Master's thesis

2019

Pauline Haavind Jensen

Master's thesis

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Pauline Haavind Jensen

Cascading Failures in Dynamic Networks

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Cascading Failures in Dynamic Networks

Pauline Haavind Jensen

Engineering and ICT
Submission date: June 2019
Supervisor: Dr. Astrid S. de Wijn

Norwegian University of Science and Technology
Department of Mechanical and Industrial Engineering

Summary

This master thesis looks at how failure spreads across a network. A simple model for failure and repair dynamics is implemented to simulate cascading failure. In it network components are disturbed and can fail if they have failing neighbours. Failed nodes are repaired with probability depending on the total repair capacity for the system.

The differential equation for the model has been analyzed using mean-field approximation and graphical analysis by Eran Reches (Reches, 2019). The analysis suggests some critical parameter values that determine whether the system crashes or not. Simulations are run to investigate if these parameter values have the same end-state in the stochastic simulations.

Results from the stochastic simulations show that the analysis can predict the *presence* of "good" and "bad" steady states in the system. Dividing time steps into smaller dt also makes the simulations approach the predicted *value* for both these steady states and threshold that divides them.

The results of simulations on different types of networks show that scale-free networks are more vulnerable than random networks, and that regular grids are more robust to cascading failure. These results are in keeping with previous studies.

Sammendrag

Denne masteroppgaven ser på hvordan feil sprer seg gjennom et nettverk. En enkel modell for feil- og reparasjonsdynamikk blir implementert for å simulere feilspredning. I modellen blir nettverkskomponenter forstyrret og kan ødelegges av å ha ødelagte naboer. Sannsynligheten for at en ødelagt komponent blir reparert er avhengig av systemets totale reparasjonskapasitet.

Differensialligningen for modellen har blitt analysert ved bruk av “mean-field approximation” og grafisk analyse av Eran Reches (Reches, 2019). Resultatene fra analysen indikerer noen kritiske parameterverdier som kan avgjøre om systemet vil bryte sammen eller ikke. Simuleringer blir kjørt for å undersøke om disse parameterverdiene gir samme *utfall* i de stokastiske simuleringene.

Resultater fra de stokastiske simuleringene viser at analysen kan forutse forekomst av “gode” og “dårlige” *utfall* i systemet. Ved å dele tidssteg i mindre deler dt går også simuleringsresultatene mot de *verdiene* for disse *utfallene* som ble forutsett av analysen.

Resultatene fra simuleringene på ulike typer nettverk viser at “scale-free networks” er mer sårbare enn “random networks”, og at “grids” er mer robust mot kaskadefeil. Dette stemmer overens med tidligere studier.

Preface and Acknowledgements

This documents my master thesis in Industrial ICT, Engineering Design and Materials at the Department of Mechanical and Industrial Engineering, NTNU. The work was carried out during the spring of 2019 and supervised by Dr. Astrid S. de Wijn.

This thesis has come to be with great help from many people. I want to thank Astrids whole group of master students and PhD candidates for giving me invaluable moral support, fun distractions and cake every Friday!

Thank you to everyone at “Masterkontor 212” for making me want to come into the office, and to Marthe, Katja and Hilde for making me want to go home again!

I owe a *huge* thank you to Eran Reches for doing the analysis on the system, and for being a great tutor and friend this spring. I hope both you and your Lego man are doing great! And thank you to Dr. Baruch Barzel at Bar Ilan University in Israel for your great advice, your interest in the project, and for sending Eran our way.

Lastly, the biggest thank you to my supervisor Astrid who truly is an amazing scientist, teacher and person. Any student that gets you as their supervisor is incredibly lucky!

Pauline Haavind Jensen
Trondheim, June 2019

Table of Contents

1 | Introduction

In this chapter an introduction to the topic and scope of the thesis is given.

1.1 Background and Motivation

Network science is the study of complex network systems such as railroad networks, power-grids, social networks and neural networks. Anything that can be represented as a number of components or **nodes** with some connection, or **link**, between them can be considered a network. An important area of research within the field has been aiming to understand how something, be it a meme, a disease, or a power outage, spreads across a network. Typically using some dynamic model of how the failure of a node or an edge will affect the rest of the system, with specific interest in identifying what differentiates the cases where the failure is contained from the cases of **cascading failure**.

Useful information about networks can be gained from investigating dynamic network failure models. An example is the simulation of the spread of the 2009 swine flu pandemic. Using Monte Carlo simulations and a network of human mobility the scientists were able to predict how the disease would spread (Balcan et al., 2009). Such knowledge of if, when and where an infection will peak can be used to prevent diseases from spreading, for example by identifying to whom vaccines should be distributed. Studies on the subject have found that it is more effective to isolate large cities than to shut down strong travel edges (Hufnagel et al., 2004). Similarly, simulations of power outages can be used to identify weak spots in a power grid. After a major power grid failure in North America in 1996, a new model for simulating such failures was developed, and led to better understanding of how to reinforce the network (Kosterev et al., 1999).

This thesis presents a dynamic network failure model where components can fail if they have a failed neighbour. Their chance of failure depends on the ratio of failed neighbours to the total number of neighbours, and the **sensitivity** of the system. A **repair rate** is introduced and aims to mimic the presence of a fixed number of repair crews. All the crews are available to fix broken nodes, but if the number of broken nodes exceeds the number of repair crews, the repair crews will be randomly distributed among the broken nodes. The dynamics of the differential equation for the model has been analyzed by

Eran Reches. Using mean-field approximations and graphical analysis of the differential equation he identified some critical value-combinations for the repair capacity and the sensitivity of nodes (Reches, 2019). These parameter values can determine if the system is able to stabilize on a functional level, or if a cascade happens.

1.2 Objectives

The aim of this thesis is to present and study a dynamic network failure model with a repair rate. The model will be implemented and results from stochastic simulations will be presented, discussed and compared with theoretical analyses of the dynamics.

Some questions that will be attempted answered are

- **Do the theoretical analyses give a good approximation of the stochastic simulations?**.
- **How can any potential differences between the stochastic simulations and theoretical analysis be explained?**
- What are the features of our model?

1.3 Approach

A simulation of a dynamic model for failure cascades in networks is written in Python utilizing the NetworkX package (Hagberg et al., 2008). The specific dynamics will be presented in detail in chapter 3. Stochastic simulations will be run with different initial conditions on different networks, and the results will be considered and compared with the theoretical analysis.

The main result from the simulations will be the average number of failed nodes as a function of time.

1.4 Structure of the Report

Chapter 1: Introduction gives an introduction to the topic and scope of the thesis.

Chapter 2: Theory gives a more thorough explanation of the subject of networks, cascading failure and graphical analysis of differential equations.

Chapter 3: Methods gives a description of our model along with a summary of the analysis performed by Eran Reches (Reches, 2019). The implementation and how simulations were run is described.

In **Chapter 4: Results** the results from the simulations is presented, and attempts are made to locate fixed points for the simulations.

In **Chapter 5: Discussion** the model and the results from the simulations are discussed and the questions stated in the introduction are attempted answered.

Chapter 6: Conclusion gives a short summary of the findings in this thesis.

2 Theory

In this chapter an introduction to network theory and graphical dynamics analysis is given. Some definitions and terms that will be assumed familiar in the rest of the thesis is presented and explained. There will also be a presentation of some previous works on the subject of cascading failures in networks.

2.1 Network Properties

In this section some basic terms used to describe networks are explained. The web-book *Network Science* by Albert-László Barabási (Barabási, 2016) is the theoretical basis for this section, and a more in-depth explanation of the topic can be found there. This section is also based on the theory chapter of my project thesis (Jensen, 2018).

A network is a collection of **components** or **nodes** that are connected with **edges** or **links**. A **complex network** (Figure 2.1c) is one that has an intricate topology, as opposed to for example a complete graph (Figure 2.1b), where every node is connected to every node, or a mesh-graph or **grid** (Figure 2.1a), where the nodes and edges form a regular lattice pattern. Networks found in the real world are typically complex (Barabási and Albert, 1999; Strogatz, 2001), and some examples are neurons and synapses in a brain, power grids, the internet, transportation networks or the World-Wide-Web.

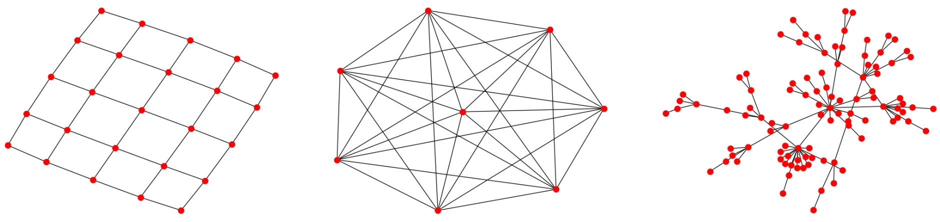

(a) Grid. **(b)** Complete network. **(c)** Complex network.

Figure 2.1: Different types of networks.

The **degree** of a node in a network tells you how many neighbours it has—how many nodes it has a direct link to. In a friendship-network the degree of a person-component will be the number of friends it has. An often used descriptor of a network is its **average degree**, $\langle k \rangle$. In the literature, different network topologies are typically described by their **degree distribution**, meaning the probability distribution of node degrees over the whole network. Figures 2.2, 2.3 and 2.4 show different networks along with their degree distributions.

Two nodes are connected if there is a **path** between them, and a network is connected if there is a path from every node to every node. Notice that a path does not have to be direct, like an edge or link, but can go via other nodes. The network in Figure 2.3 is disconnected, where 4 nodes have a degree of zero, these have no connection to the **giant component** that makes up the rest of the network.

The **diameter** of a network is the longest *shortest path* between any two nodes. In a complete network, like the one in Figure 2.1b, the diameter is one since there is a direct link from every node to every node. The network in Figure 2.3 has an infinite diameter since it is disconnected, but the diameter of the giant component is 6.

One characteristic of most real-world networks is that they are **sparse**, meaning that if two random nodes are selected, the probability of an edge between them is low. They also tend to have relatively small diameters compared to the number of nodes and edges. This is known as the **small-world** phenomenon, referring to the *Small World* experiment by Stanley Milgram (Milgram, 1967). In this famous experiment letters containing a target person were sent to different residents in the U.S. Instructions were given to forward it to a person they knew that was more likely to know the target. The average path length for the letters that actually made it to the target was between 5 and 6, indicating a short path between nodes in the social network of the U.S.—it's a small world! The small-world property is typically defined as $\langle d \rangle \propto \frac{\langle k \rangle}{N}$.

The small-world property can be explained by the presence of **cliques** and **hubs** in the networks (Barabási and Albert, 1999; Strogatz, 2001). Cliques are sub-networks where almost every node has a link between them. Hubs are nodes with very high degree, far beyond the average degree of the system. In a social network a clique would be a school class or a work place, and a hub would be a celebrity or a politician.

2.1.1 Different Types of Networks

A very basic type of network is the regular **grid** where every node has the same degree, except the nodes on the edge of the network. Figure 2.2 shows a grid network of 100 nodes along with its degree distribution. A grid network has a very **homogeneous degree distribution**—almost all nodes have the same degree and the standard deviation is very small. Grids do not have the small-world property.

Another type of network that will be used a lot in this thesis is the **random network**. It is generated by giving some probability that each pair of nodes are neighbours. The resulting network does not have hubs, but it does have a high amount of cliques, which gives it the small-world property. Like the grids, the random networks have quite homogeneous

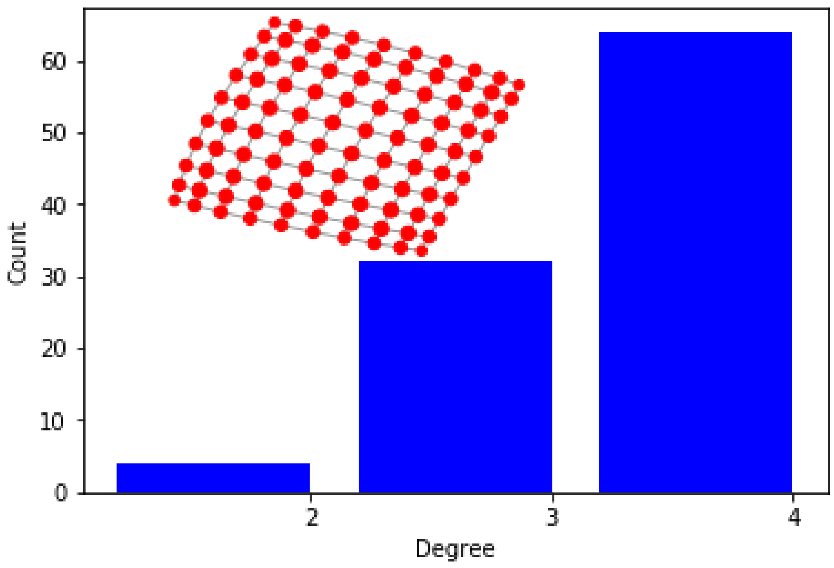


Figure 2.2: A grid of 100 nodes and 180 edges along with its degree distribution. This network has a diameter of 18. Nodes are sized relative to their degree.

