ECOLE POLYTECHNIQUE
PROMOTION 2009
CASSIER Lionel

RAPPORT DE STAGE DE RECHERCHE

Systemic risk and indicators for the stability of a financial network

NON CONFIDENTIEL

Option : Mathématiques Appliquées
Champ de l'option : Finance
Directeurs de l'option : Dr E. Gobet, Dr N. Touzi
Directeur de stage : Dr T. R. Hurd, Dr M. Grasselli
Dates du stage : 16/04/2012 - 20/07/2012
Nom et adresse de l'organisme :
McMaster University
Department of Mathematics and statistics
Hamilton Hall
Hamilton, ON L8S 4K1
And the Field Institute, Toronto

Résumé

Pendant ce stage, nous avons cherché à étudier les crises dues au risque systémique. Nos économies sont de plus en plus liées de nos jours et ce qui affecte une banque aujourd'hui a de fortes chances d'avoir des répercussions sur les autres banques du pays mais aussi sur celles du monde entier. Ainsi, si une banque subit de lourdes pertes, il est possible qu'elle cause la chute des banques avec qui elle échangeait des produits. Ces dernières, si elles tombent, peuvent à leur tour causer la chute de leurs voisines, et ainsi de suite. Un effet de cascade peut donc se produire et être à l'origine d'une crise majeure. Tout le monde se souvient que la chute de Lehman brothers avait touché le monde entier (et pas simplement quelques grosses banques Américaines) et que beaucoup de banques avaient été sauvées grâce à des renflouements par les différents États. Une crise mondiale peut donc simplement être causée par la transmission d'un simple choc à travers un réseau.

Nous avons donc essayé d'étudier pendant ce stage comment un réseau bancaire pourrait réagir à la chute de l'une ou de plusieurs banques. Nous avons premièrement adapté des modèles, que nous avons trouvés dans la littérature, à des réseaux qui nous semblaient plus réalistes. La plupart de ces modèles s'intéressaient en effet à des réseaux infinis et très spécifiques. Nous avons dû par la suite essayer de comprendre et de venir a bout des difficultés que nous rencontrions suite à la perte de cette spécificité dans nos réseaux. Enfin nous avons pu mettre en évidence deux indicateurs présentant des propriétés complémentaires qui nous permettent de savoir si un réseau financier est stable, c'est à dire si nous risquons une crise majeure ou non.

Summary

During this summer, we have been studying systemic crises in a financial graph. Our economy has been very linked lately and it is impossible to think of a bank as a business relatively independent of other banks. On the contrary, we can really think of the banking business as a network. They are indeed exchanging many assets every day, lend money and borrow money from each other. Therefore, if one bank fails, many other might be affected and in some cases go to bankruptcy as well. One knows how the default of Lehman brothers affected the entire world and one knows that the economy was saved thanks to the intervention of the governments through bailouts.

Thus, during this summer we decided to study how a banking network would react to the default of several banks. We have first adapted the existing models to more realistic banking networks, then we tried to understand and overcome the difficulties we met considering finite and general graphs instead of ideal situations. Finally we have derived two indicators to see whether a graph is stable or not, in other words, whether we risk a huge crisis or not.

Contents

Introduction

I have done my research internship in both the financial laboratory of McMaster university in Hamilton, Canada and the Fields Institute in Toronto. I have worked mainly with Pr. Hurd, who is the director of the math finance program at McMaster university and Pr. Grasselli who is the deputy director of the Fields Institute and a professor at McMaster university as well. They are both part of the PhiMac group which is composed of professors, postdoctoral and graduate students working on various aspects of financial mathematics including for instance credit risk, portfolio risk management, and what I have been mostly dealing with : systemic risk. In this report, I am going to focus on the work I did on systemic risk, even though it is not the only thing I have been working on here. I have also dealt with a different approach on the study of crises under the supervision of Pr Grasselli. We used an empirical method, free of any macro economical model, to find indicators for the different types of financial crises. This was a very enlightening experience, however to keep a unity in this report I will only focus on my work on systemic risk, which accounts for around two third of the time I spent here. Let me now introduce the general framework of what I have been doing during these three months.

Systemic risk can be given a lot of definition but the way we looked at it is how something can spread into the entire financial network and cause trouble to the global economy. We used to think of crisis as something very unlikely and very strong that could happen and cause the default of many financial institutions, however, systemic crises in a systemically important network may rather be like very little shocks that can occur quite often and that are likely to have very bad consequences on the whole network. We will therefore try to model how a financial network is going to be affected by the default of one or several banks. Is default going to spread over the whole network or will only several banks be touched ? This is the sort of questions we are going to try to answer in this paper.

To achieve this goal, I will in the first chapter introduce the models that have inspired us during this internship. Pr Hurd has been working with these models for a few years but we decided at the beginning of this internship to have a different approach of what had been done by himself, Pr Gleeson, and the authors of the models we used. We wanted our models to be able to address more realistic networks and therefore we changed our line of attack.

In chapter two, I will introduce the mathematical models that Pr Hurd and I have derived adapting the model we introduced in the first part to fixed graph. We will therefore be able, given some data and a strong hypothesis, to derive the set of banks that can be badly touched by default if an initial shock occurs somewhere in the graph. We will also adapt to our new framework a condition for a large default cascade, that Pr Hurd and Pr Gleeson derived.

Given that we have done a very strong assumption in chapter two to be able to derive the math formulas for our new model, we will implement all this, and run numerical simulations to understand when our results are correct or not. We will therefore understand a lot more how our model works, and we will be able to set limits to our approach. Nevertheless, we will reach our goal and we will be able to give a good indicator to find out whether a graph is systemically important or not. Some might think that we don't have to imagine sophisticated models to answer this kind of question. It is obvious that we could quite simply run Monte Carlo simulations, but actually with the models we are considering in this paper, if we look at medium size graphs with 300 or 400 nodes, Monte Carlo simulations will take a long time to run. Our mathematical results enable us to derive good indicators for systemically important graphs in a reasonable amount of time.

Finally, I would like to stress that I have written this report in a very simple way so that anybody with a basic mathematical background and knowledge of probability theory can understand it. Our work doesn't require very sophisticated mathematical tools but it uses, I believe, beautiful mathematical arguments, and I really enjoyed working on it with my supervisors.

Chapter 1

Description of the problem

1.1 A financial network description

In all this paper, we are going to focus on a simplified model of bank interactions in the real world. We will thus consider a network whose nodes will be banks and whose links will be the transactions between two banks. Every bank will also have a buffer, which represents the amount of cash it can use in case it suffers a loss.

Thus a financial network will be defined this way :

Definition : A financial network (A,Γ,W) on a set of N vertices V is given by :

- the adjacency matrix of the graph $(A_{(i,j)})_{1\leq i\leq N, 1\leq j\leq N}$
- the weights of the links $(W_{(i,j)})_{1\leq i\leq N, 1\leq j\leq N}$
- a buffer distribution $\Gamma_{1\leq i\leq N}$. This represents the amount of cash a bank has to cover losses and pay for its liabilities.

Thus, a node i owes a node j an amount of money a if and only if $W_{i,j} = a$.

Our goal is to understand how this graph is going to react to an initial perturbation. We want to know for instance whether a large number of the bank considered in this network are also going to default if an initial bank goes into bankruptcy.

We are therefore going to consider the following sets in all our mathematical study :

- N_i^- the in-neighbors of a node i, in other words, the nodes that owe money to i. Also, N_i^+ will be the creditors of a node i and N_i all the neighbors of i.
- D_0 the set of initially defaulted banks
- D_n the set of defaulted banks after n steps that were not in the set of initially defaulted banks.

Our aim is to study the set D_∞, i.e. all the banks that are eventually going to default.

Before giving all the results we obtained during this summer, let us first look at some interesting models developed in the last few years. They are known as the Watts model, the Gai and Kapadia model (GK) and the Gai, Haldane and Kapadia model (GHK). These models are the cornerstone of our mathematical results, and we will therefore have to recall some of the main results of these papers, at least those that are going to be useful to our study.

1.2 The different systemic models considered in this paper

All these three models study the propagation of a property through a network. Our aim is eventually to deal with financial crises, but we can look at financial crises, in a more simplified way, as a disease spreading over all the financial network. If an initial bank collapses, then it is going to be unable to pay back its creditors, and so these creditors will face a loss themselves. If they don't have enough money to overcome this situation, they will go into bankruptcy and the same problem is going to affect their neighbors. This might then spread over all the network. Therefore it makes sense to look at epidemic or friendship models such as the Watts model.

1.2.1 The Watts model

The Watts model studies the origin of global cascades in a social network. The goal of this paper is to derive a condition to the existence of a global cascade, even though it was triggered by a small initial condition. Duncan J. Watts looks thus at random non-directed graphs, that we could think of as a friendship network.

Every node is therefore linked to its neighbors or friends and he studies the propagation of a property through the network. The property is said to be adopted by a node if enough of its neighbors have already adopted it. The contagion is therefore based on a threshold property. Watts shows that we can actually derive the existence of a global cascade by looking at the size of the largest vulnerable cluster (nodes whose threshold is small enough so that one of its neighbors alone can have him adopt the property). He actually brings back this problem of contagion to a mere problem of percolation.

Hurd and Gleeson, inspired by this model decided to adapt the Watts model to our finance problem. They decided thus to consider a random financial non directed graph, whose links, buffers and weights are random. However instead of saying that a node defaults when a big enough number of its neighbors have, we consider that a node v defaults after n steps when he satisfies the following insolvency condition (note that for an non-directed graph, two banks that are linked have the same loans to each other):

$$\sum_{v' \in N(v)} W_{v'v} \mathbb{1}(v' \in D_{n-1} \cup D_0) \geq \Gamma_v \tag{1.1}$$

This condition is self explanatory, it means that the amount of money owed by the defaulted in-neighbors of i (i.e. the money that i can't be given back) is larger than the amount of cash i has. These neighbors are indeed not able to pay back the money they borrowed and i suffers a loss larger than its buffer and defaults.

This model is of course very simple because every bank has a loan involving the same amount of money as its neighbors have borrowed from them. This is of course not what happens in the real world. This led thus Hurd and Gleeson to study other models like the GK model.

1.2.2 The GK model

The GK model is very close the Watts model we have just described. The main difference with the Watts model is that we now ask the links to be directed. Therefore either a bank borrows from another or it lends money, but never both. In this model, the solvency condition remains the same as in the Watts model, except that we only consider the in-neighbors.
Thus a node has defaulted after n steps if and only if :

$$\sum_{v' \in N^-(v)} W_{v'v} \mathbb{1}(v' \in D_{n-1} \cup D_0) \geq \Gamma_v \tag{1.2}$$

This means that the default cascade is only going downstream, one defaulted out-neighbor will therefore never immediately affect a node. However, we know that things don't always happen this way. We all know that many other parameters can favor new crises. Stress, for instance, played a huge role in the recent crises. This is what the GHK model tries to take into account.

1.2.3 The GHK model

In their paper on the determinants of Banking Crises in developing and developed countries, Demirguç-Kunt and Detragiache explain that a banking crises can occur because banks have short term liabilities (inter banks loans) and long term or middle term assets to business and consumers. Thus, when their assets fall, creditors may become unwilling to extend their daily credits. A bank may therefore be unable to borrow money and become insolvent. Dropping assets can also put banks in a stress condition. If they are afraid that they might fall, banks can also have an attitude that is contrary to the well functioning of the market : liquidity hoarding. They might want to find as much cash as they can to prevent their balance sheet from going below zero. If they do so, banks will prevent the market from functioning. A bank doesn't only need to default to slow our economy, stress alone can make the market highly inefficient. This is the kind of macro-economical reasoning that led the two authors of the GK model along with Haldane to create a new model which would take stress into account.

In this new model, a bank can be either normal, stressed or defaulted. We will therefore have some new notations :

- A stress buffer $B_{1\leq i\leq n}$
- M_0 the set of initially stressed banks
- M_n the set of stressed banks after n steps that were not initially stressed.

We have also these conditions to describe the state of a bank after n steps :

- Like in the GK model, a bank has defaulted if and only if :

$$\sum_{v'\in N^-(v)} W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0) \geq \Gamma_v \tag{1.3}$$

- We will also have the following stress condition for a bank v :

$$\begin{cases} \displaystyle\sum_{v'\in N^+(v)} W_{vv'}\mathbb{1}(v' \in M_{n-1} \cup M_0) \geq B_v \\ \text{or} \\ v \in D_{n-1} \end{cases} \tag{1.4}$$

Thus a bank is stressed if it has defaulted or if enough of its out neighbors are stressed. This stress condition makes the GHK model very different from the GK model because we are now looking at two different properties that can strongly affect the financial network. We now have two different possible types of cascade. Besides, the Watts model as well as the GK model, has only downstream cascades. Only its in-neighbors can affect one node, in the GHK model, the default cascade is obviously going to propagate downstream but the stress cascade is going upstream, and even downstream in some way since default causes stress.

