

Modélisation d'une épidémie ou d'un incendie

Comment lutter contre une épidémie ou un incendie de la manière la plus efficace possible ?

I – Un modèle compartimental simple

- Construction du modèle
- Résolution informatique et résultats

II – Efficacité d'une perturbation

- Définition d'une perturbation et impact de celle-ci
- Définition de l'efficacité, variation d'un paramètre
- Variation de deux paramètres de la perturbation

III – Un modèle spatial

- Construction du modèle
- Résultats et intérêt du modèle

IV – Optimisation spatiale d'une perturbation

I – Un modèle compartimental simple

Hypothèses :

- Il y a 3 états possibles S (sain), I (infecté) et R (rétabli) qui correspondent pour l'incendie à l'herbe saine, en train de brûler et brûlée.
- Les individus infectés sont les individus infectieux (pas de période d'incubation, l'herbe peut brûler dès qu'elle brûle).
- L'infection et la guérison (ou la propagation et la consomption) ont des lois sans vieillissement.
- Les proportions des 3 états sont les mêmes dans tout l'espace.

Le modèle s'inspire du modèle compartimental de Kermack et McKendrick.

Soient N_s , N_i et N_r les nombres d'individus sains, infectés et rétablis et S , I et R leur proportions

$$(S = \frac{N_s}{P}, I = \frac{N_i}{P}, R = \frac{N_r}{P}) \text{ avec } P \text{ la population.}$$

Le principe de diffusion est similaire pour l'épidémie et l'incendie.

Les individus correspondent aux brins d'herbe.

On doit définir des flux d'individus $dF1$ et $dF2$ tel que :

$$\begin{cases} dNs = -dF1 \\ dNi = dF1 - dF2 \\ dNr = dF2 \end{cases}$$

Chaque individu infecté peut infecter chaque individu sain donc $dF1$ est proportionnel à $Ns * Ni$.

$dF1 = k * Ns * Ni * dt$ avec k une constante en temps $^{-1}$ car la loi est sans vieillissement.

De même, $dF2$ est proportionnel à Ni .

$dF2 = l * Ni$ avec l une constante en temps $^{-1}$.

On a donc un système :

$$\begin{cases} \frac{dNs}{dt} = -k * Ns * Ni \\ \frac{dNi}{dt} = k * Ns * Ni - l * Ni \\ \frac{dNr}{dt} = l * Ni \end{cases} \quad \text{puis} \quad \begin{cases} \frac{dS}{dt} = -kP * Ns * Ni \\ \frac{dI}{dt} = kP * Ns * Ni - l * Ni \\ \frac{dR}{dt} = l * Ni \end{cases}$$

On appelle k la constante d'infection et l la constante de guérison.

Le problème est qu'il n'est pas logique que le flux de S à I dépende de P à densité constante. J'ai compris que cela était dû au fait qu'à densité constante, augmenter la population signifie augmenter la surface, donc la surface de propagation par hypothèse. C'est cela qui modifie le résultat. J'ai donc ajouté une hypothèse et modifié les équations.

Hypothèse : la propagation se fait dans une surface s autour de l'individu.

On a un nouveau système :

$$\begin{cases} \frac{dNs}{dt} = -\frac{kds * Ns * Ni}{P} & \text{avec } P \text{ la population} \\ \frac{dNi}{dt} = \frac{kds * Ns * Ni}{P} - l * Ni & s \text{ la surface de propagation en m}^2 \\ \frac{dNr}{dt} = l * Ni & d \text{ la densité en hab/km}^2 \\ \end{cases}$$

k et l des constantes

Le système est donc :

$$\begin{cases} \frac{dS}{dt} = -kds * S * I \\ \frac{dI}{dt} = kds * S * I - l * I \\ \frac{dR}{dt} = l * I \end{cases}$$

Remarque : il est inutile d'introduire un état « décès » ou « détruit » car étant donné les hypothèses prises, cela reviendrait uniquement à diviser R en deux parts de proportions constantes. Cela n'a un intérêt que si les individus rétablis sont capables de redevenir sains, ce qui est négligé ici et évidemment impossible dans le cas de l'incendie.

Ce système est non linéaire, on ne peut pas le résoudre analytiquement. Il faut le résoudre numériquement.

J'ai utilisé la méthode de Runge-Kutta à l'ordre 4 (RK4) et j'ai pu vérifier qu'elle était plus efficace que les méthodes d'Euler et de prédiction-correction (la précision est meilleure pour le même temps de calcul)

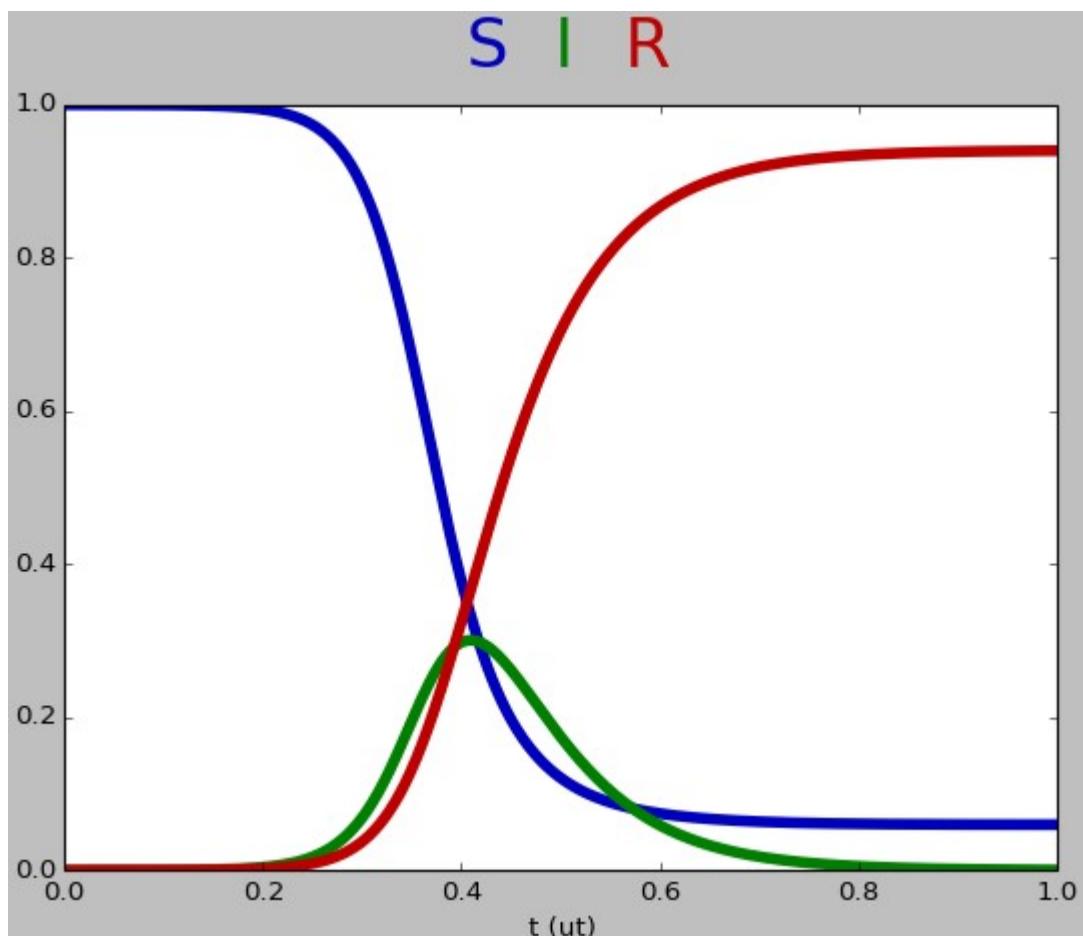
Fonction **simple(k,l,s,d,P,N,x)**

k , l , s , d et P ont déjà été définis. J'ai dû définir une unité de temps arbitraire ut tel que k et l sont en ut^{-1} .

N est le nombre de points utilisés pour la résolution et x est le temps (en ut) jusqu'auquel on veut observer l'évolution de l'épidémie.

Voici un exemple de résultat (les valeurs sont a priori exagérées pour bien voir le processus).

simple(300000,15,1.5,100,100000,1000,1) :



II – Efficacité d'une perturbation

Pour que mon travail soit utile, j'ai voulu évaluer l'impact d'une perturbation et optimiser son efficacité.

Une perturbation peut être un vaccin ou un choix politique (annonces pour se laver les mains par exemple) pour une épidémie, de l'eau ou autres produits pour l'incendie.

On définit une perturbation par :

- Son temps d'introduction t
- Sa force de ralentissement λ entre 0 et 1
- Sa force de guérison μ entre 0 et 1

tel que à t ,
$$\begin{cases} k = (1 - \lambda) * k \\ l = \frac{l}{1 - \mu} \end{cases}$$

λ caractérise le ralentissement de l'infection et μ , l'accélération de la guérison.

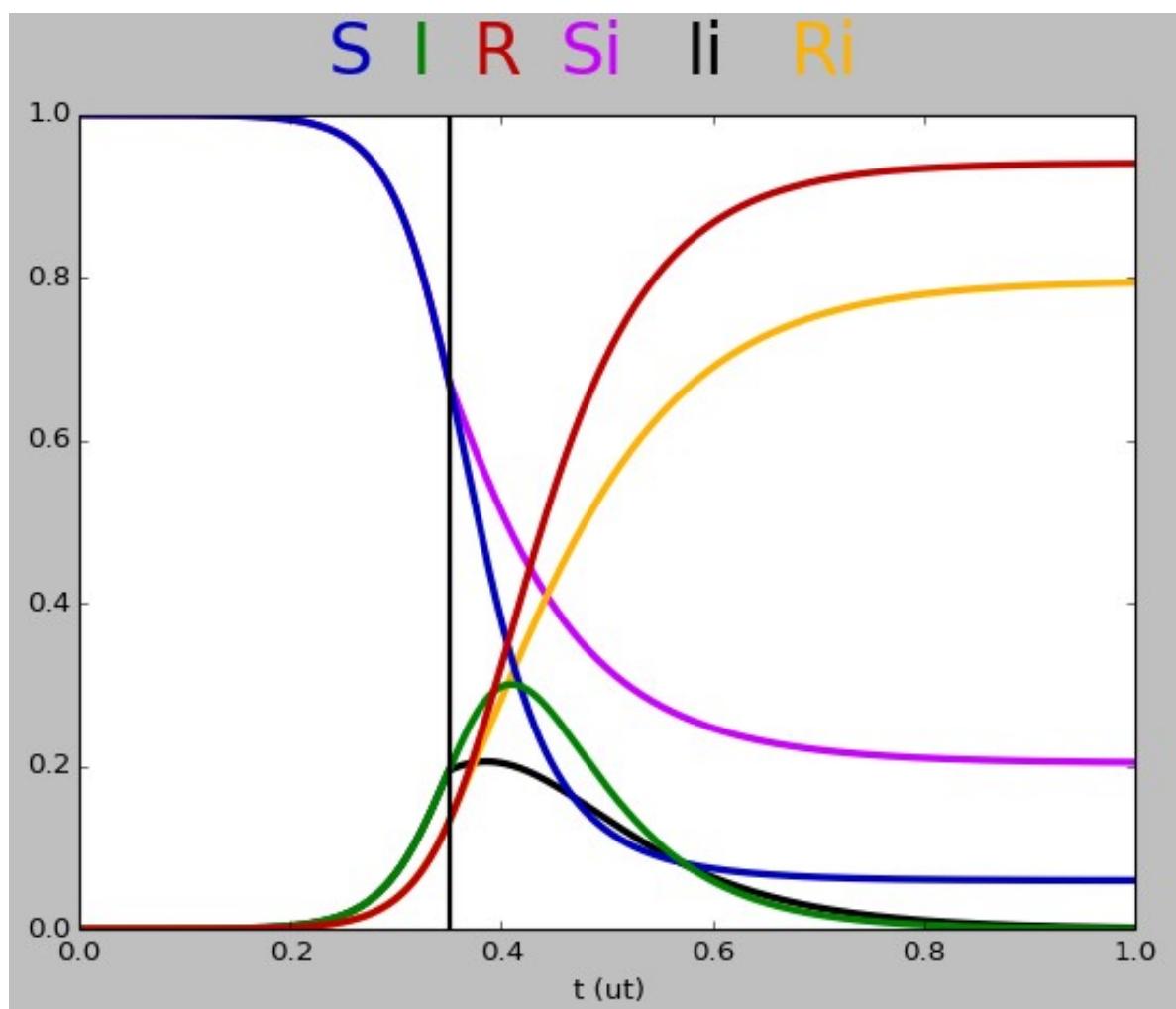
Remarque : au vu du système différentiel, le résultat final ne dépend que du rapport $\frac{k}{l}$, modifier k et l en conservant ce rapport ne modifie que la vitesse de la transformation. On peut donc prendre $\mu = 0$ et adapter λ sans modifier le résultat final.

