

Network Science
 Spring 2024
 Project 2 solutions

In lectures we were studying epidemics on networks, in particular we produced a naive network SI model to determine the expected spread of disease on a network. In this project, the SIR model will be considered, where another state is added to the SI model, namely a recovery state, whereby someone recovers from the disease.

In the SIR model a person can be infected, susceptible and recovered, where latter is defined as a person who had the disease and recovered, and are therefore no longer infected or susceptible. We will consider the model in terms of networks and again consider what happens on average for a network of N nodes. Define A_{ij} to be the adjacency matrix for the network under consideration and s_i, x_i, r_i to be the probabilities that node i is susceptible, infected or recovered, respectively. It can be shown that

$$\frac{ds_i}{dt} = -\beta s_i \sum_{j=0}^{N-1} A_{ij} x_j \quad (1)$$

$$\frac{dx_i}{dt} = \beta s_i \sum_{j=0}^{N-1} A_{ij} x_j - \gamma x_i \quad (2)$$

$$\frac{dr_i}{dt} = \gamma x_i \quad (3)$$

where

$$s_i(t) + r_i(t) + x_i(t) = 1. \quad (4)$$

Here, γ and β are model parameters where γ is the recovery rate and β is the infection rate that an infected individual will be infected. It is assumed that deaths do not occur from natural causes, and deaths from the disease are encapsulated in γ . Note that we number the graph nodes from 0 to $N - 1$, which is why the sums in the equations are over the same range.

0.1 Part 1 solutions

1. (6 marks) Following a similar method to Lectures for the SI model, consider a small perturbation to the equilibrium point of the system of Equations (1)-(3), where everyone is susceptible. Linearise the resulting equations and show that (2) can be written in matrix form as

$$\frac{d\mathbf{x}}{dt} = \beta \mathbf{M} \mathbf{x} \quad (5)$$

and define \mathbf{M} .

Solution: An equilibrium point of equations (1)-(2) is given by

$$x_i(t) = 0, s_i(t) = 1, r_i(t) = 0. \quad (6)$$

Note also that we could have

$$x_i(t) = 1, s_i(t) = 0, r_i(t) = 0, \quad (7)$$

and

$$x_i(t) = 0, s_i(t) = 1, r_i(t) = 1. \quad (8)$$

Considering a small perturbation to the solution given by (6) gives

$$x_i(t) = \epsilon \tilde{x}_i(t) + O(\epsilon^2), s_i(t) = 1 + \epsilon \tilde{s}_i(t) + O(\epsilon^2), r_i(t) = \epsilon \tilde{r}_i(t) + O(\epsilon^2). \quad (9)$$

where $\epsilon \ll 1$. Substituting (9) into (1)-(3) gives

$$\epsilon \frac{d\tilde{s}_i}{dt} = -\beta(1 + \epsilon \tilde{s}_i) \sum_j A_{ij} \tilde{x}_j + O(\epsilon^2), \quad (10)$$

$$\epsilon \frac{d\tilde{x}_i}{dt} = \epsilon \beta(1 + \epsilon \tilde{s}_i) \sum_j A_{ij} \tilde{x}_j - \epsilon \gamma \tilde{x}_i + O(\epsilon^2), \quad (11)$$

$$\epsilon \frac{d\tilde{r}_i}{dt} = \epsilon \gamma \tilde{x}_i + O(\epsilon^2). \quad (12)$$

Diving through by ϵ we get

$$\frac{d\tilde{s}_i}{dt} = -\beta \sum_j A_{ij} \tilde{x}_j + O(\epsilon), \quad (13)$$

$$\frac{d\tilde{x}_i}{dt} = \beta \sum_j A_{ij} \tilde{x}_j - \gamma \tilde{x}_i + O(\epsilon), \quad (14)$$

$$\frac{d\tilde{r}_i}{dt} = \gamma \tilde{x}_i + O(\epsilon), \quad (15)$$

and taking the limit as $\epsilon \rightarrow 0$ gives

$$\frac{d\tilde{s}_i}{dt} = -\beta \sum_j A_{ij} \tilde{x}_j, \quad (16)$$

$$\frac{d\tilde{x}_i}{dt} = \beta \sum_j A_{ij} \tilde{x}_j - \gamma \tilde{x}_i, \quad (17)$$

$$\frac{d\tilde{r}_i}{dt} = \gamma \tilde{x}_i. \quad (18)$$

This is the linearised system of equation, where Equation (17) is in terms of x_i only, and can be solved to then give s_i, r_i . Using the Kronecker delta we can write (17) in vector form, namely

$$\frac{dx_i}{dt} = \sum_j \left(\beta A_{ij} - \gamma \delta_{ij} \right) x_j, \quad (19)$$

where we have dropped the tilde on the of the variables. Thus, in vector form, we have

$$\frac{d\mathbf{x}}{dt} = \beta \mathbf{M} \mathbf{x}, \quad (20)$$

where

$$\mathbf{M} = \mathbf{A} - \frac{\gamma}{\beta} \mathbf{I}, \quad (21)$$

\mathbf{A} is the adjacency matrix of the network, and \mathbf{I} the identity matrix.

