2014-05-01

A discrete model for the default risk of inter-banking networks

Mihnea Stefan Andrei
Worcester Polytechnic Institute

A discrete model for the default risk of inter-banking networks

Master's Thesis submitted to
the Faculty of
Worcester Polytechnic Institute
In partial fulfillment of the requirements for the
Degree of Master of Science in
Financial Mathematics

Mihnea Stefan Andrei

Adviser: Stephan Sturm

Department Head: Luca Capogna

May 1, 2014

Acknowledgments

I would like to thank my parents for supporting me throughout my education and I would also like to thank professor Stephan Sturm, without whom this project would not have been possible.

Abstract

During the most recent financial crisis, a myriad of banks defaulted. This scenario encouraged the development of a mathematical model for how default spreads through a system of banks. As we will see, the problem brings together ideas from many fields in Mathematics: Combinatorics, Linear Algebra, Calculus, Statistics and Probabilities. Afterwards, we will turn our attention not only towards implementing the model in MATLAB, but also towards interpreting the results obtained.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Financial Context

During the most recent financial crisis many companies went bankrupt and had to be bailed out by the Government[1]. In the process, the Federal Reserve spent a total of $615B. Albeit the total remaining net that the U.S. Government is due to receive is only $27.1B, some of those companies with financial difficulties even failed to repay their liabilities. This financial crisis caused economic and, especially in Europe, social distress. The latter is a result of job insecurity and rising unemployment. The following figure[2] depicts some of the European countries that have been facing the biggest problems with unemployment.

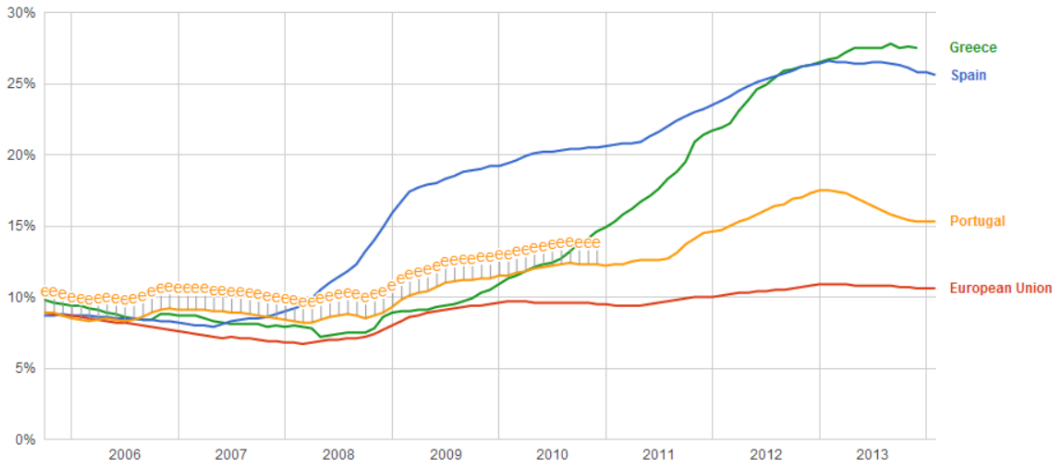


Figure 1: Unemployment by country

It is important to remember however, that the data presented accounts only for the people that could work, but cannot find a job. Hence, for example, people that have certain disabilities or people that are receiving unemployment benefits from their Government are not accounted for. We now realize that, in Spain and Greece, there is less than 3 people that have to provide not only for their young and elder members of their families, but, together, they also have to provide for at least another person that could not find a job. The consequences of such high unemployment for a long period of time are devastating. For example, the young cannot get the experience necessary when they get out of school or experienced workers that have been unemployed for a long period of time start loosing their skills.

Although the rest is history, it is important to try to learn from the mistakes of the past. But what are those mistakes? Many reasons can be found for the financial crisis, but an unmistakable one is the strong connections that institutions have with each other. There are more companies, conducting business faster and faster, mainly

[1]For a complete list visit http://projects.propublica.org/bailout/list
[2]Data obtained from Eurostat

due to the advances in technology. It is those advances that brought not only individuals closer, but also country's economies closer. As we become more and more aware of the importance that connections have in a system, we would like to be able to investigate which ones do not facilitate the spread of default.

One way in which we could do this is by studying how contagion spreads through a banking system. In this paper, we will develop a discrete model that characterizes the spread of contagion by finding the expected time until the first bank defaults, the probabilities that each bank will default and some conditional probabilities.

1.2 The Model

We consider that the chances for a certain bank to default depend on the capital that it retains and on the liabilities and assets that it has with respect to other institutions.

Now, that we have established the 2 most important criteria for the default of a bank, the next question is how we would mathematically represent the assets and the liabilities that each bank might have to other banks. One way is to use graphs. We consider the vertices of a graph to be our banks. Each vertex has a weight equal to the capital that it retains. We draw a directed edge from vertex i to vertex j if the bank with a starting capital of i (for the rest of the paper we will refer to it simply as bank i) has a liability to bank j. This edge has a weight equal to the amount of money that bank i has to pay to bank j.

2 Discrete Time Model

2.1 Two Banks

Before venturing any further into thinking about our problem in the general case, for any number of banks, we will first present the problem when there are only 2 banks in a financial system. The only configuration of 2 banks that interests us is the one in which one of them has liabilities to the other one.

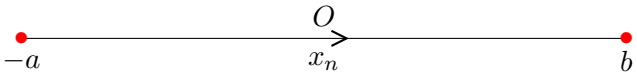


Figure 2: System with two banks

In the above picture, a and b are the starting capitals for the two banks, and WLOG, assume that the bank with a capital of a has liabilities to pay to the other bank. Let us use a weight on the edge between the two banks equal to the value of a random walk at time n:

$$x_n = \sum_{i=1}^{n} y_i \tag{1}$$

Here

$$y_i = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ -1 & \text{with probability } \frac{1}{2} \end{cases}$$

For example, if $x_n > 0$ then the bank with a starting capital of a owes x_n to the bank with starting capital b. Meanwhile, if $x_n < 0$, the bank with a starting capital of b owes money to the second bank. One way to think about this configuration is to consider the edge of the graph between the two banks to be the real line. This way, in order to be consistent with this new framework, we can relabel the bank with a starting capital of a with $-a$. Also, our random walk starts at 0 and, as time passes by, it will move either to the left, closer to $-a$, or to the right, closer to b. With this set-up, the first bank will default when $x_n = -a$ and the second bank will default when $x_n = b$. Hence, we are interested in studying when is the first time when one of the 2 banks defaults and what are the chances that they will default. The first time when either of the banks defaults can be represented as a stopping time defined in the following way:

$$\tau = \inf\{n | x_n = -a \text{ or } x_n = b\} \tag{2}$$

Moreover, consider $\mathcal{F}_n$ to be the filtration associated with our random walk x_n.

We are interested in finding $E[\tau]$, $P(x_\tau = -a)$ and $P(x_\tau = b)$. However, before finding those values, we will first prove two useful lemmas.

Lemma 1. *y_i^2 and $1_{\{\tau > i-1\}}$ are independent random variables for any i.*

Proof. Since τ is a stopping time, from the definition, we know that $\{\tau = j\} \in \mathcal{F}_j$ for any j. Hence, by using the closure under union and the closure under complementarity for a sigma algebra, we obtain that

$$\bigcup_{j=1}^{i-1} \{\tau = j\} \in \mathcal{F}_{i-1} \Leftrightarrow \{\tau \leq i-1\} \in \mathcal{F}_{i-1} \Leftrightarrow \{\tau > i-1\} \in \mathcal{F}_{i-1}$$

But, since $y_i \in \mathcal{F}_i \Rightarrow y_i^2 \in \mathcal{F}_i$ and since y_i are independent for any i, we conclude that y_i^2 and $1_{\{\tau > i-1\}}$ are independent random variables for any i. □

Lemma 2. *If $j > i$ then y_j and $y_i 1_{\{\tau > i-1\}} 1_{\{\tau > j-1\}}$ are independent random variables.*

Proof. Just like in the proof for the lemma presented above, we can show that $\{\tau > k-1\} \in \mathcal{F}_{k-1}$ for any $k \geq 1$. Therefore, $\{\tau > i-1\} \in \mathcal{F}_{i-1}$ and $\{\tau > j-1\} \in \mathcal{F}_{j-1}$. Since $y_i \in \mathcal{F}_i$, we conclude that $y_i 1_{\{\tau > i-1\}} \in \mathcal{F}_i$. But since $i \leq j-1 \Rightarrow \mathcal{F}_i \subset \mathcal{F}_{j-1}$ we observe that $y_i 1_{\{\tau > i-1\}} 1_{\{\tau > j-1\}} \in \mathcal{F}_{j-1}$. The last step is to remember that $y_j \in \mathcal{F}_j$ and $\mathcal{F}_{j-1}$ are independent. □

