simulation and Network Analysis*, Ecological Modeling, 2007.

[100] Wilensky, *NetLogo Wolf Sheep Predation Model*, Center for Connected Learning and Computer-Based Modeling, Northwestern University, 2005. http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation(docked)

[101] The MathWorks, Inc., February 12, 2009.

A high-level summary of the programming design and development of our dynamics model is provided in this section. Details are provided in Appendices C through F. The dynamics model software has four major subsets:

1. Model network structure, parameters, and relationships
2. Propagation process
3. Analysis activities
    - Network input, output, stock, and flow analysis
    - Network propagation event analysis
    - Path length analysis
    - Indirect effects analysis
    - Network connectivity analysis
4. Graphics generation
    - Network propagation diagrams
    - Comprehensive set of other analysis graphics

Within each subset, we establish and initialize the associated program variables. Appendix C contains a *glossary* of these variables. The glossary includes a set of naming conventions for the variables and a definition for each of the more than one hundred variables used in the model. Appendix D contains the software *master m-file* – which implements major subsets 1 and 2 (above). Appendix E contains the analysis m-file that implements the *analysis activities* major subset. Appendix F contains the graphics m-file that implements the *graphics generation* major subset. The development of each of the dynamics model software m-files proceeds in manageable groupings called MATLAB *program cells*. Furthermore, each of the m-files is heavily commented to describe and document the software.

High-level summaries of the four major subsets of the dynamics model software are provided in the next four subsections, respectively.

5.2.1 Model Network Structure, Parameters, and Relationships

The *model network structure, parameters, and relationships* subset describes the model network and sets the stage for propagation processing. In this portion of the software we specify the underlying ecological network compartment model (i.e., specify its adjacency matrix, input vector, and output vector); build the model network node grid and its relationship to the compartments; identify the network input nodes and output nodes; define variables for the values of model network inputs, outputs, and stocks as they change with time; and create the needed node adjacency matrices.

We are modeling the dynamics of propagation in complex ecological networks. This, of course, requires that we define a network node adjacency matrix that is derived from the underlying compartment adjacency matrix. The node adjacency matrix is a "candidate" adjacency matrix that represents candidate propagation links. The compartment to node transformation approach that we use preserves network density (connectance) from the compartment adjacency matrix to the node candidate adjacency matrix. Per our model requirements, however, we need more "adjacency" information than that provided by the basic node adjacency matrix. We need the following:

1. For each from-node, must know the candidate connect-to nodes (network node adjacency matrix).
2. For each of the candidate connect-to nodes, must know the adjacency type so that we can implement our neighborhood concept.

3. For each candidate connect-to node, must know the home compartment number so that we can implement our compartment preference concept.
4. For each of the candidate connect-to nodes in each of the candidate connect-to compartments, must know the node attachment preference strength so that we can implement our node preferential attachment concept.

Item 1 is the basic network node adjacency matrix. Items 2 through 4 represent persistent data that could be stored, for convenience, in adjacency-like matrices – but with differing element definitions. We have, therefore, defined four persistent adjacency/adjacency-like matrices:

Matrix 1 (named *AA*) – basic node adjacency

Matrix 2 (named *AAT*) – connect-to node adjacency type

Matrix 3 (named *ACN*) – connect-to node compartment number

Matrix 4 (named *AAP*) – connect-to node attachment preference strength

Since the basic node adjacency matrix *AA* is a *candidate* adjacency matrix, we have also defined a set of network node *operational* adjacency (*AAO*) matrices and arrays that represent the actual propagation links formed as the network operates and develops over time. Furthermore, since we need to keep track of the amount of flow between nodes and compartments over time, we have defined flow-value arrays for that purpose.

In this portion of the software, we also specify the node stock and propagation rules and the number of model simulation time steps.

5.2.2 Propagation Process

The *propagation process* subset is the core of our software model of propagation dynamics in ecological networks. The process proceeds by model simulation time step, by stages within a time step, and by node propagation events within a stage. Figure 5.3 provides a

simple depiction of the propagation process. In the figure, the input node is green, propagating nodes are red, and non-propagating nodes are white. Note that this "tree" depiction allows a given node to appear in more than one stage and, therefore, accommodates cycling.

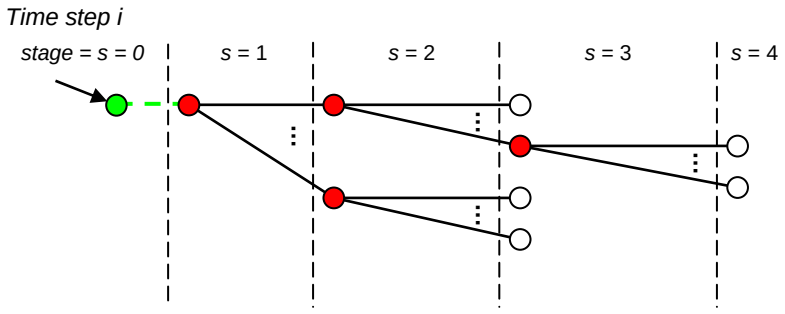


Figure 5.3 Depiction of the Propagation Process

At each model simulation *time step*, a unit input is applied to a randomly selected input node. That input may or may not cause the node to propagate (per the node stock and propagation rules). If the node propagates, we enter *stage* 1 of the time step. We have a *node propagation event* and a unit of stock flows to each of several connect-to nodes.[102] Each unit flow represents a *node propagation instance*.[103] If the flow causes one or more of the connect-to nodes to propagate, we enter stage 2 – and so on. When we reach a stage where there are no more propagating nodes, the simulation time step concludes. The propagation process, therefore, has three primary nested loops:

1. Time step loop
2. Stage loop
3. Node propagation event loop

[102] A node propagation *event* involves one "from" node and multiple "to" nodes.

[103] A node propagation *instance* is a subset of a propagation event and involves one "from" node and one "to" node.

Throughout the total propagation process there are also flow control logical constructs as required.

Time step loop:

A model simulation run consists of NumTS time steps. The time step loop, therefore, has NumTS iterations. Each model simulation time step can consist of multiple stages. At Stage 0, the input stage, a unit input is applied to a randomly selected input node. Every time step has a Stage 0 input. Propagation stages begin with Stage 1.

Stage loop:

Stage 1 is the first potential propagation stage. Stage 1 can have zero or one node propagation events – zero if the input node does not propagate and one if it does. If the node does propagate, we move to stage 2. Stage 2 (and any subsequent stages) can have zero, one, or more than one node propagation events. Whenever a given stage has one or more propagating nodes, we move to the next stage. [The presence of multiple propagation stages indicates that a "cascade" is in progress.] When we reach a stage where there are no more propagating nodes, the simulation time step concludes. The number of propagation stages for the time step is the last stage number minus 1.

Node propagation event loop:

Any given propagation stage can have 0, 1, or more node propagation events. The extent of a node propagation event is determined by the total flow from the propagating node. This is specified by variable npfq (node propagation flow quantity) – which is determined from the node stock and propagation rules. Each individual event corresponds to an iteration of the node propagation event loop.

Here's how a node propagation event proceeds. The from-node first attempts to propagate within its local neighborhood, then its extended local neighborhood, and finally its

global neighborhood [*neighborhood selection* processing]. Within each neighborhood, the from-node attempts to propagate to its preferred compartments (e.g., species groups) in rank order [*compartment selection* processing]. Within each neighborhood and compartment, the from-node propagates to its preferred to-nodes (e.g., species) according to their attachment preference probabilities [*node selection and propagation instance* processing] until the available flow quantity (per the node stock and propagation rules) is exhausted. The node propagation event loop, therefore, contains three additional interior nested loops:

a. Neighborhood selection loop

b. Compartment selection loop

c. Node selection and propagation instance loop

Here is how we proceed with *propagation process* development. We begin with a high-level view of the program flow logic and progress to the detailed MATLAB code. In this progression, we first define the high-level view of the flow logic and then define the basic propagation process variables (that show up in the high-level view). Next we define the four sub-processes (and their variables) that are named in the high-level flow, i.e., input node processing, propagating node processing, output node processing, and node propagation instance processing. We also provide expanded descriptions of other processing activities in the high-level flow as required. As we proceed, we define additional variables needed to store data for our subsequent analysis and graphics generation activities. We then add all of this detail to the high-level program flow logic to yield the detailed MATLAB program code for the propagation process.

See Appendix D for the *master* m-file which provides the detailed implementation of the model network and propagation process.

5.2.3 Analysis Activities

The *analysis activities* subset covers our analysis of the dynamics of propagation in complex ecological networks. The analysis categories are:

- Network operational propagation flow analysis
- Network propagation event analysis
- Path length analysis
- Indirect effects analysis
- Network connectivity analysis

These five categories are summarized in the following subsections.

5.2.3.1 Network Operational Propagation Flow Analysis

An important vehicle for depicting network operational propagation flow dynamics is the network node-and-link propagation flow diagram. The variables and data needed to produce these network diagrams are generated in the first two major subsets of the model software (as implemented in the model master m-file). The actual production of the diagrams is accomplished in the graphics generation major subset of the software (addressed in Section 5.2.4).

An effective means of depicting network input, output, stock, and flow value histories includes the use of 3-D grid bar charts displayed at selected time steps. We use an appropriate MATLAB 3-D/discrete-surface plotting capability. For this set of history diagrams, the necessary variables and data are created in the master m-file and the chart production is accomplished in the graphics generation m-file.

The full set of diagrams is expected to show continuing ecological network fluctuations without reaching a steady state.

5.2.3.2 Network Propagation Event Analysis

For the entire simulation run (after all time steps), we perform over-time network propagation event analysis and then plot and display the results. We calculate network propagation event size per-time-step (NetPESize_TS) in the master m-file. The propagation event size at a time step is equal to the total number of node instances involved in propagation at that time step, i.e., the number of node propagation instances (variable *nnpi*) plus one (the input node). Since any given node can participate in a propagation event multiple times (due to cycling), propagation event size can be larger than the number of physical nodes involved in the event and even larger than the total number of nodes in the network.

We develop a network propagation event time series (event size vs. time) and plot/display the results. We expect to observe punctuated dynamics. We then develop a network propagation event distribution (number of events vs. size of events) as follows:

- Sort the elements of NetPESize_TS from smallest to largest.
- Define a size interval (≥ 1), partition the size domain into intervals, and count the number of events in each interval.
- Generate a distribution with the ordered event size intervals as abscissa and the number of events in each of those size intervals as ordinate.
- Plot and display the event distribution in both normal coordinates and log-log coordinates.

We test for power-law/scale-invariant/fractal behavior of the network propagation event distribution.

We also perform discrete Fourier transform (DFT) analysis of the event time series to obtain its frequency spectrum and then test the spectrum for "*1/f noise*" behavior and temporal scale-invariant/fractal behavior.

All of the network propagation event analysis plots are produced using the graphics generation m-file.

5.2.3.3 Path Length Analysis

We perform network path length analysis for the entire simulation run (after all time steps). We develop a path length time series and path length distribution for that purpose. We also perform path length analysis for each time step – and develop individual-time-step and cumulative path length distributions for that purpose. For the entire simulation run and for selected time steps, we plot and display the results. The source data needed to do this has been generated in the master m-file. The plots are produced using the graphics generation m-file.

5.2.3.3.1 Path Length Calculations

Figure 5.4 depicts propagation flow and illustrates the path length calculations.

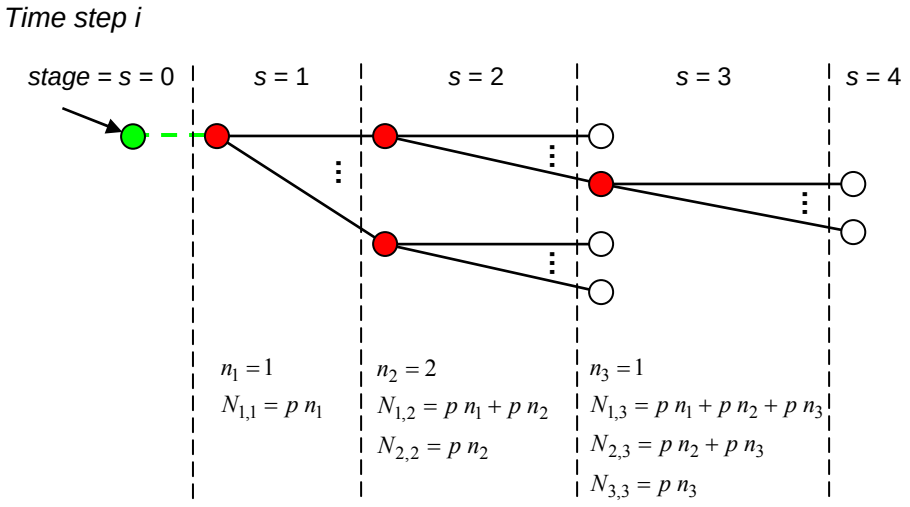


Figure 5.4 Path Length Calculations

The equations in the lower portion of the figure calculate $N_{\ell,s}$ – the number of paths of length ℓ at stage s. (The other equation parameters are defined shortly.) Using this approach, we can calculate the path length numbers for a time step. We can then calculate the simulation-run path length time series and distribution as well as per-time-step path length distributions. The development follows.

$s = simulation\ time - step\ stage$
$s = 0 \quad input\ stage$
$s \geq 1 \quad propagation\ stages$
$n_j = number\ of\ propagating\ nodes\ at\ stage\ s = j \geq 1$
$\ell = path\ length$
$N_{\ell,s} = number\ of\ paths\ of\ length\ \ell\ at\ stage\ s$
$p = number\ of\ unit\ flows\ per\ propagating\ node$

$$N_{1,s} = \sum_{j=1}^{s} p\, n_j \qquad N_{2,s} = \sum_{j=2}^{s} p\, n_j \qquad N_{3,s} = \sum_{j=3}^{s} p\, n_j$$

$$N_{\ell,s} = p \sum_{j=\ell}^{s} n_j \quad for \quad \ell \leq s \qquad N_{\ell,s} = 0 \quad for \quad \ell > s$$

$At\ each\ completed\ simulation\ time\ step$:
$s = \max(s) = S = total\ number\ of\ stages\ for\ completed\ simulation\ time\ step$
$N_\ell = N_{\ell,S} = number\ of\ paths\ of\ length\ \ell\ for\ completed\ time\ step$

Note that the number of internal flows from a propagating node is reduced by 1 for output nodes. We need to distinguish between output propagating nodes and non-output propagating nodes and their respective flow quantities.

$nnpe_j = total\ number\ of\ \ propagating\ nodes\ at\ stage\ s \geq 1$

$nopn_j = number\ of\ \ output\ \ propagating\ nodes\ at\ stage\ s \geq 1$

$nnopn_j = number\ of\ \ non-output\ \ propagating\ nodes\ at\ stage\ s \geq 1$

$nnopn_j = nnpe_j - nopn_j$

$npfq = number\ of\ \ unit\ \ flows\ \ per\ non-output\ \ propagating\ node$

$npfq - 1 = number\ of\ \ unit\ \ flows\ \ per\ output\ \ propagating\ node$

$$N_\ell = (npfq - 1)\sum_{j=\ell}^{S} nopn_j + npfq \sum_{j=\ell}^{S} nnopn_j$$

$$N_\ell = (npfq - 1)\sum_{j=\ell}^{S} nopn_j + npfq \sum_{j=\ell}^{S} (nnpe_j - nopn_j)$$

$$N_\ell = (npfq - 1)\sum_{j=\ell}^{S} nopn_j + npfq \sum_{j=\ell}^{S} nnpe_j - npfq \sum_{j=\ell}^{S} nopn_j$$

$$N_\ell = npfq \sum_{j=\ell}^{S} nnpe_j - \sum_{j=\ell}^{S} nopn_j$$

$$N_\ell = npfq \sum_{j=\ell}^{S} nnpe_j - \sum_{j=\ell}^{S} nopn_j \quad for \quad 1 \leq \ell \leq S$$

$$N_\ell = 0 \quad for \quad \ell > S$$

These equations define the numbers of paths for the time step.

We use these equations to generate the path length time series and distributions. We plot and display the distributions in both normal coordinates and log-log coordinates. We test the time series for punctuated dynamics and test the distributions for power-law/fractal behavior.

5.2.3.3.2 Integration over Space and Time

Much network analysis is performed on a model of a network that is assumed to be at "steady state." The network does not change with time – at least not over the analysis observation interval. The network nodes and their connections are persistent. Analysis, therefore, is somewhat simplified. Analysis becomes an exercise in integration over the network space. To perform path length analysis, one begins by identifying all the paths of various lengths

across the steady-state network and counting them. (To perform node degree analysis, one counts the number of connections from/to each node in the network.)

One steady-state path length analysis technique is the "box covering" method (a version is described in Chapter 4). This approach integrates over the network space. For a given network, one generates boxes of size ℓ_B. Each box contains a set of nodes for which the maximum realized path length between any two nodes in the box is ℓ_B. The entire network is then covered with such boxes. We count the number of boxes of each size. The number of boxes of size ℓ_B required to cover the network is denoted by N_B. A plot of N_B vs. ℓ_B provides the path length distribution for the network.

My analysis work, on the other hand, does not make the "steady state" assumption. An important objective of my research is to model and analyze the changing dynamics of ecological networks. In my view, network structure and dynamics are not persistent. They change with time – sometimes dramatically so. My analysis work, therefore, must involve not only integration over network space, but also integration over time.

How do I accomplish integration over space and time? I start by performing spatial analysis at each individual point in time (the simulation time steps). My initial idea for integrating the time steps was to calculate over-time cumulative path length data and analyzing that. While the cumulative data has value, that approach does not work when attempting to develop an integrated path length distribution. The resulting distribution is essentially a distribution of the average number of path lengths (vs. path length) over time. A distribution of averages is not particularly meaningful or useful in a punctuated dynamics environment.

A better idea for achieving integration over space and time is to devise a temporal analog to the spatial "box covering" approach mentioned above. Instead of generating a collection of

boxes that covers a steady-state network and then counting the number of boxes over space, we can generate one maximum-length ℓ_B box at each point in time (time step) and then count the number of boxes of each size over time. The result is a path length distribution that is integrated over space and time. This distribution, and corresponding time series, effectively captures the path length punctuated dynamics. In our model software, we develop this distribution and time series for the entire simulation run.

5.2.3.4 Indirect Effects Analysis

We perform a comprehensive set of indirect effects analyses. We define and calculate individual-time-step and over-time-cumulative indirect effects indicators – and generate simulation-run time series for each of these indicators. We develop an indirect-path-quantity time series and distribution as well as a direct-path-quantity time series and distribution for the simulation run. The time series provide the number of indirect (or direct) paths at each time step vs. time. We also develop a cumulative indirect-path-quantity time series and a cumulative direct-path-quantity time series for the simulation run. Each provides the cumulative number of paths at each time step vs. time. The source data needed for all this has been generated in the (see Section 5.2.3.3) *path length analysis* program cells of the analysis activities m-file. The graphics are produced using the graphics generation m-file.

We mathematically derive the conditions for indirect effects dominance and indirect effects prominence – and use the derivations to test our dynamics model simulation outcomes.

Further descriptions of the indirect effects analyses are provided in the following two subsections.

5.2.3.4.1 Indirect Effects Calculations

For each time step, we calculate individual-time-step and over-time-cumulative indirect effects (and direct effects) parameters. We calculate the Direct Effects Ratio (DER), Indirect Effects Ratio (IER), and Indirect Effects Index (IEI) using the following expressions:

$At\ each\ completed\ simulation\ time\ step$:
$\ell = path\ length$
$S = total\ number\ of\ stages\ for\ completed\ time\ step$
$N_\ell = number\ of\ paths\ of\ length\ \ell\ for\ completed\ time\ step$

$DER = Direct\ Effects\ Ratio$
$DER = number\ of\ paths\ of\ length\ 1\ divided\ by\ total\ number\ of\ paths$

$$DER = \frac{N_1}{\sum_{\ell=1}^{S} N_\ell}$$

$IER = Indirect\ Effects\ Ratio$
$IER = number\ of\ paths\ of\ length > 1\ divided\ by\ total\ number\ of\ paths$

$$IER = \frac{\sum_{\ell=2}^{S} N_\ell}{\sum_{\ell=1}^{S} N_\ell}$$

$IEI = Indirect\ Effects\ Index$
$IEI = number\ of\ paths\ of\ length > 1\ divided\ by\ number\ of\ paths\ of\ length\ 1$

$$IEI = \frac{\sum_{\ell=2}^{S} N_\ell}{N_1}$$

Cumulative values over simulation time steps :

$$DER = \frac{\sum_{\substack{Time \\ steps}} N_1}{\sum_{\substack{Time \\ steps}} \sum_{\ell=1}^{S} N_\ell}$$

$$IER = \frac{\sum_{\substack{Time \\ steps}} \sum_{\ell=2}^{S} N_\ell}{\sum_{\substack{Time \\ steps}} \sum_{\ell=1}^{S} N_\ell}$$

$$IEI = \frac{\sum_{\substack{Time \\ steps}} \sum_{\ell=2}^{S} N_\ell}{\sum_{\substack{Time \\ steps}} N_1}$$

We plot and display the direct effect and indirect effect indicators – both the individual time-step values vs. time and the over-time-cumulative values vs. time – using the graphics generation m-file. We observe the behavior and draw conclusions.

We also develop (and plot) the simulation run indirect-path-quantity time series (number of indirect paths at each time step vs. time) and distribution as well as the direct-path-quantity time series (number of direct paths at each time step vs. time) and distribution. We observe the behavioral dynamics. Are indirect effects punctuated and fractal?

5.2.3.4.2 Indirect Effects: Dominance or Prominence?

Ecological systems take the form of networks. Network connections (links) between network nodes can be characterized as random variables. The connections can be either *direct* or *indirect*. If a node pair is connected, the connection path length is 1 for a direct connection and greater than 1 for an indirect connection.

Let's represent the connection path length by a discrete random variable X with probability distribution function $P\{x_i\}$:

$$P\{X = x_i\} \qquad x_i \in \{1, 2, 3, \cdots, N-1, \cdots\}$$

where

N = *number of nodes in the network*

and

$N-1$ = *longest path length without cycling*

With cycling, path length can be anywhere in the set $\{1, 2, 3, \cdots, N-1, \cdots\}$.

It is my expectation [see, e.g., Section 4.4 and Section 5.2.3.3 of this document] and there is considerable evidence in the literature [see Song et al[104, 105], Vishwanathan et al[106], Odum and Barrett[107], Schiff [108], Yackinous[109]] that, for real-world networks, the probability distribution function $P\{x_i\}$ is a power-law/fractal distribution given by:

$$P\{X = x_i\} = C\, x_i^{-\gamma} \qquad x_i \in \{1, 2, 3, \cdots, N-1, \cdots\}$$

where

C *is a constant*

and

γ = *the network path length scaling exponent*

Using data from complex biological networks, Song et al and Vishwanathan et al have calculated values of γ that are in the neighborhood of 2.

---

[104] Song, Havlin, and Makse, *Self-Similarity of Complex Networks*, Nature, January 2005.

[105] Song, Gallos, Havlin, and Makse, *How To Calculate the Fractal Dimension of a Complex Network*, Journal of Statistical Mechanics, March 2007.

[106] Vishwanathan et al, *Optimizing the Success of Random Searches*, Nature, October 1999.

[107] Eugene Odum and Gary Barrett, *Fundamentals of Ecology* Fifth Edition, Thomson Brooks/Cole, 2005.

[108] Joel Schiff, *Cellular Automata*, Wiley-Interscience, 2008.

[109] Bill Yackinous, *Reductionism and Information Loss in Ecological Investigation*, March 25, 2009.

Since $P\{x_i\}$ is a probability function, its values must sum to 1:

$$\sum_{x_i} P(X = x_i) = \sum_{x_i} C\, x_i^{-\gamma} = 1$$

$$C \sum_{x_i} x_i^{-\gamma} = 1$$

The sum above has the same form as the so-called *p-series*:

For any positive real number p:

$$P\text{-}series = \sum_{n=1}^{\infty} n^{-p}$$

For $p > 1$:

$$\sum_{n=1}^{\infty} n^{-p} = \zeta(p)$$

The *p-series* always converges if $p > 1$ (in which case it is called the *over-harmonic series*). When $p > 1$, the sum of this series turns out to be equal to the *Riemann zeta function* evaluated at p: $\zeta(p)$.

Note that, while the *p-series* is infinite, our path length probability series is finite. However, assuming just 25 terms, the value of the path length series is close to the convergent value of the *p-series*. For example, with an exponent of -1.7, the value of the path length series = 1.9063 and $\zeta(p) = 2$. With an exponent of -1.63, the value of the path length series is very close to 2 (i.e., 2.0021). We will use "approximately equal" symbols when appropriate as we proceed.

Continuing with our development, we can use the p-series and the Riemann zeta function to calculate constant C:

$$C \sum_{x_i} x_i^{-\gamma} = 1$$

$$C\, \zeta(\gamma) \cong 1$$

$$C \cong \frac{1}{\zeta(\gamma)}$$

We can rewrite our path length probability sum relationship in the following way:

$$\sum_{x_i} P(X = x_i) = P(X = 1) + \sum_{x_i > 1} P(X = x_i) = 1$$

$$P(X = 1) = C\,(1)^{-\gamma} = C \cong \frac{1}{\zeta(\gamma)}$$

$$P(X = 1) \cong \frac{1}{\zeta(\gamma)}$$

$$\sum_{x_i > 1} P(X = x_i) = 1 - P(X = 1) \cong 1 - \frac{1}{\zeta(\gamma)}$$

$$\sum_{x_i > 1} P(X = x_i) \cong 1 - \frac{1}{\zeta(\gamma)}$$

Relating this to direct and indirect connections, we have:

$$P(X = 1) \cong \frac{1}{\zeta(\gamma)}$$

$$\sum_{x_i > 1} P(X = x_i) \cong 1 - \frac{1}{\zeta(\gamma)}$$

$$P(X = 1) = \Pr\{direct\ connection\}$$

$$\sum_{x_i > 1} P(X = x_i) = \Pr\{indirect\ connection\}$$

When $\zeta(\gamma) \cong 2$:

$$\Pr\{direct\ connection\} = \Pr\{indirect\ connection\}$$

When $\zeta(\gamma) > 2$:

$$\Pr\{indirect\ connection\} > \Pr\{direct\ connection\}$$

Indirect paths dominate

When $\zeta(\gamma) < 2$:

$$\Pr\{direct\ connection\} > \Pr\{indirect\ connection\}$$

Direct paths dominate

Figure 5.5 is a graph of the Riemann zeta function for real arguments >1. [The figure is an adaptation of a graph from *Wikipedia.]*

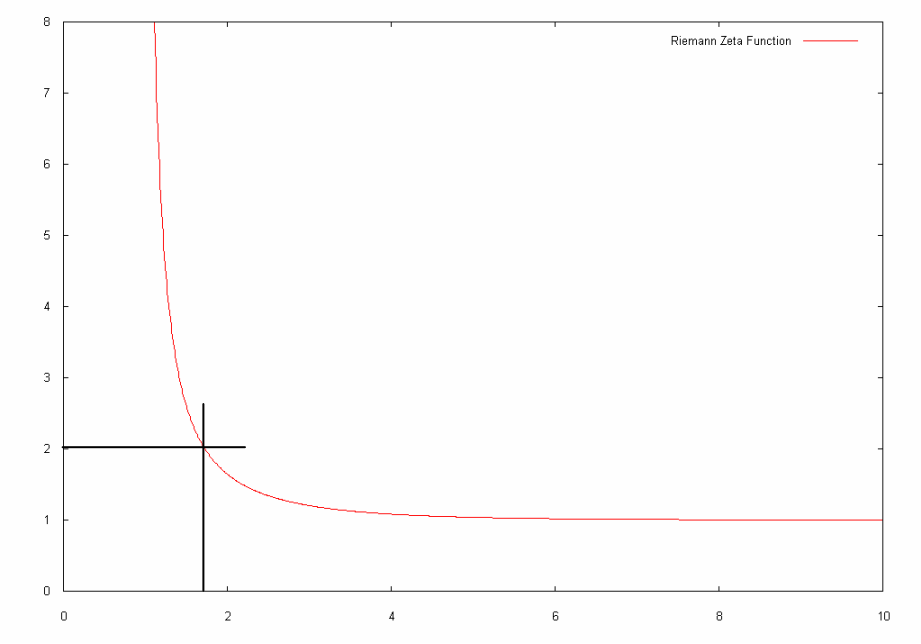


Figure 5.5 Riemann Zeta Function

The ordinate is $\zeta(\gamma)$ and the abscissa is γ – values of the path length distribution scaling exponent. Typical path length scaling exponents are in the range:

$$(1+) < \gamma < (2+)$$

We observe that:

When $1 < \gamma < 1.7$
$\zeta(\gamma) > 2$
Indirect effects dominate

When $\gamma > 1.7$
$\zeta(\gamma) < 2$
Indirect effects do not dominate
Indirect effects are likely still prominent

5.2.3.5 Network Connectivity Analysis

We perform both over-time and per-time-step network connectivity analysis. The source data needed to do this is generated in the master m-file and stored as arrays AAO_t_TS and AAO_tc_TS:

AAO_t_TS = operational node adjacency multidimensional array that provides an operational adjacency matrix for each individual time step per-time-step (100x100xNumTS)

AAO_tc_TS = operational node adjacency multidimensional array that provides a cumulative operational adjacency matrix per-time-step (100x100x NumTS)

The network connectivity analysis plots are produced using the graphics generation m-file.

5.2.3.5.1 Node Degree Analysis

We perform comprehensive node degree analysis. We develop individual-time-step node degree vector arrays and node degree grid arrays. Note that along with network propagation events, path length, and indirect effects – node degree exhibits punctuated dynamics. That can be clearly seen by observing the node degree grids (node degree overlays on the network grid) at different points in time. We need to develop node degree time series and distributions that capture those dynamics in an integrated fashion over space and time.

Integration over Space and Time

As discussed in Section 5.2.3.3.2, much network analysis is performed on a model of a network that is assumed to be at "steady state." Such analysis is spatial in nature. To perform node degree analysis, one counts the number of connections from/to each node in the network space. My analysis work does not make the steady-state assumption – and it has both spatial and temporal dimensions. In my analysis, I need to integrate over both space and time.

With respect to node degree distribution analysis, I start by performing spatial analysis at each individual simulation time step. Similar to the path length distribution case, my initial idea

for integrating the time steps was to calculate over-time cumulative node degree data and generate the associated distributions. That approach did not work in the path length distribution case and it does not work for node degree distributions either. In the node degree case, the reasons are a bit different. As node connections accumulate over time, node degree values increase. When compared with node degree distributions calculated at individual high-propagation time steps, the cumulative node degree distribution shifts to the right and changes shape quite drastically. The lower-degree range falls and the mid- to upper-degree ranges rise. The resulting cumulative distributions do not seem to have any useful physical meaning.

To accomplish integration over space and time, we devise a spatial/temporal approach very similar to the path length distribution approach. For each degree type (out, in, and combined), we develop a time series of maximum node degree achieved at each individual time step and then generate the associated distribution. The results are node degree distributions that are integrated over space and time. The distributions, and corresponding time series, effectively capture the node degree punctuated dynamics.

Another measure that may capture the node degree dynamics is the node mean degree achieved at each time step. We calculate the per-time-step values of node mean degree (and the related network connection density).[110] We then develop a simulation-run node-mean-degree time series and distribution. We will compare the results of the maximum degree and mean degree approaches.

Per-Time-Step Node Degree Distributions

We also generate per-time-step node degree distributions. We develop individual-time-step node degree distributions as well as cumulative node degree distributions (cumulative from

[110] Node mean degree is defined as the number of network edges divided by the number of nodes. Network connection density is defined as node mean degree divided by (number of nodes – 1).

time step one to the current time step) and examine them for useful information. Each per-time-step distribution provides the number of nodes with degree x vs. x. For all of these node degree distributions, the analysis starting point is the generation of node degree arrays which provide degree of each node at each time step. The per-time-step node degree distributions are developed as follows:

- Sort the node degree values from smallest to largest.
- Define a degree size interval (≥ 1), partition the degree size domain into those intervals, and count the number of nodes in each interval.
- Generate a distribution with the ordered node degree intervals as abscissa and the number of nodes in each of those intervals as ordinate.
- Plot and display each distribution in both normal coordinates and log-log coordinates.

We test all of the resulting distributions for degree power-law/scale-invariant/fractal behavior.

5.2.3.5.2 Other Connectivity Considerations

We will attempt to perform network critical connectivity/percolation analysis. At any given time step, the network may achieve critical connectivity and percolate. If that occurs, what is the value of node mean degree? [The theoretical value for a directed random network is node mean degree = 2.] How many nodes are linked together and what is the fractional size of the resulting “giant component”? To answer these questions, in addition to our above calculation of node mean degree, we calculate the following two individual-time-step parameters. We calculate the number of nodes linked together at each time step [vector NumLN_t_TS]. For each individual propagation event, all involved nodes (all from-nodes and to-nodes) are linked together. We also calculate the fractional size of the resulting candidate "giant component" at

each individual time step [vector CompSize_t_TS]. The size of the candidate giant component equals the number of nodes linked together divided by the total number of nodes in the network.

Another important parameter is the network connection density – which we calculate at each time step. We can then develop and plot several pertinent time series (e.g., node mean degree, network density, and number of linked nodes), identify time steps of potential critical connectivity, and observe the values of our network connectivity parameters at those time steps. We may be able to draw conclusions regarding important network connectivity traits.

Please see Appendix E for the *analysis* m-file which provides the detailed implementation of all of the analysis activities.

5.2.4 Graphics Generation

The *graphics generation* subset produces the graphs and diagrams that result from our analysis of the dynamics of propagation in complex ecological networks. The required graphics include:

- Network node-and-link flow graphs
- Network flow value diagrams and input, output, and stock histories
- Network propagation event time series (event size vs. time) and distribution (number of events vs. size of events)
- Network propagation event time series frequency spectrum
- Path length time series and distributions (number of paths vs. path length)
- Indirect effects and direct effects indicators vs. time
- Indirect-path and direct-path time series (number of paths at each time step vs. time) and their distributions
- Network node degree time series and distributions

- Network critical connectivity graphics

Network flow graphics include node-and-link flow graphs, flow value diagrams, and input/output/stock history diagrams. The MATLAB *gplot* function is used to generate the network node-and-link flow graphs. An important advantage of gplot is that it preserves the spatial positioning of the network nodes. We illustrate node-and-link flow at individual time steps as well as cumulative node-and-link flow over time. For improved visual clarity, we color-code the nodes involved in propagation (green for input nodes, red for propagating from-nodes, and dark blue for non-propagating to-nodes). In addition to those graphs, cumulative flow value diagrams are generated to provide an "adjacency matrix" depiction of network flow. Network input, output, and stock histories are also plotted (and displayed on 3-D network grid bar charts).

The *network propagation event* graphics include a propagation event time series and distribution. The distribution is plotted in both normal and log-log coordinates. On the log-log plot, we fit a straight line to the data. The slope and y-intercept of the straight line are used to create a power-law curve – which is then overlaid on the normal-coordinates plot of the network propagation event distribution for comparison purposes. We also generate a network propagation event frequency spectrum and add an appropriate power-law overlay for comparison.

The *path length* graphics consist of time series and associated distributions. We produce a path length time series and integrated path length distribution using an approach analogous to the network “box covering” method. We also produce a time series and integrated distribution based on the mean path length at each individual time step. We generate distribution power-law overlays for both cases. In addition, per-time-step path length distributions (for individual time

steps as well as cumulative across time steps) are generated so that their behavior can be examined.

To illustrate ecological network *indirect effects* and direct effects, we produce a variety of time series and, in some cases, their corresponding distributions. These graphics reveal the dominance of indirect effects. We generate direct effects ratio, indirect effects ratio, and indirect effects index time series. We also plot the cumulative behavior of these indicators. We plot indirect path quantity and direct path quantity time series and their distributions – and overlay power-law curves on the distributions for comparison.

The *network connectivity* graphics consist of node degree three-dimensional stem plots, appropriate node degree time series and distributions, and other connectivity graphs. We plot node degree over the plane of the network grid at various simulation-run time steps. These three-dimensional graphics illustrate the fluctuating dynamics of node degree over time. We produce node degree time series and distribution plots that capture the dynamics. We add power-law overlays to the distributions. To test for network critical connectivity, we generate graphs that allow us to observe instances of possible criticality and the values of network connectivity parameters that occur at those instances.

Please see Appendix F for the *graphics* m-file that implements the generation of all of these graphics.

CHAPTER 6

RESULTS

Our results are provided and described in five categories in the next five subsections, respectively. The categories are:

- Operational Propagation Flow
- Network Propagation Events
- Path Length
- Indirect Effects
- Network Connectivity

6.1 Operational Propagation Flow Results

Our modeling and analysis focus is network propagation. To set the context for the results, we begin with the network propagation time series shown in Figure 6.1.

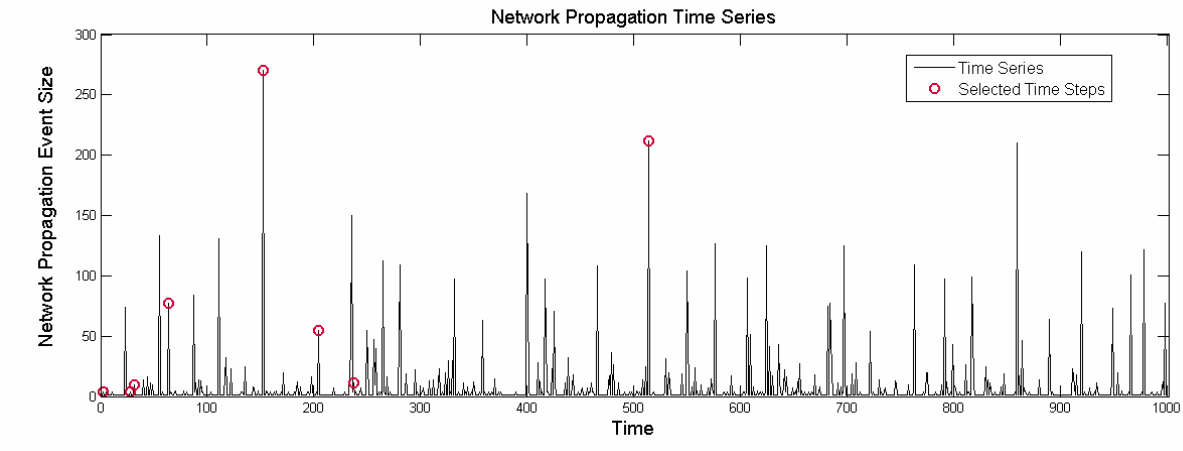


Figure 6.1 Network Propagation Time Series

We'll discuss details of this time series (and related statistical results) later. For now, just note the ever-changing propagation behavior – from very small to very large propagation events. We'll examine the propagation flow for eight specific events next. The red selection circles in Figure 6.1 highlight those events.

6.1.1 Network Node-and-Link Propagation Flow Results

Let's quickly review the propagation process. At each model simulation time step, a unit input is applied to a randomly selected input node. That input may or may not cause the node to propagate (per the node stock and propagation rules). If the node propagates, then that from-node propagates a unit of stock to each of several to-nodes. Propagation proceeds by neighborhood, compartment preference, and node preference – in that order. Figure 6.2 depicts a propagation from-node (in red) and its potential propagation to-nodes (in blue) for each of the three neighborhood levels.

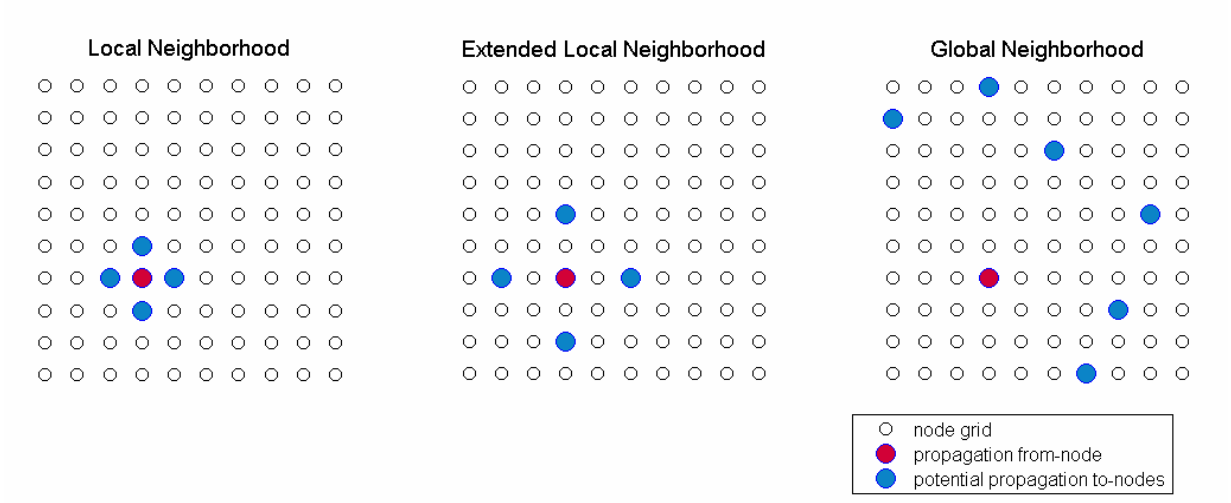


Figure 6.2 Explanation of Neighborhoods

Propagation is first attempted within the local neighborhood, then the extended local neighborhood, and finally the global neighborhood. Within each neighborhood, propagation is to preferred compartments (e.g., species groups) in rank order. Within each neighborhood and preferred compartment, propagation is to preferred nodes (e.g., species) according to their

attachment preference probabilities. Propagation proceeds until the from-node flow quantity is exhausted. For these results, we have set the node flow quantity parameter to four. Note that if the propagating node is also an output node, it propagates one unit of flow to the external environment and three units of flow internally. Otherwise, four units of flow are propagated internally.

Network node-and-link propagation flow diagrams follow. In each diagram, the background is the network spatial grid consisting of 100 nodes. The solid lines represent propagation-flow links. Here's the node color code. The green node is the propagating input node. The red nodes are subsequent propagating nodes. The dark blue nodes are propagation to-nodes that do not propagate further. Any of these nodes could also be an output node.

Figure 6.3 displays the diagrams for four small network propagation events. The title of each diagram gives the time step number and the event size (number of involved nodes).

