ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Zombie Outbreak: The Effect of Inter-State Collaboration on the Survival of Humanity

by

Matthieu G. Mottet
Basile I. M. Wicky

Zurich
December 2012

Agreement for free-download

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

Matthieu G. Mottet Basile I. M. Wicky

Contents

1 Abstract

Investigation of the application of the SIR model to a zombie outbreak has already been studied, raising the fear of dark days for humanity. However, we would like to deepen this investigation to a multi-state system to see how interactions between subpopulation may brighten the future of the human race. Moreover, we are interested in seeing to what extent the different paradigms of international politics, Realpolitik, Liberalism and Neoconservatism as defined by Daniel W. Drezner in Theories of International Politics and zombies may lead to different outcomes.

2 Individual contributions

M.G.M. and B.I.M.W. formulated the question in mathematical terms and discussed the implementation in MATLAB. M.G.M. wrote the code. M.G.M. and B.I.M.W. analysed and discussed the results and B.I.M.W. wrote the report.

3 Acknowledgments

We wish to thank Karsten Donnay and Stefano Bielleti for there support in our work, fruitful discussions and an open mind to accept such a project. We would also like to thank the chair of sociology for the computational support provided with simulation time on Brutus.

4 Introduction and Motivations

While models of international relationship have already been studied under different pressure components, the effect of a zombie outbreak on international collaboration and equilibrium is a question that has been underestimated and was never addressed to the best of our knowledge. It is remarkable that the effect of such an intense event has not been looked at, although the fear of zombies and the threat they represent is vivid for many of us as reflected by the importance of the zombie culture. zombies, compared to other unnatural creatures such as vampires or aliens, have the very peculiar property not to be a minority inside the human civilization but rather to be in a way a part of it, just not quite as it was, i.e. zombified. Accordingly, zombies cause a much more deep-rooted fear as they not only threaten our lifes but also the our sense of identity as it questions our notion of what humanity is. The psychological effect of such a non-standard threat as well as the repercussion on the behaviour of large popluation systems such as states should be far from trivial. Accordingly, we decided to simulate the inter-state collaboration models in order to see the outcome on a large scale of such an extreme event. While some people might question the validity of such a study (no zombie has been observed so far), we think that the applicablity of such a reasoning could extend to a more probable large-scale epidemological event or simply, give a line of reasoning to cope with what former U.S. Secretery of Defense Donald Rumsfeld referred as the "unknown unknowns" of international security. Zombies might not be real, but the threat and stress they could impose on current world politics is.

5 Description of the Model

Figure 1: Schematic of our model including all variables. The parts in blue represent the fluxes at the micro-state level while the parts in orange represent the macro-state population fluxes. This schematic represent our simulated system where the interaction of three states (micro-states) is modeled

$$\Delta S_i^{micro} = -\alpha S_i Z_i - \gamma S_i Z_i = -(\alpha + \gamma) S_i Z_i \tag{1}$$

$$\Delta Z_i^{micro} = +\alpha S_i Z_i - \beta S_i Z_i = (\alpha - \beta) S_i Z_i \tag{2}$$

$$\Delta R_i^{micro} = +\beta S_i Z_i + \gamma S_i Z_i = (\beta + \gamma) S_i Z_i \tag{3}$$

$$\Delta S_i^{macro} = -\nu S_i \Delta S_i + \sum_{j \neq i} \nu S_j \Delta S_j \tag{4}$$

$$\Delta Z_i^{macro} = -\eta Z_i \tanh(\frac{Z_i}{S_i}) + \sum_{j \neq i} \eta Z_j \tanh(\frac{Z_j}{S_j}) \tag{5}$$