Now that we proved those useful lemmas, we are able to find $E[\tau]$. Using definition (1) for $n = \tau$, we obtain that[3]

$$x_\tau = \sum_{i=1}^{\tau} y_i = \sum_{i=1}^{\infty} y_i 1_{\{\tau > i-1\}}$$

[3]Karl Sigman

The second equality can easily be verified by starting from the right hand side and proving the left hand side. Next, if we square the equation obtained above and if we take the expected value of both sides, we obtain that

$$E[{x_\tau}^2] = E\left[\left(\sum_{i=1}^{\infty} y_i 1_{\{\tau>i-1\}})\right)^2\right] =$$
$$= \sum_{i=1}^{\infty} E\left[y_i^2 1_{\{\tau>i-1\}}\right] + \sum_{i,j=1,i\neq j}^{\infty} E\left[y_i y_j 1_{\{\tau>i-1\}} 1_{\{\tau>j-1\}}\right]$$

Using **Lemma 2** and the fact that y_i are independent and identically distributed with $E[y_i] = 1 \cdot \frac{1}{2} - 1 \cdot \frac{1}{2} = 0$, we can conclude that $E\left[y_i y_j 1_{\{\tau>i-1\}} 1_{\{\tau>j-1\}}\right] = E[y_j] E\left[y_i 1_{\{\tau>i-1\}} 1_{\{\tau>j-1\}}\right] = 0$ for any $i \neq j$.

Hence, the double sum in the right hand side of the equation that was presented above is equal to 0. Furthermore, by using **Lemma 1**, we obtain that:

$$E\left[{x_\tau}^2\right] = \sum_{i=1}^{\infty} E\left[y_i^2 1_{\{\tau>i-1\}}\right] = \sum_{i=1}^{\infty} E\left[y_i^2\right] E\left[1_{\{\tau>i-1\}}\right] = \sum_{i=1}^{\infty} E\left[y_i^2\right] P(\tau > i-1)$$

But $E[y_i^2] = 1^2 \frac{1}{2} + (-1)^2 \frac{1}{2} = 1$ for any i. Therefore, we found that

$$E[{x_\tau}^2] = \sum_{i=0}^{\infty} P(\tau > i) = E[\tau].$$

Remark 1. *We were able to interchange the series with the expectation because* $x_n = \sum_{i=1}^{n} \left(y_i 1_{\{\tau>i-1\}}\right)^2 \geq 0$ *and* x_n *is monotone increasing.*

For two banks, we have found so far that $E[{x_\tau}^2] = E[\tau]$.

Before we continue in our quest of finding $P(x_\tau = -a)$ and $P(x_\tau = b)$, let us make the observation that $E[x_n] = \sum_{i=1}^{n} E[y_i] = 0$ and let us also prove the following useful result:

Lemma 3. *If* $E[x_n] = 0$ *then* $E[x_\tau] = 0$.

Proof. Using the fact that $P(\tau < \infty) = 1$, we observe that $x_\tau = \lim_{n\to\infty} x_{\tau\wedge n}$. By taking expectation, we arrive at $E[x_\tau] = E\left[\lim_{n\to\infty} x_{\tau\wedge n}\right]$. But since we also have that $x_{\tau\wedge n} \leq \tau$, we obtain

$$E[x_\tau] = \lim_{n\to\infty} E[x_{\tau\wedge n}]$$

Therefore, we would have to show that $\lim_{n\to\infty} E[x_{\tau\wedge n}] = 0$.

Let us make use of the indicator function: $E[x_{\tau\wedge n}] = E\left[x_\tau 1_{\{\tau<n\}} + x_n 1_{\{\tau\geq n\}}\right]$. Because $-a \leq x_n \leq b$, we conclude that $-a 1_{\{\tau\geq n\}} \leq x_n 1_{\{\tau\geq n\}} \leq b 1_{\{\tau\geq n\}}$. By taking expectation, we obtain that $-aP(\tau \geq n) \leq E\left[x_n 1_{\{\tau\geq n\}}\right] \leq bP(\tau \geq n)$. Now,

by taking the limit as n goes to infinity and using the squeeze law, we conclude that $\lim_{n\to\infty} E\left[x_n 1_{\{\tau\geq n\}}\right] = 0$

Let us turn our attention to $E\left[x_\tau 1_{\{\tau<n\}}\right]$. From the definition of x_τ, we obtain that $x_\tau = \sum_{i=1}^{\infty} x_i 1_{\{\tau=i\}}$. Therefore

$$E\left[x_\tau 1_{\{\tau<n\}}\right] = E\left[\sum_{i=1}^{\infty} x_i 1_{\{\tau=i\}} 1_{\{i<n\}}\right] = E\left[\sum_{i=1}^{n-1} x_i 1_{\{\tau=i\}}\right] = \sum_{i=1}^{n-1} E[x_i] P(\tau = i) = 0$$

Hence, we showed that $\lim_{n\to\infty} E[x_{\tau\wedge n}] = 0$, which concludes our proof.

□

Theorem 1. *$P(x_\tau = -a) = \frac{b}{a+b}$ and $P(x_\tau = b) = \frac{a}{a+b}$*

Proof. From the way in which we defined our stopping time in (2), we realize that we can write $x_\tau = -a1_{\{x_\tau=-a\}} + b1_{\{x_\tau=b\}}$. By taking expectation and by using **Lemma 3**, we obtain that $0 = E[x_\tau] = -aP(x_\tau = -a) + bP(x_\tau = b)$. However, since $P(x_\tau = -a) + P(x_\tau = b) = 1$ we obtain that $P(x_\tau = -a) = \frac{b}{a+b}$ and $P(x_\tau = b) = \frac{a}{a+b}$. □

2.2 Generalization of the two bank model

For the general case, we will use the same graph representation as before. This time, each edge in the graph will be weighted by a different random walk. The picture below illustrates a three bank system with three connections (i.e. each bank has liabilities or assets to all the others).

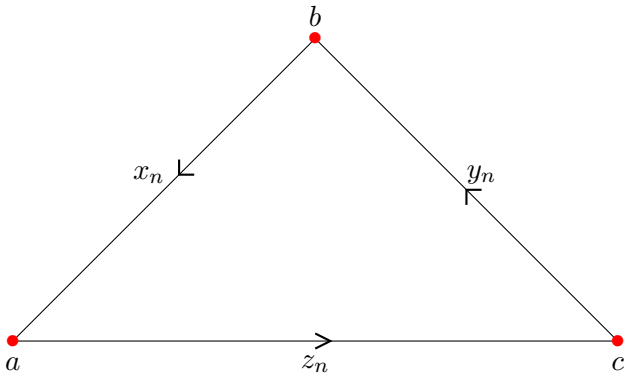


Figure 3: System of 3 banks with 3 connections

Using the configuration presented in the picture and knowing that a bank defaults on its debt when it doesn't have sufficient capital to pay all the liabilities that it has, we know that our system will not have a bank that defaults as long as the following system of inequalities is satisfied:

$$\begin{cases} a + x_n - z_n > 0 \\ b - x_n + y_n > 0 \\ c - y_n + z_n > 0 \end{cases}$$

In general, any system of banks will not default as long as we have that $AW_n > \vec{0}$, where A is a mxn matrix with $m =$ number of banks in the system and n=number of edges in the system $+ 1$, $W_n = \begin{pmatrix} 1 & x_n & y_n & . & . \end{pmatrix}^T$ and the inequality is component-wise. In the example showed in the above picture,

$$A = \begin{pmatrix} a & 1 & 0 & -1 \\ b & -1 & 1 & 0 \\ c & 0 & -1 & 1 \end{pmatrix} \text{ and}$$

$$W_n = \begin{pmatrix} 1 \\ x_n \\ y_n \\ z_n \end{pmatrix}$$

.

Hence, we have to find the first time at which at least one component of $AW_n \leq 0$ and, just like in the previous section, let it be τ. We can think of AW_n as a multidimensional random walk in $\mathcal{R}^m$ and let us consider the points for which $AW_n > 0$ to be transition points and the points for which $AW_n \leq 0$ to be absorption points. Therefore, AW_n is an absorbed Markov process.