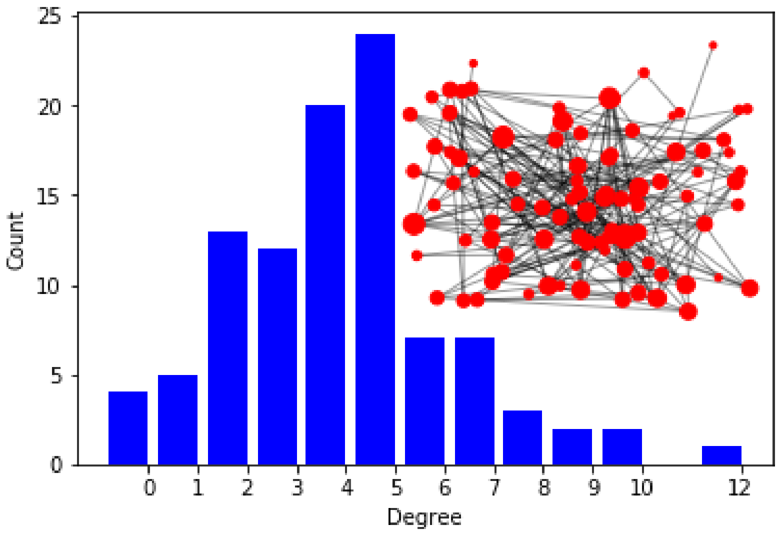


Figure 2.3: A random network of 100 nodes and 216 edges along with its degree distribution. This network has an infinite diameter as it is disconnected. Its giant component has a diameter of 6. Nodes are sized relative to their degree.

degree distribution, namely a binomial distribution. An example of a random network along with its degree distribution can be seen in Figure 2.3.

As it turns out, random networks and grids are seldom seen in real-world network systems, where the degree distribution typically more resembles a power-law. These networks are called **scale-free networks** and are often referred to as having a **heterogeneous degree distribution**—where most nodes are far from the average degree. Both cliques and hubs are present in these networks, typically giving them even shorter diameters than random networks. An example can be seen along with its degree distribution in Figure 2.4.

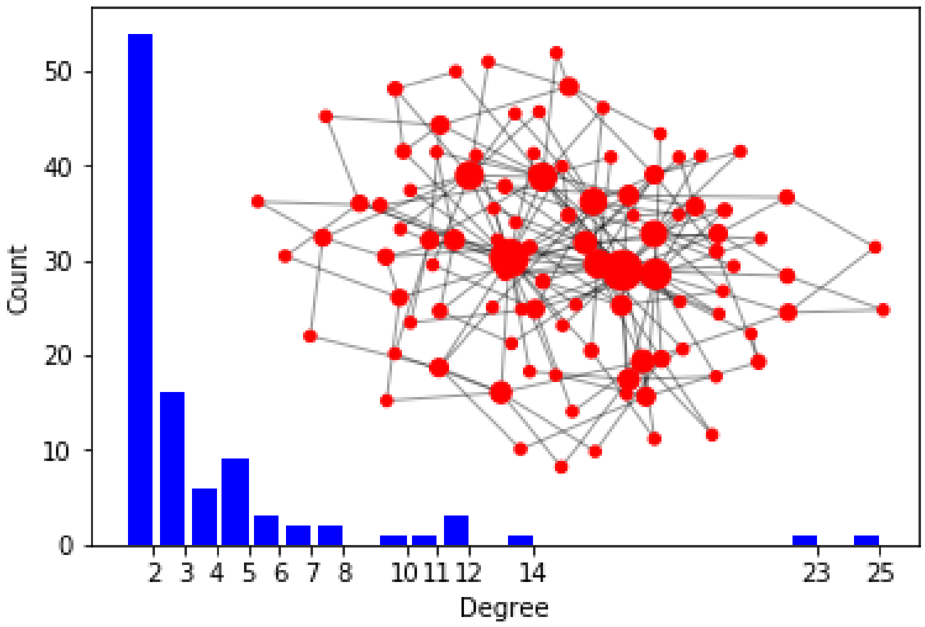
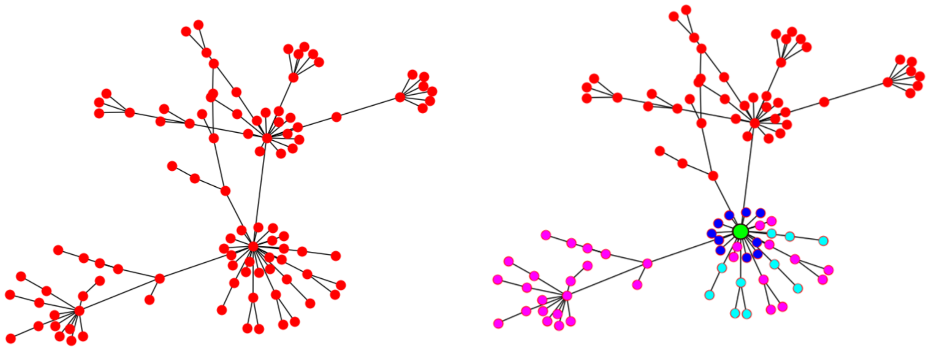


Figure 2.4: A scale-free network of 100 nodes and 196 edges along with its degree distribution. This network has a diameter of 5. Nodes are sized relative to their degree.

2.2 Cascading Failures in Networks

In this section network failure is considered and the difference between static and dynamic failure is explained. A closer look will be taken at two dynamic models for cascading failures in networks. Parts of this section are taken from my project thesis on the same topic (Jensen, 2018).

As briefly mentioned in the introduction, one way to view network robustness is to look at static properties of the networks, and consider how different network architectures handle the failure of a random subset of nodes. The World-Wide-Web is an example of a network



(a) Network with hubs.

(b) The green hub fails, and the network is fragmented into bits.

Figure 2.5: Networks with hubs (a) are vulnerable to attack on the nodes. In (b) the effect of failure of the green hub is illustrated. The network is fragmented into 20 separate parts. The blue nodes are single nodes that will be completely separated from the network. The cyan and magenta nodes are small networks that will also be separate. The red nodes represent the remaining giant component which is now less than half the size of the original network.

where these studies would be applicable. There is no dynamic load redistribution when a web-page is removed, it only affects the links to and from the page, as well as the page itself. The question to answer has typically been how many random nodes can be removed before the network is fragmented into disconnected components, or before there are pages on the web that are no longer reachable from anywhere (see Figure 2.5).

These studies have yielded the result that the typical real-world networks are very robust to random node failure. This is because most nodes and edges are not essential for the **connectedness** of the network and the chances of selecting a hub randomly is negligible. They are however more vulnerable to deliberate attacks on the hubs, as their failure may greatly increase average travel distances in the network or even disconnect it as seen in Figure 2.5 (Albert et al., 2000). Random networks are less resilient against random failures compared to scale-free networks, but it is also impossible to find especially vulnerable nodes to attack (Erdos and Renyi, 1959).

The model that is investigated in this thesis simulates how the random failure of a node can spread through the network, potentially causing a cascade of failure through the network. This type of model is applicable when the removal of nodes or links in the network creates a disturbance of the remaining nodes and edges, potentially causing new nodes to fail. This could be the case for example in a power-transmission grid where power-load is redistributed across the grid in the case of failure. Or the internet where routers transmit information, and failures may cause congestion in neighbouring nodes. This type of dynamic network failure model has been studied previously, and the results have shown the extreme vulnerability of networks with hubs when spreading to neighbouring nodes is made possible (Motter and Lai, 2002; Crucitti et al., 2004; Strogatz, 2001; Schläpfer and Shapiro, 2009).

The paper *Cascade-based attacks on complex networks* by Adilson E. Motter and Ying-Cheng Lai from 2002 presents a model for dynamic network failure. At each time step one load unit is transmitted from every node to every node through the shortest path between them. Each node has a capacity proportional to its starting load. In the simulation one random node fails, and all the shortest paths that pass through that node has to be re-routed through new shortest paths in the network. Any nodes in the new shortest-paths gets an increase of load, and if their capacity is surpassed they fail, leading to new load redistributions and possibly new node failures.

With this model, the failure will either be contained in a new state of equilibrium as the load is redistributed to nodes that can handle it, or it will cause a cascade of failure through the network. As would be expected, hubs are vulnerable, as they likely have many shortest paths that go through them. The researches found that even when the node capacity was set to twice the initial load, an attack on a hub would lead to a 20 % reduction of the network. When simulations were initiated by the failure of a random node, the network reduction was insignificant. They concluded that real-world networks have evolved naturally to be robust to random failure, but that when failure spreading is possible, they are extremely vulnerable to attack on even a single important node, potentially causing not only a fragmentation of the network (as may be the case in static models), but a complete failure.

Another model for dynamic network failure is the Susceptible-Infected (**SI**) Network Model used in epidemic modeling (Barabási, 2016). In a population of N individuals a new disease is about to break out. Initially one individual is infected (I) and all the others are susceptible (S). At each time step, every infected node has a chance of transmitting the disease to anyone it comes in contact with (it's neighbouring nodes).

In the SI model the disease will reach all nodes at some point, so any protective features of the topology of the network will only affect how quickly this happens. Again the hubs play an important role, but in addition to them being "super-spreaders" as in the previous model, they are also much more susceptible than other nodes—they are much more likely to be neighbours with an infected node. So in this model, it makes little matter if the infection (or failure) starts in a low- or high-degree node, as a hub will be reached within a few steps either way. The deciding factor for the speed of spreading is the network topology, with the presence of hubs showing detrimental effects on the failure rate. This effect is different from some other network failure models, where hubs can be protective factors—at least when they are not specifically targeted initially.

The SI model is sometimes expanded to include a recovery rate, which recovers infected individuals either back to a state of susceptible (S), or to a new state of recovered (R) and no longer susceptible. These expansions are known as the SIS and SIR models respectively. In the SI model all individuals will be infected eventually, but by introducing the possibility of recovery in the SIS model, there are two possible outcomes of a disease breakout. The recovery rate may be large enough to outperform the infection rate and then result in a disease free end state. Alternatively, the recovery rate is not large enough, and the disease spreading will be similar to that of the SI-model. The difference is that instead of ending at all infected nodes, it ends at some steady-state fraction of the population infected where there is equilibrium between the number of infected and cured individuals. With the introduction of the new state of recovered (R) (and no longer susceptible), the only possible end state is that all individuals have recovered. The question to answer then becomes how many individuals the infection gets to before it dies out.

2.3 Graphical Analysis of Differential Equations

This section contains a brief presentation of the necessary background to understand the theoretical analysis of the model. The theoretical basis of this section is chapters 2 and 3 in the book *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering* by Steven H. Strogatz (Strogatz, 2015).

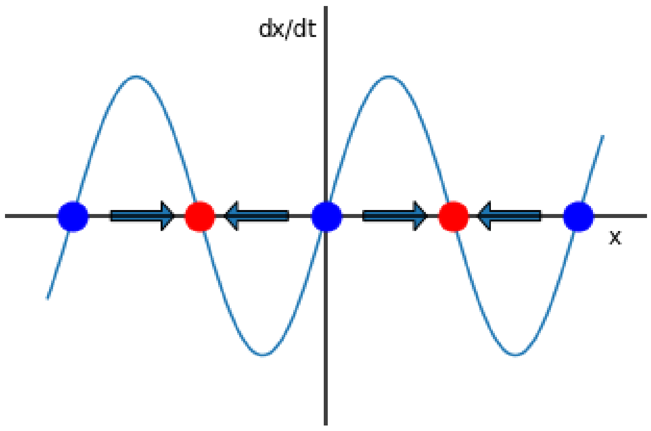
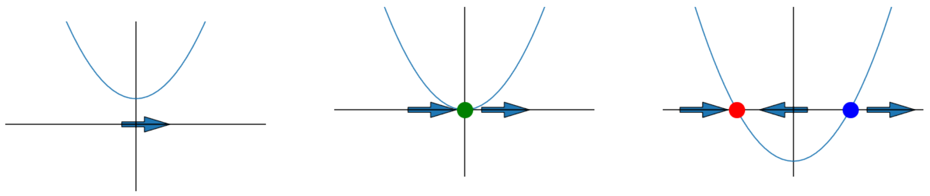


Figure 2.6: Plot of x vs. $\frac{dx}{dt}$ for $\frac{dx}{dt} = \sin x$. Flow is towards the stable fixed points in red and away from the unstable fixed points in blue.

Some non-linear differential equations are difficult or even impossible to solve analytically. In addition, the analytical solution, if it exists, can be difficult to interpret. These equations can be analyzed by using a more graphical approach. Start with a first-order differential equation, $\dot{x} = f(x)$, and plot $\dot{x}$ versus x. The resulting graph can be used to sketch a vector field on the x-axis; where $\dot{x}$ is positive the flow is to the right, and where it is negative the flow is to the left. When $\dot{x} = 0$ there is a **fixed point** that is **stable** if the flow on both sides are towards it, or **unstable** if the flow is away from it. Figure 2.6 shows a graphical analysis of $\frac{dx}{dt} = \sin x$. Even though this sketch gives little quantitative information between the fixed points, the direction of the flow is easily extracted and is showed by the arrows. The red dots represent stable fixed points at $x_{stable} = \pm\pi$ and unstable fixed points at $x_{unstable} = 0$ and $x_{unstable} = \pm 2\pi$.

