Carbapenemase-producing Klebsiella pneumoniae (CPKP) diffusion on a network

Alessandro Cotellucci

March 20, 2020

Abstract

The studies on the mathematical diffusion model of disease are very important to plan a strategy to avoid the contagion. These kind of models start by the SIS model and can be improved, an example is given by the model for the diffusion of the Carbapenemase-producing Klebsiella pneumoniae (CPKP) in a surgical unit. For this purpose the network study can be fundamental to understand the particular behaviour of the diffusion in a discreet system instead of the mean field approximation. This brief work try to seek this purpose.

Contents

Chapter 1

SIS model for epidemic diffusion

Diffusion of pathologies can be modelled using the SIS model which is a model involving two states. It can be easily implemented on a network by the fact that the network could be seen as a numerical simulation of the spread in a real network.

1.1 SIS model features

The SIS (Susceptible-Infected-Susceptible) is based on the idea that all the elements which are attacked by the pathology can have two possible state: susceptible and infected. If a susceptible elements is touched by a infected one than with a possibility λ the first one become infected:

$$S(i) + I(j) \xrightarrow{\lambda} I(i) + I(j). \tag{1.1}$$

There could be also a probability for the inverse process μ. We can make a mean field description by solving this system of differential equations:

$$\begin{aligned} \frac{ds(t)}{dt} &= \mu\rho(t) - \lambda\langle k\rangle\rho(t)s(t) \\ \frac{d\rho(t)}{dt} &= -\frac{ds(t)}{dt} \end{aligned} \tag{1.2}$$

in the condition of an isolated system and considering the initial fraction of infected nodes very little. $\langle k\rangle$ is the average number of contacts.
Using this equation we can get a estimation of the epidemic threshold by considering a steady state for ρ:

$$\begin{aligned} &\mu\rho(t) - \lambda\langle k\rangle\rho(t)(1-\rho(t)) = 0 \\ &\rho = 1 - \frac{\mu}{\lambda\langle k\rangle} \end{aligned} \tag{1.3}$$

so we can define two cases:

$$\begin{aligned} &\lambda < \frac{\mu}{\langle k\rangle} \Rightarrow \rho = 0 \quad \textit{No epidemic} \\ &\lambda > \frac{\mu}{\langle k\rangle} \Rightarrow \rho = 1 - \frac{\mu}{\lambda\langle k\rangle} \quad \textit{Endemic state} \end{aligned} \tag{1.4}$$

This can be seen in the figure 1.1 where one can observe the phase transition with order parameter ρ.

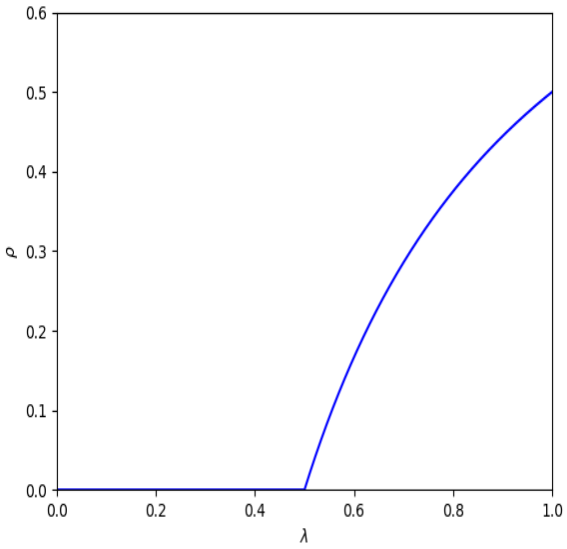


Figure 1.1: Typical behaviour of an order parameter in presence of a phase transition.

If we want to introduce a way to decrease the contagion we can add an immunization effect which decrease the transmission probability λ by a factor $(1-g)$. By doing this we can define a critical value for g over which we stop the contagion g_c.

