ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems

Project Report

Ebola Simulation using Networks

Couteau Arthur & Torredimare Marco

Zürich
Dec 2018

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of COSS. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Couteau Arthur

Torredimare Marco

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

EBOLA SIMULATION USING NETWORKS

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
COUTEAU	ARTHUR
TORRENIMARE	MARCO

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
ZÜRICH, 04/12/2018	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

Abstract

In 2013, the major outbreak of Ebola spread in the western African countries, infecting over 40 thousands people. This pushed the scientific community to study this virus and its behaviour, in particular the medical and the social fields had an important role. Again in 2017 and 2018, another outbreak took place in the Democratic Republic of Congo, the new medical treatments allowed a smaller number of causalities, but the models for its behaviour failed in their predictions.
To study the Ebola outbreaks, we have initially used a SIRPD model, an implementation of the SIR model. This system considers various states, which are accounted in the transmission of the Ebola virus, but the resolution of the ordinary differential equations reaches an asymptotic behaviour after a certain time. For this reason and to take into account more properties of the Ebola virus, we changed our model to a network system.
Using an activity driven network, we managed to simulate the interactions between individuals, considered as nodes, and study the development of their state. Fitting the parameters of the latest outbreak, we were able to simulate the transmission of the Ebola virus in a complex system, considering also the medical procedures adopted for this disease.

Individual contributions

The concept and the general procedure to follow was planned by both components of the team, Arthur Couteau and Marco Torredimare. The *Matlab* codes and their implementations were performed by Arthur Couteau. The description of the model and the *Gephi* simulation were done by Marco Torredimare.
The implementation and the results chapters have been realized by both members.

1 Introduction and Motivations

At the end of 2013, the largest Ebola outbreak took place in West Africa. On March 2016, it was not considered an emergency anymore, but in this lapse of time it caused over 11 thousands deaths and almost 30 thousands cases of infection.
More recently, in 2017 and in 2018, new outbreaks happened mainly in the Democratic Republic of Congo [7]. Even thought the disease was identified in 1976 and it was largely studied, especially since 2013, the models present in the scientific literature could not predict the most recent outbreaks of the past two years.
For these reasons, we have decided to study the behaviour of Ebola, starting from a simplified model, and adding more information regarding its peculiarities and ways of transmission.
In doing so, the main questions that we asked ourselves were:

- Which model approximates, in the most accurate way, the Ebola virus?
- Are there stochastic parameters in the outbreaks?
- Can we obtain a model which can predict accurately the behaviour of Ebola outbreaks?

We will try to achieve these goals analyzing the available data, in particular for the Democratic Republic of Congo, one of the principal countries in Ebola outbreaks.

2 Description of the Model

2.1 SIR Model

One of the most used model to study epidemics is the so-called SIR model, introduced by Kermack and McKendrick [3]. Its name refers to the three main groups studied with this model: S, susceptible individuals, I, infected, R, removed.
This model makes the assumptions of a constant population and of no incubation time, which implies that the change of state from S to I is immediate and the infectious individual can already spread the disease.
The equations governing this model are:

$$\frac{dS}{dt} = -\beta I(t)S(t) \tag{1}$$

$$\frac{dI}{dt} = \beta I(t)S(t) - \gamma I(t) \tag{2}$$

$$\frac{dR}{dt} = \gamma I(t)) \tag{3}$$

S, I, R are the number of susceptible, infected and removed individuals. With removed is intended a recovered or a dead person, who cannot contract the disease. While β and γ represent respectively the infecting rate and the recovery/death rate. Every system based on this model will eventually end, when every S becomes I, which will recover or die, resulting in every individual being removed.
For this reason, this model is not accurate to simulate an Ebola outbreak and its development.

2.2 SIRPD Model

For the base of our model, we started from the SIRPD model realized by Berge et al. [2]. This model includes susceptible, S, infected, I, infected and deceased, D, recovered, R, and the pathogens in the environment, P.
A SIRPD model is more suited for an epidemics such as Ebola, because it takes into considerations some infecting characteristic of the virus. In particular:

- Deceased human can still infect during the funerals;
- Human fluids of an infected or deceased person are infective;
- After an individual recovers, he cannot contract the virus for a second time.

In this model more parameters are taken into consideration, a representation can be seen in Fig. 1:

Symbol	Description
S	Susceptible human individuals
I	Infectious human individuals
D	Ebola infected and deceased human individuals
R	Recovered human individuals
P	Ebola virus pathogens in the environment
π	Recruitment rate of susceptible human individuals
η	Decay rate of Ebola virus in the environment
ζ	Shedding rate of infectious human individuals
α	Shedding rate of deceased human individuals
δ	Disease-induced death rate of human individuals
$\beta 1$	Effective contact rate of infectious human individuals
$\beta 2$	Effective contact rate of deceased human individuals
λ	Effective contact rate of Ebola virus
Υ	Recovered rate of human individuals
μ	Natural death rate of human individuals
1/b	Mean caring duration of deceased human individuals
σ	Recruitment rate of Ebola virus in the environment

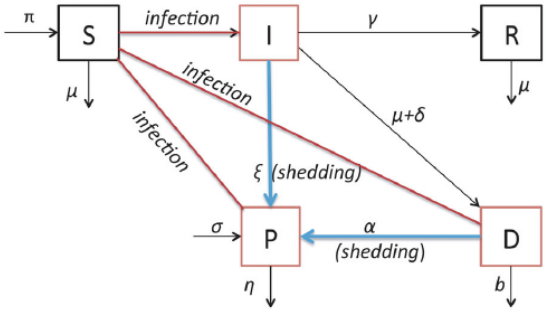


Figure 1: Ebola transmission diagram, from Berge et al. [2].

Resulting in the following system of ordinary differential equations:

$$\begin{aligned}
\frac{dS(t)}{dt} &= \pi - (\beta 1 I + \beta 2 D + \lambda P)S - \mu S \\
\frac{dI(t)}{dt} &= (\beta 1 I + \beta 2 D + \lambda P)S - (\mu + \delta + \gamma)I \\
\frac{dR(t)}{dt} &= \gamma I - \mu R \\
\frac{dD(t)}{dt} &= (\mu + \delta)I - bD \\
\frac{dP(t)}{dt} &= \sigma + \zeta I + \alpha D - \eta P
\end{aligned} \tag{4}$$

With a proper starting setting of the initial conditions for S, I, R, D and P, we can combine the previous equations to obtain another differential equation to describe the active population, H = S + I + R.

$$\frac{dH(t)}{dt} = \pi - \mu H - \delta I \tag{5}$$

This model takes into consideration most of the parameters for the spreading of the Ebola virus, but it doesn't take into account the transmission by random encounters. All its data come from the World Health Organization or are determined by the authors.