Observation 1. *If m is the number of banks in our system and if S is the sum of the capitals that each bank has, then the number of transition points is at most $\binom{S-1}{m-1}$.*

Proof. We first notice that if we add all the left hand side quantities in our inequalities from one of our linear system, the result will just be S. This is because what is a liability for a bank becomes an asset for the other. Let us denote by $(AW_n)_i$ the i^{th} component of the vector AW_n. In this new framework, we are trying to find the number of positive integer solutions to the equation $\sum_{i=1}^{m}(AW_n)_i = S$. But the number of positive integer solutions to m independent variables that sum up to S is $\binom{S-1}{m-1}$. However, our $(AW_n)_i$ are not pairwise independent, since a particular value for some might uniquely determine other components of AW_n.

□

Now that we know that the number of transition states is bounded, we move on to finding some information about the number of absorption states.

Observation 2. *If m is the number of banks in our system, then the number of absorption states is at most $2^m - 2$.*

Proof. The number of absorption states is at most equal to the number of ways in which we can choose one component to be non-positive+the number of ways in which we can choose 2 components to be non-positive+...+ the number of ways in which we can choose $m-1$ components to be non-positive. This, in turn, is equal to $\binom{m}{1} + \binom{m}{2} + ... + \binom{m}{m-1} = (1+1)^m - \binom{m}{0} - \binom{m}{m} = 2^m - 2$. □

Let $P = \begin{pmatrix} T & \vec{r}_1 & \vec{r}_2 & ... & \vec{r}_{n_a} \\ 0 & & & I & \end{pmatrix}$ be the transition matrix that corresponds to the random walk AW_n, where T is the transition matrix for the transition states and r_i are vectors with probabilities of arriving in the i^{th} absorption states. More specifically,

$$T_{ij} = P(AW_n \text{ reaches the } j^{th} \text{ transition point from the } i^{th} \text{ transition point})$$

$$(\vec{r_i})_j = P(AW_n \text{ reaches the } i^{th}\text{absorption point from the } j^{th} \text{ transition point})$$

Just as before, $(\vec{r_i})_j$ denotes the j^{th} component of $\vec{r_i}$. Moreover, since P is stochastic, the sum of the probabilities across each row has to be 1. We obtain that $T\vec{1}+\sum_{i=1}^{n_a} r_i = \vec{1}$, where $\vec{1} = \begin{pmatrix} 1 & 1 & ... & 1 \end{pmatrix}^T$. Furthermore, let us denote by $R = \begin{pmatrix} r_1 & r_2 & ... & r_{2^m-2} \end{pmatrix}$ the absorption matrix. Hence, we have that

$$T\vec{1} + R\vec{1} = \vec{1} \tag{3}$$

.

Before we can continue on our quest of modeling default in a system of banks, we will have to introduce a new concept[4].

Definition 1. *The distribution of the time τ until absorption for a random walk with transition matrix P and initial probability distribution $\vec{p}$ is called a PH distribution.*

Remark 2. *In our model:*

(i) The random walk for which we want to use this concept is AW_n

(ii) $\vec{p} = \begin{pmatrix} 1 & 0 & 0 & ... & 0 \end{pmatrix}^T$

The following theorem gives the distribution of a PH random variable.

Theorem 2. *Let τ follow a PH($\vec{p}$,T) distribution. Then:*

(i) $P(\tau = k) = \vec{p}(T^{k-1} - T^k)\vec{1}$;

(ii) $F_\tau(k) = P(\tau \leq k) = \vec{p}(I - T^k)\vec{1}$

Proof. We will first prove (ii). Since AW_n is an absorbed Markov Chain, the chance that the time to default is smaller than some integer k is the same as the chance that the random walk AW_k will be absorbed. Now, by conditioning on all the possible starting points for our random walk, we obtain:

$$\begin{aligned} F_\tau(k) = P(\tau \leq k) = P(AW_k \leq 0) = \\ = \sum_{i=0}^{k} P(AW_k \leq 0 | AW_0 = v_i) P(AW_0 = v_i) = \\ = \vec{p}(I + T + T^2 + ... + T^{k-1})R\vec{1} = \vec{p}(I - T^k)(I - T)^{-1}R\vec{1} \end{aligned}$$

[4]Latouche and Ramaswami (1999)

By using equation (3), we can deduce that

$$F_\tau(k) = \vec{p}(I - T^k)(I - T)^{-1}(\vec{1} - T\vec{1}) = \vec{p}(I - T^k)(I - T)^{-1}(I - T)\vec{1} = \vec{p}(I - T^k)\vec{1}$$

Please note that **Proposition 2** assures us that $(I - T)$ is invertible. Sub-point (i) follows more easily:

$$\begin{aligned} f_\tau(k) = P(\tau = k) = F(\tau \leq k) - F(\tau \leq k-1) = \\ = \vec{p}(I - T^k)\vec{1} - \vec{p}(I - T^{k-1})\vec{1} = \vec{p}(T^{k-1} - T^k)\vec{1} \end{aligned}$$

□

Now that we know the distribution for the time to default random variable, we can find $E[\tau]$ and $P(\text{default in } \alpha)$, the latter probability been just the likelihood that our random walk will be absorbed in state α.

Proposition 1. *Let τ be the time to default random variable. Then τ has a PH distribution with transition matrix T, initial probability distribution given by $\vec{p}$ and:*

(i) $E[\tau] = \vec{p}(I - T)^{-1}\vec{1}$;

(ii) $P(AW_n = \vec{r_\alpha}) = \vec{p}(I - T)^{-1}\vec{r_\alpha}$

Before we can start proving this proposition, we will need to prove the following 2 results.

Theorem 3. *The probability that an absorbed Markov chain will eventually be absorbed is 1.*

Proof. We have to show that $\lim_{n\to\infty} T^n = 0$. Let us take t_j a transition point. Let m_j be the minimum number of steps that it takes for our process to reach an absorbing state from t_j.Finally, consider p_j to be the probability that starting from t_j the process will not reach an absorbing state in m_j and let $p = \max_j p_j$ and $m = \max_j m_j$. From the way in which we constructed the notations, we know that we can reach an absorbing state from point t_j in m_j steps. Hence, $p_j < 1$ and we know that the probability of not been absorbed in m steps is at most p. Using the fact that AW_n has independent time increments, we can also conclude that the probability of not been absorbed in $2m$ steps is at most p^2. In general, the probability of not been absorbed in km steps is at most p^k. If we let $k \to \infty$ we obtain that $\lim_{n\to\infty} T^n = 0$. □

Now that we have proved this theorem, we are ready to show that $I - T$ is invertible.

Proposition 2. *If T is a transition matrix, then $I - T$ is invertible.*

Proof. Let us start from the equation $(I-T)\vec{x} = 0 \Leftrightarrow \vec{x} = T\vec{x}$. If we repeatedly multiply the last equation by T to the left, we will obtain that $\vec{x} = T^k\vec{x}$ for any integer k. However, by letting $k \to \infty$ and by using **Theorem 3**, we obtain that $\vec{x} = 0$. Therefore, $I - T$ is invertible and let N be its inverse. We know that the following identity holds: $(I-T)(I+T+T^2+...+T^k) = I-T^{k+1}$ for any positive integer k. By multiplying both sides with N to the left, we obtain that $I+T+T^2+...+T^k = N(I-T^{k+1})$. By letting once again $k \to \infty$ and by using **Theorem 3** we conclude that $(I-T)^{-1} = N = \sum_{i=0}^{\infty} T^i$. □

Since we have shown that $I-T$ is invertible, we can start giving the proof for **Proposition 1**.

Proof. The proof uses **Theorem 2** as follows.

(i) By applying the definition of expectation, we obtain that

$$E[\tau] = \sum_{k=1}^{\infty} k\vec{p}\left(T^{k-1} - T^k\right)\vec{1} = \vec{p}\left(\sum_{k=1}^{\infty} kT^{k-1} - kT^k\right)\vec{1} = \vec{p}\left(\sum_{k=1}^{\infty} T^k\right)\vec{1}$$

Using **Proposition 2**, we can conclude that $E[\tau] = \vec{p}(I-T)^{-1}\vec{1}$.

(ii) By conditioning on all the possible values that the time to default random variable can take, we obtain

$$P(\text{default in } \alpha) = \sum_{k=0}^{\infty} P(\text{default in } \alpha, \tau = k) = \sum_{k=1}^{\infty} \vec{p}T^{k-1}r_\alpha$$

The last equality holds because in order for the system to default in the absorption case α after k steps, it has to have $k-1$ transition points and the last step has to absorb the random walk. Hence, $P(\text{default in } \alpha) = \vec{p}(\sum_{k=0}^{\infty} T^k)r_\alpha$ and by using again **Proposition 2** we conclude that $P(\text{default in } \alpha) = \vec{p}(I-T)^{-1}r_\alpha$.