2. (3 marks) By considering an appropriate form for the perturbation, $\sim e^{\lambda t}$, determine how the perturbation changes with time. In particular, comment on how the spread of disease will change with γ , β and the highest degree in the Network.

You may find it useful to use the result from Linear Algebra that given a diagonalisable, $n \times n$ matrix \mathbf{B} with eigenvectors \mathbf{v}_i and eigenvalues λ_i , the matrix $\bar{\mathbf{B}} = \mathbf{B} + b\mathbf{I}$ has the same eigenvectors \mathbf{v}_i with eigenvalues $\lambda_i + b$.

Solution: Substitute $x_i = \hat{x}_i e^{\lambda t}$ into (20) to give

$$\mathbf{M}\hat{\mathbf{x}} = \frac{\lambda}{\beta}\hat{\mathbf{x}} \quad (22)$$

and therefore we have an eigenvalue problem, where λ/β is the eigenvalue of \mathbf{M} . Since $\beta > 0$, if $\text{Re}(\lambda) > 0$ solutions will grow, and if $\text{Re}(\lambda) < 0$ they will decay. Now, we know \mathbf{A} is symmetric, and therefore using the result from lectures the eigenvalues of \mathbf{A} , λ_A , are such that $\bar{k} \leq \max(\lambda_A) \leq k_{\max}$. Now, by the Linear Algebra result given in the question, it follows that the eigenvalues of \mathbf{M} , $\lambda_M = \lambda/\beta$, are the eigenvalues of \mathbf{A} , shifted by $-\gamma/\beta$, and therefore $\max(\lambda_M) = \max(\lambda_A) - \gamma/\beta$. We know $\max(\lambda_A) > 0$ since $\bar{k} > 0$ and $k_{\max} > 0$, and therefore if $\gamma/\beta < \max(\lambda_A)$ the solution will grow, but if $\gamma/\beta > \max(\lambda_A)$ the solution will decay. Physically, this tells us that if the recovery rate is sufficiently small, the disease will spread on small time-scales. Moreover, we know λ_A is maximised for Networks with larger degrees, and therefore the disease is more likely to spread on networks with larger maximum degrees, given a specified γ and β . This makes sense physically, since we expect there is a higher probability of a disease spreading in a population, if the individuals have more interactions/connections/links with other individuals.

0.1 An example solution for Part 2:

```
[ ]: # Do not modify this cell or import any other modules
# without explicit permission.
# You should run this cell before running the code below.
import numpy as np
import networkx as nx
%matplotlib inline
import matplotlib.pyplot as plt
import scipy.sparse as sp
from scipy.integrate import solve_ivp
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = [8, 6]
```

```
[ ]: #Function which given gamma, beta and the Adjacency matrix, returns the Matrix M, and the ratio gamma/beta
def M_matrix(gamma,beta, A):
    #Input: gamma the recovery rate
    #        beta the infection rate
    #        A the adjacency matrix of the Network

    #Output: M the matrix from part 1 corresponding to the linearisation
    #        s the ratio of gamma/beta

    s = gamma/beta
    M = (A.toarray()-s * np.identity(N))
    return M, s

# Generate a graph using the Barabasi- Albert model for N=100, m=2
N = 100
G_BA = nx.barabasi_albert_graph(N, 2)
A = nx.adjacency_matrix(G_BA)
```

```
[ ]: #We return the largest eigenvalue in algebraic value
M, s = M_matrix(0.2, 0.5, A)
```

```

A = A.astype(float)
l1, v1 = sp.linalg.eigsh(A,k=1,which='LA',return_eigenvectors=True)
print("Most positive eigenvalue of A = %f" %(l1))
l2, v2 = sp.linalg.eigsh(M,k=1,which='LA',return_eigenvectors=True)
print("Most positive eigenvalue of M = %f" %(l2))

#Check the maximum eigenvalue of A is bounded by bar_k < l1 < k_max
degree_sequence = sorted((d for n, d in G_BA.degree()), reverse=True)
kmax = max(degree_sequence)
k_bar = (sum(degree_sequence)/N)

if l1 < k_bar:
    print("Error: most positive eigenvalue of A is less than the average degree")
elif l1 > kmax:
    print("Error: most positive eigenvalue of A is greater than the maximum degree")
else:
    print("Most positive eigenvalue of A is in the correct bound")

#Now the eigenvalue of M l2, corresponds to the eigenvalue of A shifted such that l2
#-(l1 - gamma/beta)=0
#We want machine accuracy so
if l2 - (l1-0.2/0.5)< 0.001:
    print("Solution agrees with theory of eigenvalue shifted")
else:
    print("Error: solution does not agree with the shift of eigenvalue")

#The theory said that if gamma/beta was greater than max(lambda_A) there would
#be stability.
if s > l1:
    if l2 > 0.0:
        print("Error2")
    else:
        print("Solutions will decay and it agrees with theory")
else:
    if l2 < 0.0:
        print("Error3")
    else:
        print("Solutions will grow and it agrees with theory")