J'ai fait un second programme qui montre l'évolution de l'épidémie avec et sans la perturbation. On peut ainsi voir l'impact de la perturbation.

Fonction **simple_impact(k,l,s,d,P,t,λ,μ,N,x)**

On a les mêmes arguments que précédemment, plus t , λ et μ qui caractérisent la perturbation.

`simple_impact(300000,15,1.5,100,100000,0.35,0.4,0,1000,1)` :



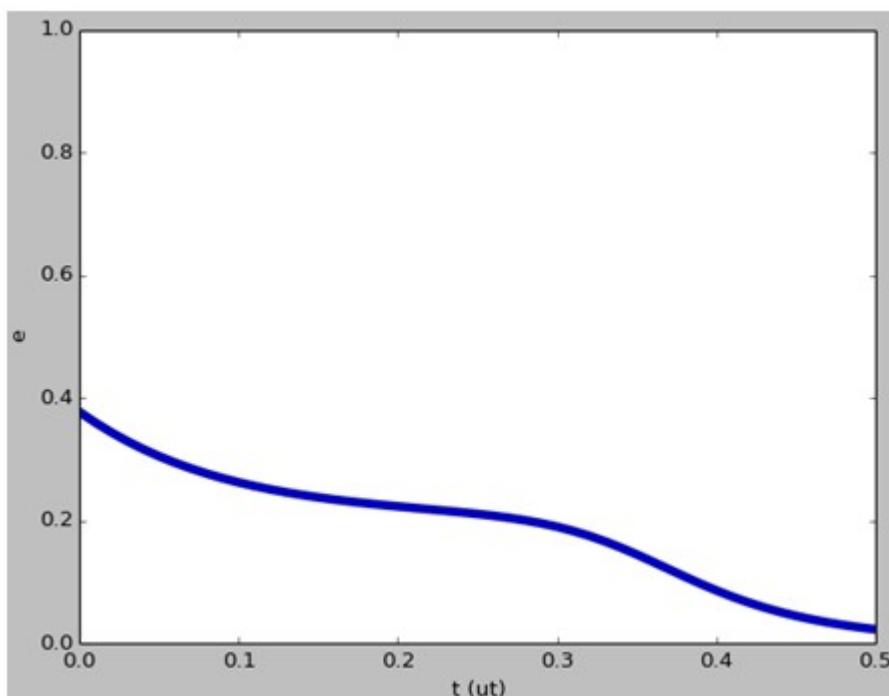
On définit l'efficacité : $\mathbf{e} = \lim_{t \rightarrow \infty} (Sp(t) - S(t))$

Fonctions `simple_e(k,l,s,d,P,t,λ,μ,N,x)`, `simple_et(k,l,s,d,P,λ,μ,N,x,N2,x2)`,
`simple_eλ(k,l,s,d,P,t,μ,N,x,N2)` et `simple_eμ(k,l,s,d,P,t,λ,N,x,N2)`

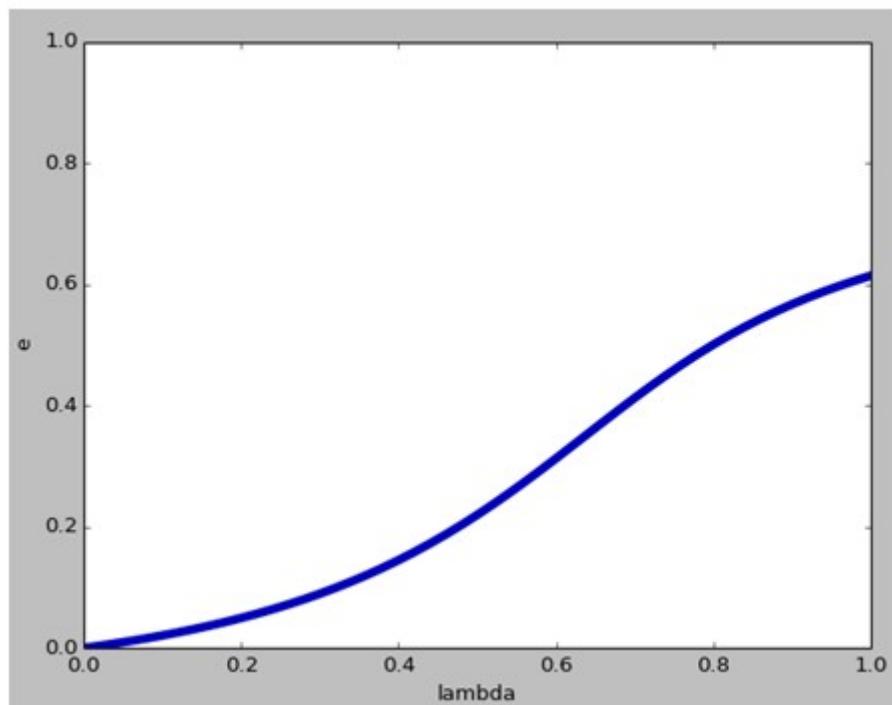
Le but est de faire varier un paramètre et d'observer en quoi il modifie l'efficacité. `simple_e` calcule l'efficacité. Les autres paramètres étant fixés, `simple_et` fait varier t , `simple_eλ` fait varier λ et `simple_eμ` fait varier μ .

$N2$ est le nombre de valeurs prises par le paramètre qui varie et $x2$ est la valeur limite qu'on veut observer pour le temps (pour λ et μ , c'est 1 par défaut).

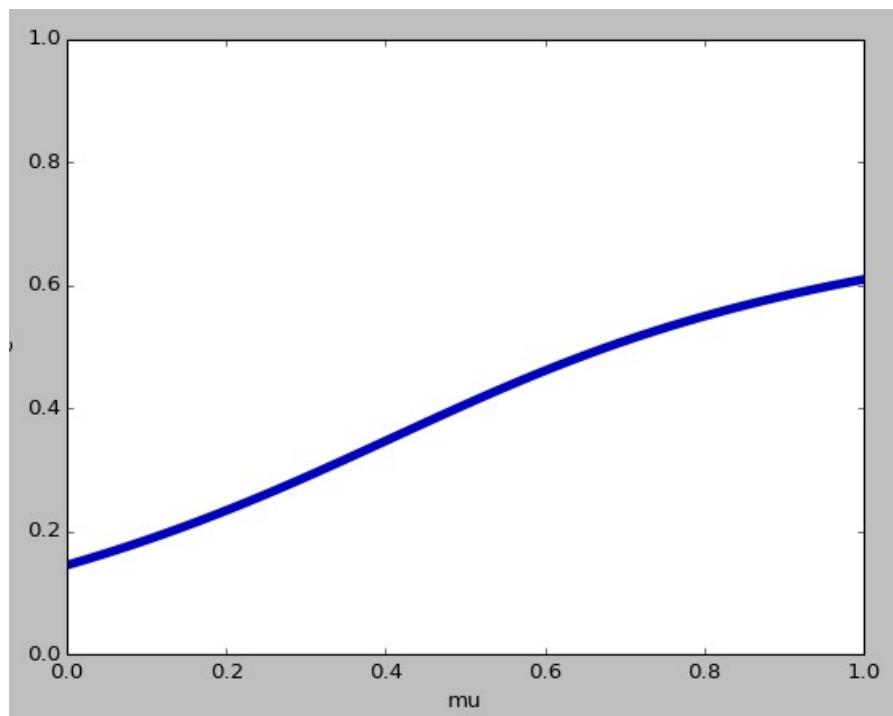
`simple_et(300000,15,1.5,100,100000,0.4,0,1000,1,50)` :



simple_eλ(300000,15,1.5,100,100000,0.35,0,1000,1,50):