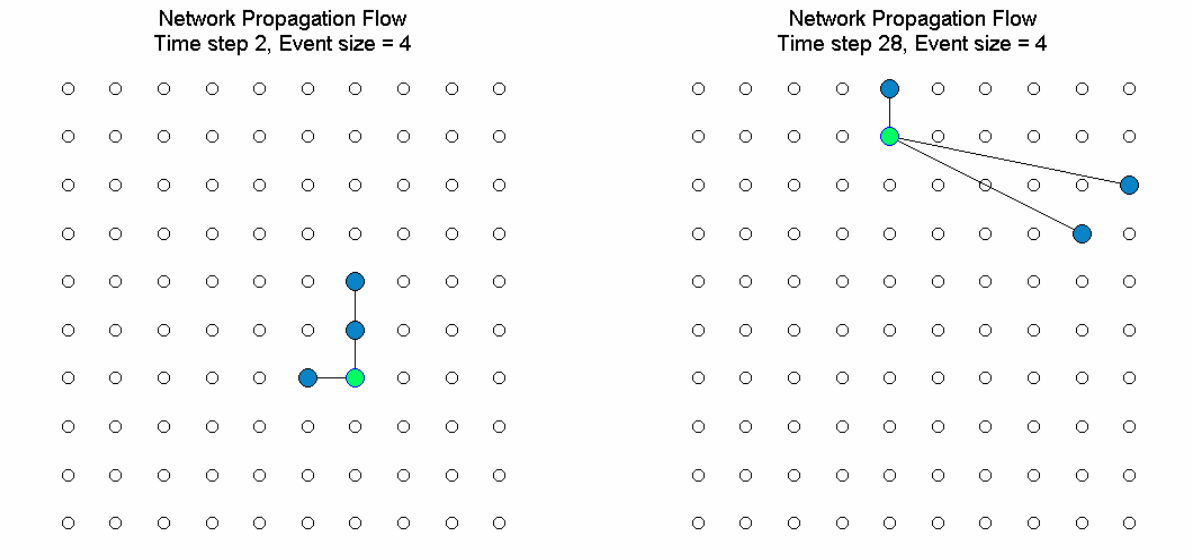


Figure 6.3a Network Propagation Flow (small propagation events)

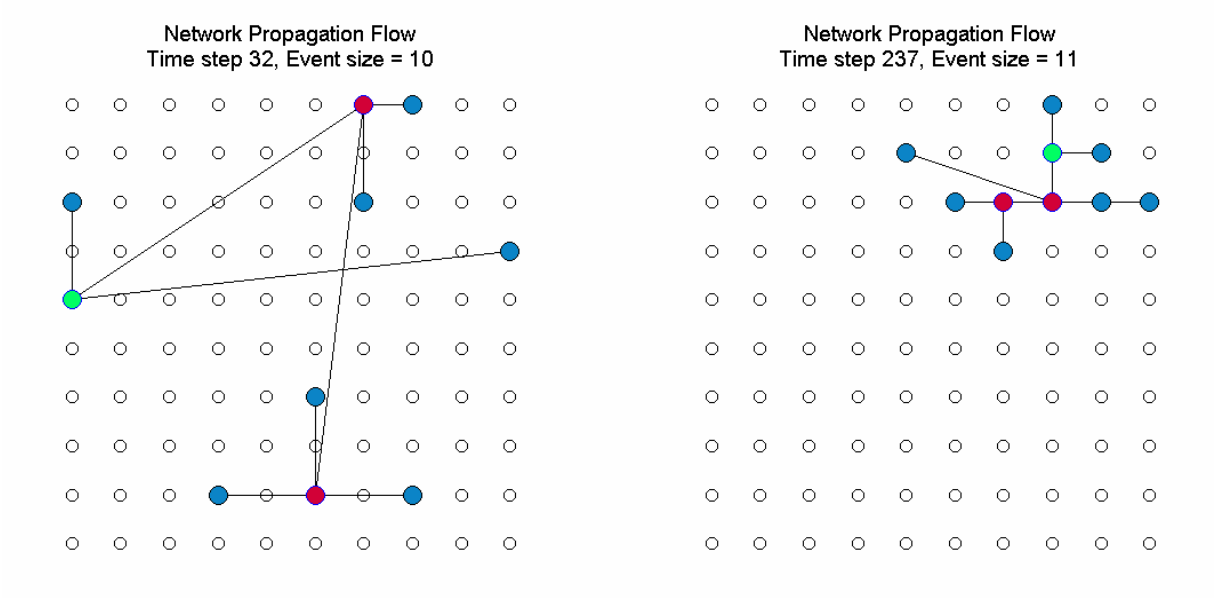


Figure 6.3b Network Propagation Flow (small propagation events)

At time step 2, the input node (which is also an output node) propagates one unit of stock to the external environment (not shown explicitly), one unit of stock to each of two nodes in the local neighborhood, and one unit of stock to a node in the extended local neighborhood. At time step 28, the input node (also an output node) propagates to one node in the local neighborhood and two nodes in the global neighborhood. At time step 32, we have a 3-stage network propagation event involving local, extended local, and global neighborhoods. The event size is only 10, but it has a relatively large network span. The time step 237 event is also a 3-stage event, but has more local processing. Count the colored nodes. The count does not equal the event size. The red propagating node on the right is involved more than once. It is first a propagation from-node and then a propagation to-node. This is our first glimpse of cycling in the model network.

Figure 6.4 displays diagrams for two mid-size network propagation events (at time steps 64 and 204). The event sizes are 77 and 55, respectively.

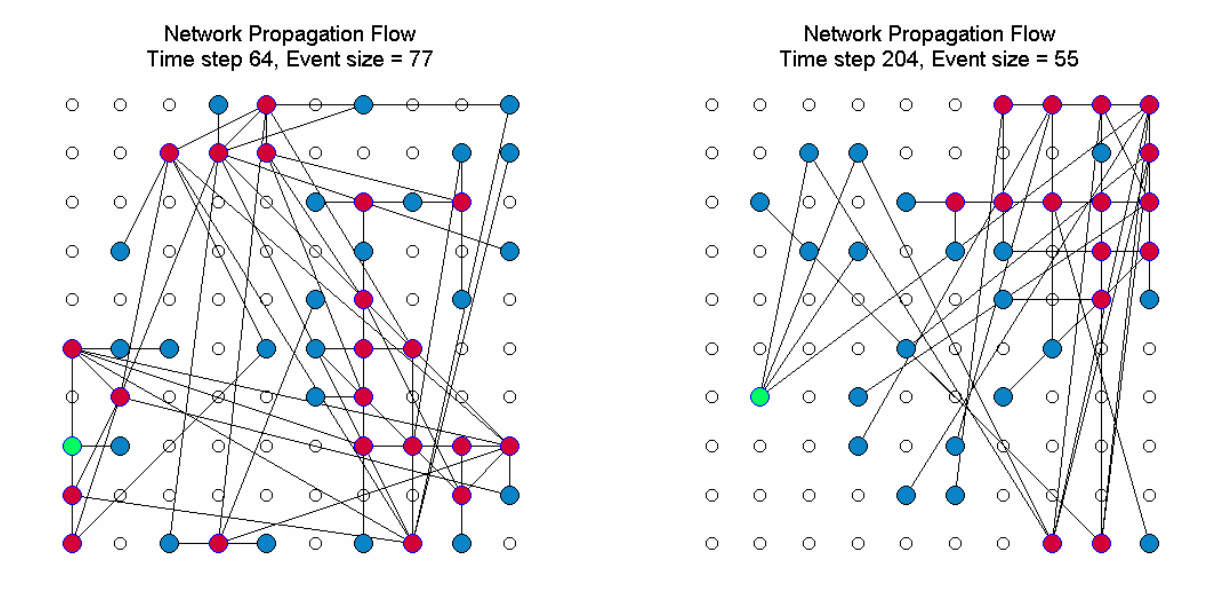


Figure 6.4 Network Propagation Flow (mid-size propagation events)

We see a mix of local, extended local, and global neighborhoods being utilized. We see relatively large network spans. Significant cycling is evident. An approximate "back of the envelope" expression for cycling index can be defined as follows:

$$cycling\ index = \frac{number\ of\ cycling\ propagation\ flows}{total\ number\ of\ propagation\ flows}$$

$$cycling\ index \approx \frac{(event\ size - node\ count)}{event\ size}$$

Using this expression, at time step 64, the cycling index is approximately 0.4. At time step 204, the cycling index is about 0.3.

Two of the largest individual-time-step network propagation events are illustrated in Figures 6.5 and 6.6. Compared to the mid-size events, we see greater network span and higher levels of local-to-global processing here. A large fraction of all network nodes are involved. There is lots of cycling going on. The node-and-link diagram of Figure 6.5 has a cycling index of about 0.7, and the diagram of Figure 6.6 has a cycling index around 0.6.

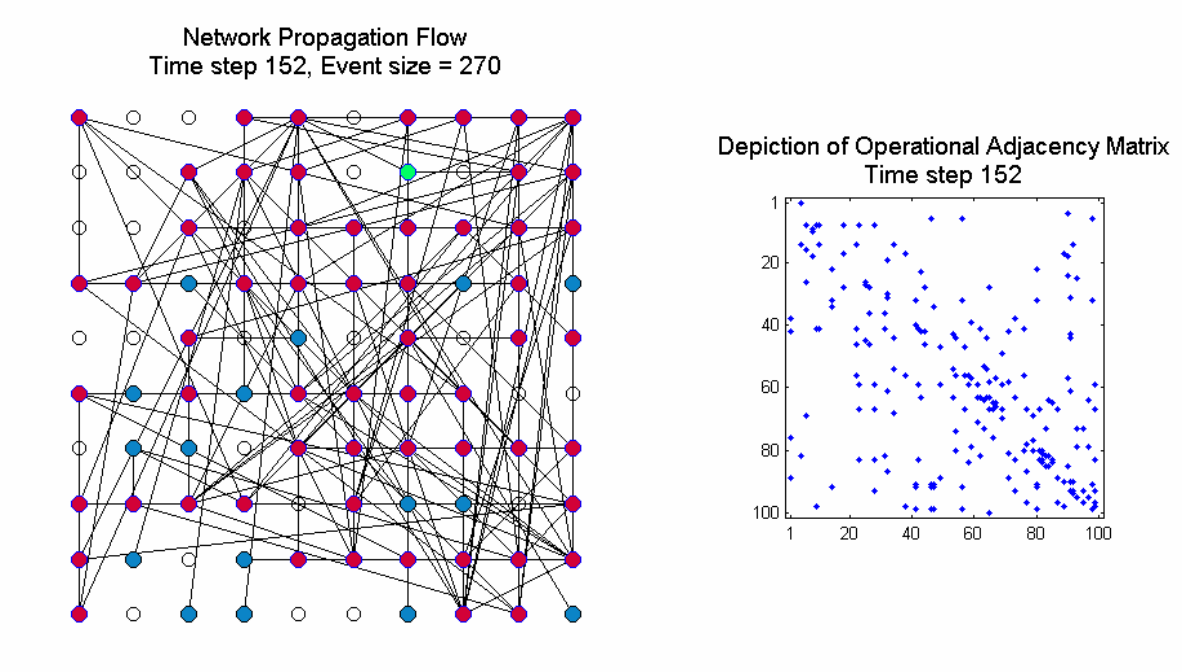


Figure 6.5 Network Propagation Flow (large event; time step 152)

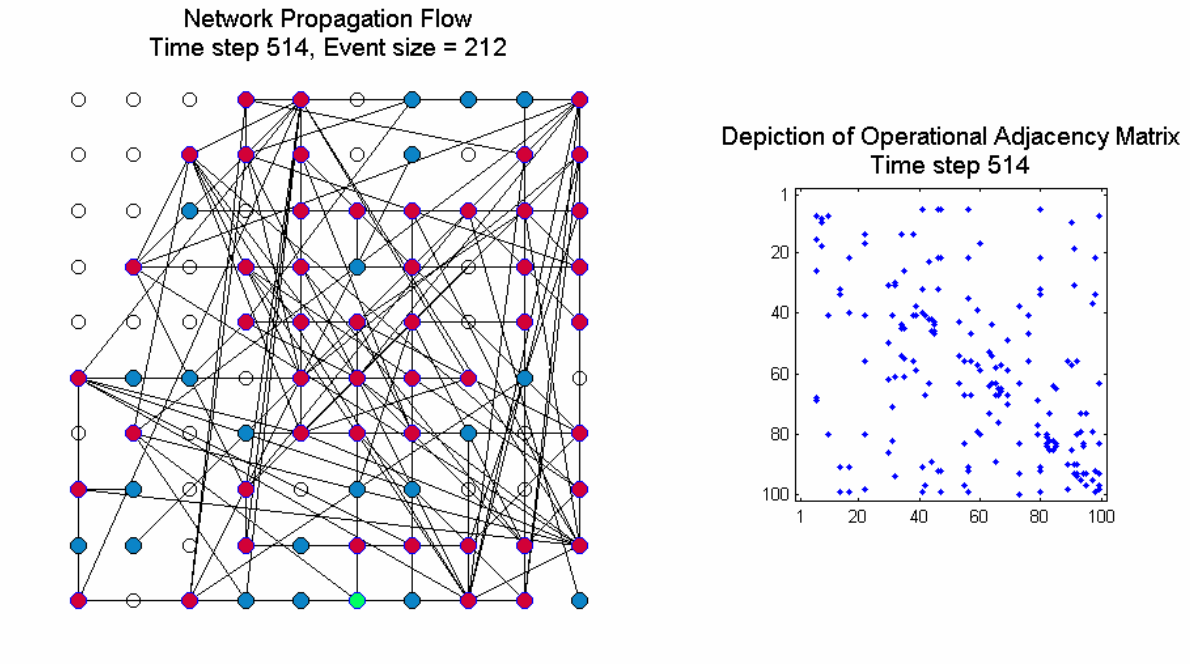


Figure 6.6 Network Propagation Flow (large event; time step 514)

On the right-hand-side of each of the figures, we have added a depiction of the network operational adjacency matrix for the time step. For the 100x100 matrices, the markers (blue dots) indicate ones and white space indicates zeros. The matrices have ones distributed

throughout with a particular concentration near the matrix major diagonal (more about this pattern shortly).

As we look back over the above figures (the time series and the diagrams for the eight representative time steps), several things become evident. Propagation behavior over time is punctuated. Most events, and perhaps all of the mid-size to large events, exhibit local-to-global propagation behavior. Cycling is prominent. The network continually changes with time – and sometimes dramatically so. The view of the network operational dynamics is clear: we are dealing with ever-changing, "flickering" ecological networks.

It is also instructive to look at the network cumulative propagation flow over time to see the dense web of node interactions that develops and the participation of (very nearly) the entire network in propagation.

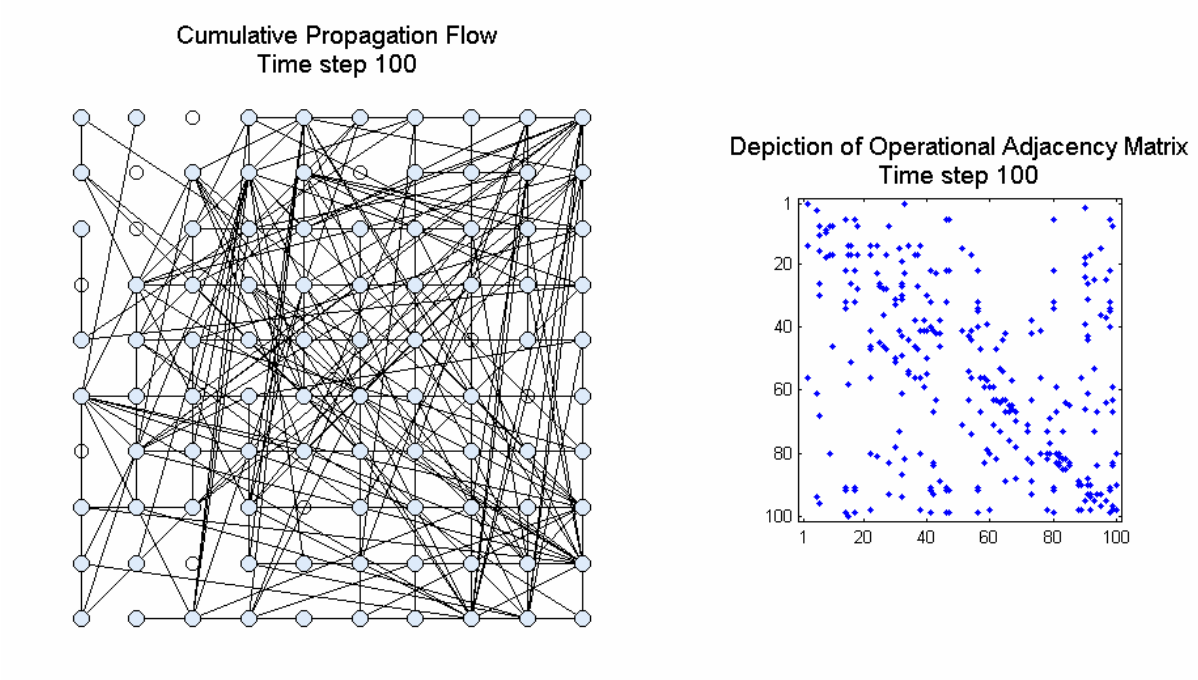


Figure 6.7 Cumulative Propagation Flow (time step 100)

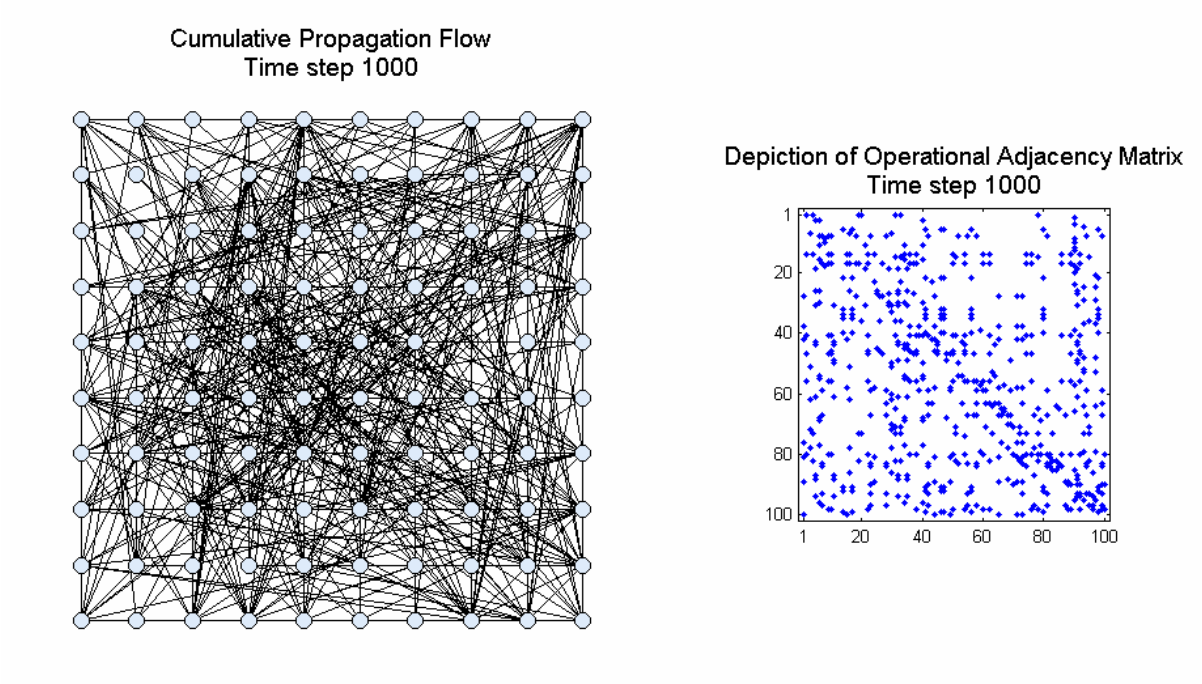


Figure 6.8 Cumulative Propagation Flow (time step 1000)

Figure 6.7 displays the cumulative propagation flow at time step 100 (i.e., the cumulative flow from time steps 1 through 100). The colored (light blue) nodes are the nodes participating in propagation. By time step 100, very few network nodes are not participating. Figure 6.8 displays the cumulative propagation flow at time step 1000 (the cumulative flow for the entire simulation run). Only *one* node (node 75 – when counting top-to-bottom and left-to-right) does not participate in propagation. Here's why: as it turns out node 75 is not an input node, not in the local or extended local neighborhoods of potential connecting nodes, and has low attachment preference strength. Node 75 is in danger.

Figures 6.7 and 6.8 include a depiction of the network operational adjacency matrix on the right-hand-side of each of the figures. We see that the distribution pattern of ones and zeros for these cumulative matrices is similar to the distribution pattern that we saw earlier for the individual-time-step matrices – only denser. All of these matrices have ones distributed

throughout with a particular concentration near the matrix major diagonal (again – more about this pattern shortly – at the end of the subsection).

The cumulative operational adjacency matrix depictions show where the network connections are, but do not show the intensity (flow value) of the connections. We have devised a companion view that provides the cumulative flow values of these connections. Figure 6.9 displays the cumulative flow value "adjacency matrix" (or *intensity matrix*) at time step 100. Figure 6.10 displays the cumulative intensity matrix at time step 1000.

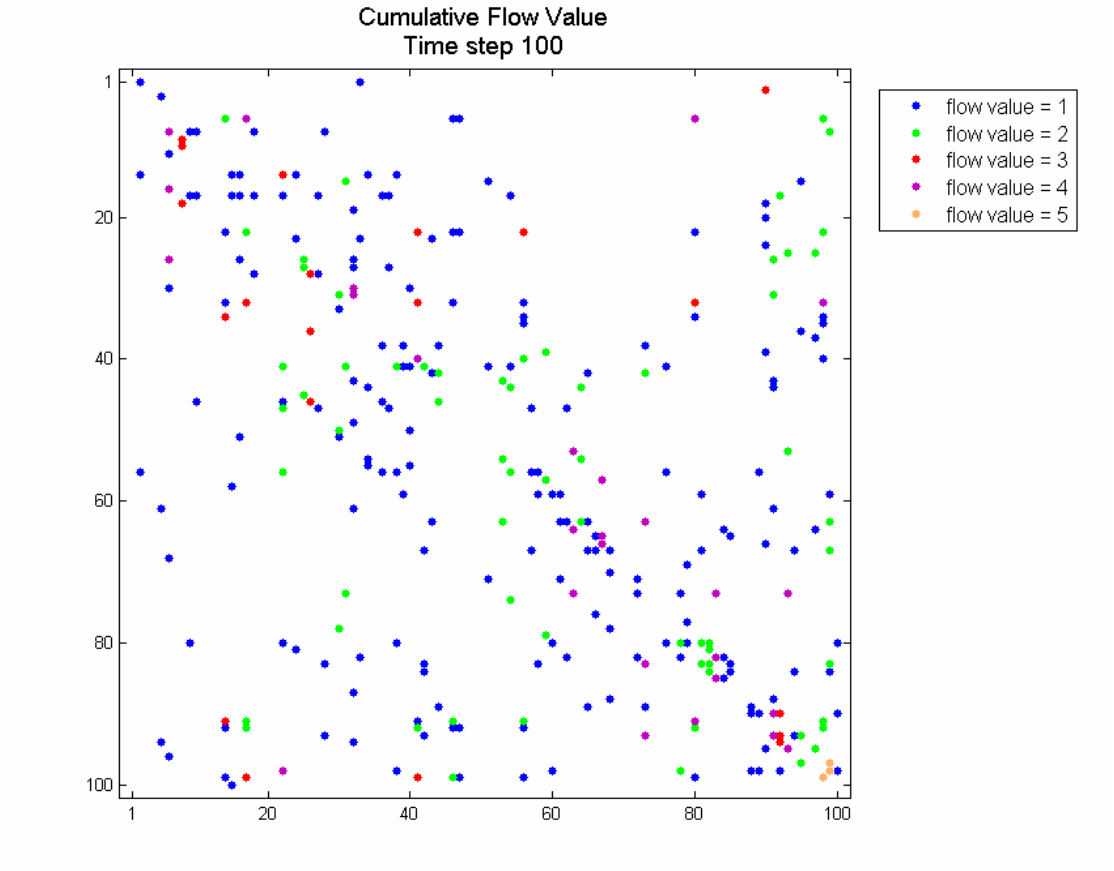


Figure 6.9 Cumulative Flow Value "Adjacency Matrix" (time step 100)

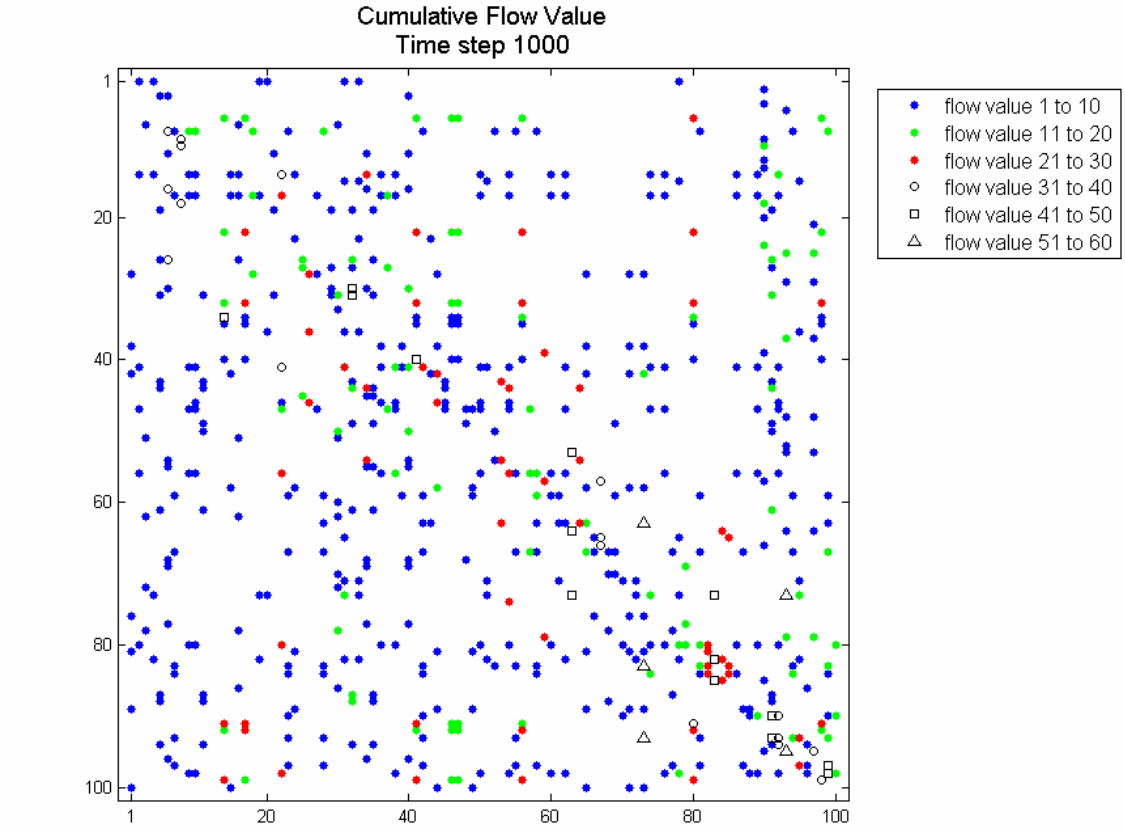


Figure 6.10 Cumulative Flow Value "Adjacency Matrix" (time step 1000)

The underlying distribution pattern in Figures 6.9 and 6.10 is the same as the pattern in Figures 6.7 and 6.8, respectively. We can, however, make an important additional observation: the highest flow intensities also tend to concentrate near the matrix major diagonal.

What is the significance of the adjacency matrix and intensity matrix patterns – and our observations about them? Herbert Simon has offered an explanation. In work published in 1976, Simon[111] said:

> *For real-world complex systems, if we construct an intensity matrix which reflects the intensity of interactions, "the large entries will be close to the diagonal, for these near-diagonal entries represent the interactions among elements that are close neighbors."*

[111] Herbert Simon, *How Complex Are Complex Systems?*, Proceedings of the Biennial Meeting of the Philosophy of Science Association, Volume 2, 1976.

The reasons for this involve complex system architecture. Simon[112] had earlier explained that complex system architecture is very often *hierarchical*:

> *"Empirically, a large proportion of the complex systems we observe in nature exhibit hierarchic structure."*

and that these hierarchic complex systems are very often *nearly decomposable*, i.e., at a given level of hierarchy, interactions among subsystems are weaker – often orders of magnitude weaker – than interactions within subsystems:

> *For example, "in organic substances, intermolecular forces will generally be weaker than molecular forces, and molecular forces than nuclear forces."*

So, close-neighbor (within a subsystem) interactions occur with higher intensity – as reflected by larger values close to the major diagonal of the intensity matrix. Furthermore, Simon[113] suggested that close-neighbor interactions occur with higher frequency:

> *"It is probably true that in social as in physical systems, the higher frequency dynamics are associated with the subsystems, the lower frequency dynamics with the larger systems."*

Higher frequency close-neighbor interactions are reflected by a higher concentration of ones close to the major diagonal of the adjacency matrix. We have observed these effects, described by Simon, in our adjacency matrix and intensity matrix results in Figures 6.5 through 6.10.

6.1.2 Input/Output/Stock Histories

We can generate representations of cumulative input, output, and stock values for the nodes in our model network at each simulation time step. The representations are three-

[112] Herbert Simon, *The Architecture of Complexity*, Proceedings of the American Philosophical Society, Volume 106, December 1962.

[113] Ibid.

dimensional bar charts overlaid on the network node grid. Results for time steps 100, 400, 700, and 1000 are displayed in the following sets of diagrams.

Figure 6.11 displays the input value results. Recall that there is one-unit-valued input to a randomly selected input node at each time step. In the underlying ecological network compartment model, five of the ten compartments are input compartments. There are, therefore, approximately 50 input nodes (~ 50% of the nodes) in our network node model. (I say "approximately" because I have randomly assigned nodes to compartments using a random number generator.) In the figure, some of the zero-valued grid elements are blocked from view by nonzero-valued elements. Since these are cumulative charts, the sequence of diagrams in Figure 6.11 shows monotonically increasing input values.

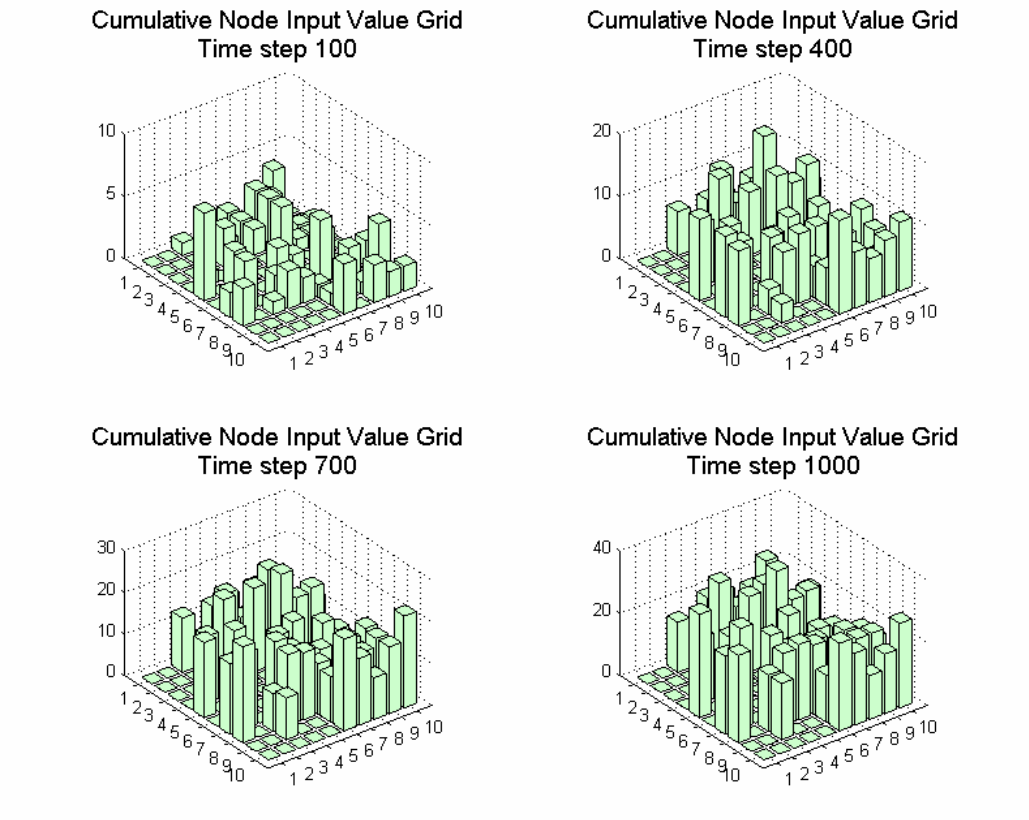


Figure 6.11 Cumulative Node Input Value Grid

Figure 6.12 displays the cumulative output value results. At each time step, each propagating node that is also an output node sends one unit-valued output to the external environment. If the node propagates more than once at a time step (cycling), it outputs more than once at the time step. In the underlying ecological network compartment model, six of the ten compartments are output compartments – so there are approximately 60 output nodes in our network node model. Some nodes, of course, are both input nodes and output nodes. Again, since these are cumulative charts, the sequence of diagrams in Figure 6.12 shows monotonically increasing values.

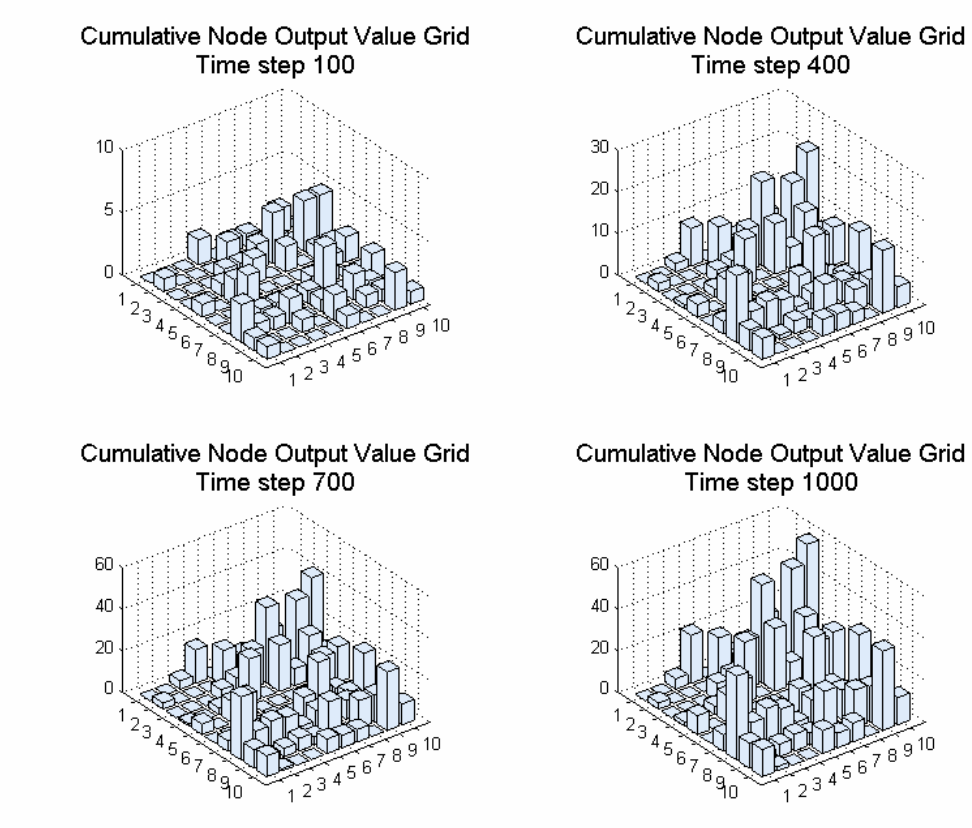


Figure 6.12 Cumulative Node Output Value Grid

Figure 6.13 provides the cumulative stock value of each of the network nodes. The node stock value initial conditions have been set using a uniform random assignment of zero to three before the start of the simulation run. A node propagation threshold of four and a node

propagation flow quantity of four are used for the simulation run, i.e., when node stock value becomes equal to (or greater than) 4, the node propagates 4 units of stock. As shown in Figure 6.13, therefore, node stock values continually change but remain in the zero-to-three range.

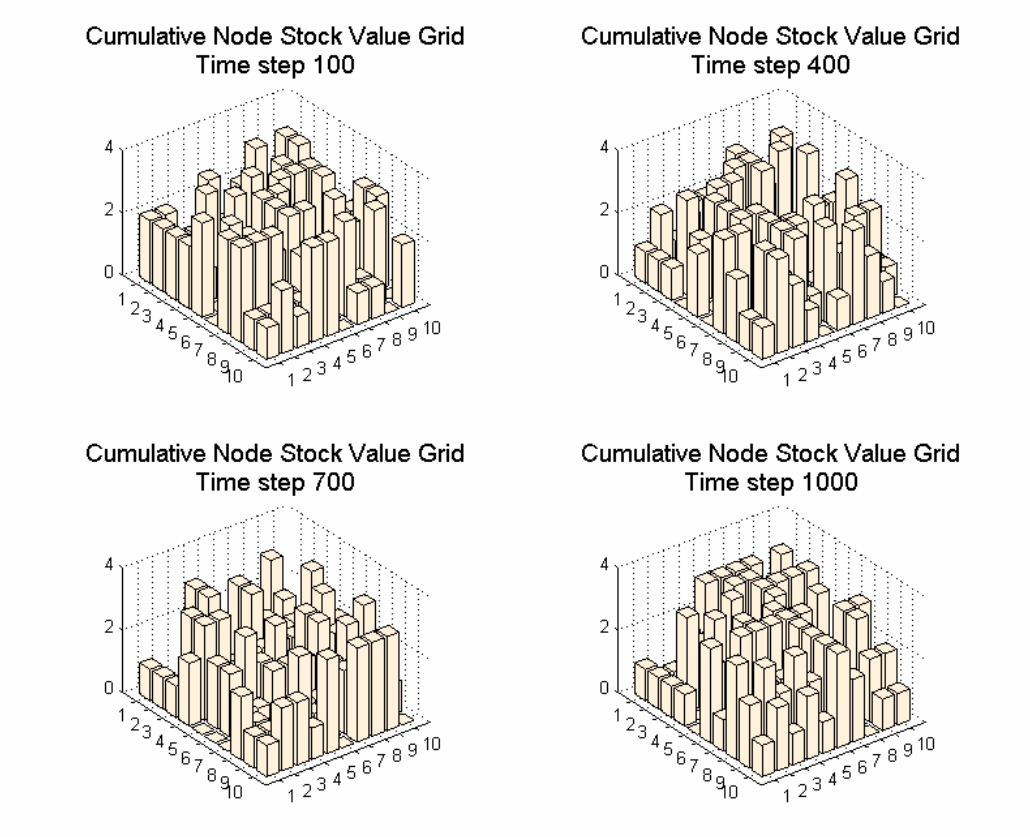


Figure 6.13 Cumulative Node Stock Value Grid

6.2 Network Propagation Event Results

Figure 6.14 displays the network propagation event time series and distribution. The time series plots network propagation event size vs. time. Network propagation event size at a time step is equal to the total number of node instances involved in propagation at that time step, i.e., the number of node propagation instances (variable *nnpi*) plus one (for the initial input node instance). Since any given node can participate in propagation more than once at a time step (due to cycling), propagation event size can be larger than the number of physical nodes involved in propagation at the time step and sometimes larger than the total number of nodes in the

network. The resulting time series clearly indicates punctuated and local-to-global dynamics – with many small events but also medium, large, and even a few very large propagation events interspersed. Smaller events are mostly local and larger events are global. The network propagation event distribution (number of events vs. size of events) shown in Figure 6.14 is derived from the time series.

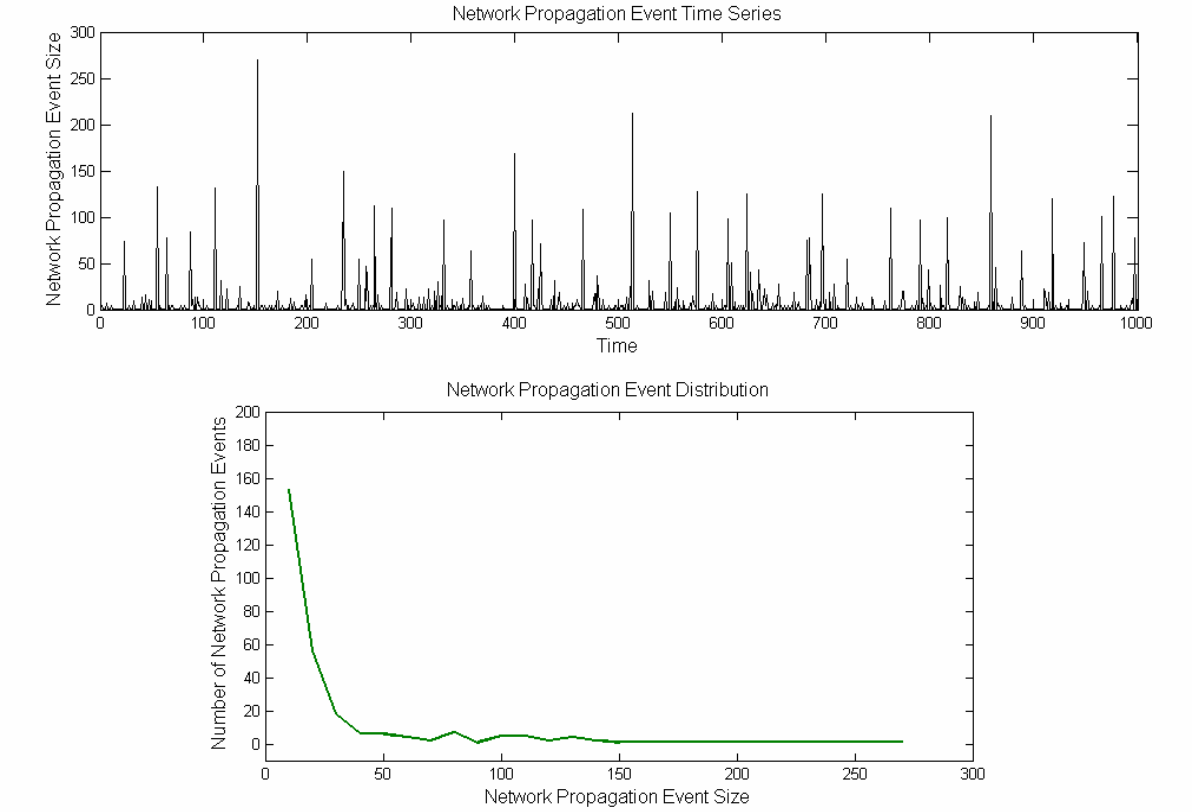


Figure 6.14 Network Propagation Event Time Series and Distribution

Let's take a closer look at the distribution with the help of Figure 6.15. At the upper left-hand corner of the figure, we again see the distribution plotted in normal coordinates. The curve is definitely "long-tailed" and perhaps it is a power-law curve.

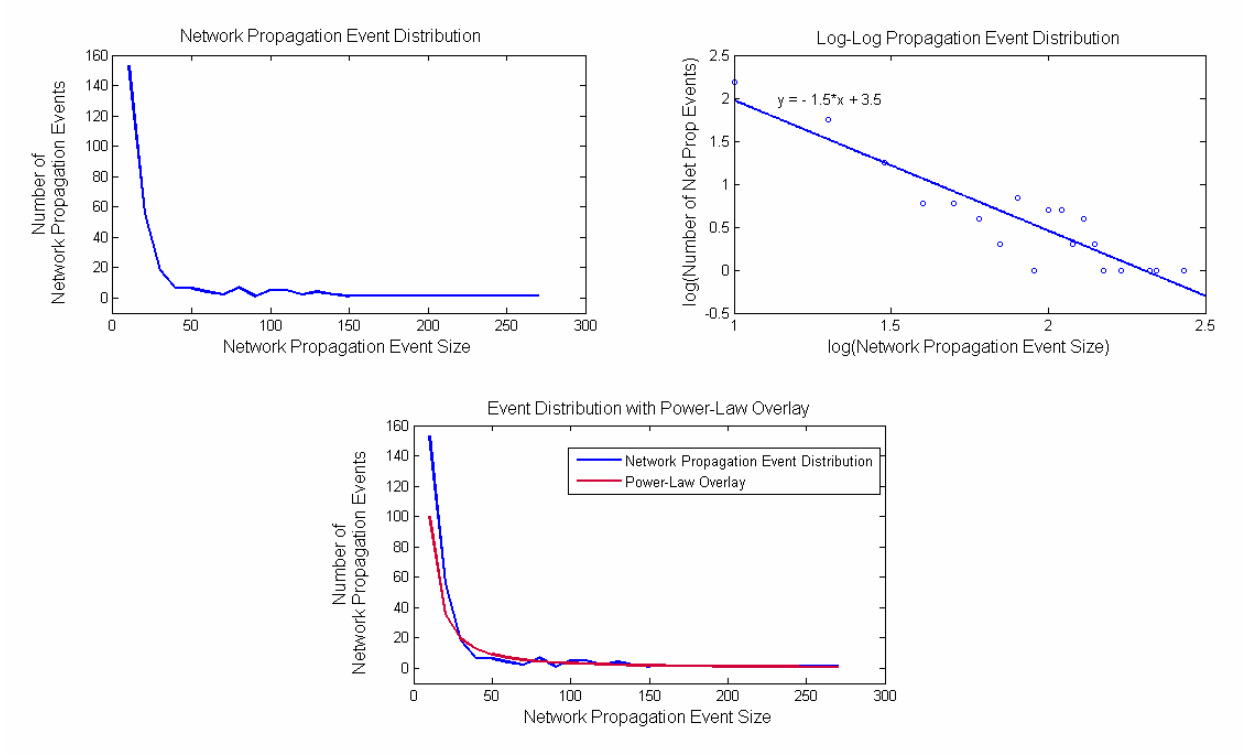


Figure 6.15 Network Propagation Event Distribution Set

Let's see if we can confirm power-law behavior. First, we note that a power-law curve has the form

$$y = C\,x^{-\lambda}$$

where C is a constant and λ is the *power* or *scaling exponent*. Taking the logarithm of both sides of the power-law equation, we obtain

$$\log y = \log C - \lambda \log x$$

In log-log space this, of course, is the equation of a straight line with *y-intercept* equal to log$C$ and slope equal to the negative of the scaling exponent. So here is the determining question: does our network propagation event distribution approximate a straight line in log-log coordinates? The log-log plot of our propagation event distribution is shown at the upper right-hand corner of Figure 6.15. We fit a straight line to the data points and observe that the equation of the log-log straight line is

$$y = -1.5x + 3.5$$

Now let's do one more plot in normal coordinates (lower half of Figure 6.15). We plot our original propagation event distribution (in blue) and overlay the power-law distribution calculated from the log-log plot (in red). The equation of the power-law overlay is

$$y = C\, x^{-\lambda}$$
$$y = 10^{(\text{loglog y-intercept})} * x^{(\text{loglog slope})}$$
$$y = 10^{3.5}\, x^{-1.5}$$

As shown in the figure, the fit is quite good! Power-law dynamics are confirmed. Network propagation events exhibit power-law/fractal behavior in space.