It is also important to mention that in all these models we consider random buffers, random weights and random links. It might be strange to put as much randomness in these models but we have to realize that unfortunately we might never be able to have precise information about the amount of money involved in a transaction between two banks. It is not even sure that someone clearly knows these figures, even inside the bank, and if by chance someone does, these data might be very confidential. Thus putting random numbers enables us to find out a lot of systemic information without making too unrealistic assumption. We just want our figures to be in the right order of magnitude.

Now that we have introduced the models we've used during all this internship, let us introduce the mathematical results we've obtained.

Chapter 2

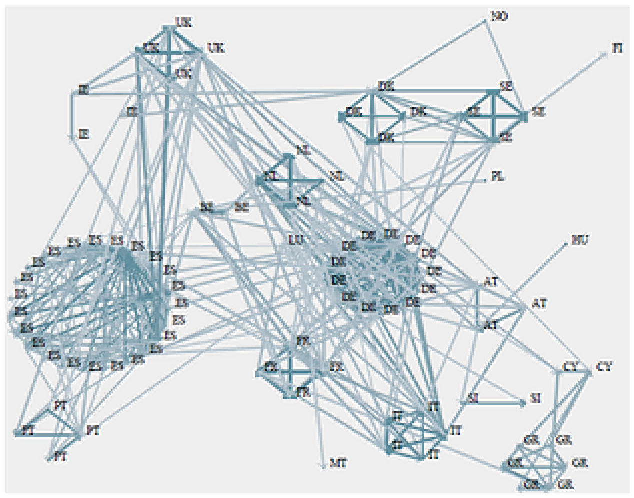
Mathematical results

In all the paper we worked on (Watts, Gai and Kapadia, Gleeson and Hurd etc.), the researchers always chose to work on a random graph. This means that not only were their weights random, but that their links were also random. They were working most of the time on infinite random graphs. This is an assumption that is actually helpful because, as we are going to understand below, infinite random graphs have some strong properties that simplify a lot what happens in a finite network.

2.1 A new model and new possibilities

At the beginning of the summer, Pr. Hurd and I tried to get a more precise picture of a real financial network. Infinite random graphs look actually very different from what we thought a real financial network would look like. For instance, random Poisson infinite graphs have no loops. This very fact sounded very unlikely for a real financial graph. This is hard indeed to imagine that Goldmann Sachs is not in business with Barclays and JP Morgans and that these two banks don't make any business together. We therefore felt that we needed to change our approach.

We also contacted many people to ask them if they had more precise data about the weights of the links and the look of a real financial graph. We used the Polytechnique network as well as Dr Hurd's contacts, and we contacted many regulatory agencies, the World Bank and the ECB. We didn't get much information about the weights of the links especially because this information is confidential but we got a picture of the European financial network that one can see below.



;

Figure 2.1: *European inter-bank relations*
ECB Financial Stability Review

We can see that this graph doesn't really look like an infinite Poisson random graph. The graph we can see is very highly clustered and contains many loops.

Thus, our idea was to give up the infinite graph hypothesis and to work on fixed graph. So, during this summer we decided to work only with random buffers, and random weights, but we decided that the links should be fixed in our graphs. One can think that it is only a special case of what was studied before, but it is not, and the math become different because in the infinite random network, every node plays the same role, whereas in our model a node has a fixed number of neighbors that is for instance different from that of another node. Thus, every node has a different systemic importance, and this changes the way we have to look at the problem.

However this gives our approach a lot of strength since we can work on every graph, and, if we get a more precise picture of a financial network, we can then choose a graph much closer to reality than a random Poisson infinite graph.
Let us now introduce our main mathematical results.

2.2 Derivation of the propagation in the different models

We are now going to derive the nodes that have defaulted after n steps and that were not initially defaulted, i.e. the set D_n in every model we have described before. Let us begin with the Watts model.

2.2.1 The Watts model

Let's first begin with a naive approach that is going to make us really understand the need of another argument that is a beautiful idea and that is the corner stone of the proofs of all our mathematical results.

Naive Approach

We can feel, thanks to the definition of D_n, that the proof is going to be based on some inductive properties. Let v be a vertex in our graph.

$$\begin{aligned}
&\mathbb{P}(v \in D_n) = \mathbb{P}(v \in D_n | v \notin D_0)\mathbb{P}(v \notin D_0) \\
&\mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\mathbb{P}(\sum_{v' \in N(v)} W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0) \geq \Gamma_v | v \notin D_0) \\
&\mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\mathbb{P}(\sum_{v' \in N(v)} W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0) \geq \Gamma_v)
\end{aligned} \tag{2.1}$$

There are two remarks we have to do. First, we have the condition on v not being an initially defaulted bank, because even if v satisfies the insolvency condition, if it is initially defaulted, it is not in D_n. Also to write the last line, we have to suppose that the initially defaulted banks are independent of the weights, the geometry of the graph etc. Actually this is not a demanding assumption, and let us suppose it is true.

Now we need to derive $\mathbb{P}(\sum_{v' \in N(v)} W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0) \geq \Gamma_v)$.
Let $\tilde{W}^{n-1}_{v'v}$ be equal to $W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0)$.

It seems easy to show that $\tilde{W}^{n-1}_{v'v}$ has the same law as $\mathbb{P}(v' \in D_{n-1} \cup D_0)W_{v'v}$ $+ (1 - \mathbb{P}(v' \in D_{n-1} \cup D_0))\delta_0$ where δ_0 is a Dirac mass at the point zero. Then we see that our formula is recursive and that we could easily derive D_n.

Actually, we did something wrong in this approach. We are trying to derive the probability that a node defaults, and as the model suggests, we look at its defaulted neighbors at the former step and then we try to find the probability that it satisfies the insolvency condition. However we don't take into account that these neighbors must have defaulted before v defaults and that v cannot contribute to their default. In this approach we state nowhere that these defaulted neighbors must have gone into bankruptcy without regard to v.

The WORT condition

In order to make this idea more formal, we are going to derive the probability that a node defaults without regard to another one, in other words, just as if the latter didn't exist at all. Thus, we won't make any mistake when we write the default condition, as we did just above.

Our results

Let us now derive the probability that a node has defaulted after n steps without regard to one of its neighbors. We are going to see that we have this time an inductive sequence and that we are going to be able to complete the proof.

Let v and v' be two nodes that are neighbors, and let us still assume that the initial defaulted banks set is independent of everything else concerning the graph.

(2.2)
$$\mathbb{P}(v \in D_n \text{ WORT v' }) = \mathbb{P}(v \in D_n \text{ WORT v' } | v \notin D_0)\mathbb{P}(v \notin D_0 \text{ WORT v' })$$
$$\mathbb{P}(v \in D_n \text{ WORT v' }) = \mathbb{P}(v \notin D_0)\mathbb{P}(\sum_{v'' \in N(v), v'' \neq v'} W_{v''v}\mathbb{1}(v'' \in D_{n-1} \cup D_0 \text{ WORT v,v' }) \geq \Gamma_v | v \notin D_0)$$

And since D_0 is independent of everything else concerning the graph, since the "WORT v" makes "D_n WORT v" independent of v being an initially defaulted bank or not, we have :

$$\begin{aligned}&\mathbb{P}(v \in D_n \text{ WORT v' }) = \mathbb{P}(v \notin D_0)\\ &\mathbb{P}(\sum_{v'' \in N(v), v'' \neq v'} W_{v''v}\mathbb{1}(v'' \in D_{n-1} \cup D_0 \text{ WORT v,v' }) \geq \Gamma_v)\end{aligned} \tag{2.3}$$

In order to have an inductive sequence, we have to make another assumption that we will formalize later in the tree-like assumption section. We want indeed v' to disappear from $\mathbb{1}(v' \in D_{n-1} \cup D_0 \text{ WORT v,v' }) > \Gamma_v)$ to have on the left hand side of the equation the probability that a node has defaulted after n steps without regard to one node and on the right hand side something involving the probability that another node has defaulted after n-1 steps without regard to another single node.

So, in order to write :

$$\begin{aligned}&\mathbb{P}(\sum_{v' \in N(v), v'' \neq v'} W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0 \text{ WORT v,v' }) \geq \Gamma_v)\\ &= \mathbb{P}(\sum_{v' \in N(v), v'' \neq v'} W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0 \text{ WORT v }) \geq \Gamma_v)\end{aligned} \tag{2.4}$$

We have to assume that the node v' is not linked to any of the neighbors of v. Otherwise, v' would have an influence on a neighbor v" of v that we consider just above, and the last equation would be wrong.

Given that assumption, we can now properly derive the mapping that takes us from rank n-1 to rank n.

Let $\tilde{W}^{n-1}_{v'v}$ be again equal to $W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0$ WORT v) :

We can also show easily that :

$$\begin{aligned} &\tilde{W}^{n-1}_{v''v} \sim [\mathbb{P}(v'' \in D_0) + \mathbb{P}(v'' \in D_{n-1} \text{ WORT v})]W_{v''v} \\ &+ [1 - \mathbb{P}(v'' \in D_0) + \mathbb{P}(v'' \in D_{n-1} \text{ WORT v})]\delta_0. \end{aligned} \tag{2.5}$$

Let us now use these last equations to derive the probability of default after n steps. We have :

$$\mathbb{P}(v \in D_n \text{ WORT v' }) = \mathbb{P}(v \notin D_0)\mathbb{P}(\textstyle\sum_{v' \in N(v), v'' \neq v'} \tilde{W}^{n-1}_{v'v}) \geq \Gamma_v) \tag{2.6}$$

Before we go any further, we have to do a few remarks this equation. First, it is easy to see that if u and v are two random integer variables, we have :

$$\begin{aligned} \mathbb{P}(u \geq v) &= \sum_{1 \leq i \leq \infty} \mathbb{P}(u = i)CDF(v)(i) \\ \text{P(u} \geq v) &= \langle pdf(u), CDF(v)\rangle \end{aligned} \tag{2.7}$$

Where the brackets mean the scalar product on $\mathbb{N}$. This is going to help us derive the last term of the right hand-side of our equation.

It is also easy to show that the sum of n independent random variables has the law of the convolution product of their pdf. Thus we have :

$$\boxed{\begin{aligned} &\mathbb{P}(v \in D_n \text{ WORT v' }) = \mathbb{P}(v \notin D_0)\langle \tilde{W}^{n-1}_{v_1v} \star ... \\ &\star \tilde{W}^{n-1}_{v_iv} \star .... \star \tilde{W}^{n-1}_{v_{k_v}v}, CDF(\Gamma_v)\rangle \\ &v_i \text{ neighbor of v, } v_i \neq v' \end{aligned} \tag{2.8}}$$

We have just found the expression of $\mathbb{P}(v \in D_n$ WORT v'), it is an inductive sequence since we can derive the law of $\tilde{W}^{n-1}_{v_iv}$ thanks to $\mathbb{P}(v_i \in D_n$ WORT v).

Of course, what is interesting for us is $\mathbb{P}(v \in D_n)$ and not $\mathbb{P}(v \in D_n$ WORT v'), but it is actually easy to derive the first probability with the second one :

(2.9)

$$\mathbb{P}(v \in D_n) = \mathbb{P}(v \in D_n | v \notin D_0)\mathbb{P}(v \notin D_0)$$

$$\mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\mathbb{P}(\sum_{v'' \in N(v)} W_{v''v}\mathbb{1}(v'' \in D_{n-1} \cup D_0 \text{ WORT v }) \geq \Gamma_v | v \notin D_0))$$

and thus :

$$\boxed{\begin{aligned} &\mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\langle \tilde{W}^{n-1}_{v_1 v} \star ... \\ (2.10) \qquad &\star \tilde{W}^{n-1}_{v_i v} \star .... \star \tilde{W}^{n-1}_{v_{k_v} v}, CDF(\Gamma_v)\rangle \\ &v_i \text{ neighbor of v} \end{aligned}}$$

We have now found a way to derive the set D_n numerically, i.e. iterating the formula we have just obtained. It is the only possibility we have to derive it, it indeed sounds impossible to get a completely analytic function with this type of graph, since every node plays a different role and since our results depend so much on the geometry of the graphs (i.e. number of neighbors of node, their number of neighbors again etc.).

Let us now adapt this method to find the set D_n in the GK model and in the GHK model.

2.2.2 The GK model

The GK model and the Watts model are very similar, except that the GK model has directed links. In this model, we have now two types of neighbors, the in-neighbors and the out-neighbors, and only the in-neighbors can propagate the default cascade. There are now also two possibilities, either the in-neighbors and out-neighbors don't intersect, in other words, there are no bi-directed links, or there are. We are going to derive the set D_n in both cases, assuming again that there are no loops.