1.2 SIS model for heterogeneous sample

$\langle k \rangle$ is a mean quantity but on the network the connectivity could follow a distribution $p(k)dk$, following this idea we can model the SIS diffusion on a heterogeneous sample. For an heterogeneous sample we have a differential equation for each degree k:

$$\begin{aligned} \frac{ds_k(t)}{dt} &= \mu\rho_k(t) - \lambda k s_k(t)\Theta(t) \\ \frac{d\rho_k(t)}{dt} &= -\frac{ds_k(t)}{dt} \end{aligned} \tag{1.5}$$

where:

$$\Theta = \frac{\sum_k kp(k)\rho_k(t)}{\sum_k kp(k)} \tag{1.6}$$

is the weighted average parameter
There is a big difference between the mean system and the heterogeneous system that can be seen by analyzing Θ, in fact we obtain:

$$\Theta = \frac{\sum_k kp(k)\frac{\lambda k\Theta}{1+\lambda k\Theta}}{\sum_k kp(k)} \tag{1.7}$$

so $\Theta = 0$ implies $\rho = 0$, the solution $\Theta > 0$ is given by:

$$\frac{d}{d\Theta}\left(\frac{\sum_k kp(k)\frac{\lambda k\Theta}{1+\lambda k\Theta}}{\sum_k kp(k)}\right)|_{\Theta=0} \geq 1 \tag{1.8}$$

so the critical value for λ will be: $\langle k \rangle / \langle k^2 \rangle$. This means that if the second momentum of the distribution diverges than the epidemic threshold goes to zero replacing the discontinuity of the figure 1.1 with a smooth function.

1.3 SIS model on network

The SIS model can be implemented on a network using a Markov chain. The following code is based on an event driven algorithm.

```
import math
import random
from networkx import *
from numpy import *

#Preparing the graph
G=fast_gnp_random_graph(20, 0.5, seed=None, directed=False)
State=['Susceptible']
set_node_attributes(G, State, 'State')

#Preparing the infection source
rand=random.randint(0,19)
G.nodes[0]['State']=['Infected']
infection_list=[]
infection_list.append(rand)

#Parameters
lambd=0.1
mu=0.1
Time=10

#Infection
for t in range(Time):

  #Decontamination
    for node in G.nodes():
        if G.nodes[node]['State']==['Infected']:
            rand_head=random.uniform(0,1)
            if rand_head<=mu:
                G.nodes[node]['State']=['Susceptible']

    for node in G.nodes():
        if G.nodes[node]['State']==['Infected']:
            neighbours=list(G.neighbors(i))
            for j in range(len(neighbours)):
                if Node_state[neighbours[j]]==['Susceptible']:
                    rand_inf=random.uniform(0,1)
                    if rand_inf<=lambd:
                        infection_list.append(neighbours[j])
    infect_list=list(set(infect_list))  #Keeping only the unique value
   for each infected node
    for i in range(len(infection_list)):
        G.nodes[infection_list[i]]['State']=['Infected']
```

Chapter 2

CPKP diffusion

In 2010 Vana Sypsa, Mina Psichogiou, Georgia-Aikaterina Bouzala, Linos Hadjihannas, Angelos Hatzakis1, Georgios L. Daikos conduced a study abouth the diffusion of the carbapenemase-producing Klebsiella pneumoniae (CPKP) in a surgical unit in order to understand the behaviour and the possible way to control the infection.

2.1 The model features

The model include two possible role: the HCW (health care worker) and the patient. The model is based on the scheme in figure 2.1.

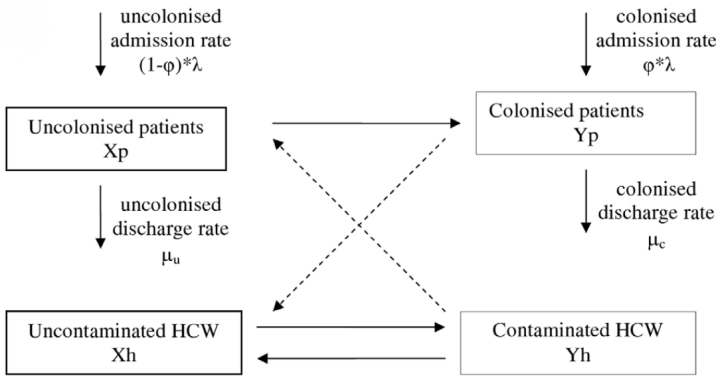


Figure 2.1: Scheme of CPKP diffusion from the article.

The control model scheme is shown by the figure 2.2.