2.3 Ebola Variation of the SEIR Model

We need a further model which takes into account the parameters for the natural spreading of the virus, plus it has to consider the pseudo-random variables for the transmission between two random individuals.
For this type of system, we have taken the model proposed by Legrand et al. [4], it has been developed using a stochastic model, which best suits our assumptions and parameters.
In this case, a total of six states are taken into consideration: S, susceptible individuals; E, exposed individuals; I, infectious individuals; H, infectious and hospitalized individuals; F, not buried corpses; R, recovered individuals or dead buried corpses. A scheme of the various transitions can be seen at Fig. 2.
This model follows the following assumptions:

- The entire population initially belongs to S;

Transition		Transition rate (λ_i)
1	(S, E)→(S − 1, E + 1)	$(\beta_I SI + \beta_H SH + \beta_F SF)/N$
2	(E, I)→(E − 1, I + 1)	αE
3	(I, H)→(I − 1, H + 1)	$\gamma_h \theta_1 I$
4	(H, F)→(H − 1, F + 1)	$\gamma_{dh} \delta_2 H$
5	(F, R)→(F − 1, R + 1)	$\gamma_f F$
6	(I, R)→(I − 1, R + 1)	$\gamma_i (1 - \theta_1)(1 - \delta_1) I$
7	(I, F)→(I − 1, F + 1)	$\delta_1 (1 - \theta_1) \gamma_d I$
8	(H, R)→(H − 1, R + 1)	$\gamma_{ih} (1 - \delta_2) H$

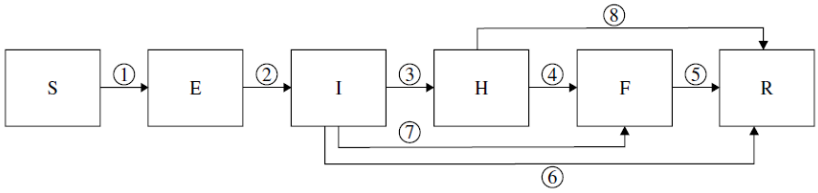


Figure 2: Ebola transmission scheme for the SEIHFR model, from Legrand et al. [4].

- Interventions have been taken into considerations from the dates provided by the Weekly Epidemiological Record [1]. So the transmission in the hospitals or during funeral after this date is considered zero;
- The input parameters are taken from the official data for the specific case study of the outbreak in Uganda in 2000;
- All transmissions are considered from human-to-human.

3 Implementation

Using *Matlab*, we created our code starting from the models described before, adapting and implementing them to our purpose.

3.1 Ideal Behaviour of the SIRPD Model

The first model we have created is the SIRPD model, which exploits the ordinary differential equations for the development over time of the states of the individuals. Our first implementation was to simplify the equations so that they could be easily computed by *Matlab*. Running the program allowed us to see that this model reaches an asymptotic behaviour after some days. This is not what happens in reality.
The reason for this behaviour is due to the definition of the parameters, they are all constant over time, and there is no stochastic parameter to allow a random behaviour of the individuals.

3.2 Network System

3.2.1 Activity Driven Network

To exploit the new stochastic time dependent parameters, we decided to implement the SEIHFR model. To achieve this goal, we have transformed it using a network system, as proposed by Rizzo et al. [8].
In this case each node represents an individual and the edges represent the interactions. In particular, the model we have used is called Activity Driven Network (ADN), which describes the interactions that evolve over time-varying networks, this type of system is well suited for an Ebola outbreak development.

3.2.2 Implementation of the ADN

The parameters of the ADN are defined in the main function `Batch`, and are used to generate the network function `EVD_Network`. This function calls in turn `Activity_distribution`, which associates an activity (meaning a probability of becoming active during a time step) to each node of the network.
As the network is now fully initialized, the function `ADNTempIter` is called to iterate the network over time. In `ADNTempIter`, at each time step, some nodes become active,then each active node creates edges with other random nodes in the network. Then, depending on the state of the active ones and their connected nodes, their state may change.

The transition rates are computed through `RateComputation`. Finally, `PrintOutput` is called and it prints a text file in a folder named *OutputEbola*. Once all the trials are computed, the function `Batch` calls `OutputInterpreter` to plot the evolution of the population over time.

3.3 Dataset

To make the parameters time-dependent, we need to have data from previous Ebola outbreaks. In our simulations we took a dataset from the most recent Ebola outbreak in the Democratic Republic of Congo [5].
We decided to use the most updated data because the medical infrastructures to prevent and to cure Ebola have improved in the recent years, so that we could have a good prediction of the Ebola behaviour.
In general, finding a precise dataset, with weekly information about infected, hospitalized, recovered individuals and deaths, is very hard. Furthermore, a big dataset, which is easier to find, is very hard to handle and computation consuming. On the other hand, smaller dataset come from outbreaks in the second half of the twentieth century, so they are not as detailed as we need them to be to implement our network.

3.4 Visualization of Dynamic Systems

Using networks, we thought it would be helpful to have a clear visualization of the interactions between individuals and how they change their state during the spreading of the disease.
To do so, we have exploited *Gephi*, a visualization software, where we used dynamic nodes, to observe, on a smaller sample, the behaviour of the disease.

4 Simulation Results and Discussion

4.1 Model calibration

The model has been calibrated based on the available WHO epidemic data of the August 2018 North-Kivu outbreak in RDC [5]. As the number of parameters in this simulation is large, and in order to avoid overfitting, only the ADN related parameters have been identified: the number of edges m created by an active node and the activity rates η_{SE} and η_I. All the other parameters are taken from [8].
Our EVD model is calibrated based on both count of total cases and of total deaths, from August 4, 2018 to December 4, 2018, for a time-span of 123 days. Two discrete-time epidemic curves $I_{cases}(t)$ and $I_{deaths}(t)$ are obtained from the dataset. Then, the EVD model is run on a network population $N = 10^4$, with time step $\Delta t = 1$ day, varying the parameters m, η_{SE} and η_I. The parameters were in the following ranges: $m \in \{5, ..., 8\}$ with step 1, $\eta_{SE} \in \{5, ..., 7\}$ with step 0.5, $\eta_{SE} \in \{4, ..., 6.5\}$ with step 0.5. Each EVD model with its set of parameters has been run for 20 independent trials, with initial number of cases 43 and initial number of deaths 33.
By averaging the results of those 20 trials, 2 epidemics curves are obtained: $I_{cases,sim}(t)$ and $I_{deaths,sim}(t)$. The parameters are then identified by minimizing the function $J(m, \eta_{SE}, \eta_I)$:

$$J(m, \eta_{SE}, \eta_I) = \frac{1}{123} \sum_{t=0}^{123} \left[\left(I_{cases}(t) - I_{cases,sim}(t) \right)^2 + \left(I_{deaths}(t) - I_{deaths,sim}(t) \right)^2 \right] \tag{6}$$

The identified parameters are:

$$\begin{aligned} m &= 7 \\ \eta_{SE} &= 5 \\ \eta_I &= 6 \end{aligned} \tag{7}$$

The evolution of the population of the simulation and the data can be found in Figure 3. The ADN parameters are different from the ones obtained by Rizzo et al. [8], this is explained by the fact that the network population N is much lower in our simulation for computational reasons. Also, as the outbreak is still in the spreading phase (there are, until now, no drastic changes in the number of people getting infected per day), the time-varying parameters of the model have been kept constant throughout the whole run and set equal to those described by [8] as phase 1 parameters.