□

3 Implementation in MATLAB

We would like to be able to implement the ideas presented in the previous section in order to be able to study how different banking networks react under the spread of contagion. Also, for each such network, we would like to see the impact that different starting capitals has. As we have seen in the previous section, we have managed to find both the expected time until the first bank defaults and the probability that we will end up in a certain defaulting scenario. Hence, in order to find the probability that a bank will default, we just have to add all the probabilities of the outcomes in which the bank's capital becomes non-positive. However, as **Proposition 1** suggests, before doing so, we have to be able to compute the transition matrix and the absorption matrix. But in order to be able to fill in those matrices, we have to know all the transition points and all the absorption points for our multidimensional random walk. **Observation 1** and **Observation 2** from the previous section suggest that the two matrices might become large for big systems of banks with big starting capitals, which, in turn, will affect drastically the run time.

Each system of banks has associated with it a system of linear inequalities, which, in turn, have an augmented matrix, denoted here by A. This time, for the purposes of implementation, we define $W_n = \begin{pmatrix} x_n & y_n & . & . & z_n \end{pmatrix}^T$ and A to be the matrix that

has as entries only the coefficients of the random walks. Using the example of 3 banks with 3 connections given in the previous section, this time

$$A = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}$$

Moreover, from the definition of our random walks in (2), we realize that $x_{n+1} - x_n = \pm 1$. Hence, we obtain that

$$AW_{n+1} - AW_n = A(W_{n+1} - W_n) = A \begin{pmatrix} \pm 1 \\ \pm 1 \\ . \\ . \\ \pm 1 \end{pmatrix} \tag{4}$$

For simplicity, let us consider a network that has 4 banks. We can think of our linear system as a multidimensional random walk that starts at $\begin{pmatrix} a & b & c & d \end{pmatrix}^T$ and can only move in the directions given by (4). This geometric interpretation gives us an idea of implementing the model. We can start from the origin and we can add to it the step vectors found in (4) until one of the components of the result becomes non-positive.

Algorithm 1. *Finding The Step Vectors*

```
for i=0:1:1
  for j=0:1:1
    for k=0:1:1
      for l=0:1:1
        v(:,counter)=[(-1)^i; (-1)^j; (-1)^k; (-1)^l];
        SV(:,counter)=A*v(:,counter);
        counter=counter+1;
      end
    end
  end
end
```

Observation 3. *Since we are adding the step vectors to the origin, we are only interested in those that are linearly independent, since the others can be obtained as a linear combination of the independent vectors.*

Algorithm 2. *Linearly Independent Step Vectors*

```
rrefSV=rref(SV);
rows=size(SV,1);
columns=size(SV,2);
counter=1;
counter1=1;
max=0;
```

```
for i=1:1:columns
  flag=0;
  for j=1:1:rows
    if rrefSV(j,i)==1
      flag=1;
      k=j;
    end
  end
  flag1=-1;
  if flag==1
    flag1=1;
    for j=1:1:rows
      if k ≠ j
        if rrefSV(j,i)≠ 0
          flag1=0;
        end
      end
    end
  end
  if flag1==1
    SVli(:,counter)=SV(:,i);
    counter=counter+1;
  else
    if norm(SV(:,i),1)>max
      max=norm(SV(:,i),1);
    end
  end
end
```

As we can see in the algorithm presented above, we are row reducing the matrix that has as columns the step vectors obtained from **Algorithm 1**. By picking those columns from the original matrix that have pivot positions in the row reduced one, we obtain a set of linearly independent vectors. Moreover, we notice that this algorithm also finds the step vector that has the biggest norm. As we will see in the following algorithm, we need to know this since we have to impose a condition on when to stop searching for transition and absorption points. Since we have to keep on searching until the condition is not met anymore, we decide to use while loops.

Algorithm 3. *Searching for Points*

```
W=zeros(banks,1);
v1=SVli(:,1);
v2=SVli(:,2);
v3=SVli(:,3);
T=zeros(1,1);
```

```
vstop=-max*ones(4,1);
Abs=zeros(banks,1);
W(:,1)=[a;b;c;d];
j=2;
k=0;
l=0;
m=0;
while (W(:,1)+k*v1+l*v2+m*v3>vstop)
   while (W(:,1)+k*v1+l*v2+m*v3>vstop)
      while (W(:,1)+k*v1+l*v2+m*v3>vstop)
         W(:,j)=W(:,1)+k*v1+l*v2+m*v3;
        j=j+1;
        m=m+1;
      end
   l=l+1;
   m=0;
   end
k=k+1;
m=0;
l=0;
end
```

Notice that we imposed the condition $W(:,1)+k*v1+l*v2+m*v3 > vstop$ instead of $W(:,1)+k*v1+l*v2+m*v3 > 0$. This is because, in the latter situation it is possible that the code finds a negative coordinate for a certain combination of k, l, m, albeit, on the next iteration, that same coordinate would become positive. In order to make sure that this will not happen, we impose the condition that has *vstop* (the step vector with the biggest norm).

Observation 4. *Since we have to loop through all possible k, l, m, we would have to construct, in a similar manner, while loops that take into account all the possible orderings of those 3 parameters and the fact that they can also be negative.*

Once we construct the matrix W with columns the points of our multidimensional random walk can reach, we are ready to find the transition points (those that have all their coordinates positive).

Algorithm 4. *Transition Points*

```
WT=zeros(banks,1);
counter=1;
for i=1:1:(j-1)
   if W(:,i)>0
      WT(:,counter)=W(:,i);
      counter=counter+1;
   end
end
```

In order to find the absorption points, we have to choose those columns in W that have at least one negative coordinate and they have to be one step away from a transition point.

Algorithm 5. *Absorption Points*

```
Abs=zeros(banks,1);
sizeSV=size(SV,2);
counter=1;
for k=1:1:sizeSV
  for l=1:1:(j-1)
    if (W(:,l)+SV(:,k)>0)
      counter=counter;
    else
      Abs(:,counter)=W(:,l)+SV(:,k);
      counter=counter+1;
    end
  end
end
```

However, the points found by the presented algorithms might not be unique. Thus, we need to create a different matrix with the unique points

Algorithm 6. *Unique Transition Points*

```
WU=zeros(banks,1);
WU(:,1)=WT(:,1);
sizeWT=size(WT,2);
su=1;
  for p=2:1:sizeWT
  flag=1;
    for q=1:1:su
      if WT(:,p)==WU(:,q)
        flag=0;
      end
    end
    if flag==1
      su=su+1;
      WU(:,su)=WT(:,p);
    end
  end
```

In a similar manner, we can find the unique absorption points. Now that we finally have both our unique transition points and our unique absorption points, we can start constructing our transition and absorption matrices. We notice that

$$T(i,i) = \frac{\text{number of step vectors equal to } \vec{0}}{2^{edges}}$$

Moreover, if two transition points (let them be the p^{th} and q^{th} point in WU) are one step away from each other, then

$$T(p,q) = T(q,p) = \frac{\text{number of ways of moving from point p to point q}}{2^{\text{edges}}}$$

.

Algorithm 7. *Transition Matrix Construction*

```
zerov=zeros(banks,1);
count=0;
for k=1:1:sizeSV
  if(SV(:,k)==zerov)
    count=count+1;
  end
end
T(1,1) = count/(2^edges);
for p=1:1:su
  for q=1:1:su
    count1=0;
    for k=1:1:sizeSV
      if ((WU(:,p)-WU(:,q)==SV(:,k))&(p≠q))
        count1=count1+1;
      end
    end
    T(p,q) = count1/(2^edges);
    T(q,p) = count1/(2^edges);
  end
  T(p,p) = count/(2^edges);
end
```

For the absorption matrix, we have to make sure that the absorption point found is only one step away from a transition point. Once this is confirmed, we set

$$R(q,p) = \frac{\text{number of ways of moving from point q to p}}{2^{edges}}$$

Algorithm 8. *Absorption Matrix Construction*

```
siu=size(AbsU,2);
R=zeros(su,siu);
for p=1:1:siu
  for q=1:1:su
  count=0;
    for k=1:1:sizeSV
      if (WU(:,q)-AbsU(:,p)==SV(:,k))
        count=count+1;
      end
```

```
    end
  R(q,p) = count/2^edges;
  end
end
```

Finally, by using **Proposition 1** from the previous section, we can easily find the expectation and the probability of default in each scenario.