For each state (microstate), we define a SZR model that evaluates the evolution of the different populations under studies: susceptibles (S), zombies (Z) and removed (R). Epidemological-like transfer of populations between the states (at the macrostate level) also occurs as defined above and models the refugees and zombie transfer across states. Those transfers are parameters dependants, which depend on the cost-hypothesis as defined by the Game-Theoretical paradigm implemented. The apparition of zombies in one state will start the game. Each state will then evolve on the domestic and international level. The domestic level will follow a standard SZR model, whereas the international level will introduce exchange in the population of suceptible and zombie between the states. These exchanges will be influenced by the state decisions on foreign policies such as humanitarian or military actions determined by our game theory framework. The Game-Theoretical framework is defined as the possibility of undertaking military action of foreign soil (exporting S) or changing the refugee politics by modifying the mu parameter (allowing more S to come into one's state, and with a collateral cost of having more zombies crossing as well). Each action will be defined with a specific payoff, which in turn will depend on the international cooperation system under scrutiny. For simplicity, we will only model homogenous systems, i.e. all the states will adopt the same international politics paradigm. Finally, we will also introduce a "feedback" loop on the payoff depending on the success of a previously undertaken action (positive or negative affectation of the payoffs). This effect models the psychological effect of a successful or unsuccessful action on future action, for example the effectiveness of a military attack. This effect will be made as to converge after a certain time to model the wearing out of the psychological effect over time. The system will be implemented as a step-based update. This implies the ignorance of the actors (the states) of the action of the other actors. This rationalisation comes as the idea that the outbreak would occur over a short period of time, forcing for rapid decision-making and therefore not allow a reaction-based decision-making process.

6 Implementation

Research Methods

We would like to tackle this question by using a two-level model. Intra-state populations would be modelled by a standard SIR (Kermack-McKendrick) model

or an evolution of it that might include quarantined poplutions and more evolved parameters. On the next level, the inter-state relationships would be modelled using Game-Theory under different cost-hypothesis related to the main paradigms of international relationships as previously defined. Other

The material necessary for defining our models of zombies and 'zombification' parameters will be extracted from the canon of the zombie popular culture such as World War Z (Brooks, 2006), 28 Days Later (Boyle, 2002), The Night of the Living Dead (Romero, 1968), Zombieland (Fleischer, 2009), Resident Evil (Capcom, 1996), etc.

7 Simulation Results and Discussion

Fundamental Questions

Investigation of the application of the SIR model to a zombie outbreak has already been studied, raising the fear of dark days for humanity. However, we would like to deepen this investigation to a multi-state system to see how interactions between subpopulation may brighten the future of the human race. Moreover, we are interested in seeing to what extent the different paradigms of international politics, Realpolitik, Liberalism and Neoconservatism as defined by Daniel W. Drezner in Theories of International Politics and zombies may lead to different outcomes.
Expected Results

As describe in Drezner's book, we except different equilibrium outcomes depending on the paradigm under consideration. He postulates the possibile appearance of zombie states under Realpolitik and Liberalism paradigms while Neoconservatism would not allow such an outcome.

8 Summary and Outlook

blablabla[1]

9 References

[1] T.C. Reluga. An sis epidemiology game with two subpopulations. *Journal of Biological Dynamics*, 3(5):515–531, 2009.

[2] P.G. Bennett. Modelling decisions in international relations: game theory and beyond. *Mershon International Studies Review*, pages 19–52, 1995.

[3] D. Balcan and A. Vespignani. Phase transitions in contagion processes mediated by recurrent mobility patterns. *Nature physics*, 7(7):581–586, 2011.

[4] S. Funk, M. Salathé, and V.A.A. Jansen. Modelling the influence of human behaviour on the spread of infectious diseases: a review. *Journal of The Royal Society Interface*, 7(50):1247–1256, 2010.

[5] T.C. Reluga. Game theory of social distancing in response to an epidemic. *PLoS computational biology*, 6(5):e1000793, 2010.

[6] P. Munz, I. Hudea, J. Imad, and R.J. Smith. When zombies attack!: mathematical modelling of an outbreak of zombie infection. *Infectious Disease Modelling Research Progress. Hauppauge NY: Nova Science Publishers*, pages 133–150, 2009.

[7] D.W. Drezner. *Theories of international politics and zombies*. Princeton University Press, 2011.