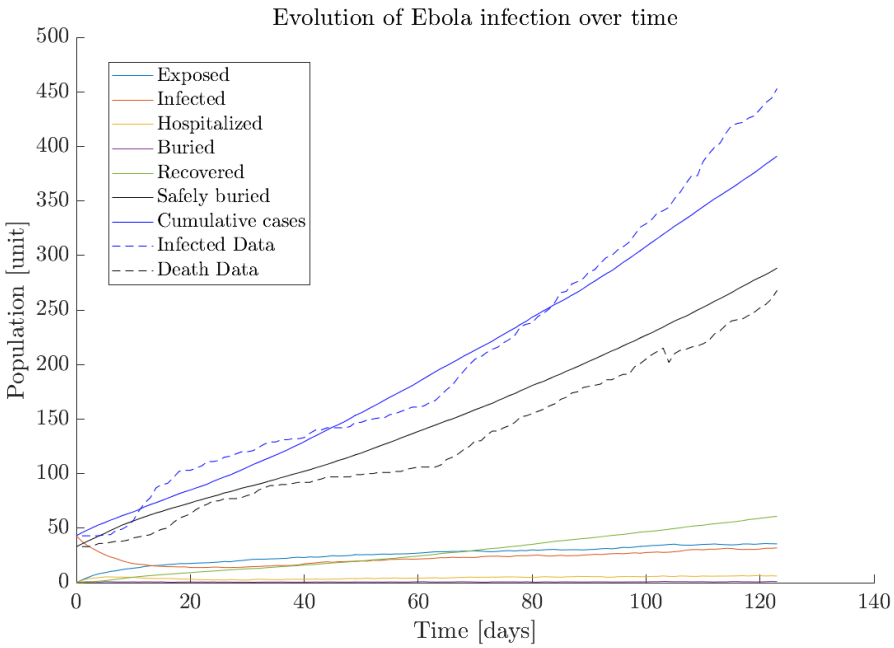


Figure 3: Evolution of populations over time, August 2018 outbreak in North-Kivu

4.1.1 Model validation

To validate the model and the choice of parameters, we have run 50 trials and compared the results with the available epidemic curve obtained from the May 2018 Equator outbreak in RDC [6]. The initial number of cases and of deaths are respectively 34 and 18.
The distribution over time of the time-varying parameters has been set accordingly to the data: the first phase runs from the 11th to the 16th of May, the second from the 17th to the 20th of August and the third for the rest of the simulation. The evolution of the population is represented in Figure 4.

4.2 Network Visualization

Using the *Matlab* codes, we obtained the evolution for a limited batch to simulate its graphical representation with *Gephi*. We considered a population of 100 individuals, with 8 initial cases and one initial death.
After assembling the input file for the nodes, to create a dynamic network, we defined for each state a color and a size.

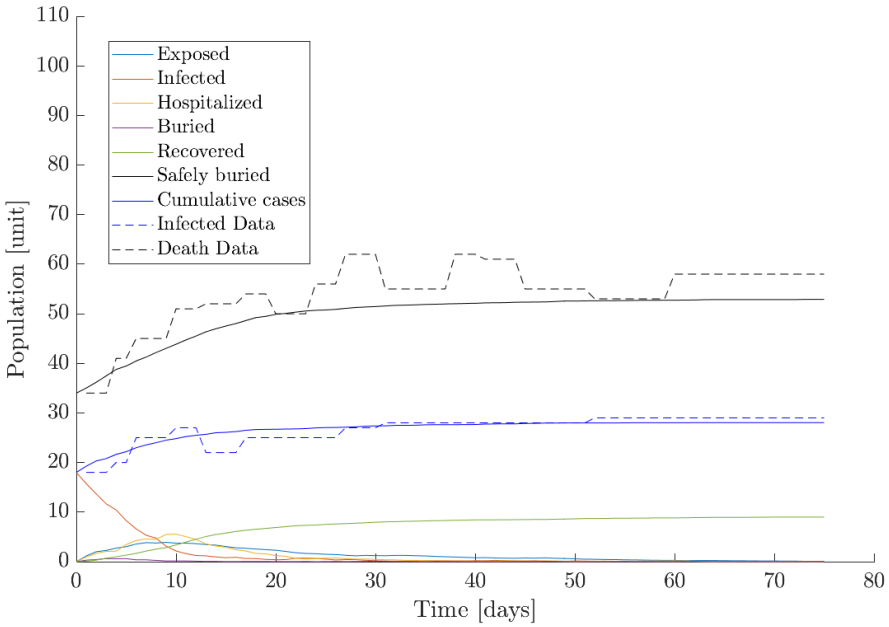
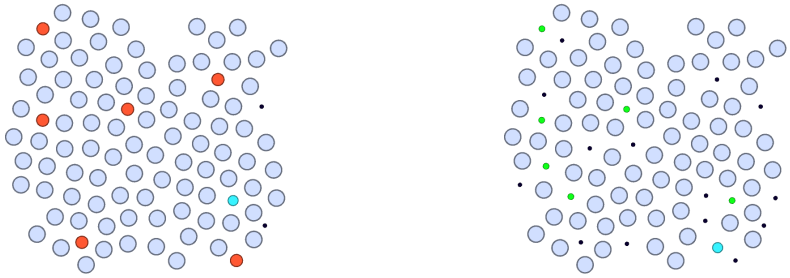


Figure 4: Evolution of populations over time, May 2018 outbreak in Equateur, RDC

- S = gray;
- E = yellow;
- I = red;
- H = light blue;
- F = blue;
- R = green;
- D = black.

The size defines the quantity of iterations with the disease that an individual can have, so after the end of the infection (R or D state), the node will have no more interactions with the virus.
The results of the simulation can be seen in Fig. 5



[a] [b]

Figure 5: The two networks show the system at the beginning and at the end of the time period considered.