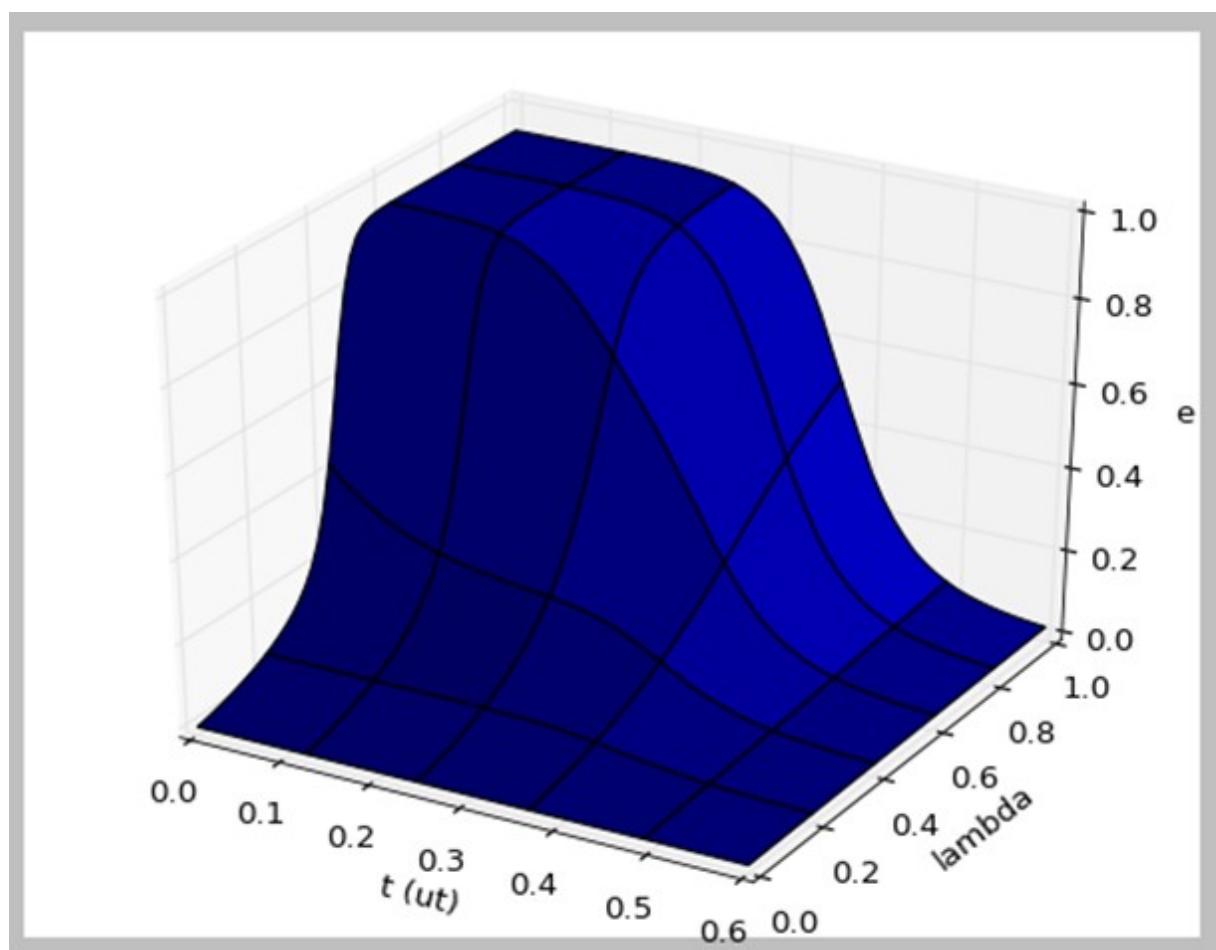
simple_eμ(300000,15,1.5,100,100000,0.35,0.4,1000,1,50):



Fonction **simple_etλ(k,l,s,d,P,μ,N,x,N2,x2)**

Je voulais ensuite combiner les programmes précédents pour faire varier t et λ , μ pouvant être pris nul par exemple comme je l'ai justifié. Ceux-ci prennent un même nombre de valeurs $N2$ et t varie entre 0 et $x2$.

simple_etλ(300000,15,1.5,100,100000,0,1000,1,50,0.6)



III – Un modèle spatial

Dans le premier modèle, j'ai supposé S, I et R uniformes dans l'espace. Cette hypothèse est beaucoup trop forte en général et acceptable uniquement localement, pour une surface très petite.

On retire à présent cette hypothèse.

Idée pour le modèle spatial :

- On divise l'espace en cases uniformes sur lesquelles S, I et R sont supposés constants.
- 2 étapes se répètent l'une après l'autre :
 - **Evolution de chaque case suivant le premier modèle**
 - **Echange d'individus entre les cases**

On utilise pour cet échange une matrice de transition Dm qui s'écrit en densité de mouvement.

Avec un grand nombre de cases, on peut ainsi calculer l'évolution de l'épidémie ou de l'incendie pour un milieu où non seulement S, I et R mais aussi k, l, d et Dm[i][j] varient dans l'espace (Dm est décrite à la page suivante).

Fonctions **espace(K,L,s,D,li,Dm,T,N,x,N3,zlim)**

espace_adj(K,L,s,D,li,Dm,T,N,x,N3,zlim)

et **espace_uni(k,l,s,d,P,dm,T,N,x,N3,zlim)**

N3 est la résolution du milieu : le milieu est divisé en N3 * N3 cases.

K, L et D sont des matrices de taille N3 * N3 tel que K[i][j] contient la valeur de k pour la case (i,j) et de même pour L avec l et pour D avec d. En effet k et l peuvent changer car elles dépendent de l'environnement.

li est une matrice de taille N3 * N3 tel que li[i][j] contient la valeur initiale de l dans la case (i,j).

Dm est en fait une matrice de matrices : Dm[i][j] contient les mouvements de population (en densité) depuis la case (i,j). Dm[i][j][i2][j2] est la densité de mouvement depuis la case (i,j) vers la case (i2,j2). Dm contient N3 * N3 matrices de taille N3 * N3.

T est la liste des temps pour lesquels on veut voir s'afficher le graphique 3D de la densité de population infectée.

zlim est le maximum voulu pour l'axe des z du graphique.

Au début, j'ai créé la fonction espace, qui fonctionne dans le cas quelconque. Toutefois, à l'étape d'échange d'individus, elle parcourt toute la matrice Dm, ce qui était beaucoup trop long.

Or dans le cas réel, on peut considérer que chaque case n'échange des individus qu'avec les cases adjacentes si le temps d'échange est suffisamment court (donc si N est suffisamment grand).

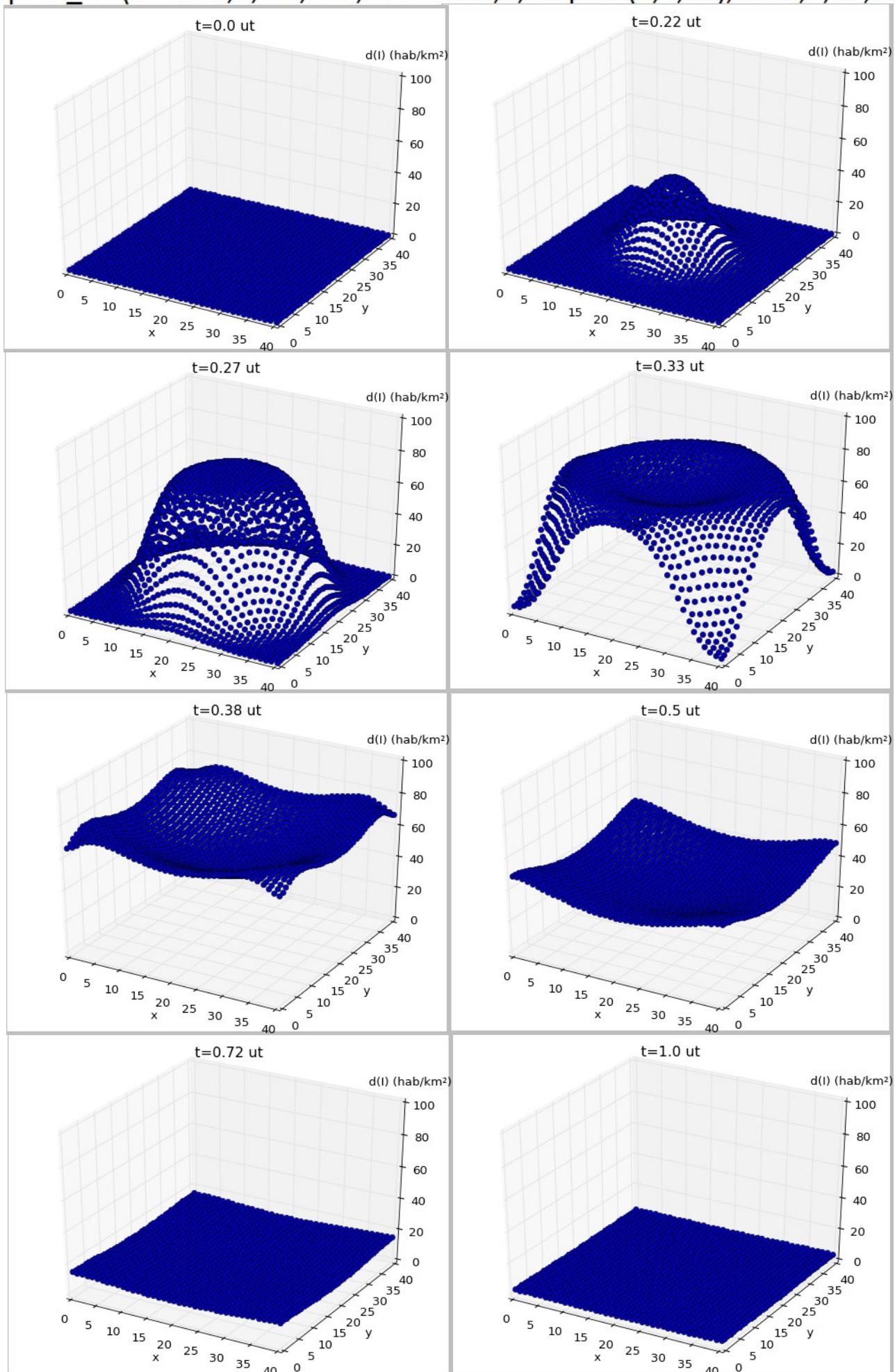
J'ai donc fait une fonction espace_adj qui ne vérifie que les cases utiles de Dm. Sa complexité est donc en $O(N * N^3^2)$ au lieu d'être en $O(N * N^3^4)$.

La fonction espace_uni sert pour un milieu où k, l, d et Dm[i][j] sont les mêmes partout, afin de ne pas avoir à configurer K, L, D et Dm, ce qui peut être assez long.