First case : no bi-directional links
In the first case, it is really easy and we can just follow the naive approach we did in the Watts model :

$$\begin{aligned} &\mathbb{P}(v \in D_n) = \mathbb{P}(v \in D_n | v \notin D_0)\mathbb{P}(v \notin D_0) \\ (2.11) \qquad &\mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\mathbb{P}(\overset{-}{\sum_{v' \in N(v)}} W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0) \geq \Gamma_v | v \notin D_0) \\ &\mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\mathbb{P}(\sum_{v' \in N^-(v)} W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0) \geq \Gamma_v) \end{aligned}$$

The only difference here is that we only consider the in-neighbors of v in the sum. We don't have the same problem we met in the Watts model because since there are no bi-directional links and no loops, the in-neighbors are upstream and cannot have been affected by bank v. So, the WORT condition is unnecessary here. We can now follow what we did before in the Watts naive approach and assuming that $\tilde{W}_{v'v}^{n-1}$ is here $W_{v'v}\mathbb{1}(v' \in D_{n-1} \cup D_0)$, we have :

$$\boxed{\begin{array}{ll} (2.12) & \begin{array}{l} \tilde{W}_{v'v}^{n-1} \sim [\mathbb{P}(v' \in D_0) + \mathbb{P}(v' \in D_{n-1})]W_{v'v} \\ + [1 - \mathbb{P}(v' \in D_0) + \mathbb{P}(v' \in D_{n-1})]\delta_0 \end{array} \end{array}}$$

and

$$\boxed{\begin{array}{ll} (2.13) & \begin{array}{l} \mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0) \\ \langle \tilde{W}_{v_1v}^{n-1} \star ... \star \tilde{W}_{v_iv}^{n-1} \star .... \star \tilde{W}_{v_{k_v}v}^{n-1}, CDF(\Gamma_v)\rangle \\ v_i \text{ in-neighbor of v} \end{array} \end{array}}$$

Thus, we have the iterative analytic formula we can use to derive the set D_n numerically. Let us now adapt the proof of the Watts model to the case when we allow bi-directional links.

second case : possible bi-directional links

If there are bidirectional links, the proof above doesn't hold. We would have the same problem as we had with the naive approach in the Watts model. In this case, we have to use the WORT condition. Therefore just like for the Watts model, we have :

(2.14)

$$\mathbb{P}(v \in D_n \text{ WORT v' }) = \mathbb{P}(v \notin D_0)\langle \tilde{W}_{v_1v}^{n-1} \star ... \star \tilde{W}_{v_iv}^{n-1} \star .... \star \tilde{W}_{v_{k_v}v}^{n-1}, CDF(\Gamma_v)\rangle$$
$$v_i \text{ in-neighbor of v, } v_i \neq v'$$

Then it is easy to show that we have exactly the same formula as the one for the Watts model :

(2.15)

$$\mathbb{P}(v \in D_n) = \mathbb{P}(v \in D_n | v \notin D_0)\mathbb{P}(v \notin D_0)$$
$$\mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\mathbb{P}(\sum_{v'' \in N^-(v)} W_{v''v}\mathbb{1}(v'' \in D_{n-1} \cup D_0 \text{ WORT v }) \geq \Gamma_v | v \notin D_0))$$

Indeed we can also add the WORT on the right hand-side because if a link is bi-directional it is the same reasoning as in the Watts model, and if it is not, we consider the probability of one of v's in-neighbors to default without regard to v, but since the link is not bi-directional and that there is no loop, v has no impact on this node's defaulting. Looking at it WORT v or not doesn't change anything. Thus we have almost the same formula as in the Watts model :

$$\boxed{\begin{array}{ll} & \mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\langle \tilde{W}_{v_1 v}^{n-1} \star ... \\ (2.16) & \star \tilde{W}_{v_i v}^{n-1} \star .... \star \tilde{W}_{v_{k_v} v}^{n-1}, CDF(\Gamma_v)\rangle \\ & v_i \text{ in-neighbor of v} \end{array}}$$

Let us now derive the math for the GHK model under the same conditions as well as the initially stressed bank set being independent from everything in the graph.

2.2.3 The GHK model

The GHK model is very different from the two other ones to the extent that we have both an upstream and a downstream cascade. This makes things a little bit more complicated but we can still follow the same method as we used before.

We need first to derive the default cascade. It is indeed important to notice that the default cascade is not affected at all by the stress cascade. However it affects strongly the stress cascade, so we absolutely need to know the set D_n before we try to derive the set M_n. It is important to notice that we also need to use the WORT condition.

Like in the GK model, we have :

(2.17)

$$\mathbb{P}(v \in D_n \text{ WORT v' }) = \mathbb{P}(v \notin D_0)\langle \tilde{W}_{v_1 v}^{n-1} \star ... \star \tilde{W}_{v_i v}^{n-1} \star .... \star \tilde{W}_{v_{k_v} v}^{n-1}, CDF(\Gamma_v)\rangle$$

v_i in-neighbor of v, $v_i \neq v'$

We can also derive the real probability of default, i.e. not WORT one neighbor :

$$\boxed{\begin{array}{ll} & \mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\langle \tilde{W}_{v_1 v}^{n-1} \star ... \\ (2.18) & \star \tilde{W}_{v_i v}^{n-1} \star .... \star \tilde{W}_{v_{k_v} v}^{n-1}, CDF(\Gamma_v)\rangle \\ & v_i \text{ in-neighbor of v} \end{array}}$$

Let us now first recall the stress condition and then let us derive the stress cascade :

$$\begin{cases} \displaystyle\sum_{v' \in N^+(v)} W_{vv'} \mathbb{1}(v' \in M_{n-1} \cup M_0) \geq B_v \\ \text{or} \\ v \in D_n \end{cases}$$

(2.19)

Let v be a node. First we need to make sure that v is not in the set M_0 :

(2.20)
$$\mathbb{P}(v \in M_n \text{ WORT v' }) = \mathbb{P}(v \in M_n \text{ WORT v' } | v \notin M_0)\mathbb{P}(v \notin M_0 \text{ WORT v' })$$

We now have to condition on the fact that v is not in D_n. Indeed, to be in M_n, either v is in D_n or it satisfies the stress condition.

(2.21)
$$\mathbb{P}(v \in M_n \text{ WORT v' }) = \mathbb{P}(v \notin M_0 \text{ WORT v' })$$
$$[(1 - \mathbb{P}(v \in D_n | v \notin M_0))\mathbb{P}(v \in M_n \text{ WORT v' } | v \notin M_0, v \notin D_n) + \mathbb{P}(v \in D_n | v \notin M_0)]$$

Here we have to notice one very important thing, since the set M_0 is chosen independently from the graph and also has no influence on the cascade of defaulted banks, we have :

$$\mathbb{P}(v \in D_n | v \notin M_0) = \mathbb{P}(v \in D_n) \tag{2.22}$$

This holds really because the stress cascade has no influence on default, if it had, it would be completely wrong, and we would have to find another method to complete the proof.

Thus we have :

$$\begin{aligned} \mathbb{P}(v \in M_n \text{ WORT v' }) = \mathbb{P}(v \notin M_0)[(1 - \mathbb{P}(v \in D_n) \\ \mathbb{P}(v \in M_n \text{ WORT v' } | v \notin M_0, v \notin D_n) + \mathbb{P}(v \in D_n)] \end{aligned} \tag{2.23}$$

Let us now work on the last unknown term :

$$\mathbb{P}(v \in M_n \text{ WORT v' } | v \notin M_0,\, v \notin D_n) = \tag{2.24}$$
$$\mathbb{P}(\sum_{v'' \in N^+(v), v'' \neq v'} W_{vv''} \mathbb{1}(v'' \in M_{n-1} \cup M_0 \text{ WORT v}) \geq B_v | v \notin M_0,\, v \notin D_n)$$

Let $\hat{W}^n_{vv''}$ be $W_{vv''}\mathbb{1}(v'' \in M_{n-1} \cup m_0)$, our condition is thus given by :

$$\mathbb{P}(v \in M_n \text{ WORT v' } | v \notin M_0,\, v \notin D_n) = \tag{2.25}$$
$$\mathbb{P}(\sum_{v'' \in N^+(v), v'' \neq v'} \hat{W}^n_{vv''} \geq B_v | v \notin M_0,\, v \notin D_n)$$

Now we have to notice that $\hat{W}^n_{vv''}$ is independent of both $v \notin D_n$ and $v \notin M_0$ because of the WORT assumption. However, unlike what we had for the Watts and GK model, we have here conditioned on $v \notin D_n$ and, D_n unlike M_0, is not an initial set, and therefore depends on many properties of the graph, like the geometry, the weights etc. Fortunately we are only looking at nodes that are out-neighbors of v, and not in-neighbors, and thus they have no impact at all on v being in D_n or not. We have to notice though that it wouldn't necessary work if v had neighbors that were both in and out, if stress could favor default and of course if the usual assumption that there is no loop was not satisfied.

Thus, we have :

$$\mathbb{P}(v \in M_n \text{ WORT v' } | v \notin M_0,\, v \notin D_n) = \mathbb{P}(\textstyle\sum_{v'' \in N^+(v), v'' \neq v'} \hat{W}^n_{vv''} \geq B_v) \tag{2.26}$$

and so :

(2.27)
$$\mathbb{P}(v \in M_n \text{ WORT v' } | v \notin M_0,\, v \notin D_n) = \langle \hat{W}^{n-1}_{vv_1} \star \ldots \star \hat{W}^{n-1}_{vv_i} \star \ldots. \star \hat{W}^{n-1}_{vv_{k_v}}, CDF(B_v) \rangle$$
v_i out-neighbor of v, $v_i \neq v'$

Finally, we get the expression for the stress set M_n WORT v' and then as usual we find the correct expression without the WORT condition :

$$\boxed{\begin{aligned} &\mathbb{P}(v \in M_n \text{ WORT v' }) = \mathbb{P}(v \notin M_0)[(1 - \mathbb{P}(v \in D_n)) \\ &\langle \hat{W}^{n-1}_{vv_1} \star \ldots \star \hat{W}^{n-1}_{vv_i} \star \ldots. \star \hat{W}^{n-1}_{vv_{k_v}}, CDF(B_v) \rangle + \mathbb{P}(v \in D_n)] \\ &v_i \text{ out-neighbor of v, } v_i \neq v' \end{aligned} \tag{2.28}}$$

Since we have already derived $\mathbb{P}(v \in D_n)$, we have got all the mappings to derive both the stressed banks and the defaulted banks set numerically.

Now that we have found a way to compute numerically the cascade in the three models we are working with, let us write formally the condition under which our calculation are exact.

2.3 The tree-like assumption

In the former sections, we said that we needed some independence so that our math formula are correct. Actually we wanted that two neighbors of a same node didn't interact. Let us be more precise about it.

Let us have the following notations :

- Let S be a set of nodes in the finance Graph.
- Let $\mathcal{F}(S)$ be the following $\sigma - algebra$
  F(S)=$\sigma(W_{vv'}, \Gamma_v$, for every $v, v' \in S$, $\mathrm{D}_0 \cap S, ...)$. Thus, it is generated by the balance sheets, the shocks, the degree of a node, and the weight of edges that are inside the subset.

If for every subset of the graph S and S' that only intersect in v, $\mathcal{F}(S)$ and $\mathcal{F}(S')$ are independent conditioned on $\mathcal{F}(\{v\})$, then we say that the graph satisfies the tree-like assumption. This is an assumption that is globally satisfied by the tree graphs only, for the simple reason that loops make this assumption wrong.

Therefore the tree graphs are the only graphs with which all this method is going to be totally correct. However, we can expect that it is going to be approximately true in a lot of graphs though, for instance if the loops are long enough. This is the kind of issues we are actually going to study in the chapter 3. We are now aware that our ideas won't be true in every case, so we are going to need a more precise idea of when our method is approximately correct or not. This is the whole point of the next chapter on numerical analysis.

To conclude our mathematical results, let us now derive another cascade condition that is going to be really interesting to understand how cascades grow in a graph.

2.4 The cascade condition

We are now going to derive a cascade condition using a totally different approach. Our first approach was very interesting because it enables us to compute the probability that a node defaults. It is a node by node approach, and given the distribution of probability of the initially defaulted banks, the distribution of the weights and buffers, we can derive everything concerning the probability that a node eventually defaults or not.

In this section we are going to focus on a more global approach : we would like to know quickly if the graph we have is systemically dangerous, without trying to know how a given node is going to be affected, and without taking into consideration the initial defaulted set. We are focusing on a financial graph property, involving only the buffers and the weights probability distributions.