Algorithm 9. *Expectation and Probabilities of Default in each Scenario*

```
tauinv=(tau/(eye(s,s)-T));
for j=1:1:siu
  str=['The chance of default in the scenario ', mat2str(AbsU(:,j)),
    ' is ', num2str(tauinv*R(:,j))];
  disp(str);
  sum1=sum1+(tauinv*R(:,j));
end
```

Now, we have everything we need to compute the chances that each bank, in a given network, will default. In order to do so, we first have to know how many banks have defaulted in any given absorption scenario. Afterwards, as mentioned at the beginning of this section, we would just have to add the probabilities of all the outcomes in which the bank (or banks) that interests (interest) us defaults (default).

Algorithm 10. *Probability of Bank(Banks) Defaulting*

```
pbank=zeros(banks,banks);
AbsCond=AbsU;
count=zeros(siu,1);
for k=1:1:siu
  for j=1:1:banks
    if (AbsU(j,k)≤0)
      count(k,1)=count(k,1)+1;
    end
  end
end
for k=1:1:siu
  if count(k,1)==1
    for j=1:1:banks
      if AbsU(j,k)≤0
        pbank(j,j)=pbank(j,j)+tauinv*R(:,k);
      end
    end
  else
    if count(k,1)==2
      for j=1:1:banks
```

```
        for l=(j+1):1:banks
          if ((AbsU(l,k)≤0)&&(AbsU(j,k)≤0))
            pbank(l,j)=pbank(l,j)+tauinv*R(:,k);
            pbank(j,l)=pbank(l,j);
          end
        end
      end
    end
  end
end
disp('Probability of default matrix:');
disp(pbank);
```

We can also compute conditional probabilities, by assuming that, for example, if a bank defaults, its neighboring banks will have to equally pay the loss recorded throughout the bank's lifetime. It is easier to implement this assumption when there is only one bank defaulting, as opposed to when there are 2. Let us deal with the easier case first. The formula in this scenario is

$$\text{new capital} = \text{old capital} - \frac{\text{initial capital of defaulting bank}}{\text{number of neighboring banks}} - \\ - \frac{\text{ending capital of defaulting bank}}{\text{number of neighboring banks}} \quad (5)$$

Algorithm 11. *Capitals Adjustment if 1 Bank Defaults*

```
for k=1:1:siu
  if (count (k,1)==1)
    for l=1:1:banks
      if (AbsU(l,k)≤0)
        [v,counter]=adjacentVertecies(B,l);
        for j=1:1:counter-1
          AbsCond(v(j,1),k) = AbsCond(v(j,1),k) - (WU(l,1)-AbsCond(l,k))/(degree(B,l));
        end
      end
    end
  end
end
```

We also note that we have used another function, *adjacentVertecies* that takes as argument the incidence matrix of a graph and a vertex and returns a vector that contains all the neighboring vertices of the specified one and the number of neighbors. The implementation of this function and of the whole code can be found in the *Appendix*.

Now, let us focus the attention on the situation in which 2 banks default at the same time. In this scenario, the formula (5) might be different. For example, if the defaulting banks have a common neighbor and they also have an edge between them, then its new capital will be:

$$\text{new capital} = \text{old capital} - \frac{\text{(initial-ending) capital } 1^{st} \text{ defaulting bank}}{\text{number of neighboring banks} - 1} - \\ - \frac{\text{(initial-ending) capital } 2^{nd} \text{ defaulting bank}}{\text{number of neighboring banks} - 1}$$

For the scenario in which there is no edge between the 2 defaulting banks, we simply do not have to subtract 1 at the denominator.

Algorithm 12. *Capitals Adjustment if 2 Banks Default*

if count(k,1)==2
  for l=1:1:banks
    for j=l:1:banks
      if (AbsU(l,k)≤0)
        if (AbsU(j,k)≤0)
          if (l≠j)
          [vl,counter1]=adjacentVertecies(B,l);
          [vj,counter2]=adjacentVertecies(B,j);
          lasteqvl=0;
            for p=1:1:counter1-1
              for q=1:1:counter2-1
                if ((vl(p,1)==vj(q,1)))
                  if B(l,j)==1
                    lasteqvl=vl(p,1);
                    $AbsCond(lasteqvl, k) = AbsU(lasteqvl, k) - \\ - \frac{WU(l,1)+WU(j,1)-AbsU(l,k)-AbsU(j,k)}{degree(B,l)-1};$
                  else
                    lasteqvl=vj(q,1);
                    $AbsCond(lasteqvl, k) = AbsU(lasteqvl, k) - \frac{WU(l,1)-AbsU(l,k)}{degree(B,l)};$
                    inter=AbsCond(lasteqvl,k);
                    $AbsCond(lasteqvl, k) = inter - \frac{WU(j,1)-AbsU(j,k)}{degree(B,j))};$
                  end
                else
                  if ((lasteqvl≠vl(p,1))&&(B(l,j)==1) && (vl(p,1)≠j))
                    $AbsCond(vl(p, 1), k) = AbsU(vl(p, 1), k) - \frac{WU(l,1)-AbsU(l,k)}{degree(B,l)-1};$
                  end
                  if ((lasteqvl≠vj(q,1))&&(B(l,j)==1) && (vl(p,1)≠j))
                    $AbsCond(vj(q, 1), k) = AbsU(vj(q, 1), k) - \frac{WU(j,1)-AbsU(j,k)}{degree(B,j)-1};$
                  end
                  if ((lasteqvl≠vl(p,1))&&(B(l,j)==0) && (vl(p,1)≠j))

$AbsCond(vl(p,1),k) = AbsU(vl(p,1),k) - \frac{WU(l,1)-AbsU(l,k)}{degree(B,l)}$;
end
if ((lasteqvl≠vj(q,1))&&(B(l,j)==0) && (vl(p,1)≠j))
$AbsCond(vj(q,1),k) = AbsU(vj(q,1),k) - \frac{WU(j,1)-AbsU(j,k)}{degree(B,j)}$;
end
end
end
end
end
end
end
end
end

4 Results and Simulations

In this section, we will present some results obtained using the code explained in the previous section. But before we delve into them, let us first look at the limitations of the code, as they impact its running speed.

Remark 3. *Limitations of the code and possible improvements*

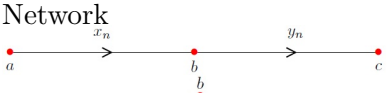
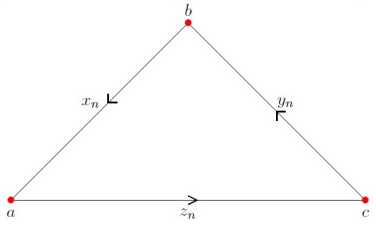
- *The code uses an exhaustive method to find the absorption and transition points, which is time consuming, especially as the initial capitals and number of banks increase.*
- *Vectorizing the code would increase speed.*

We are concerned with speed since, on a typical computer, the above code would have to run for around 1 week in order to simulate a specific four bank network with initial capitals for each bank ranging from 1 to 10, in increments of 1. This is also the reason why most of our results are obtained for networks with 3 banks. As the following table depicts, there are only 2 different possible such networks. This is because if we would reduce the number of edges to 1, we would obtain a vertex that is not connected to any of the others and, thus, we would deal with a two bank network.

For the first test, all the starting capitals ranged from 1 to 20 in increments of 1. For each such initial distribution of capitals, the expected time to default was computed. The figure presented in the first column of the following table represents the average of all the expected times to default. By taking the standard deviation of those expected times to default, we obtain the figures in the second column:

Using the table, the first observation that we can make is one that goes in tandem with our intuition.

Table 1: Expected Time of Default for 3 Bank Networks

Mean Time Default	Std Time Default	Network
48.8452	40.4897	
34.1891	27.1418	

Observation 5. 3 *Bank Network conclusions*

- *The more connected the network, the smaller the time to default.*
- *The more connected the network, the less sensitive it is to changes in capitals.*

The second observation follows from the standard deviation figures. Since the standard deviation of the network presented first in the table is bigger than that of the second network, the expected time of default for the former, when the initial capitals are small, is smaller than those of the latter. In the same way, when the initial starting capitals get bigger, the expected value of the default time for the network that has 2 edges is bigger than that of the second network.

Next, let us try to make a plot of the sum of the capitals that banks have versus the expected time to default. Since there are multiple ways of obtaining a given sum using 3 integers, we would have to create a vector that contains the unique sums. Moreover, we would like to average the expected time of default in the scenarios in which the sum of capitals are equal.