I have decided to implement the model including the possibility of a direct contagion between the HCW to get a more general result. The modified model is described by the equations:

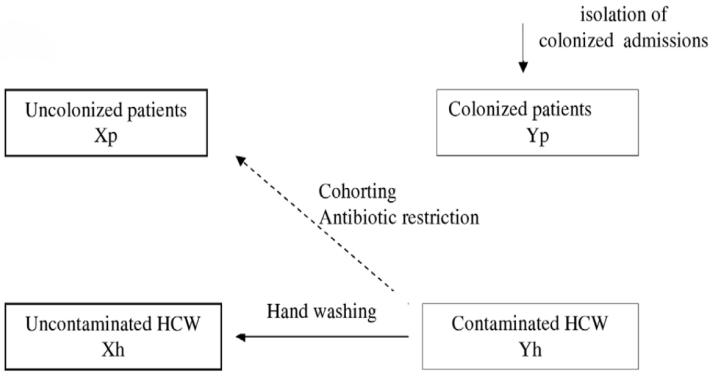


Figure 2.2: Scheme of CPKP diffusion with control mechanism from the article.

$$\begin{aligned}
\frac{dX_P(t)}{dt} &= (1-\varphi)\lambda(B - X_P - Y_P) - \mu_U X_P - \alpha b_P X_P Y_H \\
\frac{dY_P(t)}{dt} &= \varphi\lambda(B - X_P - Y_P) - \mu_C Y_P + \alpha b_P X_P Y_H \\
\frac{dX_H(t)}{dt} &= \mu_H Y_H - \alpha b_H Y_P X_H - \alpha_H b_{HH} X_H Y_H \\
\frac{dY_H(t)}{dt} &= -\mu_H Y_H + \alpha b_H Y_P X_H + \alpha_H b_{HH} X_H Y_H
\end{aligned} \tag{2.1}$$

Where X_P is the number of uncolonized patient, Y_P is the number of colonized patient, X_H is the number of uncolonized HCW, Y_H is the number of colonized HCW, λ is the probability to add a new patient, φ is the probability that the added patient is colonized, B is the number of bed, μ_U is the discharge rate for the uncolonized patient, μ_C is the discharge rate for colonized patient, μ_H is the probability of a colonized HCW to become uncolonized, α is the average connectivity of the patient-HCW interaction, b_P is the probability that a colonized patient become colonized interacting with a colonized HCW, b_H is the probability that a colonized HCW become colonized interacting with a colonized patient, b_{HH} is the probability that a colonized HCW become colonized interacting with a colonized HCW and α_H is the average connectivity of the HCW-HCW interaction. In the presence of hand disinfection, the probability of contamination from a patient to an HCW and between HCW's is reduced by $p\%$, where p denotes the hand hygiene compliance rate in the surgical unit. The new model will be:

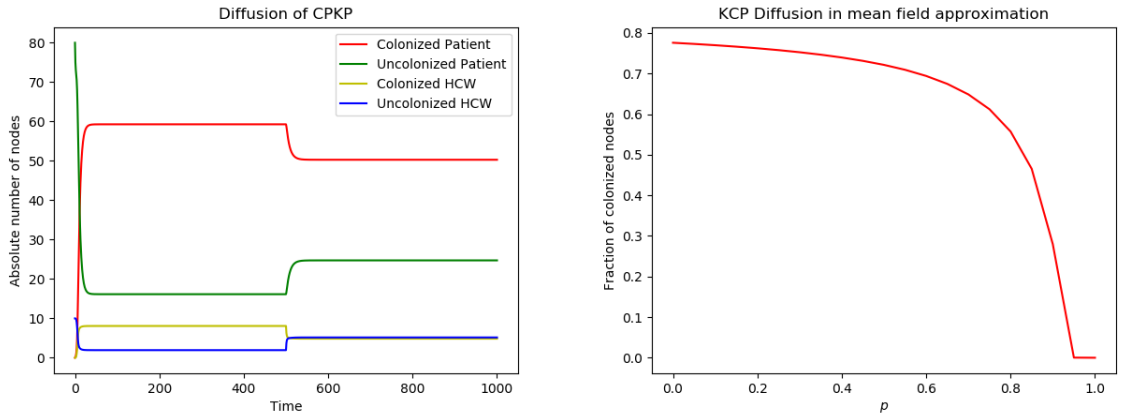
$$\begin{aligned}
\frac{dX_P(t)}{dt} &= (1-\varphi)\lambda(B - X_P - Y_P) - \mu_U X_P - \alpha b_P X_P Y_H \\
\frac{dY_P(t)}{dt} &= \varphi\lambda(B - X_P - Y_P) - \mu_C Y_P + \alpha b_P X_P Y_H \\
\frac{dX_H(t)}{dt} &= \mu_H Y_H - \alpha b_H (1-p) Y_P X_H - \alpha_H b_{HH} (1-p) X_H Y_H \\
\frac{dY_H(t)}{dt} &= -\mu_H Y_H + \alpha b_H (1-p) Y_P X_H + \alpha_H b_{HH} (1-p) X_H Y_H
\end{aligned} \tag{2.2}$$

Another feature that can be taken into account is the effect of antibiotics on colonization. Antibiotics may provide CPKP with a selective growth advantage that will result in higher probability of colonization. Thus, the probability b_P of a patient being colonized

per contact with a contaminated HCW can be decomposed into the product of a baseline probability b_{P0} and a factor f representing the impact of antibiotics. If r is the relative risk of colonization associated with these agents and patients receive them for a fraction d of their stay in the unit, then $f = 1 + d(r - 1)$. Antibiotic restriction policies could target to reducing the duration of administration of these agents from d to d'.