Next question: is network propagation event behavior also fractal *in time*? To answer that question, we need to examine a time-related distribution function that captures the apparently "noise-like" [114] ecological fluctuations represented by the network propagation event time series. We know from signal processing theory that the relevant distribution function is the so-called frequency spectrum – which indicates the contribution of each frequency to the overall time series. (Recall that frequency and signal time period are inversely related.) We examine the frequency spectrum looking for the power-law "pink *1/f noise*" characteristic, i.e., a broad range of frequencies with emphasis on the lower frequencies. A power-law "pink *1/f noise*" spectrum is a necessary condition for temporal fractal behavior.

The network propagation event frequency spectrum is obtained by taking the discrete Fourier transform (DFT) of the discrete propagation event time series. The result is shown in the top diagram of Figure 6.16.

[114] "Apparently" is the operative word here. These ecological fluctuations are <u>not</u> noise.

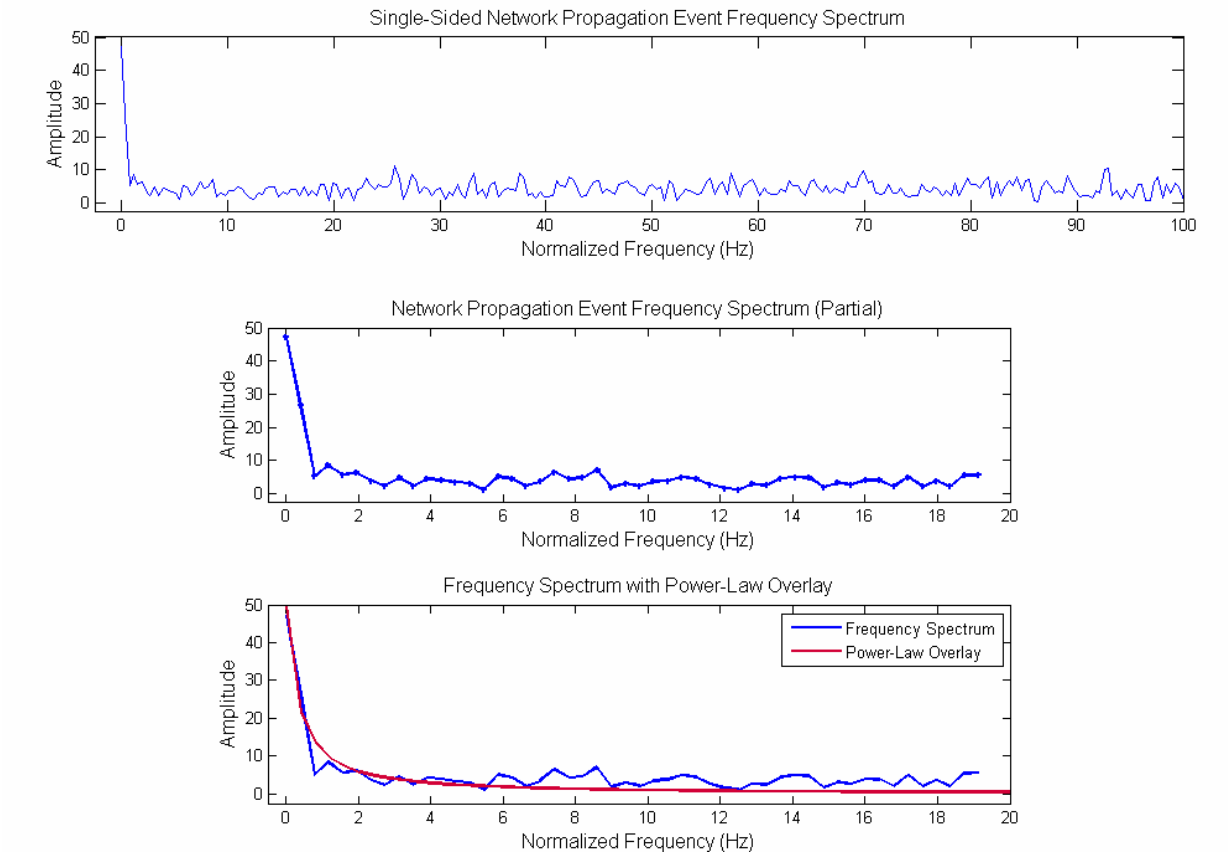


Figure 6.16 Network Propagation Event Frequency Spectrum

[Note that the normalized frequency abscissa in the figure is somewhat arbitrary. The frequency range depends on the signal sampling frequency which depends on the time duration of a time step (sampling frequency in Hz = 1/time step duration). Because my model is intended to apply to a broad set of real ecological networks, I have not specified any particular time step duration.] Since the lower frequencies are of most interest, the middle diagram of Figure 6.16 focuses in on the first 20 percent or so of the waveform. The spectrum appears to follow a power-law curve, but I cannot confirm this with a straight-line log-log plot. The accuracy of the Fourier transform data is not high enough for that.[115] On the bottom diagram of Figure 6.16, however, I do overlay

[115] The accuracy of these discrete Fourier transform computations is limited by the *relatively* short length of the signal time series. Much improved accuracy would result from signal lengths on the order of 10,000 or even 25,000 time steps. Such values, however, are way beyond the capacity of my computer.

a power-law curve on the frequency spectrum for comparison. The power-law curve parameters are chosen for "best fit." The equation of the power-law overlay is:

$$y = 50(x+1)^{-1.2}$$

In a power-law equation, when the independent variable is zero, the dependent variable goes to infinity. To avoid that situation, I have shifted the independent variable by one. The frequency spectrum fit to a power-law curve is good. This result[116] strongly suggests that the spectrum reflects "*pink 1/f noise*" and that network propagation event behavior is fractal in time.

Overall, our propagation event results indicate that network propagation event behavior is punctuated, local-to-global, and fractal in space and time.

6.3 Path Length Results

Path length is essentially a spatial concept. Our analysis results, however, show that propagation path length behavior changes (in dramatic punctuated fashion) with time – so that time considerations are also very important. To fully capture path length dynamics, therefore, we have devised an approach that integrates over both space and time. The approach is a spatial/temporal analog of the network "box covering" method for path length analysis (discussed in Chapters 4 and 5). Our approach uses the maximum path length achieved at a time step as a measure of path length behavior at that time step. Figure 6.17 shows the corresponding path length time series and space/time integrated distribution for the simulation run.

[116] Notice that our results here are quite similar to the results of the heart rate dynamics analysis that we discussed in Chapter 4.

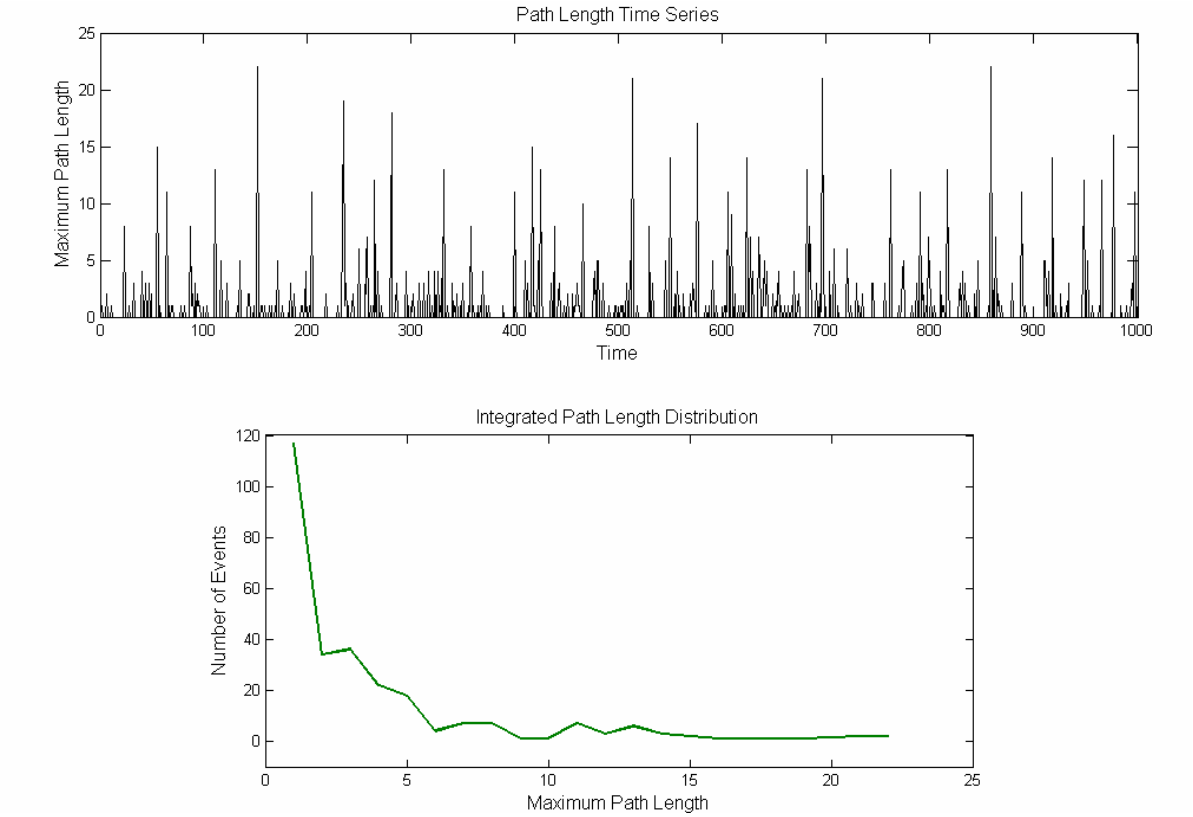


Figure 6.17 Path Length Time Series and Integrated Distribution

The time series is clearly punctuated and exhibits convincing evidence of local-to-global propagation. Path length values vary from very small (local propagation) to large (global propagation). The integrated path length distribution is "long-tailed" and suggests a power-law curve. [Note that the path length distribution is a bit "ragged." This is due to the small interval size on the abscissa (it's equal to 1) and, therefore, the relatively small number of samples per interval. The simulation run consists of 1000 time steps, but less than 300 are propagation time steps. These < 300 time steps yield path lengths from 1 to 22. In the distribution, we have a relatively large number of abscissa intervals (22) and a relatively small number of sample sets (less than 300). There are no "interval smoothing" effects here. The distribution data set is large enough to yield definitive behavior results, but not large enough to yield a "smooth" curve connecting the data points. Of course, more time steps would be expected to reduce or eliminate

the raggedness. In the literature, I've seen cellular automata models (e.g., the sandpile model and a model for binary genetic networks) that use 25,000 time steps. For my complex ecological network model, however, I am limited to 1000 time steps by the capabilities of my computer.]

We take a closer look at the path length distribution via Figure 6.18. The upper-left diagram in the figure is a repeat of the path length distribution plotted in normal coordinates.

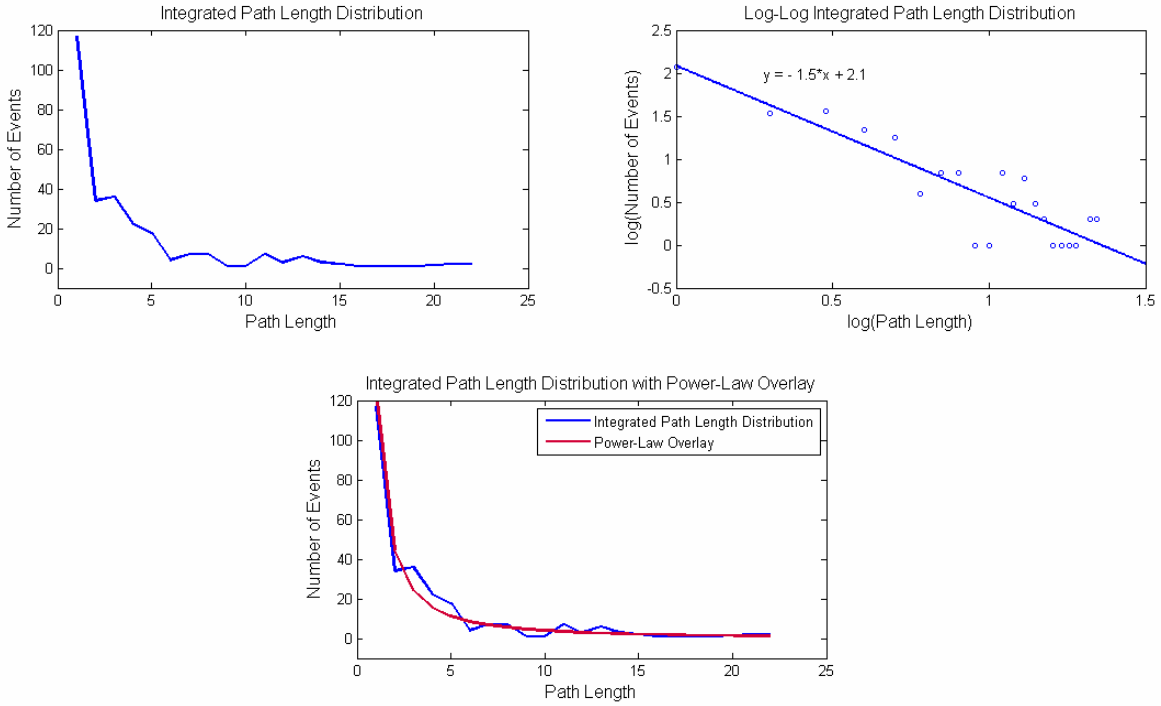


Figure 6.18 Integrated Path Length Distribution Set

Let's see if we can confirm power-law behavior. Let's see if the log-log plot of the path length distribution approximates a straight line. That plot is shown in the upper-right of Figure 6.18. We fit a straight line to the data points and observe that the equation of the log-log straight line is

$$y = -1.5x + 2.1$$

In the lower half of Figure 6.18, we plot our original normal-coordinates path length distribution (in blue) and overlay the power-law distribution calculated from the log-log plot (in red). The equation of the power-law overlay is

$$y = C\,x^{-\lambda}$$
$$y = 10^{(\text{loglog y-intercept})} * x^{(\text{loglog slope})}$$
$$y = 10^{2.1}\,x^{-1.5}$$

As shown in the figure, the fit is good. Power-law dynamics are confirmed. Path length exhibits fractal behavior.

Here's an interesting side point. *Network diameter* is defined as the longest of the shortest path lengths between any two nodes in a network. Note that the longest path length in an integrated path length distribution (as defined here) serves as an upper bound for the network diameter of the simulation-run network. For the simulation run that we are analyzing:

$$\textit{Network diameter} \leq 22$$

In Chapter 5, we mathematically derived a distribution-dependent condition for dominance of indirect effects: if the path length distribution scaling exponent is greater than 1 and less than 1.7, then indirect effects are dominant. Our data results here satisfy that condition. According to the mathematics, indirect effects should dominate in this simulation run! In the next subsection, we discuss indirect effects results. We will see if our indirect effects data results agree with the mathematical prediction. [Hint: they do.]

To obtain the above path length time series and distribution results (Figures 6.17 and 6.18), the measure of path length behavior that I chose was the maximum path length achieved at each time step. Are our observations and conclusions about path length dynamics sensitive to the choice of this "measure"? Let's pick another measure and see. Let's try the *mean* path

length achieved at each time step as the measure.[117] Figure 6.19 provides the resulting mean path length time series and integrated distribution set.

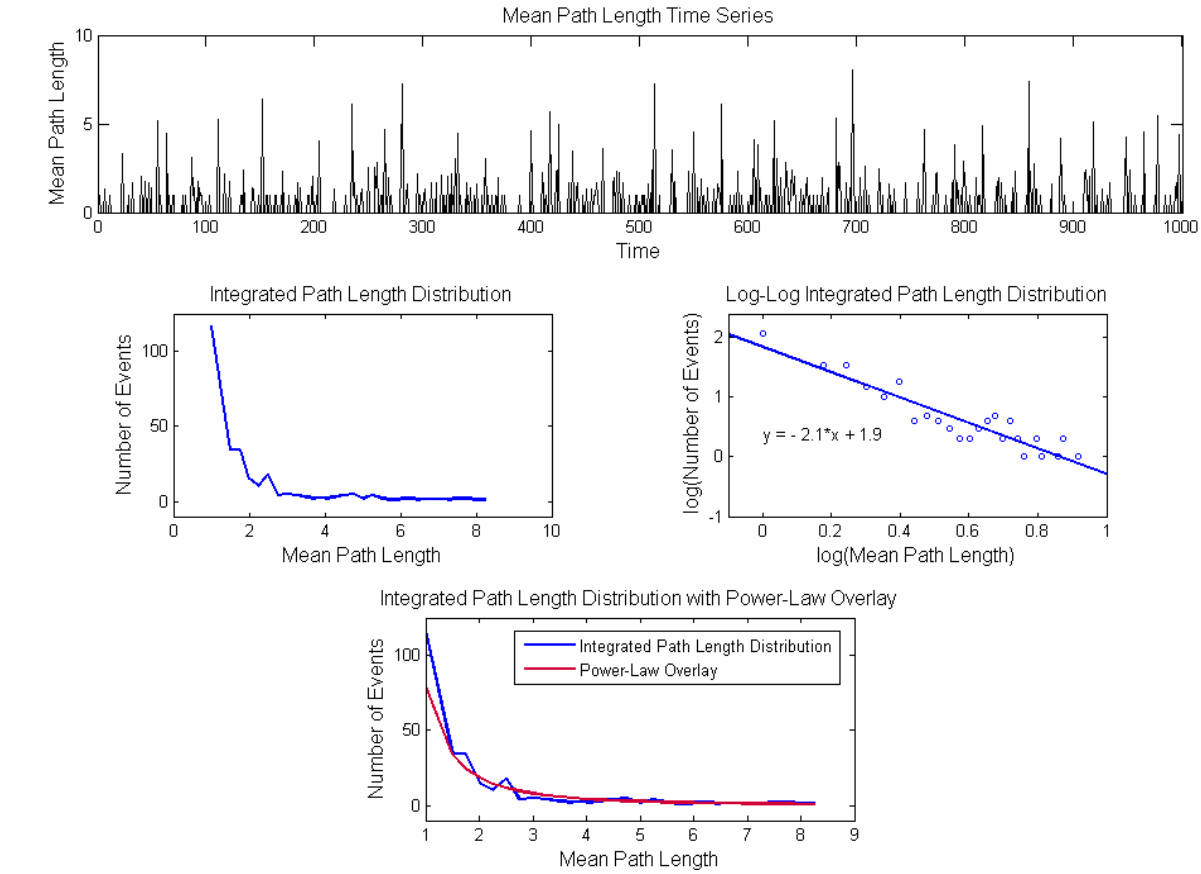


Figure 6.19 Mean Path Length Time Series and Distribution Set

The time series (top diagram) is clearly punctuated and exhibits convincing evidence of local-to-global propagation (for the same reasons given above for Figure 6.17). The integrated path length distribution (mid-left diagram) is "long-tailed" and suggests a power-law curve. In the log-log plot of the distribution (mid-right diagram), we fit a straight line to the data points and observe that the equation of the log-log straight line is

[117] The mean path length that we use here is an average at each individual time step and not an average over time. We know that averages over time "smooth" and lose the instantaneous dynamics of a network. Do individual-time-step averages (means) preserve instantaneous dynamics? We will see.

$$y = -2.1x + 1.9$$

The bottom diagram plots the normal-coordinates path length distribution from the mid-left diagram (in blue) and overlays a power-law distribution (in red). The difference between the data plot (blue) and the curve-fit overlay (red) at mean path length equal to one is due to "cutoff effects." For the data plot, I considered only time steps with propagation (which have a minimum mean path length of one). This "cuts off" the data distribution at mean path length = 1 and alters the low end (low abscissa values) of the curve-fit overlay. The equation of the power-law overlay is

$$y = C\, x^{-\lambda}$$
$$y = 10^{(\text{loglog y-intercept})} * x^{(\text{loglog slope})}$$
$$y = 10^{1.9} x^{-2.1}$$

Path length power-law/fractal behavior is corroborated. Important additional points are suggested by the results shown in Figure 6.19. Our path length analysis is not particularly sensitive[118] to the "measure" of path length dynamics that we use – as long as that measure reflects the time-varying nature of the dynamics. It appears that individual-time-step averages (means) do reflect and generally preserve those dynamics. It appears that it is valid to use such mean values in this and other similar analyses. [We will use this approach again (in a later subsection) in our node degree analysis.]

Overall, our results here indicate that path length exhibits punctuated dynamics, local-to-global propagation, and fractal behavior.

[118] Although the specific distribution parameters differ somewhat, the analysis general conclusions (punctuated, local-to-global, and fractal behavior) are the same.

6.4 Indirect Effects Results

We have defined several ecological network direct effects and indirect effects indicators. The *direct effects ratio* equals the number of direct paths (path length = 1) divided by the total number of paths. The *indirect effects ratio* equals the number of indirect paths (path length > 1) divided by the total number of paths. The *indirect effects index* equals the number of indirect paths divided by the number of direct paths. These indicators are defined for a given simulation-run network at a given point in time.

Figure 6.20 provides the individual-time-step time series for each of the indicators, i.e., the individual-time-step values vs. time. Perhaps the first thing that you notice about the figure is that all of these time series are highly punctuated. There is not a hint of gradual or continuous behavior. Next, you may notice that the direct effects ratio time series looks particularly "dense" with values of 1. Why is that? Recall, from the network propagation event time series and distribution, that most propagation events are small. In small one-stage propagation events, all paths are direct paths. These events yield all the 1 values in the direct effects time series. Notice also that there are quite a few values of two-thirds and one-half in that time series. Due to the propagation granularity in the model, small two-stage events often have twice as many direct paths as indirect paths and small three-stage events often have an equal number of direct and indirect paths. These effects yield the two-thirds and 0.5 values, respectively. The appearance of the direct effects ratio time series is simply a reflection of the large number of small propagation events. The indirect effects ratio time series in Figure 6.20, on the other hand, is a reflection of the high percentage of indirect paths that occur in mid-size to large propagation events. For these events, 50% to 90% or more of the paths are indirect paths. The indirect effects index time series (the bottom diagram in the figure) shows that the number of indirect

paths can be 5 to 10 or more times the number of direct paths at these time steps. For the larger propagation events, therefore, indirect effects far exceed direct effects.

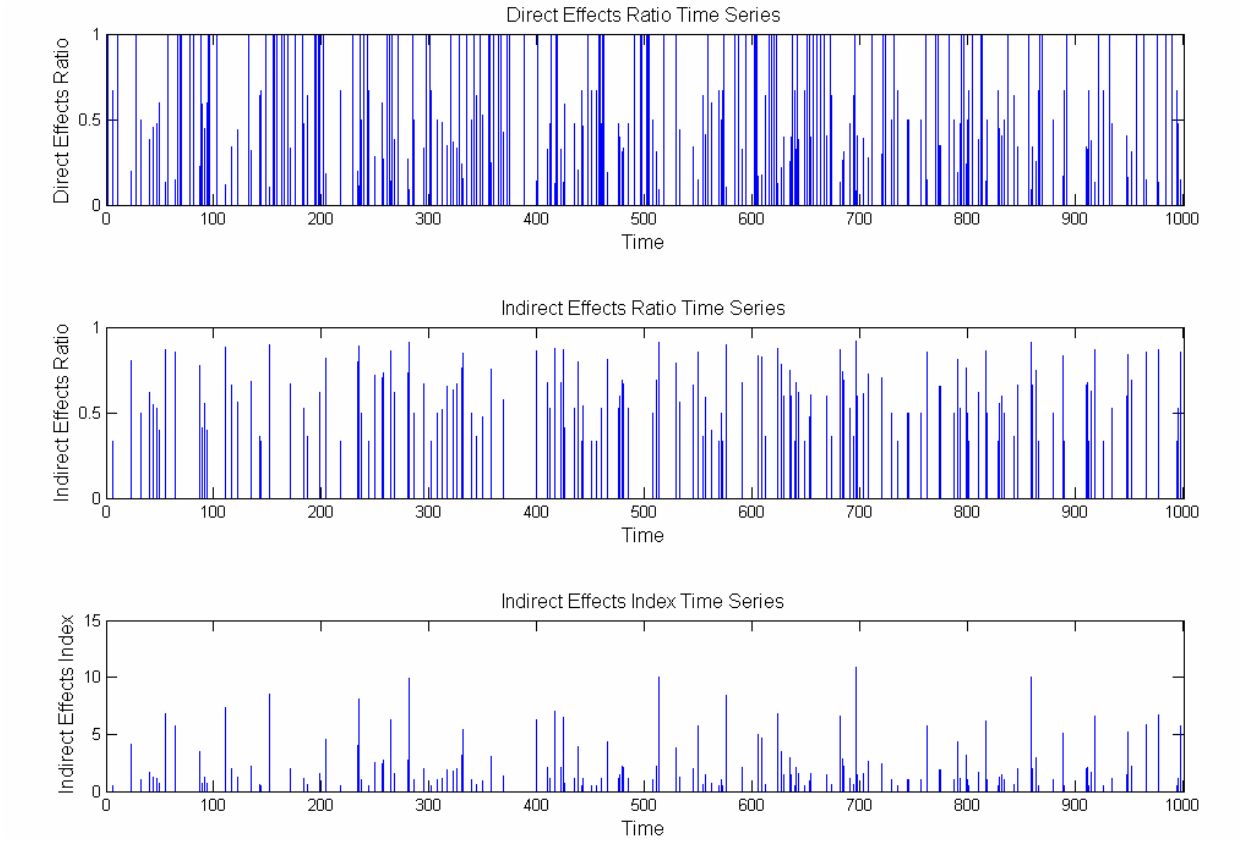


Figure 6.20 Indicator Individual-Time-Step Time Series

Earlier in this chapter, in the node-and-link propagation flow diagrams, we saw that the mid-size to large propagation events have broad network span. From a single (local) input node, there is propagation to an extensive (global) portion of the network. The path between any two given nodes is very often an indirect path. Clearly, indirect effects are enablers of local-to-global processing.

Okay, we see that indirect effects are dominant at high-propagation time steps. Are indirect effects dominant overall? To answer that question, we can look at the over-time

cumulative indicator time series in Figure 6.21. At any given time (time step), each of these time series shows the cumulative value of the indicator up to that point in time.

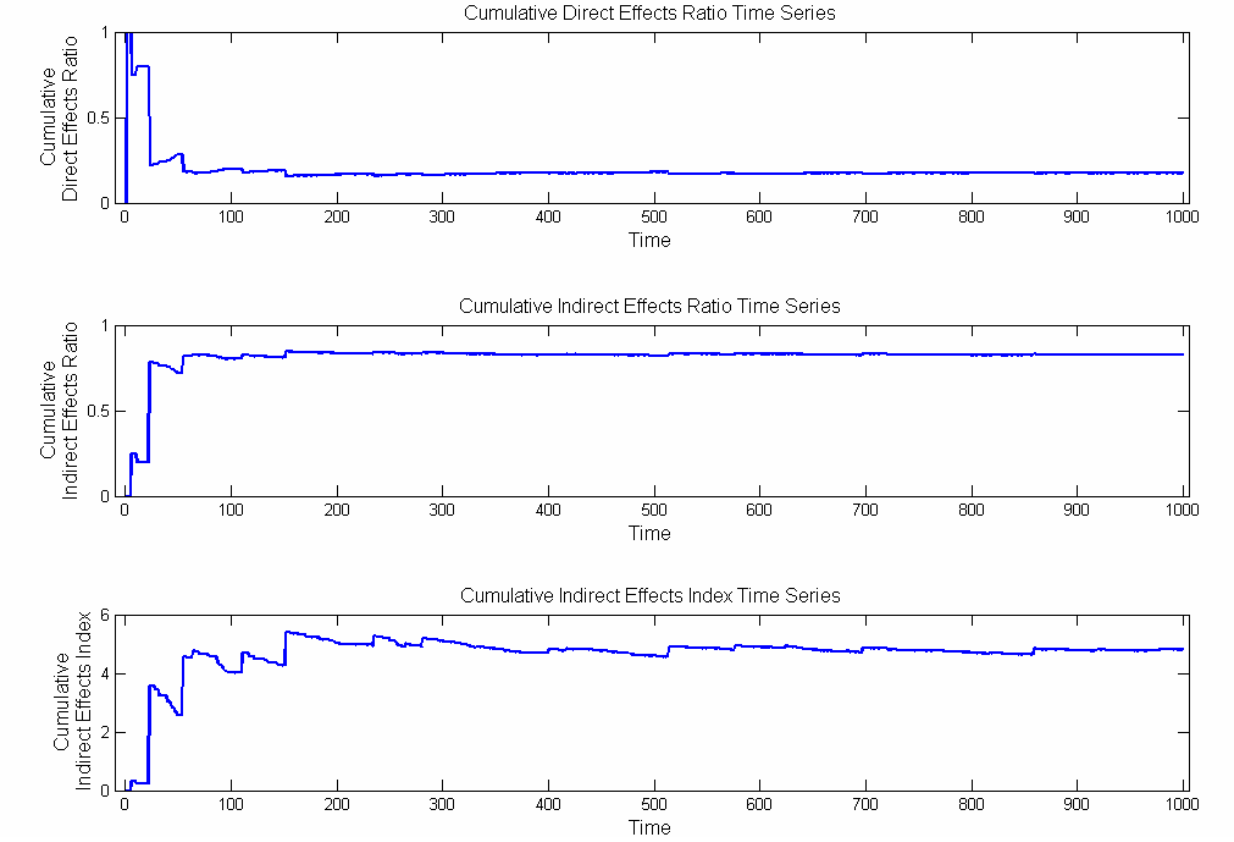


Figure 6.21 Indicator Cumulative Time Series

The first thing we notice here (compared with Figure 6.20) is that these cumulative averages "smooth" the instantaneous dynamics. The cumulative direct effects ratio settles in at a value of about 0.17. The cumulative indirect effects ratio reaches and maintains a value of about 0.83. Cumulative direct paths are < 20% of total paths and cumulative indirect paths are > 80% of total paths. The cumulative number of indirect paths is almost 5 times (actually 4.8 times) the number of direct paths. The 4.8 result is confirmed by the indirect effects index time series (bottom diagram of Figure 6.21). The following figure is also interesting.

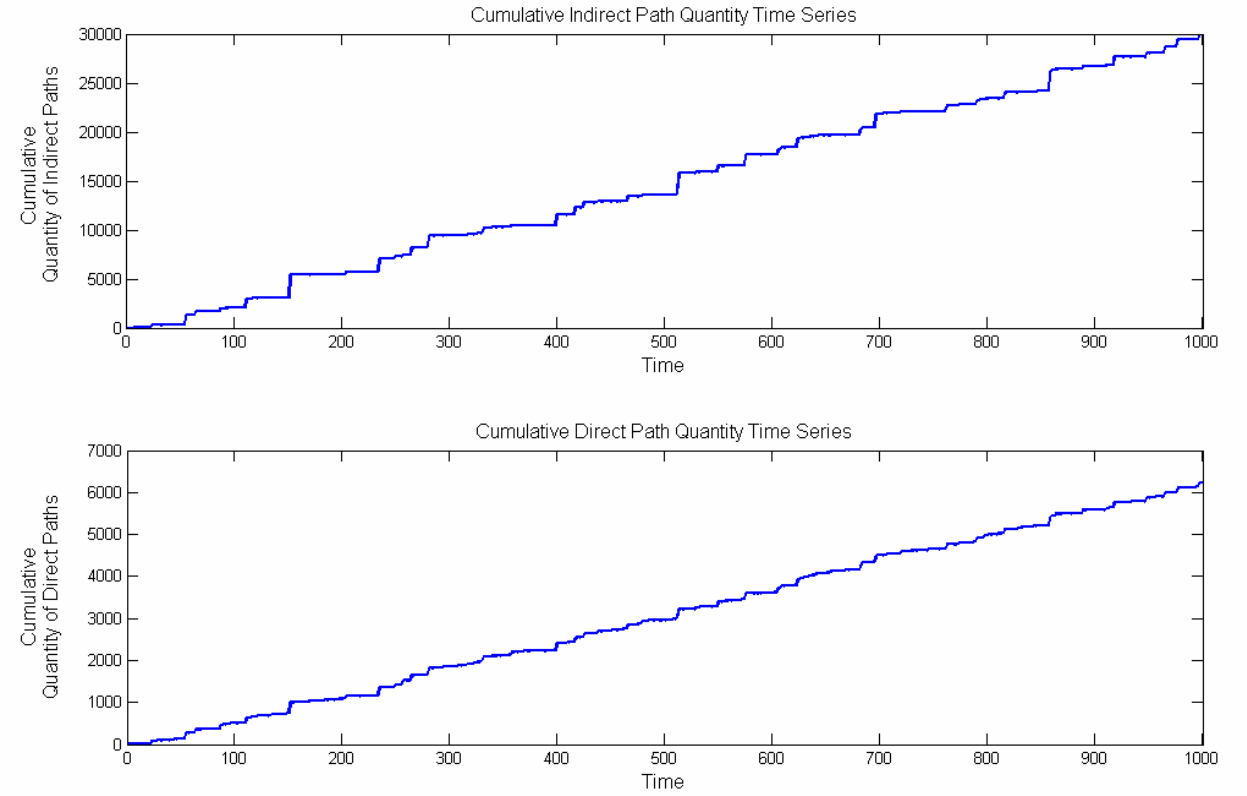


Figure 6.22 Cumulative Path Quantity Time Series

Figure 6.22 shows the cumulative quantity of indirect paths vs. time and the cumulative quantity of direct paths vs. time. Both plots are monotonically increasing (actually non-decreasing) as expected, but note the final values. There are about 6000 direct paths and almost 30000 indirect paths (that's the 4.8 factor again).

Indirect effects are definitely dominant.

Next, we want to turn to path quantity time series and distributions – and test them for fractal behavior. Figure 6.23 provides the indirect path results.

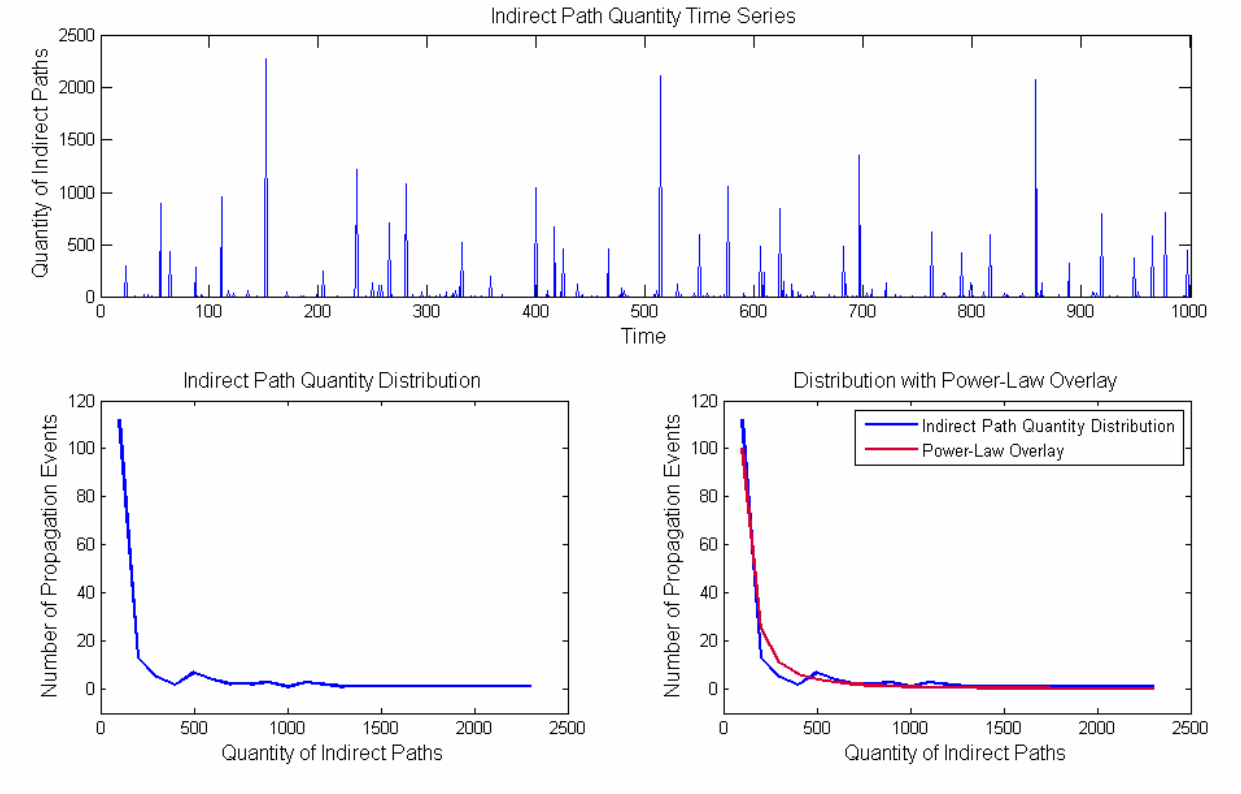


Figure 6.23 Indirect Path Quantity Time Series and Distribution

The time series (upper diagram in the figure) shows that there are very small (or zero) quantities of indirect paths at time steps corresponding to small propagation events and very large quantities of indirect paths at time steps corresponding to mid-size and large propagation events. For the largest propagation events, the quantity exceeds 2000. The lower-left diagram in Figure 6.23 displays the indirect path quantity distribution. The lower-right diagram repeats that distribution (in blue) and adds a power-law overlay (in red) for comparison. The equation of the power-law overlay is:

$$y = 10^6 x^{-2}$$

The fit is good. Indirect path quantity is fractal.

The analogous set of results for direct paths is provided in Figure 6.24.

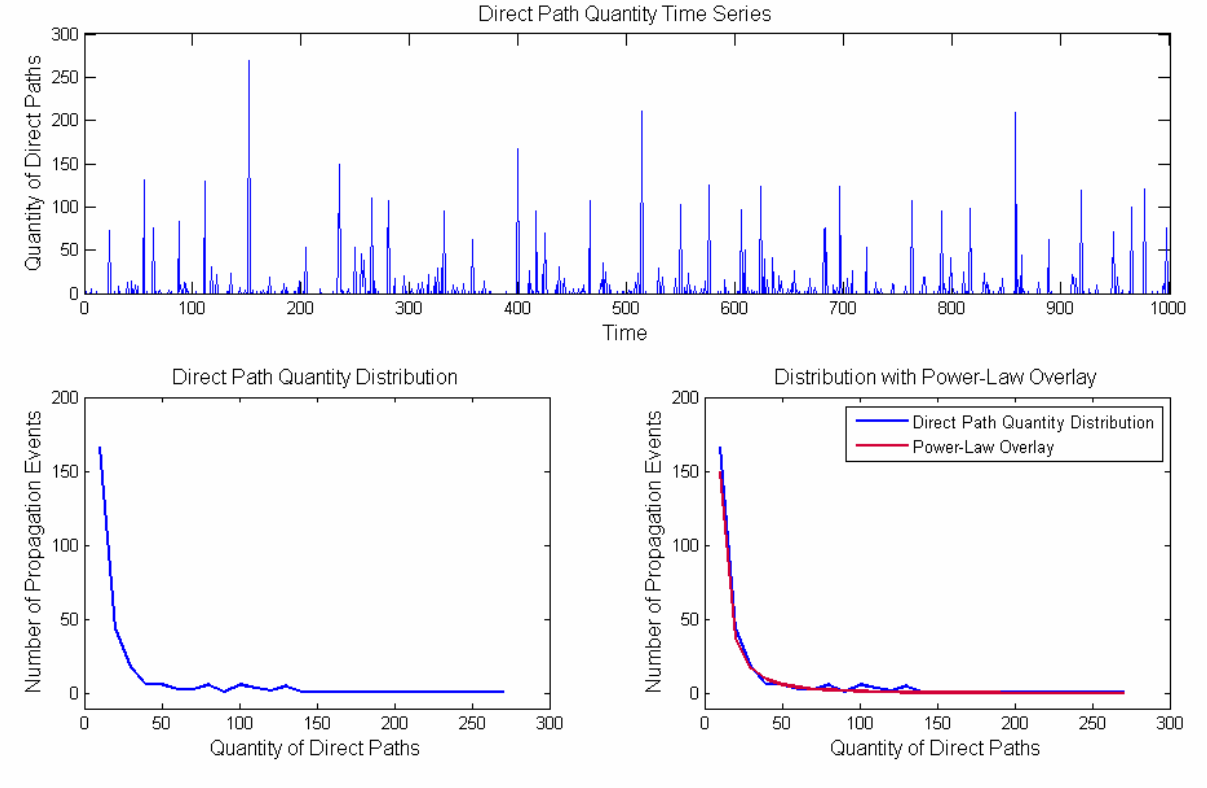


Figure 6.24 Direct Path Quantity Time Series and Distribution

The equation of the power-law overlay (shown in red on the lower-right diagram) is:

$$y = 1.5\left(10^4\right) x^{-2}$$

Direct path quantity is also fractal.

In summary, our results here show that indirect effects are dominant and that they are punctuated, enablers of local-to-global propagation, and exhibit fractal behavior.

6.5 Network Connectivity Results

Along with network propagation events, path length, and indirect effects, our results show that network connectivity exhibits punctuated, fluctuating dynamics. Node degree (the number of connections to/from a node) is an important indicator of network connectivity. Let's observe node degree behavior at several representative time steps. As shown on Figure 6.25, we

have selected three time steps (32, 64, and 152) that correspond to small, mid-size, and large propagation events, respectively.

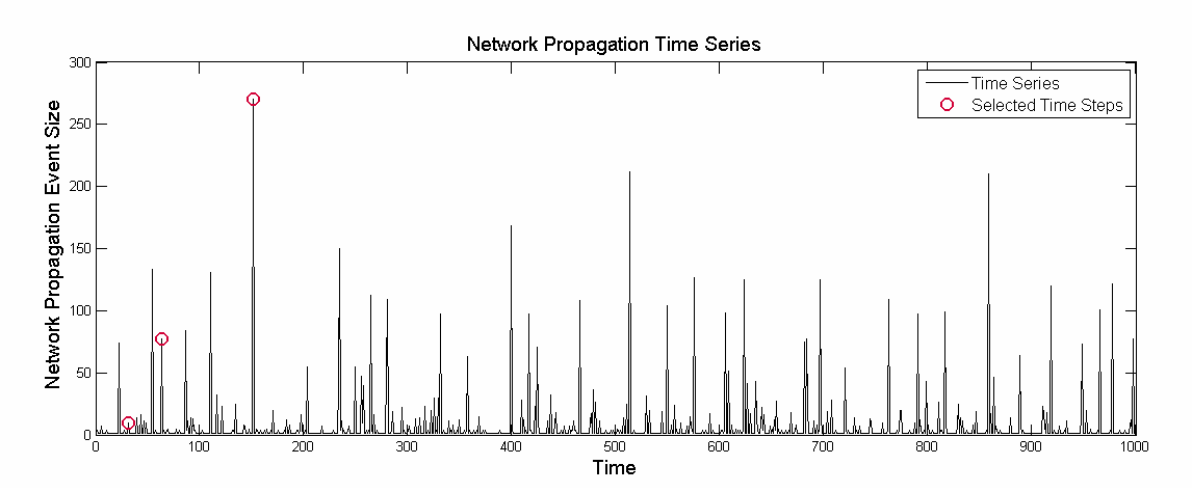


Figure 6.25 Network Propagation Time Series with Selections

To observe node degree behavior at those time steps, we have devised three-dimensional node degree grids (node degree overlays on the network grid). The node "in-degree" results are displayed in Figure 6.26 and the "combined-degree" (in-degree plus out-degree) results are displayed in Figure 6.27.