10 Appendix

10.1 outbreak.m

```
function dump = outbreak( varargin )

    %
    % OUTBREAK runs the simulation of a zombie oubreak in a 3 states
    %   system.
    %
    %   DUMP = OUTBREAK( ARGS )
    %       ARGS are name-value pairs. Possible arguments are :
    %           params: structure containing rate parameters (alpha, beta,
    %                   gamma, eta, nu) as arrays (3x1 for alpha, beta and
    %                   gamma or 3x2 for eta and nu).
    %           paramfile: path to a mat-file storing the parameter
    %                   structure.
    %           zombies: 3x1 array containing the number of initial zombie
    %                   per state (default: 0, 0, 0).
    %           population: 3x1 array containing the number of initial
    %                   susceptible population per state (default: 1000,
    %                   1000, 1000).
    %           steps: maximal number of steps for the simulation (default:
    %                   1e8).
    %           silent:  level of output for the simulation:
    %                           0 = all outputs
    %                           1 = all outputs excepts graphs
    %                           2 = no outputs
    %           dslength: size of the sliding window used for the
    %                   calculation of the mean of dS (used as weight
    %                   factor for the inter-state susceptible transfer).
    %
    %       DUMP is a structure containing the following fields:
```

```
30      %           S: 4xn array containing the evolution of the susceptible
31      %              population.
32      %           dS: 4xn array containing the evolution of the susceptible
33      %              population's variation.
34      %           Z: 4xn array containing the evolution of the zombie
35      %              population.
36      %           dZ: 4xn array containing the evolution of the zombie
37      %              population's variation.
38      %           R: 4xn array containing the evolution of the removed
39      %              population.
40      %           dR: 4xn array containing the evolution of the removed
41      %              population's variation.
42      %           step: number of steps performed.
43      %           alpha, beta, gamma, eta, nu: transfer rate parameters.
44      %           time: simulation time.
45      %
46
47
48      %% Variable initialization
49      silent = 0;
50      dslength = 10;
51      maxSteps = 1e8;
52      rates = struct( 'alpha', [ 0.00095 ; 0.00095 ; 0.00095 ], 'beta', ...
            [0.00025 ; 0.00025 ; 0.00025 ], 'gamma', [ 0.00005 ; 0.00005 ; ...
            0.00005 ], 'mu', [ 0.00000005 , 0.00000005 ; 0.00000005 , ...
            0.00000005 ; 0.00000005 , 0.00000005 ], 'nu', [ 0.00000005 , ...
            0.00000005 ; 0.00000005 , 0.00000005 ; 0.00000005 , 0.00000005 ], ...
            'eta', [ 0.00000005 , 0.00000005 ; 0.00000005 , 0.00000005 ; ...
            0.00000005 , 0.00000005 ] );
53      zombies = [ 0 ; 0 ; 0 ; 0 ];
54      populations = [ 1e3 ; 1e3 ; 1e3 ; 3e3 ];
55
56      % Structure storing populations and populations variation
57      states = struct( 'pop', zeros( 4, 3 ), 'dpop', zeros( 4, 3 ) );
58
59      % See exit condition at the end of the loop.
60      exitThreshold = 1e-1;
61
62      p = [ 'alpha' ; 'beta ' ; 'gamma' ; 'eta  ' ; 'mu   ' ; 'nu   ' ];
63      reverseStr = '';
64
65
66      %% Argument parsing
67      % All argument are optional.
68      for i = 1:2:nargin
69
70          % Argument are passed in name-value pairs. If i + 1 does not exist,
71          % then the value is absent and the program stops.
72          if( nargin < i + 1 )
73
74              error( [ 'Missing argument for parameter "' varargin{ i } ...
                        '".' ] );
75          end
```

```
76
77          switch varargin{ i }
78
79              % params excepts a structure containing the rates values for the
80              % epidemiologic model. Not all fields have to be present. If a
81              % field is missing, the default value is used.
82              case 'params'
83                  for j = 1:6
84
85                      if isfield( varargin{ i + 1 }, strtrim( p( j, : ) ) )
86
87                          rates.( strtrim( p( j, : ) ) ) = varargin{ i + 1 ...
                                }.( strtrim( p( j, : ) ) );
88                      end
89                  end
90
91              % paramfile excepts the path to a mat-file containing a
92              % parameter structure (refer to parms).
93              case 'paramfile'
94                  pfile = load( varargin{ i + 1 }, '-mat' );
95                  for j = 1:6
96
97                      if isfield( pfile{ i + 1 }, strtrim( p( j, : ) ) )
98
99                          rates.( strtrim( p( j ) ) ) = pfile{ i + 1 }.( ...
                                strtrim( p( j ) ) );
100                     end
101                 end
102
103             % zombies excepts an array containing the initial zombie
104             % populations in each state.
105             case 'zombies'
106                 temp = varargin{ i + 1 };
107                 for j = 1:3
108
109                     zombies( j ) = temp( j );
110                     zombies( 4 ) = zombies( 4 ) + temp( j );