On the fixed points there is no flow in either direction, and the *stable* fixed points represent steady-state values for the system. But in real systems that are being modeled by a differential equation, the flow will never exactly hit an *unstable* fixed point. Instead, it functions as a *threshold*, for example between two stable fixed points. In the case of cascading failures in a network, there may be a stable fixed point at no failed nodes and all failed nodes, and between them there will be an unstable fixed point. This point will then function as a threshold—exceed it and the failure will cascade into total system failure—stay below it, and the system will move towards the “good” stable fixed point where everything is in working order.



(a) The parameter a is positive and there are no fixed points. The flow is to the right.

(b) The parameter a is zero and there is a half stable fixed point at the origin. The flow is still all to the right.

(c) The parameter a is negative and there are two fixed points. One stable at $x < 0$ and one unstable at $x > 0$.

Figure 2.7: Bifurcation in $\frac{dx}{dt} = x^2 + a$ as the parameter a is lowered.

Half stable fixed points also exist, and can show up when bifurcation happens. A **bifurcation** is when a parameter in the differential equation is adjusted so that there is a qualitative change in the dynamics of the system. For example if a fixed point is created or destroyed or changes its stability. Figure 2.7 shows a graphical analysis of the differential equation $\dot{x} = x^2 + a$. Initially the parameter a is positive, and there are no fixed points. When a is lowered until the graph of $\dot{x}$ touches the x-axis, a half-stable fixed point is created. Further lowering a yields two fixed points (one stable, one unstable) where before there were none. Changing a below the point where the bifurcation occurs has completely altered the behavior of the system. This point is sometimes refered to as the **bifurcation point** or **bifurcation value**.

Identifying values of the parameters that result in bifurcations can show the possible qualitatively different ways the system can act, and can be used to draw up a **phase space** diagram. An example of this can be seen in Figure 3.7 where the different value-combinations of the parameters α and $\tilde{g}$ determine what qualitative behavior the system will have. If $\alpha = 2$ and $\tilde{g} = 0.2$ the system has four fixed points. Two unstable ones, US, and two stable ones, S.

3 | Methods

In this chapter our model for cascading failure is described. A summary of the mathematical analysis of the model that Eran Reches did is given (Reches, 2019). Then the implementation of the model and the simulations is described in detail.

3.1 The Model - A Generalized SIS Model

In this model failure spreads across a network with N nodes. Each node x_i can be functioning, having a value of $x_i = 0$, or failed having a value of $x_i = 1$. If two nodes x_i and x_j are neighbours, then $A_{ij} = 1$, otherwise $A_{ij} = 0$.

On each time step every working node will have some chance of failure and each failed node will have some chance of being repaired. These probabilities are given by:

$$\mathrm{P}(\text{failure}) = \left(\frac{\sum_{j=1}^{N} x_j A_{ij}}{\sum_{j=1}^{N} A_{ij}} \right)^{\frac{1}{\alpha}}, \tag{3.1}$$

where α will be referred to as the **sensitivity** of the system, and

$$\mathrm{P}(\text{repair}) = \frac{g}{1 + Nx}. \tag{3.2}$$

where g represents the **repair capacity** of the system. The x can be viewed as the relative fraction of infected nodes or the **average response** of the network given by:

$$x = \frac{1}{N} \sum_{j=1}^{N} x_j.$$

The model will be referred to either just as *our model* or a *generalized SIS model.*

3.2 Theoretical Analysis - The Phase Space

This section is a summary of the analyses that were performed by Eran Reches (Reches, 2019).

The model as presented in section 3.1 can be written as a continuous differential equation:

$$\frac{dx_i}{dt} = -\underbrace{\frac{g}{1+Nx}}_{\text{repairing term}} x_i + (1-x_i)\underbrace{\left(\frac{\sum_{j=1}^{N} x_j A_{ij}}{\sum_{j=1}^{N} A_{ij}}\right)^{\frac{1}{\alpha}}}_{\text{failure term}} .$$

Here each node x_i has a value ranging from 0 to 1, functional and failed respectively. N is the number of nodes in the network, $x = \frac{1}{N}\sum_{j=1}^{N} x_j$ is the average response of the network (the fraction of failed nodes), α is the sensitivity measure, and g is the repair capacity.

When studying stochastic models with a large number of individual components that interact, **mean field** approximations are a useful tool to simplify the model. Instead of having to deal with each individual component, an assumption is made that the component values can be approximated to the system average. Using mean-field dynamics the equation is simplified by setting the x_i to their average value x. The resulting differential equation describes the average behavior of the model as follows:

$$\frac{dx}{dt} = -\underbrace{\frac{gx}{1+Nx}}_{h_1(x)} + \underbrace{(1-x)x^{\frac{1}{\alpha}}}_{h_2(x)} . \tag{3.3}$$

A fixed point can be found wherever $\frac{dx}{dt} = 0$, which is where $h_1(x) = h_2(x)$. There will be flow to the right when $h_1(x) < h_2(x)$ and flow to the left when $h_1(x) > h_2(x)$.

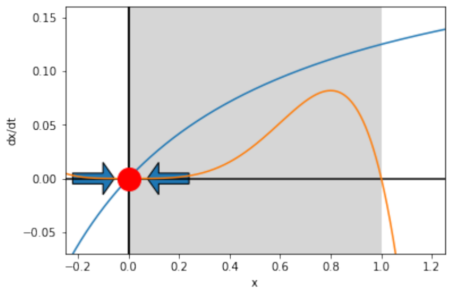


Figure 3.1: $\alpha = 0.25$, $g = 0.25$ and $N = 1$. One “good” stable fixed point at $x = 0$.

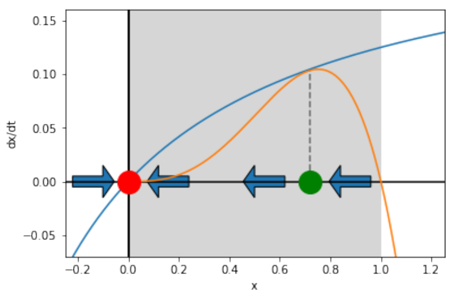


Figure 3.2: $\alpha = 0.33$, $g = 0.25$ and $N = 1$. One "good" stable fixed point at $x = 0$, and one half-stable fixed point at $x \approx 0.7$.

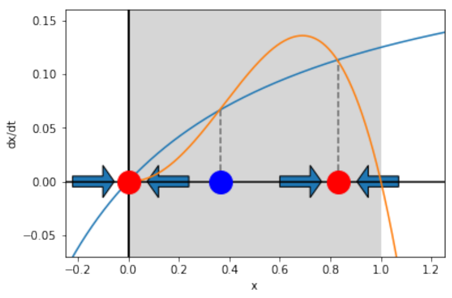


Figure 3.3: $\alpha = 0.45$, $g = 0.25$ and $N = 1$. One "good" stable fixed point at $x = 0$, one unstable fixed point at $x \approx 0.35$, and one "bad" stable fixed point at $x \approx 0.8$.

Figures 3.1, 3.2 and 3.3 shows the different fixed points that can show up when $\alpha < 1$. The shaded area marks the x-values that are possible in our model. In Figure 3.1 the sensitivity is so low that the only fixed point is the "good" stable fixed point (red) at $x = 0$ (no failed nodes) and all flow is to the left. In Figure 3.2 a bifurcation happens. Figure 3.3 shows the resulting "bad" stable fixed point in red and the unstable fixed point that functions as a *threshold* for the flow in blue.

Figures 3.4, 3.5a and 3.6a show the qualitatively different states when $\alpha > 1$. In Figure 3.4 the repair rate is so low that the only stable fixed point is the "bad" one near total system failure. As repair capacity is increased, two new fixed points show up. One unstable at around $x = 0.1$, and one stable close to the origin can be seen better in Figure 3.5b.

As repair capacity is further increased like in Figure 3.6a, another bifurcation makes the "bad" stable fixed point and the threshold disappear. Now only the "good" stable point close to the origin remains along with the unstable point at $x = 0$ (Figure 3.6b).

These different possible outcomes are summarized in the phase space shown in Figure 3.7. This Figure is from Eran Reches' unpublised analysis *Generalized SI Model: Including Sensitivity* (Reches, 2019). There is always a fixed point at the origin, and it is marked in red. The parameter $\tilde{g} = \frac{g}{N}$ is used instead of g to limit it between 0 and 1.

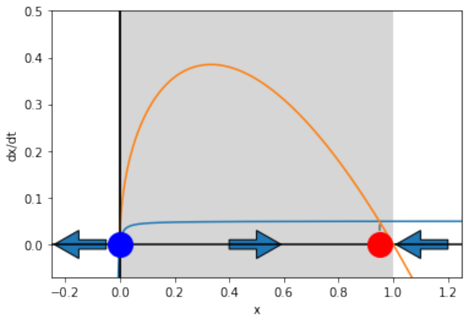
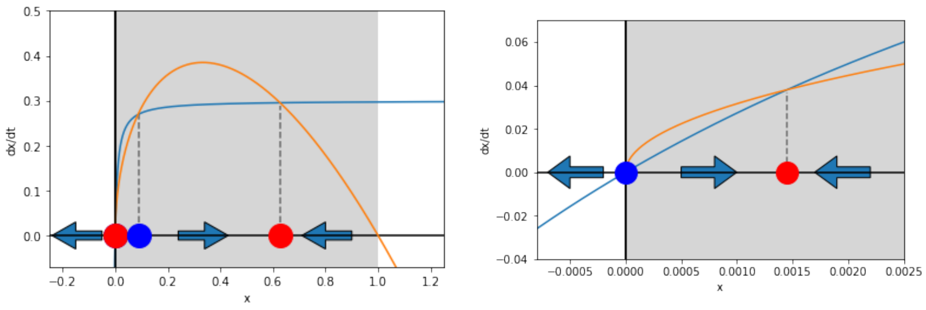


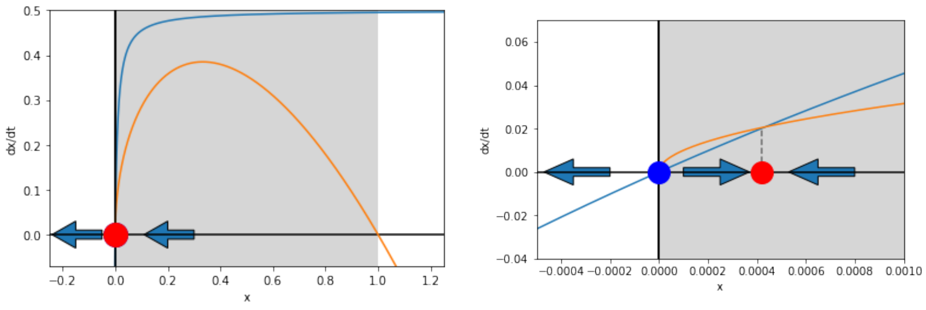
Figure 3.4: $\alpha = 2$, $g = 5$, $N = 100$. One unstable fixed point at $x = 0$, and one “bad” stable fixed point at $x \approx 0.95$.



(a) Four fixed points.

(b) Zoom in at origin.

Figure 3.5: $\alpha = 2$, $g = 30$, and $N = 100$. One unstable fixed point at $x = 0$, one “good” stable fixed point at $x \approx 0$, one unstable fixed point at $x \approx 0.1$, and one “bad” stable fixed point at $x \approx 0.6$.



(a) Two fixed points.

(b) Zoom in at origin.

Figure 3.6: $\alpha = 2$, $g = 50$, and $N = 100$. One unstable fixed point at $x = 0$, and one “good” stable fixed point at $x \approx 0$.

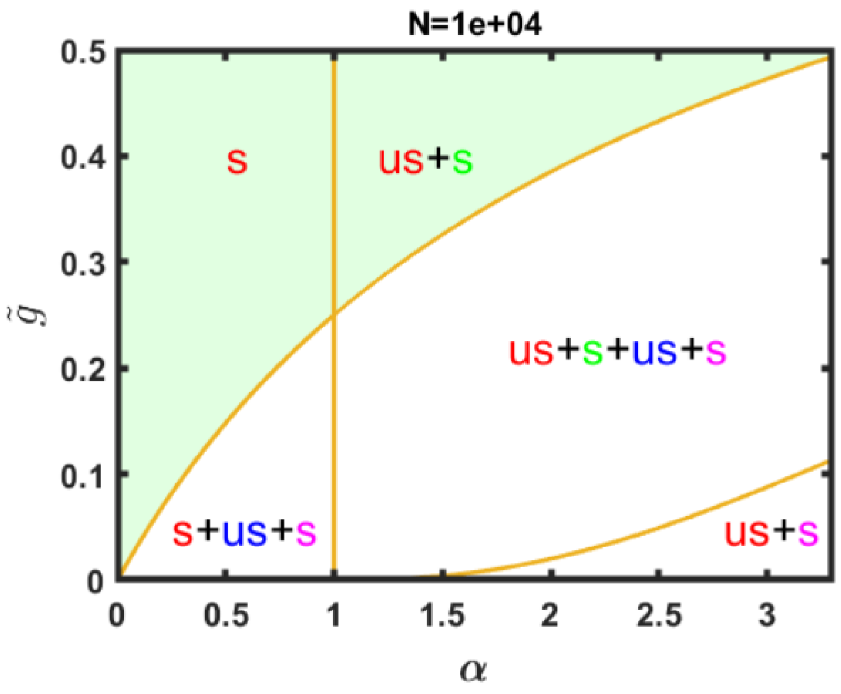


Figure 3.7: Phase space. Figure from the unpublished analysis of Eran Reches (Reches, 2019).