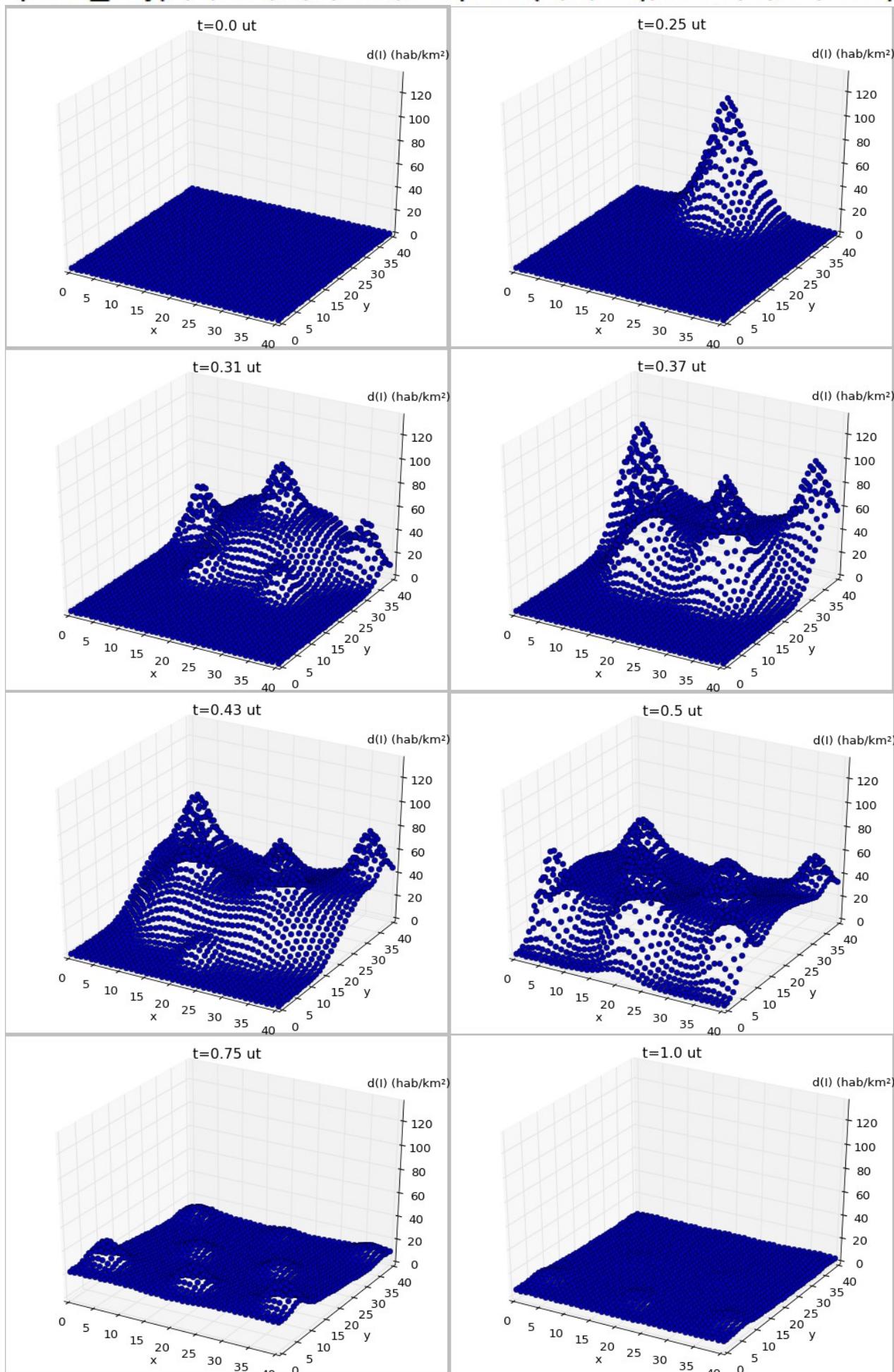
Elle prend en argument P au lieu de li, de telle sorte qu'on a initialement une personne infectée sur la case centrale du milieu et toutes les autres personnes sont saines.

Voici deux exemples pour un milieu uniforme, puis pour un milieu que j'ai créé avec des pics de densité localisés (ces pics sont des fonctions gaussiennes en deux dimensions).

espace_uni(500000,5,1.5,100,67000000,5,linspace(0,1,19),1000,1,41,100) :



espace_adj(K,L,1.5,D,li,Dm,linspace(0,1,19),1000,1,41,134) :



IV – Optimisation spatiale d'une perturbation

Il peut être trop cher d'effectuer une perturbation sur tout le milieu (tant en ce qui concerne les vaccins pour les épidémies que l'eau pour les incendies).

Le but est de voir où il est le plus efficace d'appliquer une perturbation si on a le droit de perturber un certain nombre fixé de cases.

Les fonctions `espace_adj_impact(K,L,s,D,Id,Dm,T,N,x,N3,t,λ,μ,Q,zlim)` et `espace_uni_impact(k,l,s,d,P,dm,T,N,x,N3,t,λ,μ,Q,zlim)`

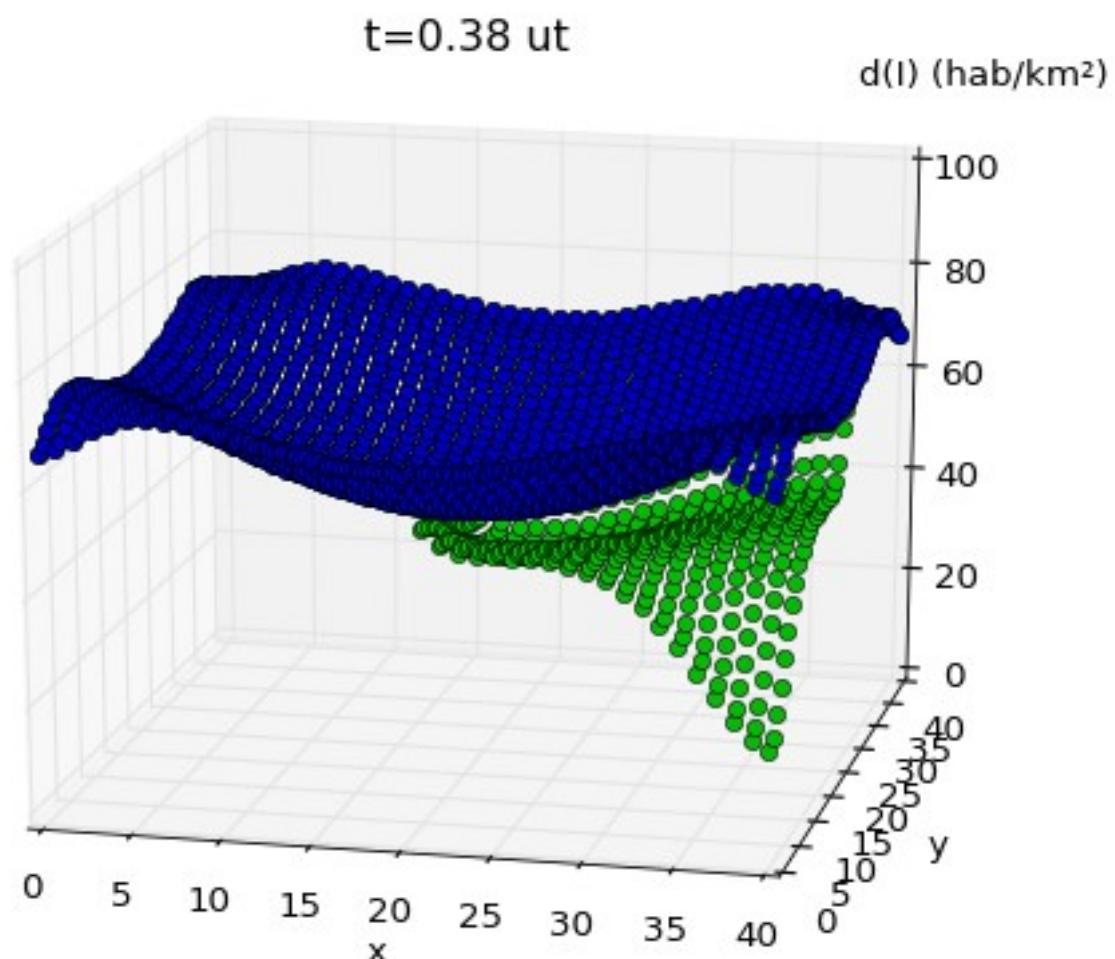
permettent de visualiser l'effet d'une perturbation

et les fonctions `espace_adj_e(K,L,s,D,Id,Dm,N,x,N3,t,λ,μ,Q,zlim)` et `espace_uni_e(k,l,s,d,P,dm,N,x,N3,t,λ,μ,Q,zlim)`

calculent son efficacité.

Q est une matrice de taille $N3 * N3$ telle que $Q[i][j]$ vaut 1 si la perturbation est appliquée à la case (i,j) et 0 sinon.

Par exemple pour le premier exemple à $t = 0.38$ ut, on observe (avec en bleu la propagation sans perturbation et en vert la propagation perturbée pour une très forte perturbation appliquée uniquement dans le coin inférieur droit à 0.35 ut) :



Le problème est que tester toutes les combinaisons possibles de perturbations pour un nombre fixé de cases à perturber demande un temps de calcul totalement inexploitable.

On utilise aujourd’hui, pour les incendies en particulier, des perturbations (sous la forme de tranchées) en forme d’anneaux. Dans mon modèle, cela est aussi vrai pour les épidémies car leur comportement est similaire.

Toutefois cela ne marche pas bien en réalité pour les épidémies car la maladie peut se propager rapidement grâce en particulier aux transports (avions, bateaux, trains, voitures) d’un lieu à un autre pouvant être éloignés, ce qui est impossible pour un incendie.

J’ai négligé cet aspect pour ne m’intéresser qu’aux configurations en anneaux car le temps de calcul l’exigeait.

J'ai créé de nouvelles fonctions dans ce but.

Les fonctions **espace_adj_elieu(K,L,s,D,Id,Dm,N,x,N3,t,λ,μ,p,R0,zlim)**
et **espace_uni_elieu(k,l,s,d,P,dm,N,x,N3,t,λ,μ,p,R0,zlim)**

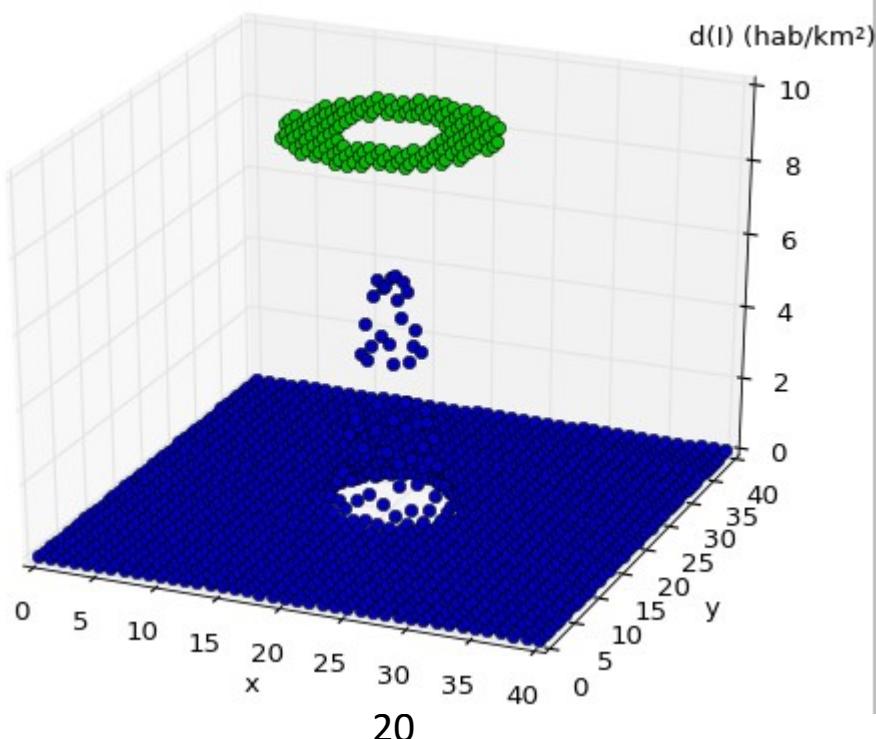
Permettent :

- De visualiser le propagation à l'instant t.
- De calculer le rayon optimal de l'anneau sur lequel il faut appliquer la perturbation.
- De calculer l'efficacité de cette perturbation et le nombre d'individus qui auraient été touchés sans elle.

R0 est la liste des rayons à tester.