111                 end
112
113             % zombies excepts an array containing the initial susceptible
114             % population in each state.
115             case 'population'
116                 temp = varargin{ i + 1 };
117                 populations( 4 ) = 0;
118                 for j = 1:3
119
120                     populations( j ) = temp( j );
121                     populations( 4 ) = populations( 4 ) + populations( j );
122                 end
123
124             % steps excepts the maximum number of steps allowed for the
125             % simulation.
126             case 'steps'
```

```
127                 maxSteps = varargin{ i + 1 };
128
129             % silent except a int defining the output type :
130             % 2 : No output
131             % 1 : All outputs but the graphs
132             % 0 : Normal output
133             case 'silent'
134                 silent = varargin{ i + 1 };
135
136             % dslength excepts the size of the sliding window used for the
137             % calculation of the mean of dS (used as weight factor for the
138             % inter-state susceptible transfer).
139             case 'dslength'
140                 dslength = varargin{ i + 1 };
141
142             otherwise
143                 error( [ 'Unknown parameter "', varargin{ i }, '".' ] );
144         end
145     end
146
147
148
149     %% Initialization of simulation
150
151     % Setting the initial populations.
152     states.pop( :, 1 ) = populations;
153     states.pop( :, 2 ) = zombies;
154
155     % Initialization of the sliding window for inter-state population
156     % transfer.
157     dshistory = zeros( 3, dslength );
158
159     % Setting up the initial dump structure. Large matrices are used to
160     % avoid the need of dynamic memory allocation to improve code
161     % efficency.
162     d = zeros( 4, .001 * maxSteps );
163     dump = struct( 'S', d, 'dS', d, 'Z', d, 'dZ', d, 'R', d, 'dR', d, ...
            'step', 1, 'alpha', mean( rates.alpha ), 'beta', mean( rates.beta ...
            ), 'gamma', mean( rates.gamma ), 'eta', mean( mean( rates.eta ) ...
            ), 'nu', mean( mean( rates.nu ) ), 'time', 0 );
164     dump.S( :, 1 ) = states.pop( 1:4, 1 );
165     dump.Z( :, 1 ) = states.pop( 1:4, 2 );
166     dump.R( :, 1 ) = states.pop( 1:4, 3 );
167
168     % Initialize timer.
169     tic;
170
171     %% Update loop.
172     for i = 1:maxSteps
173
174         % Output of step # and current population count.
175         if silent < 2
176
```

```
177             msg = sprintf('Processing step %d... Human population %d, ...
                    Zombie population %d\n', i, dump.S( 4, dump.step ), ...
                    dump.Z( 4, dump.step ) );
178             fprintf([ reverseStr, msg]);
179 
180             reverseStr = repmat(sprintf('\b'), 1, length(msg));
181         end
182 
183         % Update function (see update.m for details).
184         [ states, rates, dshistory ] = update( states, rates, dshistory );
185 
186         % Dumping of the latest state of the simulation.
187         % Update of the step number.
188         dump.step = i + 1;
189 
190         % Dumping of the various population and population variation.
191         dump.S( :, dump.step ) = states.pop( :, 1 );
192         dump.dS( :, dump.step ) = states.dpop( :, 1 );
193         dump.Z( :, dump.step ) = states.pop( :, 2 );
194         dump.dZ( :, dump.step ) = states.dpop( :, 2 );
195         dump.R( :, dump.step ) = states.pop( :, 3 );
196         dump.dR( :, dump.step ) = states.dpop( :, 3 );
197 
198         % Stops simulation if S is equal to 0.
199         if states.pop( 4, 1 ) == 0
200 
201             if silent < 2
202 
203                 disp( ' ' );
204                 disp( 'End of the human race.' );
205             end
206             break;
207         end
208 
209         % Stops simulation if Z is equal to 0.
210         if states.pop( 4, 2 ) == 0
211 
212             if silent < 2
213 
214                 disp( ' ' );
215                 disp( 'Humanity survived.' );
216             end
217             break;
218         end
219 
220         % Stops simulation if the average fluctuation of both dS and dZ
221         % over a sliding window is lower than the threshold (simulation
222         % reached equilibrium in the limit of our time frame).
223         if i > 100 && mean( abs( dump.dS( 4, ( i - 100 ):i ) ) ) < ...
                exitThreshold && mean( abs( dump.dZ( 4, ( i - 100 ):i ) ) ) < ...
                exitThreshold
224 
225 
```

```
226             if silent < 2
227 
228                 disp( ' ' );
229                 disp( 'Equilibrium reached.' );
230             end
231             break;
232         end
233     end
234 
235     % Store simulation time.
236     dump.time = toc;
237 
238     % Resizing of the dump matrices before plotting.
239     dump.S = dump.S( :, 1:dump.step );
240     dump.dS = dump.dS( :, 1:dump.step );
241     dump.Z = dump.Z( :, 1:dump.step );
242     dump.dZ = dump.dZ( :, 1:dump.step );
243     dump.R = dump.R( :, 1:dump.step );
244     dump.dR = dump.dR( :, 1:dump.step );
245 
246     if ¬silent
247 
248         % Plot the simulation history (see plotResults.m for details).
249         plotResults( dump );
250     end
251 end
```