3.3 Probing the Phase Space

The following procedure will be repeated for different values of the **sensitivity** α and the **repair capacity** $\tilde{g}$ to look for different fixed points.

- Set initial values for N, α, $\tilde{g}$.
- Values are chosen for x ranging from 0 to 1 step-wise.
- Simulations are run for all the different values of x until they reach a steady state.
- The different steady states are recorded.
- The x are plotted against time to identify thresholds.

Steady states represent stable fixed points in the phase space. Thresholds represent unstable fixed points. Some of the points may be impossible to find in the simulations, such as the unstable fixed point at zero for $\alpha > 1$. Mainly the stable fixed points referred to as the "good" and the "bad", along with the threshold between them will be attempted found. The "good" stable fixed point refers to the stable fixed point at $x = 0$ or at $x \approx 0$. The "bad" stable fixed point refers to the stable fixed point where most of the network has failed.

To simulate continuous time, many of the simulations will also be run with the time steps divided into smaller dt. The failure chance and repair chance in equations 3.1 and 3.2 will then be calculated by multiplying with dt as follows:

$$\mathrm{P(failure)} = \left(\frac{\sum_{j=1}^{N} x_j A_{ij}}{\sum_{j=1}^{N} A_{ij}} \right)^{\frac{1}{\alpha}} \cdot dt, \tag{3.4}$$

and

$$\mathrm{P(repair)} = \frac{g}{1+Nx} \cdot dt. \tag{3.5}$$

3.4 Resources

The simulation of the model is written in Python and run on networks generated using NetworkX. NetworkX is a Python package for creation and analysis of complex networks (Hagberg et al., 2008).

All figures of networks in this thesis are made using networks generated in NetworkX, and drawn using the Python library Matplotlib (Hunter, 2007).

3.5 Implementation

3.5.1 Network Generation

When networks are generated a built-in function in NetworkX is used. For random networks, especially those of lower average degree, there is no guarantee that the networks generated are connected. This can cause problems in the simulations, since a failure initiated in a disconnected component will never reach the whole network. To avoid this possible source of error only the giant component is kept after the network is generated. This will increase the average degree of the network somewhat, since nodes are removed, but not edges. Algorithm 1 shows the pseudo code of how networks were generated.

The **create_network** function takes as input:

- `N`: the number of nodes in the network.
- `degree`: the average degree in the network.

The function outputs:

- `G`: the network the simulation is to be run on.
- `N`: the number of nodes in the network.

```
function create_network:
    Input: N, degree

    # Use NetworkX function to create random network
    G_disconnected ← nx.erdos_renyi_graph(N, degree/N)

    # Remove disconnected components
    G = giant component in G_disconnected
    N = number of nodes in G
    return G, N
```

Algorithm 1: Pseudo code for network generation.

3.5.2 Simulations

Algorithm 2 shows the pseudo code of a single simulation on a network. Algorithm 3 shows how many simulations were run in order to look at averages.

The **simulation** function takes as input:

- `G`: the network the simulation is to be run on.
- `N`: the number of nodes in the network.
- `g`: the parameter g in equation 3.2.
- `alpha`: the parameter α in equation 3.1.
- `start_nodes`: a list of nodes that are failed at the beginning of the simulation.
- `time_steps`: how many time steps the simulation should run for.
- `dt`: how time steps are divided.

The function outputs:

- `x`: a list that has recorded the fraction of failed nodes at each dt.

The functions **calculate_fail_chance** and **calculate_repair_chance** uses the probabilities stated in equations 3.4 and 3.5 respectively.

The **many_simulations** function takes as input:

- `N`: the number of nodes in the network.
- `avg_deg`: the average degree in the network.
- `alpha`: the parameter α in equation 3.1.
- `g_tilde`: used to calculate the parameter g in equation 3.2. $g = \tilde{g}N$
- `x`: the fraction of failed nodes at the beginning of each simulation.
- `time_steps`: how many time steps the simulation should run for.
- `dt`: how time steps are divided.
- `realizations`: how many different networks the simulation is to be run on.
- `triggers`: how many simulations are to be run on each network.

The function outputs:

- `mean_results`: the mean of the outputs from **simulation**.
- `std_results`: the standard deviation of the outputs from **simulation**.

```
function simulation:
    Input: G, N, g, alpha, start_nodes, time_steps, dt

    # Use network F to keep track of failed nodes
    F ← add start nodes to network F

    # Update the number of failed neighbours
    for failed_node in F:
        for nbr of failed_node in G:
            increment num_failed_nbrs

    # Set the fraction of infected nodes, x
    x = (number of nodes in F)/N

    for each time step:
        for each dt:
            # Initialize empty networks
            NF  newly failed
            NR  newly repaired
            T   tested

            # Test which nodes fail on this dt
            for failed_node in F:
                for nbr of failed_node in G:
                    if nbr not in F and nbr not in T:
                        T ← add node nbr
                        fail_chance = calculate_fail_chance
                        if random number < fail_chance:
                            NF ← add node nbr

            # Update num_failed_nbrs and F
            for failed_node in NF:
                for nbr of failed_node in G:
                    increment num_failed_nbrs
            F ← add nodes from NF

            # Test which nodes are repaired on this dt
            repair_chance = calculate_repair_chance
            for failed_node in F:
                if random < repair_chance:
                    NR ← add node failed_node

            # Update num_failed_nbrs and F
            for repaired_node in NR:
                for nbr of repaired_node in G:
                    reduce num_failed_nbrs
            F ← remove nodes from NR

            # Set the fraction of infected nodes, x
            x ← append (number of nodes in F)/N
    return x
```

Algorithm 2: Pseudo code for a simulation.

```
function many_simulations:
    Input: N, avg_deg, alpha, g_tilde, x, time_steps,
            dt, realizations, triggers

    for each realization:
        # Create a new network G
        create_network(G,N)
        g = g_tilde * N
        start_nodes ← add x * N random nodes in G

        for each trigger:
            # Run a simulation on the network G
            results ← append simulation(G,N,g,alpha,
                                start_nodes,time_steps,dt)

    mean_results ← append mean of results
    std_results ← append standard deviation of results

    return mean_results, std_results
```

Algorithm 3: Pseudo code for many simulations.

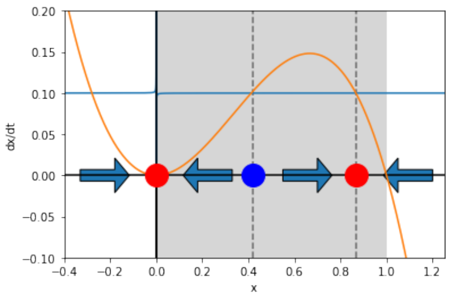
4 | Results

In this chapter the results from the simulations are presented. Some explanation and attempts at locating fixed points is done, but more thorough discussion of what the results mean is done in chapter 5.

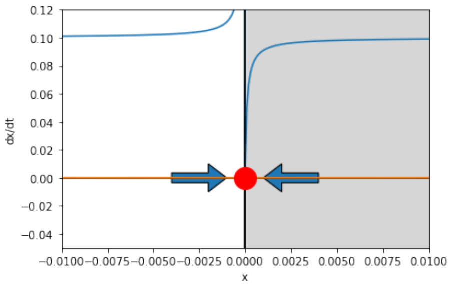
4.1 About the Visualisations

In this thesis two different color schemes are used. In the first color scheme the color red is used to indicate stable fixed points and steady-states. Blue is used for unstable fixed points and thresholds. The color coding of simulation results are meant to give an indication to the fraction of simulations that went to which stable fixed point. If all simulations went to the same point, the line will be red. If 50 % went up and 50 % down, the line will be blue, then other fractions are something in between. In the second color scheme the color green is used together with red to plot simulation results and the *color coding* of fraction to the same state *no longer applies*.

Unless otherwise states **all simulation results are averages over 5 triggers on 5 network realizations**. This number was chosen because of limited computational power, but also



(a) Three fixed points.



(b) Zoom in at origin.

Figure 4.1: $\alpha = 0.5$, $g = 1000$ and $N = 10000$.

because simulation results showed quite small variations. The standard deviation is plotted as a shaded area around the graphs to show this. Only simulations initiated at x close to a threshold value show large standard deviations. And this is because simulations ended up at different steady state values.

All simulations are done on networks with approximately **10 000 nodes**. The networks have an **average degree of 4**, except for the cases when average degree is looked closer at in sections 4.3 and 5.5. All simulations are done on **random networks**, except the results from section 4.5, where grids and scale-free networks are used.

4.2 Low Sensitivity ($\alpha = 0.5$)

This section looks at simulations done with $\alpha = 0.5$ on random networks of $N \approx 10000$ nodes.

4.2.1 Low Repair Capacity ($\tilde{g} = 0.1$)

From the phase space presented in section 3.2 in Figure 3.7 the fixed points predicted by analysis can be found. A 10 000 node network with $\tilde{g} = 0.1$ and $\alpha = 0.5$ is expected to have a "good" stable fixed point at the origin and another "bad" stable fixed point near the total failure of the system. Between them is an unstable fixed point. $h_1(x)$ and $h_2(x)$ from equation 3.3 for this case is plotted in Figure 4.1. The unstable fixed point is at $x \approx 0.42$ and the stable fixed point is at $x \approx 0.87$. Figure 4.1b is zoomed in on the origin to show

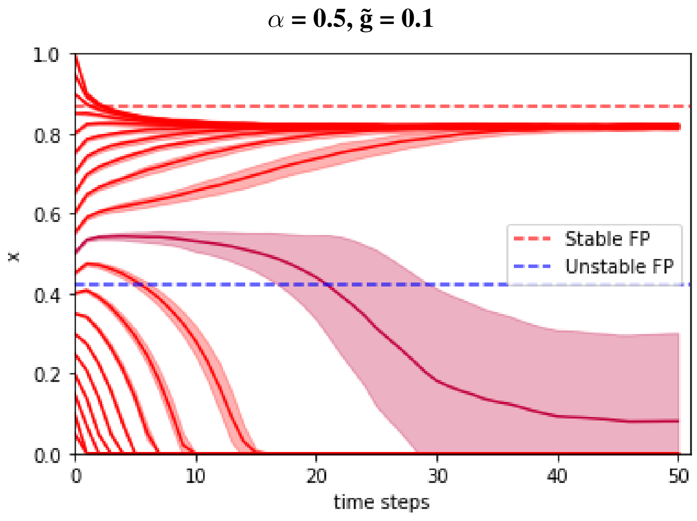


Figure 4.2: Simulations with $dt = 1$. Shaded area show standard deviation.

that the two lines of $h_1(x)$ and $h_2(x)$ do in fact intersect at zero, which results in the stable fixed point there.

Figure 4.2 shows results from simulations run on random networks of 10 000 nodes, average degree 4, $\alpha = 0.5$, $\tilde{g} = 0.1$ and $dt = 1$. The initial values of x are from 0.05 to 1.0 with increments of 0.05. The values of the fixed points predicted by the analysis are marked with stipled lines. The results from these simulations indicate that the "bad" stable

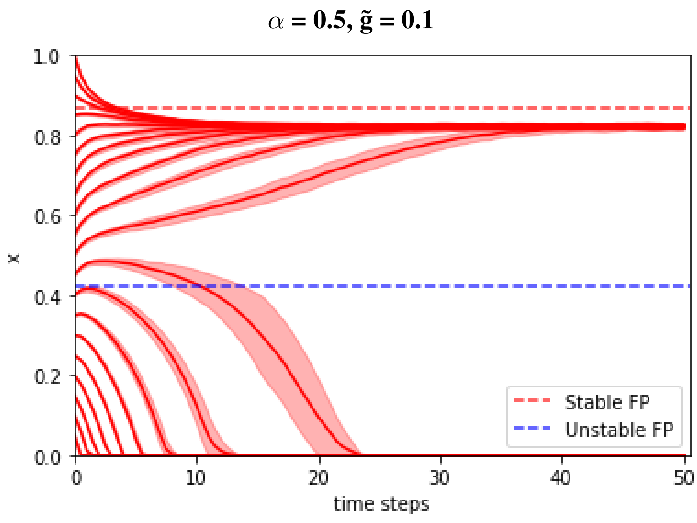


Figure 4.3: Simulations with $dt = 0.5$. Shaded area show standard deviation.