Voici un exemple de résultat (la zone en vert est l'anneau optimal) :

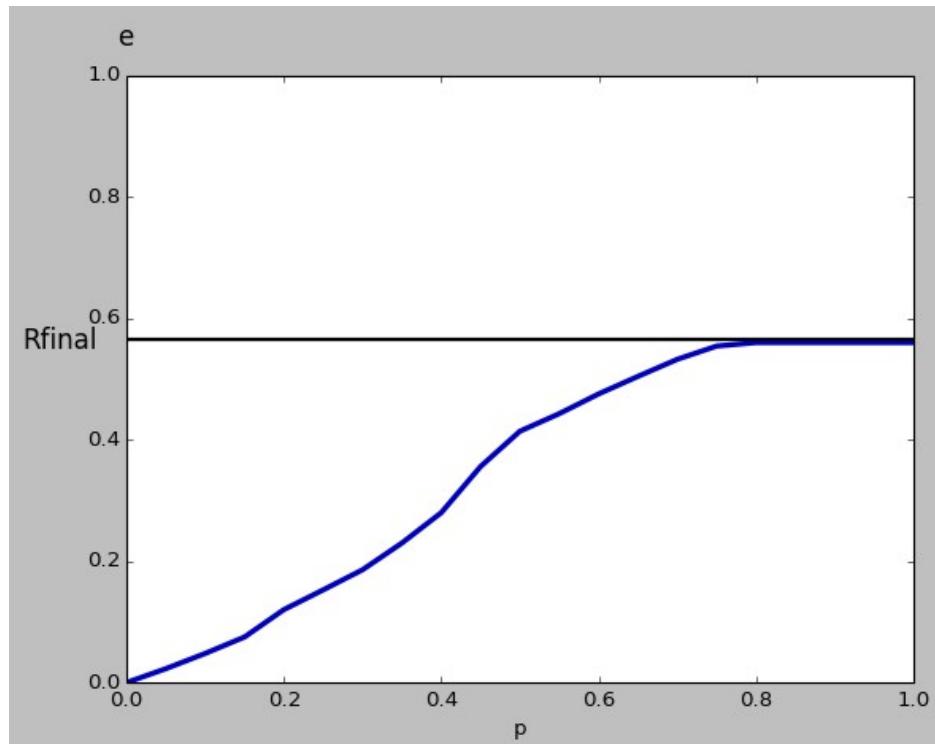
```
espace_uni_elieu(1200000,120,1.5,100,67000000,0.5,1000,1,41,0.2,0.8,0.1,0.1,linspace(0,1,21),10) :  
t=0.2 ut           e=0.04805           Rfinal=0.5642
```



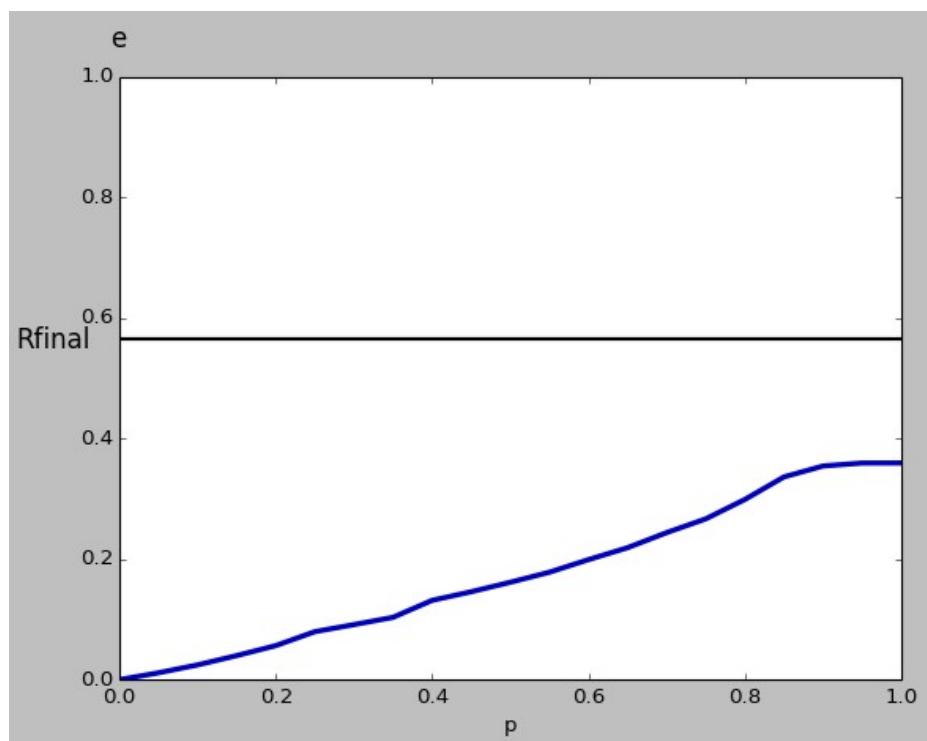
On peut tracer l'efficacité optimale en fonction de la proportion de cases perturbées.

Par exemple pour la même propagation,

Pour $\lambda = 0.8$ et $\mu = 0.1$:



et pour $\lambda = 0.2$ et $\mu = 0$:



Conclusion : intérêts et limites des modèles

Intérêts :

- On connaît l'évolution de la propagation ainsi que l'efficacité d'une éventuelle perturbation en fonction de leurs paramètres.
- Cela permet non seulement de prévoir mais aussi de connaître les progrès qu'on doit faire pour atteindre un objectif fixé.

Limites :

- Il m'est impossible de comparer les résultats à la réalité car je n'ai trouvé aucune étude utilisant ce modèle ni aucun relevé réel d'une épidémie ou d'un incendie.
- Pour les épidémies, les naissances, les morts non dues à la maladie et la période d'incubation sont négligées.
- L'hypothèse de lois sans vieillissement est forte.
- Des relevés réels sont nécessaires pour connaître les caractéristiques des perturbations que l'on applique.

Programmes :

Seule d n'est pas entrée en unité du système international mais en hab/km², c'est pourquoi on la divise par 10⁶

```
1  from numpy import*
2  from pylab import*
3  from matplotlib.pyplot import*
4  from mpl_toolkits.mplot3d import Axes3D
5
6  def simple(k,l,s,d,P,N,x) :
7      d=d/1000000
8      S=[1-1/P]+[0]*N
9      I=[1/P]+[0]*N
10     R=[0]*(N+1)
11     h=x/N
12     for i in range(N):
13         s1=-k*d*s*S[i]*I[i]
14         i1=k*d*s*S[i]*I[i]-l*I[i]
15         r1=l*I[i]
16         S2=S[i]+(h/2)*s1
17         I2=I[i]+(h/2)*i1
18         s2=-k*d*s*S2*I2
19         i2=k*d*s*S2*I2-l*I2
20         r2=l*I2
21         S3=S[i]+(h/2)*s2
22         I3=I[i]+(h/2)*i2
23         s3=-k*d*s*S3*I3
24         i3=k*d*s*S3*I3-l*I3
25         r3=l*I3
26         S4=S[i]+h*s3
27         I4=I[i]+h*i3
28         s4=-k*d*s*S4*I4
29         i4=k*d*s*S4*I4-l*I4
30         r4=l*I4
31         S[i+1]=S[i]+(h/6)*(s1+2*s2+2*s3+s4)
32         I[i+1]=I[i]+(h/6)*(i1+2*i2+2*i3+i4)
33         R[i+1]=R[i]+(h/6)*(r1+2*r2+2*r3+r4)
34     X=linspace(0,x,N+1)
35     ax=figure().gca()
36     ax.set_xlim(0,1)
37     ax.set_xlabel('t (ut)')
```

```

38     ax.text(x*0.41,1.05,'S',color=(0,0,0.75),size=30)
39     ax.text(x*0.5,1.05,'I',color=(0,0.5,0),size=30)
40     ax.text(x*0.57,1.05,'R',color=(0.75,0,0),size=30)
41     ax.plot(X,S,color=(0,0,0.75),linewidth=5)
42     ax.plot(X,I,color=(0,0.5,0),linewidth=5)
43     ax.plot(X,R,color=(0.75,0,0),linewidth=5)
44
45 def simple_impact(k,l,s,d,P,t,λ,μ,N,x) :
46     d=d/1000000
47     S=[1-1/P]+[0]*N
48     I=[1/P]+[0]*N
49     R=[0]*(N+1)
50     Si=[1-1/P]+[0]*N
51     Ii=[1/P]+[0]*N
52     Ri=[0]*(N+1)
53     h=x/N
54     i=0
55     while i<t/h :
56         s1=-k*d*s*Si[i]*I[i]
57         il=k*d*s*S[i]*I[i]-l*I[i]
58         rl=l*I[i]
59         S2=S[i]+(h/2)*s1
60         I2=I[i]+(h/2)*il
61         s2=-k*d*s*S2*I2
62         i2=k*d*s*S2*I2-l*I2
63         r2=l*I2
64         S3=S[i]+(h/2)*s2
65         I3=I[i]+(h/2)*i2
66         s3=-k*d*s*S3*I3
67         i3=k*d*s*S3*I3-l*I3
68         r3=l*I3
69         S4=S[i]+h*s3
70         I4=I[i]+h*i3
71         s4=-k*d*s*S4*I4
72         i4=k*d*s*S4*I4-l*I4
73         r4=l*I4
74         S[i+1]=S[i]+(h/6)*(s1+2*s2+2*s3+s4)
75         I[i+1]=I[i]+(h/6)*(il+2*i2+2*i3+i4)
76         R[i+1]=R[i]+(h/6)*(rl+2*r2+2*r3+r4)
77         Si[i+1]=S[i+1]
78         Ii[i+1]=I[i+1]
79         Ri[i+1]=R[i+1]
80         i=i+1
81     ki=k*(1-λ)
82     li=l/(1-μ)
83     while i<N :
84         s1=-k*d*s*S[i]*I[i]
85         il=k*d*s*S[i]*I[i]-l*I[i]
86         rl=l*I[i]
87         S2=S[i]+(h/2)*s1
88         I2=I[i]+(h/2)*il
89         s2=-k*d*s*S2*I2
90         i2=k*d*s*S2*I2-l*I2
91         r2=l*I2
92         S3=S[i]+(h/2)*s2
93         I3=I[i]+(h/2)*i2
94         s3=-k*d*s*S3*I3
95         i3=k*d*s*S3*I3-l*I3
96         r3=l*I3
97         S4=S[i]+h*s3
98         I4=I[i]+h*i3
99         s4=-k*d*s*S4*I4
100        i4=k*d*s*S4*I4-l*I4
101        r4=l*I4
102        S[i+1]=S[i]+(h/6)*(s1+2*s2+2*s3+s4)
103        I[i+1]=I[i]+(h/6)*(il+2*i2+2*i3+i4)
104        R[i+1]=R[i]+(h/6)*(rl+2*r2+2*r3+r4)
105        s1=-ki*d*s*Si[i]*Ii[i]
106        il=ki*d*s*Si[i]*Ii[i]-li*Ii[i]
107        rl=li*Ii[i]
108        S2=Si[i]+(h/2)*s1
109        I2=Ii[i]+(h/2)*il
110        s2=-ki*d*s*S2*I2
111        i2=ki*d*s*S2*I2-li*I2