2.2 Mean field results

Solving the system of differential equations 2.2 the results are shown by the figure 2.3a with the critical value of p to avoid the contagion, the initial conditions where a pure uncolonized surgical unit that evolves in time.



(a) Absolute (b) Contagion threshold

Figure 2.3: Plots of the mean field results.

The figure 2.3a show the absolute number of patients and HCWs in the hypothesis that the control activities are used from the half of the observation time. These results are obtained by solving the differential equation system using a Runge-Kutta at the fourth order in the script $CPKP\_Mean\_field.py$ and $CPKP\_Mean\_field\_Analysis.py$.
As we can see the shape of the relative number of contaminated people as function of the probability of contagion follow the shape of a classical order parameter. The threshold for this model is very high (it depends on the parameters of the model) it is around the value of 1.
In the figure 2.3a we can see that the diffusion of the CPKP and the action of the control activities are very fast this is caused by the high values of the parameter but the value of $p(= 0.7)$ is not enough to avoid the contagion because, as it is shown by the Figure 2.3b, the value is below the threshold.

2.3 Application on the network

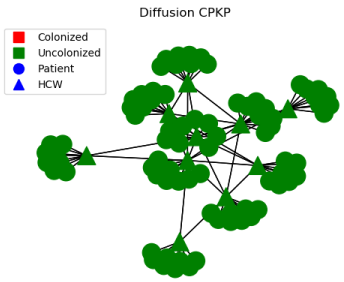
2.3.1 Different models for the network

In order to implement the model on the network I first had to decide a right network to model the surgical unit. First I decided to consider a big cluster network to model the HCW's interactions and the to add a limited number of patient for each HCW node, than I decide to try to consider a network in which the HCWs shift between the rooms, one time for each time step, in order to interact at least with all the possible patients, in this case the network between the HCWS is absent to simplify the simulation and to have a network which respected the original idea of the model and to reproduce maintenance crew of the HCWs in the surgical unit.
In all the networks I used 10 HCWs and for each of them 8 patients. The network are shown in the Figure 2.4:

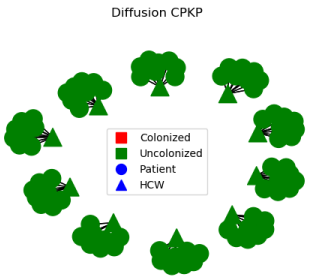
(a) the HCW's cluster has a Barabasi-Albert structure with number of edges to attach from a new node to existing nodes equal to 4

(b) the Moving-HCW structure with no network between the HCWs

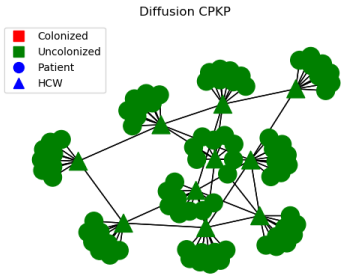
(c) the HCW's cluster has an Erdos-Renyi structure with probability of having a node 0.5.



(a) Barabasi-Albert



(b) Moving-HCW

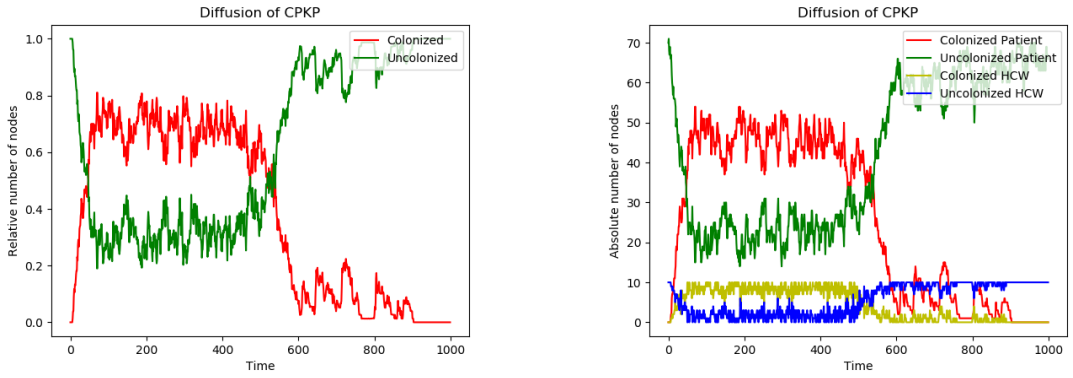


(c) Erdos-Renyi

Figure 2.4: Plots of the three networks to model the surgical unit.