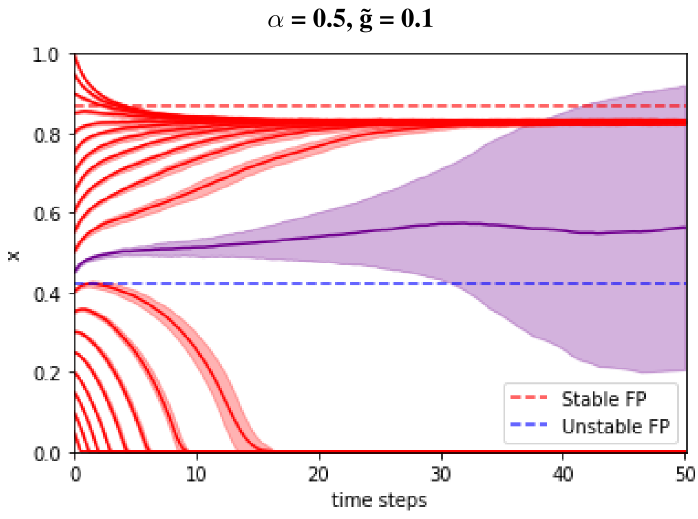


Figure 4.4: Simulations with $dt = 0.1$. Shaded area show standard deviation.

fixed point is around $x \approx 0.8$, and the unstable fixed point is somewhere in the interval $x = (0.45,\ 0.55)$.

The same simulations were run again with $dt = 0.5$. Figure 4.3 shows the results. The "bad" stable fixed point is still at around $x \approx 0.8$. The interval where the unstable fixed point is located is somewhat narrower here. All simulations initiated at $x = 0.5$ went up too the "bad" stable fixed point, and all simulations started at $x = 0.45$ went down to the "good" stable fixed point. This indicates that the unstable fixed point is somewhere in the interval $x = (0.45,\ 0.50)$.

A further decrease of dt was attempted. Figure 4.4 show results from simulations where $dt = 0.1$. Now the unstable fixed point is likely located in the interval $x = (0.40,\ 0.50)$, as simulations started at $x = 0.45$ were the only ones that split both ways.

4.2.2 Locating the Unstable Fixed Point

In order to more finely locate the unstable fixed point, or the *threshold* x-value, more simulations were run started near the probable locations found in the preceding subsection 4.2.1.

Figure 4.5 show simulations where $dt = 1$ and x ranges from 0.45 to 0.54 with increments of 0.01. Two of the initial x-values showed a split in their steady-state value: $x = 0.50$ and $x = 0.51$. 44 % of the simulations started at $x = 0.51$ went to the "good" stable fixed point, and 56 % went up to the "bad", making $x \approx 0.51$ a strong contender for the location of the unstable fixed point.

Results from simulations for $dt = 0.5$ is shown in Figure 4.6. x ranges from 0.45 to 0.49

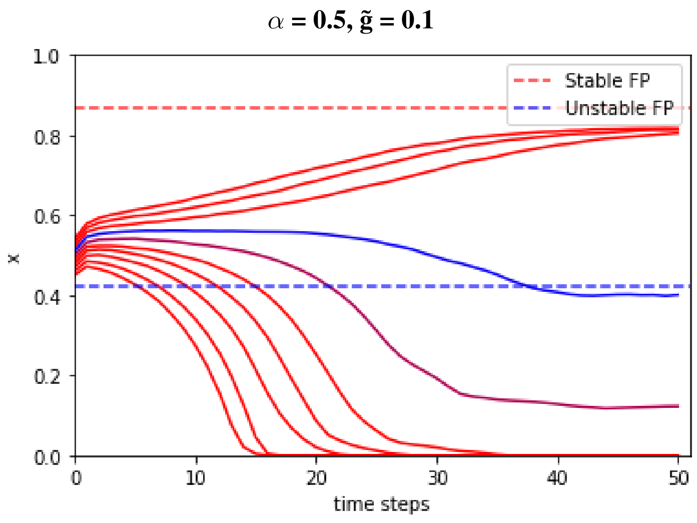


Figure 4.5: Simulations with $dt = 1$. x ranges from 0.45 to 0.54.

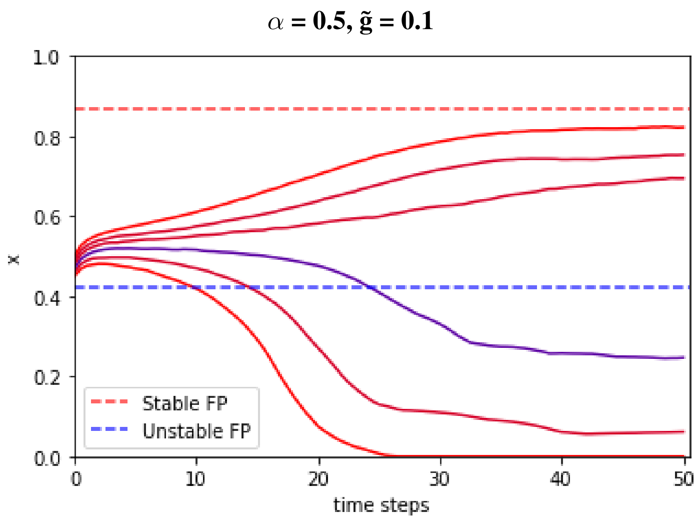


Figure 4.6: Simulations with $dt = 0.5$. x ranges from 0.45 to 0.49.

with increments of 0.01. The threshold seems to be located somewhere close to $x \approx 0.47$, where 68 % of simulations went to the "good" stable fixed point and 32 % went to the "bad" stable fixed point.

Results from simulations for $dt = 0.1$ is shown in Figure 4.7. x ranges from 0.42 to 0.47 with increments of 0.01. The threshold seems to be located somewhere close to $x \approx 0.45$ and $x \approx 0.44$. Simulations started at these values had 68 % and 32 % end up in the "good"

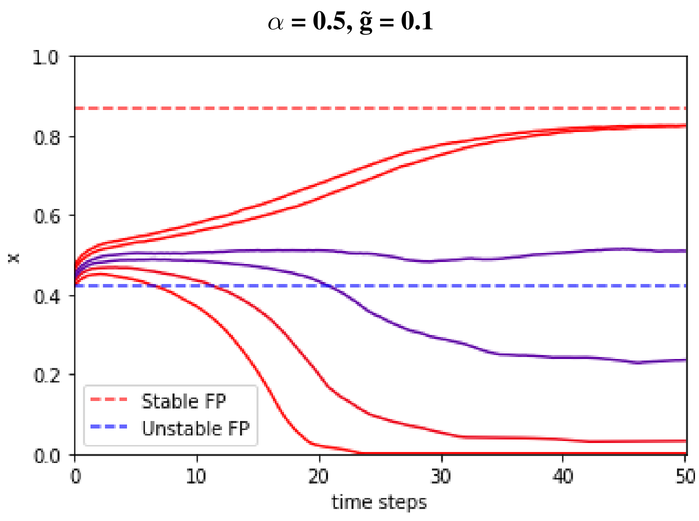


Figure 4.7: Simulations with $dt = 0.1$. x ranges from 0.42 to 0.49.

state, respectively.

Having taken a closer look at the threshold x-values for different dt, there is no longer any overlap between the likely locations. What is seen instead is that *the x-value of the threshold seems to approach the predicted unstable fixed point as dt gets smaller*. Why this happens is discussed in section 5.3. Why simulations at higher dt have a tendency to "jump" down and not up is also discussed.

4.2.3 Locating the "Bad" Stable Fixed Point

Continuing to look at the case of $\alpha = 0.5$, $\tilde{g} = 0.1$ on random networks of 10 000 nodes, an attempt is made to find the x-value of the "bad" stable fixed point.

Figure 4.8 shows simulations started at $x = 0.8$ that all went to the "bad" steady state. As dt is decreased, the stabilisation seems to happen closer and closer to the predicted stable fixed point. For $dt = 1$ the value seems to be $x \approx 0.82$, increasing as dt is decreased up to $x \approx 0.83$ for $dt = 0.1$.

As with the locating of the unstable fixed point in section 4.2.2, the simulations show a tendency to *get closer to the predicted x-value as dt is decreased*. Further discussion on can be found in section 5.3.

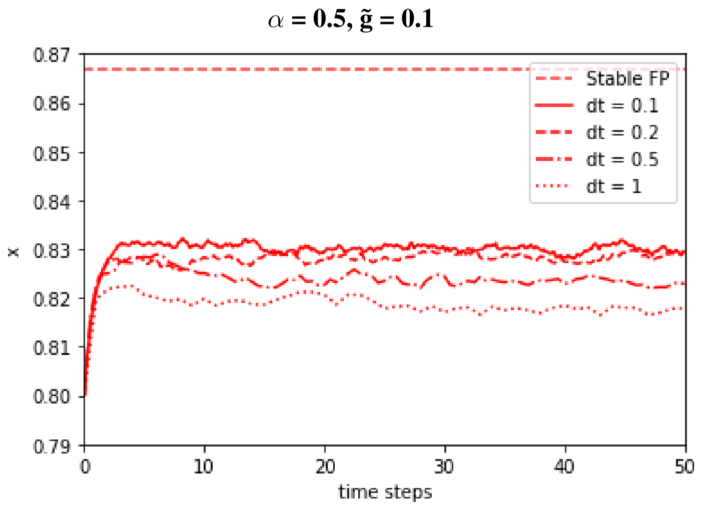


Figure 4.8: dt varies. $x = 0.8$.

4.2.4 Increasing the Repair Capacity ($\tilde{g} > 0.1$)

Still considering $\alpha = 0.5$ on 10 000 node random networks, the value of $\tilde{g}$ is increased to 0.15. In the phase space in Figure 3.7 this is close to the bifurcation where the "bad" stable fixed point disappears, making the dynamics more similar to the SI-model where all simulations go to the "good" stable fixed point at zero. Figure 4.9 shows the prediction for this case, where there is one stable fixed point at the origin, and one half-stable fixed point around $x \approx 0.67$. Because this second fixed point is half stable, all flow is down towards the "good" stable fixed point.

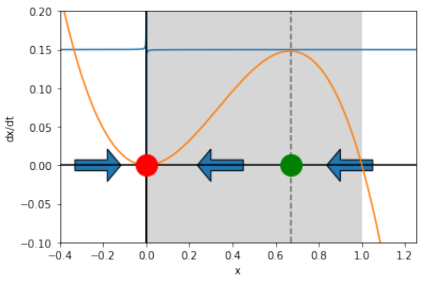


Figure 4.9: $\alpha = 0.5$, $g = 1500$, $N = 10000$. One "good" stable fixed point at $x = 0$, and one half-stable fixed point at $x \approx 0.65$

Figure 4.10 shows results from simulations run on random networks of 10 000 nodes, average degree 4, $\alpha = 0.5$, $\tilde{g} = 0.15$ and $dt = 0.1$. The initial values of x are from 0.05 to 1.0 with increments of 0.05. The results from these simulations indicate exactly what was predicted by the analysis: the "bad" stable fixed point has vanished, and all simulations go down to $x = 0$.

The general tendency of simulations to "jump" down and not up is also present when the bifurcation point is to be located. Figure 4.11 shows simulations started with $\alpha = 0.5$,

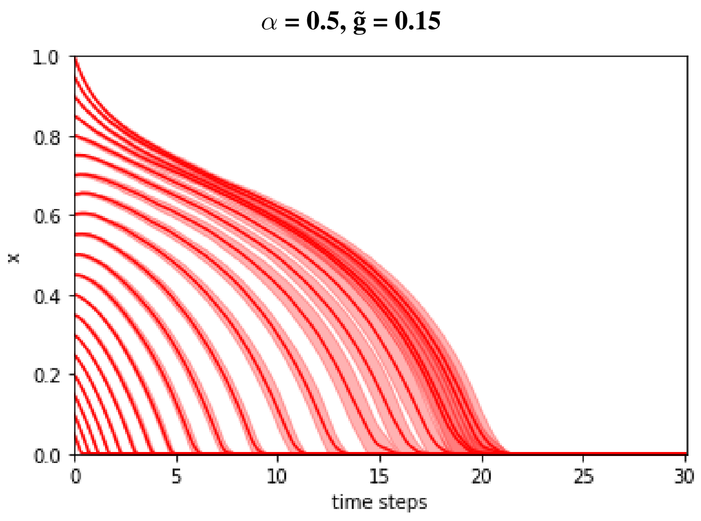


Figure 4.10: Simulations with $dt = 0.1$. Shaded area show standard deviation.

$\tilde{g} = 0.13$ and $dt = 1$. The analysis predicts the prescense of a "bad" stable fixed point here, but with bigger dt, the value of $\tilde{g}$ can be lower than the predicted value of 0.15, and still have all simulations go down to $x = 0$. The bifurcation when the "bad" stable fixed point vanishes happens at lower value of $\tilde{g}$. This is discussed further in section 5.3.