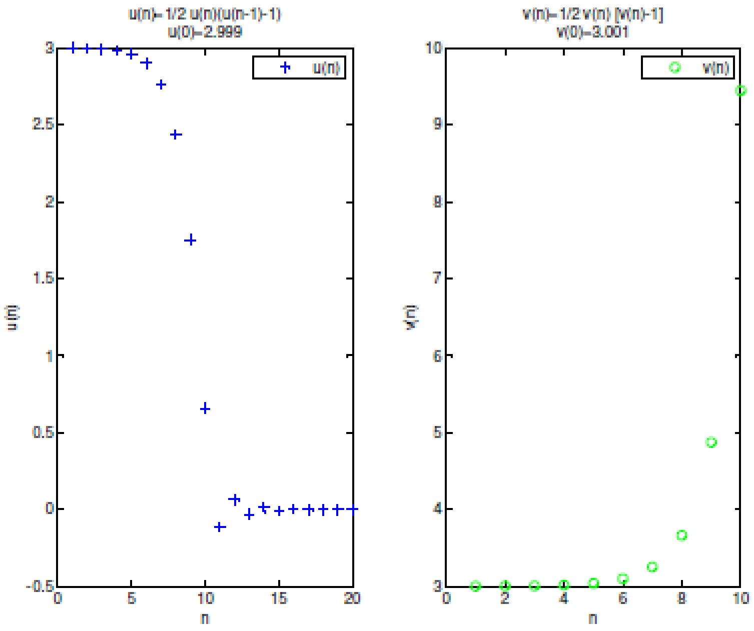
Even though this approach gives a lot less information than our former results, it is actually very interesting to know if a graph is systemically dangerous without regard to the initial condition. Actually, in the real financial world we might seldom have a precise idea of which banks are likely to default soon and cause a first initial shock that could trigger a cascade. Thus, having a more global approach makes sense and can be very helpful for policy makers as well as banks heads to take some measures to prevent the whole system from collapsing if there is some risk.

In this paper, the three models we consider are iterative models. We derive the sets of banks that have defaulted after n steps from those which have already defaulted after n-1 steps. We thus have an inductive sequence of vectors representing the probability of default of each node after n steps. Let Δ_n be a vector in $\mathbb{R}^N$ whose nth component is the probability of default of node n.

There exists a C^1 mapping Φ (that we actually have derived in the last three subsections for each model) such that :

$$\Delta_{n+1} = \Phi(\Delta_n) \tag{2.29}$$

The aim of our study is to know whether $\lim_{n\to+\infty} \Delta_n$ is close to zero or not. First zero has to be a fixed point, but it is the case here and it is obviously not enough. We know that there is an easy way to check condition to know whether an inductive sequence converges to 0 or not. For real sequences, we all know that we only need to check if the absolute first derivative at zero is strictly larger than one. If it is, the sequence cannot converge to zero and we can easily prove that by contradiction. If the first derivative is on the contrary strictly smaller than one, we can assert that there is small interval such that if the sequence ever enters this interval, we are sure that it is going to converge to zero, but of course we can't really say anything since it really depends on the initial conditions, as show the two following graphs next page. The first derivative at zero is in both case smaller than one, however if the initial condition moves a little bit, it changes completely the behavior of the sequence :



;

Figure 2.2: *Behavior of a same iterative sequence depending on the initial condition*

The same reasoning can of course be made in a larger dimension, but again we will only get an indication because it really depends on the initial condition. If the largest absolute eigenvalue of the first derivative at zero is strictly larger than one, and that the initial condition is not too bad (We could of course specify the meaning of "is not too bad", but there is actually no use doing that, because as we said before, we treat the initial condition in this problem as if it was unknown), it is then impossible that our sequence converges to zero. Thus, there will necessary be a cascade.

Let us derive the cascade condition for the GK model with possible bidirectional links, we proved in the last section that :

$$\begin{aligned} &\mathbb{P}(v \in D_n \text{ WORT v' }) = \langle \tilde{W}^{n-1}_{v_1 v} \star ... \star \tilde{W}^{n-1}_{v_i v} \star .... \star \tilde{W}^{n-1}_{v_{k_v} v}, CDF(\Gamma_v)\rangle \\ &v_i \text{ in-neighbor of v, } v_i \neq v' \end{aligned} \tag{2.30}$$

We can use this expression to differentiate the function Φ, differentiating with regard to $\mathbb{P}(v''_n$ WORT v'").Actually, we don't need to do that for every node v, v', v", v"', we only need to do that for vv' being a link, as well as v"v"'. Thus, if L is a number of links, our derivative can be described by a matrix in $\mathbb{R}^{L^2}$.

One can notice we don't use exactly the vector Δ_n, we use the probabilities that a node defaults without regard to one of its neighbors. The derivative is much easier to compute with this WORT condition, and actually it is easy to show that the components of Δ_n are close to one if and only if those ones are close to one, therefore if there is a cascade without the WORT condition, there is one with it and the other way round.

Let M be the first derivative in zero matrix that we have just described. If i and j are two different links of the graph, we can show easily after some lines of calculus that :

$$\begin{cases} M_{i,j} = \mathbb{P}(w_i < \Gamma_v) \text{ if the origin of i is the same node as the end of j} \\ M_{i,j} = 0 \text{ if it is not} \end{cases} \tag{2.31}$$

We just need now to compute the eigenvalues of this matrix and find out if one is larger than 1 or smaller than -1. We can also see that this matrix is really simple and that we don't need to compute anything to find its coefficients. These coefficients are only related to the vulnerability of a link and we can find here some interesting parallel with the paper by Watts. In this paper Watts finds a cascade condition that is related to the size of the largest vulnerable cluster (nodes that can adopt the property if only one of its neighbors has already adopted it). Here we are almost looking at the same kind of thing, the coefficients of this matrix are the probability that the weight of one neighbor link alone can cause the bank v to default.

Thus we have got a very interesting cascade condition that is only related to the graph itself. However, we have to be aware that this cascade condition is not really correct in some extent. We are actually only looking at the first derivative at zero, and we know that if all the absolute eigenvalues of the matrix are smaller than one, then there is going to be a small ball centered in zero such that if the initial condition is inside the ball, then the sequence is going to converge to zero. However, we don't know at all the size of this ball, and the initial condition might not be in this small ball, and the sequence might actually never reach the small ball. Moreover, we must be aware that if the initial condition is not zero, there is actually no chance that every bank has a zero probability of default. So the initial condition will actually never be in the small ball, but there is a big chance that if the first derivative's absolute eigenvalues are largely smaller than one at zero, they might all be smaller than one around zero, and therefore, if our initial condition is not too far from zero (which we hope is usually true), then it would converge to vector whose norm is close to zero which would be another fixed point of the mapping.

We have therefore given an indicator here : if the maximum absolute eigenvalue is smaller than one, we have a good chance of facing no big problems in the future, whereas if it is not, we have a large chance to have a default cascade, at least the financial network is going to be very unstable.

Given all these mathematical results, let us now find out more about all these models and about all our indicators through numerical analysis. We have to bear in mind that this subject is a very complicated mathematical area and that we cannot understand how these network behave without numerical simulations. Since we chose to be able to address every type of graph, we cannot prove the efficiency of our mathematical results in a more formal manner than what we've done. Besides, to get these results we had to make this tree-like assumption, and we know that the only graphs that satisfy it are the trees. Obviously our financial graphs are going to contain loops, therefore we have also to be able to know to what extent our results are going to be affected by the presence of loops.

Chapter 3

Numerical Analysis

Being able to picture what actually happens during the propagation of default in our graphs has been really helpful to understand to what extent we can accurately predict defaults and on which graphs. As Pr Hurd saw the software functioning for the first time, he was really excited and called it our mathematical microscope. He meant that we would be able to understand many things through these experiments and he couldn't be more right. As we started this project, and after we derived the math, we had actually no idea whether the tree like assumption was going to be realistic or not and if we were going to have some very bad results sometimes, some really good results some other times. After thousands of run, we can say that we understand a lot better the conditions that make our math realistic or not. Before we get into the details, let us first give some results about the discrete Fourier transform. This method has played a very important role in making our codes much faster, and actually fast enough to do enough simulations and learn a lot of things. After that, we will describe our codes and finally present our results.

3.1 The discrete Fourier Transform

The discrete Fourier transform we are going to use is a map on $\mathbb{C}^N$ that is really similar to the continuous Fourier transform we are used to working with. This kind of transformation is really useful when we have to deal with convolution. In our mathematical models, to derive the set D_n from D_{n-1}, we have to derive probabilities of that kind :

$$\mathbb{P}(v \in D_n \text{ WORT v' }) = \mathbb{P}(v \notin D_0)\mathbb{P}(\textstyle\sum_{v' \in N(v), v'' \neq v'} \tilde{W}^{n-1}_{v'v}) \geq \Gamma_v) \tag{3.1}$$

We have shown that this probability is actually just a kind of convolution since the sum of two independents random variables is the convolution of their probability densities. We can describe the probability density of a discrete random variable as a vector with its i^{st} component being the probability that this random variable is equal to i. Thus, our probability is going to be equal to :

$$\begin{aligned}&\mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\langle \tilde{W}^{n-1}_{v_1 v} \star ... \star \tilde{W}^{n-1}_{v_i v} \star .... \star \tilde{W}^{n-1}_{v_{k_v} v}, CDF(\Gamma_v)\rangle \\ &v_i \text{ neighbor of v}\end{aligned} \tag{3.2}$$

We are just going to show that it is an ideal case for using the Discrete Fourier Transform but let us first define the transform and give some properties.

Let u be a vector in C^n, and let u_0, ...,u_{n-1} be its components for a simple reason of convenience. Then the Discrete Fourier Transform of u is a vector in C^n whose k^{st} coefficient is equal to :

$$\mathcal{F}(u)_k = \sum_{0 \leq j \leq n-1} u_n e^{\frac{2ik\pi}{n}} \tag{3.3}$$

Thus, we get a new vector in C^n and we can find the former vector using the inverse :

$$\tilde{\mathcal{F}}(u)_k = \frac{1}{n} \sum_{0 \leq j \leq n-1} u_n e^{\frac{-2ik\pi}{n}} \tag{3.4}$$

Let us now give a few properties about the Discrete Fourier Transform that are going to be useful in our programs :

- $\tilde{\mathcal{F}}(\mathcal{F}(u)) = u$
- The Parseval identity : $\langle u, v\rangle = \frac{1}{n}\langle \mathcal{F}(u), \mathcal{F}(v)\rangle = n\langle \tilde{\mathcal{F}(u)}, \tilde{\mathcal{F}(v)}\rangle$
- $\mathcal{F}(u \star v) \simeq \mathcal{F}(u).\mathcal{F}(v)$ where the dot means the product of the two vectors components by components.

In the last equality, we wrote that the Discrete Fourier transform of a convolution product of two vectors was almost equal to the 'product' of the two Discrete Fourier Transform of the vectors. The reason for this almost equality is that, for this formula to be true, we have to use a circular convolution instead of the convolution we are using. Of course, we cannot do that in our problem because the sum of two independent random variables' law is a convolution of their laws and not a circular convolution. This errors occurs because we are working with vectors in a finite dimension space, and we will call that error an aliasing error.

Even though we don't have the choice and we have to make this error, we can make it very small. Let us model our law distributions with vectors in $[0, 1]^M$, we obviously can't take M infinitely large because we are working on computers. However by taking it large enough, the law is not very different from the real law, we just forbid too large numbers to take larger values than M. We have to notice that we are also doing an error with our convolution. We are indeed looking at the sum of two or even more random variables and we are trying to derive its law. We are therefore saying that the sum of all these random variables is also going to be smaller than M, but it is obviously wrong. We are therefore doing a bigger error than for a single random variable because this sum is much more likely to be larger than M. Yet, if we take M very large and only look at random variables with small tail distributions, the probability that a sum of such random variables is larger than M becomes really small and we make the aliasing error reasonably small.

Thus, in our programs, we are always going to adjust M. We have to take it large so that we don't make a big aliasing error, but small enough so that it doesn't run for too long. For example, if we are looking at Poisson random variables with a maximum parameter of 10 and that at most we sum 10 of them, we are going to choose M equal to around 200. For a Poisson random variables with coefficient 10, there is less than 2 percent chance that it is larger than 20. Thus for the sum of 10 independent such random variables, there is going to be a very tiny probability that it is larger than 200.

Now that we have explained the basic properties of the Discrete Fourier transform, let us apply these results to our problem :

We have to derive :

$$\begin{aligned}&\mathbb{P}(v \in D_n) = \mathbb{P}(v \notin D_0)\langle \tilde{W}^{n-1}_{v_1 v} \star ... \star \tilde{W}^{n-1}_{v_i v} \star .... \star \tilde{W}^{n-1}_{v_{k_v} v}, CDF(\Gamma_v)\rangle \\ &v_i \text{ neighbor of v}\end{aligned} \tag{3.5}$$

Using the results for the convolution, as well as the Parseval identity in the scalar product just above, we easily show that :

$$\begin{aligned}&\mathbb{P}(v \in D_n) = M\langle \mathcal{F}(\tilde{W}^{n-1}_{v_1 v}).\mathcal{F}(\tilde{W}^{n-1}_{v_2 v}). \ ... \ .\mathcal{F}(\tilde{W}^{n-1}_{v_i v}). \ ... \ .\mathcal{F}(\tilde{W}^{n-1}_{v_{k_v} v}), \mathcal{F}(CDF(\Gamma_v))\rangle \\ &v_i \text{ neighbor of v}\end{aligned} \tag{3.6}$$

We just have to look at this last equation to see how more efficient we are going to be. The usual method to derive the convolution of two vectors is equivalent to a matrix product. Here we are only deriving the Discrete Fourier Transform (Matlab has a very efficient algorithm for that) and then we do a component by component product. If we have for instance 20 neighbors for each node in average, it is clear that this method is going to make our program run much faster.