2.3.2 Results on a network

The figure 2.5 shows the evolution of the diffusion respect to time, in the figure 2.5a we can see the evolution of the uncolonized and colonized fractions of nodes while in the figure 2.5b we have the evolution of patient and HCW's number. In this simulation the control of the contagion starts at half of the time interval of the observation. The two plots are the results of the code in appendix B. As we can see comparing these results



(a) Fraction

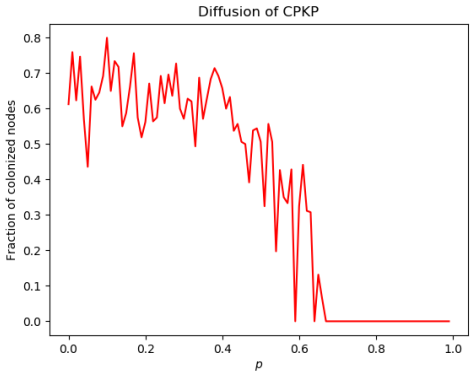
(b) Absolute

Figure 2.5: Plots of the evolution in time of the diffusion (the network of this simulation is the one with the Erdos-Renyi cluster) using the same parameters and initial conditions of the mean field simulation.

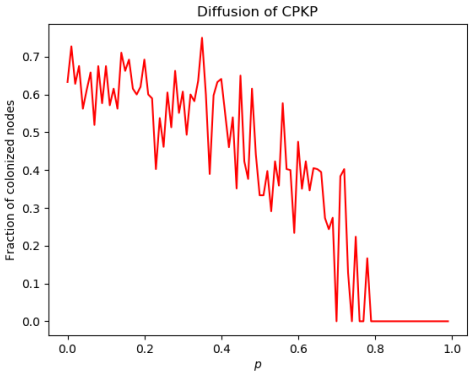
with the mean field one, shown by Figure 2.3a, the epidemic phase of the diffusion reach the same value for the number of colonized and uncolonized patients and HCWs, up to statistical fluctuations, both for network and mean field simulation. The differences comes when we introduce the control activities at half of the simulation time getting a different result because with the same hand hygiene rate in the surgical unit we get a less effect for the mean field result respect to the network. This could be caused by the particular structure of the network which is made by a cluster of HCWs that dominate the diffusion while the patients are only peripheral nodes, an effect is also given by the low number of nodes (90) which doesn't allow the mean field approximation.

2.3.3 Analysis of the effects of the topology on the threshold

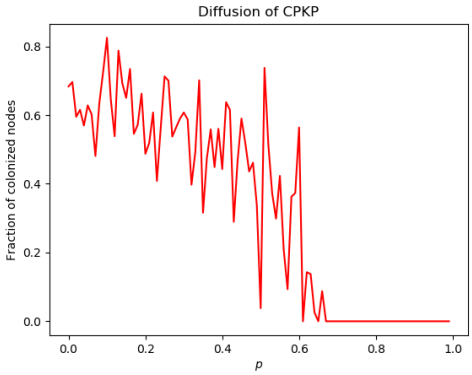
In figure 2.6 are shown the plots of the epidemic threshold for the three different networks models.



(a) Barabasi-Albert



(b) Moving-HCW



(c) Erdos-Renyi

Figure 2.6: Plots of the order parameter in the phase transition between non epidemic and epidemic phase.

As we can see the Barabasi-Albert and the Erdos-Renyi have the same threshold while for the moving HCWs the threshold is bigger, this corroborates the hypothesis of the centrality of the HCW's cluster for the network diffusion. Comparing these plots with the figure 2.3b we see the differences between the network and the mean field diffusion. The first difference is given by the fact that the mean field has a smooth distribution while the network is discontinued but these are statistical fluctuation due to the low number of nodes in the network which doesn't allow the mean field limit. The second and more important difference is the threshold which is bigger for the mean field respect to the network, this can be caused by the structure of the network that, as we have see comparing the different topologies, has a central role in the dynamic of the diffusion.
In conclusion in this work we have corroborate the centrality of the topology in order to avoid the contagion in fact as we have seen a fixed room HCWs network is more efficient than a moving HCWs network even if there are contacts between the HCWs, for small number we have seen that the structure of this contact is less relevant than what could be expected.