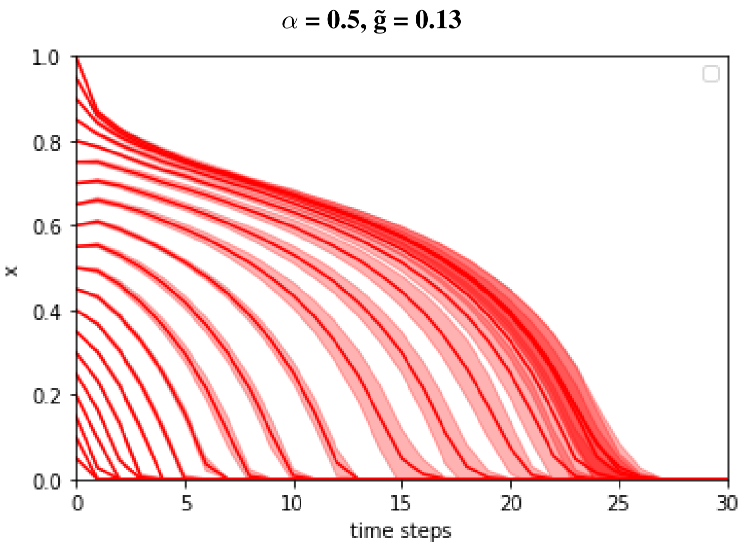


Figure 4.11: Simulations with $dt = 1$. dt is too big to locate the "bad" stable fixed point. Shaded area show standard deviation.

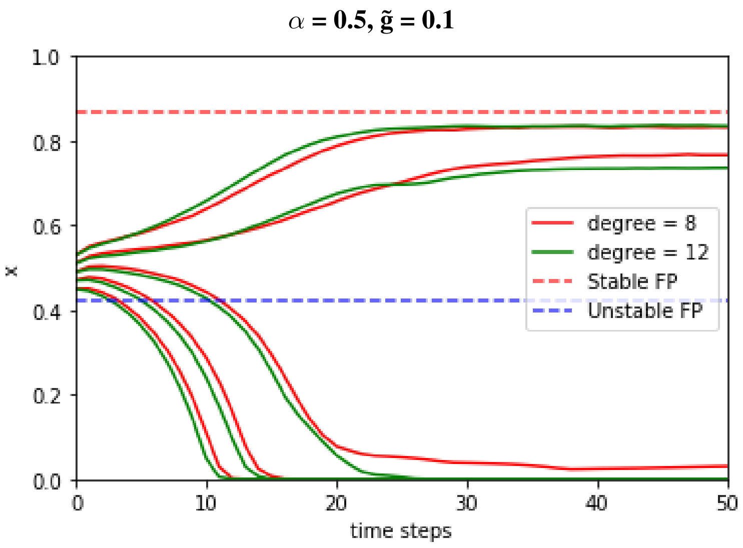


Figure 4.12: Networks of average degree 8 and 12. $dt = 1$.

4.3 Changing Average Degree

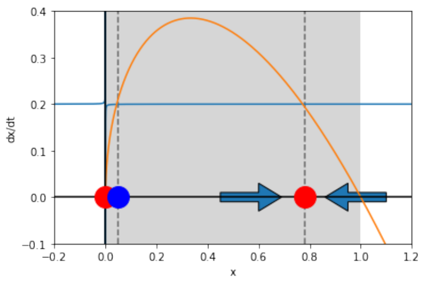
Some simulations were run on networks with different degrees. Figure 4.12 show results from simulations on 10 000 node networks with $\alpha = 0.5$, $\tilde{g} = 0.1$ and networks of average degree of 8 and 12. The simulations were started at initial x-values from 0.2 to 1 with increments of 0.2. There is a small tendency for the simulations on lower average degree networks to take longer to reach the steady states. Also a bigger fraction of the simulations on average degree 8 seem to go down to the "good" stable fixed point than those of average degree 12. Some thoughts on why this happens can be found in section 5.5.

4.4 High Sensitivity ($\alpha = 2$)

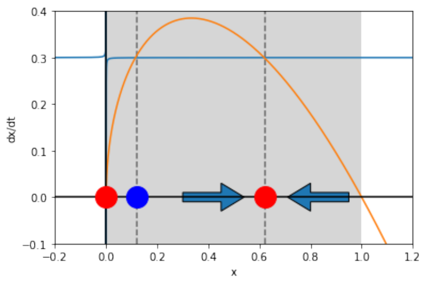
The phase space from the analysis can be seen in section 3.2 in Figure 3.7. A 10 000 node network with $\tilde{g} = 0.2$ and $\alpha = 2$ is expected to have four fixed points. Figure 4.13a shows a plot of $h_1(x)$ and $h_2(x)$ from equation 3.3 for these values. Only three of the fixed points are visible, because the unstable fixed point at $x = 0$ is so close to the "good" stable fixed point at $x \approx 0$. How these fixed points near the origin come to be is visualized better in the analysis section 3.2 in Figure 3.5b. For the practical purposes of the simulation it is not that important, since these fixed points are not possible to locate with the computational power available for this thesis.

The unstable fixed point is at $x \approx 0.05$ and the "bad" stable fixed point is at $x \approx 0.78$. Further increasing the repair capacity to $\tilde{g} = 0.3$ is seen in Figure 4.13b. This moves the threshold up to $x \approx 0.12$ and the "bad" point down to $x \approx 0.62$.

Simulation results with $\tilde{g} = 0.2$, $\alpha = 2$ and $dt = 0.1$ are shown in Figure 4.14. The "bad" steady state seems to be at around $x = 0.75$, and the unstable fixed point at x-values between 0.15 and 0.2.



(a) $\tilde{g} = 0.2$.



(b) $\tilde{g} = 0.3$.

Figure 4.13: $\alpha = 2$, $N = 10000$. Four fixed points. "Bad" stable fixed point and unstable fixed point move closer together as $\tilde{g}$ is increased.

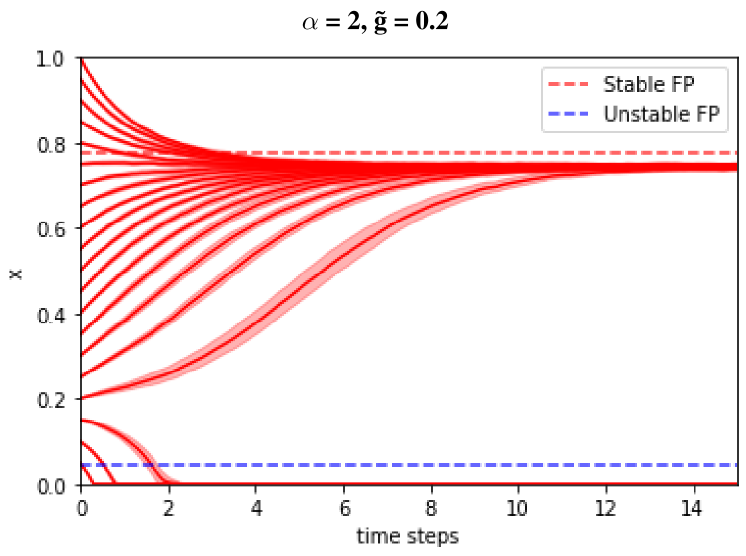


Figure 4.14: Simulations with $dt = 0.1$. Shaded area show standard deviation.

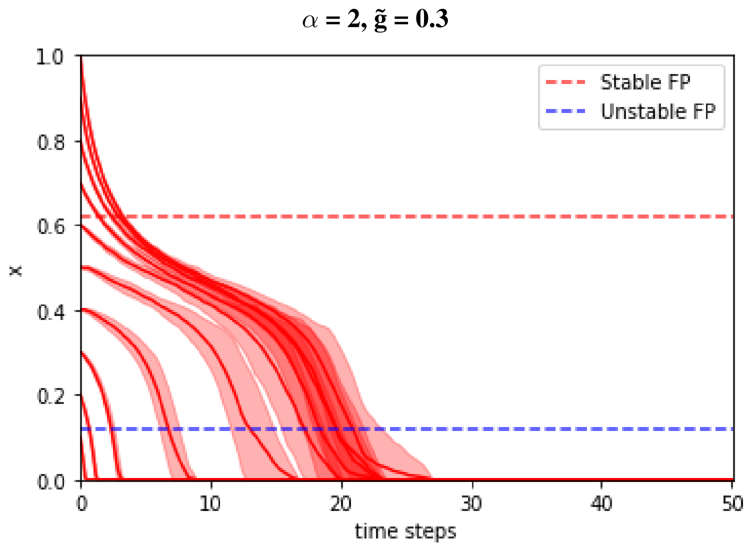


Figure 4.15: Simulations with $dt = 0.1$. "Bad" stable fixed point is not found. Shaded area show standard deviation.

Simulation results with $\tilde{g} = 0.3$ and $dt = 0.1$ are shown in Figure 4.15. According to the analysis, this scenario will have a "bad" steady state around $x = 0.62$ (see Figure 4.13b), but all simulations went down to the "good" steady state at $x = 0$. An attempt was made at studying this further by running simulations with $dt = 0.01$. This is at the

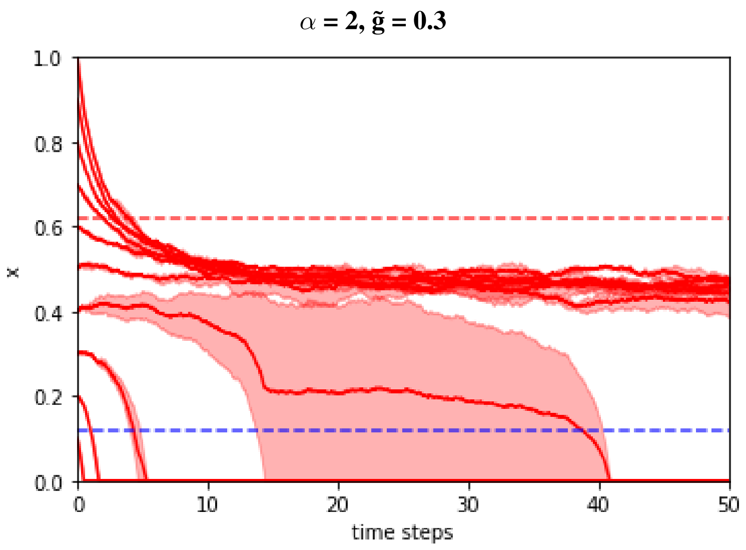


Figure 4.16: Simulations with $dt = 0.01$. With very small dt, the "bad" stable fixed point shows up, but at lower x-value than predicted. Shaded area show standard deviation.

limit of the computational power available for this thesis, so only 1 trigger was run on 2 realizations for each initial x-value. The results can be seen in Figure 4.16. Even with so few simulations it is clear that some stable fixed point has appeared around $x = 0.45$, with an unstable threshold close to $x \approx 0.4$. A discussion of this tendency for the simulations to "jump" down to the "good" final state can be found in section 5.3.

4.5 Other Networks

Simulations were also run on scale-free networks of 10 000 nodes with $\alpha = 0.5$, $\tilde{g} = 0.1$, and $dt = 0.1$. Simulations were started with x-values ranging from 0.05 to 1 with increments of 0.05. The results are shown in Figure 4.17. All simulations seem to do a big failure "jump" in the first time-step. This tendency is discussed in section 5.4. Ignoring the jump, the simulations look quite similar to the same simulations on random networks seen in Figure 4.4, but with the likely threshold x-value a little higher.

Simulations were also run on grids of 10 000 nodes with $\alpha = 0.5$, $\tilde{g} = 0.1$, and $dt = 0.1$. Simulations were started with x-values ranging from 0.05 to 1 with increments of 0.05. The results are shown in Figure 4.18. These results look very different from all the other simulations done on both random and scale-free networks. The simulations would have to be run for longer to make sure, but there is a possibility that there is no "bad" stable fixed point, and that all simulations slowly decrease their fraction of infected nodes down to zero. Alternatively, the "bad" stable fixed point is there, but with the unstable fixed point very close to it, making it hard to reach the "bad" state.

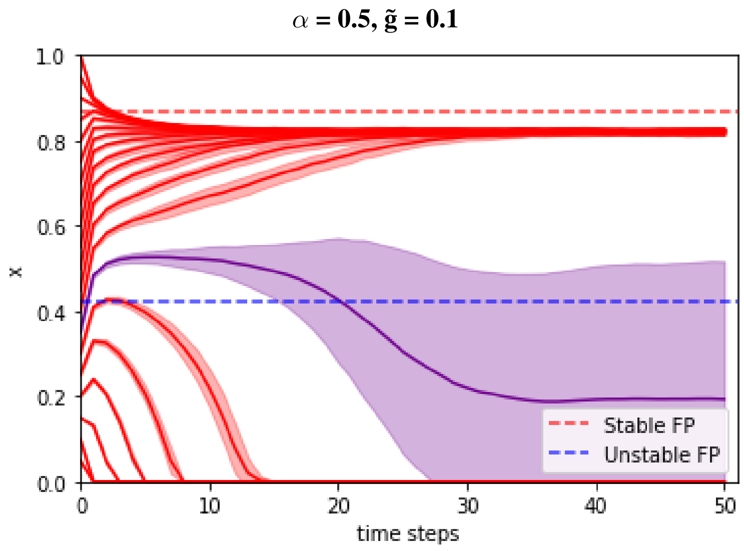


Figure 4.17: Simulations on scale-free networks with $dt = 0.1$.