```

Most positive eigenvalue of A = 6.557389
 Most positive eigenvalue of M = 6.157389
 Most positive eigenvalue of A is in the correct bound
 Solution agrees with theory of eigenvalue shifted
 Solutions will grow and it agrees with theory

For this Network, and $\beta = 0.5$, what is the critical value of γ , called γ_c , for which $\gamma < \gamma_c$ we have

a spread of disease at short times, and for $\gamma > \gamma_c$ there isn't a spread? γ_c must be equal to β multiplied by the most positive eigenvalue of A .

```
[ ]: print("The critical value of gamma_c= %f" %(0.5*11))
```

The critical value of gamma_c= 3.278694

0.2 What happens as we increase m?

We now consider m increased from two to four.

```
[ ]: G_BA_2 = nx.barabasi_albert_graph(N, 4)
A2 = nx.adjacency_matrix(G_BA_2)
M2, s2 = M_matrix(0.2, 0.5, A2)
A2 = A2.astype(float)
l12, v12 = sp.linalg.eigsh(A2,k=1,which='LA',return_eigenvectors=True)
print("The critical value of gamma_c= %f" %(0.5*l12))
```

The critical value of gamma_c= 5.675808

As m increases, more links are added, and therefore the spread of infection is more likely. The critical value of γ is therefore increased. A larger recovery rate is needed to decrease the rate of spread of infection.

Note: Eigenvalues will be different for individual solutions due to the random graph generator. Eigenvalues are dependent on the graph generated by the random model, but theory and results from Part 1 should hold.