Appendix A

Python code for CPKP mean field

```
import matplotlib
from matplotlib import pyplot as plt
import math
import random
from numpy import *

def Deriv(x,y):
    #Parameters of the model
    mu_c=0.05      #Colonized discharge rate
    mu_u=0.1       #Uncolonized discharge rate
    alpha=0.21     #Probability of colonization
    phi=0.05       #Probability that the patient added is colonized
    lambd=1        #Probability to add a patient
    mu_HC=0.5      #Probability of a colonized HCW to become uncolonized
    p=0
    b=0.1
    c=0.5
    B=80
    dy=zeros(n,'double')

  dy[0]=(1-phi)*lambd*(B-y[0]-y[1])-mu_u*y[0]-alpha*b*y[0]*y[3]
    dy[1]=phi*lambd*(B-y[0]-y[1])-mu_c*y[1]+alpha*b*y[0]*y[3]
    dy[2]=mu_HC*y[3]-alpha*b*y[1]*y[2]-alpha*c*y[2]*y[3]
    dy[3]=-mu_HC*y[3]+alpha*b*y[1]*y[2]+alpha*c*y[2]*y[3]
    return dy

def Derivmod(x,y):
    #Parameters of the model
    mu_c=0.05      #Colonized discharge rate
    mu_u=0.1       #Uncolonized discharge rate
    alpha=0.21     #Probability of colonization
    phi=0       #Probability that the patient added is colonized
    lambd=1        #Probability to add a patient
    mu_HC=0.5      #Probability of a colonized HCW to become uncolonized
    p=0.7
    b=0.1
    c=0.5
    B=80
    dy=zeros(n,'double')

```

```
  dy[0]=(1-phi)*lambd*(B-y[0]-y[1])-mu_u*y[0]-alpha*b*y[0]*y[3]
    dy[1]=phi*lambd*(B-y[0]-y[1])-mu_c*y[1]+alpha*b*y[0]*y[3]
    dy[2]=mu_HC*y[3]-(1-p)*alpha*b*y[1]*y[2]-(1-p)*alpha*c*y[2]*y[3]
    dy[3]=-mu_HC*y[3]+(1-p)*alpha*b*y[1]*y[2]+(1-p)*alpha*c*y[2]*y[3]
    return dy

def RK4(x,y,n,h,xf):
    k1=zeros(n, 'double')
    k2=zeros(n, 'double')
    k3=zeros(n, 'double')
    k4=zeros(n, 'double')
    ym=zeros(n, 'double')
    ye=zeros(n, 'double')
    slope=zeros(n, 'double')

    if x<xf/2:
        k1=Deriv(x,y)
    else:
        k1=Derivmod(x,y)

    for i in range(n):
        ym[i]=y[i]+k1[i]*h/2

    if x<xf/2:
        k2=Deriv(x+h/2,ym)
    else:
        k2=Derivmod(x+h/2,ym)

    for i in range(n):
        ym[i]=y[i]+k2[i]*h/2

    if x<xf/2:
        k3=Deriv(x+h/2,ym)
    else:
        k3=Derivmod(x+h/2,ym)

    for i in range(n):
        ye[i]=y[i]+k3[i]*h
    if x<xf/2:
        k4=Deriv(x+h,ye)
    else:
        k4=Derivmod(x+h,ye)

    for i in range(n):
        slope[i]=(k1[i]+2*(k2[i]+k3[i])+k4[i])/6
        y[i]=y[i]+slope[i]*h
    x=x+h
    return x, y

def integrator(x,y,n,h,xend,xf):
    while x<xend:
        if xend-x<h:
            h=xend-x
        x,y=RK4(x,y,n,h,xf)
    return x ,y