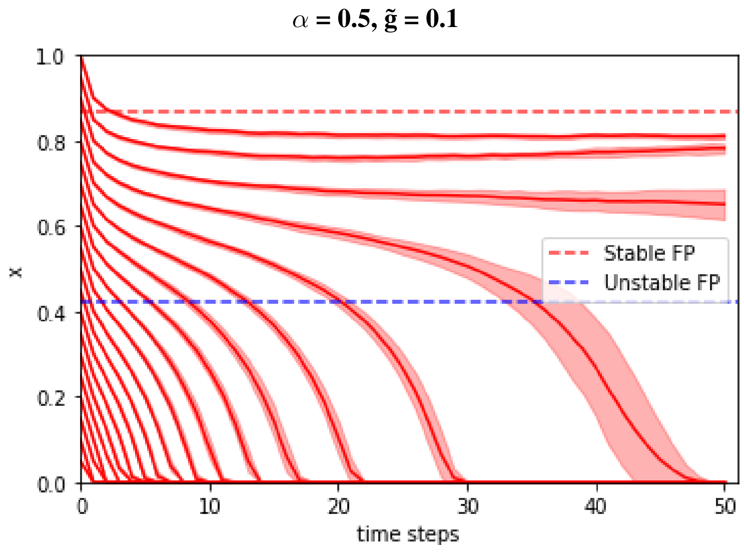


Figure 4.18: Simulations on grids with $dt = 0.1$

5 | Discussion

In this chapter the model itself, and the results from the simulations are discussed. An attempt is made to answer the questions stated in the introduction in section 1.2.

5.1 Features of our Model

Like with the SIS/SIR models for epidemiology, there is no accumulation of load in our model. Nodes only fail if they have one or more failed neighbours and the failure is independent on whether its neighbour was the first to fail or not. In practice this means that the weight of, or disturbance from, a failed component is only picked up by its closest neighbours, and is not redistributed across the whole network, as was the case with the model from the 2002 paper by Adilson E. Motter and Ying Cheng Lai (see section section 2.2). This feature *might* not translate well to the kinds of real-world situations where redistribution of load is an essential part of node failure like routers failing on the internet or power stations failing in the power grid. Our model instead considers how nodes are disturbed simply by having failing neighbours, which will be the case for example for sickness spreading and weeds.

The failure term of our model (equation 3.1) is similar to the SI-model which was presented in section 2.2 where failure can only spread to neighbours of failed nodes. One important difference is that nodes in our model are protected in proportion to their degree. So while hub nodes will be neighbour to a failed node quickly, making them vulnerable, they are still protected by their large number of neighbours. Our model also has the **sensitivity parameter** α which can amplify or dampen the failure chances of nodes.

The repairing term in our model (Equation 3.2) simulates a fixed number of repairing units limited by the repair capacity g. All the repairing units are available to fix broken nodes, but as more and more nodes fail, $x \to 1$ and the function $\frac{g}{1+Nx}$ saturates on $\frac{g}{N}$ and approximately g nodes can be repaired on each step. In a real-world system, this could be the case if there are a number of available repair crews on stand-by, ready to step in as components fail, but left to play catch-up if the system crash is too big. The repair chance is equal for all nodes—any failed node has exactly the same chance of being repaired. The failure rate does not share this property. Only nodes that are in the **frontier** of the failure can fail, and any node with all neighbours failed *will* fail on the next time step. And while this is not true when time-steps is divided into dt, it is still true on average.

5.2 The “Jump” of the First Time-Step - A Weakness of the Simulations

Many of the simulations show a failure bump on the first time step, as can be seen in Figure 4.2. Figure 5.3 also shows that there is something about the first time step that makes more nodes fail. This is a weakness of the simulations, and it is likely caused by the fact that the nodes that are failed initially are random, while the failure-mechanism is not random. This means that initially there will be more nodes that are particularly vulnerable than if the simulation had reached the same fraction of failed nodes x naturally. This hypothesis is strengthened by the fact that this jump is much bigger on the simulations on the scale-free networks (see Figure 4.17) where a bigger fraction of the nodes have a low degree and are extra vulnerable.

When the fraction of initial failed nodes x is set for the simulation. The first $x \cdot N$ nodes in the network of N nodes are selected. The way random and scale-free networks are generated, this results in a random selection of nodes in the networks. For grids however, the first $x \cdot N$ nodes in the network are all neighbours. This means that the failure **frontier** will be unnaturally small, and fewer than predicted nodes will fail, resulting in a repairing bump instead as seen in Figure 4.18.

This “jump” on the first time-step is a weakness of the simulations, and though somewhat reduced when dt is decreased, it can still be seen for example in Figure 4.4 where $dt = 0.1$.

5.3 The Difference *dt* Makes

The simulations in this thesis are done in a step-wise fashion, meaning that first all nodes eligible for failure are tested, and some fraction of them fail, then all nodes eligible for repair are tested, and some fraction of them are repaired. After this the value of x is recorded, and the simulation goes back to step one. The fluctuations caused by this step-wise process is what the smaller dt are meant to counter-act.

Figure 5.1 shows the effect smaller dt have on the repair rate (Equation 3.2) in green and the failure rate (Equation 3.1) in red. What is found is that the repair is not significantly affected by the value of dt. If $\tilde{g} = 0.1$ one tenth of the network will be repaired each full time-step, and the network will be completely repaired in ten time-steps. The failure rate, however, *is* affected by the introduction of dt. As dt is decreased, the failure rate steepens for middle range values of x. The number of time-steps until complete failure is not changed significantly, because the failure rate flattens out once x approaches 1.

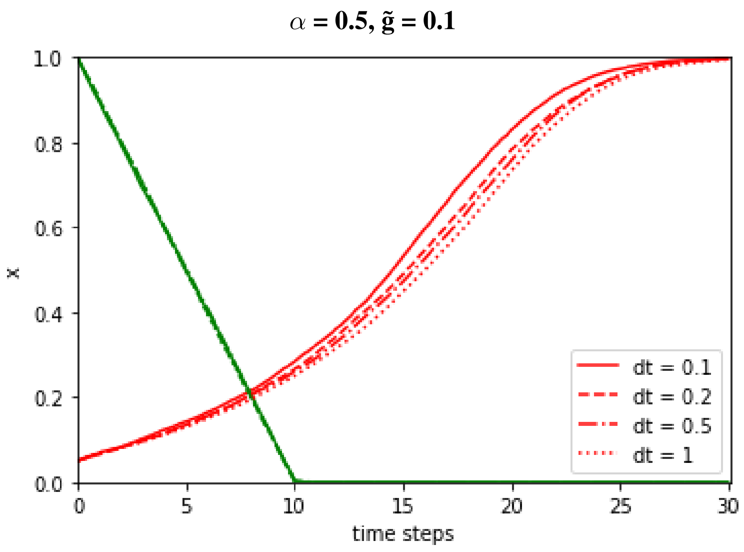


Figure 5.1: Simulations with only the fail term where $\alpha = 0.5$, and only the repair term with $\tilde{g} = 0.1$ were run with different dt.

The effect changing dt has on the failure rate can be explained by the fact that the only nodes that can fail at any time are the ones in the **frontier** of the failure (neighbours to failed nodes that have not failed yet themselves). Making dt smaller, the frontier is updated more frequently, and when looking at only the failure term, this update means an increase in the number of nodes vulnerable to failure. If the time-steps are divided, more nodes will have had the chance to “throw the dice” before the end of the time-step. This does not happen with the repair rate. The number of nodes eligible for repair may change more often as dt is decreased, but the number of nodes that *will* be repaired does not depend on how many nodes are eligible, it is constant for all x-values (except when fewer than g nodes are failed).

5.4 The Tendency to “Jump” Down

Figures 4.2 and 4.4 indicated that the x-value for the unstable fixed point is decreased as dt is decreased. This tendency to “jump” down to the “good” stable fixed point for higher values of dt was also seen when $\tilde{g}$ was close to the bifurcation point in Figures 4.15 and 4.16. This “jump” is never seen the other way (with an unexpected leap to the “bad” stable fixed point), indicating that it is caused by some property of the system, and are not just random fluctuations in the simulations.

Figure 5.2 show the first four time-steps of simulations with the same initial conditions started at $x = 0.45$ and with $dt = 1$ and $dt = 0.1$. The simulations with $dt = 1$ are headed down to the “good” stable fixed point, while the simulations with $dt = 0.1$ are headed up

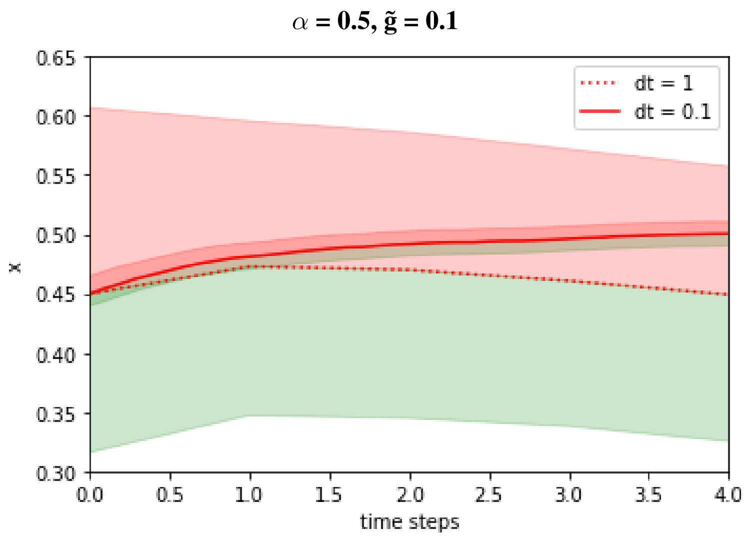


Figure 5.2: Simulations with $dt = 1$, and $dt = 0.1$). Zoom in on first few time steps. Red shade is number of nodes failed on each dt. Green shade is number of nodes repaired on each dt.

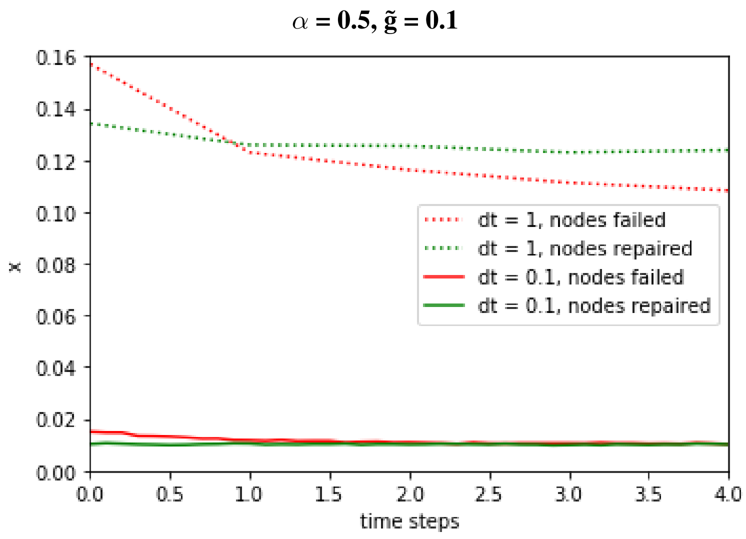


Figure 5.3: Fraction of nodes failed and repaired on each dt for different dt.

to the "bad" stable fixed point. The red shaded areas show the number of nodes that fail on each time-step and the green shaded area shows repair. Figure 5.3 shows the number of nodes failed and repaired on each time step. After one elapsed time-step the number of repaired nodes exceeds the number of failed nodes for $dt = 1$, but not for $dt = 0.1$, even though simulations were started with the same x, α and $\tilde{g}$.

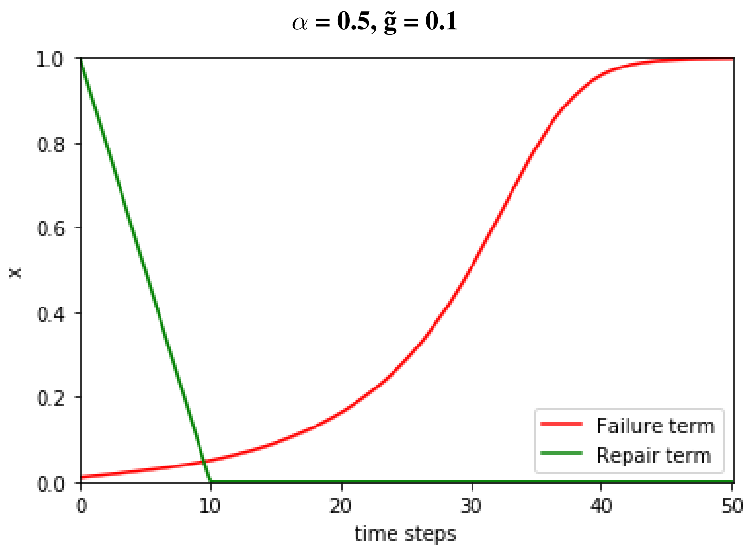


Figure 5.4: Simulations with only the fail term where $\alpha = 0.5$, and only the repair term with $\tilde{g} = 0.1$ were run with $dt = 0.1$.