0.3 Numerical solution

```
[ ]: def SIR_model(G,i0,x0, beta, gamma,tf,Nt=10000):
    """
    Simulate SIR-network model
    Input:
    G: N-node Undirected Networkx graph with nodes numbered from 1 to N
    i0: Node which is initially infected with x_s=x0
    x0: Magnitude of initial condition
    beta, gamma: model parameters
    tf, Nt: Solutions are computed at Nt time steps from t=0 to t=tf

    Output:
    tarray: size Nt+1 array
    sarray: N x Nt+1 array containing s across the N network nodes at
            each time step
    xarray: N x Nt+1 array containing x across the N network nodes at
            each time step.
    """
    N = G.number_of_nodes()
```

```

tarray = np.linspace(0,tf,Nt+1)
sarray = np.zeros((N,Nt+1))
xarray = np.zeros((N,Nt+1))
tol = 1e-8
A = nx.adjacency_matrix(G)
A = A.astype(float)

def RHS(t,y):
    """Compute RHS of model1 at time t
    input: y should be a size 2N array with y[:N] and y[N:2*N] corresponding to s on nodes 1 to N and x on nodes 1 to N, respectively.
    output: dy, also a size 2N array corresponding to dy/dt """
    #Add code for 2.(a) here
    dy = np.zeros(2*N)
    dy[:N] = -beta*y[:N]*A.dot(y[N:])
    dy[N:] = - gamma*y[N:] + beta*y[:N]*A.dot(y[N:])
    return dy

yinit = np.zeros(2*N)
yinit[:N] = 1.0
yinit[N+i0] = x0
yinit[i0] = 1.0-x0
out = solve_ivp(RHS,[0,tf],yinit,method='RK45',t_eval=tarray,rtol=tol,atol=tol)

yarray = out.y
sarray, xarray = yarray[:N,:], yarray[N:,:]
return tarray,sarray,xarray

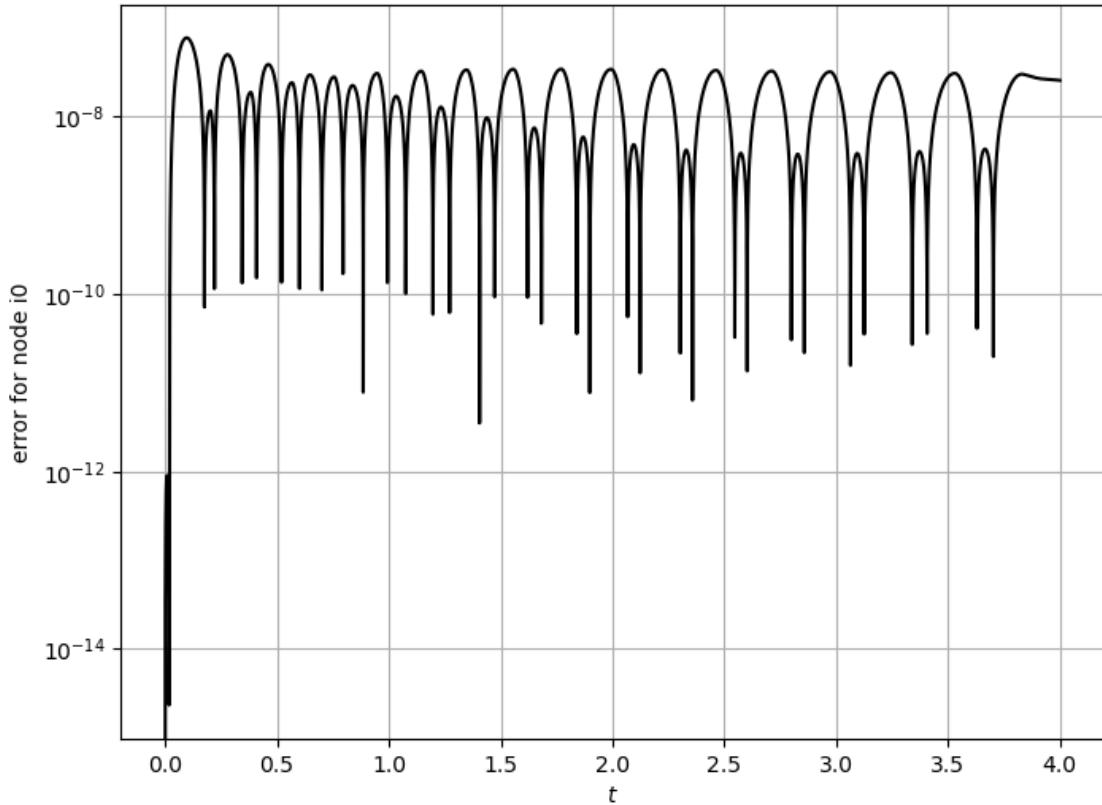
```

```

[ ]: #Here we include code to check the error
ta,sa,xa = SIR_model(G_BA,i0=0,x0=1,beta=0,gamma=1,tf=4,Nt=10000)
sa = sa.T
xa = xa.T
x1 = np.exp(-ta) #exact solution
error1 = np.abs(xa[:,0]-x1)
error2 = np.mean(xa[:,1:],axis=1)
plt.figure()
plt.semilogy(ta,error1,'k-',label=r'error, $|x_{-1}-x_{\{1,exact\}}|$ node 1')
plt.grid()
plt.xlabel(r'$t$')
plt.ylabel(r'error for node i0')
print("maximum error for node i0 over all times:", error1.max())
print("maximum error for all other nodes at all times:", error2.max())

```

```
maximum error for node i0 over all times: 7.651362587957067e-08
maximum error for all other nodes at all times: 0.0
```