```

```
n=4
y=zeros(n,'double')
yi=zeros(n,'double')
yi[0]=80
yi[1]=0
yi[2]=10
yi[3]=0
xi=0
dx=0.01
xf=1000
xout=1

j=int(dx*xf)
xp=zeros(1001,'double')
yp=zeros((n,1001), 'double')

x=xi
m=0
xp[m]=x
for i in range(n):
    yp[i,m]=yi[i]
    y[i]=yi[i]

while x<xf:
    xend=x+xout
    if xend>xf:
        xend=xf
    h=dx
    x,y=integrator(x,y,n,h,xend,xf)
    m=m+1
    xp[m]=x
    for i in range(n):
        yp[i,m]=y[i]

#Plot the total number of colonized nodes for time step
plt.plot(xp,yp[1],'r',label='Colonized Patient')
plt.plot(xp,yp[0],'g',label='Uncolonized Patient')
plt.plot(xp,yp[3],'y',label='Colonized HCW')
plt.plot(xp,yp[2],'blue',label='Uncolonized HCW')
plt.legend(loc='upper right')
plt.xlabel('Time')
plt.ylabel('Absolute number of nodes')
plt.title('Diffusion of CPKP')
plt.show()
```

Appendix B

Python code for CPKP

The python code for the network diffusion on an Eros-Renyi is based on an event driven algorithm as shown in the section 1.3

```
import matplotlib.pyplot as plt
import math
import random
import graphviz
from networkx import *
from numpy import *

#Preparing the graph
#Num_HCW HCW nodes, each with 8 patient nodes as maximum number
Num_HCW=10
G=fast_gnp_random_graph(Num_HCW, 0.5, seed=None, directed=False)
Role=['HCW']
set_node_attributes(G, Role, 'Role')
fixed_pos=spring_layout(G)     #Setting the layout to fix the position of the HCW nodes in the plot

for n in range(len(G.nodes())):            #Adding the patient nodes, 8 for
    for i in range(8):                     #each HCW
        new=9*(n+1)+i+1
        G.add_node(new,Role=['Patient'])
        G.add_edge(n,new)

State=['Uncolonized']
set_node_attributes(G, State,'State')   #Setting all the states to uncolonized
pos = spring_layout(G,pos=fixed_pos,fixed=fixed_pos.keys())   #Layout of the node with fixed the HCW nodes

infected_list=[]  #Inizialization of the infected list
#Parameters of the model
mu_c=0.05     #Colonized discharge rate
mu_u=0.1      #Uncolonized discharge rate
alpha=0.21    #Probability of colonization
phi=0.05      #Probability that the patient added is colonized
lambd=1       #Probability to add a patient
mu_HC=0.5     #Probability of a colonized HCW to become uncolonized
beta=0.21
```

```
p=0.7
#Colonized admition rate phi*lambd
#Uncolonized admition rate (1-phi)*lambd
Time=1000 #Time steps of the simulation (days)
Frac_Col=zeros(Time,'double')
Frac_Uncol=zeros(Time,'double')
Uncol_HCW=zeros(Time,'double')
Col_HCW=zeros(Time,'double')
Uncol_pat=zeros(Time,'double')
Col_pat=zeros(Time,'double')
tim=zeros(Time,'int')
#Begin of the time simulation
for t in range(Time):

    #Adding the methods to remove the diffusion after the beginning of the contagion
    if t>Time/2:
        phi=0        #Isolation of admitted colonized patient
        beta=0.21*(1-p)    #Head washing to decrease the probability to became colonized after the contact


    #Decontaminatio of the HCW
    for i in range(Num_HCW):
        if G.nodes[i]['State']==['Colonized']:
            rand_head=random.uniform(0,1)
            if rand_head<=mu_HC:
                G.nodes[i]['State']=['Uncolonized']

  #Removing the nodes
    to_remove_list=[]
    for node in G.nodes():
        if G.nodes[node]['Role']==['Patient']:
            if G.nodes[node]['State']==['Colonized']:
                rand_colon=random.uniform(0,1)
                if rand_colon<=mu_c:
                    to_remove_list.append(node)
            if G.nodes[node]['State']==['Uncolonized']:
                rand_uncolon=random.uniform(0,1)
                if rand_uncolon<=mu_u:
                    to_remove_list.append(node)
    for j in range(len(to_remove_list)):
        G.remove_node(to_remove_list[j])

    big_node=amax(G.nodes())+1  #Setting the bigger node to begin the adding process

    #Adding the nodes kepping the maximum number of patient for each node equal to 8
    to_add_list_col=[]
    to_add_list_uncol=[]
    to_add_list_link=[]
    newnum=big_node
    for i in range(Num_HCW):                          #Cicling only on the HCW because we can only add patient to the HCW
        neighbours=list(G.neighbors(i))
```

```
86          patient=[j for j in range(len(neighbours)) if G.nodes[