Algorithm 13. *Plotting Sum of Capitals Versus Expected Default Time*

```
counterexpect=0;
for a=1:1:20
  for b=1:1:20
    for c=1:1:20
      sumCap(counterexpect,1)=a+b+c;
      counterexpect=counterexpect+1;
    end
  end
end
sumCapU(1,1)=sumCap(1,1);
counter=1;
for i=2:1:counterexpect-1
  flag=0;
  for j=1:1:counter
```

```
    if (sumCap(i,1)==sumCapU(j,1))
      flag=1;
    end
  end
  if flag==0
    sumCapU(counter+1,1)=sumCap(i);
    counter=counter+1;
  end
end
countSum=zeros(counter,1);
expectSum=zeros(counter,1);
for i=1:1:counterexpect-1
  for j=1:1:counter
    if (sumCap(i,1)==sumCapU(j,1))
      countSum(j,1)=countSum(j,1)+1;
      expectSum(j,1)=expectSum(j,1)+expect(1,i);
    end
  end
end
for i=1:1:counter
  if countSum(i,1)>0
  expectSum(i,1)=expectSum(i,1)/countSum(i,1);
  end
end
grid on;
plot(sumCapU,expectSum);
```

Remark 4. *Note that we had to create another vector, countSum, that counts how many times a certain sum has been obtained.*

The following figures depict the plot that we wanted to obtain.

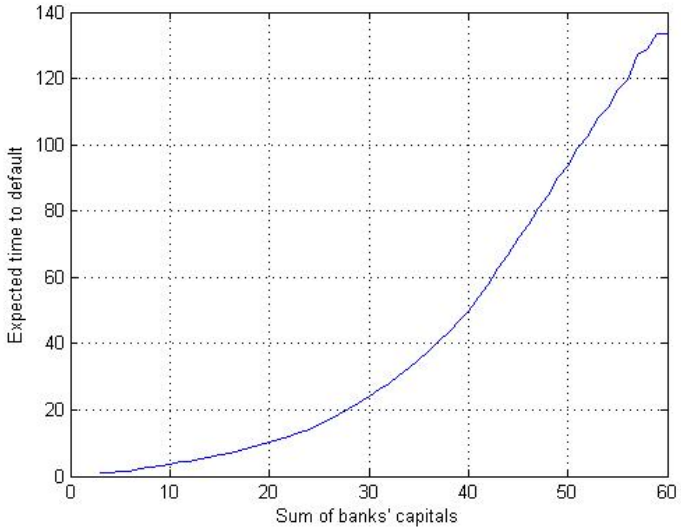


Figure 4: 3 banks 3 connections default time

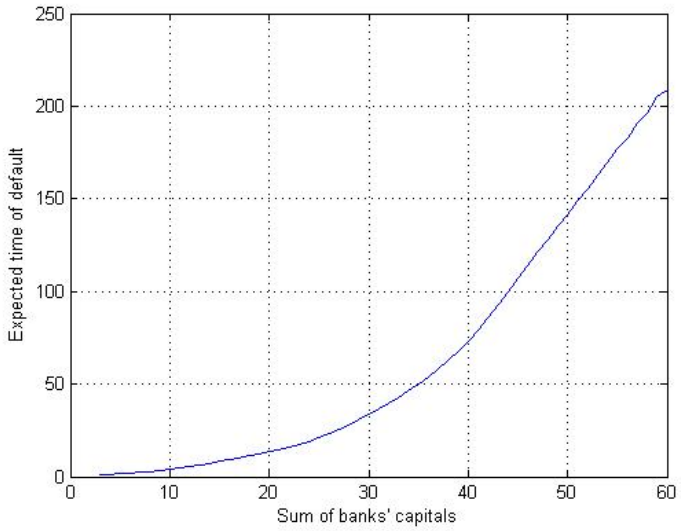


Figure 5: 3 banks 2 connections default time

Observation 6. *From the pictures above, we can observe that the more capital there is in the whole network, the longer we expect to take until the first bank defaults. Moreover, the relationship between the two is almost exponential.*

For the next test, we will fix one of the bank's capital to equal 15 and we will let the other 2 to range from 1 to 35 with increments of 1. As, usual, on each iteration through the initial starting capitals we can compute not only the expected time to default, but also probabilities that the banks will default. Since we already have used the expectation, let us turn our attention towards how to make use of the latter. We are fixing one of the bank's capitals because we would like to see how the other banks' money impact the likelihood that either one or the other will default. Before we delve into the results, let us see how we would implement this idea.

Algorithm 14.
```
counterexpect=0;
for a=1:1:20
  for b=15
    for c=1:1:20
      zaxis(counterexpect)=pbank(1,1)+pbank(3,3)-pbank(1,3);
      xaxis(counterexpect)=a;
      yaxis(counterexpect)=c;
      counterexpect=counterexpect+1;
    end
  end
end
limit=floor(counterexpect/N);
hold on;
grid on;
for i=1:1:limit
  xaxisnew=xaxis(((i-1)*limit+1):(i*limit));
  yaxisnew=yaxis(((i-1)*limit+1:(i*limit)));
  zaxisnew=zaxis(((i-1)*limit+1):(i*limit));
  plot3(xaxisnew,yaxisnew,zaxisnew);
  view(3);
end
plot3(xaxis,yaxis,zaxis);
```

Remark 5. *Note that:*

- *We had to divide the vectors xaxis, yaxis and zaxis in equal parts of length N because every N iterations c starts looping from* 1 *all over again. If this would have been omitted, the plot would have had some extra lines.*
- *This particular code loops through a and c.*

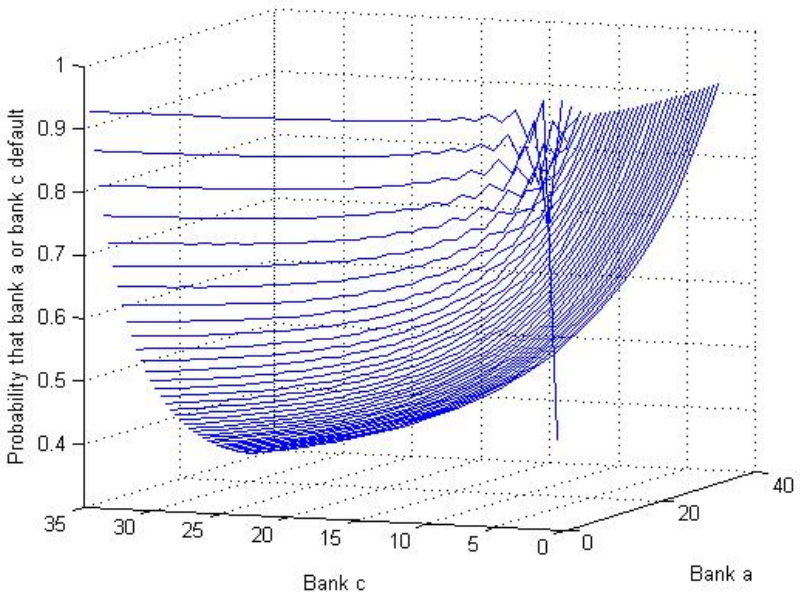


Figure 6: 3 banks 2 connections b=15 default probability

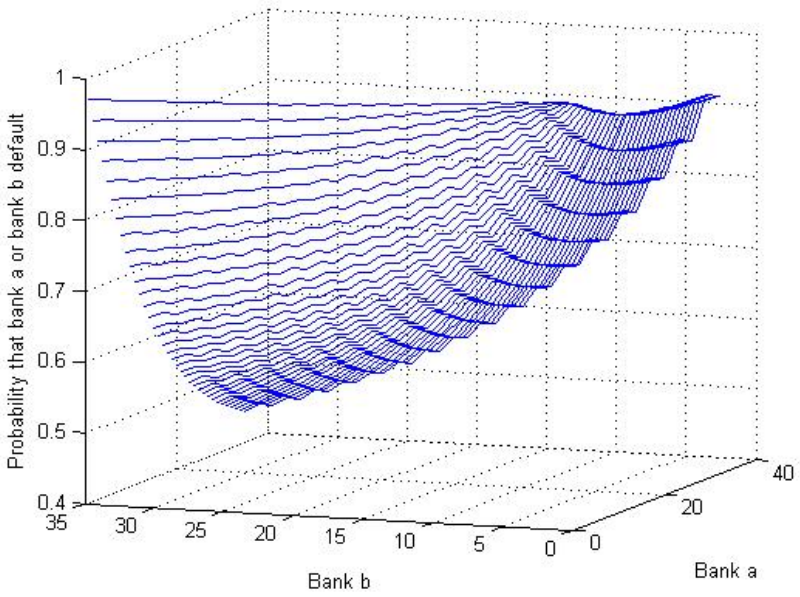


Figure 7: 3 banks 2 connections c=15 default probability

Finally, we are ready to see the results of our plots. We will first compare the plots that resulted for a network in which we have 3 banks and 2 connections:

As we can see, the surface obtained when fixing the central bank (bank b) is shifted up slightly compared to the one that we obtain by fixing one of the other banks (in this case bank c). This is more evident as the capitals of the varying banks increase. This leads us to the following:

Observation 7. *The chances that default happens in a system are smaller when isolated banks have as much capital as possible, than it is when the central bank has big reserves.*

Moreover, in the scenario in which we fix the central bank, the rate at which the chances that either of the other two banks will default decrease faster than in the other scenario.