0.4 $\gamma < \gamma_c$:

```
[ ]: from time import time
#Run simulation
t1 = time()
#Desired parameters
i0 = 1
tf = 10.0
x0 = 0.01
beta = 0.5
gamma = 0.1
ta,sa,xa = SIR_model(G_BA,i0,x0,beta,gamma,tf=tf,Nt=10000)
sa = sa.T
xa = xa.T
t2 = time()
#Check elapsed time taken
print("elapsed time:", t2-t1)
```

```

#Find the expected growth rate for the linearised system
M, s = M_matrix(gamma, beta, A)
l2, v2 = sp.linalg.eigsh(M,k=1,which='LA',return_eigenvectors=True)
#Note that eigenvalue is given at l2=lambda/beta. lambda is the growth rate.

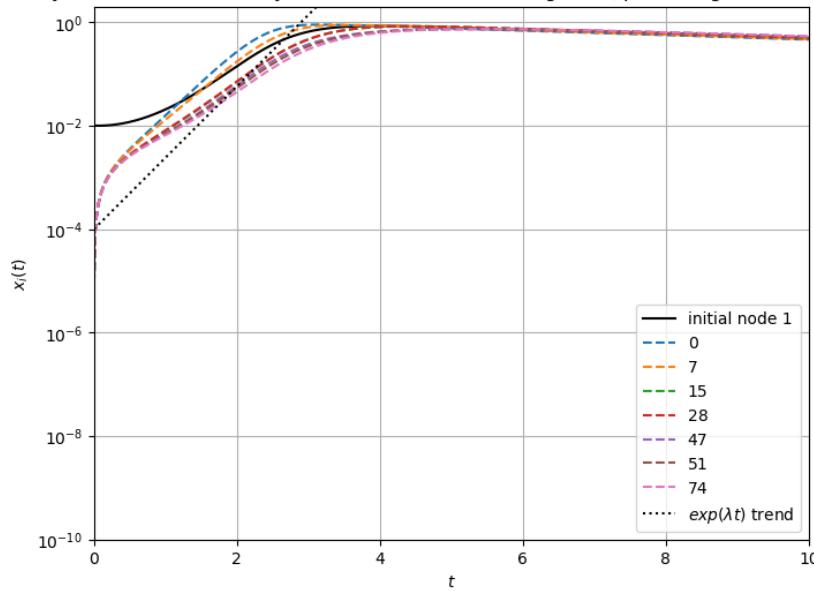
#Plot a figure showing how the disease distribution for the neighbours of the
#starting node change with time
plt.figure()
i0_neighbours = list(G_BA.adj[i0])
plt.semilogy(ta, xa[:, i0], 'k-',label='initial node %d' %i0)
plt.semilogy(ta,xa[:,i0_neighbours], '--')
plt.semilogy(ta,x0*0.01*np.exp(beta*ta*l2), 'k: ')
plt.legend(['initial node %d' %i0]+i0_neighbours+[r'$\exp(\lambda t)$'#
            'trend'],loc='lower right')
plt.grid()
plt.xlabel(r'$t$')
plt.ylabel(r'$x_i(t)$')
plt.title('Fig 1. Probability of infection for initially infectious node and its
            neighbors, plotted against time when $\gamma=%f' %gamma)
plt.xlim([0,10])
plt.ylim([1e-10,2])

```

elapsed time: 0.0886993408203125

[]: (1e-10, 2)

Fig 1. Probability of infection for initially infectious node and its neighbors, plotted against time when $\gamma=0.100000$