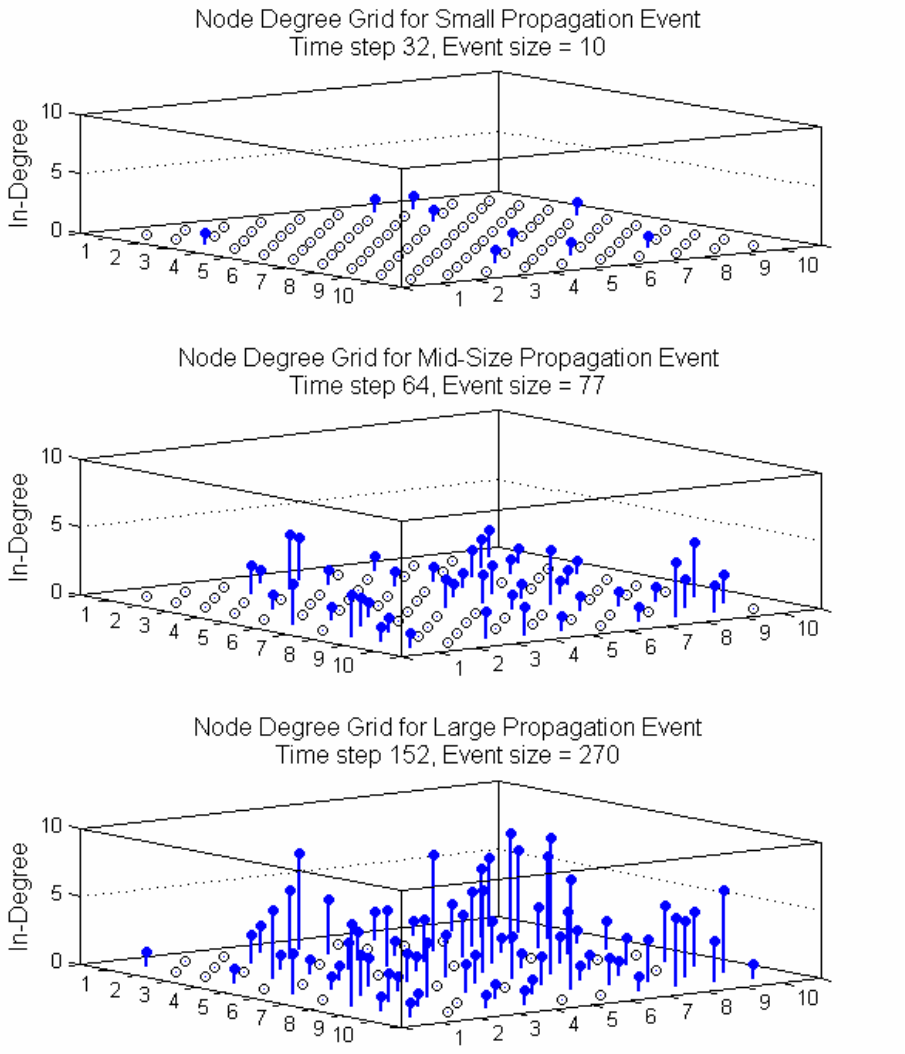


Figure 6.26 Node In-Degree Grid at Representative Time Steps

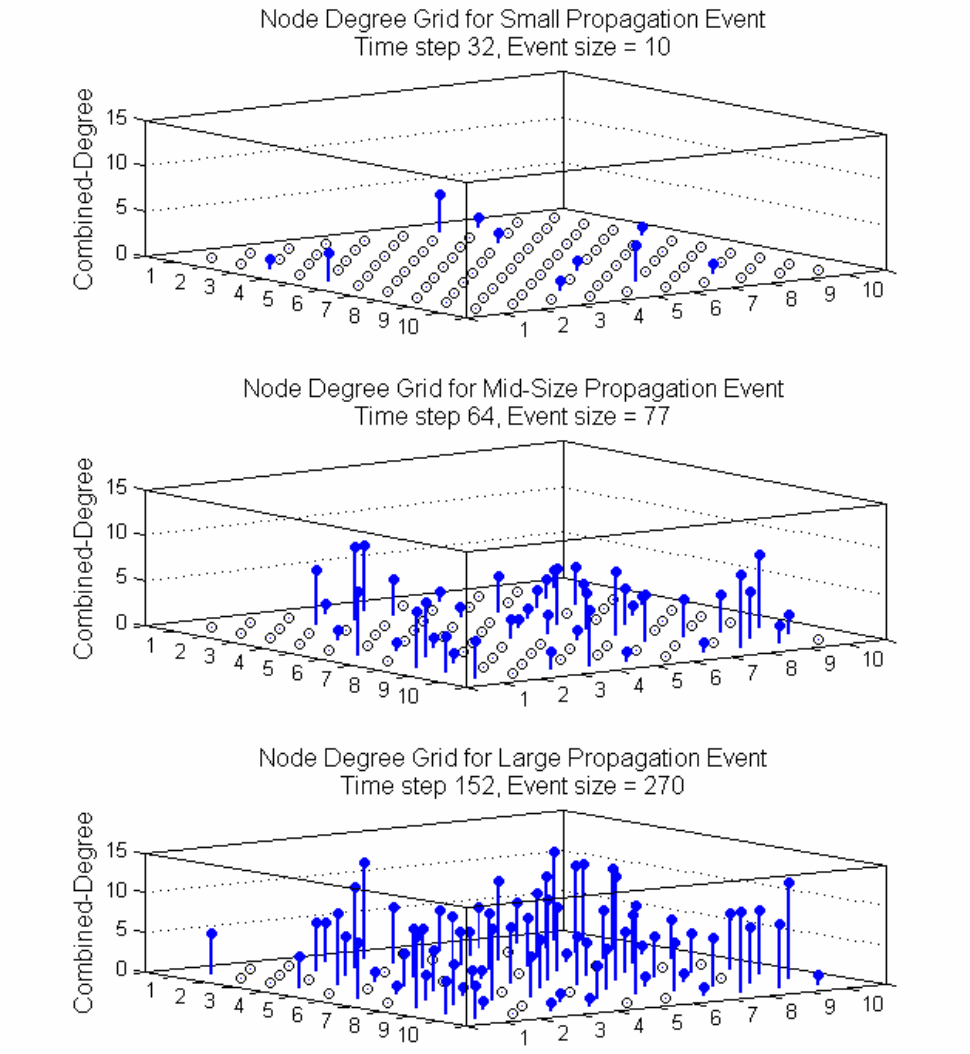


Figure 6.27 Node Combined-Degree Grid at Representative Time Steps

Figures 6.26 and 6.27 clearly illustrate the fluctuating dynamics of node degree over time. You can see that the instantaneous dynamics are ever-changing.

Before we proceed, note that there are some model "granularity effects" at play here. Since individual node inputs always occur in increments of 1, the effects are minimized for the in-degree case. For the out-degree case, on the other hand, granularity effects are maximized. Individual node outputs occur in quanta of 3 or 4. Out-degree data contains the most "jumps." The combined-degree case lies between the in-degree and out-degree cases. These granularity

effects show up especially in some of the distribution results, e.g., the combined-degree distribution of Figure 6.29 (that follows).

Next, we focus on node degree time series and distributions. We need to be able to capture the fluctuating dynamics (depicted in Figures 6.26 and 6.27) in an integrated fashion over space and time. To accomplish that, we use a spatial/temporal approach very similar to the path length approach (discussed and applied earlier). The first measure of node degree dynamics that we use is the maximum node degree achieved at each individual time step. The results are displayed in the time series and distribution sets of Figures 6.28 and 6.29 for in-degree and combined-degree, respectively. Observe Figure 6.28.

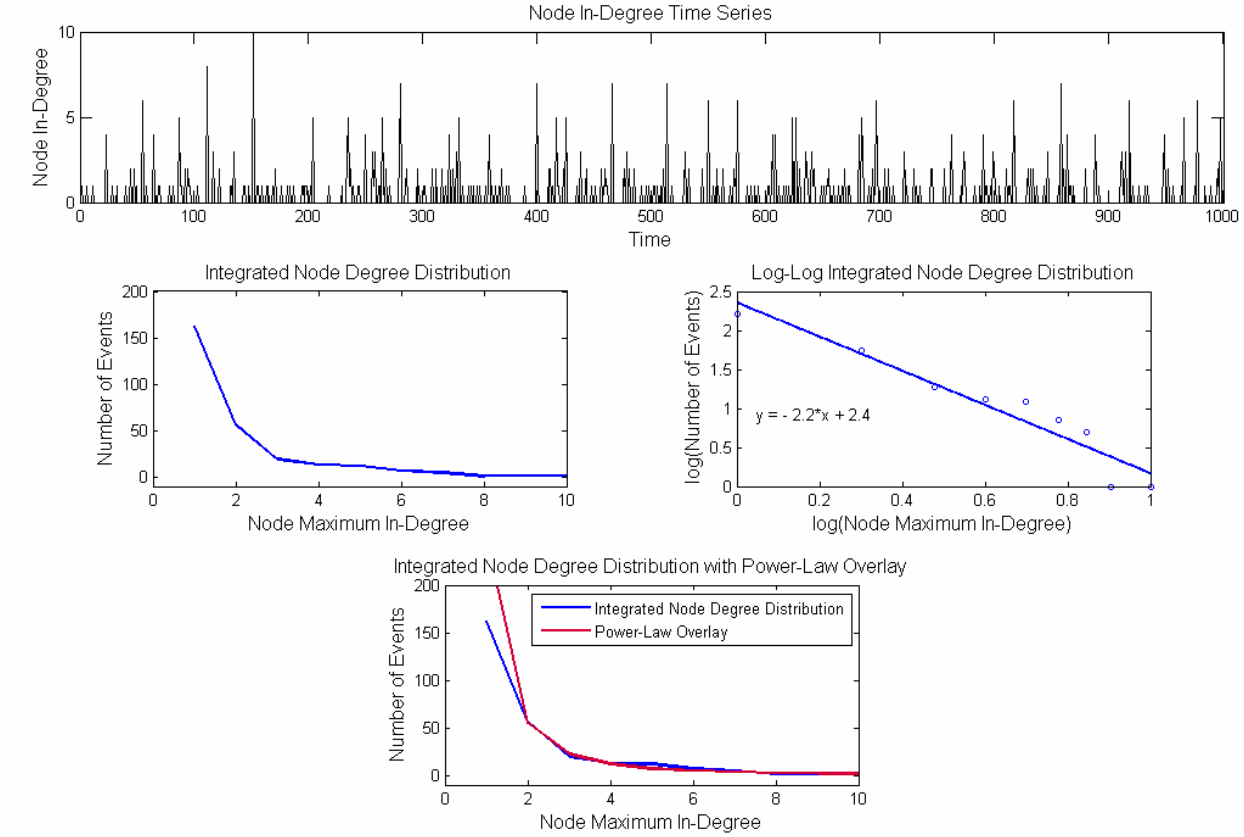


Figure 6.28 Node In-Degree Time Series and Distribution Set

The time series is clearly punctuated. The integrated node degree distribution is "long-tailed" and suggests a power-law curve. On the log-log plot, we fit a straight line to the data points and observe that the equation of the log-log straight line is

$$y = -2.2x + 2.4$$

Finally, we plot the original normal-coordinates path length distribution (in blue) and overlay the power-law distribution calculated from the log-log plot (in red). The equation of the power-law overlay is

$$y = C\,x^{-\lambda}$$
$$y = 10^{(\text{loglog y-intercept})} * x^{(\text{loglog slope})}$$
$$y = 10^{2.4} x^{-2.2}$$

As shown in the figure, the fit is good given that the largest value of node degree here is 10 – so that we are working with only about 10 data points. Power-law dynamics are corroborated. Node in-degree exhibits fractal behavior.

Consider Figure 6.29 for a moment.

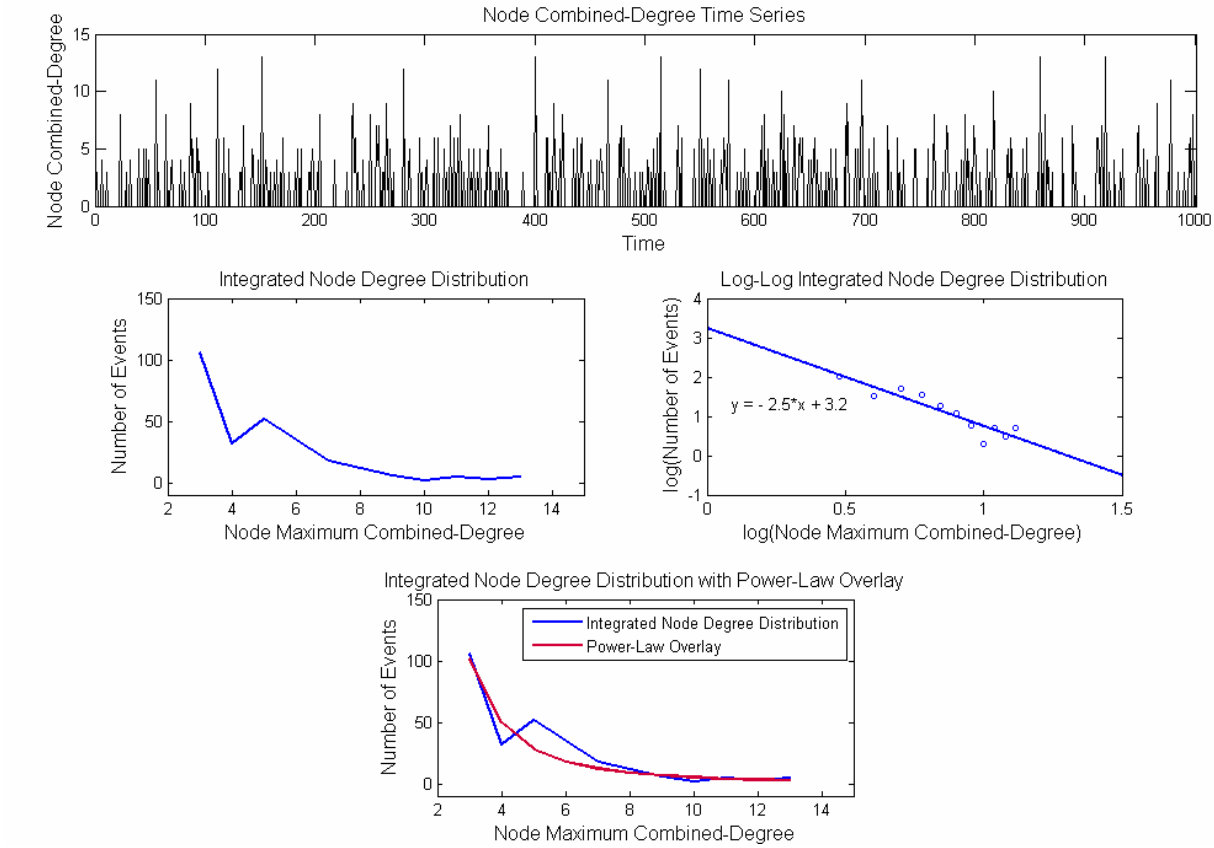


Figure 6.29 Node Combined-Degree Time Series and Distribution Set

These combined-degree results are similar to the in-degree results. The granularity effects that we mentioned earlier are reflected in the "jumps" in the distribution and the deviation from the power-law overlay curve in that region. Still, power-law dynamics and node combined-degree fractal behavior are corroborated.

Another measure that can capture the node degree dynamics is the node mean degree achieved at each time step. [Node mean degree is defined as the number of network edges divided by the number of network nodes.] The node-mean-degree time series and distribution set is shown in Figure 6.30.

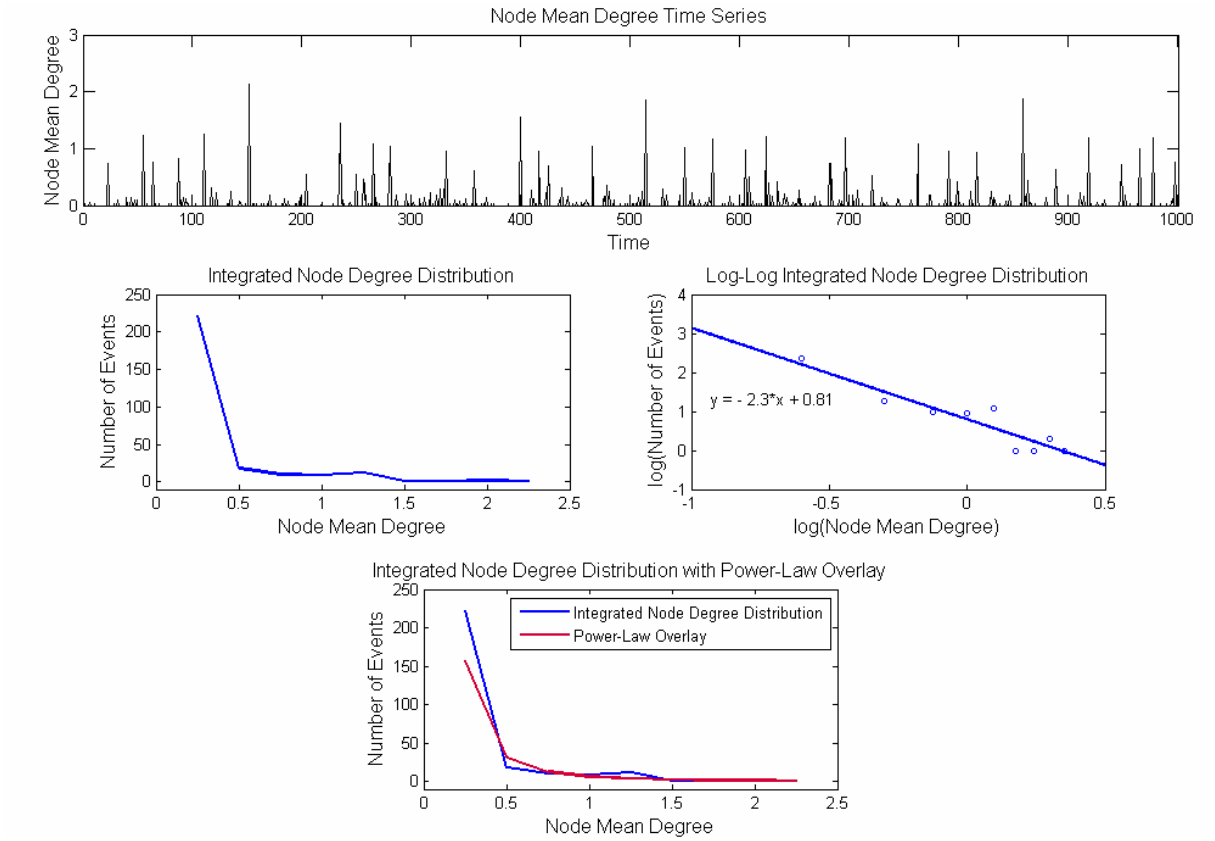


Figure 6.30 Node Mean Degree Time Series and Distribution Set

We see a punctuated time series and a "long-tailed" power-law node degree distribution. Even though we are dealing here with both granularity effects and relatively few data points, node-mean-degree power-law dynamics and fractal behavior are corroborated.

Let us now examine some other important network connectivity traits. To do this, we use the set of time series provided in Figure 6.31.

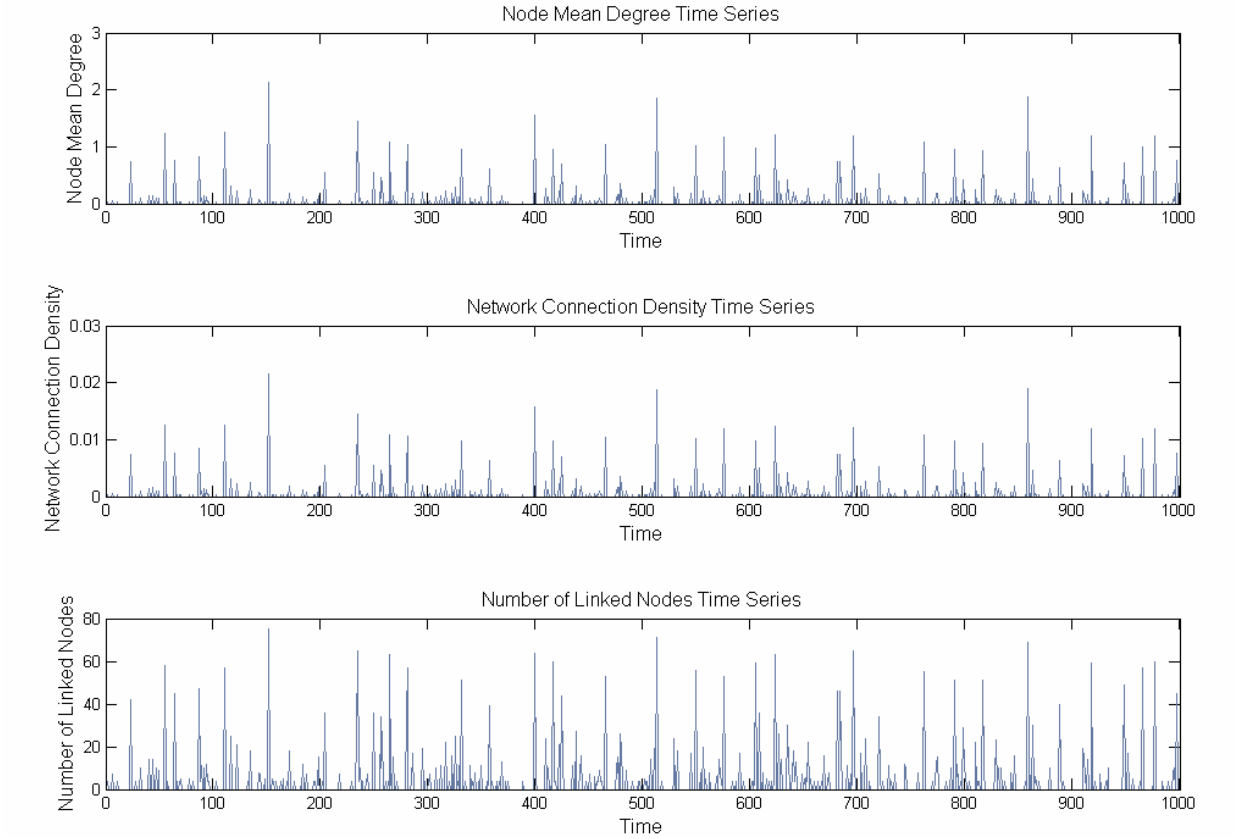


Figure 6.31 Network Connectivity Time Series Set

The first time series is the node mean degree time series. The second is the network connection density time series. Network connection density (also called connectance) is given by

$$Network\ density = \frac{\langle k \rangle}{n-1}$$

where

$\langle k \rangle$ = *node mean degree*

n = *number of nodes in the network*

Since node mean degree and network connection density differ only by a multiplicative factor, these two time series have the same form. The third time series provides the number of linked nodes at each time step.

Let's look at the time series values at the time steps that correspond to the largest propagation events in the simulation run: time steps 152, 514, and 859. At these time steps, the

number of linked nodes ranges from 69 to 75 – out of 100 nodes in the network. This is a very significant fraction of the total network. Although our model does not support a full network critical connectivity/percolation analysis,[119] our results here strongly suggest that network critical connectivity is achieved at these three time steps. We know that, for a directed random network at critical connectivity, the theoretical value of node mean degree is 2. For our network, we see from Figure 6.31 that node mean degree is approximately 2 at all three of our "critical" time steps. We also see that network connection density is approximately 0.02 at these time steps.

So – at the highest-propagation, highest-density time steps, node mean degree is approximately 2 and network connection density is approximately 0.02.[120] This is very *sparse* connectivity! As Stuart Kauffman[121] has said in a somewhat similar context (Boolean networks), this result should "blow your socks off." Kauffman says further that "astonishingly simple rules, or constraints, suffice to ensure that unexpected and profound dynamical order emerges spontaneously." "If the network is 'sparsely connected', then the system exhibits stunning order." "Our intuitions about the requirements for order have, I contend, been wrong for millennia. We do not need careful construction; we do not require crafting. We require only that extremely complex webs of interacting elements are sparsely coupled."

---

[119] A full network critical connectivity/percolation analysis would require development of multiple clusters at each simulation time step. If the multiple clusters should coalesce into a "giant cluster" that covers a major fraction of the network, then critical connectivity/percolation is achieved. Our model has one input and, therefore, develops just one cluster per time step. We can, however, observe that even a single cluster can approach a giant cluster at high-propagation time steps.

[120] Note that the actual operational values of node mean degree and network connection density that are achieved by our model network are much less than the candidate values. For comparison, the candidate node adjacency matrix yields a node mean degree of 30.35 and a node connection density of 0.31.

[121] Stuart Kauffman, *At Home in the Universe – The Search for Laws of Self-Organization and Complexity*, Oxford University Press, 1995.

In a 1976 paper, Herbert Simon[122] also talks about sparseness. "It will be convenient to represent the interconnectedness of a system by an incidence matrix [adjacency matrix], a matrix of zeros and ones, the (i, j)th element being 1 if the ith element interacts with the jth, and 0 otherwise. There are a number of different reasons why we might expect most real-world systems to have rather sparse incidence matrices." The "different reasons" involve the *hierarchy* and *nearly decomposable* properties of complex systems – discussed earlier in this chapter.

Our results here can be summarized as follows. Ecological network connectivity exhibits punctuated, fluctuating dynamics. Node degree is fractal. It appears that network critical connectivity is achieved at high-propagation time steps. All of this occurs in sparsely connected networks.

[122] Herbert Simon, *How Complex Are Complex Systems?*, Proceedings of the Biennial Meeting of the Philosophy of Science Association, Volume 2, 1976.

CHAPTER 7

SUMMARY AND CONCLUSIONS

The work presented here is an innovative and effective blend of synthesis and analysis. Early on, we *synthesize* a functional framework to set the context and direction for the work. In systems engineering, I have found that such a framework is essential for *specifying* and *guiding* the design and development of an artificial (human-made) system. In systems ecology, I believe that such a framework is equally essential for *understanding* natural systems, i.e., ecosystems. The framework identifies three "core functions" of ecosystems: self-organization, adaptation/regulation, and propagation. This set of interrelated functions is fundamentally important to my research. The *propagation* function becomes the focus for the work. The framework also recognizes that ecological systems consistently take the form of networks. The implementation architecture of ecosystems is the *network*. Next, based on an extensive review of the complex systems and network literature, we *synthesize* a view of the propagation dynamics of ecological system networks. This view, in turn, is the basis for the central hypothesis of the research: ecological networks are ever-changing, "flickering" networks with propagation dynamics that are punctuated, fractal, local-to-global, and enabled by indirect effects. At this point, *analysis* takes the spotlight in the work. We define, design, and develop an ecological network dynamics model to analyze and fully test the hypothesis. We make every effort to produce a realistic ecological network propagation model, e.g., we link our node model to an underlying existing and verified ecological network compartment model; we spatially distribute the network nodes and implement von Neumann propagation neighborhoods for further realism;

we include the preferential attachment behavior that is observed in many real-world networks. The resulting model software development effort is very substantial and includes not only a comprehensive implementation of the propagation process but also a full complement of analysis capabilities and graphics generation capabilities. We analyze and display the dynamics of operational propagation flow, network propagation events, propagation path length, indirect effects, and network connectivity.

Our modeling, analysis, and results fully and comprehensively test and corroborate the central hypothesis of the research – as summarized in the following five paragraphs:

We have analyzed *operational propagation flow* for a range of propagation events from very small to very large. Several points are evident. Flow behavior over time is punctuated. Many propagation events, and perhaps all of the mid-size to large events, have large network span and exhibit local-to-global propagation behavior. Lots of cycling is evident in the flow diagrams. The network continually changes with time – and sometimes dramatically so. The view of the network operational dynamics is clear: we are dealing with ever-changing, "flickering" ecological networks.

Our *network propagation event* analysis results complement the propagation flow results – and go further. The network propagation event time series indicates punctuated and local-to-global dynamics – with many small events but also medium, large, and even a few very large propagation events interspersed. The corresponding event distribution confirms power-law dynamics. Network propagation events exhibit fractal behavior in space. The event frequency spectrum strongly suggests "pink *1/f noise*" and fractal behavior in time as well.

To capture the network time-varying *path length* dynamics, we have devised an analysis approach that integrates over both space and time. We first use the maximum path length

achieved at a time step as a measure of path length behavior at that time step. Examination of the resulting time series and distribution confirms that path length exhibits punctuated, local-to-global, and fractal characteristics. We also checked this path length distribution against a condition for dominance of indirect effects that we had previously derived mathematically. The path length distribution satisfies that condition. Accordingly, indirect effects should dominate in our model network. (They do.) Next, to test the sensitivity of the time series/distribution analysis results to our choice of the "measure" of path length dynamics, we chose another measure (*mean* path length achieved at each time step) and performed the analysis again. Although the specific parameter values of the resulting distribution differed somewhat from the first case, the same general behavior (punctuated, local-to-global, and fractal) was confirmed.

To analyze ecological network *indirect effects* dynamics, we have defined several indirect effects indicators, calculated their values at each individual time step, and plotted the resulting time series. We see that the time series are highly punctuated – with no sign of gradual or continuous behavior. We see, at time steps of larger propagation events, that indirect effects far exceed direct effects. The number of indirect paths can be 5 to 10 or more times the number of direct paths. Given the large number of indirect paths and the global span of network propagation in these cases, we can conclude that indirect effects are enablers of local-to-global processing. In addition to their dominance at high-propagation time steps, are indirect effects dominant overall? To answer that question, we have developed indicator over-time cumulative time series. Results show that cumulative direct paths are < 20% of total paths and cumulative indirect paths are > 80% of total paths. The cumulative number of indirect paths is almost 5 times the number of direct paths. Indirect effects are dominant overall. Next, we turn to path quantity time series and distributions – and test them for fractal behavior. Results show that

indirect path quantity is fractal (and so is direct path quantity). In summary, indirect effects are dominant and they are punctuated, enablers of local-to-global propagation, and exhibit fractal behavior.

Network connectivity also exhibits punctuated, fluctuating dynamics. Node degree (the number of connections to/from a node) is an important indicator of network connectivity. Using three-dimensional node degree grids, we observe dramatically changing node degree behavior over time. We need to capture those fluctuating dynamics in our node degree time series/distribution analyses. To do that, we use an integrated spatial/ temporal approach very similar to the path length approach (applied earlier). The first measure of node degree behavior that we use is the maximum node degree achieved at each individual time step. The resulting time series is punctuated and the power-law distribution indicates node degree fractal dynamics. The second measure of node degree behavior that we use is the node mean degree achieved at each time step. Again, the results show a punctuated time series and a power-law distribution that indicates node degree fractal dynamics. Next, we examine some other important network connectivity traits. Our analysis results strongly suggest that network critical connectivity is achieved at time steps that correspond to the largest propagation events. At these highest-propagation highest-density time steps, node mean degree is approximately 2 and network connection density is approximately 0.02. This is very *sparse* connectivity. In summary, our results indicate that ecological network connectivity exhibits punctuated, fluctuating dynamics. Node degree is fractal. Network critical connectivity is likely achieved at high-propagation time steps. All of this occurs in sparsely connected networks.

Our ecological network dynamics model reflects the probabilistic nature of real-world ecological networks:

The model is probabilistic in many key respects, i.e., in the spatial distribution of nodes (and, therefore, their compartments); in the establishment of model network initial conditions; in the application of ecological network inputs to input nodes; in the node-to-compartment attachment preferences; and in the node-to-node attachment preferences. The probabilistic nature of the model and the simulation runs authentically reflects the "chance and change" aspects[123] of real ecosystems. Each simulation run, although different in specifics, demonstrates the same general behaviors. The principles we have developed here are robust – and in many respects are universal (more on robustness and universality next).

The model and the principles that it represents are robust:

Model general results are insensitive to changes in specific model settings (e.g., node propagation threshold, node propagation flow quantity). Results are not sensitive to variations in initial conditions. Results are robust with respect to choice of measure of the instantaneous dynamics of a given network propagation parameter. It seems that any reasonable measure will do – e.g., when determining path length dynamics or node degree dynamics, both *extrema* measures and *mean* measures yield the same general behaviors.

The principles that we have identified and described in this research are robust and universal:

We have corroborated our research central hypothesis and have demonstrated dynamics principles that apply across many important dimensions of ecological networks. From a careful review of the literature, one can surmise that these principles may apply to a wide range of systems in other disciplines and subject areas as well. In the course of our work, we have

[123] William Holland Drury Jr., *Chance and Change*, University of California Press, 1998.

encountered additional robust and universal principles. The operational *adjacency matrix pattern* and operational *intensity matrix pattern* that we observe in our ecological system model seems to be a general characteristic – not only of ecosystems, but also of many real-world natural systems. Our work here also suggests that the *network connectivity sparseness* that we observe in our model results is a general characteristic of effective ecological networks and, apparently (from the literature), of natural networks across disciplines.

When observing the total behavioral picture, we see a general equivalence – a universality – in ecological network dynamics. Our broad set of dynamics results all exhibit fundamentally the same form of behavioral statistics. These behaviors and the principles they represent apply across space and time – across network parameters, processes, and phenomena – and, it seems, they may also apply to a wide range of systems across other disciplines and subject areas as well. These are emerging universal principles.

REFERENCES

1. Albert and Barabási, *Statistical Mechanics of Complex Networks*, Reviews of Modern Physics, Volume 74, January 2002, pp. 47–97.

2. Albert et al, *Error and Attack Tolerance of Complex Networks*, Nature, Volume 406, July 2000, pp. 378–382.

3. Alexander, C., *Notes on the Synthesis of Form*, Harvard University Press, 1964.

4. Alexander, C., *The Timeless Way of Building*, Oxford University Press, 1979.

5. Bak, Per, *How Nature Works – The Science of Self-Organized Criticality*, Copernicus (Springer-Verlag), New York, 1996.

6. Bak and Sneppen, *Punctuated Equilibrium and Criticality in a Simple Model of Evolution*, Physical Review Letters, Volume 71, December 1993, pp. 4083–4086.

7. Bak et al, *Self-Organized Criticality: An Explanation of 1/f Noise*, Physical Review Letters, Volume 59, July 1987, pp. 381–384.

8. Barabási, Albert-Laszlo, *Linked: The New Science of Networks*, Perseus Publishing, Cambridge, MA, 2002.

9. Barabási and Albert, *Emergence of Scaling in Random Networks*, Science, New Series, Volume 286, October 1999, pp. 509–512.

10. Bar-Yam, Yaneer, *Dynamics of Complex Systems*, Westview Press, 2003.

11. Bar-Yam, Yaneer, *Making Things Work – Solving Complex Problems in a Complex World*, NECSI Knowledge Press, 2004.

12. Bar-Yam and Epstein, *Response of Complex Networks to Stimuli*, Proceedings of the National Academy of Sciences of the USA, Volume 101, March 2004, pp. 4341–4345.

13. Bohm, David, *Wholeness and the Implicate Order,* Ark Paperbacks, London, 1983.

14. Bonacich, Phillip, *Cellular Automata for the Network Researcher*, UCLA Department of Sociology, June 10, 2002.

15. Broder et al, *Graph Structure in the Web*, Computer Networking, Volume 33, 2000, pp. 309–320.

16. Buede, Dennis M., *The Engineering Design of Systems: Models and Methods*, Wiley, 2000.

17. Butterfield, Herbert, *The Origins of Modern Science*, Macmillan, New York, 1960.

18. Callaway, Newman, Strogatz, and Watts, *Network Robustness and Fragility: Percolation on Random Graphs*, Physical Review Letters, Volume 85, 2000, pp. 5468–5471.

19. Cancho and Solé, *Optimization in Complex Networks*, Santa Fe Institute, Working Paper 01-11-068, 2001.

20. Capra, Fritjof, *The Web of Life*, Anchor Books Doubleday, New York, 1996.

21. Capra, Fritjof, *The Hidden Connections*, Doubleday, New York, 2002.

22. Csermely, Peter, *Weak Links – Stabilizers of Complex Systems from Proteins to Social Networks*, Springer, Berlin Germany, 2006.

23. Dauer, Francis Watanabe, *Critical Thinking*, Oxford University Press, 1989.

24. Dawkins, Richard, *The Blind Watchmaker*, Norton, New York, 1996.

25. Derenyi et al, *Topological Phase Transitions of Random Networks*, Physica A, Volume 334, 2004, pp. 583–590.

26. Dorigo and Gambardella, *Ant Colonies for the Traveling Salesman Problem*, Biosystems, Volume 43, 1997.

27. Drury, William Holland Jr., *Chance and Change*, University of California Press, 1998.

28. Ellson et al, *Graphviz and Dynagraph - Static and Dynamic Graph Drawing Tools* – in Junger, M. and Mutzel, P., editors, *Graph Drawing Software*, Springer-Verlag, 2003, pp. 127–148.

29. Erdös and Rényi, *On the Evolution of Random Graphs*, Publ. Math. Inst. Hung. Acad. Sci., Volume 5, 1960, pp. 17–61.

30. Evans, J. R., *Creative Thinking in the Decision and Management Sciences*, South-Western Publishing, Cincinnati, 1991.

31. Feigenbaum, *Quantitative Universality for a Class of Nonlinear Transformations*, Journal of Statistical Physics, Volume 19, 1978, pp. 25–52.

32. Flood, R. L., and Carson, E. R., *Dealing with Complexity*, Plenum Press, New York, 1993.

33. *Fractal Geometry*, Yale University at http://classes.yale.edu/fractals/.

34. Gharajedaghi, J., *Systems Thinking – Managing Chaos and Complexity*, Butterworth-Heinemann, Boston, MA, 1999.

35. Gould and Eldredge, *Punctuated Equilibria: The Tempo and Mode of Evolution Reconsidered*, Paleobiology, Volume 3, Spring 1977, pp. 115–151.

36. Gould, S. J., *The Hedgehog, the Fox, and the Magister's Pox*, Harmony Books, New York, 2003.

37. Granovetter, *The Strength of Weak Ties*, The American Journal of Sociology, Volume 78, May 1973, pp. 1360–1380.

38. Gribbin, John, *Deep Simplicity: Bringing Order to Chaos and Complexity*, Random House, 2005.

39. Halley, *Ecology, Evolution and 1/f-noise*, Trends in Ecology and Evolution, Volume 11, January 1996, pp. 33–37.

40. Holland, John H., *Hidden Order – How Adaptation Builds Complexity*, Helix Books, New York, 1996.

41. Ivanov, Goldberger, et al, *Scaling Behavior of Heartbeat Intervals obtained by Wavelet-Based Time-Series Analysis*, Nature, Volume 383, 1996, pp. 323–327.

42. Jacob, Francois, *Evolution and Tinkering*, Science, New Series, Volume 196, June 1977, pp. 1161–1166.

43. Jeong et al, *The Large-Scale Organization of Metabolic Networks*, Nature, Volume 407, October 2000, pp. 651–654.

44. Ji, L., Peng, K., and Nisbett, R. E.**,** *Culture, Control, and Perception of Relationships in the Environment*, Journal of Personality and Social Psychology, Volume 78, 2000, pp. 943–955.

45. Kauffman, Stuart A., *The Origins of Order: Self Organization and Selection in Evolution*, Oxford University Press, New York, 1993.

46. Kauffman, Stuart A., *At Home in the Universe – The Search for Laws of Self-Organization and Complexity*, Oxford University Press, New York, 1995.

47. Kauffman, Stuart A., *Investigations*, Oxford University Press, New York, 2002.

48. Kazanci, C. and Tollner, E. W., *EcoNet, A New Software for Ecological Model Simulation and Network Analysis*, Ecological Modeling, Volume 208, 2007, pp. 3–8.

49. Klir, George J., *Complexity: Some General Observations*, Systems Research, Volume 2, 1985, pp. 131–140.

50. Klir, George J., *Facets of Systems Science*, Plenum Press, New York, 1991.

51. Krugman, Paul, *The Self-Organizing Economy*, Blackwell Publishers, Cambridge MA, 1996.

52. Kuhn, Thomas S., *The Structure of Scientific Revolutions*, Third Edition, The University of Chicago Press, 1996.

53. Lack, David, *The Natural Regulation of Animal Numbers*, Clarendon Press, Oxford, 1954.

54. Laszlo, Ervin, *The Systems View of the World*, G. Braziller, New York, 1972.

55. Mandelbrot, Benoit B., *The Fractal Geometry of Nature*, W.H. Freeman, San Francisco, 1990.

56. MATLAB release R2009a, The MathWorks, Inc., February 12, 2009.

57. May, Robert M., *Biological Populations with Non-Overlapping Generations: Stable Points, Stable Cycles, and Chaos*, Science, Volume 186, 1974, pp. 645–647.

58. May, Robert M., *Simple Mathematical Models with Very Complicated Dynamics*, Nature, Volume 261, 1976, pp. 459–467.

59. Merriam-Webster Online, http://www.merriam-webster.com/.

60. Milgram, *The Small World Problem*, Psychology Today, Volume 2, 1967, pp. 60–67.

61. Mitchell, Melanie, *Complexity: A Guided Tour*, Oxford University Press, 2009.

62. Molofsky and Bever, *A New Kind of Ecology*, BioScience, Volume 54, May 2004, pp. 440-446.

63. Montoya and Solé, *Small World Patterns in Food Webs*, Santa Fe Institute, Working Paper 00-10-059, 2000, 6 pages.

64. Müller, Felix, *Emergent Properties of Ecosystems – Consequences of Self-Organizing Processes?*, Senckenbergiana Maritima, Volume 27, 1996, pp. 151–168.

65. Naeem, Shahid and Golley, Frank B. et al, *Biodiversity and Ecosystem Functioning*, Ecological Society of America, Issues in Ecology, Number 4, Fall 1999, 12 pages (http://www.epa.gov/watertrain/pdf/issue4.pdf).

66. Newman, *The Structure and Function of Complex Networks*, SIAM Review, Volume 45, May 2003, pp. 167–256.

67. Newman and Watts, *Scaling and Percolation in the Small-World Network Model*, Physical Review E, Volume 60, December 1999, pp. 7332–7342.

68. Nicolis & Prigogine, *Exploring Complexity*, W. H. Freeman and Company, New York, 1989.

69. Nisbett, Richard, *The Geography of Thought: How Asians and Westerners Think Differently ... and Why*, Free Press, New York, March 3, 2003.

70. Odum, Eugene P. and Barrett, Gary W., *Fundamentals of Ecology* Fifth Edition, Thomson Brooks/Cole, Belmont, CA, 2005.

71. Odom, Howard T., *Environment, Power, and Society*, Wiley-Interscience, New York, 1971.

72. Oppenheimer, J. Robert, "A Science in Change," in *Science and the Common Understanding*, Simon and Schuster, New York, 1954.

73. *Pajek – Program for Large Network Analysis*, http://vlado.fmf.uni-lj.si/pub/networks/pajek/.

74. Patten, B. C. and Odum, E. P., *The Cybernetic Nature of Ecosystems*, The American Naturalist, Volume 118, 1981, pp. 886–895.

75. *Philosophy and Design – From Engineering to Architecture*, edited by Vermaas et al, Part 2, chapter by Kristo Miettinen, Springer Netherlands, 2008.

76. Popper, Karl R., "Heroic Science," in Bolles (editor), *Galileo's Commandment*, Henry Holt & Company, 1999.

77. Popper, Karl R., *Realism and the Aim of Science*, Rowman and Littlefield, Totowa, NJ, 1983 edition.

78. Rechtin and Maier, *The Art of Systems Architecting*, CRC Press, 1997.

79. Roughgarden, J., *Theory of Population Genetics and Evolutionary Ecology: An Introduction*, Macmillan Publishing Company, New York, 1979.

80. Schiff, Joel, *Cellular Automata – A Discrete View of the World*, Wiley-Interscience, 2008.

81. Schneider, Eric D. and Kay, James J., "Order from Disorder: The Thermodynamics of Complexity in Biology," in Murphy and O'Neill (editors), *What is Life: The Next Fifty Years. Reflections on the Future of Biology*, Cambridge University Press, 1995, pp. 161-172.

82. Schön, Donald A., *The Reflective Practitioner*, Basic Books, New York, 1983.

83. Shannon, Claude E., *A Mathematical Theory of Communication*, Bell System Technical Journal, Volume 27, July & October 1948, pp. 379–423 & 623–656.