```

```

112     r2=li*I2
113     S3=Si[i]+(h/2)*s2
114     I3=Ii[i]+(h/2)*i2
115     s3=-ki*d*s*S3*I3
116     i3=ki*d*s*S3*I3-li*I3
117     r3=li*I3
118     S4=Si[i]+h*s3
119     I4=Ii[i]+h*i3
120     s4=-ki*d*s*S4*I4
121     i4=ki*d*s*S4*I4-li*I4
122     r4=li*I4
123     Si[i+1]=Si[i]+(h/6)*(s1+2*s2+2*s3+s4)
124     Ii[i+1]=Ii[i]+(h/6)*(i1+2*i2+2*i3+i4)
125     Ri[i+1]=Ri[i]+(h/6)*(r1+2*r2+2*r3+r4)
126     i=i+1
127 X=linspace(0,x,N+1)
128 ax=figure().gca()
129 ax.set_ylim(0,1)
130 ax.set_xlabel('t (ut)')
131 ax.text(x*0.24,1.05,'S',color=(0,0,0.75),size=30)
132 ax.text(x*0.32,1.05,'I',color=(0,0.5,0),size=30)
133 ax.text(x*0.38,1.05,'R',color=(0.75,0,0),size=30)
134 ax.text(x*0.46,1.05,'Si',color=(0.8,0,1),size=30)
135 ax.text(x*0.58,1.05,'Ii',color=(0,0,0),size=30)
136 ax.text(x*0.68,1.05,'Ri',color=(1,0.7,0),size=30)
137 ax.plot(X, Si, color=(0.8,0,1), linewidth=3)
138 ax.plot(X, Ii, color=(0,0,0), linewidth=3)
139 ax.plot(X, Ri, color=(1,0.7,0), linewidth=3)
140 ax.plot(X, S, color=(0,0,0.75), linewidth=3)
141 ax.plot(X, I, color=(0,0.5,0), linewidth=3)
142 ax.plot(X, R, color=(0.75,0,0), linewidth=3)
143 ax.plot([t,t],[0,1],color=(0,0,0), linewidth=2)
144
145 def simple_e(k,l,s,d,P,t,lambda,mu,N,x) :
146     d=d/1000000
147     S=[1-1/P]+[0]*N
148     I=[1/P]+[0]*N
149     Si=[1-1/P]+[0]*N
150     Ii=[1/P]+[0]*N
151     h=x/N
152     i=0
153     while i<t/h :
154         s1=-k*d*s*S[i]*I[i]
155         i1=k*d*s*S[i]*I[i]-l*I[i]
156         S2=S[i]+(h/2)*s1
157         I2=I[i]+(h/2)*i1
158         s2=-k*d*s*S2*I2
159         i2=k*d*s*S2*I2-l*I2
160         S3=S[i]+(h/2)*s2
161         I3=I[i]+(h/2)*i2
162         s3=-k*d*s*S3*I3
163         i3=k*d*s*S3*I3-l*I3
164         S4=Si[i]+h*s3
165         I4=Ii[i]+h*i3
166         s4=-k*d*s*S4*I4
167         i4=k*d*s*S4*I4-l*I4
168         Si[i+1]=S[i]+(h/6)*(s1+2*s2+2*s3+s4)
169         Ii[i+1]=Ii[i]+(h/6)*(i1+2*i2+2*i3+i4)
170         Si[i+1]=S[i+1]
171         Ii[i+1]=Ii[i+1]
172         i=i+1
173     ki=k*(1-lambda)
174     li=l/(1-mu)
175     while i<N :
176         s1=-k*d*s*S[i]*I[i]
177         i1=k*d*s*S[i]*I[i]-l*I[i]
178         S2=S[i]+(h/2)*s1
179         I2=I[i]+(h/2)*i1
180         s2=-k*d*s*S2*I2
181         i2=k*d*s*S2*I2-l*I2
182         S3=S[i]+(h/2)*s2
183         I3=I[i]+(h/2)*i2
184         s3=-k*d*s*S3*I3
185         i3=k*d*s*S3*I3-l*I3

```

```

186     S4=S[i]+h*s3
187     I4=I[i]+h*i3
188     s4=-k*d*s*S4*I4
189     i4=k*d*s*S4*I4-l*I4
190     S[i+1]=S[i]+(h/6)*(s1+2*s2+2*s3+s4)
191     I[i+1]=I[i]+(h/6)*(i1+2*i2+2*i3+i4)
192     s1=-ki*d*s*Si[i]*Ii[i]
193     i1=ki*d*s*Si[i]*Ii[i]-li*Ii[i]
194     S2=Si[i]+(h/2)*s1
195     I2=Ii[i]+(h/2)*i1
196     s2=-ki*d*s*S2*I2
197     i2=ki*d*s*S2*I2-li*I2
198     S3=Si[i]+(h/2)*s2
199     I3=Ii[i]+(h/2)*i2
200     s3=-ki*d*s*S3*I3
201     i3=ki*d*s*S3*I3-li*I3
202     S4=Si[i]+h*s3
203     I4=Ii[i]+h*i3
204     s4=-ki*d*s*S4*I4
205     i4=ki*d*s*S4*I4-li*I4
206     Si[i+1]=Si[i]+(h/6)*(s1+2*s2+2*s3+s4)
207     Ii[i+1]=Ii[i]+(h/6)*(i1+2*i2+2*i3+i4)
208     i=i+1
209
210     return(Si[N]-S[N])
211
212 def simple_et(k,l,s,d,P,λ,μ,N,x,N2,x2) :
213     E=[0]*(N2+1)
214     h2=x2/N2
215     for j in range(N2+1) :
216         t=j*h2
217         E[j]=simple_e(k,l,s,d,P,t,λ,μ,N,x)
218     X=linspace(0,x2,N2+1)
219     ax=figure().gca()
220     ax.set_ylim(0,1)
221     ax.set_xlabel('t (ut)')
222     ax.set_ylabel('e')
223     ax.plot(X,E,color=(0,0,0.75),linewidth=5)
224
225 def simple_eλ(k,l,s,d,P,t,μ,N,x,N2) :
226     E=[0]*(N2+1)
227     h2=1/N2
228     for j in range(N2+1) :
229         λ=j*h2
230         E[j]=simple_e(k,l,s,d,P,t,λ,μ,N,x)
231     X=linspace(0,1,N2+1)
232     ax=figure().gca()
233     ax.set_ylim(0,1)
234     ax.set_xlabel('lambda')
235     ax.set_ylabel('e')
236     ax.plot(X,E,color=(0,0,0.75),linewidth=5)
237
238 def simple_eμ(k,l,s,d,P,t,λ,N,x,N2) :
239     E=[0]*N2
240     h2=1/N2
241     for j in range(N2) :
242         μ=j*h2
243         E[j]=simple_e(k,l,s,d,P,t,λ,μ,N,x)
244     X=linspace(0,1,N2)
245     ax=figure().gca()
246     ax.set_ylim(0,1)
247     ax.set_xlabel('mu')
248     ax.set_ylabel('e')
249     ax.plot(X,E,color=(0,0,0.75),linewidth=5)
250
251 def simple_etλ(k,l,s,d,P,μ,N,x,N2,x2) :
252     d=d/1000000
253     fig=figure()
254     ax=fig.gca(projection='3d')
255     u=linspace(0,x2,N2)
256     v=linspace(0,1,N2)
257     X=zeros((N2,N2))
258     Y=zeros((N2,N2))
259     Z=zeros((N2,N2))
260     for p in range(N2) :

```

```

260     for q in range(N2) :
261         X[p][q]=u[p]
262         Y[p][q]=v[q]
263         t=u[p]
264         λ=v[q]
265         Z[p][q]=simple_e(k,l,s,d,P,t,λ,μ,N,x)
266     ax.set_zlim3d(0,1)
267     ax.set_xlabel('t (ut)')
268     ax.set_ylabel('lambda')
269     ax.set_zlabel('e')
270     ax.plot_surface(X,Y,Z,color=(0,0,0.75))
271
272 def espace(K,L,s,D,Id,Dm,T,N,x,N3,zlim) :
273     u=linspace(0,N3-1,N3)
274     h=x/N
275     X=[0]*(N3**2)
276     Y=[0]*(N3**2)
277     S=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
278     I=[Id]+[[[0 for i in range(N3)] for j in range(N3)] for i in range(N)]
279     S0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
280     I0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
281     for i in range(N3) :
282         for j in range(N3) :
283             X[N3*i+j]=u[i]
284             Y[N3*i+j]=u[j]
285             S[0][i][j]=1-I[0][i][j]
286     for p in range(N) :
287         for i in range(N3) :
288             for j in range(N3) :
289                 k=K[i][j]
290                 l=L[i][j]
291                 d=D[i][j]/1000000
292                 s1=-k*d*s*S[p][i][j]*I[p][i][j]
293                 i1=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
294                 S2=S[p][i][j]+(h/2)*s1
295                 I2=I[p][i][j]+(h/2)*i1
296                 s2=-k*d*s*S2*I2
297                 i2=k*d*s*S2*I2-l*I2
298                 S3=S[p][i][j]+(h/2)*s2
299                 I3=I[p][i][j]+(h/2)*i2
300                 s3=-k*d*s*S3*I3
301                 i3=k*d*s*S3*I3-l*I3
302                 S4=S[p][i][j]+h*s3
303                 I4=I[p][i][j]+h*i3
304                 s4=-k*d*s*S4*I4
305                 i4=k*d*s*S4*I4-l*I4
306                 S0[p+1][i][j]=S[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
307                 I0[p+1][i][j]=I[p][i][j]+(h/6)*(i1+2*i2+2*i3+i4)
308                 S[p+1][i][j]=S0[p+1][i][j]
309                 I[p+1][i][j]=I0[p+1][i][j]
310                 S[p][i][j]=D[i][j]*S[p][i][j]
311                 I[p][i][j]=D[i][j]*I[p][i][j]
312     for i in range(N3) :
313         for j in range(N3) :
314             for i2 in range(i,N3) :
315                 for j2 in range(N3) :
316                     if (i2>i or (i2==i and j2>j)) :
317                         dm=Dm[i][j][i2][j2]/1000000
318                         d1=D[i][j]/1000000
319                         d2=D[i2][j2]/1000000
320                         S[p+1][i][j]=S[p+1][i][j]+(dm/d1)*(S0[p+1][i2][j2]-
321                         S0[p+1][i][j])
322                         I[p+1][i][j]=I[p+1][i][j]+(dm/d1)*(I0[p+1][i2][j2]-
323                         I0[p+1][i][j])
324                         S[p+1][i2][j2]=S[p+1][i2][j2]+(dm/d2)*(S0[p+1][i]
325                         [j]-S0[p+1][i2][j2])
326                         I[p+1][i2][j2]=I[p+1][i2][j2]+(dm/d2)*(I0[p+1][i]
327                         [j]-I0[p+1][i2][j2])
328             for i in range(N3) :
329                 for j in range(N3) :
330                     S[N][i][j]=D[i][j]*S[N][i][j]
331                     I[N][i][j]=D[i][j]*I[N][i][j]
332             n=len(T)
333             for q in range(n) :