5 Summary and Outlook

In this project, we have implemented an Activity Driven Network to simulate an outbreak of the Ebola virus.
The network evolution is based on an improved SIR model developed by Legrand et al. [4]. This SEIHFR model takes into account more possible states to include more specific dynamics (such as transmission via unsafe traditional funerals) in the evolution of the populations.
The modelization of the network has been adapted from Rizzo et al. [8], which described an ADN implementation of the SEIHFR model for the 2014-2016 Ebola outbreak in Central Africa, to fit the August 2018 outbreak in North-Kivu, RDC.
A more intensive analysis can be performed using more calculation power, to study bigger dataset, to simulate the whole network, considering all inhabitants of a specific country.
Further implementations can consider the animal-to-human transmission, in particular by bats. This animal is very important in the transmission of the virus, but its behaviour is very complex to be taken into account in a system, considering its movements are dependent on climatic and seasonal parameters.
Another parameter that can be explored is the location of the individuals, if they come from a big city or from a village. This correlates their possibility to have access to health care treatments and it gives an idea on their average living conditions.

References

[1] Anonymous. Outbreak of ebola haemorrhagic fever, uganda, august 2000-january 2001. *Weekly Epidemiological Record*, 76:41–46, 2001.

[2] T. Berge, J.-S. Lubuma, G. Moremedi, N. Morris, and R. Kondera-Shava. A simple mathematical model for ebola in africa. *Journal of Biological Dynamics*, 11(1):42–74, 2017.

[3] W. Kermack and A. McKendrick. Contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London Series A-Containing Papers of a Mathematical and Physical Character*, 115(772):700–721, 1927.

[4] J. Legrand, R. F. Grais, P. Y. Boelle, A. J. Valleron, and A. Flahault. Understanding the dynamics of ebola epidemics. *Epidemiol. Infect.*, 135:610–621, 2007.

[5] W. H. Organization. Dr congo north-kivu - ebola cases and deaths, 2018. URL `https://data.humdata.org/dataset/ebola-cases-and-deaths-drc-north-kivu`.

[6] W. H. Organization. Dr congo - ebola cases and deaths, 2018. URL `https://data.humdata.org/dataset/ebola-cases-and-deaths-drc`.

[7] W. H. Organization. Ebola outbreak in drc ends: Who calls for international efforts to stop other deadly outbreaks in the country, 2018. URL `http://www.who.int/news-room/detail/24-07-2018-ebola-outbreak-in-drc-ends--who-calls-for-international-efforts-to-stop-other-deadly-outbreaks-in-the-country`.

[8] A. Rizzo, B. Pedalino, and M. Porfiri. A network model for ebola spreading. *Journal of Theoretical Biology*, 394:212–222, 2016.

Appendix

A Light-test instructions

To perform the light test, follow the instructions:

- Verify that all the functions detailed in Appendix C are in the same folder, along with the data files `Data_May2018.mat` and `Data_Aug2018.mat`.
- In the command window, enter the following command line:
  `>> maxNumCompThreads`
  This will allow Matlab to use the maximum number of threads available. If you want to limit the number of cores Matlab will use, type:
  `>> LASTN = maxNumCompThreads(N)`
  Where N is the number of threads, and LASTN the previous maximum number.
- Open function `Batch.m`.
- Make sure the parameters defined in part 1 have the same value as in Appendix C.
- Make sure the part 2 of the function is uncommented (`CTRL + T` to uncomment selected text) and that the part 3 is commented (`CTRL + R` to comment selected text).
- Run function `Batch.m`. The light test computes the evolution of populations over time, and compares it to data coming from WHO. This will take $\approx$ 4 min on 4 cores.
- A folder named `OutputEbola` is created by the program. It contains text files of the different trials of the simulation.
- The graph depicted in Figure 3 will appear.

B Full test instructions

- Open function `Batch.m`.
- Make sure the parameters defined in part 1 have the same value as in Appendix C.
- Make sure the part 2 of the function is commented and that the part 3 is uncommented.
- Run function `Batch.m`. This will take $\approx$ 2h on 4 cores. This will compute the best set of ADN parameters based on least square. The results will be printed in command window:
  `m = 7 etaSE = 5 etaI = 6`
  A file named `J.mat` is created, containing the values of the function defined in Equation (6).

- A graph of the comparison with data from May 2018 outbreak will appear. This graph is depicted in Figure 4

C MATLAB codes

C.1 Batch.m

```
function Batch
    % Main function
    % Set the parameters of the network and run the simulation

    %% 1. Parameters
        % 1.1 Network parameters
        N     = 1e4;       % Size of the population
        nc    = 43;        % Number of initial cases
        nd    = 33;        % Number of initial dead
        tEnd  = 123;       % Number of time steps [days]
        m     = 7;         % Number of edges created
        etaSE = 5;         % Activation factor for states S and E
        etaI  = 6;         % Activation factor for state I

        Parameters = [N, nc, nd, tEnd, m, etaSE, etaI];

        % 1.2 Reproducibility parameters
        rng('default');

        % 1.3 Output
        if ~exist('OutputEbola', 'dir')
            mkdir('OutputEbola')
        end
        rootDir = pwd;
        relDir = 'OutputEbola';
        delete('OutputEbola\*') % Clean directory (make sure to keep your data
            elsewhere before running Batch again!)

    %% 2. Light test (Instructions can be found in Appendix A)
    numberLTest = 50;  % Number of trials
    parfor iTest = 1:numberLTest
        rng(100 + iTest); % Reproducibility
        fileID = fopen(fullfile(rootDir, relDir, strcat(['Data_LightTest'
            num2str(iTest) '.txt'])), 'w');
        fprintf(fileID,'%8s %8s %8s %8s %8s %8s %8s %8s
            %8s\n','Time','S','E','I','H','F','R','D','Cumul');
        EVD_Network(Parameters,fileID,'Aug');
        fclose(fileID);
    end

    OutputInterpreter(rootDir, relDir, 'LightTest', tEnd, numberLTest)
```

```
    %% 3. Full test (Instructions can be found in Appendix B)
    % 3.1 Comparison with May 2018 outbreak
    numberCTest = 50;  % Number of trials
    parfor iTest = 1:numberCTest
        rng(300 + iTest); % Reproducibility
        Parameters = [1e4, 18, 34, 75, 7, 5, 6];
        fileID = fopen(fullfile(rootDir, relDir, strcat(['Data_TestComp'
            num2str(iTest) '.txt'])), 'w');
        fprintf(fileID,'%8s %8s %8s %8s %8s %8s %8s %8s
            %8s\n','Time','S','E','I','H','F','R','D','Cumul');
        EVD_Network(Parameters,fileID,'May');
        fclose(fileID);
    end

    OutputInterpreter(rootDir, relDir, 'Comparison', 75, numberCTest)

    % 3.2 Fit parameters
    m_range = 5:8;
    etaSE_range = 5:0.5:7;
    etaI_range = 4:0.5:6.5;
    numberFTest = 20;

    parfor im = 1:length(m_range)
        for iSE = 1:length(etaSE_range)
            for iI = 1:length(etaI_range)
                for iTest = 1:numberFTest
                    rng(200 + iTest); % Reproducibility
                    Parameters = [N, nc, nd, tEnd, m_range(im), etaSE_range(iSE),
                        etaI_range(iI)];
                    fileID = fopen(fullfile(rootDir, relDir,strcat(['data_m'
                        num2str(m_range(im)) '_SE' num2str(etaSE_range(iSE)) '_I'
                        num2str(etaI_range(iI)) '_Test' num2str(iTest)
                        '.txt'])),'w');
                    fprintf(fileID,'%8s %8s %8s %8s %8s %8s %8s %8s
                        %8s\n','Time','S','E','I','H','F','R','D','Cum');
                    EVD_Network(Parameters,fileID,'Aug');
                    fclose(fileID);
                end
            end
        end
    end

    OutputInterpreter(rootDir, relDir, 'FullTest', tEnd, numberFTest, m_range,
        etaSE_range, etaI_range)
end
```