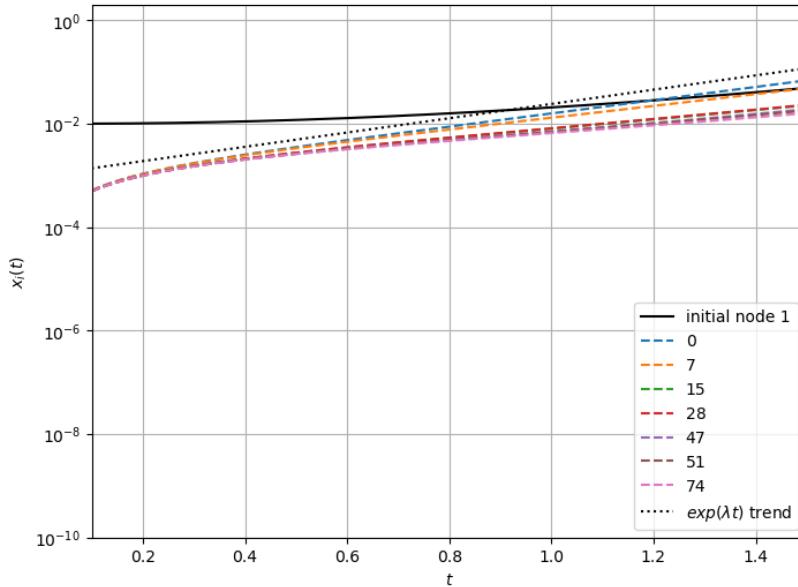


Below, we consider the time interval $0 < t < 1.5$ to see what happens for short times.

```
[ ]: plt.figure()
plt.semilogy(ta, xa[:, i0], 'k-', label='initial node %d' %i0)
plt.semilogy(ta, xa[:, i0_neighbors], '--')
plt.semilogy(ta, x0*0.1*np.exp(beta*ta*12), 'k: ')
plt.legend(['initial node %d' %i0]+i0_neighbors+[r'$\exp(\lambda t)$' %trend'], loc='lower right')
plt.grid()
plt.xlabel(r'$t$')
plt.ylabel(r'$x_i(t)$')
plt.title('Fig 2. Probability of infection for initially infectious node and its neighbors, plotted against time when $\gamma=0.100000$')
plt.xlim([0.1, 1.5])
plt.ylim([1e-10, 2])
```

```
[ ]: (1e-10, 2)
```

Fig 2. Probability of infection for initially infectious node and its neighbors, plotted against time when $\gamma=0.100000$



Below, we consider $2 < t < 4.5$ for a better idea of what happens near the maximum value of $x_i(t)$ for each neighbouring node of the initially infected node.

```
[ ]: plt.figure()
i0_neighbors = list(G_BA.adj[i0])
plt.semilogy(ta, xa[:, i0], 'k-', label='initial node %d' %i0)
plt.semilogy(ta, xa[:, i0_neighbors], '--')
plt.semilogy(ta, x0*0.1*np.exp(beta*ta*12), 'k: ')
```

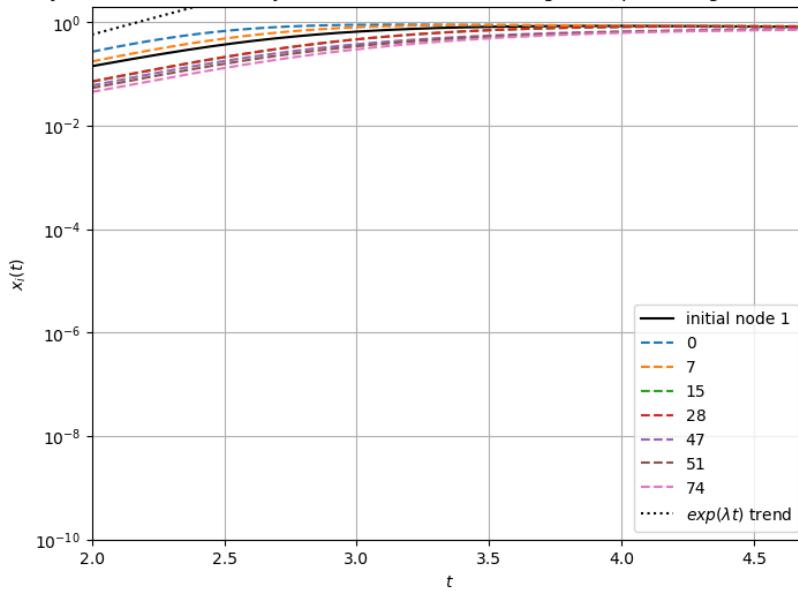
```

plt.legend(['initial node %d' %i0]+i0_neighbors+[r'$\exp(\lambda t)$'_
    ↪trend'], loc='lower right')
plt.grid()
plt.xlabel(r'$t$')
plt.ylabel(r'$x_i(t)$')
plt.title('Fig 3. Probability of infection for initially infectious node and'_
    ↪its neighbors, plotted against time when $\gamma=%f' %gamma)
plt.xlim([2.0,4.7])
plt.ylim([1e-10,2])

```

[]: (1e-10, 2)

Fig 3. Probability of infection for initially infectious node and its neighbors, plotted against time when $\gamma=0.100000$



[]: G_BA.degree(i0_neighbors)

[]: DegreeView({0: 18, 7: 15, 15: 6, 28: 6, 47: 2, 51: 2, 74: 2})

Above, I have found the degrees of the neighbours of node 1. Figure 1 and 2 show that after the initial diffusive transfer of infection to the neighbours, the growth rate found in the linear theory agrees well with the growth rate of the neighbouring nodes for short times. In Figure 2 we see that as t increases past $t \sim 0.7$, the growth rate of nodes with a lower degree, for example node 74 whose degree is 2, decreases and starts to diverge away from the linear theory. However, nodes who have a higher degree (for example node 0, whose degree is 18) carry on following the linear theory trend for longer times. Figures 1 and 3 show that there exists a critical value of time t_c , where for times greater than t_c the probability of infection decreases (for the γ and β considered here). Note, for $\beta = 0.5$ but $\gamma = 0$, as $t \rightarrow \infty$ the probability of infection will go to one for all nodes.