The following figure is obtained in the same way as described before, the only difference been that now we are in a network of 3 banks with 3 connections. Since this network in symmetric in any of the 3 banks, we can just pick at random one of the banks for which we want to fix the capital. The following simulation was obtained by fixing bank c to equal 15.

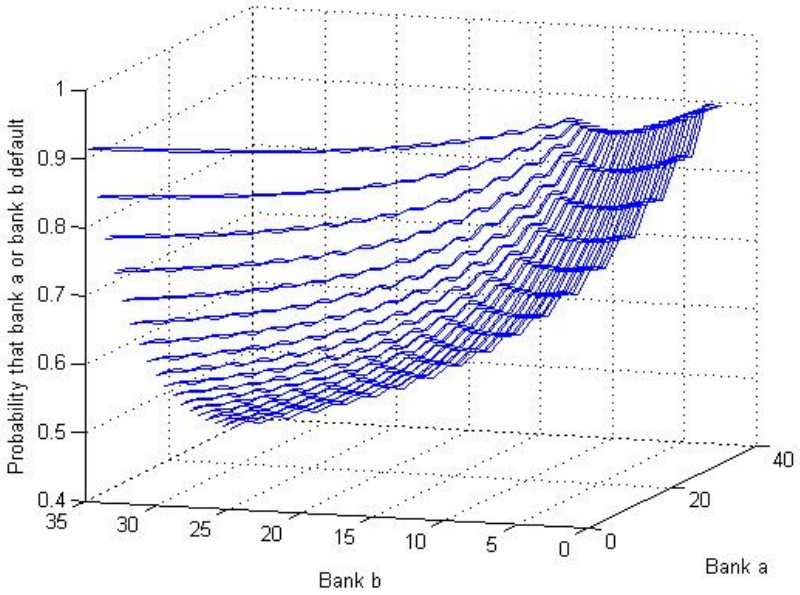


Figure 8: 3 banks 3 connections c=15 default probability

We notice that this scenario seems to give a curve that is in between the other two. As we have seen in **Observation 5**, we are expecting this more connected network to facilitate the spread of default. On the other hand, in case of default, a bank that has many connections should influence the chances of default in our system less than a bank that is very isolated. This is because, in the former situation, the capital that the defaulting bank lost is distributed equally to its neighbors (from equation (5)).

Observation 8. *K_3 (the complete graph with 3 vertices) not only facilitates the spread of default because of its strong connectivity, but also hinders it because of the higher degree that each vertex has.*

Moreover, we would like to see how our curve changes if we fix our bank at a capital of 20. Our intuition tells us that the curve should shift upwards:

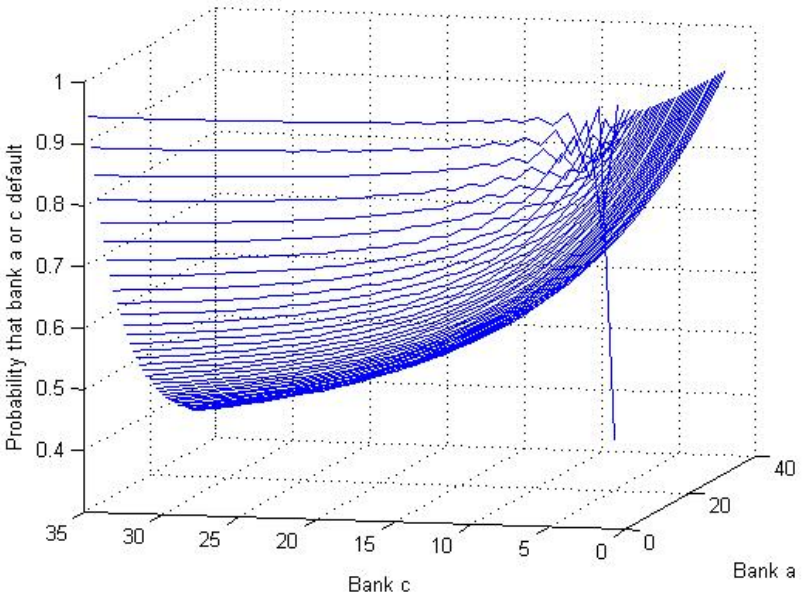


Figure 9: 3 banks 2 connections b=20 default probability

Observation 9. *All the observations previously made for the test in which one of the banks had a fixed capital of* 15 *can be applied here also.*

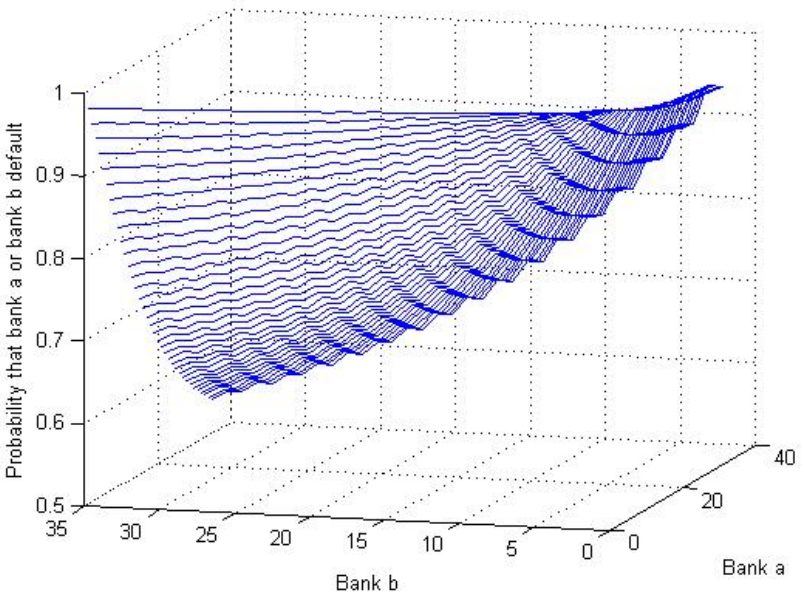


Figure 10: 3 banks 2 connections c=20 default probability

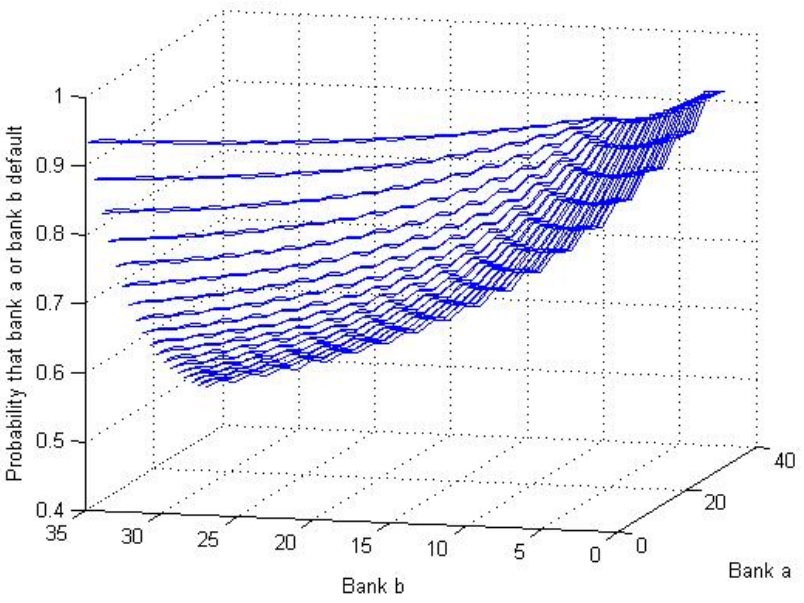


Figure 11: 3 banks 3 connections c=20 default probability

In our simulations we have only used the expected time to default and the probabilities that banks will default. However, it would be interesting to see how conditional probabilities are affected as the capitals for those banks increase. This gives us an idea of a similar test to the one before, that makes use of conditional probabilities. Again, for a given 3 bank network, we will fix a bank's capital to 15 and we will let the others to range from 1 to 35 in increments of 1. For each such iteration, we will be computing and plotting probabilities conditional on the fact that the bank with a fixed capital defaults. The only modification that we have to do to **Algorithm 14** is to replace the assignation of the vector $zaxis$ with $zaxis(counterexpect) = pbankCond(1,3) + pbankCond(2,3)$. Please note that this code is specific for the scenario in which we fix bank c's capital.