```

```

330     p=int(T[q]/h)
331     espace_sub(I,X,Y,p,h,N3,zlim)
332
333 def espace_sub(I,X,Y,p,h,N3,zlim) :
334     Z=[0]*(N3**2)
335     for i in range(N3) :
336         for j in range(N3) :
337             Z[N3*i+j]=I[p][i][j]
338     fig=figure()
339     ax=fig.gca(projection='3d')
340     ax.set_zlim3d(0,zlim)
341     ax.set_title('t='+(str(int(p*h*100)/100))+' ut')
342     ax.set_xlabel('x')
343     ax.set_ylabel('y')
344     ax.text(N3*0.95,N3*0.75,zlim*1.2,'d(I) (hab/km2)',color=(0,0,0),size=12)
345     ax.plot(X,Y,Z,'o',color=(0,0,0.75))
346
347 def espace_adj(K,L,s,D,Id,Dm,T,N,x,N3,zlim) :
348     u=linspace(0,N3-1,N3)
349     h=x/N
350     X=[0]*(N3**2)
351     Y=[0]*(N3**2)
352     S=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
353     I=[Id]+[[0 for i in range(N3)] for j in range(N3)] for i in range(N)]
354     S0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
355     I0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
356     for i in range(N3) :
357         for j in range(N3) :
358             X[N3*i+j]=u[i]
359             Y[N3*i+j]=u[j]
360             S[0][i][j]=1-I[0][i][j]
361     for p in range(N) :
362         for i in range(N3) :
363             for j in range(N3) :
364                 k=K[i][j]
365                 l=L[i][j]
366                 d=D[i][j]/1000000
367                 s1=-k*d*s*S[p][i][j]*I[p][i][j]
368                 i1=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
369                 S2=S[p][i][j]+(h/2)*s1
370                 I2=I[p][i][j]+(h/2)*i1
371                 s2=-k*d*s*S2*I2
372                 i2=k*d*s*S2*I2-l*I2
373                 S3=S[p][i][j]+(h/2)*s2
374                 I3=I[p][i][j]+(h/2)*i2
375                 s3=-k*d*s*S3*I3
376                 i3=k*d*s*S3*I3-l*I3
377                 S4=S[p][i][j]+h*s3
378                 I4=I[p][i][j]+h*i3
379                 s4=-k*d*s*S4*I4
380                 i4=k*d*s*S4*I4-l*I4
381                 S0[p+1][i][j]=S[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
382                 I0[p+1][i][j]=I[p][i][j]+(h/6)*(i1+2*i2+2*i3+i4)
383                 S[p+1][i][j]=S0[p+1][i][j]
384                 I[p+1][i][j]=I0[p+1][i][j]
385                 S[p][i][j]=D[i][j]*S[p][i][j]
386                 I[p][i][j]=D[i][j]*I[p][i][j]
387                 S[p+1][0][0]=S[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])*(S0[p+1][1][0]-
388                 S0[p+1][0][0])+((Dm[0][0][0][1]/D[0][0])*(S0[p+1][0][1]-S0[p+1][0][0]))
389                 I[p+1][0][0]=I[p+1][0][0]+((Dm[0][0][1][0]/D[0][0])*(I0[p+1][1][0]-
390                 I0[p+1][0][0])+((Dm[0][0][0][1]/D[0][0])*(I0[p+1][0][1]-I0[p+1][0][0]))
391                 S[p+1][0][N3-1]=S[p+1][0][N3-1]+((Dm[0][N3-1][1][N3-1]/D[0]
392                 [N3-1]))*(S0[p+1][1][N3-1]-S0[p+1][0][N3-1])+((Dm[0][N3-2][0][N3-1]/D[0]
393                 [N3-1]))*(S0[p+1][0][N3-2]-S0[p+1][0][N3-1])
394                 I[p+1][0][N3-1]=I[p+1][0][N3-1]+((Dm[0][N3-1][1][N3-1]/D[0]
395                 [N3-1]))*(I0[p+1][1][N3-1]-I0[p+1][0][N3-1])+((Dm[0][N3-2][0][N3-1]/D[0]
396                 [N3-1]))*(I0[p+1][0][N3-2]-I0[p+1][0][N3-1])
397                 S[p+1][N3-1][0]=S[p+1][N3-1][0]+((Dm[N3-1][0][N3-1][1]/D[N3-1]
398                 [0]))*(S0[p+1][N3-1][1]-S0[p+1][N3-1][0])+((Dm[N3-2][0][N3-1][0]/D[N3-1]
399                 [0]))*(S0[p+1][N3-2][0]-S0[p+1][N3-1][0])
400                 I[p+1][N3-1][0]=I[p+1][N3-1][0]+((Dm[N3-1][0][N3-1][1]/D[N3-1]
401                 [0]))*(I0[p+1][N3-1][1]-I0[p+1][N3-1][0])+((Dm[N3-2][0][N3-1][0]/D[N3-1]
402                 [0]))*(I0[p+1][N3-2][0]-I0[p+1][N3-1][0])

```

```

393     S[p+1][N3-1][N3-1]=S[p+1][N3-1]+(Dm[N3-1][N3-2][N3-1]
394     [N3-1]/D[N3-1][N3-1])*(S0[p+1][N3-1][N3-2]-S0[p+1][N3-1][N3-1])+(Dm[N3-2][N3-1]
395     [N3-1][N3-1]/D[N3-1][N3-1])*(S0[p+1][N3-2][N3-1]-S0[p+1][N3-1][N3-1])
396     I[p+1][N3-1][N3-1]=I[p+1][N3-1][N3-1]+(Dm[N3-1][N3-2][N3-1]
397     [N3-1]/D[N3-1][N3-1])*(I0[p+1][N3-1][N3-2]-I0[p+1][N3-1][N3-1])+(Dm[N3-2][N3-1]
398     [N3-1][N3-1]/D[N3-1][N3-1])*(I0[p+1][N3-2][N3-1]-I0[p+1][N3-1][N3-1])
399     for j in range(1,N3-1) :
400         S[p+1][0][j]=S[p+1][0][j]+(Dm[0][j-1][0][j]/D[0][j])*(S0[p+1][0]
401         [j-1]-S0[p+1][0][j])+(Dm[0][j][0][j+1]/D[0][j])*(S0[p+1][0][j+1]-S0[p+1][0]
402         [j])+(Dm[0][j][1][j]/D[0][j])*(S0[p+1][1][j]-S0[p+1][0][j])
403         I[p+1][0][j]=I[p+1][0][j]+(Dm[0][j-1][0][j]/D[0][j])*(I0[p+1][0]
404         [j-1]-I0[p+1][0][j])+(Dm[0][j][0][j+1]/D[0][j])*(I0[p+1][0][j+1]-I0[p+1][0]
405         [j])+(Dm[0][j][1][j]/D[0][j])*(I0[p+1][1][j]-I0[p+1][0][j])
406         S[p+1][N3-1][j]=S[p+1][N3-1][j]+(Dm[N3-1][j-1][N3-1][j]/D[N3-1]
407         [j])*(S0[p+1][N3-1][j-1]-S0[p+1][N3-1][j])+(Dm[N3-1][j][N3-1][j+1]/D[N3-1]
408         [j])*(S0[p+1][N3-1][j+1]-S0[p+1][N3-1][j])+(Dm[N3-2][j][N3-1][j]/D[N3-1]
409         [j])*(S0[p+1][N3-2][j]-S0[p+1][N3-1][j])
410         for i in range(1,N3-1) :
411             S[p+1][i][0]=S[p+1][i][0]+(Dm[i-1][0][i][0]/D[i][0])*(S0[p+1][i-1]
412             [0]-S0[p+1][i][0])+(Dm[i][0][i+1][0]/D[i][0])*(S0[p+1][i+1][0]-S0[p+1][i]
413             [0])+(Dm[i][0][i][1]/D[i][0])*(S0[p+1][i][1]-S0[p+1][i][0])
414             I[p+1][i][0]=I[p+1][i][0]+(Dm[i-1][0][i][0]/D[i][0])*(I0[p+1][i-1]
415             [0]-I0[p+1][i][0])+(Dm[i][0][i+1][0]/D[i][0])*(I0[p+1][i+1][0]-I0[p+1][i]
416             [0])+(Dm[i][0][i][1]/D[i][0])*(I0[p+1][i][1]-I0[p+1][i][0])
417             S[p+1][i][N3-1]=S[p+1][i][N3-1]+(Dm[i-1][N3-1][i][N3-1]/D[i]
418             [N3-1])*(S0[p+1][i-1][N3-1]-S0[p+1][i][N3-1])+(Dm[i][N3-1][i+1][N3-1]/D[i]
419             [N3-1])*(S0[p+1][i+1][N3-1]-S0[p+1][i][N3-1])+(Dm[i][N3-2][i][N3-1]/D[i]
420             [N3-1])*(S0[p+1][i][N3-2]-S0[p+1][i][N3-1])
421             for i in range(1,N3-1) :
422                 for j in range(1,N3-1) :
423                     S[p+1][i][j]=S[p+1][i][j]+(Dm[i-1][j][i][j]/D[i][j])*(S0[p+1]
424                     [i-1][j]-S0[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])*(S0[p+1][i+1][j]-S0[p+1][i]
425                     [j])+(Dm[i][j-1][i][j]/D[i][j])*(S0[p+1][i][j-1]-S0[p+1][i][j])+(Dm[i][j][i]
426                     [j+1]/D[i][j])*(S0[p+1][i][j+1]-S0[p+1][i][j])
427                     I[p+1][i][j]=I[p+1][i][j]+(Dm[i-1][j][i][j]/D[i][j])*(I0[p+1]
428                     [i-1][j]-I0[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])*(I0[p+1][i+1][j]-I0[p+1][i]
429                     [j])+(Dm[i][j-1][i][j]/D[i][j])*(I0[p+1][i][j-1]-I0[p+1][i][j])+(Dm[i][j][i]
430                     [j+1]/D[i][j])*(I0[p+1][i][j+1]-I0[p+1][i][j])
431                     for i in range(N3) :
432                         for j in range(N3) :
433                             S[N][i][j]=D[i][j]*S[N][i][j]
434                             I[N][i][j]=D[i][j]*I[N][i][j]
435                             n=len(T)
436                             for q in range(n) :
437                                 p=int(T[q]/h)
438                                 espace_sub(I,X,Y,p,h,N3,zlim)
439
440 def espace_uni(k,l,s,d,P,dm,T,N,x,N3,zlim) :
441     d=d/1000000
442     dm=dm/1000000
443     Id=[[0 for i in range(N3)] for j in range(N3)]
444     a=int((N3-1)/2)
445     Id[a][a]=(N3**2)/P
446     u=linspace(0,N3-1,N3)
447     h=x/N
448     X=[0]*(N3**2)
449     Y=[0]*(N3**2)
450     S=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
451     I=[Id]+[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
452     S0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
453     I0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
454     for i in range(N3) :
455         for j in range(N3) :
456             X[N3*i+j]=u[i]
457             Y[N3*i+j]=u[j]