0.5 $\gamma > \gamma_c$

```
[ ]: #Desired parameters for gamma>gamma_c
i0 = 1
tf = 8
x0 = 0.01
beta = 0.5
gamma = 4.1
ta,sa,xa = SIR_model(G_BA,i0,x0,beta,gamma,tf=tf,Nt=10000)
sa = sa.T
xa = xa.T
t2 = time()
#Check elapsed time taken
print("elapsed time:", t2-t1)

#Find the expected growth rate for the linearised system
M, s = M_matrix(gamma, beta, A)
l2, v2 = sp.linalg.eigsh(M,k=1,which='LA',return_eigenvectors=True)
print(beta/l2)
#Note that eigenvalue is given at l2=lambda/beta. lambda is the growth rate.

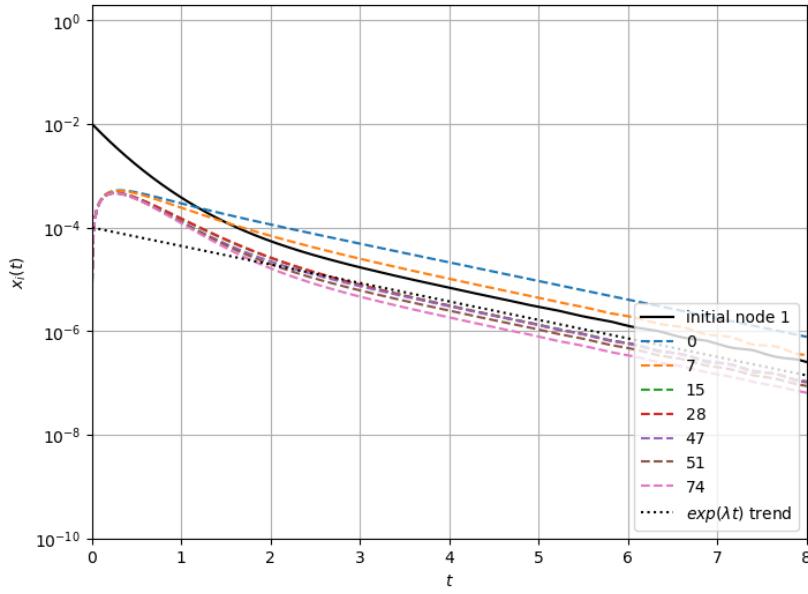
#Plot a figure showing how the disease distribution for the neighbours of the
#starting node change with time
plt.figure()
i0_neighbours = list(G_BA.adj[i0])
plt.semilogy(ta, xa[:, i0], 'k-', label='initial node %d' %i0)
plt.semilogy(ta,xa[:,i0_neighbours], '--')
plt.semilogy(ta,x0*0.01*np.exp(beta*ta*l2), 'k: ')
plt.legend(['initial node %d' %i0]+i0_neighbours+[r'$\exp(\lambda t)$' %trend'], loc='lower right')
plt.grid()
plt.xlabel(r'$t$')
plt.ylabel(r'$x_i(t)$')
plt.title('Fig 4. Probability of infection for initially infectious node and
#its neighbors, plotted against time when $\gamma=%f' %gamma)
plt.xlim([0,tf])
plt.ylim([1e-10,2])
```

elapsed time: 39.705477237701416

[-0.30439341]

[]: (1e-10, 2)

Fig 4. Probability of infection for initially infectious node and its neighbors, plotted against time when $\gamma=4.100000$