10.2 update.m

```
1     function [ states, rates, dshistory ] = update( states, rates, ...
          dshistory )
2 
3     %
4     % UPDATE compute the population evolution for one step of the
5     %   epidemiological model.
6     %
7     % [ STATES, RATES, DSHISTORY, DUMP ] = UPDATE( STATES, RATES, ...
          DSHISTORY, DUMP )
8     %       STATES is the structure storing the populations and population
9     %           variations as defined in the OUTBREAK function.
10    %       RATES is the structure containing the epidemiological transfer
11    %           rates as defined in the OUTBREAK function.
12    %       DSHISTORY is the array containing the sliding window for
13    %           inter-state susceptible transfer.
14    %
15 
16    %% Initialization of parameters :
17 
18    s = states.pop( 1:3, 1 );
```

```
19      z = states.pop( 1:3, 2 );
20
21      alpha = rates.alpha;
22      beta = rates.beta;
23      gamma = rates.gamma;
24      nu = rates.nu;
25      eta = rates.eta;
26
27      % Permutation matrix used to set the input inter-state flux from the
28      % inter-state output flux to avoid redundant calculations. See flux
29      % update.
30      permutation = [ 0 0 1 ; 1 0 0 ; 0 1 0 ];
31
32      % Matrix containing the coordinates of the input inter-state flux from
33      % each state (line) to the other two (block of two colons). See flux
34      % correction.
35      iFluxCoor = [ 2 2 3 1 ; 3 2 1 1 ; 1 2 2 1 ];
36
37      %% Update of the susceptible population.
38
39      % Depletion of the susceptible population goes according to :
40      %
41      %   S  -> Z  = - alpha * s * z
42      %   S  -> R  = - gamma * s * z
43      %   S1 -> S2 = nu * dSmean
44      %   S1 -> S3 = nu * dSmean
45      %
46      % Immigration is the only source of susceptible population.
47      %
48      % dS is a 3x6 matrix containing for each state :
49      %
50      %   Row 1 : n( S  -> Z  )
51      %   Row 2 : n( S  -> R  )
52      %   Row 3 : n( S1 -> S2 ) | n( S2 -> S3 ) | n( S3 -> S1 )
53      %   Row 4 : n( S1 -> S3 ) | n( S2 -> S1 ) | n( S3 -> S2 )
54      %   Row 5 : n( S2 -> S1 ) | n( S3 -> S2 ) | n( S1 -> S3 )
55      %   Row 6 : n( S3 -> S1 ) | n( S1 -> S2 ) | n( S2 -> S3 )
56      %
57      % The sum of each line gives the population variation for the state.
58
59      % Mean variation of the susceptible in the sliding window.
60      dsmean = mean( dshistory, 2 );
61
62      dS = zeros( 3, 6 );
63      dS( :, 1 ) = - alpha .* s .* z;
64      dS( :, 2 ) = - gamma .* s .* z;
65      dS( :, 3 ) = min( nu( :, 1 ) .* dsmean, 0 );
66      dS( :, 4 ) = min( nu( :, 2 ) .* dsmean, 0 );
67      dS( :, 5 ) = - permutation * permutation * dS( :, 4 );
68      dS( :, 6 ) = - permutation * dS( :, 3 );
69
70      % There is the possibility that any of the |-dSi| is larger than the
71      % actual population in the state (negative population fluctuation larger
```

```
72      % than the actual population). If so, a correction is applied to
73      % avoid negative population.
74      % As long as any state falls into this category:
75      while sum( sum( dS( :, 1:4 ), 2 ) + s + sum( dS( :, 5:6 ), 2 ) < ...
            -1e-4 )
76
77          for i = 1:3
78
79              % Correct the state that falls into this category:
80              if( sum( dS( i, 1:4 ), 2 ) + ( s( i ) + sum( dS( i, 5:6 ), 2 ...
                    ) ) < -1e-4 )
81
82                  % Population is considered equal to the sum of previous
83                  % population and the inter-state input transfer. The
84                  % negative contributions to the dS are reduced so that it
85                  % equals this population.
86                  dS( i, 1:4 ) = dS( i, 1:4 ) / abs( sum( dS( i, 1:4 ) ) ) ...
                        * ( sum( dS( i, 5:6 ) ) + s( i ) );
87
88                  % The effect of the correction on the output flux is