C.2 EVD_Network.m

```
function EVD_Network(Parameters,fileID,Outbreak)
       % Input: - Network: Structure containing the network
       %        - fileID: Reference to output file
       %        - Outbreak: May or Aug
       %
       % Calls Activity_distribution to initialize the nodes' activity potential.
       % Calls ADNTempIter to iterate over time.

    % Creating a network of size N
    N = Parameters(1);
    Network.parameters = Parameters;                    % Parameters set in Batch
    Network.nodes = 1:N;                                % Cardinality of node
    Network.edges = spalloc(N,N,1);                     % Edges of node
    Network.state = cell(1,N);                          % State of node
    Network.active = false(1,N);                        % Activity of node
    Network.populations = {'S','E','I','H','F','R','D'}; % States
    Network.populationscount = zeros(1,7);              % Count of each populations
    Network.changed = false(1,N);                       % Determine if a node has
        changed its state
    Network.cumcases = 0;                               % Cumulative cases

    % Associate each node an activity potential distributed by power law
    gamma = 2.1;  % power factor
    Network.activity_potential = Activity_distribution(N, gamma);

    % Evolution of the network
    ADNTempIter(Network, fileID, Outbreak);
end
```

C.3 Activity_distribution.m

```
function [x] = Activity_distribution(N, gamma)
    % Input: - N: Size of the population
    %        - gamma: Exponant of the power law
    %
    % Computes the activity potential x in ADN of N nodes with respect to
    % the PDF:
    %      f(x) = x^(-gamma)
    % using inverse transform sampling.

    x_sampling = 1e-3 : 1e-4 : 1; % Avoid singularity at x = 0 of power law
```

```
    PowerLawPDF = x_sampling.^(-gamma);     % PDF f(x)
    PowerLawCDF = cumsum(PowerLawPDF);      % CDF F(x)
    PowerLawCDF = PowerLawCDF/PowerLawCDF(end); % Normalization

    % Generating random numbers based on uniform random distribution U ~ [0,1]
    U_distribution = rand(N,1);
    x = zeros(N,1);

    for i = 1:N
        index = find(PowerLawCDF >= U_distribution(i), 1, 'first');
        x(i) = x_sampling(index);
    end
end
```