3.2 Description of our programs

I have spent a huge part of this internship working on Matlab codes. In systemic risk as well as on random network theory, computers are very important because there are only a few things that can be proved theoretically and without any approximations.

Let us in this section briefly describe some of the codes we wrote. One can also find the main codes in the appendix of this report.

The Main program

Our main program is adapting on Matlab the mathematical results we have proved just before. We therefore give as inputs the buffer probability distribution, the adjacency matrix of the graph (as a sparse matrix to gain some space), the weight distribution and the initial probability of default for each node.

Then we can apply the methods we have described above, should it be for the Watts model, for the GK model or for the GHK model. Thus we can derive the probability that a given node has defaulted after n steps. To do that, we loop n times and use our algorithm to find the set of probabilities of defaults of a node WORT one of its neighbors after n times. When these probabilities don't change more than 10^{-9} in two consecutive entries in the loop, we stop the program and find the real probabilities of default without the WORT.

Also, in our model we have made some assumptions, like the tree-like assumption, and we want to know how correct our results are. Thus, in parallel to these simulations we have also run Monte Carlo simulations to find the difference between our model and reality.

The Different types of graphs we considered

We have also worked on different types of graph among which the tree graphs, the Poisson random graphs and the cluster graphs. Let us explain a little bit more how we built these graphs and why we chose with them.

We are going to devote a section in the application chapter to explain what we mean with cluster graphs, so let us focus here only on the two other ones.

The Tree graphs play an important role in our model because they are the ones for which our mathematical results work best. They satisfy of course the tree-like assumption property and therefore enable us to see how large the other approximation errors can be. We thought right from the beginning that the tree-like assumption would be the largest cause of errors, yet we had to be sure of that fact. We therefore created a simple program to build tree graphs. We could think of many ways to build tree graphs but we didn't want our program to be too biased and create a lot of unusual tree graphs. Therefore natural recursive methods like, given a tree graph, attributing a random number of neighbors to every leaf, wouldn't actually be clever. At the last iteration, we would have way too many leaves.

Thus we decided to use another recursive method. Given a tree graph with n nodes, we link the $n+1^{st}$ node randomly to one of the n nodes we already have. Thus we get in the end a lot of different tree graphs with many less particularities than with the former method.

We decided to work also on Poisson random networks because the result proved in all the paper we worked with (Watts, GK, GHK, Hurd and Gleeson etc.) give results for infinite Poisson random graphs, which actually respect the tree like assumption. We wanted to see how accurate our method was with finite Poisson random graphs. Let us now explain how we built these graphs.

We call a Poisson random non directed graph with parameter λ and with N nodes a graph whose nodes have a random number of neighbors according to independent Poisson random numbers with parameter λ.

To build these graphs, we use the following algorithm :

- simulate N random Poisson variables with parameter λ. If the sum of these variables is odd, we add one to one of these numbers randomly.
- Then we have to think of these number of neighbors for each nodes as stubs attached to the nodes. Then we just need to randomly match these stubs. When all the stubs of a node have been matched we take it out of the possible matching nodes.

Then Pr Hurd and Pr Gleeson imagined a more complicated method to build the Poisson directed random graphs, with a finite or infinite number of nodes. This is not a very complicated method, it only adds some constraints so that there are enough out-stubs to match the in-stubs. However, we won't explain this method more clearly because it hasn't been published yet. We also used another method to create non-directed random graphs using the method just above. Of course it didn't create Poisson random graphs, but it was still very interesting to work with these graphs for the GK and the GHK model. We just basically used non directed graphs and suppressed one of the two directions randomly (or not all if we wanted some links to be bi-directional).

Now that we have described the graphs we mainly used, let us talk about the ways we are going to present the results in this paper.

How we presented the results

Our main goal is to understand how accurate our mathematical results are. We therefore have to compare our results with the reality through Monte Carlo simulations. In order to have a good view of what is happening we decided to do two different types of graphs.

The first one is focusing on a single financial network and we run our program in parallel to Monte Carlo simulations. In the end, we get two vectors representing the probability of default for every node of the graph with both methods. Then, assuming that the Monte Carlo simulation is closest to reality, we give show the different probability of default for each node sorted on the Monte Carlo results. This approach gives us easy to read information, but it is focused on a single graph, it cannot therefore give us good indications concerning how accurate the tree like assumption is on different kinds of graphs.

Therefore, we decided to run our program on a large number of different networks and store every result in a matrix. Then, using all these results, we derive the average probability of default in the network with the Monte Carlo simulations and with the analytic methods we have built. Then we show these two numbers for a large number of different networks on a graph, using different parameters for the buffer distribution, the weight distribution and the initially defaulted nodes.

3.3 The tree like assumption and reality

Before we go any further, let us give the main results about our models. Our model, as we are going to see below, works perfectly with tree graphs, and more or less in the general case. In this section, we will explain why it works less well with general graphs and eventually show at the end of this chapter that we can nonetheless get a lot of information about the systemic risk even with general graphs. We will indeed be able to build indicators in the cascade signal section that can warn us if the financial network is very unstable, and this actually the aim of all that study. Let us insist again on one point : we cannot merely run Monte Carlo simulations all the time because as soon as we work with 300 nodes graphs, they would take a very long time to run. It is incomparably faster to derive our indicators.

So let us first focus on the tree graphs, the ideal case, to see if our program runs well and if our mathematical results make sense. We are also going to see if the aliasing error is big or if, as we predicted, it gets really small by taking M large enough.

3.3.1 The tree graphs

Let us show some graphs showing the results in the tree graph case. Let us first use the same graph, and different buffer distributions so that we can have a small, a large and a medium cascade.

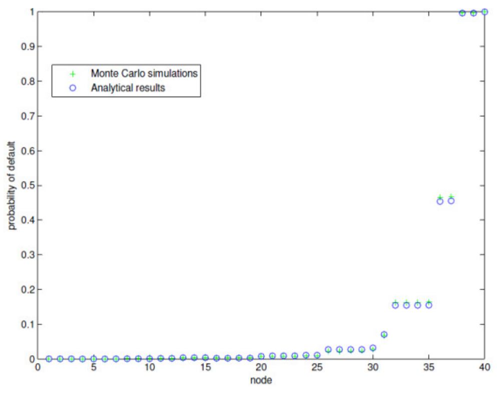


Figure 3.1: *small cascade in a tree graph*

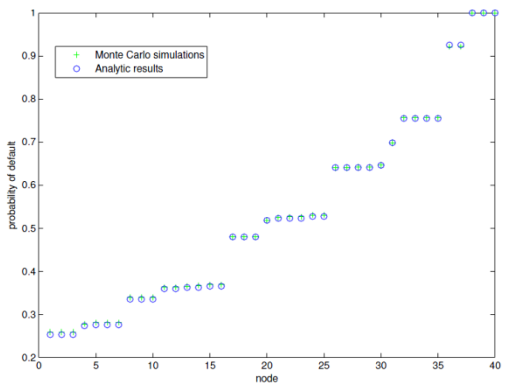


Figure 3.2: *medium cascade in a tree graph*

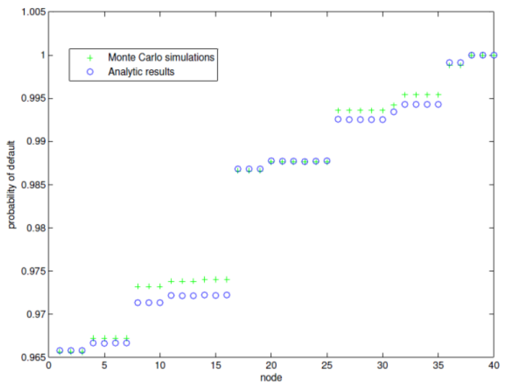


Figure 3.3: *large cascade in a tree graph*

As we can see on these three graphs, our model works almost perfectly on this tree graph, should we have a large, a small or a medium cascade. Of course, we cannot conclude that if it works on this tree graph, it works on every tree graph, but we can see below that with more than a hundred random tree graphs, the average probability of default is really close to the one derived with the Monte Carlo. To prove it more formally, we could do some statistical tests to check that the bias with the Monte Carlo is equal to zero. Pr Hurd and I imagined a way of testing it with a basic χ_2 test but this wouldn't give us a lot more information, because we are dealing with a infinite number of graphs, with a lot of parameters that can vary and we cannot do easily a χ_2 test to prove it is going to work with every graph and every parameter. Thus the best way to show these results is as far as we are concerned to show these graphs and run a lot of numerical experiments.

This kind of proofs was difficult for me at the beginning because I am used to very formal proves, but there is no way we can prove our mathematical models are working or not except through numerical experiments. Actually I felt bitter in some way that we couldn't prove things just with a piece of paper, but it turned out to be really interesting because during a major part of this internship we had to deal with our intuition only. Neither Pr Hurd nor I knew what was going to happen and we actually had to imagine a lot of numerical experiments. Sometimes they worked sometimes they didn't, but every time we understood more accurately how our model worked.

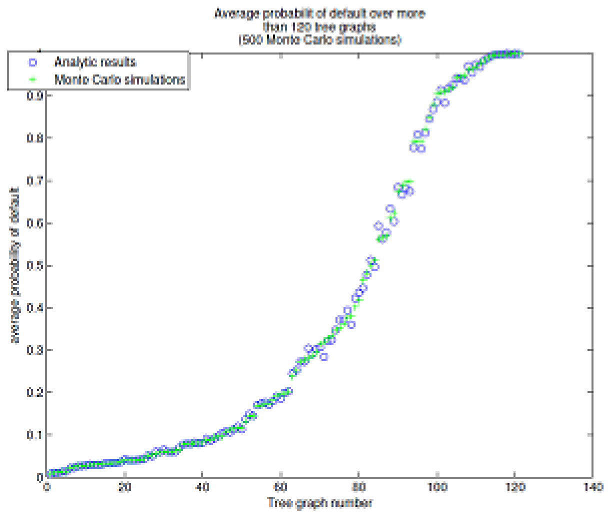


Figure 3.4: *Analytic results and Monte Carlo simulations for 120 tree graphs*

As we can see on the graph above, the analytic probability of default is very close to the Monte Carlo estimated one. We could get closer results by taking a larger number of simulations (we only used 400 Monte Carlo simulation for each point here). Had we used 10 000, the two graphs would be perfectly the same, but it would have taken a lot more time.

We have now shown that it works with tree graphs, let us now see the same kind of graphs with a finite random Poisson network.

3.3.2 The general case

Unfortunately, as we said before, it is not working that well with general graphs. On the next page, we can see the results we get for a finite Poisson random graph with different buffer distribution so that we can have different types of cascades. As these graphs show us, we can manage to predict very accurately small and large cascades, but we can't predict accurately the medium cascades at all. Even though we again only showed our results on a single graph, we ran a lot of experiments and obtained always the same results. Besides, we are going to provide in the cascade signal section a graph with 400 different simulations on different random graphs very consistent to these results.

Thus we can accurately predict small and large cascades, but not the medium ones, let us now explain why there is actually no real hope that we can do any better with these models. In order to do that, we are going to explain the effect of loops in our model in two following subsections.

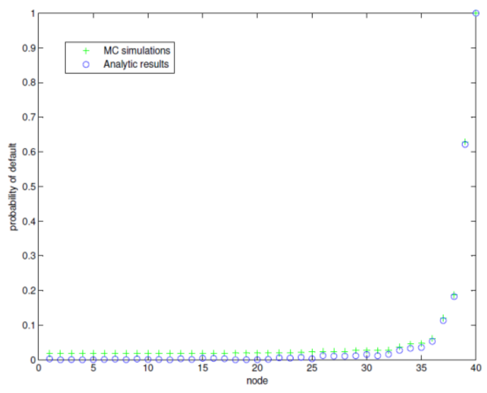


Figure 3.5: *small cascade in a normal graph*

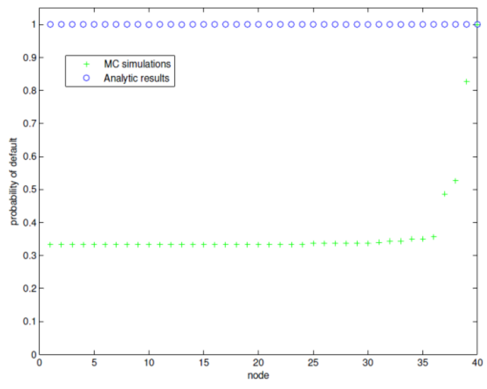


Figure 3.6: *medium cascade in a normal graph*

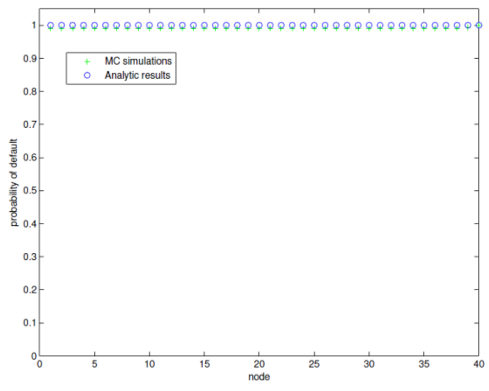


Figure 3.7: *large cascade in a normal graph*

3.3.3 The effect of a single triangle

Let us here focus first on the easiest type of loops : the triangles. We are going to show how the mere addition of a triangle in a tree graph can completely spoil our results for a medium cascade. However for a large and a small cascade, the results will still be very good.