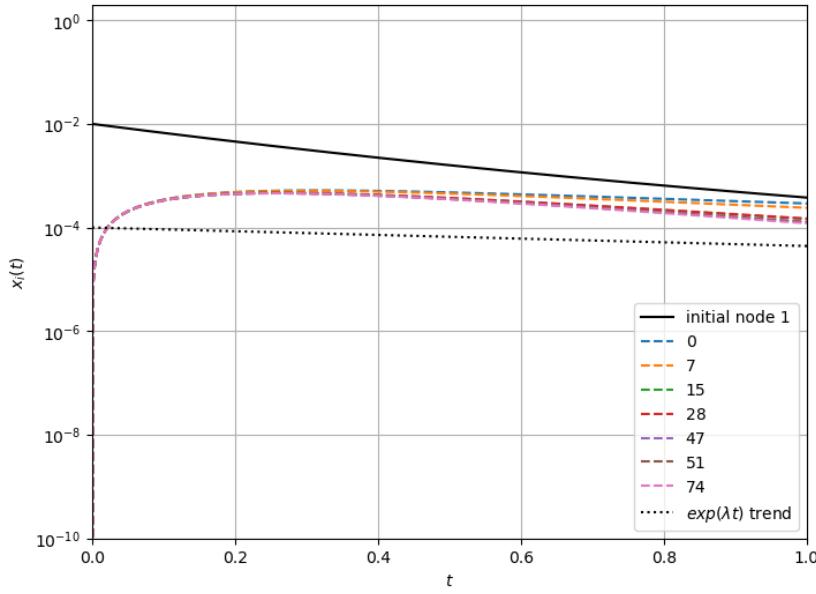


Here, I plot the solution for $0 < t < 1$.

```
[ ]: plt.figure()
i0_neighbors = list(G_BA.adj[i0])
plt.semilogy(ta, xa[:, i0], 'k-', label='initial node %d' %i0)
plt.semilogy(ta, xa[:, i0_neighbors], '--')
plt.semilogy(ta, x0*0.01*np.exp(beta*ta*12), 'k: ')
plt.legend(['initial node %d' %i0]+i0_neighbors+[r'$\exp(\lambda t)$' %trend'], loc='lower right')
plt.grid()
plt.xlabel(r'$t$')
plt.ylabel(r'$x_i(t)$')
plt.title('Fig 5. Probability of infection for initially infectious node and its neighbors, plotted against time when $\gamma=%f' %gamma)
plt.xlim([0,1])
plt.ylim([1e-10,2])
```

```
[ ]: (1e-10, 2)
```

Fig 5. Probability of infection for initially infectious node and its neighbors, plotted against time when $\gamma=4.100000$

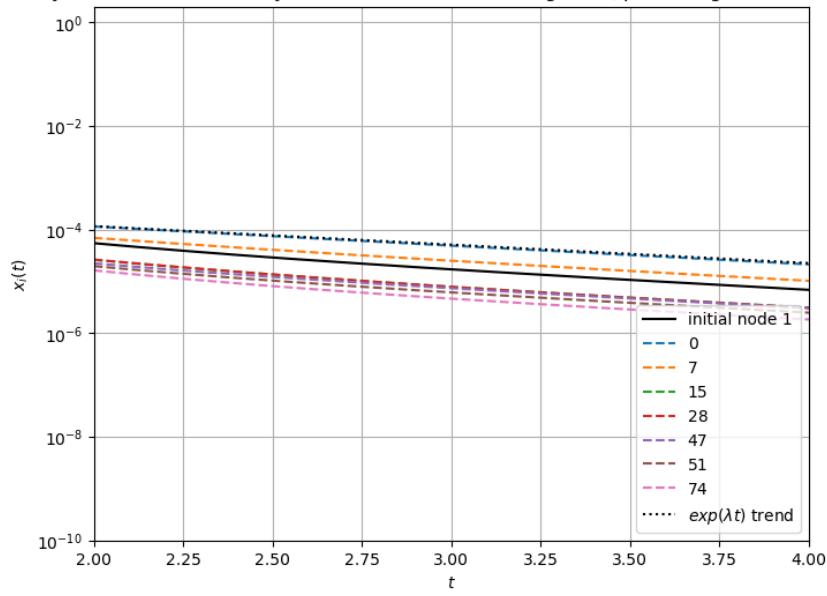


Here, I plot the solution for $2 < t < 4$.

```
[ ]: plt.figure()
i0_neighbors = list(G_BA.adj[i0])
plt.semilogy(ta, xa[:, i0], 'k-', label='initial node %d' %i0)
plt.semilogy(ta, xa[:, i0_neighbors], '---')
plt.semilogy(ta, x0*0.06*np.exp(beta*ta*12), 'k: ')
plt.legend(['initial node %d' %i0]+i0_neighbors+[r'$\exp(\lambda t)$' %trend'], loc='lower right')
plt.grid()
plt.xlabel(r'$t$')
plt.ylabel(r'$x_i(t)$')
plt.title('Fig 6. Probability of infection for initially infectious node and its neighbors, plotted against time when $\gamma=%f' %gamma')
plt.xlim([2,4.0])
plt.ylim([1e-10,2])
```

[]: (1e-10, 2)

Fig 6. Probability of infection for initially infectious node and its neighbors, plotted against time when $\gamma=4.100000$



Here we have considered $\gamma > \gamma_c$ and we see that although there is an initial rise in the probability of infection at very small times, due to an initial diffusive transfer of infection to the neighbours, we see that the probability of infection decreases rapidly to then agree with the linear theory trend.