C.4 ADNTempIter.m

```
function ADNTempIter(Network, fileID, Outbreak)
    % Input: - Network: Structure containing the network
    %        - fileID: Reference to output file
    %        - Outbreak: May or Aug
    %
    % Iterates network over time: create edges between nodes, calls
    % RateComputation, spreads the infection, update population states and
    % calls PrintOutput.

    N = Network.parameters(1);
    s = RandStream('mlfg6331_64');  % Reproducibility
    for iTemp = 1:Network.parameters(4)
        %% 0. Initialization
        if (iTemp == 1)
            % Initial cases
            initialCases = Network.parameters(2);
            initialDead = Network.parameters(3);
            Network.cumcases = Network.cumcases + initialCases;
            indexes = randsample(N, initialCases + initialDead);
            indexesCases = datasample(s, indexes, initialCases, 'Replace', false);

            % Iterations over the nodes
            for iNode = 1:N
                if (~isempty(nonzeros(iNode == indexes)))
                    if (~isempty(nonzeros(iNode == indexesCases)))
                        Network.state{iNode} = 'I';
                    else
                        Network.state{iNode} = 'D';
```

```
                end
            else
                Network.state{iNode} = 'S';
            end
        end

        % Counting the population
        for iPop = 1:7 % 7 different populations
            Network.populationscount(iPop) = sum(ismember(Network.state,
                Network.populations{iPop}));
        end
        % Print out results
        PrintOutput(0, Network.populationscount, Network.cumcases, fileID)
    end

    % Increase the performance: check if solution can not evolve
    if ((Network.populationscount(1) + Network.populationscount(6) +
        Network.populationscount(7)) ~= N)
        %% 1. Creation of the edges

        % Determine which nodes are active
        for iNode = 1:N
            eta = 0;
            if ((Network.state{iNode} == 'S') || (Network.state{iNode} == 'E'))
                eta = Network.parameters(6);
            elseif (Network.state{iNode} == 'I')
                eta = Network.parameters(7);
            end
            u = rand();
            if (eta*Network.activity_potential(iNode) >= u)
                Network.active(iNode) = true;
            end
        end
        activesIndices = find(Network.active == true);

        % Link each active node to m other nodes
        m = Network.parameters(5);
        numberEdges = zeros(1,N);

        for iNode = 1:length(activesIndices)    % For each active node
            randOther = randsample(N, m);       % Randomly choose m other
                nodes to create edges with
            for jNode = 1:length(randOther)     % Loop over the randomly
                chosen nodes
               % Cannot create edge with itself
```

```
                if (activesIndices(iNode) ~= randOther(jNode) &&
                    (numberEdges(activesIndices(iNode)) < m))
                  % Check if edge has already been created and if jNode has
                      already created m edges
                  if ((Network.edges(activesIndices(iNode),randOther(jNode))
                      == 0) && (numberEdges(randOther(jNode)) < m))
                      Network.edges(activesIndices(iNode),randOther(jNode)) =
                          1;
                      Network.edges(randOther(jNode),activesIndices(iNode)) =
                          1; % Create edge
                      numberEdges(activesIndices(iNode)) =
                          numberEdges(activesIndices(iNode)) + 1;
                      numberEdges(randOther(jNode)) =
                          numberEdges(randOther(jNode)) + 1; % Actualize
                          numberEdges
                  end
                end
            end

            % While the node has not created m edges
            while (numberEdges(activesIndices(iNode)) < m)
                jNode = randsample(N,1);
                if (activesIndices(iNode) ~= jNode)
                    if ((Network.edges(activesIndices(iNode),jNode) == 0) &&
                        (numberEdges(jNode) < m))
                        Network.edges(activesIndices(iNode),jNode) = 1;
                        Network.edges(jNode,activesIndices(iNode)) = 1;
                        numberEdges(activesIndices(iNode)) =
                            numberEdges(activesIndices(iNode)) + 1;
                        numberEdges(jNode) = numberEdges(jNode) + 1;
                    end
                end
            end
        end

        %% 2. Change of state

        %% 2.1 Stochastic compartemental model (cf Table 2)
        [popFraction, transitionRate] = RateComputation(iTemp, Outbreak);

        for iNode = randperm(length(activesIndices)) % For each active node
            state = Network.state{activesIndices(iNode)};
            switch state
                case 'S'
                    % Infection spreading
```

```
                    connectedNodes =
                        find(Network.edges(activesIndices(iNode),:) ~= 0); %
                        Nodes connected
                    for iNodeCon = 1:length(connectedNodes)
                        stateConnected =
                            Network.state{connectedNodes(iNodeCon)};
                        if (stateConnected == 'I')
                            u = rand();
                            if ((transitionRate(1) >= u) &&
                                (Network.changed(activesIndices(iNode)) ==
                                false))
                                Network.state{activesIndices(iNode)} = 'E'; % I
                                    infects S
                                Network.changed(activesIndices(iNode)) = true;
                                Network.cumcases = Network.cumcases + 1;
                            end
                        elseif (stateConnected == 'H')
                            u = rand();
                            if ((transitionRate(2) >= u) &&
                                (Network.changed(activesIndices(iNode)) ==
                                false))
                                Network.state{activesIndices(iNode)} = 'E'; % H
                                    infects S
                                Network.changed(activesIndices(iNode)) = true;
                                Network.cumcases = Network.cumcases + 1;
                            end
                        elseif (stateConnected == 'F')
                            u = rand();
                            if ((transitionRate(3) >= u) &&
                                (Network.changed(activesIndices(iNode)) ==
                                false))
                                Network.state{activesIndices(iNode)} = 'E'; % F
                                    infects S
                                Network.changed(activesIndices(iNode)) = true;
                                Network.cumcases = Network.cumcases + 1;
                            end
                        end
                    end
                case 'E'
                    % Do nothing
                case 'I'
                    % Infection spreading
                    connectedNodes =
                        find(Network.edges(activesIndices(iNode),:) ~= 0); %
                        Nodes connected
                    for iNodeCon = 1:length(connectedNodes)
```

```
                        stateConnected =
                            Network.state{connectedNodes(iNodeCon)};
                        if (stateConnected == 'S')
                            u = rand();
                            if ((transitionRate(1) >= u) &&
                                (Network.changed(connectedNodes(iNodeCon)) ==
                                false))
                                Network.state{connectedNodes(iNodeCon)} = 'E'; %
                                    I infects S
                                Network.changed(connectedNodes(iNodeCon)) = true;
                                Network.cumcases = Network.cumcases + 1;
                            end
                        end
                    end
                case 'H'
                    % Infection spreading
                    connectedNodes =
                        find(Network.edges(activesIndices(iNode),:) ~= 0); %
                        Nodes connected
                    for iNodeCon = 1:length(connectedNodes)
                        stateConnected =
                            Network.state{connectedNodes(iNodeCon)};
                        if (stateConnected == 'S')
                            u = rand();
                            if ((transitionRate(2) >= u) &&
                                (Network.changed(connectedNodes(iNodeCon)) ==
                                false))
                                Network.state{connectedNodes(iNodeCon)} = 'E'; %
                                    H infects S
                                Network.changed(connectedNodes(iNodeCon)) = true;
                                Network.cumcases = Network.cumcases + 1;
                            end
                        end
                    end
                case 'F'
                    % Infection spreading
                    connectedNodes =
                        find(Network.edges(activesIndices(iNode),:) ~= 0); %
                        Nodes connected
                    for iNodeCon = 1:length(connectedNodes)
                        stateConnected =
                            Network.state{connectedNodes(iNodeCon)};
                        if (stateConnected == 'S')
                            u = rand();
                            if ((transitionRate(3) >= u) &&
                                (Network.changed(connectedNodes(iNodeCon)) ==
```

```
                            false))
                            Network.state{connectedNodes(iNodeCon)} = 'E'; %
                                F infects S
                            Network.changed(connectedNodes(iNodeCon)) = true;
                            Network.cumcases = Network.cumcases + 1;
                        end
                    end
                end
            case 'R'
                % Do nothing
            case 'D'
                % Do nothing
        end
    end

    % Counting the population
    for iPop = 1:6
        Network.populationscount(iPop) = sum(ismember(Network.state,
            Network.populations{iPop}));
    end
    % I and H populations are fractioned depending on the state they may
        change to
    cumPopFraction = [1
        cumsum(round(Network.populationscount(3)*popFraction(1:3).*ones(1,3)))
        ...
                    Network.populationscount(3) ...
                    1 round(Network.populationscount(4)*popFraction(5))
                        ...
                    Network.populationscount(4) ];
    % Making sure cumPopFraction does not contain 0
    for iTest = 1:length(cumPopFraction)
        if (cumPopFraction(iTest) == 0)
            cumPopFraction(iTest) = 1;
        end
    end

    %% 2.2 Spontaneous change of state

    % Exposed nodes, 1 possibility: E -> I
    exposedNodes = find([Network.state{:}] == 'E');
    if (~isempty(exposedNodes))
        for iNode = 1:length(exposedNodes)
            u = rand();
            if ((transitionRate(4) >= u) && (Network.changed(iNode) ==
                false))
                Network.state{exposedNodes(iNode)} = 'I';
```

```
                Network.changed(exposedNodes(iNode)) = true;
            end
        end
    end

    % Infected nodes, 4 possiblities: I -> H, F, R, D
    newStates = {'H', 'F', 'R', 'D'};
    infectedNodes = datasample(s, find([Network.state{:}] == 'I'),
        Network.populationscount(3), 'Replace', false); % Infected nodes
        indices in random order
    if (~isempty(infectedNodes))
        for iPoss = 1:4
            infectedNodesFrac =
                infectedNodes(cumPopFraction(iPoss):cumPopFraction(iPoss +
                1));
            for iNode = 1:length(infectedNodesFrac)
                u = rand();
                if ((transitionRate(iPoss + 4) >= u) &&
                    (Network.changed(infectedNodesFrac(iNode)) == false))
                    Network.state{infectedNodesFrac(iNode)} =
                        newStates{iPoss};
                    Network.changed(infectedNodesFrac(iNode)) = true;
                end
            end
        end
    end

    % Hospitalized nodes, 2 possibilities: H -> R, D
    newStates = {'R', 'D'};
    hospitalizedNodes = datasample(s, find([Network.state{:}] == 'H'),
        Network.populationscount(4), 'Replace', false); % Hospitalized
        nodes indices in random order
    if (~isempty(hospitalizedNodes))
        for iPoss = 1:2
            hospitalizedNodesFrac = hospitalizedNodes(cumPopFraction(iPoss
                + 5):cumPopFraction(iPoss + 6));
            for iNode = 1:length(hospitalizedNodesFrac)
                u = rand();
                if ((transitionRate(iPoss + 8) >= u) &&
                    (Network.changed(hospitalizedNodesFrac(iNode)) ==
                    false))
                    Network.state{hospitalizedNodesFrac(iNode)} =
                        newStates{iPoss};
                    Network.changed(hospitalizedNodesFrac(iNode)) = true;
                end
            end
```

```
            end
        end

        % Buried (not safely, F) nodes, 1 possibility: F -> Rd
        buriedNodes = find([Network.state{:}] == 'F');
        if (~isempty(buriedNodes))
            for iNode = 1:length(buriedNodes)
                u = rand();
                if ((transitionRate(11) >= u) &&
                    (Network.changed(buriedNodes(iNode)) == false))
                    Network.state{buriedNodes(iNode)} = 'D';
                    Network.changed(buriedNodes(iNode)) = true;
                end
            end
        end
    end
    %% 3. Exporting data

    % Counting the population
    for iPop = 1:7
        Network.populationscount(iPop) = sum(ismember(Network.state,
            Network.populations{iPop}));
    end
    % Print out results
    PrintOutput(iTemp, Network.populationscount, Network.cumcases, fileID)

    % Reset graph
    clear Network.active Network.edges Network.changed
    Network.active = false(1,N);
    Network.edges = spalloc(N,N,1);
    Network.changed = false(1,N);

  end
end
```