On the following graphs on the next page, we added a single triangle in a tree graph, always at the same place. In these three pictures, the left graph is always the same tree graph, and on the right we always have the same triangle added at the same place.

Thus we can see that the mere addition of a loop can spoil all our results. Yet, the good news is that it doesn't spoil them in every case, if we deal with a small or a large cascade, we are still able to give very precise information. However, when we face a medium cascade, our model most of the time overestimates the probabilities of default.

When we saw these results for the first time, we understood that we couldn't hope to be able to state that our results were accurate in case we had a medium cascade. However we wanted to understand better why the loops spoil our results that much in the medium cascade case. We looked at papers dealing with triangles and loops in such kinds of models. Some of them were encouraging and said that the results could get better if the length of the loop became larger of if the graph had a larger number of node and contained a lot of loops. We decided thus to test these ideas and tried to learn more about the loops.

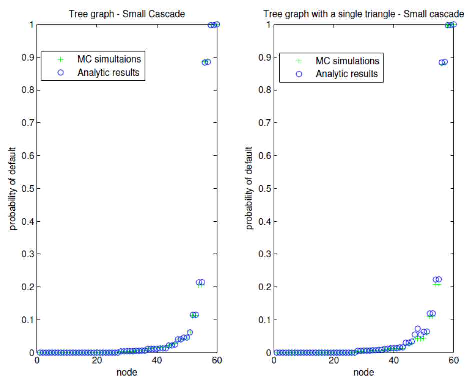


Figure 3.8: *addition of a single triangle - small cascade*

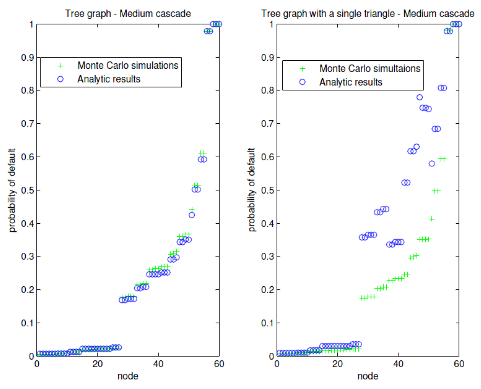


Figure 3.9: *addition of a single triangle - medium cascade*

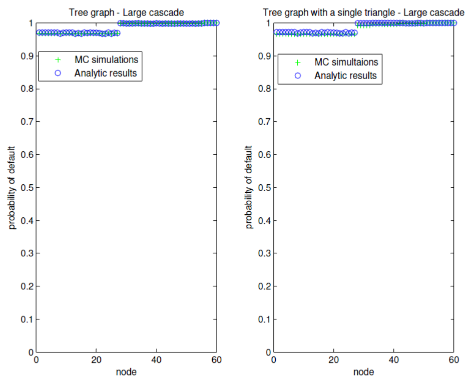


Figure 3.10: *addition of a single triangle - large cascade*

3.3.4 More about the loops

When we started our work on the loops we had three main questions in mind that we are now going to answer in this subsection :

- Our first questions stems from what we learned in the last subsection : why is the loop spoiling our results that much for medium cascades and not for a small cascade and a large one ?
- Is it going to get better with a large number of nodes and a highly clustered graph ?
- If we are forbidding triangles and squares for instance, and if we are only dealing with long loops, is our model going to give better results ?

Let us first try to answer the last two questions. We must not forget that we are writing this model for financial networks, and as we saw at the beginning of this report with the estimated picture of the European financial network, we are not looking at very huge networks. At most we are going to have around one or two thousand banks, so we cannot work on networks containing tens of thousands of banks. Running MC simulations on a very large network takes a very big amount of time so we decided to to run experiments on around two hundred nodes graphs to see if bigger networks gave more accurate results than small one. Our results can be seen on the graph below. The left panel is giving our results for small graphs and on the right we can see our results for highly clustered graphs with a large number of nodes.

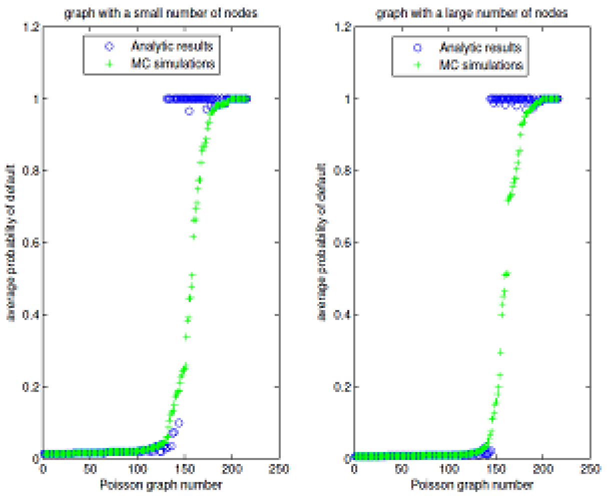


Figure 3.11: *Monte Carlo simulation and analytic results for graphs with a small or a large number of nodes*

As we can see on the graph just above, there is not much of a difference : we still predict well the small and large cascade, but the medium cascade are mistaken for large cascade by our program. It might get better for very large graphs containing tens of thousands of nodes, but that is not actually the kind of things we are studying here. Let us now try to answer to the third question.

This question is really interesting : Is the size of the loops correlated with the error we are doing ? The answer is yes, it sounds obvious that on a network of ten thousand node, if there is only one loop containing a thousand nodes, you can expect it to behave like a tree graph. However with our medium-size graphs, the answer is less easy to give. We decided to run experiments to test this assertion. We thus decided to take a graph with a suitable buffer for medium cascades. Then we created a program that took a length and a graph as inputs and that basically erased loops that had a smaller length than the length input. Thus we could test if by suppressing triangles, square, pentagons, etc. we would be able to make our results a lot better. Here are the results we got :

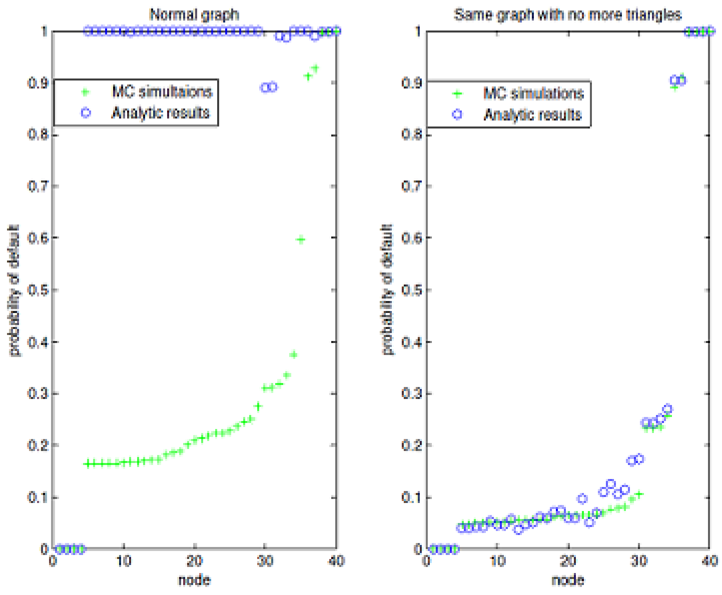


Figure 3.12: *Effect of the removal of small loops in many cases*

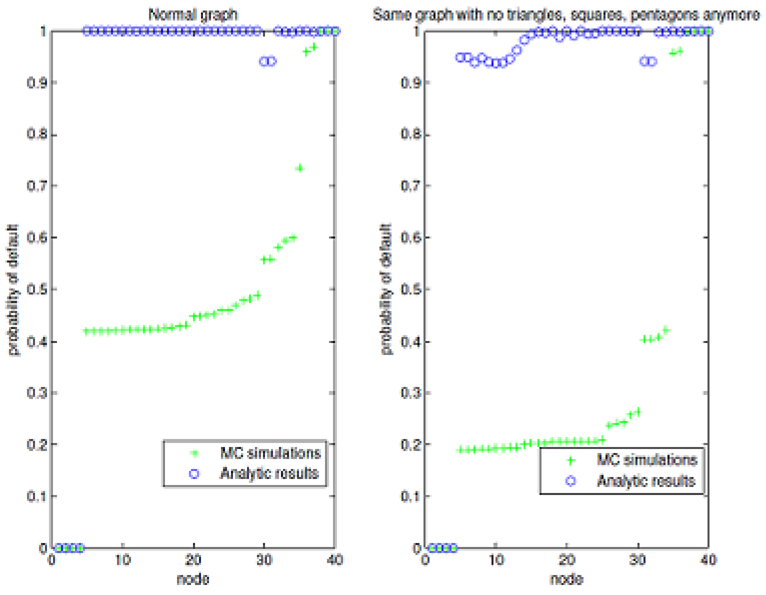


Figure 3.13: *Effect of the removal of small loops in a bad case*

We can see on the first figure a medium cascade on a graph, and a medium cascade as well but this time on a graph with no triangles anymore. The difference between the two pictures is clear, on the left hand picture, our prediction is very bad, whereas on the right hand side it is actually really good. This tends to let us think that taking small loops out of our graphs would make our prediction way better. Actually this is not what always happens. We can say that in most cases, it makes things better, but it is not always enough, and even though we suppressed triangles, squares, and pentagons on the next picture on the right, we still get a very bad prediction. We can thus conclude with these experiments that our error is somehow correlated to the size of the loops, but we cannot however assert that our predictions are correct if we consider medium cascade graphs with no small loops, for the second picture shows they are not. What this experiment shows is that it totally depends on the loops, in other words, where they are in the graphs, which nodes they involve etc. However there are still some cases where we can actually predict the effect of a loop. Let us try to answer the first questions and give some more ideas about the effect of these loops in our models.

It sounds really striking that sometimes these loops spoil completely our results and some other times they actually don't. Why don't they spoil our results when we have a small cascade or large cascade ? This is a question that has also puzzled us during this internship. Actually we managed to come up with some answers. After having done a lot of experiments on small cascades with graphs filled with loops, we have found out that when we get a small cascade, if we only consider the sub-graph of nodes with a medium or large probability of default (let us say more than 10 percent), we actually always get a tree graph. This led us to think that a loop can only affect the results of a graph when all its nodes have a medium or large probability of default. If not, the graph is going to behave like a tree graph and the loop won't have any effect on our results.

The only explanation that we can come up with (of course it is not a formal proof, it is more or less a guess from all the experiments we ran) is that the loop must most of the time "feed themselves", i.e. behave as in the following drawings :

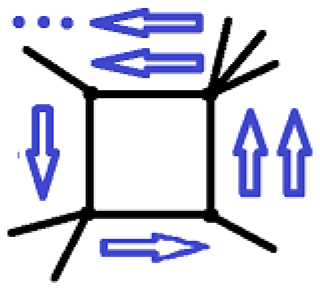


Figure 3.14: *Supposed behavior in a loop*

One node with a medium probability of default increases first the probability of default of the second node, then this one increases the third one's, which increases the first one's again, and this until they reach a probability of default close to one, and then it modifies the probabilities of all their neighbors and the rest of the graph. Following this idea, there must be a threshold under which the probability remains small and above which it grows to one or to a much larger value than it is supposed to. All this process of course is more derived from a guess after we ran a lot of experiments rather than based on a formal proof, but this is actually consistent with all our former remarks :

- If the loops has one of its nodes that has a small probability of default, then the process cannot take place and the results are good, that is exactly what we observe with a small cascade. Also contrary to the experiment we used before, if we have a tree graph with for example nodes 1, 2 and 3 having a very small probability of default (less than 5 percent) and if we link these three nodes, the results given by the Analytic model is still going to be very precise. This is because the process is too weak to feed itself.
- If all the nodes of the loop have a high probability of default in the MC simulations even if only a medium cascade occurs, then the results are still good. That is also what we can observe. If again we have a tree graph with for example nodes 1, 2 and 3 having a high probability of default (more than 0.9) then if we link these three nodes, the results given by the Analytic model is still going to be very precise. The three nodes have already a high enough probability of default and it cannot feed itself much more. It explains also why our result are still good with a large cascade.
- If we have a medium cascade with three nodes with a medium probability of default, then it is going to happen and the results are going to be wrong, this is what the experiment with the tree graph showed us.

So even though we don't have a proof of that fact, it is a good and intuitive way of explaining what we can observe through all the experiments we ran. We can truly say that all these experiments have helped us understanding our models a lot better and have enabled us to set some limits. We therefore really doubt that these kinds of models will be able to predict accurately every medium cascade because this loop effect might always happen. If we wanted to do so, we would have to correct the model to take these things into account and this sounds very difficult, maybe impossible.