89                  % applied to the other state input flux.
90                  dS( iFluxCoor( i, 1 ), 4 + iFluxCoor( i, 2 ) ) = - dS( i, ...
                        3 );
91                  dS( iFluxCoor( i, 3 ), 4 + iFluxCoor( i, 4 ) ) = - dS( i, ...
                        4 );
92              end
93          end
94          % Note that the output flux can only be lower than the original flux.
95          % Therefore, the input flux for the two other states can only
96          % be smaller than before the correction. Accordingly, the
97          % process has to be repeated until all states have a population
98          % differential such that it doesn't lead to a negative population.
99      end
100
101     %% Update of the zombie population.
102
103     % Depletion of the zombie populations goes according to :
104     %
105     %   S  -> Z  = alpha * s * z
106     %   Z  -> R  = - beta * s * z
107     %   Z1 -> Z2 = - eta * z * tanh( z / s )
108     %   Z1 -> Z3 = - eta * z * tanh( z / s )
109     %
110     % Conversion from susceptible and "immigration" are the two sources of
111     % zombies.
112     %
113     % dZ is a 3x6 matrix containing for each states :
114     %
115     %   Row 1 : n( S  -> Z  )
116     %   Row 2 : n( Z  -> R  )
117     %   Row 3 : n( Z1 -> Z2 ) | n( Z2 -> Z3 ) | n( Z3 -> Z1 )
118     %   Row 4 : n( Z1 -> Z3 ) | n( Z2 -> Z1 ) | n( Z3 -> Z2 )
119     %   Row 5 : n( Z2 -> Z1 ) | n( Z3 -> Z2 ) | n( Z1 -> Z3 )
```

```
120     %   Row 6 : n( Z3 -> Z1 ) | n( Z1 -> Z2 ) | n( Z2 -> Z3 )
121     %
122
123     dZ = zeros( 3, 6 );
124     dZ( :, 1 ) = - dS( :, 1 );
125     dZ( :, 2 ) = - beta .* s .* z;
126     dZ( :, 3 ) = - eta( :, 1 ) .* z .* tanh( z ./ s );
127     dZ( :, 4 ) = - eta( :, 2 ) .* z .* tanh( z ./ s );
128     dZ( :, 5 ) = - permutation * permutation * dZ( :, 4 );
129     dZ( :, 6 ) = - permutation * dZ( :, 3 );
130
131     % The same control procedure as for the susceptible population is
132     % applied to avoid negative population of zombies. See susceptible
133     % correction for details on the procedure.
134     while sum( ( dZ( :, 1 ) + sum( dZ( :, 5:6 ), 2 ) + z ) + sum( dZ( :, ...
            2:4 ), 2 ) < - 1e-4 )
135
136         for i = 1:3
137
138             if( ( dZ( i, 1 ) + sum( dZ( i, 5:6 ), 2 ) + z( i ) ) + sum( ...
                    dZ( i, 2:4 ) ) < - 1e-4 )
139
140                 dZ( i, 2:4 ) = dZ( i, 2:4 ) * ( sum( dZ( i, 5:6 ) ) + dZ( ...
                        i, 1 ) + z( i ) ) / abs( sum( dZ( i, 2:4 ) ) );
141
142                 dZ( iFluxCoor( i, 1 ), 4 + iFluxCoor( i, 2 ) ) = - dZ( i, ...
                        3 );
143                 dZ( iFluxCoor( i, 3 ), 4 + iFluxCoor( i, 4 ) ) = - dZ( i, ...
                        4 );
144             end
145         end
146     end
147
148     %% Update of the removed population
149
150     % Variation in the removed population is strictly positive and comes
151     % from both the susceptible and zombie population:
152     %
153     %   S  -> R  = alpha * s * z
154     %   Z  -> R  = beta * s * z
155     %
156     % The value obtained for dS and dZ are used:
157
158     dR = - [ dS( :, 2 ), dZ( :, 2 ) ];
159
160     %% Update each population
161
162     % Compilation of the populations variations
163     states.dpop( 1:3, : ) = [ sum( dS, 2 ), sum( dZ, 2 ), sum( dR, 2 ) ];
164
165     % Update of the sliding window matrix for the susceptible.
166     [ ~, dsSize ] = size( dshistory );
167     dshistory( :, 2:dsSize ) = dshistory( :, 1:( dsSize - 1 ) );
```

```
168     dshistory( :, 1 ) = states.dpop( 1:3, 1 );
169
170     % Update of the current populations
171     states.pop( 1:3, : ) = states.pop( 1:3, : ) + states.dpop( 1:3, : );
172
173     % Update of the total populations
174     states.pop( 4, : ) = sum( states.pop( 1:3, : ) );
175     states.dpop( 4, : ) = sum( states.dpop( 1:3, : ) );
176 end
```