The following figures depict the results obtained for systems of 3 banks with 2 connections. As we can see on the next page, in the scenario in which we fix the capital for one of the banks that has only one connection, the whole curve gets close to the xy plane. As a comparison, in the other scenario, in which we fixed the capital for the central bank, only part of the curve seems to get closer and closer to the xy plane. This is because, in the former case, the loss generated by bank c is completely incurred by bank b. Hence, it is important for bank b to hold sufficient funds and, as we have seen previously, the more capital it has, the less likely it is that default will spread. Meanwhile, if the central bank defaults, two banks will be affected. If one of them has a capital small enough, it is very likely that it will default, no matter what the third bank's capital is. This reasoning leads us to a conclusion similar to **Observation 9**, but this time for conditional probabilities, and therefore, better at characterizing how default spreads in the system:

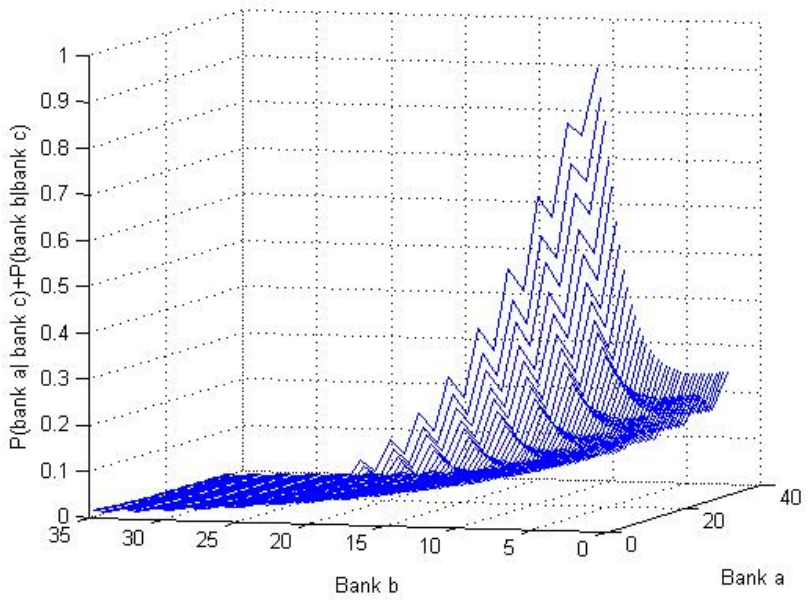


Figure 12: 3 banks 2 connections c=15 conditional default probability

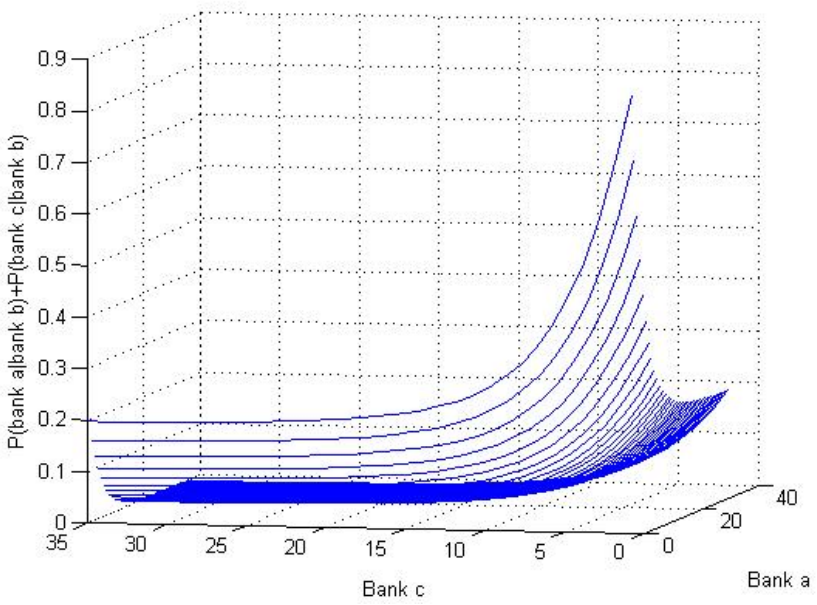


Figure 13: 3 banks 2 connections b=15 conditional default probability

Observation 10. *The chances of contagion are smaller when isolated banks have as much capital as possible, than it is when the central bank has big reserves.*

Now, we will turn our attention towards a K_3 network. The parameters were chosen in the same way as in the previous network simulation. The figure on the following page depicts our results.

Again, just like in the previous test in which we made use of the probabilities of default, the K_3 network produces a graph that is between the first two. Even when it comes to the continuity of the lines, the 3 banks with 3 connections network seems to be the middle ground of the other 2 scenarios tested before. Hence, this test only enforces **Observation 9**.

As we have seen, the way in which the program is coded and the computing power limited the simulations that could have been performed. However, the results obtained are strongly connected to what our intuition tells us. With a faster running time, the algorithm presented in the paper would not only be useful at determining the characteristics of an inter-banking system that does not facilitate the spread of default, but it would also be useful at determining probabilities of default and expected time until the first default for a given big network with given big starting capitals.

5 Future Work

In the future, as suggested by **Remark 3**, the code should be implemented in a programming language that facilitates vectorization. One such programming language would be

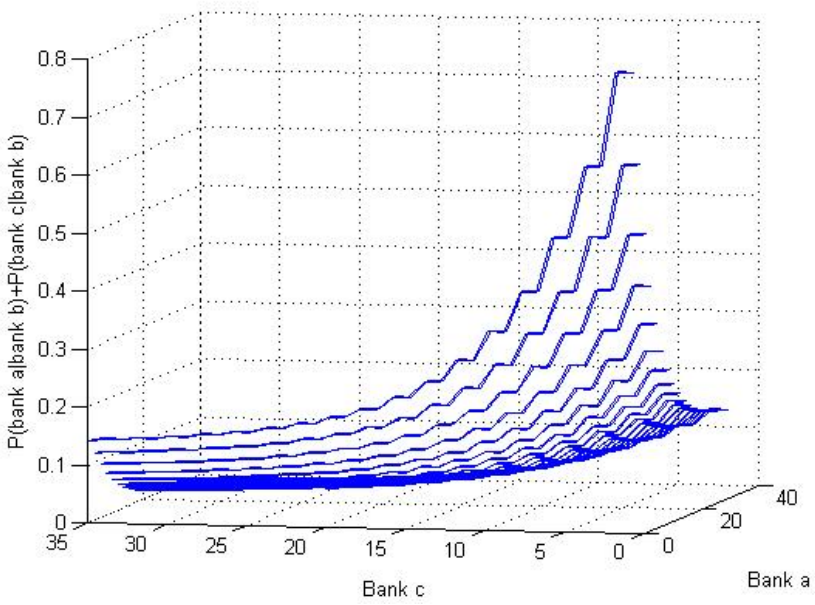


Figure 14: 3 banks 3 connections b=15 conditional default probability

the open source R. This programming language permits certain operations on vectors, without having to write a for loop. However, there is another big advantage that R has over MATLAB when it comes to implementing this algorithm. In order to become aware of this advantage, let us look back at **Algorithm 3**, which was searching for the points that can be reached by our random walk AW_n. The number of nested *while* loops is equal to the number of independent step vectors, which, in turn, varies with the number of banks in the system. Therefore, implementing a function that also takes in as parameter the number of banks in the system is impossible in MATLAB. However, this trick can be done in R, if we use the function $do.call()$.

Moreover, from equation (5), we realize that our code stops as soon as we find the first default. However, once this happens, we could delete the respective vertex in the graph and all its adjacent vertices, re-allocate the starting capitals and rerun the same function. Actually, we could keep on doing this, until the network is reduced to only one bank. This will give us a complete picture of how default contagion spreads in the whole network. Implementing this idea would come after optimizing the run time of the current algorithm.

However, the future work is not solely limited to making the code run faster. Already, there have been developments made towards finalizing a continuous time model, based on the same graph representation of our problem as the one used in discrete time.

References

[1] G. Latouche and V. Ramaswami *Introduction to matrix analytic methods in stochastic modeling* SIAM (1999).

[2] Yvik C. Swan and F. Thomas Bruss *A Matrix-Analytic Approach to the N-player ruin problem.* Journal of Applied Probability **43**, 755-766 (2006)

[3] Karl Sigman *Stopping Times* http://www.columbia.edu/ ks20/stochastic-I/ stochastic-I-ST.pdf (2009)