Now that we are aware of these limits, let us still try to learn as much from our model as we can. We have been showing its limits in the last few paragraphs but there are also a lot of information we can get from this model. We are indeed able to create an indicator or a signal for a systemically dangerous network as we are going to see in the following section, and that was exactly the aim of this study.

3.4 The cascade signal

Let us now build 2 indicators that enables us to check whether a graph is systemically important or not. In order to do that, let us have a look at one last graph. It shows the average probability of default over around 400 hundred random graphs.

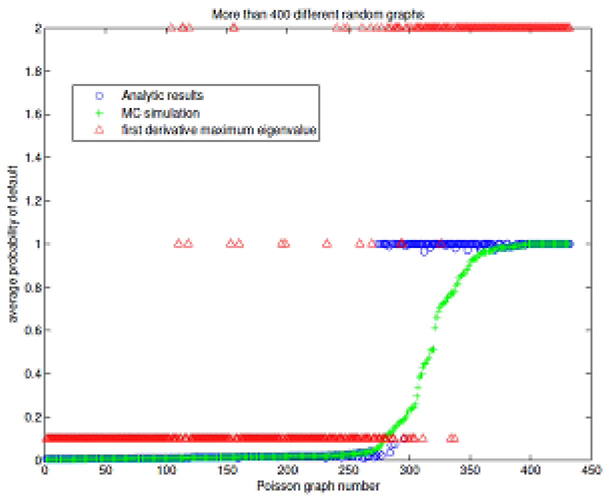


Figure 3.15: *Our indicators on more than 400 graphs*

We can see that what we have learned in the last few chapters is really consistent with what we can observe. Our analytic result can very accurately predict the small and large cascades, and as soon as we get a medium it overestimates it to a large cascade. However, we can get a clear signal from that :

- If the analytic result gives a small average probability, the financial graph is safe. We are indeed sure that it is going to fit with the Monte Carlo and we can even have a look at the analytic probability of default for each nodes to see which banks are the most dangerous ones.
- If we get a high average probability of default, then we know that the MC simulation would have given us a large or a medium cascades, so at least 15 percent of the banks in the network are likely to fall. Anyway, the financial network is in both cases very unstable and policymakers or bank CEOs should take measure to avoid an economic catastrophe.

On this picture, we can also see in red the cascade signal we derived at the end of the last chapter. We computed the maximum eigenvalue of the first derivative at zero. We set the red signal to 0.1 when this largest eigenvalue is smaller than 0.9, 1 if it is between 0.9 and 1.1, and 2 if it is larger. When the signal is equal to 1, we suppose we are in the uncertainty zone and we cannot affirm anything (as we said before, the first derivative might be smaller than one at zero but not around zero and the sequence could thus never enter the small ball around the fixed point it is supposed to converge to, or could be larger than one at zero and smaller just around zero). However when it is equal to 0.1, we expect the cascade to be really small and we expect the system to converge to a fixed point close to zero, and if the signal equals 2, we expect it not to converge to a small cascade and then we expect a large or a medium cascade.

We thus have two different indicators that can help us know whether the financial network we are looking at is systemically important or not. We can see that these indicators are almost always consistent with one another, but the red one can sometimes give wrong information. For many reasons that we have already mentioned, it is less precise, and further more when it is equal to one, we cannot give any good information except that we should be careful. However, to compute this signal we need a lot less information than for the other one. For instance, we don't need to know anything about the initial distribution of default. This information might be very hard to find and therefore using this indicator makes complete sense. If we have the initial probability of default, then the other indicator is much more precise, and can even give us in case of stable network precise information about the banks that are the most likely to default if one crashes. This piece of information might be really helpful for policymakers.

We have now reached the goal of this internship since we have build these two financial indicators for the stability of a network. They can both be very helpful and they are a lot less time consuming than using mere Monte Carlo simulations. Let us now try to use them on an interesting example.

Chapter 4

Application : the cluster graphs

In this chapter, we are going to use our results to understand how a default cascade propagates in a particular type of graph and how we can best stop it. In this section we are going to focus on the ”cluster graphs”. Everything we did during this internship was strongly influenced by the picture of the Europe financial network we received at the beginning. We decided to abandon the infinite random networks to focus on graphs that were much closer to reality.

4.1 Definition

We decided therefore to work also with another kind of graph we called cluster graphs. When we looked for the first time at the Europe financial graph, we actually thought the structure of the graph was very particular and interesting. We can easily feel the presence of the countries behind this financial network. In every country we can find a certain number of banks having mainly business relations with other banks from the same country and only a much smaller number of banks having business with the banks abroad.

Thus we decided to create a graph having that kind of structure :

- Every country is modeled with a finite Poisson random network with the same parameter.
- Among the banks of a country, some banks are considered to be international and are authorized to do business with other international banks abroad. We give their number as an input. All the international banks form actually another finite Poisson random network with a different parameter given as an input.

We thus obtain a graph that is looking like the little drawing just below and that tends to model the European financial network.

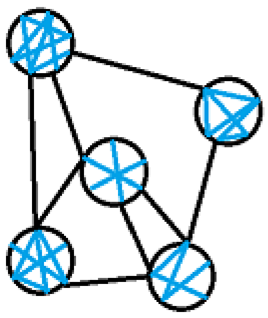


Figure 4.1: *drawing representing broadly a cluster graph*

4.2 Default cascade in a cluster graph and a policy to stop them

Like in the other graphs, we have three types of cascades and like before, our model predicts very well the small and large cascades.

Let us nonetheless have a closer look at what could prevent a cascade from happening in such a graph. We are going to try to find some policies that would make the cascade the smallest possible, even in the country where the initial shock is triggered.

A first measure

The first idea we can have, given such a graph, is to force the banks to allocate a larger part of their balance sheet to the buffers they use in case a bank they have business with defaults. It is obvious that rising the buffers to a very high value is going to contain default, but let us try to see on this very particular kind of graph how big the buffers must be to stop default.

As we can see just below, our indicators obviously assert that there is going to be no cascade if we take very high buffers. However to have both indicators saying that the the graph is safe, we have to have a very high buffer. We can see that we need the buffer to be equal to 3.6, which means that every bank allocates around 25 percent of their balance sheet to their buffer to be sure that they are going to be in safe financial graph (the figure 25 percent is easily derived from 3.6 thanks to the properties of the models and it is not interesting to give the details). This is for sure a very high number and the financial network would be highly inefficient. Let us now have a look at another policy that would be actually much more interesting.

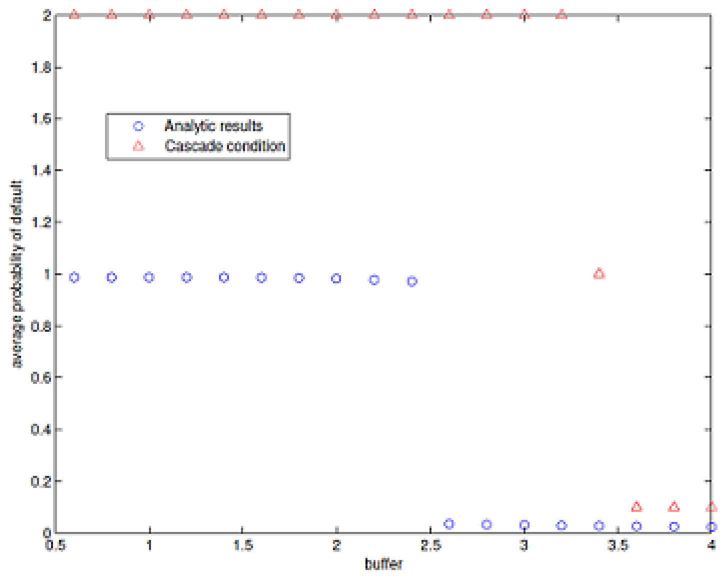


Figure 4.2: *Indicators for a cascade in a cluster graph for different buffers*

A cumulative buffer

A second idea we can have is to protect the international banks. Default cannot spread to the country if default has not touched its international banks first.

We can therefore do two things, either simply make the buffer of the international banks higher, but it would require them to devote a large part of their balance sheet to their buffer. Again this would be more efficient than what we have just done before, but we can actually think of another way of doing it. We can ask the national banks to contribute to the buffer of the international banks. Thus, they would contribute to their own protection because if the initial shock happens elsewhere, they cannot go into bankruptcy unless an international banks of their own country has not fallen. As we can see in the graph just below this method is very efficient. On the left we have the cascade if we don't do anything. Almost every bank defaults. In this graph every bank has a buffer of 10 percent of its balance sheet. If we simply reduce the buffers of the national banks to 5 percent and make the one of the international banks bigger thanks to that money, we are in the situation of the graph on the right. Only a few banks of a single country go into bankruptcy. No international bank is touched. And even in the country where the initial default occurred, not every bank, just a half of them actually go into bankruptcy. Thus, this policy of protecting the international banks thanks to every national bank is very efficient for the whole graph. We also still kept a little buffer for the national banks otherwise every time a default occurs in the country, every national bank would fail. This could even cause an international bank to fail because the majority of its business partners have already fallen, and this would put the whole graph in danger.

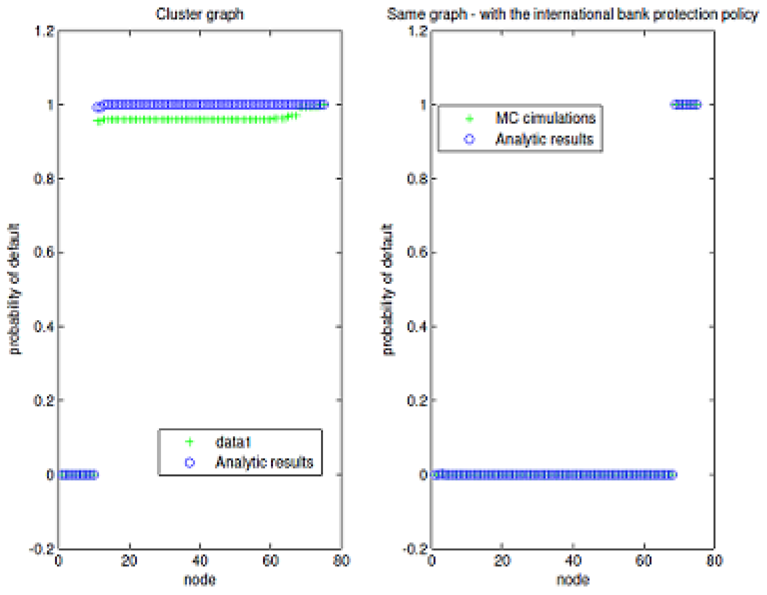


Figure 4.3: *Effect of our policy on a same cluster graph*

Thus our policy makes sense to protect such a graph and it makes sense to protect the most important banks, even in the real world. However it is just a very simple model of graph for Europe, and we would need some real data to find a much more realistic policy. The amount of money exchanged between banks is a data very hard to find, at least for us, but when we contacted people at the Banque de France and at the ACP, they told us that they had started gathering this kind of information a few years ago and that they might be able to have the order of magnitude of these transactions, but it is still highly confidential. This is a very big limit to all our work because we would need real data to adjust our models and our graphs. Only then we would be able to study the efficiency of a policy on the real financial network.

Conclusion

To conclude this report on my research internship at Mcmaster university and the Fields Institute, I want to say that It was a very enlightening piece of work for a few reasons.

First, I had the possibility to really experience academic research for the first time in my life. This was very different from what I did at Polytechnique. I was used to math that work perfectly, very formal proofs and very abstract subject. Here I had to face the unknown, be aware that I couldn't go very much further with formal proofs than the math models we derived, and therefore I had to learn to understand the subject without formal equations. I had to create experiments and test everything I had in mind. Sometimes it worked, sometimes it didn't and then I had to come up with a new experiment to try to explain my point. This was really a different kind of reasoning than the one I am used and it was therefore very interesting.

It has been also very interesting to address a very concrete problem. I have spent most of my life trying to understand math to be able to follow other math lectures. Here it was the other way round. I had to use the math I had been taught, and besides, not always the one I thought I would use. Before arriving here, I thought I would use stochastic calculus, Brownian motions etc. but I have never needed any of these mathematical concepts. However I had to use some network theory I was taught in mathematics lectures as well as in computer science courses. I also used some optimization, along with some dynamical system tools etc. One has to be very open minded and be ready to use tools from a very wide range of subjects to address their problem.

Finally, it was also very enlightening to work on a topical subject that is threatening our global economy. Not only did this work help me to understand the crises better, but I had the chance to study some different aspects of crises. This work is only a part of what I did at the Fields, and I had the chance to work with Pr Grasselli on more macro-economical views of crises that helped me understand these issues a lot better. Being at the Fields Institute and at McMaster University, I had also the opportunity to meet very skilled people in many disciplines. I had the chance to attend a conference by Steven Shreve about trading strategies. I also had the pleasure to listen to a former student of Polytechnique who is now a professor at Stanford University.