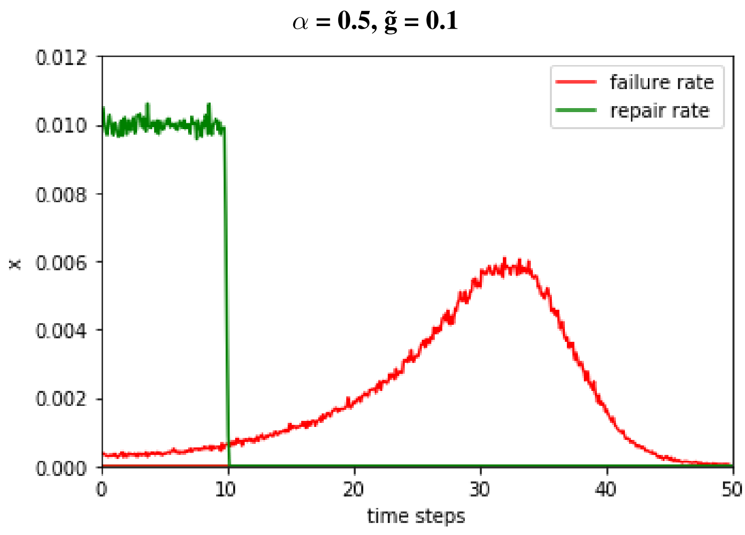


Figure 5.5: Simulations with only the fail term where $\alpha = 0.5$, and only the repair term with $\tilde{g} = 0.1$ were run with $dt = 0.1$. Fraction of nodes failed and repaired on each dt for different dt.

The tendency to “jump” down is likely caused by the fact that failure depends on the fraction of failed nodes, while repair does not. Figure 5.5 show that the number of nodes repaired on each time steps peaks around time-step 30. At time step 30 in Figure 5.4 the value of x is approximately 0.5—half the network has failed. This means that at x-values lower than this, the number of nodes repaired on each time-step or dt decreases as x decreases.

So if x is below 0.5 and at one time-step or dt there happen to be repaired a few more nodes than the expected average, it will reduce the x by a little more than the analysis predicts. Then the next time failure is calculated, *fewer* nodes will fail as a result, creating a reinforcing effect. Bigger dt will increase the chance of this happening, as a "jump" produced by the repairing of nodes at each dt can be so big that it jolts the system out of its path. Smaller dt will be less likely to have big enough jumps.

This tendency to "jump" down, is also the reason why in Figure 4.15 where $dt = 0.1$, the "bad" stable fixed point disappeared at a lower value of $\tilde{g}$ than predicted. Then as dt was further reduced to 0.01, the fixed point reappeared in Figure 4.16.

5.5 Changing the Average Degree

When comparing results of simulations on networks of different average degree, $\langle k \rangle$, there is one important thing to note. The way random networks are generated in the implementation done for this thesis, only the giant component is kept, meaning that some nodes are deleted (see algorithm 1 in section 3.5). Since almost all simulations have been done on networks of the same average degree, namely 4, this has not been a problem, as the same amount of nodes were removed from each network. However, the *connectedness* of a random network changes with degree, and networks of average degrees higher than $\ln N$ will be fully connected. Using the code in algorithm 1 would result in networks of different sizes. This is a weakness in the simulations, and because of this, the comparison between networks of different average degree is done with average degree 8 and 12.

Increasing the average degree, of the network will decrease average travel distances, making it theoretically possible for the failure to spread more quickly, but at the same time the presence of more edges are a protective factor to each *individual* node. As with changing dt, the changing of average degree does not affect the repair rate.

Figure 5.6 shows simulations with only the failure term on 10 000 node random networks with $\langle k \rangle = 8$ and $\langle k \rangle = 12$. The failure rate starts out slower when $\langle k \rangle = 12$, but as x grows, the rate becomes steeper and starts failing faster than the $\langle k \rangle = 8$ networks. The time until total failure is not affected much by an increasing average degree. The effect may be seen more clearly in Figure 5.7, where the fraction of failed nodes per time-step is plotted. At around time-step 15 the failure rates are the same. This is close to the x-value of the predicted unstable fixed point. Which means that the networks of higher average degree fail more quickly above the unstable fixed point, and more slowly below. So the simulations on $\langle k \rangle = 12$ should reach their stable fixed points faster, this is what Figure 4.12 from the results chapter shows (4.3).

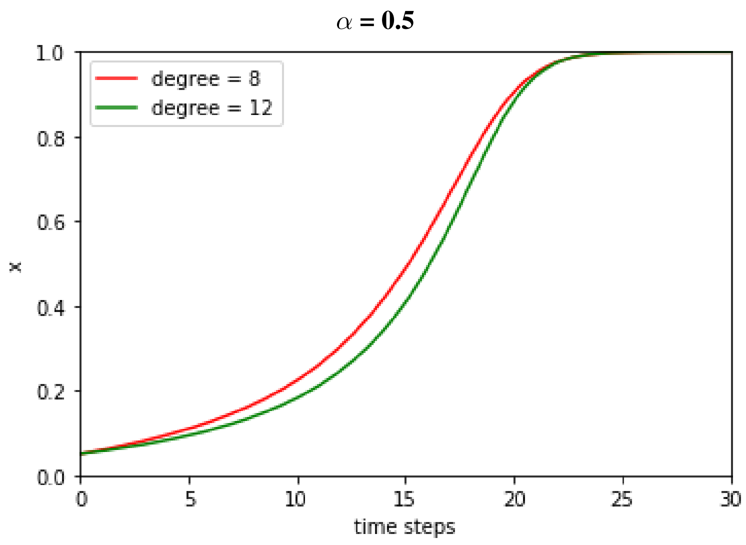


Figure 5.6: Simulations with only the fail term where average degree of 8 and 12 is compared.

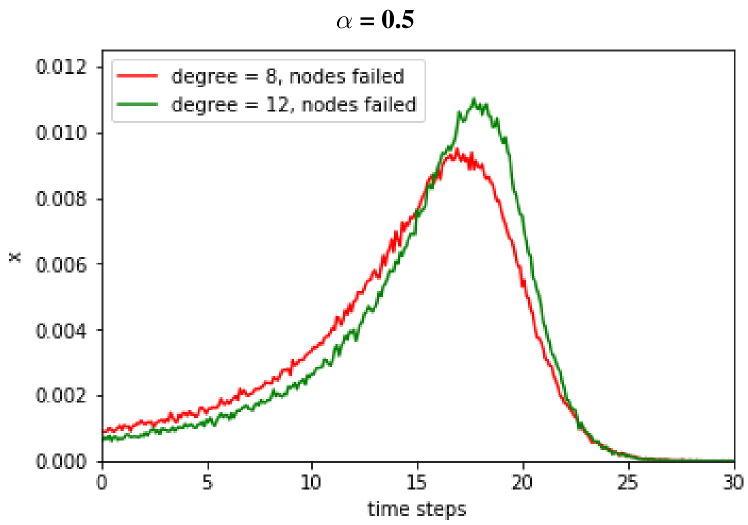


Figure 5.7: Number of nodes failed on each dt for simulations with only the fail term where average degree of 8 and 12 is compared.

5.6 Consequences of the Mean Field Approximation

Using mean field approximations, in effect it is assumed that all nodes are the same, and that all nodes are the average. In terms of the degrees of nodes these things seem like reasonable approximations for random networks and grids with their rather homogeneous degree distributions. Less so for scale-free network, where not only are the nodes not all the same degree, almost none of them are average. However, when all nodes are set to the system average x-value, it is also assumed that all nodes have the same chance to be neighbour to each failed node. This is *not* true for grids where the nodes are very predictably interconnected. Two neighbours of a node are much more likely to also be each others neighbours, than two randomly selected nodes in the network. This tendency is not found in random networks and scale-free networks, making this assumption better for these networks.

The average degree of a network also matters for the mean field approximation. Relative fluctuations around the average will be bigger when the average degree of the network is smaller, so there will be relatively bigger fluctuations away from the average. This makes networks of higher average degree fit better with the assumption that every node is the average. Figure 4.12 shows that networks of higher degree reach the fixed points faster.

5.7 Other Networks

The simulations on grids in Figure 4.18 show that the simulations tend to go to the “good” stable fixed point even when started high above the predicted unstable fixed point. In fact, for this value of $\tilde{g}$, it is not absolutely certain that there is a “bad” stable fixed point. These results are in keeping with studies that have found mesh-like networks to have an ideal architecture for surviving deliberate attacks (Baran, 1964). No single node failure has a devastating effect on the connectedness of the network, as the failure of one node can maximally affect four other nodes. That grids take a long time to fail, can also be explained by the fact that the diameter of a grid is much larger than that of random and scale-free networks.

Figure 4.17 show the results from simulations on scale-free networks. The results are quite similar to those seen with random networks, but the threshold x-value is higher on the scale free networks than on random networks. That scale-free networks are more vulnurable than random networks when failure dynamics are considered is well established in the literature (Motter and Lai, 2002; Crucitti et al., 2004; Strogatz, 2001; Schläpfer and Shapiro, 2009).

6 | Conclusion

Our model for cascading failure show simulation results that are in keeping with previous studies on the subject. Scale-free networks are more vulnerable than random networks, and regular grids are robust to cascading failure.

Results from the stochastic simulations show that the phase space from the analysis can predict the presence of stable and unstable fixed points for given values of α and $\tilde{g}$.

Plotting $h_1(x)$ and $h_2(x)$ from equation 3.3 can also be used to find the approximate x-value of these fixed points. Stochastic simulations show a tendency to “jump” down to the “good” stable fixed point at lower x-values than predicted. Dividing time steps into smaller dt lessens this tendency and makes the simulations approach the predicted values for both the stable and unstable fixed points.

Networks of high average degree seem to reach their steady state slightly faster than lower degree networks. They have lower failure rate below the unstable fixed point and higher rate above it.

Bibliography

Albert, R., Jeong, H., Barabási, A.-L., 2000. Error and attack tolerance of complex networks. Nature 406 (July), 378–382.

Balcan, D., Hu, H., Goncalves, B., Bajardi, P., Poletto, C., Ramasco, J. J., Paolotti, D., Perra, N., Tizzoni, M., Van den Broeck, W., Colizza, V., Vespignani, A., 12 2009. Seasonal transmission potential and activity peaks of the new influenza A(H1N1): a Monte Carlo likelihood analysis based on human mobility. BMC Medicine 7 (1), 45.

Barabási, A.-L., 2016. Network Science. Cambridge University Press, Cambridge. URL `http://networksciencebook.com/`

Barabási, A.-L., Albert, R., 1999. Emergence of scaling in random networks. Science 286 (5439), 509–512.

Baran, P., 1964. Introduction to Distributed Communications Networks. RAND Corporation, Santa Monica, CA.

Crucitti, P., Latora, V., Marchiori, M., 2004. Model for cascading failures in complex networks. Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics 69 (4), 4.

Erdos, P., Renyi, A., 1959. On random graphs I. Publicationes Mathematicae (Debrecen) 6, 290–297.

Hagberg, A. A., Schult, D. A., Swart, P. J., 2008. Exploring network structure, dynamics, and function using NetworkX. In: Varoquaux, G., Vaught, T., Millman, J. (Eds.), Proceedings of the 7th Python in Science Conference. Pasadena, CA USA, pp. 11–15.

Hufnagel, L., Brockman, D., Geisel, T., 2004. Forecast and control of epidemics in a globalized world. Proceedings of the National Academy of Sciences 101 (42), 15124–15129.

Hunter, J. D., 2007. Matplotlib: A 2D graphics environment. Computing In Science & Engineering 9 (3), 90–95.

Jensen, P. H., 2018. Modelling Cascading Failure in Complex Networks. Project Thesis, Department of Mechanical and Industrial Engineering, NTNU.

Kosterev, D. N., Taylor, C. W., Fellow, W., 1999. Model validation for the August 10, 1996 WSCC system outage.pdf. IEEE Transactions on Power Systems 14 (3), 967–979.

Milgram, S., 1967. The Small World Problem. Psychology Today 2, 60–67.

Motter, A. E., Lai, Y. C., 2002. Cascade-based attacks on complex networks. Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics 66 (6), 4.

Reches, E., 2019. Generalized SI Model: Including Sensitivity. Private communication.

Schläpfer, M., Shapiro, J. L., 2009. Modeling failure propagation in large-scale engineering networks. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering 5 LNICST (PART 2), 2127–2138.

Strogatz, S. H., 2001. Exploring complex networks. Nature 410, 268–276.

Strogatz, S. H., 2015. Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering. Westview Press.