10.3 plotResults.m

```
1  function plotResults( dump )
2
3      % PLOTRESULTS plots the populations' evolution during the simulation. It
4      %   plots the different populations (S, Z, R) for each state as well as
5      %   the population variation. Furthermore the global populations and
6      %   population variations are also displayed.
7      %
8      % PLOTRESULTS( DUMP )
9      %   DUMP is the dumping structure as defined in the OUTBREAK function.
10     %
11
12     x = 0:( length( dump.S ) - 1 );
13
14     % For each of the three states, plot the populations and
15     % populations' variations.
16     for i = 1:3
17
18         subplot( 4, 4, ( i * 4 - 3 ):( i * 4 - 2 ) );
19         plot( x, dump.S( i, : ), 'g', x, dump.Z( i, : ), 'r', x, dump.R( ...
               i, : ), 'k' );
20         ylim( [ 0 dump.S( 4 ) ] );
21         xlabel( 'Step' );
22         ylabel( 'Population' );
23         title( [ 'State ', int2str( i ) ], 'fontweight', 'b' );
24         if i == 1
25
26             legend( 'Susceptibles', 'Zombies', 'Removed' );
27         end
28
29         subplot( 4, 4, ( i * 4 - 1 ):( i * 4 ) );
30         plot( x, dump.dS( i, : ), 'g', x, dump.dZ( i, : ), 'r', x, ...
               dump.dR( i, : ), 'k' );
31         xlabel( 'Step' );
32         ylabel( 'Population Variation' );
33         title( [ 'State ', int2str( i ) ], 'fontweight', 'b' );
34
35     end
```

```
36
37      % Plot the total populations and total populations' variations.
38      subplot( 4, 4, 13:14 );
39      plot( x, dump.S( 4, : ), 'g', x, dump.Z( 4, : ), 'r', x, dump.R( 4, : ...
            ), 'k' );
40      ylim( [ 0 dump.S( 4 ) ] );
41      xlabel( 'Step' );
42      ylabel( 'Population' );
43      title( 'World population', 'fontweight', 'b' );
44
45      subplot( 4, 4, 15:16 );
46      plot( x, dump.dS( 4, : ), 'g', x, dump.dZ( 4, : ), 'r', x, dump.dR( ...
            4, : ), 'k' );
47      xlabel( 'Step' );
48      ylabel( 'Population Variation' );
49      title( 'World population', 'fontweight', 'b' );
50
51
52  end
```