C.5 RateComputation.m

```
function [popFraction, transitionRate] = RateComputation(iTemp, Outbreak)
  % Input: - iTemp: Time [days]
  %
  % Computes fraction of population susceptible to changes from one state
  % to another with the associated rate of transition.

  % Time-invariant parameters
```

```
lambdaI = 0.16;
lambdaF = 0.49;
muEI    = 0.09;   % [1/days]
muIF    = 0.13;   % [1/days]
muIRr   = 0.13;   % [1/days]
muIRd   = 0.13;   % [1/days]
muHRr   = 0.22;   % [1/days]
muHRd   = 0.24;   % [1/days]
muFRd   = 0.5;    % [1/days]
deltaIRr = 0;
deltaHRr = 0.46;
deltaHRd = 0.54;

% Time-varying parameters
if(strcmp(Outbreak,'Aug'))
    timePhase1 = 150;
    timePhase2 = 151;
elseif(strcmp(Outbreak,'May'))
    timePhase1 = 5;
    timePhase2 = 9;
end
if(iTemp < timePhase1)
    lambdaH = 0.33;
    muIH    = 0.10; % [1/days]
    deltaIH = 0.51;
    deltaIF = 0.10;
    deltaIRd = 0.39;
elseif(iTemp < timePhase2)
    lambdaH = 0.02;
    muIH    = 0.20; % [1/days]
    deltaIH = 0.80;
    deltaIF = 0.05;
    deltaIRd = 0.15;
else
    lambdaH = 0.02;
    muIH    = 0.43; % [1/days]
    deltaIH = 0.89;
    deltaIF = 0.01;
    deltaIRd = 0.10;
end

popFraction = [deltaIH ; ...  % Fraction of I -> H
               deltaIF ; ...  % Fraction of I -> F
               deltaIRr ; ... % Fraction of I -> Rr (Recovered)
               deltaIRd ; ... % Fraction of I -> Rd (Dead and buried)
               deltaHRr ; ... % Fraction of H -> Rr
```

```
                deltaHRd ]';    % Fraction of H -> Rd

    transitionRate = [lambdaI ; ... % Probability of infection by I
                      lambdaH ; ... % Probability of infection by H
                      lambdaF ; ... % Probability of infection by F
                      muEI    ; ... % Rate of E -> I
                      muIH    ; ... % Rate of I -> H
                      muIF    ; ... % Rate of I -> F
                      muIRr   ; ... % Rate of I -> Rr
                      muIRd   ; ... % Rate of I -> Rd
                      muHRr   ; ... % Rate of H -> Rr
                      muHRd   ; ... % Rate of H -> Rd
                      muFRd   ]';   % Rate of F -> Rd

end
```

C.6 PrintOutput.m

```
function PrintOutput(iTemp, count, cumcases, fileID)
       % Input: - iTemp: Time [days]
       %        - count: Number of people in each population
       %        - cumcases: Cumulative number of cases
       %        - fileID: Reference to output file
       %
       % Print in a text file referenced by fileID population count at given
           time step
    fprintf(fileID,'%8.2d %8.2d %8.2d %8.2d %8.2d %8.2d %8.2d %8.2d
        %8.2d\n',iTemp, ...
           count(1), ...
           count(2), ...
           count(3), ...
           count(4), ...
           count(5), ...
           count(6), ...
           count(7), ...
           cumcases);
end
```