Emmanuel Candès gave fascinating talks about signal acquisition and processing with very beautiful operation research and statistic tools. I also could attend an amazing series of seminars around Steve Keen's macroeconomic models, and all of this helped me broaden my mind and made my interest for research even larger. Before this internship at the Fields, I was almost sure I wanted to go and work in a bank after, now I really envisage continuing on a PhD after my master's degree next year, and maybe go to academia after.

Acknowledgements

I would like to thank wholeheartedly both Pr. Hurd and Pr. Grasselli for helping me write this report on systemic risk. I really appreciated their being always ready to spend a lot of time working with me and to explain me a lot of things about this very subject and many others. This has been a real pleasure working under their supervision during all this summer in Canada.

Appendix A

Codes

In this chapter, we just give the main codes we used. Of course, we used many other codes, but the report would be much too heavy if we included them all.

```
avggg_ter=z;
[N N1]=size(A);
Gamma=zeros(N,M);
w=zeros(N,N,M);
ww=9;
for i=1:N
    tab=neighbor(A,i);
    n=length(tab);
     if(n>0)
%            if(ismember(i,banque_int11))
%                 Gamma(i,2:M) = poisspdf(0:M-2,(1+coeff_int)*phi*n).';
%            else
%            Gamma(i,2:M) = poisspdf(0:M-2,(1-coeff_int*4/15)*phi*n);
%            end
        Gamma(i,2:M) = poisspdf(0:M-2,phi*n).';
        for j=1:n
           neigh=tab(j);
%             w(i,neigh,2:M)=poisspdf(0:M-2,ww).';
           w(i,neigh,10)=1;
%             w(i,neigh,2:M)=binom_pdf(5,9,3,M-1);
        end
    end
end

F=cumsum(Gamma.').';
```

```
%% cascade condition

A_S=sparse(A);
[zr,zrr]=find(A);
B=[zr,zrr];
[N N1]=size(A);
L=length(B(:,1));

Deriv=zeros(L);

for i=1:L
    for j=1:L
        u1_1=B(i,1);
        u1_2=B(i,2);
        v1_1=B(j,1);
        v1_2=B(j,2);
        if(u1_1==v1_2)
            Deriv(i,j)=F(u1_1,10);
        end

    end
end

dd=eig(Deriv);
rayon_spectral=max(dd);

%%
rho=zeros(N,1);
rho(8)=1;

% rho=1/N*ones(N,1);

tol=0.0000000001;
secu11=10000;
% Let's have two matrix for pwort, the current one and the former one to
% check for the infinite norm : pwort1 : the new one and pwort2:the ancient one.

pwort1=zeros(N);
pwort2=zeros(N);

%let's initialise wtild : the density matrix of
%wtild(i,j)=w(i,j)*1(j   Mn U Mbar WORT v :2L*2*M)

%let's initialise this wtild :
wtild=zeros(N,N,M);
```

```
wtild(:,:,1)=1;

dirach0=zeros(M,1);
dirach0(1)=1;
fftdirach0=fft(dirach0);
fftw=zeros(N,N,M);
% let's fftw be the fft of the weight distrib
for i=1:N
    for j=1:N
        fftw(i,j,:)=fft(reshape(w(i,j,:),[M 1]));
    end
end

for i=1:N
    if (rho(i)==1) % i is defaulted, so the wtild(j,i)=w(j,i)
        tab_n=neighbor(A,i);
        if(length(tab_n)>0)
            for j=1:length(tab_n)
                neigh=tab_n(j);
                wtild(neigh,i,:)=rho(i)*reshape(w(neigh,i,:),[1 M])+(1-rho(i))*
                    dirach0';
            end
        end
    end
end
fftwtild=zeros(N,N,M);
% let's take the fft of every wtild(i,j,:)
for i=1:N
    for j=1:N
        fftwtild(i,j,:)=fft(reshape(wtild(i,j,:),[1 M])); %fft
    end
end

fftF=zeros(N,M);
fftF=fft(F')';

Norm_inf=1;
while( (Norm_inf>tol) && secu11>0)
    pwort2=pwort1; % initialisation of pwort2, to see the change in the end of the
        loop.

    % we are going to update pwort1
    %we consider the node i
    for i=1:N
        tab_n=neighbor(A,i);
```

```
         nb_neighb=length(tab_n);
         if(nb_neighb>0)
             for j=1:nb_neighb
                 neighb1=tab_n(j);
                 % we are now going to update pwort1(i,neighb1)
                 vect=ones(M,1);
                 %let's do the product comp by comp of the ffts of the wtild(aa,i)
                     without neighb1
                        for z=1:nb_neighb
                            if(z~=j)

                                neighb2=tab_n(z);
                                vect=vect.*reshape(fftwtild(i,neighb2,:),[M 1]);

                            end
                        end

                  pwort1(i,neighb1)=1/M*(1-rho(i))*fftF(i,:)*vect;

             end
         end
   end
  Norm_inf=max(max(abs(pwort1-pwort2)));
  %Let's now update fftwtild(i,neighb1) with the new pwort1
  for i=1:N
       tab_n=neighbor(A,i);
         nb_neighb=length(tab_n);
         if(nb_neighb>0)
              for j=1:nb_neighb
                  neighb1=tab_n(j);
                  fftwtild(i,neighb1,:)=(1-rho(neighb1)-pwort1(neighb1,i))*fftdirach0 +
                      (rho(neighb1)+pwort1(neighb1,i))*reshape(fftw(i,neighb1,:),[M,1])
                     ;
              end
         end
  end
     secu11=secu11-1;
end

pwort=pwort1;

%let's derive p now
p1=zeros(N,1);

for i=1:N
```

```
        tab_n=neighbor(A,i);
         nb_neighb=length(tab_n);
         vect2=ones(M,1);
         if(nb_neighb==0)
              p1(i)=rho(i);
         else
               for j=1:nb_neighb
                   neighb1=tab_n(j);
                   vect2=vect2.*reshape(fftwtild(i,neighb1,:),[M 1]);
               end
         p1(i)=rho(i)+1/M*(1-rho(i))*fftF(i,:)*vect2;
         end

end
p=p1;
Norm_inf;
% end
```

Listing A.1: Analytic code

```
%UU est la matrice que l'on va utiliser pour comparer les resultats.
sensibilite=0.15;
[N N1]=size(A);
D=8;

Gamma=zeros(N,1);
W=zeros(N);
av=0;
for i=1:N
     [tabvoisins_i,nbvoisins_i]=voisins(A,i);
     if(nbvoisins_i>0)
          Gamma(i)=nbvoisins_i;
          for j=1:nbvoisins_i
               jjj=tabvoisins_i;
          W(i,jjj)=9;
          end
     end
end




%soit C l'ensemble des banques et si C(i)=1 alors la banque est d j
% d faite

C=zeros(N,1);
C(D)=1;
```

```
N1=C;
N2=zeros(N,1);
a=sum(N1);
securite=10000;
entree=1000;



while(a>0 & securite>0 || entree>0)
    N2=zeros(N,1);
    for i=1:N
        %si la banque est dans les banques touches lors du dernier tour ds
        %la boucle

        if(N1(i)==1)
            [tab_i nn_i]=voisins(A,i);
            for j=1:nn_i
                jjj=tab_i(j);
                %tester si j est touch
                if(C(jjj)==0)
                    [tabvoisins_jjj nbvoisins_jjj]=voisins(A,jjj);
                    %calcul du poids des arrtes relies ) des banques touches
                    poids_jjj=0;
                    for z=1:nbvoisins_jjj
                        zzz=tabvoisins_jjj(z);
                        %si le voisin zzz est touch on ajoute le poids de
                        %l'arrte
                        if(C(zzz)==1)
                            poids_jjj=poids_jjj+W(zzz,jjj);
                        end
                    end
                    if(poids_jjj>=Gamma(jjj))
                        %si oui, l'ajouter aux banques touches et  N2
                        C(jjj)=1;
                        N2(jjj)=1;
                    end
                end
            end

        end
    end
    N1=N2;
    a=sum(N1);
    securite=securite-1;
    entree=entree-1;
```

```
end



res=sum(C)
C
```

Listing A.2: Monte Carlo resolution

```
function A=tree_graph(N)

if(N==2)

    B=zeros(2);
    B(1,2)=1;
    B(2,1)=1;
    A=B;

else
    n=N-1;
    B=tree_graph(n);
    %c r ons un nombre a l atoire entre 1 et n auquel le noeud se raccorchera
    %(rd)
    aa=rand(1,1);
    rd=ceil(n*aa);
    % on va donc se raccrocher a rd
    vect=zeros(n,1);
    vect(rd)=1;
    B(N,:)=vect;
    vect2=vect;
    vect2(N,1)=0;
    B(:,N)=vect2';
    A=B;
end
```

Listing A.3: Tree graph

```
function A=matrix_2(N,z)

            % Ecrivons un programme qui simule une loi de poisson de paramtre z :
            rs = poissrnd(z,1,N);
            M=zeros(N,N);

            compteur=0;

```

```
     compteur=sum(rs);

 %compte le nombre total d'arrtes, il doit  re  pair, sinon on rajoute
    une
 %arrte   un noeud pris au hasard.

 if mod(compteur,2)>0
     t=ceil(N*rand(1,1));
     rs(1,t)=rs(1,t)+1;
     compteur=compteur+1;
 end

 u=1;
 securite=1000*z;
tab=1:1:N;
lg=N;
ooo=find(rs==0);
lg=lg-length(ooo);
tab(ooo)=[];
while(compteur>0 & securite>0)

    i=ceil(lg*rand(1,1));
    u=tab(i);
    if(rs(u)>0)
        j=ceil(lg*rand(1,1));
        v=tab(j);
        if(rs(v)>0 & v~=u & M(u,v)==0)
            M(v,u)=1;
            M(u,v)=1;
            rs(u)=rs(u)-1;
            rs(v)=rs(v)-1;
            compteur=compteur-2;
            if(rs(u)==0)
                lg=lg-1;
                ind=find(tab==u);
                tab(ind)=[];
            end
            if(rs(v)==0)
                lg=lg-1;
                ind2=find(tab==v);
                tab(ind2)=[];
            end
        end
    end
    securite=securite-1;
```

```
              end

                          % rs peut ne pas  re     z ro , car si le dernier ou il est nn nul ,
                          % il peut y avoir rs=2   celui la et 0 partt .

A=M;
end
```

Listing A.4: Poisson graph

```
%cluster_graph_2

function A=cluster_graph_2(nb_grappes ,nb_par_grappes ,nb_banques_int ,z1 ,z2)

N=nb_grappes*nb_par_grappes ;

B=zeros(N) ;

for i=1:nb_grappes
    indice1=(i-1)*nb_par_grappes+1;
    indice2=(i-1)*nb_par_grappes+nb_par_grappes ;
    C=matrix_2(nb_par_grappes ,z1) ;
    B(indice1:indice2 ,indice1:indice2)=C;
end

%Let's now create the international banks

N2=nb_banques_int*nb_grappes ;
rs = poissrnd(z2 ,1 ,N) ;
t=zeros(1 ,N2) ;

for i=1:nb_grappes
    for j=1:nb_banques_int
        a=(i-1)*nb_banques_int+j ;
        n=(i-1)*nb_par_grappes+j ;
        t(a)=n;
    end
end
compteur=sum(rs) ;
secu=100*z2 ;
lg=N2;


while(compteur>0 & secu>0)
    i=ceil(lg*rand(1 ,1)) ;
    u=t(i) ;
    if(rs(u)>0)
        j=ceil(lg*rand(1,1));
        v=t(j);
        if(rs(v)>0 & v~=u & B(u,v)==0)
            B(u,v)=1;
            B(v,u)=1;
            rs(u)=rs(u)-1;
            rs(v)=rs(v)-1;
            compteur=compteur-2;
            if(rs(u)==0)
                lg=lg-1;
                ind=find(t==u);
                t(ind)=[];
            end
            if(rs(v)==0)
                lg=lg-1;
                ind2=find(t==v);
                t(ind2)=[];
            end
        end
    end
    secu=secu-1;
end
A=B;
end
```
Listing A.5: Cluster graph

Bibliography

[1] J.P. Gleeson, T.R. Hurd, S. Melnik and A. Hackett *Systemic risk in banking without Monte Carlo Simulations* , Advances in Network analysis and its Applications, June 2012

[2] T.R. Hurd and J.P. Gleeson *A framework for analysing contagion in banking networks*, submitted paper, 2011

[3] European Central Bank *Financial Stability Review*, 2012

[4] P. Gai, A. Haldane and S. Kapadia *Complexity, concentration and contagion*, Journal of Monetary Economics, 2011

[5] P. Gai and S. Kapadia *Contagion in financial networks*, Procedings of the Royal Society A, 2010

[6] D.J. Watts *A simple model of global Cascades on random networks*, PNAS 99

[7] A. Demirgüç-Kunt and E. Detriagiache *The determinants of Banking Crises in Developed and Developing Countries*, IMF staff papers 98