84. Simon, Herbert A., *On a Class of Skew Distribution Functions*, Biometrika, Volume 42, 1955, pp. 425–440.

85. Simon, Herbert A., *The Architecture of Complexity*, Proceedings of the American Philosophical Society, Volume 106, December 1962, pp. 467–482.

86. Simon, Herbert A., *How Complex Are Complex Systems?*, Proceedings of the Biennial Meeting of the Philosophy of Science Association, Volume 2, 1976, pp. 507–522.

87. Simon, Herbert A., *Discovery, Invention, and Development: Human Creative Thinking*, Proceedings of the National Academy of Sciences of the USA, Volume 80, July 1983, pp. 4569–4571.

88. Simon, Herbert A., *The Sciences of the Artificial*, third edition, The MIT Press, 1996.

89. Solé, Ricard V. and Bascompte, Jordi, *Self-Organization in Complex Ecosystems*, Princeton University Press, 2006.

90. Solé and Goodwin, *Signs of Life – How Complexity Pervades Biology*, Basic Books (Perseus Books Group), New York, 2000.

91. Solé and Montoya, *Complexity and Fragility in Ecological Networks*, Proceedings of the Royal Society of London B, Volume 268, October 2001, pp. 2039–2045.

92. Solé and Valverde, *Spontaneous Emergence of Modularity in Cellular Networks*, Santa Fe Institute, Working Paper 07-06-013, 2007.

93. Solé et al, *Criticality and Scaling in Evolutionary Ecology*, Trends in Ecology and Evolution, Volume 14, April 1999, pp. 156–160.

94. Solé et al, *Selection, Tinkering, and Emergence in Complex Networks*, Complexity, Volume 8, 2003, pp. 20–33.

95. Song, Gallos, Havlin, and Makse, *How To Calculate the Fractal Dimension of a Complex Network*, Journal of Statistical Mechanics, March 2007, 16 pages.

96. Song, Havlin, and Makse, *Self-Similarity of Complex Networks*, Nature, Volume 433, January 2005, pp. 392–395.

97. Strogatz, *Exploring Complex Networks*, Nature, Volume 410, March 2001, pp. 268–276.

98. Sullivan, Louis H., *The Tall Office Building Artistically Considered*, Lippincott's Magazine, March 1896.

99. Taleb, Nassim, *The Black Swan*, Random House, New York, 2007.

100. The MathWorks website, http://www.mathworks.com.

101. Thompson, D'Arcy W., *On Growth and Form*, 1992 Dover reprint of 1942 2nd ed. (1st ed. was in 1917).

102. Thoreau, Henry David, *The Heart of Thoreau's Journals*, edited by Odell Shepard, Houghton Mifflin, Boston, 1927.

103. Ulanowicz, Robert E., *Ecology, the Ascendant Perspective*, Columbia University Press, 1997.

104. Valverde, Cancho, and Solé, *Scale-Free Networks from Optimal Design*, Europhysics Letters, Volume 60, 2002, pp. 512–517.

105. Vicsek, Tamas, *Complexity: The Bigger Picture*, Nature, Volume 418, July 2002, p. 131.

106. Vishwanathan et al, *Optimizing the Success of Random Searches*, Nature, Volume 401, October 1999, pp. 911–914.

107. Watts and Strogatz, *Collective Dynamics of 'Small-World' Networks*, Nature, Volume 393, June 1998, pp. 440–442.

108. Weaver, Warren, *Science and Complexity*, American Scientist, Volume 36, 1948, p. 536.

109. Weinberg, Gerald M., *An Introduction to General Systems Thinking*, Wiley, 1975.

110. Weiss, Paul A., *Dynamics of Development*, Academic Press, New York, 1968.

111. Wesson, R. G., *Beyond Natural Selection*, MIT Press, 1991.

112. Wilensky, U., *NetLogo Wolf Sheep Predation Model*, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 2005. http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation(docked)

113. Wilson, E. O., *Consilience: The Unity of Knowledge*, Vintage Books, New York, 1999.

114. Wolfram, Stephen, *A New Kind of Science*, Wolfram Media, May 14, 2002.

115. Yackinous, *System Modeling – with an Application to Ecological Theory Development*, ECOL 8000 paper, December 6, 2005 (unpublished).

116. Yackinous, *A Comparison of Ecological Modeling and Engineering Modeling: Opportunities for Improving the Current State of Ecological Modeling*, ENGR 8980 paper, April 15, 2006 (unpublished).

117. Yackinous, *PhD Research Proposal: Toward a New Framework for Ecological System Theory*, April 14, 2007 (unpublished).

118. Yackinous, *The Structure of Complex Networks*, literature review, November 5, 2007 (unpublished).

119. Yackinous, *Fundamentals of Nonlinear Dynamics*, literature review, November 27, 2007 (unpublished).

120. Yackinous, *The Dynamics of Complex Networks*, literature review, March 25, 2008 (unpublished).

121. Yackinous, *Dissertation Prospectus – Emerging Principles of Ecological Network Dynamics*, November 30, 2008.

122. Yackinous, *Control Aspects of Ecological Systems*, March 25, 2009 (unpublished).

123. Yackinous, *Evolution and Universal Development Concepts*, March 25, 2009 (unpublished).

124. Yackinous, *Reductionism and Information Loss in Ecological Investigation*, March 25, 2009 (unpublished).

125. Yackinous, *Structure, Function, and Process: Some Needed Clarity*, March 25, 2009 (unpublished).

126. Yackinous, *Thermodynamic Entropy and Order in Ecological Systems*, March 25, 2009 (unpublished).

127. Yale University, *Fractal Geometry*, at http://classes.yale.edu/fractals/.

128. Yang, Xin-She, *An Introduction to Computational Engineering with Matlab*, Cambridge International Science Publishing, 2006.

APPENDIX A

REVIEW OF NETWORK LITERATURE

THE STRUCTURE AND DYNAMICS OF COMPLEX NETWORKS

This effort includes the most recent perspectives on complex system networks and the resulting implications for ecological systems.

I have written a series of three informal papers on the structure and dynamics of complex networks:

- Yackinous, *The Structure of Complex Networks*, November 5, 2007.
- Yackinous, *Fundamentals of Nonlinear Dynamics*, November 27, 2007.
- Yackinous, *The Dynamics of Complex Networks*, March 25, 2008.

These papers document my thorough literature review of these three areas. (They also serve as a vehicle for me to express my views on the subjects.) I will not repeat the full content of the papers here. I'll provide some brief introductory comments and a listing of the contents for each of the three papers. Please refer to the complete papers for additional detail. They are available from the author.

The Structure of Complex Networks

We begin this paper by describing the fundamental concepts and properties that characterize the structure of networked systems. A primary underlying goal of the series of papers is to increase our understanding of real systems (particularly ecological systems). Therefore, we next take a comprehensive look at empirical data from real networks – with an emphasis on properties that are common to many of them. We then consider a theoretical view: we review mathematical developments relevant to complex system networks and their properties.

A listing of the paper's content follows:

1. Introduction
2. Concepts, Characteristics, and Definitions
    2.1 Random and Non Random Networks
    2.2 Important Network Properties
        2.2.1 Path Length
        2.2.2 Clustering
        2.2.3 Degree Distribution
    2.3 Small-World Characteristics
    2.4 Scale-Free Characteristics
    2.5 Weak Link Characteristics
3. Empirical View – Real Networks
4. Theoretical View – Mathematics of Model Networks
    4.1 Poisson Random Graphs
        4.1.1 Poisson Degree Distribution
        4.1.2 Transition to a Giant Component
        4.1.3 Other Random Graph Properties
    4.2 Generalized Random Graphs and Scale-Free Networks
    4.3 Small-World Networks
5. Next Steps

Fundamentals of Nonlinear Dynamics

Before tackling the combination of complex network structure and complex network dynamics, we need an introductory understanding of the (nonlinear) dynamics of more basic systems. In this paper, therefore, we explore the dynamics of single node systems and the dynamics of very small networks consisting of only a few nodes. After thereby gaining some understanding of nonlinear dynamics of systems at that level, we move on to consider the dynamics of complex networks in the next (third) paper in the series.

This paper first introduces fundamental concepts of nonlinear dynamical systems: rate equations, state space, and attractors. These concepts are then applied to a simple example (to help ease us into our nonlinear dynamics investigations). Next, we get into a comprehensive description of attractors – with interesting examples and illustrations. Following that, we address the phenomenon of bifurcation which plays an important role in our understanding of the

complex nonlinear dynamics of ecological systems. We then explain dependence on initial conditions and the resulting unpredictability of nonlinear systems. Finally, we discuss the fact that much system behavior seems to be general – universal – and independent of the details of a given specific system.

A listing of the paper's content follows:

1. Introduction
2. Fundamental Concepts
   2.1 Rate Equations
   2.2 State Space
   2.3 Attractors
3. Starting Simple
4. Types of Attractors
5. Bifurcation and Complex Nonlinear Behavior
6. Unpredictability
7. Universality
8. Next Steps

The Dynamics of Complex Networks

The first paper in this series of papers on the structure and dynamics of complex networks addressed network structure. The second paper in the series introduced and investigated the fundamentals of nonlinear dynamics. This third (and final) paper in the series tackles the integration of these two areas of study. Here, we explore the complex nonlinear dynamics of structurally complex networks. Our goal is a more complete understanding of real-world complex networks – and ecological networks in particular.

Our understanding of the dynamics of networked systems is nascent. In this paper, we discuss what is known and what is hypothesized. Some of the hypotheses may conflict with our intuition and our long-held views. They will challenge our usual linear way of thinking (because complex system dynamics are often nonlinear).

Early on, this paper addresses scaling, power laws, and fractals. After establishing an equivalence among these three concepts, we explore both spatial and temporal fractals in some detail. A so-called *critical* state appears to be an attractor for complex network dynamics. Then we discuss self-organized criticality and pertinent associated concepts. The next topic is the dynamics of network phase transitions. We consider network percolation as well as other phase transitions. Following that, we look at examples of processes that take place on networks – from epidemic processes to network growth processes and more. Finally, we provide a brief commentary on ecosystem management as it relates to the network dynamics that we have discussed in this paper.

Here's the listing of the paper's content:

1. Introduction
    - 1.1 The Series of Papers on Complex Networks
    - 1.2 The Importance of Dynamics
    - 1.3 Summary of the Document
2. Scaling, Power Laws, and Fractals
    - 2.1 Equivalence
    - 2.2 Fractals in Space and Time
        - 2.2.1 Spatial Fractals
        - 2.2.2 Temporal Fractals
3. Self-Organized Criticality
    - 3.1 Bak's Self-Organized Criticality Hypothesis
    - 3.2 An Example of Self-Organized Criticality
    - 3.3 Local and Global Processing on Networks
        - 3.3.1 Scale-Free and Small-World Networks
        - 3.3.2 Adaptation – Life as a Relaxation Phenomenon
4. Percolation and Other Network Phase Transitions
    - 4.1 Percolation in Networks
    - 4.2 Other Topological Phase Transitions in Networks
5. Processes Taking Place on Networks
    - 5.1 Epidemic Processes on Networks
    - 5.2 Network Failure/Attack Processes
    - 5.3 Macroevolution Processes
    - 5.4 Discrete Dynamical Processes on Networks
    - 5.5 Network Growth via Preferential Attachment
6. A View of Ecosystem Management
7. Concluding Remarks

APPENDIX B

INTRODUCTION TO CELLULAR AUTOMATA MODELING

Let's begin with some perspectives on cellular automata modeling. Xin-She Yang[124] says: "A cellular automaton (CA) is a rule-based computing machine, which was first proposed by von Neumann in the early 1950s and the systematic studies were pioneered by Wolfram from the 1980s. Since a cellular automaton consists of space and time, it is essentially equivalent to a dynamical system that is discrete in both space and time. The evolution of such a discrete system is governed by certain updating rules rather than differential equations. Although the updating rules can take many different forms, most common cellular automata use relatively simple rules." Phillip Bonacich[125] comments on cellular automata usage in network research: Cellular automata (CA) can be used in simulations of network processes and network evolution by identifying adjacent vertices in a network with neighboring cells in a cellular automaton.

Note that a cellular automata model can be used to model ecological systems for which it is difficult or infeasible to solve or even formulate the system differential equations of traditional ecological models. A well-known example can serve to introduce cellular automata modeling.

---

[124] Xin-She Yang, *An Introduction to Computational Engineering with Matlab*, Cambridge International Science Publishing, 2006.

[125] Phillip Bonacich, *Cellular Automata for the Network Researcher*, UCLA Department of Sociology, June 10, 2002.

The Sandpile Cellular Automaton

Take a look at the *sandpile* depicted in Figure B1.[126]

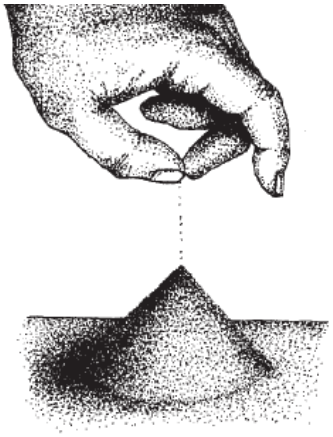


Figure B1 The Sandpile Metaphor

As suggested by the figure we add sand slowly, one grain at a time. At first, the grains stay where they land. These initial grains are basically independent and their behavior is described solely by gravity and friction forces. But, as the sand pile gets larger and its slope increases, we reach a regime where avalanches involving grains in interaction are occurring all the time. At this critical state we no longer have independent grains, but rather one complex system with its own emergent dynamics. This new state cannot be anticipated from the properties of the individual parts.

The sandpile is a metaphor for a network system that can be modeled as a cellular automaton. The sandpile resides on a grid. The grid cells are the network nodes, the cell interactions are the network links, and the grains of sand are the network flow currency. Figure B2 illustrates the sandpile cellular automaton model.[127]

---

[126] The diagram and its description are from Solé et al, *Criticality and Scaling in Evolutionary Ecology*, Trends in Ecology and Evolution, April 1999.

[127] The diagram is from Solé and Goodwin, *Signs of Life – How Complexity Pervades Biology*, Basic Books, 2000.

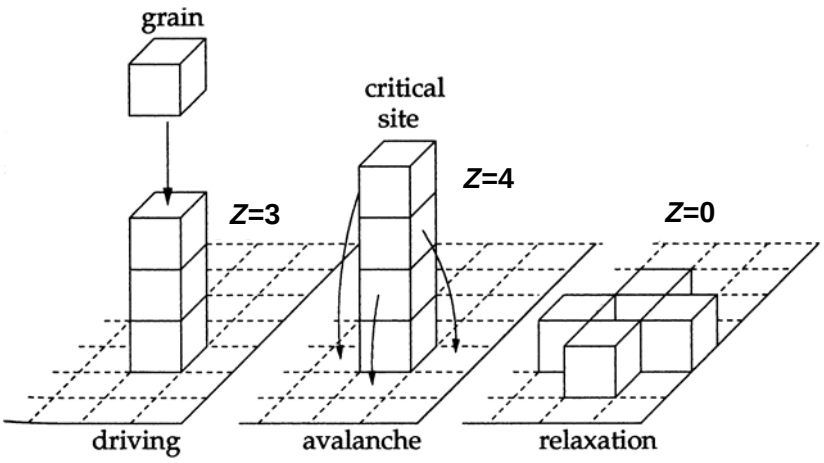


Figure B2 The Sandpile Cellular Automaton Model

Here's a mathematical description of the model.[128] The sandpile resides on a two-dimensional grid (*N*x*N*) with cells having coordinates (*x*,*y*). The state *Z*(*x*,*y*) of any cell is a number from 0 to 4:

$$Z(x,y)\,\varepsilon\,\{0,1,2,3,4\}$$

To apply input to the system, choose a cell randomly and increase *Z* by 1:

$$Z(x,y)\rightarrow Z(x,y)+1$$

Repeat the input process at each model time step (iteration step). Grains of sand are propagated through the network via a propagation rule, e.g.:

$$\begin{aligned}
&If\ Z(x,y)>Z_{crit}\quad (say\ Z_{crit}=3)\\
&Then\ propagate\ 1\ unit\ to\ each\ of\ four\ neighbors:\\
&Z(x,y)\rightarrow Z(x,y)-4\\
&Z(x\pm1,y)\rightarrow Z(x\pm1,y)+1\\
&Z(x,y\pm1)\rightarrow Z(x,y\pm1)+1\\
&If\ x=1\ or\ N\quad and/or\quad y=1\ or\ N\quad (at\ system\ boundary)\\
&Then\ units\ leave\ the\ system\ (output)
\end{aligned}$$

As propagation progresses, *avalanches* can and do occur. The size of an avalanche equals the number of cascading propagation events that occur at a given time step.

[128] Per Bak, *How Nature Works – The Science of Self-Organized Criticality*, Springer-Verlag, 1996.

Here's a summary of model results. The sandpile (network) operational "evolution" proceeds as follows:

At first there are no avalanches.

After many time steps we get a small avalanche.

Eventually we get a medium avalanche (and more small ones).

Eventually we get a large avalanche (and more small and medium ones).

Small, medium, and large (scale-invariant/fractal) avalanches continue indefinitely.

For this model, Bak et al[129] have measured the scaling exponent of the power-law avalanche event distribution as ≈ 1.1.

Figures B3 and B4 illustrate the sandpile cellular automata model results in graphical form.[130] Figure B3 shows the avalanche event distribution.

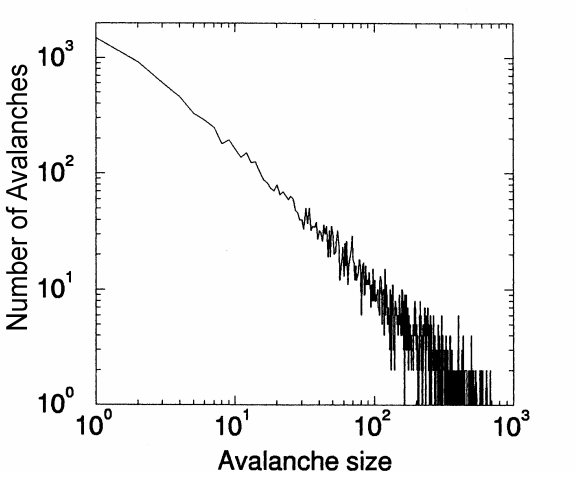


Figure B3 Avalanche Event Distribution

The log-log plot approximates a straight line (power-law/scale-invariant/fractal behavior). Figure B4 provides the avalanche event time series.

[129] Bak et al, *Self-Organized Criticality: An Explanation of 1/f Noise*, Physical Review Letters, Volume 59, July 1987.

[130] Both diagrams are from Solé and Goodwin, *Signs of Life*, Basic Books, 2000.

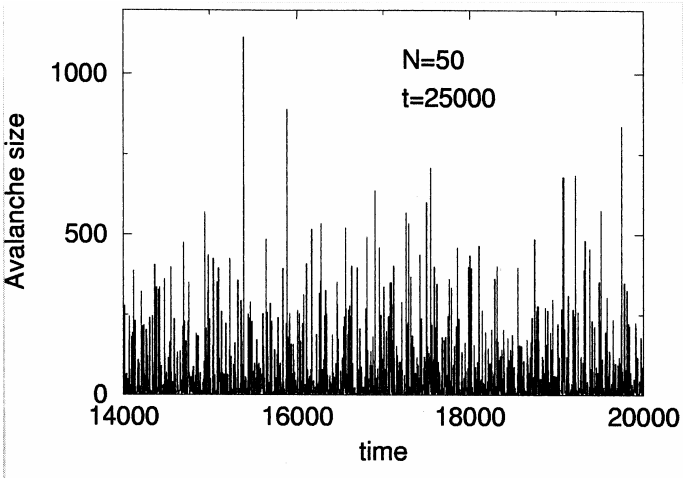


Figure B4 Avalanche Event Time Series

The parameter N=50 is the grid dimension (NxN grid) and parameter t=25000 is the total number of time steps in the simulation run. The dynamics are clearly punctuated.

Finally, here are some observations on the cellular automata model results:

The results of the simple model are very complex.

Domain of attraction dynamics create fractals:

- Avalanche size distribution is scale-invariant and fractal in space.
- Avalanche time series is 1/f and fractal in time.

Before reaching criticality, the response to small perturbations is small ⇨ approximates linear behavior.

At criticality, the response to small perturbations could be very large ⇨ nonlinear behavior.

Apparently one cannot derive sandpile criticality results analytically. It has been attempted many times without success. We need to use modeling/simulation.

APPENDIX C

GLOSSARY OF MATLAB VARIABLES BY CATEGORY

Network propagation takes place in *stages* within *time steps* within a simulation run. We use the following conventions for naming MATLAB variables.

Variable suffixes (lower case preceded by an underbar):
Variable_s = variable that applies to an individual stage within a time step
Variable_sc = variable that is cumulative over stages within a time step
Variable_t = variable that applies to an individual time step within a simulation run
Variable_tc = variable that is cumulative over time steps within a simulation run

Variable extensions (upper case preceded by an underbar):
These extend variables over stages and over time steps. Each extension effectively adds one dimension to a variable.
Variable_S = variable per stage (provides values for each stage in a time step)
Variable_TS = variable per time step (provides values for each time step in a simulation run)
Variable_S_TS = variable per stage per time step (provides values for each stage in a time step and each time step in a simulation run)

Suffixes and extensions can be used in combination.

Underlying ecological network compartment model variables:
A = compartment model adjacency matrix (10x10) with ones to identify allowable compartment connections and zeros otherwise.
zL = compartment model input logical vector (10x1) with ones to identify input compartments and zeros otherwise.
yL = compartment model output logical vector (1x10) with ones to identify output compartments and zeros otherwise.

Model network grid variables:
NNG = node-number grid matrix (10x10). Nodes are numbered 1 to 100 sequentially – from grid column to column (single index).
NCG = node-compartment grid matrix (10x10) in which the elements identify a node's home compartment number.
NHCL = node-to-home-compartment logical matrix (10x100) with a one to identify a node's home compartment number and zeros otherwise.
NCCL = node-to-connecting-compartment logical matrix (10x100) with ones to identify the connecting (connect-to) compartments and zeros otherwise.
QNinC = quantity of nodes in each compartment (10x1).

NCoord = node-coordinate matrix (100x2) in which each row provides a spatial coordinate pair that represents one node.

Number of model simulation time steps variable:
NumTS = number of model simulation time steps (1x1). The number should be on the order of 1000.

Node stock and node propagation rules variables:
thd = the node stock threshold for propagation (1x1).
npfq = the node propagation flow quantity (1x1).
The relationship ($npfq \leq thd$) is required. The value ($thd - npfq$) is the node stock remaining after a node propagation event.

Variables for model network inputs and outputs and their values:
input_nodes = model network input-node column vector that provides the single index of each input node.
INL = input-node logical grid matrix (10x10) with ones representing input nodes and zeros representing non-input nodes.
output_nodes = model network output-node column vector that provides the single index of each output node.
ONL = output-node logical grid matrix (10x10) with ones representing output nodes and zeros representing non-output nodes.
IOL = input/output-node logical matrix (2x100) with row 1 ones to identify input nodes, row 2 ones to identify output nodes, and zeros otherwise.
INVG_tc = input-node-value-grid matrix (10x10) with values (0, 1, 2, …) in which each element represents the cumulative value of the input to that node.
INVG_tc_TS = input-node-value-grid multidimensional array (10x10xNumTS) with values (0, 1, 2, …) in which each element represents the cumulative value of the input to that node at the specified time step.
ONVG_tc = output-node-value-grid matrix (10x10) with values (0, 1, 2, …) in which each element represents the cumulative value of the output from that node.
ONVG_tc_TS = output-node-value-grid multidimensional array (10x10xNumTS) with values (0, 1, 2, …) in which each element represents the cumulative value of the output from that node at the specified time step.
ICV_tc = input-compartment-value vector (1x10) with values (0, 1, 2, …) in which each element represents the cumulative value of the input to that compartment.
ICV_tc_TS = input-compartment-value multidimensional vector (1x10xNumTS) with values (0, 1, 2, …) in which each element represents the cumulative value of the input to that compartment at the specified time step.
OCV_tc = output-compartment-value vector (1x10) with values (0, 1, 2, …) in which each element represents the cumulative value of the output from that compartment.
OCV_tc_TS = output-compartment-value multidimensional vector (1x10xNumTS) with values (0, 1, 2, …) in which each element represents the cumulative value of the output from that compartment at the specified time step.
SIV_tc = system-input-value scalar with values (0, 1, 2, …) which represents the cumulative value of the input to all compartments (or all nodes).

SIV_tc_TS = system-input-value vector (1xNumTS) with values (0, 1, 2, …) which represents the cumulative value of the input to all compartments (or all nodes) at a specified time step. Since a unit input is applied at each simulation time step, SIV_tc_TS is incremented by one each time step.
SOV_tc = system-output-value scalar with values (0, 1, 2, …) which represents the cumulative value of the output from all compartments (or all nodes).
SOV_tc_TS = system-output-value vector (1xNumTS) with values (0, 1, 2, …) which represents the cumulative value of the output from all compartments (or all nodes) at a specified time step.

Variables for model network stock values:
NSVG_tc = node-stock-value-grid matrix (10x10) with values (0, 1, 2, … , thd) in which each element represents the cumulative value of the stock of that node.
NSVG_tc_TS = node-stock-value-grid multidimensional array (10x10xNumTS) with values (0, 1, 2, … , thd) in which each element represents the cumulative value of the stock of that node at the specified time step.
CSV_tc = compartment-stock-value vector (1x10) with values (0, 1, 2, …) in which each element represents the cumulative value of the stock of that compartment.
CSV_tc_TS = compartment-stock-value multidimensional vector (1x10xNumTS) with values (0, 1, 2, …) in which each element represents the cumulative value of the stock of that compartment at the specified time step.
SSV_tc = system-stock-value scalar with values (0, 1, 2, …) which represents the cumulative value of the total system stock, i.e., the cumulative value of the stock of all compartments (or all nodes).
SSV_tc_TS = system-stock-value vector (1xNumTS) with values (0, 1, 2, …) which represents the cumulative value of the total system stock, i.e., the cumulative value of the stock of all compartments (or all nodes) – at a specified time step.

Compartment selection order variables:
Each node has candidate connecting compartments. When the node propagates, we need a randomly generated compartment selection order for these connecting compartments so that we can simulate compartment preference and selection of destination. Variables QNCC and CSO satisfy the need.
QNCC = quantity-of-node-connecting-compartments row vector (1x100).
Each element of QNCC represents the quantity of the corresponding node's connect-to compartments.
CSO = compartment-selection-order matrix (10x100) in which each column's non-zero elements provide the selection order of the corresponding node's connect-to compartments.

Variables for network node adjacency matrices:
AA = network node adjacency matrix (100x100).
This is the basic network node adjacency matrix with ones to identify allowable node connections and zeros otherwise.
AAT = adjacency adjacency-type matrix (100x100) with element values (1, 2, 3).
This node "adjacency matrix" provides the adjacency type of connect-to nodes: a local neighbor is type 1, an extended local neighbor is type 2, a global neighbor is type 3. Adjacency type is part of the node *connection priority order*.

ACN = adjacency compartment-number matrix (100x100) with element values (1 to 10).
This node "adjacency matrix" provides the home compartment number of connect-to nodes. Compartment number is part of the node *connection priority order* within each adjacency type.
AAP = adjacency attachment-preference-strength matrix (100x100) (element values: 0 to 1).
This node "adjacency matrix" provides the node attachment preference strength of each connect-to node so that we can implement our preferential attachment concept. Attachment preference is part of the node *connection priority order* within adjacency type and compartment number.
NAPS = node-to-attachment-preference-strength vector (1x100) (element values: 0 to 1).
Row vector NAPS provides each connect-to node's randomly generated attachment preference strength. It is used to generate adjacency matrix AAP.

Variables for network node "operational" adjacency matrices and arrays:
These three variables provide adjacency matrices and arrays which represent the changing state of the network as it operates and develops. We refer to these as "operational" adjacency matrices/arrays. The "O" in the following variable names is for "operational."
AAO_t = cumulative operational node adjacency matrix for an individual time step (100x100).
This matrix applies to an individual time step. AAO_t is cumulative over all node propagation instances over all stages of the time step. The matrix is updated at every propagation instance within a stage.
AAO_t_TS = per-time-step operational node adjacency multidimensional array (100x100x NumTS).
This array provides the completed cumulative operational adjacency matrix AAO_t for each individual time step. AAO_t_TS(:, :, k) = completed AAO_t for time step k.
AAO_tc_TS = per-time-step over-time-cumulative operational node adjacency multidimensional array (100x100x NumTS).
This array consists of cumulative adjacency matrices. Its adjacency matrices are cumulative over time (over the time steps in the simulation run) and the array is updated at every time step. AAO_tc_TS(:, :, k) = AAO_tc_TS(:, :, k-1) + AAO_t_TS(:, :, k). In this adjacency matrix sum, if any element > 1 then element = 1.

Variables for network flow value "adjacency" multidimensional arrays:
ANFV_tc_TS = adjacency node-flow-value multidimensional array (100x100xNumTS) with values (0, 1, 2, …).
This "adjacency" multidimensional array provides the over-time cumulative values of the direct flows between adjacent nodes. The matrices of the multidimensional array are cumulative over the simulation run. The array is updated as flows occur (in time-step stages) and is saved at every time step.
ACFV_tc_TS = adjacency compartment-flow-value multidimensional array (10x10xNumTS) with values (0, 1, 2, …).
This "adjacency" multidimensional array provides the over-time cumulative values of the direct flows between adjacent *compartments*. The matrices of the multidimensional array are cumulative over the simulation run. The array is updated as flows occur (in time-step stages) and is saved at every time step.

Basic propagation process variables:
PENodeS = propagation-event-node-set column vector (temporary variable).
NumCStages = current number of completed propagation stages (scalar) (temporary variable).
NumCStages_TS = per-time-step row vector of the total number of propagation stages in each individual completed time step (1xNumTS).
AvFlow = the currently available-flow for a propagating from-node (scalar) (temporary variable). When a node propagation event occurs, the starting value of AvFlow is npfq.
FNode = propagating from-node (scalar) (values are 1 to 100) (temporary variable).
TNodeS = candidate to-node-set column vector = candidate propagation-instance-node-set column vector (temporary variable).
TNodeSC = to-node-set-compartments column vector (temporary variable).
TNodeSCR = ranks of to-node-set-compartments column vector (temporary variable).
RTNodeS = revised to-node-set column vector (temporary variable).
RTNodeSAPS = revised to-node-set-attachment-preference-strengths column vector (temporary variable).
RTNodeSAPP = revised to-node-set-attachment-preference-probabilities column vector (temporary variable).
SNode = propagation instance selected-node (scalar) (values are 1 to 100) (temporary variable).

Input node processing variables:
INS_t = input-node-selection row vector for an individual time step (1x100) with randomly assigned selection numbers that are generated anew at each simulation time step.
InputNode_t = selected input-node for an individual time step (scalar) (values are 1 to 100). Value is generated anew at each simulation time step.
INSL_t = input-node-selection-logical matrix for an individual time step (10x10) (values are 0, 1) with a one to identify the selected input node and zeros elsewhere.
INSL_t_TS = per-time-step input-node-selection-logical multidimensional array (10x10xNumTS) with matrix values (0, 1) that provides, for each time step, a matrix with a one to identify the selected input node and zeros elsewhere. Use INSL_t_TS to plot per-time-step network propagation diagrams.
INSL_tc_TS = per-time-step over-time-cumulative input-node-selection-logical multidimensional array (10x10x NumTS) (matrix values are 0, 1). INSL_tc_TS is an over-time cumulative version of INSL_t_TS for plotting the over-time network propagation diagram at each time step.

Propagating node processing variables:
PNL_sc = stage-cumulative propagating-node-logical grid matrix (10x10) (values are 0, 1) that identifies propagating nodes for a single time step. The matrix is cumulative over the stages of the time step and is updated at every time step stage. The matrix has ones to identify the propagating nodes (where the propagation condition is satisfied) and zeros elsewhere.
PNL_t_TS = per-time-step multidimensional array of individual time step (stage-cumulative) propagating-node-logical matrices (10x10xNumTS) (matrix values are 0, 1). PNL_t_TS contains the PNL_sc matrix for each time step. PNL_t_TS(:, :, k) = PNL_sc for time step k. PNL_t_TS is used for plotting network propagation diagrams.
PNL_tc_TS = per-time-step over-time-cumulative propagating-node-logical multidimensional array (10x10x NumTS) (matrix values are 0, 1) that provides matrices that are cumulative over

time (over the simulation run time steps) and are updated at every time step. PNL_tc_TS is used for plotting network propagation diagrams.
PNL_tc_TS(:, :, k) = PNL_tc_TS(:, :, k-1) + PNL_t_TS(:, :, k)
nnpe_s_S_TS = number-of-node-propagation-events multidimensional vector (1x #stages xNumTS) that provides the number of node propagation events (which equals the number of propagating nodes) in each individual propagation stage of a time step for each time step. Multidimensional vector nnpe_s_S_TS is used for path length analysis.
nnpe_sc = stage-cumulative number-of-node-propagation-events (scalar).
nnpe_sc_TS = per-time-step stage-cumulative number-of-node-propagation-events vector (1xNumTS) that provides the cumulative (total) number of node propagation events over stages in a completed time step for each time step. Vector nnpe_sc_TS is a stage-cumulative version of nnpe_s_S_TS.
nopn_s_S_TS = number-of-output-propagating-nodes multidimensional vector (1x #stages xNumTS) that provides the number of output propagating nodes in an individual stage for each propagation stage of each time step. Multidimensional vector nopn_s_S_TS is used for path length analysis.
nopn_sc = stage-cumulative number-of-output-propagating-nodes (scalar).
nopn_sc_TS = per-time-step stage-cumulative number-of-output-propagating-nodes vector (1xNumTS) that provides the cumulative (total) number of output propagating nodes over stages in a completed time step for each time step. Vector nopn_sc_TS is a stage-cumulative version of nopn_s_S_TS.

Output node processing variables:
OPNL_sc = stage-cumulative output-propagating-node-logical matrix (10x10) with values (0, 1) that provides, for a time step, a stage-cumulative matrix with ones to identify the output propagating nodes and zeros elsewhere. The matrix is updated at every time step stage.
OPNL_t_TS = per-time-step multidimensional array (10x10xNumTS) of individual time step stage-cumulative output-propagating-node-logical matrices (matrix values are 0, 1). The array contains the OPNL_sc matrix for each time step. OPNL_t_TS(:, :, k) = OPNL_sc for time step k. OPNL_t_TS is used for plotting network propagation diagrams.
OPNL_tc_TS = per-time-step over-time-cumulative output-propagating-node-logical multidimensional array (10x10x NumTS) (matrix values are 0, 1) that provides matrices that are cumulative over time (over the simulation run time steps) and are updated at every time step. OPNL_tc_TS is used for plotting network propagation diagrams.

Node propagation instance processing variables:
nnpi_sc = stage-cumulative number-of-node-propagation-instances (scalar).
nnpi_sc_TS = per-time-step stage-cumulative number-of-node-propagation-instances vector (1xNumTS) that provides the cumulative (total) number of node propagation instances in an individual completed time step for each time step. Vector nnpi_sc_TS is used for network propagation event analysis.
NetPESize_TS = per-time-step network-propagation-event-size vector (1xNumTS) that provides the size of the network propagation event in a completed time step for each time step. NetPESize_TS(k) = nnpi_sc_TS(k) + 1. NetPESize_TS is used for network propagation event analysis.

Network propagation event analysis variables:
NetPESize_TS = per-time-step network-propagation-event-size vector (1xNumTS) that provides the size of the network propagation event in a completed time step for each time step. NetPESize_TS(k) = nnpi_sc_TS(k) + 1. NetPESize_TS is used for network propagation event analysis.
nsi = network propagation event size-interval (scalar) with values ≥ 1.
nintervals = number of intervals (scalar) (temporary variable).
NetPEDistr = network propagation event distribution matrix (#size intervals x 2) in which column 1 contains event size intervals in ascending order and column 2 contains the number of events in each of those size intervals.
The following variables are used to create the frequency spectrum of the network propagation event time series.
Lsig = length of signal time series (scalar), i.e., the length of the network propagation event time series.
NFFT = next power of 2 up from Lsig (scalar). This is an input to the MATLAB fft (fast Fourier transform) function.
PEdft = propagation event discrete Fourier transform vector (1 x NFFT).
PEssAmp = propagation event single-sided amplitude vector (1 x NFFT/2+1).
normfreq = vector of normalized frequency values (1 x NFFT/2+1).
numpoints = number of frequency spectrum points for partial spectrum (scalar). The maximum number of points available = NFFT/2+1.
freq = vector of normalized frequency values for partial spectrum (1 x numpoints).
amp = vector of amplitude values for partial spectrum (1 x numpoints).

Path length analysis variables:
PLmax_TS = NumCStages_TS = maximum path length in each time step (1x NumTS).
PLltmax = maximum element in PLmax_TS = long-term maximum path length in all time steps (scalar).
plsi = path length size interval (scalar).
nplintervals = number of path length intervals (scalar) (temporary variable).
MaxPLDistr = maximum path length distribution matrix (#size intervals x 2). Column 1 contains the path length intervals and column 2 contains the number of events in each of those length intervals.
PLDistr_t_TS = per-time-step path length distribution multidimensional array (PLltmax x 2 x NumTS) that provides the path length distribution matrix (number of paths vs. length) for each time step.
PLDistr_tc_TS = path length distribution multidimensional array (PLltmax x 2 x NumTS) that provides the over-time-cumulative path length distribution per time step.
PLmean_TS = time series vector that provides the mean path length in each individual time step (1x NumTS).
PLsum = weighted sum of path lengths in an individual time step (scalar) (temporary variable).
PLtotal = total number of paths in an individual time step (scalar) (temporary variable).
mplsi = mean path length size interval (scalar).
nmplintervals = number of mean path length intervals (scalar) (temporary variable).

MeanPLDistr = mean path length distribution matrix (#size intervals x 2). Column 1 contains the path length intervals and column 2 contains the number of events in each of those length intervals.

Indirect effects analysis definitions and variables:
DER = Direct Effects Ratio. DER is defined as the number of paths of length 1 divided by the total number of paths.
IER = Indirect Effects Ratio. IER is defined as the number of paths of length > 1 divided by the total number of paths.
IEI = Indirect Effects Index. IEI is defined as the number of paths of length > 1 divided by the number of paths of length 1.
The following vector variables contain time series (individual time-step values vs. time) of the direct effect and indirect effect indicators.
DER_t_TS = Direct Effects Ratio for each individual time step in the simulation run per time step (1 x NumTS).
IER_t_TS = Indirect Effects Ratio for each individual time step in the simulation run per time step (1 x NumTS).
IEI_t_TS = Indirect Effects Index for each individual time step in the simulation run per time step (1 x NumTS).
The following vector variables contain time series (over-time-cumulative values vs. time) of the direct effect and indirect effect indicators.
DER_tc_TS = per-time-step over-time-cumulative Direct Effects Ratio (1 x NumTS).
IER_tc_TS = per-time-step over-time-cumulative Indirect Effects Ratio (1 x NumTS).
IEI_tc_TS = per-time-step over-time-cumulative Indirect Effects Index (1 x NumTS).
The following vector variables provide path quantity time series (number of paths at each time step vs. time) and corresponding distributions.
IndPathQ_TS = indirect-path-quantity time series vector (1 x NumTS).
IndPathQDistr = indirect-path-quantity distribution matrix (#size intervals x 2).
ipsi = indirect-path size interval (scalar).
nipintervals = number of indirect-path intervals (scalar) (temporary variable).
DirPathQ_TS = direct-path-quantity time series vector (1 x NumTS).
DirPathQDistr = direct-path-quantity distribution matrix (#size intervals x 2).
dpsi = direct-path size interval (scalar).
ndpintervals = number of direct-path intervals (scalar) (temporary variable).
The following vector variables provide cumulative path quantity time series (cumulative number of paths at each time step vs. time).
CuIndPathQ_TS = cumulative-indirect-path-quantity time series vector (1 x NumTS).
CuDirPathQ_TS = cumulative-direct-path-quantity time series vector (1 x NumTS).

Network connectivity analysis variables:
NodeODeg_t_TS = node-out-degree multidimensional array (1x100xNumTS) that provides a node out-degree vector for each individual time step per time step.
NodeIDeg_t_TS = node-in-degree multidimensional array (1x100xNumTS) that provides a node in-degree vector for each individual time step per time step.
NodeCDeg_t_TS = node-combined-degree multidimensional array (1x100xNumTS) that provides a node combined-degree vector for each individual time step per time step.

NodeODegGrid_t_TS = node-out-degree-grid multidimensional array (10x10xNumTS) that provides a node out-degree grid for each individual time step per time step.
NodeIDegGrid_t_TS = node-in-degree-grid multidimensional array (10x10xNumTS) that provides a node in-degree grid for each individual time step per time step.
NodeCDegGrid_t_TS = node-combined-degree-grid multidimensional array (10x10xNumTS) that provides a node combined-degree grid for each individual time step per time step.
ODmax_t_TS = vector (1x NumTS) that provides the maximum out-degree in each time step per time step.
IDmax_t_TS = vector (1x NumTS) that provides the maximum in-degree in each time step per time step.
CDmax_t_TS = vector (1x NumTS) that provides the maximum combined-degree in each time step per time step.
ODltmax_t = long-term maximum out-degree in all time steps (scalar).
IDltmax_t = long-term maximum in-degree in all time steps (scalar).
CDltmax_t = long-term maximum combined-degree in all time steps (scalar).
The following three variables are two-column node degree distribution matrices. Column 1 contains the node degree intervals and column 2 contains the number of events in each of those intervals.
MaxODDistr = maximum out-degree distribution matrix (#size intervals x 2).
MaxIDDistr = maximum in-degree distribution matrix (#size intervals x 2).
MaxCDDistr = maximum combined-degree distribution matrix (#size intervals x 2).
mdsi = maximum node degree size interval (scalar).
nmdintervals = number of maximum node degree intervals (scalar) (temporary variable).
CompMD = compartment mean degree (scalar).
CompNetDen = compartment network connection density (scalar).
CandNMD = "candidate" node mean degree (scalar).
CandNetDen = "candidate" node network connection density (scalar).
NMD_t_TS = node mean degree for each individual time step per time step (1xNumTS).
NMD_tc_TS = over-time-cumulative node mean degree per time step (1xNumTS).
Node mean degree is equal to the number of network edges divided by the number of network nodes.
NetDen_t_TS = network density for each individual time step per time step (1xNumTS).
NetDen_tc_TS = over-time-cumulative network density per time step (1xNumTS).
Network connection density is equal to node mean degree divided by (number of nodes – 1).
nmdsi = node mean degree size interval (scalar).
nnmdintervals = number of node mean degree intervals (scalar) (temporary variable).
NMDDistr = node mean degree distribution matrix (#size intervals x 2). Column 1 contains the node mean degree intervals and column 2 contains the number of events in each of those mean degree intervals.
NodeODeg_tc_TS = node-out-degree multidimensional array (1x100xNumTS) that provides an over-time-cumulative node out-degree vector per time step.
NodeIDeg_tc_TS = node-in-degree multidimensional array (1x100xNumTS) that provides an over-time-cumulative node in-degree vector per time step.
NodeCDeg_tc_TS = node-combined-degree multidimensional array (1x100xNumTS) that provides an over-time-cumulative node combined-degree vector per time step.