C.7 OutputInterpreter.m

```
function OutputInterpreter(rootDir, relDir, test, tEnd, numberTest, m_range,
    etaSE_range, etaI_range)
```

```
    % Input: - rootDir: Absolute path to directory
    %        - relDir: Relative path to output directory
    %        - test: Light test or full test
    %        - tEnd: Total number of days of the simulation
    %        - numberTest: Number of trials
    %        - m_range: Values of m (fit)
    %        - etaSE_range: Values of etaSE (fit)
    %        - etaI_range: Values of etaI (fit)
    %
    % Reads output files contained in folder OutputEbola. Creates graph
    % of cumulative number of cases for light test.

%% 1. Light test
if(strcmp(test,'LightTest'))
    Data_Aug2018 = load('Data_Aug2018.mat');
    filePattern = fullfile(rootDir, relDir,'Data_LightTest*.txt');
    theFiles = dir(filePattern);
    Cases = zeros(tEnd+1, 7, numberTest);

    for iFile = 1:length(theFiles)
        % Extract data
        Data = importdata(strcat('OutputEbola/', theFiles(iFile).name));
        Cases(:,:,iFile) = Data.data(:,3:9);
    end
    meanCases = reshape(mean(Cases,3),tEnd+1,7);

    set(groot,'DefaultTextInterpreter','LaTex');
    set(groot,'DefaultLegendInterpreter','LaTex');
    figure('Renderer', 'painters', 'Position', [300 100 1000 650])
    hold on
    c=get(gca,'colororder');
    plot((0:tEnd)', meanCases(:,1),'Color', c(1,:))
    plot((0:tEnd)', meanCases(:,2),'Color', c(2,:))
    plot((0:tEnd)', meanCases(:,3),'Color', c(3,:))
    plot((0:tEnd)', meanCases(:,4),'Color', c(4,:))
    plot((0:tEnd)', meanCases(:,5),'Color', c(5,:))
    plot((0:tEnd)', meanCases(:,6),'k')
    plot((0:tEnd)', meanCases(:,7),'b')
    plot((1:tEnd)', Data_Aug2018.Data(:,2), 'b--')
    plot((1:tEnd)', Data_Aug2018.Data(:,3), 'k--')
    xlabel('Time [days]')
    ylabel('Population [unit]')
    title('Evolution of Ebola infection over time')
    legend('Exposed', 'Infected', 'Hospitalized', 'Buried', 'Recovered',
        'Safely buried', 'Cumulative cases','Infected Data','Death
        Data','Location','Best')
```

```
    set(gca, 'Ticklabelinterpreter', 'LaTex', 'FontSize', 16)
end

%% 2. Full test
if(strcmp(test,'Comparison'))
    Data_May2018 = load('Data_May2018.mat');
    filePattern = fullfile(rootDir, relDir,'Data_TestComp*.txt');
    theFiles = dir(filePattern);
    Cases = zeros(tEnd+1, 7, numberTest);

    for iFile = 1:length(theFiles)
        % Extract data
        Data = importdata(strcat('OutputEbola/', theFiles(iFile).name));
        Cases(:,:,iFile) = Data.data(:,3:9);
    end
    meanCases = reshape(mean(Cases,3),tEnd+1,7);

    set(groot,'DefaultTextInterpreter','LaTex');
    set(groot,'DefaultLegendInterpreter','LaTex');
    figure('Renderer', 'painters', 'Position', [300 100 1000 650])
    hold on
    c=get(gca,'colororder');
    plot((0:tEnd)', meanCases(:,1),'Color', c(1,:))
    plot((0:tEnd)', meanCases(:,2),'Color', c(2,:))
    plot((0:tEnd)', meanCases(:,3),'Color', c(3,:))
    plot((0:tEnd)', meanCases(:,4),'Color', c(4,:))
    plot((0:tEnd)', meanCases(:,5),'Color', c(5,:))
    plot((0:tEnd)', meanCases(:,6),'k')
    plot((0:tEnd)', meanCases(:,7),'b')
    plot((1:tEnd)', Data_May2018.Data(:,2), 'b--')
    plot((1:tEnd)', Data_May2018.Data(:,3), 'k--')
    xlabel('Time [days]')
    ylabel('Population [unit]')
    title('Evolution of Ebola infection over time')
    axis([0 80 0 110])
    legend('Exposed', 'Infected', 'Hospitalized', 'Buried', 'Recovered',
        'Safely buried', 'Cumulative cases','Infected Data','Death
        Data','Location','Best')
    set(gca, 'Ticklabelinterpreter', 'LaTex', 'FontSize', 16)
end

if(strcmp(test,'FullTest'))
    Data_Aug2018 = load('Data_Aug2018.mat');
    MeanCount =
        zeros(length(m_range),length(etaSE_range),length(etaI_range),tEnd+1,9);
```

```
J = zeros(length(m_range),length(etaSE_range),length(etaI_range)); %
    Function to minimize

for im = 1:length(m_range)
    for iSE = 1:length(etaSE_range)
        for iI = 1:length(etaI_range)
            filePattern = fullfile(rootDir, relDir, strcat(['data_m'
                num2str(m_range(im)) '_SE' num2str(etaSE_range(iSE)) '_I'
                num2str(etaI_range(iI)) '*.txt']));
            theFiles = dir(filePattern);

            % Extract data
            Count = zeros(tEnd+1,9,numberTest);
            for iFile = 1:numberTest
                data = importdata(strcat('OutputEbola/',
                    theFiles(iFile).name));
                Count(:,:,iFile) = data.data;
            end
            MeanCount(im,iSE,iI,:,:) = mean(Count,3);
        end
    end
end

for im = 1:length(m_range)
    for iSE = 1:length(etaSE_range)
        for iI = 1:length(etaI_range)
            computedCases = reshape(MeanCount(im,iSE,iI,:,9), 1, tEnd+1);
            computedDeath = reshape(MeanCount(im,iSE,iI,:,8), 1, tEnd+1);
            if ((im == 3) && (iSE == 3) && (iI == 1))
                figure
                hold on
                plot((1:tEnd)', computedCases(2:end)','b')
                plot((1:tEnd)', Data_Aug2018.Data(:,2), 'b--')
                plot((1:tEnd)', computedDeath(2:end)','k')
                plot((1:tEnd)', Data_Aug2018.Data(:,3), 'k--')
                title(strcat(['m = ' num2str(m_range(im)) ',SE = '
                    num2str(etaSE_range(iSE)) ',I = '
                    num2str(etaI_range(iI))]))
            end
            J(im,iSE,iI) = (1/(tEnd+1))*(sum((Data_Aug2018.Data(:,2) -
                computedCases(2:end)').^2) ...
                                    + sum((Data_Aug2018.Data(:,3) -
                                        computedDeath(2:end)').^2));
        end
    end
end
```

```
        [indA,indB] = find(J == min(J(:)),1);
        resultsIndex(1) = fix(indB/length(etaI_range)) + 1;
        resultsIndex(2) = indA;
        if (mod(indB,length(etaI_range)) == 0)
            resultsIndex(3) = length(etaI_range);
            resultsIndex(1) = resultsIndex(1) - 1;
        else
            resultsIndex(3) = mod(indB,length(etaI_range));
        end
        save('J.mat','J')
        Parameters = [m_range(resultsIndex(2)) etaSE_range(resultsIndex(3))
            etaI_range(resultsIndex(1))];
        disp(strcat(['m = ' num2str(Parameters(1))]))
        disp(strcat(['etaSE = ' num2str(Parameters(2))]))
        disp(strcat(['etaI = ' num2str(Parameters(3))]))
    end
end
```