    neighbours[j]]['Role']==['Patient']] #Counting the number of patient
     for each node
87          patient_num=len(patient)
88
89          if patient_num<7:
90              rand_admis=random.uniform(0,1)
91              if rand_admis<=lambd:
92                  rand_pat=random.uniform(0,1)
93                  if rand_pat<=phi:
94                      to_add_list_col.append(newnum)   #add colonized
    node
95                      to_add_list_link.append((i,newnum))
96                      newnum=newnum+1
97                  else:
98                      to_add_list_uncol.append(newnum)  #add uncolonized
    node
99                      to_add_list_link.append((i,newnum))
100                     newnum=newnum+1
101
102     #Adding of the nodes to the graph
103     for i in range(len(to_add_list_col)):
104         G.add_node(to_add_list_col[i], Role=['Patient'], State=['
    Colonized'])
105
106     for i in range(len(to_add_list_uncol)):
107         G.add_node(to_add_list_uncol[i], Role=['Patient'], State=['
    Uncolonized'])
108
109     for i in range(len(to_add_list_link)):
110         G.add_edge(*to_add_list_link[i])
111
112     #Stop condition if the graph is empty
113     if len(G.nodes())==0:
114         break
115
116     #Infection for each node
117     to_infect_list=[]
118     for node in G.nodes():
119         if G.nodes[node]['State']==['Colonized']:
120             neighbours=list(G.neighbors(node))
121             for j in range(len(neighbours)):
122                 if G.nodes[neighbours[j]]['State']==['Uncolonized']:
123                     if G.nodes[neighbours[j]]['Role']==['HCW']:
124                         rand_inf=random.uniform(0,1)
125                         if rand_inf<=beta:
126                             to_infect_list.append(neighbours[j])
127                     else:
128                         rand_inf=random.uniform(0,1)
129                         if rand_inf<=alpha:
130                             to_infect_list.append(neighbours[j])
131
132
133
134     to_infect_list=list(set(to_infect_list))  #Keeping only the unique
    value for each colonized node
```

```
135
136     for i in range(len(to_infect_list)):
137         G.nodes[to_infect_list[i]]['State']=['Colonized']
138         infected_list.append(to_infect_list[i])
139
140     #Counting the nodes
141     for node in G.nodes():
142         if G.nodes[node]['State']==['Colonized']:
143             Frac_Col[t]=Frac_Col[t]+1
144             if G.nodes[node]['Role']==['Patient']:
145                 Col_pat[t]=Col_pat[t]+1
146             else:
147                 Col_HCW[t]=Col_HCW[t]+1
148         else:
149             if G.nodes[node]['Role']==['Patient']:
150                 Uncol_pat[t]=Uncol_pat[t]+1
151             else:
152                 Uncol_HCW[t]=Uncol_HCW[t]+1
153
154     Frac_Col[t]=Frac_Col[t]/len(G.nodes())
155     Frac_Uncol[t]=1-Frac_Col[t]
156     tim[t]=t
157
158 #Plot the ratio of colonized nodes for time step
159 plt.plot(tim,Frac_Col,'r',label='Colonized')
160 plt.plot(tim,Frac_Uncol,'g',label='Uncolonized')
161 plt.legend(loc='upper right')
162 plt.xlabel('Time')
163 plt.ylabel('Relative number of nodes')
164 plt.title('Diffusion of CPKP')
165 plt.show()
166 #Plot the total number of colonized nodes for time step
167 plt.plot(tim,Col_pat,'r',label='Colonized Patient')
168 plt.plot(tim,Uncol_pat,'g',label='Uncolonized Patient')
169 plt.plot(tim,Col_HCW,'y',label='Colonized HCW')
170 plt.plot(tim,Uncol_HCW,'blue',label='Uncolonized HCW')
171 plt.legend(loc='upper right')
172 plt.xlabel('Time')
173 plt.ylabel('Absolute number of nodes')
174 plt.title('Diffusion of CPKP')
175 plt.show()
```

Bibliography

[1] Vana Sypsa, Mina Psichogiou, Georgia-Aikaterina Bouzala, Linos Hadjihannas, Angelos Hatzakis, Georgios L. Daikos, *Transmission Dynamics of Carbapenemase-Producing Klebsiella Pneumoniae and Anticipated Impact of Infection Control Strategies in a Surgical Unit*, Athens Greece, 2012.

[2] Kiss, István Z., Miller, Joel, Simon, Péter L., *Mathematics of Epidemics on Networks*, 2017.

[3] Petter Holme, *Model versions and fast algorithms for network epidemiology*, Suwon Korea.

[4] Daniel Remondini, *Lectures notes of the course "Complex Network"*, 2019.