ODmax_tc_TS = vector (1x NumTS) that provides the maximum cumulative out-degree per time step.
IDmax_tc_TS = vector (1x NumTS) that provides the maximum cumulative in-degree per time step.
CDmax_tc_TS = vector (1x NumTS) that provides the maximum cumulative combined-degree per time step.
ODltmax_tc = long-term maximum cumulative out-degree across time steps (scalar).
IDltmax_tc = long-term maximum cumulative in-degree across time steps (scalar).
CDltmax_tc = long-term maximum cumulative combined-degree across time steps (scalar).
dsi = degree size interval (scalar).
ndintervals = number of degree intervals (scalar) (temporary variable).
The following three multidimensional arrays each provide a two-column node degree distribution matrix for each individual time step per time step. In each matrix, column 1 contains the ordered node degree size intervals and column 2 contains the number of nodes in each of those size intervals.
ODegDistr_t_TS = multidimensional array that provides the node out-degree distribution for each individual time step per time step (#size intervals x 2 x NumTS).
IDegDistr_t_TS = multidimensional array that provides the node in-degree distribution for each individual time step per time step (#size intervals x 2 x NumTS).
CDegDistr_t_TS = multidimensional array that provides the node combined-degree distribution for each individual time step per time step (#size intervals x 2 x NumTS).
The following three multidimensional arrays each provide a two-column over-time-cumulative node degree distribution matrix per time step. In each matrix, column 1 contains the ordered node degree size intervals and column 2 contains the number of nodes in each of those size intervals.
ODegDistr_tc_TS = multidimensional array that provides the node over-time-cumulative out-degree distribution per time step (#size intervals x 2 x NumTS).
IDegDistr_tc_TS = multidimensional array that provides the node over-time-cumulative in-degree distribution per time step (#size intervals x 2 x NumTS).
CDegDistr_tc_TS = multidimensional array that provides the node over-time-cumulative combined-degree distribution per time step (#size intervals x 2 x NumTS).
NumLN_t_TS = number of linked nodes for each individual time step per time step (1xNumTS).
CompSize_t_TS = candidate giant component size for each individual time step per time step (1xNumTS).
At time step i, CompSize_t_TS(i) equals NumLN_t_TS(i) divided by the total number of nodes in the network.

Graphics generation variables:
tsn = time step number (scalar).
NCoord = node-coordinate matrix (100x2) in which each row provides a spatial coordinate pair that represents one node.
PNCoord = propagating-node-coordinate matrix in which each row provides a spatial coordinate pair that represents a propagating node. Each instance of the matrix applies to a single time step.
INCoord = input-node-coordinate matrix that provides a spatial coordinate pair that represents an input node. Each instance of the matrix applies to a single time step.
fvsi = flow value size interval (scalar).

nfvintervals = number of flow value intervals (scalar) (temporary variable).
ANFVsubset = node flow value subset “adjacency matrix” (100 x 100) (temporary variable).
NodeCDegGridz = matrix for plotting node-combined-degree-grid zero values (10x10) (temporary variable).
NodeCDegGridnz = matrix for plotting node-combined-degree-grid nonzero values (10x10) (temporary variable).
NodeIDegGridz = matrix for plotting node-in-degree-grid zero values (10x10) (temporary variable).
NodeIDegGridnz = matrix for plotting node-in-degree-grid nonzero values (10x10) (temporary variable).

APPENDIX D

DYNAMICS MODEL MASTER M-FILE

```
%% Model of Ecological Network Propagation Dynamics
%  Master m-file

%% > Model network structure, parameters, and relationships

%% Specify underlying ecological network compartment model
% A = compartment model adjacency matrix (10x10)
A = [0 0 0 0 1 0 0 0 0 0;
     1 0 0 0 0 0 1 0 0 0;
     0 1 0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 1 1 1;
     1 1 1 1 0 0 1 1 1 1;
     1 1 1 1 0 0 1 1 1 1;
     0 0 0 0 0 1 0 0 0 0;
     1 1 0 0 0 0 1 0 0 0;
     0 0 1 0 0 1 0 1 0 1;
     0 0 0 0 0 1 0 0 0 0];
% zL = compartment model input logical vector (10x1)
zL = [0;
      0;
      1;
      1;
      1;
      1;
      1;
      0;
      0;
      0];
% yL = compartment model output logical vector (1x10)
yL = [0 0 1 1 0 1 1 1 0 1];

%% Build the model network node grid and its relationship to compartments
% NNG = node-number grid matrix (10x10)
NNG = [1 11 21 31 41 51 61 71 81 91;
       2 12 22 32 42 52 62 72 82 92;
       3 13 23 33 43 53 63 73 83 93;
       4 14 24 34 44 54 64 74 84 94;
       5 15 25 35 45 55 65 75 85 95;
       6 16 26 36 46 56 66 76 86 96;
       7 17 27 37 47 57 67 77 87 97;
       8 18 28 38 48 58 68 78 88 98;
       9 19 29 39 49 59 69 79 89 99;
      10 20 30 40 50 60 70 80 90 100];

%%
% NCG = node-compartment grid matrix (10x10) using function randi
```

```
% NCG = randi(10,10);
% NCG =
%      9     2     7     8     5     3     8     9     4     1
%     10    10     1     1     4     7     3     3     9     1
%      2    10     9     3     8     7     6     9     6     6
%     10     5    10     1     8     2     7     3     6     8
%      7     9     7     1     2     2     9    10    10    10
%      1     2     8     9     5     5    10     4     3     2
%      3     5     8     7     5    10     6     2     8     6
%      6    10     4     4     7     4     2     3     8     5
%     10     8     7    10     8     6     2     7     4     1
%     10    10     2     1     8     3     3     5     6     4

%%
% QNinC = quantity of nodes in each compartment (10x1)
QNinC = [sum(sum(NCG==1));
         sum(sum(NCG==2));
         sum(sum(NCG==3));
         sum(sum(NCG==4));
         sum(sum(NCG==5));
         sum(sum(NCG==6));
         sum(sum(NCG==7));
         sum(sum(NCG==8));
         sum(sum(NCG==9));
         sum(sum(NCG==10))];
% QNinC =
%      9
%     11
%     10
%      8
%      8
%      9
%     10
%     12
%      8
%     15

%%
% NHCL = node-to-home-compartment logical matrix (10x100)
NHCL = zeros(10,100);
for j = 1:100;
    NHCL(NCG(j),j) = 1;
end

%%
% NCCL = node-to-connecting-compartment logical matrix (10x100)
NCCL = zeros(10,100);
for j = 1:100;
    NCCL(:,j) = A(:,NCG(j));
end

%% Specify the number of model simulation time steps
NumTS = 1000;

%% Specify the node stock and node propagation rules
% thd = node stock threshold for propagation (scalar)
```

```
thd = 4;
% npfq = node propagation flow quantity (scalar)
npfq = 4;

%% Identify the model network input nodes and output nodes
% input_nodes = input-node column vector (#input nodes x1)
% INL = input-node logical grid matrix (10x10)
[rows, columns] = find(NHCL(zL==1,:)==1);
input_nodes = columns;
INL = zeros(10,10);
INL(input_nodes) = 1;

%%
% output_nodes = output-node column vector (#output nodes x1)
% ONL = output-node logical grid matrix (10x10)
[rows, columns] = find(NHCL(yL==1,:)==1);
output_nodes = columns;
ONL = zeros(10,10);
ONL(output_nodes) = 1;

%%
% IOL = input/output-node logical matrix (2x100)
IOL = zeros(2,100);
IOL(1,input_nodes) = 1;
IOL(2,output_nodes) = 1;

%% Define variables for the values of model network inputs
% INVG_tc = input-node-value-grid matrix (10x10)
% (values are 0, 1, 2, …)
% preallocation and initialization
INVG_tc = zeros(10,10);
% INVG_tc_TS = input-node-value-grid multidimensional array (10x10xNumTS)
% (values are 0, 1, 2, …)
% preallocation and initialization
INVG_tc_TS = zeros(10,10,NumTS);
% ICV_tc = input-compartment-value vector (1x10)
% (values are 0, 1, 2, … )
% preallocation and initialization
ICV_tc = zeros(1,10);
% ICV_tc_TS = input-compartment-value multidimensional vector (1x10xNumTS)
% (values are 0, 1, 2, … )
% preallocation and initialization
ICV_tc_TS = zeros(1,10,NumTS);
% SIV_tc = system-input-value scalar (values are 0, 1, 2, … )
% preallocation and initialization
SIV_tc = 0;
% SIV_tc_TS = system-input-value vector (1xNumTS) (values are 0, 1, 2, … )
% preallocation and initialization
SIV_tc_TS = zeros(1,NumTS);

%% Define variables for the values of model network outputs
% ONVG_tc = output-node-value-grid matrix (10x10)
% (values are 0, 1, 2, … )
% preallocation and initialization
ONVG_tc = zeros(10,10);
% ONVG_tc_TS = output-node-value-grid multidimensional array (10x10xNumTS)
```

```
% (values are 0, 1, 2, … )
% preallocation and initialization
ONVG_tc_TS = zeros(10,10,NumTS);
% OCV_tc = output-compartment-value vector (1x10)
% (values are 0, 1, 2, … )
% preallocation and initialization
OCV_tc = zeros(1,10);
% OCV_tc_TS = output-compartment-value multidimensional vector (1x10xNumTS)
% (values are 0, 1, 2, … )
% preallocation and initialization
OCV_tc_TS = zeros(1,10,NumTS);
% SOV_tc = system-output-value scalar
% (values are 0, 1, 2, … )
% preallocation and initialization
SOV_tc = 0;
% SOV_tc_TS = system-output-value vector (1xNumTS)
% (values are 0, 1, 2, … )
% preallocation and initialization
SOV_tc_TS = zeros(1,NumTS);

%% Define variables for the values of model network stocks
% NSVG_tc = node-stock-value-grid matrix (10x10)
% (values are 0, 1, 2, … , thd)
% initialization
% Initial values of NSVG_tc are assigned randomly
% in the range 0 to thd-1
temp = randi([0,thd-1],10);
NSVG_tc = temp;
% NSVG_tc =
%      0     1     0     1     3     1     3     0     2     3
%      3     0     3     3     2     0     1     1     1     3
%      1     0     0     0     1     3     0     3     2     1
%      2     3     3     1     2     3     1     0     0     1
%      0     0     3     0     1     1     0     0     2     1
%      2     3     3     0     0     1     0     0     0     1
%      1     2     0     3     0     1     3     2     1     2
%      2     3     1     2     0     3     3     2     2     2
%      2     0     1     2     0     1     2     2     3     3
%      2     1     3     0     0     0     0     1     0     3
%
% NSVG_tc_TS = node-stock-value-grid multidimensional array (10x10xNumTS)
% (values are 0, 1, 2, … , thd)
% preallocation
NSVG_tc_TS = zeros(10,10,NumTS);

%%
% CSV_tc = compartment-stock-value vector (1x10)
% (values are 0, 1, 2, …)
% preallocation
CSV_tc = zeros(1,10);
% initialization
% For vector CSV_tc, the initial value of each of the
% compartment elements is set equal to the sum of the initial stock values
% of the contained nodes
temp = NSVG_tc;
for j=1:10;
```

```
    CSV_tc(1,j) = sum(temp(NHCL(j,:)==1));
end
% CSV_tc =
%     18    21     6    16    12    11    13    14     4    22
%
% CSV_tc_TS = compartment-stock-value multidimensional vector (1x10xNumTS)
% (values are 0, 1, 2, …)
% preallocation
CSV_tc_TS = zeros(1,10,NumTS);

%%
% SSV_tc = system-stock-value scalar (values are 0, 1, 2, … )
% initialization
% The initial value of SSV_tc is set equal to the
% sum of the initial stock values of all compartments
SSV_tc = sum(CSV_tc);
% SSV_tc = 137
%
% SSV_tc_TS = system-stock-value vector (1xNumTS) (values are 0, 1, 2, … )
% preallocation
SSV_tc_TS = zeros(1,NumTS);

%% Generate compartment selection order
% QNCC = quantity-of-node-connecting-compartments row vector (1x100)
QNCC = sum(NCCL);
% CSO = compartment-selection-order matrix (10x100)
CSO = randi([100,1000],[10,100]);
CSO(NCCL==0) = 0;
% Order (1, 2, …) the non-zero values in each column of CSO:
for j = 1:100
    for i = 1: QNCC(j)
        [C,I] = max(CSO);
        % I is a row vector of the row numbers of the maximum in each
        % column of CSO
        CSO(I(j),j) = i;
    end
end

%% Construct the network node basic adjacency matrix (AA)
% AA = network node basic adjacency matrix for “candidate”
% connections (100x100)
AA = zeros(100,100);    % preallocation
for j = 1:100    % columns (“from” nodes)
    CCtemp = find(NCCL(:,j)==1);    % connect-to compartments
    for k = 1:sum(NCCL(:,j))    % rows (connect-to compartments)
        Ntemp = find(NHCL(CCtemp(k),:)==1);    % “to” nodes
        AA(Ntemp,j) = 1;
    end
end

%% Create a node adjacency-type “adjacency matrix” (AAT)
% AAT = adjacency adjacency-type matrix (100x100)
% (element values are 1, 2, 3)
% AAT provides the adjacency type of “to” nodes
% type 1 = local neighbor
% type 2 = extended local neighbor
```

```
% type 3 = global neighbor
AAT = zeros(100,100);    % preallocation
for j = 1:100    % columns (“from” nodes)
    N2temp = find(AA(:,j)==1);    % “to” nodes
    for k = 1:sum(AA(:,j))    % rows (“to” nodes)
        % grid comparison
        if logical(abs(N2temp(k)-j)==1) || logical(abs(N2temp(k)-j)==10)
            AAT(N2temp(k),j) = 1;
        elseif logical(abs(N2temp(k)-j)==2) || logical(abs(N2temp(k)-j)==20)
            AAT(N2temp(k),j) = 2;
        else
            AAT(N2temp(k),j) = 3;
        end
    end
end

%% Create a node compartment-number “adjacency matrix” (ACN)
% ACN = adjacency compartment-number matrix (100x100)
% (element values are 1 to 10)
% ACN provides home compartment numbers of “to” nodes
ACN = zeros(100,100);    % preallocation
for j = 1:100    % columns (“from” nodes)
    N2temp = find(AA(:,j)==1);    % “to” nodes
    for k = 1:sum(AA(:,j))    % rows (“to” nodes)
        ACN(N2temp(k),j) = find(NHCL(:,N2temp(k))==1);
    end
end

%% Create a node attachment-preference-strength “adjacency matrix” (AAP)
% First create a row vector (NAPS) of per-node randomly generated
% attachment preference strengths and then create matrix AAP
% NAPS = node-to-attachment-preference-strength vector (1x100)
% (element values: 0 to 1)
% AAP = adjacency attachment-preference-strength matrix (100x100)
% (element values: 0 to 1)
% AAP provides attachment preference strength of “to” nodes
AAP = zeros(100,100);    % preallocation
NAPS = rand(1,100);
for j = 1:100    % columns (“from” nodes)
    N2temp = find(AA(:,j)==1);    % “to” nodes
    for k = 1:sum(AA(:,j))    % rows (“to” nodes)
        AAP(N2temp(k),j) = NAPS(N2temp(k));
    end
end

%% Define network node “operational” adjacency matrices and arrays
% Define a stage-cumulative operational node adjacency matrix
% The matrix applies to an individual time step
% AAO_t = cumulative operational node adjacency matrix (100x100)
% AAO_t is cumulative over all node propagation instances
% over all stages of a time step
% The initial values of AAO_t are all zeros and the array must be
% re-initialized to zeros at every time step
% preallocation and initialization
AAO_t = zeros(100,100);
```

```
%%
% Define a per-time-step operational node adjacency multidimensional
% array (AAO_t_TS) that provides the completed cumulative operational
% adjacency matrix AAO_t for each individual time step
% [AAO_t_TS(:,:,k) = completed AAO_t for time step k]
% AAO_t_TS = per-time-step operational node adjacency multidimensional
% array (100x100x NumTS)
% preallocation and initialization
AAO_t_TS = zeros(100,100,NumTS);

%%
% Define an operational node adjacency multidimensional array that
% consists of over-time cumulative adjacency matrices
% AAO_tc_TS = per-time-step over-time-cumulative operational node
% adjacency multidimensional array (100x100x NumTS)
% [AAO_tc_TS(:,:,k) = AAO_tc_TS(:,:,k-1) + AAO_t_TS(:,:,k)]
% preallocation and initialization
AAO_tc_TS = zeros(100,100,NumTS);

%% Define a node-flow-value “adjacency” matrix and multidimensional array
% Define an “adjacency” matrix that provides the over-time-cumulative
% values of the direct flows between adjacent nodes as the network operates
% over the simulation run
% ANFV_tc = over-time-cumulative adjacency node-flow-value matrix
% (100x100) (values are 0, 1, 2, … )
% preallocation and initialization
ANFV_tc = zeros(100,100);
% Define a per-time-step “adjacency” multidimensional array that provides
% the over-time-cumulative values of the direct flows between adjacent
% nodes at each time step
% ANFV_tc_TS = per-time-step over-time-cumulative adjacency node-flow-value
% multidimensional array (100x100xNumTS) (values are 0, 1, 2, … )
% preallocation and initialization
ANFV_tc_TS = zeros(100,100,NumTS);

%% Define a compartment-flow-value “adjacency” matrix and array
% Define an “adjacency” matrix that provides the over-time-cumulative
% values of the direct flows between adjacent compartments as the network
% operates over the simulation run
% ACFV_tc = over-time-cumulative adjacency compartment-flow-value matrix
% (10x10) (values are 0, 1, 2, … )
% preallocation and initialization
ACFV_tc = zeros(10,10);
% Define a per-time-step “adjacency” multidimensional array that provides
% the over-time-cumulative values of the direct flows between adjacent
% compartments at each time step
% ACFV_tc_TS = per-time-step over-time-cumulative adjacency
% compartment-flow-value multidimensional array (10x10xNumTS)
% (values are 0, 1, 2, … )
% preallocation and initialization
ACFV_tc_TS = zeros(10,10,NumTS);

%% > Propagation process flow

%% Summary view of the propagation process
% We are modeling the dynamics of propagation in complex ecological
```

```
% networks.  The propagation process is the core of the model.  The process
% proceeds by model simulation time step, by stages within a time step, and
% by node propagation events within a stage.  The propagation process,
% therefore, has three primary nested loops:
%   1. Time step loop
%   2. Stage loop
%   3. Node propagation event loop
% Throughout the total propagation process there are also flow control
% logical constructs as required.
%
% Time step loop:
% A model simulation run consists of NumTS time steps.  The time step loop,
% therefore, has NumTS iterations.  Each model simulation time step can
% consist of multiple stages.  Stage 0 is the input stage.  A unit input is
% applied to a randomly selected input node.  At every time step, there is
% a Stage 0 input.  Propagation stages begin with Stage 1.
%
% Stage loop:
% Stage 1 is the first potential propagation stage.  Stage 1 can have zero
% or one node propagation events – zero if the input node does not
% propagate and one if it does.  If the node does propagate, we move to
% stage 2.  Stage 2 (and any subsequent stages) can have zero, one, or more
% than one node propagation events.  Whenever a given stage has one or more
% propagating nodes, we move to the next stage.  [The presence of multiple
% propagation stages indicates that a “cascade” is in progress.]  When we
% reach a stage where there are no more propagating nodes, the simulation
% time step concludes.  The number of propagation stages for the time step
% is the last stage number minus 1.
%
% Node propagation event loop:
% Any given propagation stage can have 0, 1, or more node propagation
% events. The extent of a node propagation event is determined by the total
% flow from the propagating node.  This is specified by variable npfq –
% which is determined from the stock and propagation rules.  Each
% individual event corresponds to an iteration of the node propagation
% event loop.
%
% A node propagation event can be quite complex.  The node propagation
% event loop, therefore, contains three additional interior nested loops:
%   a. Neighborhood selection loop
%   b. Compartment selection loop
%   c. Node selection and propagation instance loop
%
% Here is how we proceed with propagation process development.  We begin
% with a high-level view of the program flow logic and progress to the
% detailed MATLAB code.  In this progression, we first define the
% high-level view of the flow logic and then define the basic propagation
% process variables (that show up in the high-level view).  Next we define
% the four sub-processes (and their variables) that are named in the
% high-level flow, i.e., input node processing, propagating node
% processing, output node processing, and node propagation instance
% processing.  We also provide expanded descriptions of other processing
% activities in the high-level flow as required.  As we proceed, we define
% additional variables needed to store data for our subsequent analysis
% and graphics generation activities.  We then add all of this detail to
% the high-level program flow logic to yield the detailed MATLAB program
% code for the propagation process.
```

```
%% High-level view of the program flow for the overall propagation process
% This is a narrative “pseudocode-like” rendering of the flow
%
% for i = 1:NumTS    % Time step loop
% Perform input node processing
% if input node stock < thd, continue
% propagation-event-node-set = input node
%    for j = 1:100    % Stage loop
%    % loop upper limit is any number larger than maximum number of stages
%    NumCStages = j - 1
%    if propagation-event-node-set is empty, break
%    Perform propagating node processing
%       for k = 1:size of propagation-event-node-set  % Node propagation
%                                                       event loop
%       Available flow F = npfq
%       from-node = propagation-event-node-set(k)
%       if from-node is an output node, perform output node processing,
%                                       F = F – 1
%          for u = 1:3    % Neighborhood selection loop
%          candidate to-node-set = to nodes with adjacency type u
%          if to-node-set is empty, continue
%          find to-node-set compartment numbers
%          find ranks of to-node-set compartment numbers
%             for v = 1:QNCC(from-node)    % Compartment selection loop
%             revise to-node-set % to include only the nodes in compartment
%                                  with rank v
%             if revised to-node-set is empty, continue
%                for w = 1:size of revised to-node-set    % Node selection
%                                                 and propagation instance loop
%                find revised to-node-set attachment preference strengths
%                find revised to-node-set attachment pref probability set
%                determine selected-node  % use mnrnd(1, probability set)
%                % propagate to selected-node
%                Perform node propagation instance processing
%                F = F - 1
%                if F = 0, break
%                further revise to-node-set by removing selected-node
%                % proceed until propagate to all nodes in the set or
%                  until flow is exhausted
%                end
%             if F = 0, break
%             end
%          if F = 0, break
%          end
%       end
%    check for new node propagation events [>= thd] and
%    populate propagation-event-node-set
%    end
% NumCStages_TS(i) = NumCStages
% end

%% Define the basic propagation process variables
% PENodeS = propagation-event-node-set column vector (temporary variable)
% initialization
PENodeS = [ ];
```

```
% NumCStages = current number of completed propagation stages (scalar)
% (temporary variable)
% initialization
NumCStages = 0;
% NumCStages_TS = per-time-step row vector of the total number of
% propagation stages in each individual completed time step (1xNumTS)
% preallocation and initialization
NumCStages_TS = zeros(1,NumTS);
% AvFlow = the currently available-flow for a propagating from-node
% (scalar) (temporary variable)
% initialization
AvFlow = 0;
% FNode = propagating from-node (scalar) (values are 1 to 100) (temporary
% variable)
% initialization
FNode = [ ];
% TNodeS = candidate to-node-set column vector = candidate
% propagation-instance-node-set column vector (temporary variable)
% initialization
TNodeS = [ ];
% TNodeSC = to-node-set-compartments column vector (temporary variable)
% initialization
TNodeSC = [ ];
% TNodeSCR = ranks of to-node-set-compartments column vector
% (temporary variable)
% initialization
TNodeSCR = [ ];
% RTNodeS = revised to-node-set column vector (temporary variable)
% initialization
RTNodeS = [ ];
% RTNodeSAPS = revised to-node-set-attachment-preference-strengths
% column vector (temporary variable)
% initialization
RTNodeSAPS = [ ];
% RTNodeSAPP = revised to-node-set-attachment-preference-probabilities
% column vector (temporary variable)
% initialization
RTNodeSAPP = [ ];
% SNode = propagation instance selected-node (scalar) (values are 1 to 100)
% (temporary variable)
% initialization
SNode = [ ];

%% Develop input node processing
% Note that program code that is "commented" in this cell will be made
% "active" in the overall propagation process cell

% INS_t = input-node-selection row vector for an individual time step
% (1x100)
% preallocation and initialization
INS_t = zeros(1,100);
% INS_t = IOL(1,:) .* rand(1,100);
% select the node with the highest random number
% InputNode_t = selected input-node for an individual time step (scalar)
% (values are 1 to 100)
% initialization
```

```
InputNode_t = [ ];
% [C,I] = max(INS_t);
% InputNode_t = I;
% apply a unit input to the selected node
% NSVG_tc(InputNode_t) = NSVG_tc(InputNode_t) + 1;

% INSL_t = input-node-selection-logical matrix for an individual time step
% (10x10) (values are 0, 1)
% preallocation and initialization
INSL_t = zeros(10,10);
% Identify selected input node
% INSL_t(InputNode_t) = 1;
% INSL_t_TS = per-time-step input-node-selection-logical multidimensional
% array (10x10xNumTS) (matrix values are 0, 1)
% INSL_t_TS is used for network propagation diagram generation
% preallocation and initialization
INSL_t_TS = zeros(10,10, NumTS);
% update at end of time step
% INSL_t_TS(:, :, i) = INSL_t;
% INSL_tc_TS = per-time-step over-time-cumulative
% input-node-selection-logical multidimensional array (10x10xNumTS)
% (matrix values are 0, 1)
% preallocation and initialization
INSL_tc_TS = zeros(10,10, NumTS);
% update at end of time step
% if i > 1
% INSL_tc_TS(:, :, i) = INSL_tc_TS(:, :, i - 1) + INSL_t_TS(:, :, i);
% else
% INSL_tc_TS(:, :, i) = INSL_t_TS(:, :, i);
% end

% update the node, compartment, and system input and stock values
% INVG_tc(InputNode_t) = INVG_tc(InputNode_t) + 1;
% ICV_tc(NCG(InputNode_t)) = ICV_tc(NCG(InputNode_t)) + 1;
% CSV_tc(NCG(InputNode_t)) = CSV_tc(NCG(InputNode_t)) + 1;
% SIV_tc = SIV_tc + 1;
% SSV_tc = SSV_tc + 1;
% also update INVG_tc_TS, NSVG_tc_TS, ICV_tc_TS, CSV_tc_TS, SIV_tc_TS, and
% SSV_tc_TS at end of time step

%% Develop propagating node processing
% Note that program code that is "commented" in this cell will be made
% "active" in the overall propagation process cell

% PNL_sc = stage-cumulative propagating-node-logical grid matrix (10x10)
% (values are 0, 1)
% preallocation and initialization
PNL_sc = zeros(10,10);
% identify propagating nodes
% PNL_sc(PENodeS) = 1;
% PNL_t_TS = per-time-step multidimensional array of individual time step
% (stage-cumulative) propagating-node-logical matrices (10x10xNumTS)
% (matrix values are 0, 1)
% PNL_t_TS is used for network propagation diagram generation
% preallocation and initialization
PNL_t_TS = zeros(10,10, NumTS);
```

```
% update at end of time step
% PNL_t_TS(:, :, i) = PNL_sc;
% PNL_tc_TS = per-time-step over-time-cumulative propagating-node-logical
% multidimensional array (10x10x NumTS) (matrix values are 0, 1)
% preallocation and initialization
PNL_tc_TS = zeros(10,10, NumTS);
% update at end of time step
% if i > 1
% PNL_tc_TS(:, :, i) = PNL_tc_TS(:, :, i - 1) + PNL_t_TS(:, :, i);
% else
% PNL_tc_TS(:, :, i) = PNL_t_TS(:, :, i);
% end

% update the node, compartment, and system stock values
% NSVG_tc(PENodeS) = NSVG_tc(PENodeS) - npfq;
% Same compartment can occur more than once – so need loop
% for jj = 1:size(PENodeS, 1)    % 1: number of propagating nodes
% CSV_tc(NCG(PENodeS(jj))) = CSV_tc(NCG(PENodeS(jj))) - npfq;
% end
% SSV_tc = SSV_tc - npfq * size(PENodeS, 1);

% nnpe_s_S_TS = number-of-node-propagation-events multidimensional vector
% (1x #stages xNumTS)
% nnpe_s_S_TS is used for path length analysis
% preallocation and initialization
nnpe_s_S_TS = zeros(1,15, NumTS);
% update nnpe_s_S_TS for stage j and time step i
% nnpe_s_S_TS(1, j, i) = size(PENodeS, 1);
% nnpe_sc = stage-cumulative number-of-node-propagation-events (scalar)
% preallocation and initialization
nnpe_sc = 0;
% update nnpe_sc for stage j
% nnpe_sc = nnpe_sc + size(PENodeS, 1);
% nnpe_sc_TS = per-time-step stage-cumulative
% number-of-node-propagation-events vector (1xNumTS)
% preallocation and initialization
nnpe_sc_TS = zeros(1, NumTS);
% update at end of time step
% nnpe_sc_TS(i) = nnpe_sc;

% nopn_s_S_TS = number-of-output-propagating-nodes multidimensional vector
% (1x #stages xNumTS)
% nopn_s_S_TS is used for path length analysis
% preallocation and initialization
nopn_s_S_TS = zeros(1,15, NumTS);
% update nopn_s_S_TS for stage j and time step i
% nopn_s_S_TS(1, j, i) = sum(ONL(PENodeS));
% nopn_sc = stage-cumulative number-of-output-propagating-nodes (scalar)
% preallocation and initialization
nopn_sc = 0;
% update nopn_sc for stage j
% nopn_sc = nopn_sc + sum(ONL(PENodeS));
% nopn_sc_TS = per-time-step stage-cumulative
% number-of-output-propagating-nodes vector (1xNumTS)
% preallocation and initialization
nopn_sc_TS = zeros(1, NumTS);
```

```
% update at end of time step
% nopn_sc_TS(i) = nopn_sc;

%% Develop output node processing
% Note that program code that is "commented" in this cell will be made
% "active" in the overall propagation process cell

% OPNL_sc = stage-cumulative output-propagating-node-logical matrix (10x10)
% (values are 0, 1)
% preallocation and initialization
OPNL_sc = zeros(10,10);
% identify output node
% OPNL_sc(FNode) = 1;
% OPNL_t_TS = per-time-step multidimensional array of individual time step
% (stage-cumulative) output-propagating-node-logical matrices
% (10x10xNumTS) (matrix values are 0, 1)
% OPNL_t_TS is used for network propagation diagram generation
% preallocation and initialization
OPNL_t_TS = zeros(10,10, NumTS);
% update at end of time step
% OPNL_t_TS(:, :, i) = OPNL_sc;
% OPNL_tc_TS = per-time-step over-time-cumulative
% output-propagating-node-logical multidimensional array (10x10x NumTS)
% (values are 0, 1)
% preallocation and initialization
OPNL_tc_TS = zeros(10,10, NumTS);
% update at end of time step
% if i > 1
% OPNL_tc_TS(:, :, i) = OPNL_tc_TS(:, :, i - 1) + OPNL_t_TS(:, :, i);
% else
% OPNL_tc_TS(:, :, i) = OPNL_t_TS(:, :, i);
% end

% update the node, compartment, and system output values
% ONVG_tc(FNode) = ONVG_tc(FNode) + 1;
% OCV_tc(NCG(FNode)) = OCV_tc(NCG(FNode)) + 1;
% SOV_tc = SOV_tc + 1;
% also update ONVG_tc_TS, OCV_tc_TS, and SOV_tc_TS at end of time step

%% Develop node propagation instance processing
% Note that program code that is "commented" in this cell will be made
% "active" in the overall propagation process cell

% nnpi_sc = stage-cumulative number-of-node-propagation-instances (scalar)
% preallocation and initialization
nnpi_sc = 0;
% propagate to selected node
% nnpi_sc = nnpi_sc + 1;
% nnpi_sc_TS = per-time-step stage-cumulative
% number-of-node-propagation-instances vector (1xNumTS)
% nnpi_sc_TS is used for network propagation event analysis
% preallocation and initialization
nnpi_sc_TS = zeros(1, NumTS);
% update at end of time step
% nnpi_sc_TS(i) = nnpi_sc;
```

```
% NetPESize_TS = per-time-step network-propagation-event-size vector
% (1xNumTS)
% NetPESize_TS is used for network propagation event analysis
% preallocation and initialization
NetPESize_TS = zeros(1, NumTS);
% update at end of time step
% if nnpi_sc_TS(i) > 1
% NetPESize_TS(i) = nnpi_sc_TS(i) + 1;
% else
% NetPESize_TS(i) = 0;
% end

% update AAO_t
% AAO_t(SNode, FNode) = 1;
% update at end of time step
% AAO_t_TS(:, :, i) = AAO_t;
% if i > 1
% AAO_tc_TS(:, :, i) = AAO_tc_TS(:, :, i - 1) + AAO_t_TS(:, :, i);
% else
% AAO_tc_TS(:, :, i) = AAO_t_TS(:, :, i);
% end

% update the node and compartment flow value matrices and arrays
% ANFV_tc(SNode, FNode) = ANFV_tc(SNode, FNode) + 1;
% ACFV_tc(NCG(SNode), NCG(FNode)) = ACFV_tc(NCG(SNode), NCG(FNode)) + 1;
% also update ANFV_tc_TS and ACFV_tc_TS at end of time step
% ANFV_tc_TS(:, :, i) = ANFV_tc;
% ACFV_tc_TS(:, :, i) = ACFV_tc;

% update the node, compartment, and system stock values
% NSVG_tc(SNode) = NSVG_tc(SNode) + 1;
% CSV_tc(NCG(SNode)) = CSV_tc(NCG(SNode)) + 1;
% SSV_tc = SSV_tc + 1;
% also update NSVG_tc_TS, CSV_tc_TS, and SSV_tc_TS at end of time step

%% Develop other processing activities
% Note that program code that is "commented" in this cell will be made
% "active" in the overall propagation process cell

% Develop processing for “candidate to-node-set = to nodes with
% adjacency type u”
% TNodeS = find(AAT(:, FNode) == u);
% AAT row numbers are to-node numbers
% TNodeS is a column vector of to-node numbers

% Develop processing for “find to-node-set compartment numbers”
% TNodeSC = ACN(TNodeS, FNode);
% TNodeSC is a column vector of compartment numbers of the to-nodes
% When more than one to-node has the same compartment number,
% elements of TNodeSC are repeated

% Develop processing for “find ranks of to-node-set compartment numbers”
% TNodeSCR = CSO(TNodeSC, FNode);
% TNodeSCR is a column vector of ranks of compartment numbers
% of the to-nodes
```

```
% The row specification for CSO can contain repeated rows
% TNodeSCR, therefore, can contain repeated ranks

% Develop processing for “revise to-node-set to include only the nodes
% in compartment with rank v”
% RTNodeS = TNodeS(find(TNodeSCR == v));
% RTNodeS can contain zero, one, or more elements
% MATLAB gives the same result with or without the “find” function

% Develop processing for “find revised to-node-set attachment preference
% strengths”
% RTNodeSAPS = AAP(RTNodeS, FNode);
% RTNodeSAPS is a column vector of attachment preference strengths
% of the to-nodes

% Develop processing for “find revised to-node-set attachment preference
% probability set”
% RTNodeSAPP = RTNodeSAPS / sum(RTNodeSAPS);
% RTNodeSAPP is a column vector (set) of attachment preference
% probabilities of the to-nodes

% Develop processing for “determine selected-node”
% SNode = RTNodeS(find(mnrnd(1, RTNodeSAPP) == 1));
% For 1 trial and to-node probability set RTNodeSAPP, MATLAB function
% “mnrnd” provides a row vector of outcomes with a one to indicate the
% selected node and zeros elsewhere.  Function “find” provides the index
% of the selected node.  RTNodeS(index) provides the selected node number.
% MATLAB gives the same result with or without the “find” function

% Develop processing for “further revise to-node-set by
% removing selected-node”
% RTNodeS(RTNodeS == SNode) = [ ];
% This command simply removes SNode from RTNodeS

% Develop processing for “check for new node propagation events and
% populate propagation-event-node-set”
% PENodeS = find(NSVG_tc >= thd);
% We must check each node stock value to see if any are equal to or
% greater than the node stock value threshold, thd.  Any node that
% satisfies this condition is a propagating node that initiates a
% node propagation event and becomes an element of PENodeS.

%% Develop detailed program flow for the overall propagation process

% Add all of the above detail to the high-level program flow logic
% to yield the detailed MATLAB program code for the propagation process

for i = 1:NumTS    % Time step loop
INSL_t = zeros(10,10);
PNL_sc = zeros(10,10);
nnpe_sc = 0;
nopn_sc = 0;
OPNL_sc = zeros(10,10);
nnpi_sc = 0;
AAO_t = zeros(100,100);
```

```
% perform input node processing:
% select an input node at random
INS_t = IOL(1,:) .* rand(1,100);
[C,I] = max(INS_t);
InputNode_t = I;
% apply a unit input to the selected node
NSVG_tc(InputNode_t) = NSVG_tc(InputNode_t) + 1;
% identify selected input node in grid
INSL_t(InputNode_t) = 1;
% update the node, compartment, and system input and stock values
INVG_tc(InputNode_t) = INVG_tc(InputNode_t) + 1;
ICV_tc(NCG(InputNode_t)) = ICV_tc(NCG(InputNode_t)) + 1;
CSV_tc(NCG(InputNode_t)) = CSV_tc(NCG(InputNode_t)) + 1;
SIV_tc = SIV_tc + 1;
SSV_tc = SSV_tc + 1;
% check for input node propagation
if NSVG_tc(InputNode_t) < thd
PENodeS = [ ];
% input node does not propagate
% proceed to stage loop and “break” to update data for the time step
else
PENodeS = InputNode_t;
% input node propagates
% proceed to stage loop
% after all propagation concludes, update data for the time step
end

  for j = 1:100    % Stage loop
  % loop upper limit is any number larger than maximum number of stages
  NumCStages = j - 1;
  if isempty(PENodeS) == 1
  break
  end
  % perform propagating node processing:
  % identify propagating nodes in grid
  PNL_sc(PENodeS) = 1;
  % update the node, compartment, and system stock values
  NSVG_tc(PENodeS) = NSVG_tc(PENodeS) - npfq;
  % Same compartment can occur more than once – so need loop
  for jj = 1:size(PENodeS, 1)    % 1: number of propagating nodes
  CSV_tc(NCG(PENodeS(jj))) = CSV_tc(NCG(PENodeS(jj))) - npfq;
  end
  SSV_tc = SSV_tc - npfq * size(PENodeS, 1);
  % update nnpe_s_S_TS for stage j and time step i
  nnpe_s_S_TS(1, j, i) = size(PENodeS, 1);
  % update nnpe_sc for stage j
  nnpe_sc = nnpe_sc + size(PENodeS, 1);
  % update nopn_s_S_TS for stage j and time step i
  nopn_s_S_TS(1, j, i) = sum(ONL(PENodeS));
  % update nopn_sc for stage j
  nopn_sc = nopn_sc + sum(ONL(PENodeS));

    for k = 1: size(PENodeS, 1)    % Node propagation event loop
    AvFlow = npfq;
    FNode = PENodeS(k);
    if IOL(2, FNode) == 1  % if from-node is an output node
```

```
% perform output node processing:
% identify output node in grid
OPNL_sc(FNode) = 1;
% update the node, compartment, and system output values
ONVG_tc(FNode) = ONVG_tc(FNode) + 1;
OCV_tc(NCG(FNode)) = OCV_tc(NCG(FNode)) + 1;
SOV_tc = SOV_tc + 1;
AvFlow = AvFlow - 1;
end

  for u = 1:3    % Neighborhood selection loop
  % candidate to-node-set = to nodes with adjacency type u
  TNodeS = find(AAT(:, FNode) == u);
  if isempty(TNodeS) == 1
  continue
  end
  % to-node-set compartment numbers
  TNodeSC = ACN(TNodeS, FNode);
  % ranks of to-node-set compartment numbers
  TNodeSCR = CSO(TNodeSC, FNode);

    for v = 1:QNCC(FNode)    % Compartment selection loop
    % revise to-node-set to include only the nodes in compartment
    % with rank v
    RTNodeS = TNodeS(find(TNodeSCR == v));
    if isempty(RTNodeS) == 1
    continue
    end
    upperlimit = size(RTNodeS, 1);

      for w = 1: upperlimit    % Node selection and
                               % propagation instance loop
      % revised to-node-set attachment preference strengths
      RTNodeSAPS = AAP(RTNodeS, FNode);
      % revised to-node-set attachment preference probability set
      % (normalized strengths)
      RTNodeSAPP = RTNodeSAPS / sum(RTNodeSAPS);
      % determine selected-node
      SNode = RTNodeS(find(mnrnd(1, RTNodeSAPP) == 1));
      % perform node propagation instance processing:
      % propagate to selected node
      nnpi_sc = nnpi_sc + 1;
      % update AAO_t
      AAO_t(SNode, FNode) = 1;
      % update the node and compartment flow value matrices
      ANFV_tc(SNode, FNode) = ANFV_tc(SNode, FNode) + 1;
      ACFV_tc(NCG(SNode), NCG(FNode)) = ...
          ACFV_tc(NCG(SNode), NCG(FNode)) + 1;
      % update the node, compartment, and system stock values
      NSVG_tc(SNode) = NSVG_tc(SNode) + 1;
      CSV_tc(NCG(SNode)) = CSV_tc(NCG(SNode)) + 1;
      SSV_tc = SSV_tc + 1;
      % Although we do, actually don't need to update SSV
      % variables (- or +) for internal propagation
      % Net result is zero
      AvFlow = AvFlow - 1;
```

```
            if AvFlow == 0
            break
            end
            % further revise to-node-set by removing selected-node
            RTNodeS(RTNodeS == SNode) = [ ];
            % reinitialize some variables
            RTNodeSAPS = [ ] ;
            RTNodeSAPP = [ ] ;
            SNode = [ ] ;
            % proceed until propagate to all nodes in the set or
            % until flow is exhausted
            end

          % reinitialize
          RTNodeS = [ ];
          if AvFlow == 0
          break
          end
          end

        % reinitialize variables
        TNodeS = [ ] ;
        TNodeSC = [ ] ;
        TNodeSCR = [ ] ;
        if AvFlow == 0
        break
        end
        end

      % reinitialize
      FNode = [ ];
      end

    % reinitialize
    PENodeS = [ ];
    % check for new node propagation events and populate PENodeS
    PENodeS = find(NSVG_tc >= thd);
    end

% update data for the time step
NumCStages_TS(i) = NumCStages;
NSVG_tc_TS(:, :, i) = NSVG_tc;

INSL_t_TS(:, :, i) = INSL_t;
if i > 1
INSL_tc_TS(:, :, i) = INSL_tc_TS(:, :, i - 1) + INSL_t_TS(:, :, i);
else
INSL_tc_TS(:, :, i) = INSL_t_TS(:, :, i);
end
% In INSL_tc_TS, if any element > 1 then element = 1
check1 = INSL_tc_TS(:, :, i);
check1(check1 > 1) = 1;
INSL_tc_TS(:, :, i) = check1;

INVG_tc_TS(:, :, i) = INVG_tc;
```

```
ICV_tc_TS(:, :, i) = ICV_tc;
CSV_tc_TS(:, :, i) = CSV_tc;
SIV_tc_TS(1, i) = SIV_tc;
SSV_tc_TS(1, i) = SSV_tc;

% In PNL_sc, if any element > 1 then element = 1
PNL_sc(PNL_sc > 1) = 1;
PNL_t_TS(:, :, i) = PNL_sc;
if i > 1
PNL_tc_TS(:, :, i) = PNL_tc_TS(:, :, i - 1) + PNL_t_TS(:, :, i);
else
PNL_tc_TS(:, :, i) = PNL_t_TS(:, :, i);
end
% In PNL_tc_TS, if any element > 1 then element = 1
check2 = PNL_tc_TS(:, :, i);
check2(check2 > 1) = 1;
PNL_tc_TS(:, :, i) = check2;

nnpe_sc_TS(i) = nnpe_sc;
nopn_sc_TS(i) = nopn_sc;

% In OPNL_sc, if any element > 1 then element = 1
OPNL_sc(OPNL_sc > 1) = 1;
OPNL_t_TS(:, :, i) = OPNL_sc;
if i > 1
OPNL_tc_TS(:, :, i) = OPNL_tc_TS(:, :, i - 1) + OPNL_t_TS(:, :, i);
else
OPNL_tc_TS(:, :, i) = OPNL_t_TS(:, :, i);
end
% In OPNL_tc_TS, if any element > 1 then element = 1
check3 = OPNL_tc_TS(:, :, i);
check3(check3 > 1) = 1;
OPNL_tc_TS(:, :, i) = check3;

ONVG_tc_TS(:, :, i) = ONVG_tc;
OCV_tc_TS(:, :, i) = OCV_tc;
SOV_tc_TS(1, i) = SOV_tc;

nnpi_sc_TS(i) = nnpi_sc;
NetPESize_TS(i) = nnpi_sc_TS(i) + 1;

% In AAO_t, if any element > 1 then element = 1
AAO_t(AAO_t > 1) = 1;
AAO_t_TS(:, :, i) = AAO_t;
if i > 1
AAO_tc_TS(:, :, i) = AAO_tc_TS(:, :, i - 1) + AAO_t_TS(:, :, i);
else
AAO_tc_TS(:, :, i) = AAO_t_TS(:, :, i);
end
% In AAO_tc_TS, if any element > 1 then element = 1
check4 = AAO_tc_TS(:, :, i);
check4(check4 > 1) = 1;
AAO_tc_TS(:, :, i) = check4;

ANFV_tc_TS(:, :, i) = ANFV_tc;
```

```
ACFV_tc_TS(:, :, i) = ACFV_tc;
end
```

APPENDIX E

DYNAMICS MODEL ANALYSIS M-FILE

```
%% Model of Ecological Network Propagation Dynamics
%  Analysis Activities m-file

% We describe and develop the analysis activities program code
% in the following order:
%   > Network input, output, stock, and flow analysis
%   > Network propagation event analysis
%   > Path length analysis
%   > Indirect effects analysis
%   > Network connectivity analysis

%% > Network input, output, stock, and flow analysis

% An important vehicle for depicting network input, output, stock,
% and flow dynamics is the network node-and-link propagation flow diagram.
% Variables AAO_t_TS, AAO_tc_TS, INSL_t_TS, INSL_tc_TS, PNL_t_TS,
% PNL_tc_TS, OPNL_t_TS, and OPNL_tc_TS are used to produce these diagrams.
% Cumulative flow value diagrams provide an "adjacency matrix" depiction of
% network flow.  Variables ANFV_tc_TS and ACFV_tc_TS can be used to
% generate node flow value diagrams and compartment flow value diagrams,
% respectively.

% Network input, output, and stock histories can be plotted and displayed
% on 3-D grid bar/stack charts.  We use an appropriate MATLAB
% 3-D/discrete-surface plotting capability.  Some or all of the
% following cumulative variables can be plotted:
%  INVG_tc_TS = input-node-value-grid multidimensional array (10x10xNumTS)
%  ICV_tc_TS = input-compartment-value multidimensional vector (1x10xNumTS)
%  SIV_tc_TS = system-input-value vector (1xNumTS)
%  ONVG_tc_TS = output-node-value-grid multidimensional array (10x10xNumTS)
%  OCV_tc_TS = output-compartment-value multidimensional vector(1x10xNumTS)
%  SOV_tc_TS = system-output-value vector (1xNumTS)
%  NSVG_tc_TS = node-stock-value-grid multidimensional array (10x10xNumTS)
%  CSV_tc_TS = compartment-stock-value multidimensional vector (1x10xNumTS)
%  SSV_tc_TS = system-stock-value vector (1xNumTS)

% The production of these diagrams/graphs is accomplished in the
% graphics generation m-file.

%% > Network propagation event analysis

% For the entire simulation run (after all time steps), perform over-time
% network propagation event analysis.  In the master m-file, we have
% calculated network propagation event size per-time-step and have stored
% the data in vector NetPESize_TS.
```

```
%% Develop network propagation event time series
% Create network propagation event time series (event size vs. time).
% Vector NetPESize_TS provides the data.

% Time series = NetPESize_TS
% We can plot and analyze the time series with or without
% non-propagation events included.

% With non-propagation time steps:
NetPESize_TSwnp = NetPESize_TS;

% Without non-propagation time steps:
% remove non-propagation time steps (size = 1) from NetPESize_TS
NetPESize_TS(NetPESize_TS == 1) = [ ];

%% Develop network propagation event distribution
% Develop a network propagation event distribution (number of events vs.
% size of events) as follows:
%   Define a size interval (>= 1) and partition the NetPESize_TS domain
%   into intervals.
%   Count the number of events in each interval.
%   Generate a distribution with the ordered event size intervals as
%   abscissa and the number of events in each of those size intervals
%   as ordinate.
% Create a scalar variable that specifies the size interval.
% Create a temporary scalar variable that specifies the number
% of intervals.
% Create a distribution matrix variable in which column 1 contains
% the ordered event size intervals and column 2 contains the number
% of events in each of those size intervals.

% nsi = network propagation event size interval (scalar)
% nintervals = number of intervals (scalar) (temporary variable)
% NetPEDistr = network propagation event distribution matrix
% (#size intervals x 2)

% set and initialize variables
nsi = 10;
nintervals = ceil(max(NetPESize_TS) / nsi);    % rounded up
NetPEDistr = zeros(nintervals, 2);
% populate distribution matrix
for j = 1: nintervals
    NetPEDistr(j, 1) = j * nsi;
    NetPEDistr(j, 2) = sum((j - 1) * nsi < NetPESize_TS & ...
        NetPESize_TS <= j * nsi);
end

% in NetPEDistr, remove "x,y pairs" (rows) with zero propagation events
% (i.e., with zero in column 2)
% more time steps (samples) would likely fill zero-valued intervals
NetPEDistr(find(NetPEDistr(:,2) == 0),:) = [ ];

%% Develop network propagation event frequency spectrum
% Take the discrete Fourier transform (DFT) of the event time series
```

```
% to obtain its frequency spectrum.  Use the MATLAB fast Fourier transform
% function (fft) to accomplish that.  The procedure follows.

% NetPESize_TS = original NetPESize_TS with non-propagation events removed
% Lsig = length of signal time series (scalar)
% NFFT = next power of 2 up from Lsig (scalar)
% PEdft = propagation event discrete Fourier transform vector (1 x NFFT)
% PEssAmp = propagation event single-sided amplitude vector (1 x NFFT/2+1)
% normfreq = vector of normalized frequency values (1 x NFFT/2+1)
% numpoints = number of frequency spectrum points for partial spectrum
% (scalar) (the maximum number of points available = NFFT/2+1)
% freq = vector of normalized frequency values for partial spectrum
% (1 x numpoints)
% amp = vector of amplitude values for partial spectrum (1 x numpoints)

Lsig = size(NetPESize_TS, 2);
NFFT = 2^nextpow2(Lsig);
PEdft = fft(NetPESize_TS, NFFT)/Lsig;
PEssAmp = 2*abs(PEdft(1:NFFT/2+1));
normfreq = 100*linspace(0,1,NFFT/2+1);
numpoints = 50;
freq = normfreq(1:numpoints);
amp = PEssAmp(1:numpoints);

% plot the full frequency spectrum
% plot(normfreq, PEssAmp)
% title('Single-Sided Network Propagation Event Frequency Spectrum')
% xlabel('Normalized Frequency (Hz)')
% ylabel('Amplitude')

% plot partial frequency spectrum
% plot(freq, amp)
% title('Network Propagation Event Frequency Spectrum (Partial)')
% xlabel('Normalized Frequency (Hz)')
% ylabel('Amplitude')

% The resulting plots certainly suggest a "1/f noise" spectrum and fractal
% behavior in time.  I cannot confirm this with a straight-line loglog
% plot.  The accuracy of the Fourier transform data is not high enough for
% that.

% The accuracy of these discrete Fourier transform computations is limited
% by the relatively short length of the signal time series.  Much improved
% accuracy would result from signal lengths on the order of 10,000 or even
% 25,000.  Such values, however, are way beyond the capacity of my
% computer.

%% > Path length analysis

% Perform network path length analysis for the entire simulation run
% (after all time steps).
% Develop a path length time series and path length distribution.
% Based on the path length equations that I have derived (see Chapter 5
% of the dissertation document) and the data generated by the master
% m-file, path length exhibits punctuated dynamics.  We need a measure
```

```
% that effectively captures those dynamics in the time series and
% distribution.  A good measure is the maximum path length achieved at
% each time step.  We develop a maximum-path-length time series and
% distribution for the entire simulation run.
% Note that our approach here has both spatial and temporal dimensions.
% We investigate individual events at points in time rather than
% cumulative activity over time.
% Cunulative path length distributions depict over-time average activity.
% Such distributions are not particularly meaningful in a
% punctuated dynamics environment.

% Also perform path length analysis at each time step.
% Develop individual-time-step and cumulative (from time step one to the
% current time step) path length distributions.
% Each path length distribution provides the number of paths vs.
% path length.  The equations for the path length distributions are given
% in Chapter 5 of the dissertation document.

% The source data needed for path length analysis has been generated
% in the master m-file and stored as variables NumTS, NumCStages_TS,
% nnpe_s_S_TS, nopn_s_S_TS, and npfq:
%  NumTS = number of model simulation time steps
%  NumCStages_TS = total number of propagation stages in each completed
%  time step
%  nnpe_s_S_TS = total number of propagating nodes per stage per time step
%  nopn_s_S_TS = number of output propagating nodes per stage per time step
%  npfq = node propagation flow quantity

%% Develop path length time series for the entire simulation run
% Develop the path length time series (maximum path length
% at each time step vs. time) for the entire simulation run.

% Define the time series vector (PLmax_TS) (1x NumTS) that provides the
% maximum path length in each time step.
% Define a scalar (PLltmax) that provides the long-term maximum
% path length in all time steps.
PLmax_TS = NumCStages_TS;
PLltmax = max(PLmax_TS);

% Time series = PLmax_TS
% We can plot and analyze the time series with or without
% non-propagation events included.

% With non-propagation time steps:
PLmax_TSwnp = PLmax_TS;

% Without non-propagation time steps:
% remove non-propagation time steps (path length = 0) from PLmax_TS
PLmax_TS(PLmax_TS == 0) = [ ];
% in the spatial box-covering approach, such elements are sometimes
% not removed -- but probably should be

% Note that PLltmax is an upper bound for the network diameter
% of the simulation-run network.
```

```
%% Develop path length distribution for the entire simulation run
% Develop the path length distribution (number of events vs.
% maximum path length) for the entire simulation run.
% Note that our approach here (and the resulting distribution) is a
% temporal analog of the spatial box-covering approach to developing
% path length distribution.  See Chapter 4 of the dissertation document for
% a description of the spatial box-covering approach.

% Create a two-column path length distribution matrix (MaxPLDistr).
% Column 1 contains the path length intervals and column 2 contains
% the number of events in each of those length intervals.
% Create a scalar variable that specifies the length interval.
% Create a temporary scalar variable that specifies the number
% of intervals.

% plsi = path length size interval (scalar)
% nplintervals = number of path length intervals (scalar)
% (temporary variable)
% MaxPLDistr = maximum path length distribution matrix
% (#size intervals x 2)

% set and initialize variables
plsi = 1;
nplintervals = ceil(max(PLmax_TS) / plsi);    % rounded up
MaxPLDistr = zeros(nplintervals, 2);
% populate distribution matrix
for j = 1: nplintervals
    MaxPLDistr(j, 1) = j * plsi;
    MaxPLDistr(j, 2) = sum((j - 1) * plsi < PLmax_TS & ...
        PLmax_TS <= j * plsi);
end

% in MaxPLDistr, remove "x,y pairs" (rows) with zero occurrences for the
% length interval (i.e., with zero in column 2)
% more time steps (samples) would likely fill zero-valued intervals
MaxPLDistr(find(MaxPLDistr(:,2) == 0),:) = [ ];

% After developing individual-time-step path length distribution data
% (next), we will develop another time-integrated path length distribution
% using mean path length at each individual time step.
% We will compare the mean-path-length time series and distribution with
% the maximum-path-length time series and distribution (developed above)
% and observe the relative dynamics.

%% Develop individual-time-step path length distributions
% Create a multidimensional array that provides the two-column path length
% distribution matrix for each individual time step per time step.
% In each matrix, column 1 contains the path length and column 2 contains
% the number of paths for each of those lengths.
% For matrix rows greater than the maximum path length for that time step,
% column 2 contains zeros.
% For non-propagation time steps, matrix column 2 contains all zeros.
% PLDistr_t_TS = path length distribution per time step
% (PLltmax x 2 x NumTS)
```

```
% initialization
PLDistr_t_TS = zeros(PLltmax, 2, NumTS);
for ii = 1: PLltmax
    PLDistr_t_TS(ii, 1, :) = ii;
end
% populate distribution array
for i = 1: NumTS
    for j = 1: PLltmax
        PLDistr_t_TS(j, 2, i) = npfq * sum(nnpe_s_S_TS(1, j : end, i)) -...
            sum(nopn_s_S_TS(1, j : end, i));
    end
end

% can find propagating time steps and their maximum path lengths
% from vector NumCStages_TS
% can find propagating time steps and their network propagation event size
% from vector NetPESize_TSwnp
% before plotting the distribution for that time step, remove the rows
% that contain zero in column 2 (if any)

% Each of these distributions is for a single network propagation event.
% Each distribution (normal coordinates plot) is monotonically decreasing
% like a power-law curve, but falls more slowly than a power-law curve.
% The log-log plot is not a straight line.
% It appears that single network propagation events do not exhibit
% path length power-law/fractal behavior.

%% Develop mean path length time series for the entire simulation run
% This development comes at this point in the sequence because variable
% PLDistr_t_TS is required.

% Develop the mean path length time series (mean path length
% at each time step vs. time) for the entire simulation run.

% Define variables PLmean_TS, PLsum, and PLtotal.
% PLmean_TS = time series vector that provides the mean path length
% in each individual time step (1x NumTS)
% PLsum = weighted sum of path lengths in an individual time step (scalar)
% (temporary variable)
% PLtotal = total number of paths in an individual time step (scalar)
% (temporary variable)

% initialization and allocation
PLmean_TS = zeros(1,NumTS);

% populate time series vector
for i=1:NumTS  % time steps
    PLsum = 0;
    for j=1:PLltmax  % rows of PLDistr_t_TS
        PLsum = PLsum + (PLDistr_t_TS(j,1,i) * PLDistr_t_TS(j,2,i));
    end
    PLtotal = sum(PLDistr_t_TS(:,2,i));
    PLmean_TS(i) = PLsum / PLtotal;
end
```

```
% in time steps with no propagation, the above expression
% yields 0/0 or NaN
% replace the NaN elements with zeros
PLmean_TS(isnan(PLmean_TS)) = 0;

% Time series = PLmean_TS
% We can plot and analyze the time series with or without
% non-propagation events included.

% With non-propagation time steps:
PLmean_TSwnp = PLmean_TS;

% Without non-propagation time steps:
% remove non-propagation time steps (path length = 0) from PLmean_TS
PLmean_TS(PLmean_TS == 0) = [ ];

%% Develop mean path length distribution for the entire simulation run
% Develop the mean path length distribution (number of events vs.
% mean path length) for the entire simulation run.

% Create a two-column path length distribution matrix (MeanPLDistr).
% Column 1 contains the path length intervals and column 2 contains
% the number of events in each of those length intervals.
% Create a scalar variable that specifies the length interval.
% Create a temporary scalar variable that specifies the number
% of intervals.

% mplsi = mean path length size interval (scalar)
% nmplintervals = number of mean path length intervals (scalar)
% (temporary variable)
% MeanPLDistr = mean path length distribution matrix (#size intervals x 2)

% set and initialize variables
% use the next statement only if mplsi = 1
% PLmean_TS = round(PLmean_TS);
mplsi = 0.25;
nmplintervals = ceil(max(PLmean_TS) / mplsi);    % rounded up
MeanPLDistr = zeros(nmplintervals, 2);
% populate distribution matrix
for j = 1: nmplintervals
    MeanPLDistr(j, 1) = j * mplsi;
    MeanPLDistr(j, 2) = sum((j - 1) * mplsi < PLmean_TS & ...
        PLmean_TS <= j * mplsi);
end

% in MeanPLDistr, remove "x,y pairs" (rows) with zero occurrences for the
% length interval (i.e., with zero in column 2)
% more time steps (samples) would likely fill zero-valued intervals
MeanPLDistr(find(MeanPLDistr(:,2) == 0),:) = [ ];

%% Develop cumulative path length distributions
% Create another multidimensional array that provides the
% over-time-cumulative two-column path length distribution matrix
% per time step.
% In each matrix, column 1 contains the path length and column 2 contains
```

```
% the number of paths for each of those lengths.
% PLDistr_tc_TS = over-time-cumulative path length distribution
% per time step (PLltmax x 2 x NumTS)

% initialization
PLDistr_tc_TS = zeros(PLltmax, 2, NumTS);
for ii = 1: PLltmax
    PLDistr_tc_TS(ii, 1, :) = ii;
end
% populate distribution array
for i = 1: NumTS
    if i > 1
        PLDistr_tc_TS(:, 2, i) = PLDistr_tc_TS(:, 2, i - 1) + ...
            PLDistr_t_TS(:, 2, i);
    else
        PLDistr_tc_TS(:, 2, i) = PLDistr_t_TS(:, 2, i);
    end
end

% Each of these distributions is cumulative (from time step one to the
% current time step) and is essentially an over-time average path length
% distribution.  Each distribution (normal coordinates plot) is
% monotonically decreasing like a power-law curve, but falls more slowly
% than a power-law curve (and, interestingly, falls more rapidly than
% individual-time-step distributions).
% The log-log plot is not a straight line.
% These distributions do not exhibit power-law/fractal behavior.
% In a punctuated dynamics context, distributions of path length averages
% may not be meaningful.
% The cumulative values of the various path lengths, however, are useful.

%% > Indirect effects analysis

% Perform over-time and per-time-step network indirect effects analysis.
% The source data needed to do this has been generated in the above
% path length analysis program cells.

% Calculate the Direct Effects Ratio (DER), Indirect Effects Ratio (IER),
% and Indirect Effects Index (IEI) using the expressions provided in
% Chapter 5 of the dissertation document.
% DER equals the number of paths of length 1 divided by the total number
% of paths.  IER equals the number of paths of length > 1 divided by the
% total number of paths.  IEI equals the number of paths of length > 1
% divided by the number of paths of length 1.
% Generate time series for the direct effects and indirect effects
% indicators – both the individual-time-step values vs. time and the
% over-time-cumulative values vs. time.

% Develop an indirect path quantity time series and distribution
% as well as a direct path quantity time series and distribution
% for the simulation run.
% Each time series provides the number of paths at each time step vs. time.

% Also develop a cumulative indirect path quantity time series and a
% cumulative direct path quantity time series for the simulation run.
```

```
% Each time series provides the cumulative number of paths at each
% time step vs. time.

%% Develop individual-time-step time series for the indicators
% Create vectors that provide time series (individual-time-step values
% vs. time) for the direct effects and indirect effects indicators.
% DER_t_TS = Direct Effects Ratio for each individual time step in
% the simulation run per time step (1 x NumTS)
% IER_t_TS = Indirect Effects Ratio for each individual time step in
% the simulation run per time step (1 x NumTS)
% IEI_t_TS = Indirect Effects Index for each individual time step in
% the simulation run per time step (1 x NumTS)

% preallocation and initialization
DER_t_TS = zeros(1, NumTS);
IER_t_TS = zeros(1, NumTS);
IEI_t_TS = zeros(1, NumTS);

% calculate the vector values
for i = 1: NumTS
    DER_t_TS(i) = PLDistr_t_TS(1, 2, i) / ...
        sum(PLDistr_t_TS(1:end, 2, i));
    IER_t_TS(i) = sum(PLDistr_t_TS(2:end, 2, i)) / ...
        sum(PLDistr_t_TS(1:end, 2, i));
    IEI_t_TS(i) = sum(PLDistr_t_TS(2:end, 2, i)) / ...
        PLDistr_t_TS(1, 2, i);
end

% in time steps with no propagation, the above expressions
% yield 0/0 or NaN
% replace the NaN elements with zeros
DER_t_TS(isnan(DER_t_TS)) = 0;
IER_t_TS(isnan(IER_t_TS)) = 0;
IEI_t_TS(isnan(IEI_t_TS)) = 0;

%% Develop cumulative time series for the indicators
% Create vectors that provide time series (over-time-cumulative values
% vs. time) for the direct effects and indirect effects indicators.
% DER_tc_TS = per-time-step over-time-cumulative Direct Effects Ratio
% (1 x NumTS)
% IER_tc_TS = per-time-step over-time-cumulative Indirect Effects Ratio
% (1 x NumTS)
% IEI_tc_TS = per-time-step over-time-cumulative Indirect Effects Index
% (1 x NumTS)

% preallocation and initialization
DER_tc_TS = zeros(1, NumTS);
IER_tc_TS = zeros(1, NumTS);
IEI_tc_TS = zeros(1, NumTS);

% calculate the vector values
for i = 1: NumTS
    DER_tc_TS(i) = PLDistr_tc_TS(1, 2, i) / sum(PLDistr_tc_TS(:, 2, i));
    IER_tc_TS(i) = sum(PLDistr_tc_TS(2:end, 2, i)) / ...
        sum(PLDistr_tc_TS(:, 2, i));
```

```
    IEI_tc_TS(i) = sum(PLDistr_tc_TS(2:end, 2, i)) / ...
        PLDistr_tc_TS(1, 2, i);
end

% replace the NaN elements with zeros
DER_tc_TS(isnan(DER_tc_TS)) = 0;
IER_tc_TS(isnan(IER_tc_TS)) = 0;
IEI_tc_TS(isnan(IEI_tc_TS)) = 0;

%% Develop indirect path quantity time series
% Develop the simulation run indirect path quantity time series
% (number of indirect paths at each time step vs. time).
% IndPathQ_TS = indirect-path-quantity time series vector (1 x NumTS)

% preallocation and initialization
IndPathQ_TS = zeros(1, NumTS);

% calculate the vector values
for i = 1: NumTS
    IndPathQ_TS(i) = sum(PLDistr_t_TS(2:end, 2, i));
end

% We can plot and analyze the time series with or without
% "zero events" (events that have zero indirect paths) included.

% With zero-event time steps:
IndPathQ_TSwze = IndPathQ_TS;

% Without zero-event time steps:
% remove zero-event time steps (quantity = 0) from IndPathQ_TS
IndPathQ_TS(IndPathQ_TS == 0) = [ ];

%% Develop indirect path quantity distribution
% Develop an indirect path quantity distribution (number of
% propagation events vs. quantity of indirect paths) as follows:
%   Define a size interval (>= 1) and partition the IndPathQ_TS domain
%   into intervals.
%   Count the number of events in each interval.
%   Generate a distribution with the ordered event size intervals as
%   abscissa and the number of events in each of those size intervals
%   as ordinate.
% Create a scalar variable that specifies the size interval.
% Create a temporary scalar variable that specifies the number
% of intervals.
% Create a distribution matrix variable in which column 1 contains
% the ordered event size intervals and column 2 contains the number
% of events in each of those size intervals.

% ipsi = indirect path size interval (scalar)
% nipintervals = number of indirect path intervals (scalar)
% (temporary variable)
% IndPathQDistr = indirect path quantity distribution matrix
% (#size intervals x 2)

% set and initialize variables
```

```
ipsi = 100;
nipintervals = ceil(max(IndPathQ_TS) / ipsi);    % rounded up
IndPathQDistr = zeros(nipintervals, 2);
% populate distribution matrix
for j = 1: nipintervals
    IndPathQDistr(j, 1) = j * ipsi;
    IndPathQDistr(j, 2) = sum((j - 1) * ipsi < IndPathQ_TS & ...
        IndPathQ_TS <= j * ipsi);
end

% in IndPathQDistr, remove "x,y pairs" (rows) with "zero events"
% (i.e., with zero in column 2)
% more time steps (samples) would likely fill zero-valued intervals
IndPathQDistr(find(IndPathQDistr(:,2) == 0),:) = [ ];

%% Develop direct path quantity time series
% Develop the simulation run direct path quantity time series
% (number of direct paths at each time step vs. time).
% DirPathQ_TS = direct-path-quantity time series vector (1 x NumTS)

% preallocation and initialization
DirPathQ_TS = zeros(1, NumTS);

% calculate the vector values
for i = 1: NumTS
    DirPathQ_TS(i) = PLDistr_t_TS(1, 2, i);
end

% We can plot and analyze the time series with or without
% "zero events" (events that have zero direct paths) included.

% With zero-event time steps:
DirPathQ_TSwze = DirPathQ_TS;

% Without zero-event time steps:
% remove zero-event time steps (quantity = 0) from DirPathQ_TS
DirPathQ_TS(DirPathQ_TS == 0) = [ ];

%% Develop direct path quantity distribution
% Develop a direct path quantity distribution (number of
% propagation events vs. quantity of direct paths) as follows:
%   Define a size interval (>= 1) and partition the DirPathQ_TS domain
%   into intervals.
%   Count the number of events in each interval.
%   Generate a distribution with the ordered event size intervals as
%   abscissa and the number of events in each of those size intervals
%   as ordinate.
% Create a scalar variable that specifies the size interval.
% Create a temporary scalar variable that specifies the number
% of intervals.
% Create a distribution matrix variable in which column 1 contains
% the ordered event size intervals and column 2 contains the number
% of events in each of those size intervals.

% dpsi = direct path size interval (scalar)
```

```
% ndpintervals = number of direct path intervals (scalar)
% (temporary variable)
% DirPathQDistr = direct path quantity distribution matrix
% (#size intervals x 2)

% set and initialize variables
dpsi = 10;
ndpintervals = ceil(max(DirPathQ_TS) / dpsi);    % rounded up
DirPathQDistr = zeros(ndpintervals, 2);
% populate distribution matrix
for j = 1: ndpintervals
    DirPathQDistr(j, 1) = j * dpsi;
    DirPathQDistr(j, 2) = sum((j - 1) * dpsi < DirPathQ_TS & ...
        DirPathQ_TS <= j * dpsi);
end

% in DirPathQDistr, remove "x,y pairs" (rows) with "zero events"
% (i.e., with zero in column 2)
% more time steps (samples) would likely fill zero-valued intervals
DirPathQDistr(find(DirPathQDistr(:,2) == 0),:) = [ ];

%% Develop cumulative indirect path and direct path quantity time series
% Develop the cumulative indirect path quantity time series (cumulative
% number of indirect paths at each time step vs. time).
% CuIndPathQ_TS = cumulative-indirect-path-quantity time series vector
% (1 x NumTS)

% preallocation and initialization
CuIndPathQ_TS = zeros(1, NumTS);

% calculate the vector values
for i = 1: NumTS
    CuIndPathQ_TS(i) = sum(PLDistr_tc_TS(2:end, 2, i));
end

% Develop the cumulative direct path quantity time series (cumulative
% number of direct paths at each time step vs. time).
% CuDirPathQ_TS = cumulative-direct-path-quantity time series vector
% (1 x NumTS)

% preallocation and initialization
CuDirPathQ_TS = zeros(1, NumTS);

% calculate the vector values
for i = 1: NumTS
    CuDirPathQ_TS(i) = PLDistr_tc_TS(1, 2, i);
end

%% > Network connectivity analysis

% Perform comprehensive node degree analysis.  Develop individual-time-step
% node degree vector arrays and node degree grid arrays.  Note that along
% with network propagation events, path length, and indirect effects - node
% degree exhibits punctuated dynamics.  That can be seen by observing the
% node degree grids at different points in time.  We need to develop node
```

```
% degree time series and distributions that capture those dynamics in an
% integrated fashion over space and time.  By analogy with our path length
% development, a measure that effectively captures the dynamics is the
% maximum node degree achieved at each time step.  We therefore develop a
% simulation-run maximum-node-degree time series and distribution.

% Another measure that may capture the dynamics is the node mean degree
% achieved at each time step.  We calculate node mean degree and (the
% related) network connection density.  We then develop a simulation-run
% node-mean-degree time series and distribution.

% Also develop individual-time-step node degree distributions and
% cumulative (from time step one to the current time step) node degree
% distributions.  Each distribution provides the number of nodes with
% degree x vs. x.

% To cap the network connectivity investigations, perform
% network critical connectivity/percolation analysis.

% The source data needed to do this analysis has been generated in the
% master m-file and stored as arrays AAO_t_TS and AAO_tc_TS:
%   AAO_t_TS = operational node adjacency multidimensional array that
%   provides an operational adjacency matrix for each individual time step
%   per-time-step (100x100x NumTS)
%   AAO_tc_TS = operational node adjacency multidimensional array that
%   provides an over-time-cumulative operational adjacency matrix
%   per-time-step (100x100x NumTS)

%% Develop individual-time-step node degree vector arrays
% Develop multidimensional arrays that provide the out-degree, in-degree,
% and combined-degree (both out and in) of each node at each
% individual time step vs. time.
% NodeODeg_t_TS = node-out-degree multidimensional array (1x100xNumTS)
% that provides a node out-degree vector for each individual time step
% per time step
% NodeIDeg_t_TS = node-in-degree multidimensional array (1x100xNumTS)
% that provides a node in-degree vector for each individual time step
% per time step
% NodeCDeg_t_TS = node-combined-degree multidimensional array (1x100xNumTS)
% that provides a node combined-degree vector for each individual time step
% per time step

% preallocation and initialization
NodeODeg_t_TS = zeros(1,100, NumTS);
NodeIDeg_t_TS = zeros(1,100, NumTS);
NodeCDeg_t_TS = zeros(1,100, NumTS);

% calculate the vector values
% external connections (inputs and outputs) with the environment are not
% counted in these node degree calculations
for i = 1: NumTS
    NodeODeg_t_TS(1, :, i) = sum(AAO_t_TS(:, :, i));
    NodeIDeg_t_TS(1, :, i) = sum(transpose(AAO_t_TS(:, :, i)));
    NodeCDeg_t_TS(1, :, i) = NodeODeg_t_TS(1, :, i) + ...
        NodeIDeg_t_TS(1, :, i);
```

```
end

%% Develop individual-time-step node degree grid arrays
% Develop multidimensional arrays that provide out-degree, in-degree,
% and combined-degree (both out and in) node grids at each
% individual time step vs. time.
% NodeODegGrid_t_TS = node-out-degree-grid multidimensional array
% (10x10xNumTS) that provides a node out-degree grid for each
% individual time step per time step
% NodeIDegGrid_t_TS = node-in-degree-grid multidimensional array
% (10x10xNumTS) that provides a node in-degree grid for each
% individual time step per time step
% NodeCDegGrid_t_TS = node-combined-degree-grid multidimensional array
% (10x10xNumTS) that provides a node combined-degree grid for each
% individual time step per time step

% preallocation and initialization
NodeODegGrid_t_TS = zeros(10,10, NumTS);
NodeIDegGrid_t_TS = zeros(10,10, NumTS);
NodeCDegGrid_t_TS = zeros(10,10, NumTS);

% calculate the grid values
% external connections (inputs and outputs) with the environment are not
% counted in these node degree calculations
for i=1:NumTS    % time steps
    for j=1:10    % columns
        for k=1:10    % rows
            NodeODegGrid_t_TS(k, j, i) = NodeODeg_t_TS(1, k+10*(j-1), i);
            NodeIDegGrid_t_TS(k, j, i) = NodeIDeg_t_TS(1, k+10*(j-1), i);
            NodeCDegGrid_t_TS(k, j, i) = NodeCDeg_t_TS(1, k+10*(j-1), i);
        end
    end
end

%% Develop node degree time series for the entire simulation run
% Develop a set of node degree time series (maximum node degree
% at each time step vs. time) for the entire simulation run.

% Define the time series vectors that provide the maximum node degrees
% in each time step:
% ODmax_t_TS = vector (1x NumTS) that provides the maximum out-degree
% in each time step per time step
% IDmax_t_TS = vector (1x NumTS) that provides the maximum in-degree
% in each time step per time step
% CDmax_t_TS = vector (1x NumTS) that provides the maximum combined-degree
% in each time step per time step

% Define scalars that provide the long-term maximum node degrees
% in all time steps:
% ODltmax_t = long-term maximum out-degree in all time steps (scalar)
% IDltmax_t = long-term maximum in-degree in all time steps (scalar)
% CDltmax_t = long-term maximum combined-degree in all time steps (scalar)

% preallocation and initialization
ODmax_t_TS = zeros(1, NumTS);
```

```
IDmax_t_TS = zeros(1, NumTS);
CDmax_t_TS = zeros(1, NumTS);

% calculate the variable values
for i = 1: NumTS
    ODmax_t_TS(i) = max(NodeODeg_t_TS(1, :, i));
    IDmax_t_TS(i) = max(NodeIDeg_t_TS(1, :, i));
    CDmax_t_TS(i) = max(NodeCDeg_t_TS(1, :, i));
end
ODltmax_t = max(ODmax_t_TS);
IDltmax_t = max(IDmax_t_TS);
CDltmax_t = max(CDmax_t_TS);

% We can plot and analyze the time series with or without
% non-propagation events included.

% With non-propagation time steps:
ODmax_t_TSwnp = ODmax_t_TS;
IDmax_t_TSwnp = IDmax_t_TS;
CDmax_t_TSwnp = CDmax_t_TS;

% Without non-propagation time steps:
% remove non-propagation time steps (node degree = 0) from the
% time series vectors:
ODmax_t_TS(ODmax_t_TS == 0) = [ ];
IDmax_t_TS(IDmax_t_TS == 0) = [ ];
CDmax_t_TS(CDmax_t_TS == 0) = [ ];

% We use the in-degree and combined-degree cases for our plots.

%% Develop node degree distributions for the entire simulation run
% Develop a set of node degree distributions (number of events vs. maximum
% node degree) for the entire simulation run.
% Note that our approach here has both spatial and temporal dimensions.
% We investigate individual events at points in time rather than
% cumulative activity over time.
% As suggested earlier, cunulative distributions are not particularly
% meaningful in a punctuated dynamics environment.

% Create two-column node degree distribution matrices.
% Column 1 contains the node degree intervals and column 2 contains
% the number of events in each of those intervals.
% The matrices are defined as:
% MaxODDistr = maximum out-degree distribution matrix (#size intervals x 2)
% MaxIDDistr = maximum in-degree distribution matrix (#size intervals x 2)
% MaxCDDistr = maximum combined-degree distribution matrix
% (#size intervals x 2)

% Create a scalar variable that specifies the node degree interval.
% Create a temporary scalar variable that specifies the number
% of intervals.
% mdsi = maximum node degree size interval (scalar)
% nmdintervals = number of maximum node degree intervals (scalar)
% (temporary variable)
```

```
% set and initialize out-degree variables
mdsi = 1;
nmdintervals = ceil(max(ODmax_t_TS) / mdsi);    % rounded up
MaxODDistr = zeros(nmdintervals, 2);
% populate out-degree distribution matrix
for j = 1: nmdintervals
    MaxODDistr(j, 1) = j * mdsi;
    MaxODDistr(j, 2) = sum((j - 1) * mdsi < ODmax_t_TS & ...
        ODmax_t_TS <= j * mdsi);
end

% in MaxODDistr, remove "x,y pairs" (rows) with zero occurrences for the
% degree interval (i.e., with zero in column 2)
% more time steps (samples) would likely help fill zero-valued intervals
MaxODDistr(find(MaxODDistr(:,2) == 0),:) = [ ];

% set and initialize in-degree variables
mdsi = 1;
nmdintervals = ceil(max(IDmax_t_TS) / mdsi);    % rounded up
MaxIDDistr = zeros(nmdintervals, 2);
% populate in-degree distribution matrix
for j = 1: nmdintervals
    MaxIDDistr(j, 1) = j * mdsi;
    MaxIDDistr(j, 2) = sum((j - 1) * mdsi < IDmax_t_TS & ...
        IDmax_t_TS <= j * mdsi);
end

% in MaxIDDistr, remove "x,y pairs" (rows) with zero occurrences for the
% degree interval (i.e., with zero in column 2)
% more time steps (samples) would likely help fill zero-valued intervals
MaxIDDistr(find(MaxIDDistr(:,2) == 0),:) = [ ];

% set and initialize combined-degree variables
mdsi = 1;
nmdintervals = ceil(max(CDmax_t_TS) / mdsi);    % rounded up
MaxCDDistr = zeros(nmdintervals, 2);
% populate combined-degree distribution matrix
for j = 1: nmdintervals
    MaxCDDistr(j, 1) = j * mdsi;
    MaxCDDistr(j, 2) = sum((j - 1) * mdsi < CDmax_t_TS & ...
        CDmax_t_TS <= j * mdsi);
end

% in MaxCDDistr, remove "x,y pairs" (rows) with zero occurrences for the
% degree interval (i.e., with zero in column 2)
% more time steps (samples) would likely help fill zero-valued intervals
MaxCDDistr(find(MaxCDDistr(:,2) == 0),:) = [ ];

% We use the in-degree and combined-degree cases for our plots.

%% Calculate node mean degree and network connection density
% Calculate node mean degree and network connection density for
% each individual time step and cumulatively.
% Node mean degree = number of edges divided by number of nodes
% Network connection density = node mean degree divided by
```

```
% (number of nodes – 1)

% For later comparisons, we calculate mean degree and network connection
% density for the underlying compartment network and for the
% "candidate" node network.

% Underlying compartment network values:
% CompMD = compartment mean degree (scalar)
% CompNetDen = compartment network connection density (scalar)
CompMD = sum(sum(A)) / 10;
% CompMD = 3.200
CompNetDen = CompMD / (10 - 1);
% CompNetDen = 0.3556

% "Candidate" node network values:
% CandNMD = "candidate" node mean degree (scalar)
% CandNetDen = "candidate" node network connection density (scalar)
CandNMD = sum(sum(AA)) / 100;
% CandNMD = 30.3500
CandNetDen = CandNMD / (100 - 1);
% CandNetDen = 0.3066

%% Calculate individual-time-step node mean degree and connection density
% NMD_t_TS = node mean degree for each individual time step per time step
% (1xNumTS)
% NetDen_t_TS = network density for each individual time step per time step
% (1xNumTS)

% preallocation and initialization
NMD_t_TS = zeros(1, NumTS);
NetDen_t_TS = zeros(1, NumTS);

% calculate the vector values
for i = 1: NumTS
    NMD_t_TS(i) = sum(sum(AAO_t_TS(:, :, i))) / 100;
    NetDen_t_TS(i) = NMD_t_TS(i) / (100 - 1);
end

%% Calculate cumulative node mean degree and connection density values
% NMD_tc_TS = over-time-cumulative node mean degree per time step
% (1xNumTS)
% NetDen_tc_TS = over-time-cumulative network density per time step
% (1xNumTS)

% preallocation and initialization
NMD_tc_TS = zeros(1, NumTS);
NetDen_tc_TS = zeros(1, NumTS);

% calculate the vector values
for i = 1: NumTS
    NMD_tc_TS(i) = sum(sum(AAO_tc_TS(:, :, i))) / 100;
    NetDen_tc_TS(i) = NMD_tc_TS(i) / (100 - 1);
end

%% Prepare individual-time-step node mean degree time series
```

```
% Time series = NMD_t_TS (from above)
% We can plot and analyze the time series with or without
% non-propagation events included.

% With non-propagation time steps:
NMD_t_TSwnp = NMD_t_TS;

% Without non-propagation time steps:
% remove non-propagation time steps (path length = 0) from PLmean_TS
NMD_t_TS(NMD_t_TS == 0) = [ ];

%% Develop node mean degree distribution for the entire simulation run
% Develop the node mean degree distribution (number of events vs.
% node mean degree) for the entire simulation run.

% Create a two-column node mean degree distribution matrix (NMDDistr).
% Column 1 contains the node mean degree intervals and column 2 contains
% the number of events in each of those mean degree intervals.
% Create a scalar variable that specifies the mean degree interval.
% Create a temporary scalar variable that specifies the number
% of intervals.

% nmdsi = node mean degree size interval (scalar)
% nnmdintervals = number of node mean degree intervals (scalar)
% (temporary variable)
% NMDDistr = node mean degree distribution matrix (#size intervals x 2)

% set and initialize variables
nmdsi = 0.25;
nnmdintervals = ceil(max(NMD_t_TS) / nmdsi);    % rounded up
NMDDistr = zeros(nnmdintervals, 2);
% populate distribution matrix
for j = 1: nnmdintervals
    NMDDistr(j, 1) = j * nmdsi;
    NMDDistr(j, 2) = sum((j - 1) * nmdsi < NMD_t_TS & ...
        NMD_t_TS <= j * nmdsi);
end

% in NMDDistr, remove "x,y pairs" (rows) with zero occurrences for the
% mean degree interval (i.e., with zero in column 2)
% more time steps (samples) would likely fill zero-valued intervals
NMDDistr(find(NMDDistr(:,2) == 0),:) = [ ];

%% Develop cumulative node degree arrays
% Consider the over-time-cumulative per-time-step case.
% Develop multidimensional arrays that provide the over-time-cumulative
% out-degree, in-degree, and combined-degree of each node vs. time.
% NodeODeg_tc_TS = node-out-degree multidimensional array (1x100xNumTS)
% that provides an over-time-cumulative node out-degree vector
% per time step
% NodeIDeg_tc_TS = node-in-degree multidimensional array (1x100xNumTS)
% that provides an over-time-cumulative node in-degree vector
% per time step
% NodeCDeg_tc_TS = node-combined-degree multidimensional array
% (1x100xNumTS) that provides an over-time-cumulative node combined-degree
```

```
% vector per time step

% preallocation and initialization
NodeODeg_tc_TS = zeros(1,100, NumTS);
NodeIDeg_tc_TS = zeros(1,100, NumTS);
NodeCDeg_tc_TS = zeros(1,100, NumTS);

% calculate the vector values
for i = 1: NumTS
    NodeODeg_tc_TS(1, :, i) = sum(AAO_tc_TS(:, :, i));
    NodeIDeg_tc_TS(1, :, i) = sum(transpose(AAO_tc_TS(:, :, i)));
    NodeCDeg_tc_TS(1, :, i) = NodeODeg_tc_TS(1, :, i) + ...
        NodeIDeg_tc_TS(1, :, i);
end

%% Develop additional cumulative variables
% Develop additional variables for the over-time-cumulative
% per-time-step case.
% Define vectors that provide the maximum degrees in each time step:
% ODmax_tc_TS = vector (1x NumTS) that provides the maximum cumulative
% out-degree per time step
% IDmax_tc_TS = vector (1x NumTS) that provides the maximum cumulative
% in-degree per time step
% CDmax_tc_TS = vector (1x NumTS) that provides the maximum cumulative
% combined-degree per time step
% Define scalars that provide the long-term maximum cumulative degrees
% across time steps:
% ODltmax_tc = long-term maximum cumulative out-degree across time steps
% (scalar)
% IDltmax_tc = long-term maximum cumulative in-degree across time steps
% (scalar)
% CDltmax_tc = long-term maximum cumulative combined-degree across
% time steps (scalar)

% preallocation and initialization
ODmax_tc_TS = zeros(1, NumTS);
IDmax_tc_TS = zeros(1, NumTS);
CDmax_tc_TS = zeros(1, NumTS);

% calculate the variable values
for i = 1: NumTS
    ODmax_tc_TS(i) = max(NodeODeg_tc_TS(1, :, i));
    IDmax_tc_TS(i) = max(NodeIDeg_tc_TS(1, :, i));
    CDmax_tc_TS(i) = max(NodeCDeg_tc_TS(1, :, i));
end
ODltmax_tc = max(ODmax_tc_TS);
IDltmax_tc = max(IDmax_tc_TS);
CDltmax_tc = max(CDmax_tc_TS);

%% Develop per-time-step node degree distributions
% Develop individual-time-step and over-time-cumulative network node degree
% distributions (number of nodes with degree x vs. x) from the node degree
% multidimensional arrays.  Develop out-degree, in-degree, and combined
% (both out and in) degree distributions as follows:
%   Define a degree size interval (>= 1) and partition the degree size
%   domain into those intervals.
```

```
%   Count the number of nodes in each interval.
%   Generate a distribution with the ordered node degree intervals as
%   abscissa and the number of nodes in each of those intervals as
%   ordinate.

%% Develop individual-time-step node degree distributions
% Develop distributions for the individual-time-step per-time-step case.
% Create a scalar variable that specifies the size interval.
% Create a temporary scalar variable that specifies the
% number of intervals.
% Create multidimensional arrays that provide a two-column degree
% distribution matrix for each individual time step per time step.
% In each matrix, column 1 contains the ordered node degree size intervals
% and column 2 contains the number of nodes in each of those
% size intervals.
% For matrix rows > ODmax_t_TS(i) or IDmax_t_TS(i) or CDmax_t_TS(i),
% respectively, column 2 contains zeros.

% dsi = degree size interval (scalar)
% ndintervals = number of degree intervals (scalar) (temporary variable)
% ODegDistr_t_TS = out-degree distribution per time step
% (#size intervals x 2 x NumTS)
% IDegDistr_t_TS = in-degree distribution per time step
% (#size intervals x 2 x NumTS)
% CDegDistr_t_TS = combined-degree distribution per time step
% (#size intervals x 2 x NumTS)

% set and initialize out-degree variables
dsi = 1;
ndintervals = ceil(ODltmax_t / dsi);    % rounded up
ODegDistr_t_TS = zeros(ndintervals, 2, NumTS);
% populate distribution matrix
for i = 1: NumTS
    for j = 1: ndintervals
        ODegDistr_t_TS (j, 1, i) = j * dsi;
        ODegDistr_t_TS (j, 2, i) = ...
            sum((j - 1) * dsi < NodeODeg_t_TS(1,:,i) ...
            & NodeODeg_t_TS(1,:,i) <= j * dsi);
    end
end

% set and initialize in-degree variables
dsi = 1;
ndintervals = ceil(IDltmax_t / dsi);    % rounded up
IDegDistr_t_TS = zeros(ndintervals, 2, NumTS);
% populate distribution matrix
for i = 1: NumTS
    for j = 1: ndintervals
        IDegDistr_t_TS (j, 1, i) = j * dsi;
        IDegDistr_t_TS (j, 2, i) = ...
            sum((j - 1) * dsi < NodeIDeg_t_TS(1,:,i) ...
            & NodeIDeg_t_TS(1,:,i) <= j * dsi);
    end
end

% set and initialize combined-degree variables
```

```
dsi = 1;
ndintervals = ceil(CDltmax_t / dsi);    % rounded up
CDegDistr_t_TS = zeros(ndintervals, 2, NumTS);
% populate distribution matrix
for i = 1: NumTS
    for j = 1: ndintervals
        CDegDistr_t_TS (j, 1, i) = j * dsi;
        CDegDistr_t_TS (j, 2, i) = ...
            sum((j - 1) * dsi < NodeCDeg_t_TS(1,:,i) ...
            & NodeCDeg_t_TS(1,:,i) <= j * dsi);
    end
end

% Use the in-degree case for plots.
% For a three-dimensional array, do not know a general way to remove
% "x,y pairs" (rows) with zero occurrences for the degree interval
% (i.e., with zero in column 2) from matrices that comprise the array.
% Do that individually for specific matrices at time steps of interest.

%% Develop cumulative node degree distributions
% Develop distributions for the over-time-cumulative per-time-step case.
% Create a scalar variable that specifies the size interval.
% Create a temporary scalar variable that specifies the
% number of intervals.
% Create multidimensional arrays that provide a two-column cumulative
% degree distribution matrix per time step.
% In each matrix, column 1 contains the ordered node degree size intervals
% and column 2 contains the number of nodes in each of those
% size intervals.
% For matrix rows > ODmax_tc_TS(i) or IDmax_tc_TS(i) or CDmax_tc_TS(i),
% respectively, column 2 contains zeros.

% dsi = degree size interval (scalar)
% ndintervals = number of degree intervals (scalar) (temporary variable)
% ODegDistr_tc_TS = cumulative out-degree distribution per time step
% (#size intervals x 2 x NumTS)
% IDegDistr_tc_TS = cumulative in-degree distribution per time step
% (#size intervals x 2 x NumTS)
% CDegDistr_tc_TS = cumulative combined-degree distribution per time step
% (#size intervals x 2 x NumTS)

% set and initialize out-degree variables
dsi = 1;
ndintervals = ceil(ODltmax_tc / dsi);    % rounded up
ODegDistr_tc_TS = zeros(ndintervals, 2, NumTS);
% populate distribution matrix
for i = 1: NumTS
    for j = 1: ndintervals
        ODegDistr_tc_TS (j, 1, i) = j * dsi;
        ODegDistr_tc_TS (j, 2, i) = ...
            sum((j-1) * dsi < NodeODeg_tc_TS(1,:,i) ...
            & NodeODeg_tc_TS(1,:,i) <= j * dsi);
    end
end

% set and initialize in-degree variables
```

```
dsi = 1;
ndintervals = ceil(IDltmax_tc / dsi);    % rounded up
IDegDistr_tc_TS = zeros(ndintervals, 2, NumTS);
% populate distribution matrix
for i = 1: NumTS
    for j = 1: ndintervals
        IDegDistr_tc_TS (j, 1, i) = j * dsi;
        IDegDistr_tc_TS (j, 2, i) = ...
            sum((j-1) * dsi < NodeIDeg_tc_TS(1,:,i) ...
            & NodeIDeg_tc_TS(1,:,i) <= j * dsi);
    end
end

% set and initialize combined-degree variables
dsi = 1;
ndintervals = ceil(CDltmax_tc / dsi);    % rounded up
CDegDistr_tc_TS = zeros(ndintervals, 2, NumTS);
% populate distribution matrix
for i = 1: NumTS
    for j = 1: ndintervals
        CDegDistr_tc_TS (j, 1, i) = j * dsi;
        CDegDistr_tc_TS (j, 2, i) = ...
            sum((j-1) * dsi < NodeCDeg_tc_TS(1,:,i) ...
            & NodeCDeg_tc_TS(1,:,i) <= j * dsi);
    end
end

%% Critical connectivity and percolation
% At any given time step, the network may achieve critical connectivity
% and percolate.
% Create variables that provide the number of nodes linked together and
% the fractional size of the resulting candidate "giant component" at each
% individual time step.  For propagation at a particular time step,
% all involved nodes are linked together.  The size of the candidate
% giant component equals the number of nodes linked together divided by
% the total number of nodes in the network.
% NumLN_t_TS = number of linked nodes for each individual time step
% per time step (1xNumTS)
% CompSize_t_TS = candidate giant component size for each individual
% time step per time step (1xNumTS)

% preallocation and initialization
NumLN_t_TS = zeros(1, NumTS);
CompSize_t_TS = zeros(1, NumTS);

% I have developed three approaches for calculating the
% vector values – as follows.
% Since we'll use the third, the first two are "commented."

% calculate the vector values
% for i = 1: NumTS
%     result = find(AAO_t_TS(:, 1, i) == 1);
%     for j = 2 : 100    % find all to-nodes without repeats
%         result = union(result, find(AAO_t_TS(:, j, i) == 1));
%     end
%     % make sure the input node is included
```

```
%     result = union(result, find(INSL_t_TS(:, :, i) == 1));
%     NumLN_t_TS(i) = size(result, 1);
%     CompSize_t_TS(i) = NumLN_t_TS(i) / 100;
% end

% calculate the vector values
% for i = 1: NumTS
%   temp1 = find(sum(transpose(AAO_t_TS(:, :, i))) > 0);  % find to-nodes
%   temp2 = find(INSL_t_TS(:, :, i) == 1);  % find the input-node
%   result = union(temp1, temp2);  % all linked nodes without repeats
%   NumLN_t_TS(i) = size(result, 1);
%   CompSize_t_TS(i) = NumLN_t_TS(i) / 100;
% end

% calculate the vector values
for i = 1: NumTS
  temp1 = find(sum(transpose(AAO_t_TS(:, :, i))) > 0);  % find the to-nodes
  temp2 = find(sum(AAO_t_TS(:, :, i)) > 0);  % find the from-nodes
  result = union(temp1, temp2);  % all linked nodes without repeats
  NumLN_t_TS(i) = size(result, 2);
  CompSize_t_TS(i) = NumLN_t_TS(i) / 100;
end
```

APPENDIX F

DYNAMICS MODEL GRAPHICS M-FILE

```
%% Dynamics Model Graphics

%% Network Node-and-Link Propagation Flow Graphics

%% Time series with selected time steps
% Figure name:  PF1
% Figure title in doc:  Network Propagation Time Series

% plot time series
plot(NetPESize_TSwnp)
hold on

% define selection circles
xx = [2 28 32 64 152 204 237 514];
yy = NetPESize_TSwnp(xx);

% plot selection circles
plot(xx,yy,'o','LineWidth',2, 'MarkerEdgeColor',[.824 .004 .216],...
    'MarkerSize',8)
hold off

% use Plot Tools:
% Title:  Network Propagation Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Network Propagation Event Size
% time series color:  black
% update display names and add a legend
% black line – Time Series
% red circles – Selected Time Steps

%% Propagation flow at individual time steps
% Figure names:  PF2, PF3
% Figure layout:  each is 1x2
% Figure title (doc): Network Propagation Flow (small propagation events)
% tsn:  2, 28, 32, 237

% Figure name:  PF4
% Figure layout:  1x2
% Figure title (doc): Network Propagation Flow (mid-size
% propagation events)
% tsn:  64, 204

% tsn = time step number
tsn = 2;
```

```
% establish axes
axis square
axis ([1 21 1 21])
axis off
hold on

% plot node grid
xx= NCoord(:,1);
yy= NCoord(:,2);
plot(xx,yy,'o')
hold on

% plot propagation flow
gplot(AAO_t_TS(:,:,tsn),NCoord,'-ok')
hold on

% use Plot Tools:
% node grid: marker size 8 and edge color black
% propagation event nodes: marker size 12 and face color dark blue:
% RGB 11, 132, 199

%% continue
% plot and color propagating nodes (red)
% PNCoord = propagating node coordinates
PNCoord = NCoord;
PNCoord(find(PNL_t_TS(:,:,tsn) == 0),:) = [ ];
xx= PNCoord(:,1);
yy= PNCoord(:,2);
plot(xx,yy,'o','MarkerFaceColor',[.824 .004 .216],'MarkerSize',12)
hold on

% plot and color input node (green)
% INCoord = input node coordinates
INCoord = NCoord;
INCoord(find(INSL_t_TS(:,:,tsn) == 0),:) = [ ];
xx= INCoord(:,1);
yy= INCoord(:,2);
plot(xx,yy,'o','MarkerFaceColor',[0 1 .392],'MarkerSize',12)
hold off

% use Plot Tools:
% Title:  Network Propagation Flow
% Time step tsn, Event size = NetPESize_TSwnp(tsn)
% create and select second axes (if applicable)

%% Propagation flow plus adjacency matrix at individual time step
% Figure names:  PF5, PF6
% Figure layout:  each is 1x2
% Figure title (doc): Network Propagation Flow (large propagation event)
% tsn:  152, 514

% time step number
tsn = 152;
```

```
% establish axes
axis square
axis ([1 21 1 21])
axis off
hold on

% plot node grid
xx= NCoord(:,1);
yy= NCoord(:,2);
plot(xx,yy,'o')
hold on

% plot propagation flow
gplot(AAO_t_TS(:,:,tsn),NCoord,'-ok')
hold on

% use Plot Tools:
% node grid: marker size 8 and edge color black
% propagation event nodes: marker size 10 and face color dark blue:
% RGB 11, 132, 199

%% continue
% plot and color propagating nodes (red)
PNCoord = NCoord;
PNCoord(find(PNL_t_TS(:,:,tsn) == 0),:) = [ ];
xx= PNCoord(:,1);
yy= PNCoord(:,2);
plot(xx,yy,'o','MarkerFaceColor',[.824 .004 .216],'MarkerSize',10)
hold on

% plot and color input node (green)
INCoord = NCoord;
INCoord(find(INSL_t_TS(:,:,tsn) == 0),:) = [ ];
xx= INCoord(:,1);
yy= INCoord(:,2);
plot(xx,yy,'o','MarkerFaceColor',[0 1 .392],'MarkerSize',10)
hold off

% use Plot Tools:
% Title:  Network Propagation Flow
% Time step tsn, Event size = NetPESize_TSwnp(tsn)
% create and select second axes

%% continue
% establish axes
axis square
axis ([0 101 0 101])
hold on

% plot depiction of adjacency matrix
spy(AAO_t_TS(:,:,tsn))
hold off

% use Plot Tools:
% Title:  Depiction of Operational Adjacency Matrix
```

```
% Time step tsn
% remove nz text box
% x and y: change tick location and label from 0 to 1
% x and y limits: -1 to 102
% resize smaller than node grid

%% Cumulative propagation flow plus adjacency matrix at a time step
% Figure names:  PF7, PF8
% Figure layout:  each is 1x2
% Figure title (doc): Cumulative Propagation Flow
% tsn:  100, 1000

% time step number
tsn = 100;

% establish axes
axis square
axis ([1 21 1 21])
axis off
hold on

% plot node grid
xx= NCoord(:,1);
yy= NCoord(:,2);
plot(xx,yy,'o')
hold on

% plot cumulative propagation flow
gplot(AAO_tc_TS(:,:,tsn),NCoord,'-ok')
hold off

% use Plot Tools:
% node grid: marker size 8 and edge color black
% propagation event nodes: marker size 10 and face color light blue:
% RGB 222, 235, 250

% use Plot Tools:
% Title:  Cumulative Propagation Flow
% Time step tsn
% create and select second axes

%% continue
% establish axes
axis square
axis ([0 101 0 101])
hold on

% plot depiction of adjacency matrix
spy(AAO_tc_TS(:,:,tsn))
hold off

% use Plot Tools:
% Title:  Depiction of Operational Adjacency Matrix
% Time step tsn
% remove nz text box
```

```
% x and y: change tick location and label from 0 to 1
% x and y limits: -1 to 102
% resize smaller than node grid

%% Cumulative flow value "adjacency matrix" - with the specific flow values
% Figure name:  PF9
% Figure title (doc): Cumulative Flow Value "Adjacency Matrix"
% tsn:  100

% time step number
tsn = 100;

% establish axes
axis square
axis ([0 101 0 101])
hold on

% ANFVsubset = node flow value subset “adjacency matrix” (100 x 100)
% (temporary variable)

% calculate "adjacency matrix" for each value of flow
% plot each "adjacency matrix" on the same axes
for k = 1: max(max(ANFV_tc_TS(:,:,tsn)))
    ANFVsubset = ANFV_tc_TS(:,:,tsn);
    ANFVsubset(ANFVsubset ~= k) = 0;
    spy(ANFVsubset)
    hold on
end
hold off

% use Plot Tools:
% Title:  Cumulative Flow Value
% Time step tsn
% remove nz text box
% x and y: change tick location and label from 0 to 1
% x and y limits: -1 to 102
% color each plot a different color
% use blue, green, red, purple, orange
% marker size 14
% add legend: flow value = 1; flow value = 2; flow value = 3;
% flow value = 4; flow value = 5

%% Cumulative flow value "adjacency matrix" - with flow value intervals
% Figure name:  PF10
% Figure title (doc): Cumulative Flow Value "Adjacency Matrix"
% tsn:  1000

% time step number
tsn = 1000;

% establish axes
axis square
axis ([0 101 0 101])
hold on
```

```
% fvsi = flow value size interval (scalar)
% nfvintervals = number of flow value intervals (scalar)
% (temporary variable)
% ANFVsubset = node flow value subset “adjacency matrix” (100 x 100)
% (temporary variable)

% set variables
fvsi = 10;
nfvintervals = ceil(max(max(ANFV_tc_TS(:,:,tsn))) / fvsi);    % rounded up

% calculate "adjacency matrix" for each interval of flow
% plot each "adjacency matrix" on the same axes
for k = 1: nfvintervals
ANFVsubset = ANFV_tc_TS(:,:,tsn);
ANFVsubset(ANFVsubset < (((k-1) * fvsi) + 1)) = 0;
ANFVsubset(ANFVsubset > k * fvsi) = 0;
spy(ANFVsubset)
hold on
end
hold off

% use Plot Tools:
% Title:  Cumulative Flow Value
% Time step tsn
% remove nz text box
% x and y: change tick location and label from 0 to 1
% x and y limits: -1 to 102
% make each plot a different shape/size/color
% use dot/14/blue, dot/14/green, dot/14/red, circle/5/black,
% square/5/black, triangle/6/black
% add legend: flow value 1 to 10; flow value 11 to 20; flow value 21 to 30;
% flow value 31 to 40; flow value 41 to 50; flow value 51 to 60

%% Input/Output/Stock History Graphics

%% Cumulative node input value at a time step
% Figure name:  PF11
% Figure layout:  2x2
% Figure title (doc): Cumulative Node Input Value Grid
% tsn:  100, 400, 700, 1000

% time step number
tsn = 100;

% plot grid for 4 time steps on 4 axes
bar3(INVG_tc_TS(:,:,tsn),'cyan')

% use Plot Tools on each plot:
% Title:  Cumulative Node Input Value Grid
% Time step tsn
% face color: light green
% x and y limits: 0 to 11
% z limits: modify if necessary
% snap each plot to grid
% move each plot 5 blocks toward center
```

```
%% Cumulative node output value at a time step
% Figure name:  PF12
% Figure layout:  2x2
% Figure title (doc): Cumulative Node Output Value Grid
% tsn:  100, 400, 700, 1000

% time step number
tsn = 100;

% plot grid for 4 time steps on 4 axes
bar3(ONVG_tc_TS(:,:,tsn),'cyan')

% use Plot Tools on each plot:
% Title:  Cumulative Node Output Value Grid
% Time step tsn
% face color: light blue
% x and y limits: 0 to 11
% z limits: modify if necessary
% snap each plot to grid
% move each plot 5 blocks toward center

%% Cumulative node stock value at a time step
% Figure name:  PF13
% Figure layout:  2x2
% Figure title (doc): Cumulative Node Stock Value Grid
% tsn:  100, 400, 700, 1000

% time step number
tsn = 100;

% plot grid for 4 time steps on 4 axes
bar3(NSVG_tc_TS(:,:,tsn),'cyan')

% use Plot Tools on each plot:
% Title:  Cumulative Node Stock Value Grid
% Time step tsn
% face color: light pink
% x and y limits: 0 to 11
% z limits: modify if necessary
% snap each plot to grid
% move each plot 5 blocks toward center

%% Network Propagation Event Graphics

%% Time series and distribution
% Figure name:  NetPE1
% Figure layout:  2x1
% Figure title (doc): Network Propagation Event Time Series and
% Distribution

% plot time series
plot(NetPESize_TSwnp)
```

```
% use Plot Tools:
% Title:  Network Propagation Event Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Network Propagation Event Size
% color:  black
% create and select second axes

%% continue
% plot distribution
xx = NetPEDistr(:,1);
yy = NetPEDistr(:,2);
plot(xx,yy)

% use Plot Tools:
% Title:  Network Propagation Event Distribution
% x:  Network Propagation Event Size
% y:  Number of Network Propagation Events
% y lower limit: -10
% color:  dark green
% line width:  2

%% Distribution set
% Figure name:  NetPE2
% Figure layout:  set of three – eyes and nose
% Figure title (doc): Network Propagation Event Distribution Set

% plot distribution
xx = NetPEDistr(:,1);
yy = NetPEDistr(:,2);
plot(xx,yy)

% use Plot Tools:
% Title:  Network Propagation Event Distribution
% x:  Network Propagation Event Size
% y:  Number of Network Propagation Events
% y lower limit: -10
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot log-log distribution
xx = log10(NetPEDistr(:,1));
yy = log10(NetPEDistr(:,2));
plot(xx,yy,'o')

% use Plot Tools:
% Title:  Log-Log Propagation Event Distribution
% x:  log(Network Propagation Event Size)
% y:  log(Number of Network Propagation Events)
% marker size:  3
% basic fitting:  linear with equation
% color:  default blue
% line width:  2
```

```
% create and select next axes

%% continue
% plot distribution with power-law overlay
% do this entirely with Plot Tools:

% copy and paste distribution from axes 1 to these axes
% y lower limit: -10

% prepare power-law overlay
% obtain y-intercept (for C) and slope (for alpha) from linear equation on
% the log-log axes
% logC = y-intercept
% C = 10.^(y-intercept)
% alpha = slope
% y = C*x.^alpha
% y = (10.^(y-intercept))* x.^(slope)

% add power-law overlay to axes (use Add Data on Plot Tools)
% xsource:  NetPEDistr(:,1)
% ysource:  (10.^(y-intercept))* NetPEDistr(:,1).^(slope)
% color:  red  (RGB = 210  1  55)
% line width:  2

% update display names and add a legend
% blue line – Network Propagation Event Distribution
% red line – Power-Law Overlay

% add figure title and axis labels
% Title:  Event Distribution with Power-Law Overlay
% x:  Network Propagation Event Size
% y:  Number of Network Propagation Events

%% Frequency spectrum
% Figure name:  NetPE3
% Figure layout:  set of three – vertical 3x1
% Figure title (doc): Network Propagation Event Frequency Spectrum

% plot entire single-sided spectrum
xx = normfreq;
yy = PEssAmp;
plot(xx,yy)

% use Plot Tools:
% Title:  Single-Sided Network Propagation Event Frequency Spectrum
% x:  Normalized Frequency (Hz)
% y:  Amplitude
% x lower limit = -2.5
% y lower limit = -2.5
% color:  default blue
% line width:  1
% create and select next axes

%% continue
% plot partial single-sided spectrum
```

```
xx = freq;
yy = amp;
plot(xx,yy)

% use Plot Tools:
% Title:  Network Propagation Event Frequency Spectrum (Partial)
% x:  Normalized Frequency (Hz)
% y:  Amplitude
% x lower limit = -.5
% y lower limit = -2.5
% color:  default blue
% line width:  2
% add markers to show points (+, 3)
% create and select next axes

%% continue
% plot partial spectrum with power-law overlay
% do this entirely with Plot Tools:

% copy and paste partial spectrum from axes 2 to these axes
% x lower limit = -.5
% y lower limit = -2.5

% add power-law overlay to axes (use Add Data on Plot Tools)
% xsource:  100*linspace(0, 0.2, numpoints)
% ysource:  50*[1:numpoints].^(-1.2)
% This overlay is an arbitrary power-law overlay with parameters chosen to
% suggest that the frequency spectrum is a “1/f noise” spectrum with
% fractal behavior in time
% color:  red  (RGB = 210  1  55)
% line width:  2

% update curve display names and add a legend
% blue line – Frequency Spectrum
% red line – Power-Law Overlay

% add figure title and axis labels
% Title:  Frequency Spectrum with Power-Law Overlay
% x:  Normalized Frequency (Hz)
% y:  Amplitude

%% Path Length Graphics

%% Time series and distribution
% Figure name:  PL1
% Figure layout:  2x1
% Figure title (doc): Path Length Time Series and Integrated Distribution

% plot time series
plot(PLmax_TSwnp)

% use Plot Tools:
% Title:  Path Length Time Series
% x:  Time
% x limits:  0 to 1001
```

```
% y:  Maximum Path Length
% color:  black
% create and select second axes

%% continue
% plot distribution
xx = MaxPLDistr(:,1);
yy = MaxPLDistr(:,2);
plot(xx,yy)

% use Plot Tools:
% Title:  Integrated Path Length Distribution
% x:  Maximum Path Length
% y:  Number of Events
% y lower limit: -10
% color:  dark green
% line width:  2

%% Distribution set
% Figure name:  PL2
% Figure layout:  set of three – eyes and nose
% Figure title (doc): Integrated Path Length Distribution Set

% plot distribution
xx = MaxPLDistr(:,1);
yy = MaxPLDistr(:,2);
plot(xx,yy)

% use Plot Tools:
% Title:  Integrated Path Length Distribution
% x:  Path Length
% y:  Number of Events
% y lower limit: -10
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot log-log distribution
xx = log10(MaxPLDistr(:,1));
yy = log10(MaxPLDistr(:,2));
plot(xx,yy,'o')

% use Plot Tools:
% Title:  Log-Log Integrated Path Length Distribution
% x:  log(Path Length)
% y:  log(Number of Events)
% marker size:  3
% basic fitting:  linear with equation
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot distribution with power-law overlay
```

```
% do this entirely with Plot Tools:

% copy and paste distribution from axes 1 to these axes
% y lower limit: -10

% prepare power-law overlay
% obtain y-intercept (for C) and slope (for alpha) from linear equation on
% the log-log axes
% logC = y-intercept
% C = 10.^(y-intercept)
% alpha = slope
% y = C*x.^alpha
% y = (10.^(y-intercept))* x.^(slope)

% add power-law overlay to axes (use Add Data on Plot Tools)
% xsource:  MaxPLDistr(:,1)
% ysource:  (10.^( y-intercept))* MaxPLDistr(:,1).^( slope)
% color:  red  (RGB = 210  1  55)
% line width:  2

% update display names and add a legend
% blue line – Integrated Path Length Distribution
% red line – Power-Law Overlay

% add figure title and axis labels
% Title:  Integrated Path Length Distribution with Power-Law Overlay
% x:  Path Length
% y:  Number of Events

%% Mean path length time series and distribution set
% Figure name:  PL3
% Figure layout:  set of four – eyebrow, eyes, and nose
% Figure title (doc): Mean Path Length Time Series and Distribution Set

% plot time series
plot(PLmean_TSwnp)

% use Plot Tools:
% Title:  Mean Path Length Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Mean Path Length
% color:  black
% create and select next axes

%% continue
% plot distribution
xx = MeanPLDistr(:,1);
yy = MeanPLDistr(:,2);
plot(xx,yy)

% use Plot Tools:
% Title:  Integrated Path Length Distribution
% x:  Mean Path Length
% y:  Number of Events
```

```
% y limits: -10 to 125
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot log-log distribution
xx = log10(MeanPLDistr(:,1));
yy = log10(MeanPLDistr(:,2));
plot(xx,yy,'o')

% use Plot Tools:
% Title:  Log-Log Integrated Path Length Distribution
% x:  log(Mean Path Length)
% y:  log(Number of Events)
% x limits: -0.1 to 1
% y limits: -1 to 2.4
% marker size:  3
% basic fitting:  linear with equation
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot distribution with power-law overlay
% do this entirely with Plot Tools:

% copy and paste distribution from axes 2 to these axes
% y limits: -10 to 125

% prepare power-law overlay
% obtain y-intercept (for C) and slope (for alpha) from linear equation on
% the log-log axes
% y = C*x.^alpha
% y = (10.^(y-intercept))* x.^(slope)

% add power-law overlay to axes (use Add Data on Plot Tools)
% xsource:  MeanPLDistr(:,1)
% ysource:  (10.^( 1.9))* MeanPLDistr(:,1).^( -2.1)
% color:  red  (RGB = 210  1  55)
% line width:  2

% update display names and add a legend
% blue line – Integrated Path Length Distribution
% red line – Power-Law Overlay

% add figure title and axis labels
% Title:  Integrated Path Length Distribution with Power-Law Overlay
% x:  Mean Path Length
% y:  Number of Events

%% Comparison of individual time step and cumulative distributions
% Figure name:  PL4
% Figure layout:  set of four – 2x2
% Figure title (doc): Path Length Distribution Comparison
```

```
% time step number
tsn = 152;

% plot individual time step distribution
xx = PLDistr_t_TS(:,1,tsn);
yy = PLDistr_t_TS(:,2,tsn);
plot(xx,yy)

% use Plot Tools:
% Title:  Individual Time Step Path Length Distribution
% Time step tsn
% x:  Path Length
% y:  Number of Paths
% x limits:  0 to 25
% y limits:  -100 to 1000
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot individual time step log-log distribution
xx = log10(PLDistr_t_TS(:,1,tsn));
yy = log10(PLDistr_t_TS(:,2,tsn));
plot(xx,yy)

% use Plot Tools:
% Title:  Log-Log Individual Time Step Distribution
% Time step tsn
% x:  log(Path Length)
% y:  log(Number of Paths)
% x limits:  0 to 1.5
% y limits:  0 to 4
% Auto off
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot cumulative distribution
xx = PLDistr_tc_TS(:,1,tsn);
yy = PLDistr_tc_TS(:,2,tsn);
plot(xx,yy)

% use Plot Tools:
% Title:  Cumulative Path Length Distribution
% Time step tsn
% x:  Path Length
% y:  Number of Paths
% x limits:  0 to 25
% y limits:  -100 to 1000
% color:  default blue
% line width:  2
% create and select next axes
```

```
%% continue
% plot cumulative log-log distribution
xx = log10(PLDistr_tc_TS(:,1,tsn));
yy = log10(PLDistr_tc_TS(:,2,tsn));
plot(xx,yy)

% use Plot Tools:
% Title:  Log-Log Cumulative Distribution
% Time step tsn
% x:  log(Path Length)
% y:  log(Number of Paths)
% x limits:  0 to 1.5
% y limits:  0 to 4
% Auto off
% color:  default blue
% line width:  2

%% Indirect Effects Graphics

%% Indicator individual time step time series
% Figure name:  IE1
% Figure layout:  3x1
% Figure title (doc): Indicator Individual-Time-Step Time Series

% plot direct effects ratio time series
stem(DER_t_TS)

% use Plot Tools:
% Title:  Direct Effects Ratio Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Direct Effects Ratio
% color:  default blue
% marker:  none
% create and select next axes

%% continue
% plot indirect effects ratio time series
stem(IER_t_TS)

% use Plot Tools:
% Title:  Indirect Effects Ratio Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Indirect Effects Ratio
% color:  default blue
% marker:  none
% create and select next axes

%% continue
% plot indirect effects index time series
stem(IEI_t_TS)

% use Plot Tools:
% Title:  Indirect Effects Index Time Series
```

```
% x:  Time
% x limits:  0 to 1001
% y:  Indirect Effects Index
% y limits:  0 to 15
% color:  default blue
% marker:  none

%% Indicator cumulative time series
% Figure name:  IE2
% Figure layout:  3x1
% Figure title (doc): Indicator Cumulative Time Series

% plot cumulative direct effects ratio time series
plot(DER_tc_TS)

% use Plot Tools:
% Title:  Cumulative Direct Effects Ratio Time Series
% x:  Time
% x limits:  -10 to 1005
% y:  Cumulative Direct Effects Ratio
% line width:  2
% color:  default blue
% create and select next axes

%% continue
% plot cumulative indirect effects ratio time series
plot(IER_tc_TS)

% use Plot Tools:
% Title:  Cumulative Indirect Effects Ratio Time Series
% x:  Time
% x limits:  -10 to 1005
% y:  Cumulative Indirect Effects Ratio
% line width:  2
% color:  default blue
% create and select next axes

%% continue
% plot cumulative indirect effects index time series
plot(IEI_tc_TS)

% use Plot Tools:
% Title:  Cumulative Indirect Effects Index Time Series
% x:  Time
% x limits:  -10 to 1005
% y:  Cumulative Indirect Effects Index
% line width:  2
% color:  default blue

%% Indirect path quantity time series and distribution set
% Figure name:  IE3
% Figure layout:  set of three – eyebrow and eyes
% Figure title (doc): Indirect Path Quantity Time Series and Distribution

% plot time series
```

```
plot(IndPathQ_TSwze)

% use Plot Tools:
% Title:  Indirect Path Quantity Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Quantity of Indirect Paths
% color:  default blue
% create and select next axes

%% continue
% plot distribution
xx = IndPathQDistr(:,1);
yy = IndPathQDistr(:,2);
plot(xx,yy)

% use Plot Tools:
% Title:  Indirect Path Quantity Distribution
% x:  Quantity of Indirect Paths
% y:  Number of Propagation Events
% y limits:  -10 to 120
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot distribution with power-law overlay
% do this entirely with Plot Tools:

% copy and paste distribution from axes 2 to these axes
% y limits:  -10 to 120

% prepare power-law overlay
% y = C*x.^alpha
% y = (1000000)*x.^(-2)
% parameters chosen for "best fit"

% add power-law overlay to axes (use Add Data on Plot Tools)
% xsource:  IndPathQDistr(:,1)
% ysource:  (1000000)* IndPathQDistr(:,1).^(-2)
% color:  red  (RGB = 210  1  55)
% line width:  2

% update display names and add a legend
% blue line – Indirect Path Quantity Distribution
% red line – Power-Law Overlay

% add figure title and axis labels
% Title:  Distribution with Power-Law Overlay
% x:  Quantity of Indirect Paths
% y:  Number of Propagation Events

%% Direct path quantity time series and distribution set
% Figure name:  IE4
% Figure layout:  set of three – eyebrow and eyes
```

```
% Figure title (doc): Direct Path Quantity Time Series and Distribution

% plot time series
plot(DirPathQ_TSwze)

% use Plot Tools:
% Title:  Direct Path Quantity Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Quantity of Direct Paths
% color:  default blue
% create and select next axes

%% continue
% plot distribution
xx = DirPathQDistr(:,1);
yy = DirPathQDistr(:,2);
plot(xx,yy)

% use Plot Tools:
% Title:  Direct Path Quantity Distribution
% x:  Quantity of Direct Paths
% y:  Number of Propagation Events
% y limits:  -10 to 200
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot distribution with power-law overlay
% do this entirely with Plot Tools:

% copy and paste distribution from axes 2 to these axes
% y limits:  -10 to 200

% prepare power-law overlay
% y = C*x.^alpha
% y = (15000)*x.^(-2)
% parameters chosen for "best fit"

% add power-law overlay to axes (use Add Data on Plot Tools)
% xsource:  DirPathQDistr(:,1)
% ysource:  (15000)* DirPathQDistr(:,1).^(-2)
% color:  red  (RGB = 210  1  55)
% line width:  2

% update display names and add a legend
% blue line – Direct Path Quantity Distribution
% red line – Power-Law Overlay

% add figure title and axis labels
% Title:  Distribution with Power-Law Overlay
% x:  Quantity of Direct Paths
% y:  Number of Propagation Events
```

```
%% Cumulative indirect path quantity and direct path quantity time series
% Figure name:  IE5
% Figure layout:  2x1
% Figure title (doc): Cumulative Path Quantity Time Series

% plot cumulative indirect path quantity time series
plot(CuIndPathQ_TS)

% use Plot Tools:
% Title:  Cumulative Indirect Path Quantity Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Cumulative Quantity of Indirect Paths
% line width:  2
% color:  default blue
% create and select next axes

%% continue
% plot cumulative direct path quantity time series
plot(CuDirPathQ_TS)

% use Plot Tools:
% Title:  Cumulative Direct Path Quantity Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Cumulative Quantity of Direct Paths
% line width:  2
% color:  default blue

%% Network Connectivity Graphics

%% Time series with selected time steps
% Figure name:  NC1
% Figure title in doc:  Network Propagation Time Series with Selections

% plot time series
plot(NetPESize_TSwnp)
hold on

% define selection circles
xx = [32 64 152];
yy = NetPESize_TSwnp(xx);

% plot selection circles
plot(xx,yy,'o','LineWidth',2, 'MarkerEdgeColor',[.824 .004 .216],...
    'MarkerSize',10)
hold off

% use Plot Tools:
% Title:  Network Propagation Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Network Propagation Event Size
% time series color:  black
% update display names and add a legend
```

```
% black line – Time Series
% red circles – Selected Time Steps

%% Individual-time-step node “combined” degree grid
% Figure name:  NC2
% Figure layout:  3x1
% Figure title (doc):  Node “Combined” Degree Grid at Representative Time
% Steps
% tsn:  32, 64, 152

% tsn = time step number
tsn = 32;

% define a matrix NodeCDegGridz (10x10) (temporary variable) for plotting
% node-combined-degree-grid zero values
NodeCDegGridz = NodeCDegGrid_t_TS(:,:,tsn);
NodeCDegGridz(NodeCDegGridz >= 1) = NaN;

% plot grid zero values
stem3(NodeCDegGridz)
hold on

% define a matrix NodeCDegGridnz (10x10) (temporary variable) for plotting
% node-combined-degree-grid nonzero values
NodeCDegGridnz = NodeCDegGrid_t_TS(:,:,tsn);
NodeCDegGridnz(NodeCDegGridnz == 0) = NaN;

% plot grid nonzero values
stem3(NodeCDegGridnz, 'fill')
hold off

% use Plot Tools for overall plot:
% Title:  Node Degree Grid for Small Propagation Event
% Time step tsn, Event size = NetPESize_TSwnp(tsn)
% z:  Combined-Degree
% z limits: 0 to 15
% x,y limits: 0 to 11; y reverse; ticks from 1 to 10 in increments of 1
% x,y grid lines off

% use Plot Tools for zero value plot:
% line width:  0.5
% marker:  o and 5
% color:  black

% use Plot Tools for nonzero value plot:
% line width:  2
% marker:  o and 5
% color:  default blue

% create and select second axes
% update tsn to second value
% rerun this cell
% repeat Plot Tools operations, but change plot title
% Title:  Node Degree Grid for Mid-Size Propagation Event
% Time step tsn, Event size = NetPESize_TSwnp(tsn)
```

```
% create and select third axes
% update tsn to third value
% rerun this cell
% repeat Plot Tools operations, but change plot title
% Title:  Node Degree Grid for Large Propagation Event
% Time step tsn, Event size = NetPESize_TSwnp(tsn)

%% Individual-time-step node “in” degree grid
% Figure name:  NC3
% Figure layout:  3x1
% Figure title (doc):  Node “In” Degree Grid at Representative Time Steps
% tsn:  32, 64, 152

% tsn = time step number
tsn = 32;

% define a matrix NodeIDegGridz (10x10) (temporary variable) for plotting
% node-in-degree-grid zero values
NodeIDegGridz = NodeIDegGrid_t_TS(:,:,tsn);
NodeIDegGridz(NodeIDegGridz >= 1) = NaN;

% plot grid zero values
stem3(NodeIDegGridz)
hold on

% define a matrix NodeIDegGridnz (10x10) (temporary variable) for plotting
% node-in-degree-grid nonzero values
NodeIDegGridnz = NodeIDegGrid_t_TS(:,:,tsn);
NodeIDegGridnz(NodeIDegGridnz == 0) = NaN;

% plot grid nonzero values
stem3(NodeIDegGridnz, 'fill')
hold off

% use Plot Tools for overall plot:
% Title:  Node Degree Grid for Small Propagation Event
% Time step tsn, Event size = NetPESize_TSwnp(tsn)
% z:  In-Degree
% z limits: 0 to 10
% x,y limits: 0 to 11; y reverse; ticks from 1 to 10 in increments of 1
% x,y grid lines off

% use Plot Tools for zero value plot:
% line width:  0.5
% marker:  o and 5
% color:  black

% use Plot Tools for nonzero value plot:
% line width:  2
% marker:  o and 5
% color:  default blue

% create and select second axes
% update tsn to second value
```

```
% rerun this cell
% repeat Plot Tools operations, but change plot title
% Title:  Node Degree Grid for Mid-Size Propagation Event
% Time step tsn, Event size = NetPESize_TSwnp(tsn)

% create and select third axes
% update tsn to third value
% rerun this cell
% repeat Plot Tools operations, but change plot title
% Title:  Node Degree Grid for Large Propagation Event
% Time step tsn, Event size = NetPESize_TSwnp(tsn)

%% Node combined-degree time series and distribution set
% Figure name:  NC4
% Figure layout:  set of four – eyebrow, eyes, and nose
% Figure title (doc): Node Combined-Degree Time Series and Distribution Set

% plot time series
plot(CDmax_t_TSwnp)

% use Plot Tools:
% Title:  Node Combined-Degree Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Node Combined-Degree
% color:  black
% create and select next axes

%% continue
% plot distribution
xx = MaxCDDistr(:,1);
yy = MaxCDDistr(:,2);
plot(xx,yy)

% use Plot Tools:
% Title:  Integrated Node Degree Distribution
% x:  Node Maximum Combined-Degree
% y:  Number of Events
% y lower limit: -10
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot log-log distribution
xx = log10(MaxCDDistr(:,1));
yy = log10(MaxCDDistr(:,2));
plot(xx,yy,'o')

% use Plot Tools:
% Title:  Log-Log Integrated Node Degree Distribution
% x:  log(Node Maximum Combined-Degree)
% y:  log(Number of Events)
% marker size:  3
% basic fitting:  linear with equation
```

```
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot distribution with power-law overlay
% do this entirely with Plot Tools:

% copy and paste distribution from axes 2 to these axes
% y lower limit: -10

% prepare power-law overlay
% obtain y-intercept (for C) and slope (for alpha) from linear equation on
% the log-log axes
% y = C*x.^alpha
% y = (10.^(y-intercept))* x.^(slope)

% add power-law overlay to axes (use Add Data on Plot Tools)
% xsource:  MaxCDDistr(:,1)
% ysource:  (10.^( y-intercept))* MaxCDDistr(:,1).^( slope)
% color:  red  (RGB = 210  1  55)
% line width:  2

% update display names and add a legend
% blue line – Integrated Node Degree Distribution
% red line – Power-Law Overlay

% add figure title and axis labels
% Title:  Integrated Node Degree Distribution with Power-Law Overlay
% x:  Node Maximum Combined-Degree
% y:  Number of Events

%% Node in-degree time series and distribution set
% Figure name:  NC5
% Figure layout:  set of four – eyebrow, eyes, and nose
% Figure title (doc): Node In-Degree Time Series and Distribution Set

% plot time series
plot(IDmax_t_TSwnp)

% use Plot Tools:
% Title:  Node In-Degree Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Node In-Degree
% color:  black
% create and select next axes

%% continue
% plot distribution
xx = MaxIDDistr(:,1);
yy = MaxIDDistr(:,2);
plot(xx,yy)

% use Plot Tools:
```

```
% Title:  Integrated Node Degree Distribution
% x:  Node Maximum In-Degree
% y:  Number of Events
% y lower limit: -10
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot log-log distribution
xx = log10(MaxIDDistr(:,1));
yy = log10(MaxIDDistr(:,2));
plot(xx,yy,'o')

% use Plot Tools:
% Title:  Log-Log Integrated Node Degree Distribution
% x:  log(Node Maximum In-Degree)
% y:  log(Number of Events)
% marker size:  3
% basic fitting:  linear with equation
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot distribution with power-law overlay
% do this entirely with Plot Tools:

% copy and paste distribution from axes 2 to these axes
% y lower limit: -10

% prepare power-law overlay
% obtain y-intercept (for C) and slope (for alpha) from linear equation on
% the log-log axes
% y = C*x.^alpha
% y = (10.^(y-intercept))* x.^(slope)

% add power-law overlay to axes (use Add Data on Plot Tools)
% xsource:  MaxIDDistr(:,1)
% ysource:  (10.^( y-intercept))* MaxIDDistr(:,1).^( slope)
% color:  red  (RGB = 210  1  55)
% line width:  2

% update display names and add a legend
% blue line – Integrated Node Degree Distribution
% red line – Power-Law Overlay

% add figure title and axis labels
% Title:  Integrated Node Degree Distribution with Power-Law Overlay
% x:  Node Maximum In-Degree
% y:  Number of Events

%% Node mean degree time series and distribution set
% Figure name:  NC6
% Figure layout:  set of four – eyebrow, eyes, and nose
```

```
% Figure title (doc): Node Mean Degree Time Series and Distribution Set

% plot time series
plot(NMD_t_TSwnp)

% use Plot Tools:
% Title:  Node Mean Degree Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Node Mean Degree
% color:  black
% create and select next axes

%% continue
% plot distribution
xx = NMDDistr(:,1);
yy = NMDDistr(:,2);
plot(xx,yy)

% use Plot Tools:
% Title:  Integrated Node Degree Distribution (Node Mean Degree)
% x:  Node Mean Degree
% y:  Number of Events
% y lower limit: -10
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot log-log distribution
xx = log10(NMDDistr(:,1));
yy = log10(NMDDistr(:,2));
plot(xx,yy,'o')

% use Plot Tools:
% Title:  Log-Log Integrated Node Degree Distribution (Node Mean Degree)
% x:  log(Node Mean Degree)
% y:  log(Number of Events)
% marker size:  3
% basic fitting:  linear with equation
% color:  default blue
% line width:  2
% create and select next axes

%% continue
% plot distribution with power-law overlay
% do this entirely with Plot Tools:

% copy and paste distribution from axes 2 to these axes
% y lower limit: -10

% prepare power-law overlay
% obtain y-intercept (for C) and slope (for alpha) from linear equation on
% the log-log axes
% y = C*x.^alpha
```

```
% y = (10.^(y-intercept))* x.^(slope)

% add power-law overlay to axes (use Add Data on Plot Tools)
% xsource:  NMDDistr(:,1)
% ysource:  (10.^( y-intercept))* NMDDistr(:,1).^( slope)
% color:  red  (RGB = 210  1  55)
% line width:  2

% update display names and add a legend
% blue line – Integrated Node Degree Distribution
% red line – Power-Law Overlay

% add figure title and axis labels
% Title:  Integrated Node Degree Distribution with Power-Law Overlay
% x:  Node Mean Degree
% y:  Number of Events

%% Individual time step time series set
% Figure name:  NC7
% Figure layout:  3x1
% Figure title (doc): Network Connectivity Time Series Set

% plot direct effects ratio time series
plot(NMD_t_TSwnp)

% use Plot Tools:
% Title:  Node Mean Degree Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Node Mean Degree
% color:  darker blue
% create and select next axes

%% continue
% plot indirect effects ratio time series
plot(NetDen_t_TS)

% use Plot Tools:
% Title:  Network Connection Density Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Network Connection Density
% color:  darker blue
% create and select next axes

%% continue
% plot indirect effects index time series
plot(NumLN_t_TS)

% use Plot Tools:
% Title:  Number of Linked Nodes Time Series
% x:  Time
% x limits:  0 to 1001
% y:  Number of Linked Nodes
% color:  darker blue
```