```

```

436     S[0][i][j]=1-I[0][i][j]
437     for p in range(N) :
438         for i in range(N3) :
439             for j in range(N3) :
440                 s1=-k*d*s*S[p][i][j]*I[p][i][j]
441                 i1=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
442                 S2=S[p][i][j]+(h/2)*s1
443                 I2=I[p][i][j]+(h/2)*i1
444                 s2=-k*d*s*S2*I2
445                 i2=k*d*s*S2*I2-l*I2
446                 S3=S[p][i][j]+(h/2)*s2
447                 I3=I[p][i][j]+(h/2)*i2
448                 s3=-k*d*s*S3*I3
449                 i3=k*d*s*S3*I3-l*I3
450                 S4=S[p][i][j]+h*s3
451                 I4=I[p][i][j]+h*i3
452                 s4=-k*d*s*S4*I4
453                 i4=k*d*s*S4*I4-l*I4
454                 S0[p+1][i][j]=S[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
455                 I0[p+1][i][j]=I[p][i][j]+(h/6)*(i1+2*i2+2*i3+i4)
456                 S[p+1][i][j]=S0[p+1][i][j]
457                 I[p+1][i][j]=I0[p+1][i][j]
458                 S[p][i][j]=d*S[p][i][j]*1000000
459                 I[p][i][j]=d*I[p][i][j]*1000000
460                 S[p+1][0][0]=S[p+1][0][0]+(dm/d)*(S0[p+1][1][0]+S0[p+1][0][1]-2*S0[p+1]
461 [0][0])
462                 I[p+1][0][0]=I[p+1][0][0]+(dm/d)*(I0[p+1][1][0]+I0[p+1][0][1]-2*I0[p+1]
463 [0][0])
464                 S[p+1][0][N3-1]=S[p+1][0][N3-1]+(dm/d)*(S0[p+1][1][N3-1]+S0[p+1][0]
465 [N3-2]-2*S0[p+1][0][N3-1])
466                 I[p+1][0][N3-1]=I[p+1][0][N3-1]+(dm/d)*(I0[p+1][1][N3-1]+I0[p+1][0]
467 [N3-2]-2*I0[p+1][0][N3-1])
468                 S[p+1][N3-1][0]=S[p+1][N3-1][0]+(dm/d)*(S0[p+1][N3-1][1]+S0[p+1][N3-2]
469 [0]-2*S0[p+1][N3-1][0])
470                 I[p+1][N3-1][0]=I[p+1][N3-1][0]+(dm/d)*(I0[p+1][N3-1][1]+I0[p+1][N3-2]
471 [0]-2*I0[p+1][N3-1][0])
472                 S[p+1][N3-1][N3-1]=S[p+1][N3-1][N3-1]+(dm/d)*(S0[p+1][N3-1][1]
473 [N3-2]+S0[p+1][N3-2][N3-1]-2*S0[p+1][N3-1][N3-1])
474                 I[p+1][N3-1][N3-1]=I[p+1][N3-1][N3-1]+(dm/d)*(I0[p+1][N3-1]
475 [N3-2]+I0[p+1][N3-2][N3-1]-2*I0[p+1][N3-1][N3-1])
476                 for j in range(1,N3-1) :
477                     S[p+1][0][j]=S[p+1][0][j]+(dm/d)*(S0[p+1][0][j-1]+S0[p+1][0]
478 [j+1]+S0[p+1][1][j]-3*S0[p+1][0][j])
479                     I[p+1][0][j]=I[p+1][0][j]+(dm/d)*(I0[p+1][0][j-1]+I0[p+1][0]
480 [j+1]+I0[p+1][1][j]-3*I0[p+1][0][j])
481                     S[p+1][N3-1][j]=S[p+1][N3-1][j]+(dm/d)*(S0[p+1][N3-1][j-1]+S0[p+1]
482 [N3-1][j+1]+S0[p+1][N3-2][j]-3*S0[p+1][N3-1][j])
483                     I[p+1][N3-1][j]=I[p+1][N3-1][j]+(dm/d)*(I0[p+1][N3-1][j-1]+I0[p+1]
484 [N3-1][j+1]+I0[p+1][N3-2][j]-3*I0[p+1][N3-1][j])
485                     for i in range(1,N3-1) :
486                         for j in range(1,N3-1) :
487                             S[p+1][i][j]=S[p+1][i][j]+(dm/d)*(S0[p+1][i-1][j]+S0[p+1][i+1]
488 [j]+S0[p+1][i][j-1]+S0[p+1][i][j+1]-4*S0[p+1][i][j])
489                             I[p+1][i][j]=I[p+1][i][j]+(dm/d)*(I0[p+1][i-1][j]+I0[p+1][i+1]
490 [j]+I0[p+1][i][j-1]+I0[p+1][i][j+1]-4*I0[p+1][i][j])
491                         for i in range(N3) :
492                             for j in range(N3) :
493                                 S[N][i][j]=d*S[N][i][j]*1000000
494                                 I[N][i][j]=d*I[N][i][j]*1000000
495
496 n=len(T)
497 for q in range(n) :
498     p=int(T[q]/h)
499     espace_sub(I,X,Y,p,h,N3,zlim)

```

```

491 def espace_adj_impact(K,L,s,D,Id,Dm,T,N,x,N3,t,λ,μ,Q,zlim) :
492     u=linspace(0,N3-1,N3)
493     h=x/N
494     X=[0]*(N3**2)
495     Y=[0]*(N3**2)
496     S=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)
497     I=[Id]+[[0 for i in range(N3)] for j in range(N3)] for i in range(N)]
498     S0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
499     I0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
500     Si=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
501     Ii=[Id]+[[0 for i in range(N3)] for j in range(N3)] for i in range(N)]
502     S0i=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
503     I0i=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
504     for i in range(N3) :
505         for j in range(N3) :
506             X[N3*i+j]=u[i]
507             Y[N3*i+j]=u[j]
508             S[0][i][j]=1-I[0][i][j]
509             Si[0][i][j]=1-I[0][i][j]
510     p=0
511     while p<t/h :
512         for i in range(N3) :
513             for j in range(N3) :
514                 k=K[i][j]
515                 l=L[i][j]
516                 d=D[i][j]/1000000
517                 s1=-k*d*s*S[p][i][j]*I[p][i][j]
518                 il=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
519                 S2=S[p][i][j]+(h/2)*s1
520                 I2=I[p][i][j]+(h/2)*il
521                 s2=-k*d*s*S2*I2
522                 i2=k*d*s*S2*I2-l*I2
523                 S3=S[p][i][j]+(h/2)*s2
524                 I3=I[p][i][j]+(h/2)*i2
525                 s3=-k*d*s*S3*I3
526                 i3=k*d*s*S3*I3-l*I3
527                 S4=S[p][i][j]+h*s3
528                 I4=I[p][i][j]+h*i3
529                 s4=-k*d*s*S4*I4
530                 i4=k*d*s*S4*I4-l*I4
531                 S0[p+1][i][j]=S[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
532                 I0[p+1][i][j]=I[p][i][j]+(h/6)*(il+2*i2+2*i3+i4)
533                 S[p+1][i][j]=S0[p+1][i][j]
534                 I[p+1][i][j]=I0[p+1][i][j]
535                 S[p][i][j]=D[i][j]*S[p][i][j]
536                 I[p][i][j]=D[i][j]*I[p][i][j]
537                 S[p+1][0][0]=S[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])*(S0[p+1][1][0]-
538                 S0[p+1][0][0])+((Dm[0][0][0][1]/D[0][0])*(S0[p+1][0][1]-S0[p+1][0][0]))
539                 I[p+1][0][0]=I[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])*(I0[p+1][1][0]-
540                 I0[p+1][0][0])+((Dm[0][0][0][1]/D[0][0])*(I0[p+1][0][1]-I0[p+1][0][0]))
541                 S[p+1][0][N3-1]=S[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0]
542                 [N3-1])*(S0[p+1][1][N3-1]-S0[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0]
543                 [N3-1])*(S0[p+1][0][N3-2]-S0[p+1][0][N3-1])
544                 I[p+1][0][N3-1]=I[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0]
545                 [N3-1])*(I0[p+1][1][N3-1]-I0[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0]
546                 [N3-1])*(I0[p+1][0][N3-2]-I0[p+1][0][N3-1])

```

```

547 [j-1]-I0[p+1][0][j])+(Dm[0][j][0][j+1]/D[0][j])*(I0[p+1][0][j+1]-I0[p+1][0]
548 [j])+(Dm[0][j][1][j]/D[0][j])*(I0[p+1][1][j]-I0[p+1][0][j])
549 S[p+1][N3-1][j]=S[p+1][N3-1][j]+(Dm[N3-1][j-1][N3-1][j]/D[N3-1]
550 [j])*(S0[p+1][N3-1][j-1]-S0[p+1][N3-1][j])+(Dm[N3-1][j][N3-1][j+1]/D[N3-1]
551 [j])*(S0[p+1][N3-1][j+1]-S0[p+1][N3-1][j])+(Dm[N3-2][j][N3-1][j]/D[N3-1]
552 [j])*(S0[p+1][N3-2][j]-S0[p+1][N3-1][j])
553 I[p+1][N3-1][j]=I[p+1][N3-1][j]+(Dm[N3-1][j-1][N3-1][j]/D[N3-1]
554 [j])*(I0[p+1][N3-1][j-1]-I0[p+1][N3-1][j])+(Dm[N3-1][j][N3-1][j+1]/D[N3-1]
555 [j])*(I0[p+1][N3-1][j+1]-I0[p+1][N3-1][j])+(Dm[N3-2][j][N3-1][j]/D[N3-1]
556 [j])*(I0[p+1][N3-2][j]-I0[p+1][N3-1][j])
557 for i in range(1,N3-1) :
558     S[p+1][i][0]=S[p+1][i][0]+(Dm[i-1][0][i][0]/D[i][0])*(S0[p+1][i-1]
559 [0]-S0[p+1][i][0])+(Dm[i][0][i+1][0]/D[i][0])*(S0[p+1][i+1][0]-S0[p+1][i]
560 [0])+(Dm[i][0][i][1]/D[i][0])*(S0[p+1][i][1]-S0[p+1][i][0])
561 I[p+1][i][0]=I[p+1][i][0]+(Dm[i-1][0][i][0]/D[i][0])*(I0[p+1][i-1]
562 [0]-I0[p+1][i][0])+(Dm[i][0][i+1][0]/D[i][0])*(I0[p+1][i+1][0]-I0[p+1][i]
563 [0])+(Dm[i][0][i][1]/D[i][0])*(I0[p+1][i][1]-I0[p+1][i][0])
564 S[p+1][i][N3-1]=S[p+1][i][N3-1]+(Dm[i-1][N3-1][i][N3-1]/D[i]
565 [N3-1])*(S0[p+1][i-1][N3-1]-S0[p+1][i][N3-1])+(Dm[i][N3-1][i+1][N3-1]/D[i]
566 [N3-1])*(S0[p+1][i+1][N3-1]-S0[p+1][i][N3-1])+(Dm[i][N3-2][i][N3-1]/D[i]
567 [N3-1])*(S0[p+1][i][N3-2]-S0[p+1][i][N3-1])
568 I[p+1][i][N3-1]=I[p+1][i][N3-1]+(Dm[i-1][N3-1][i][N3-1]/D[i]
569 [N3-1])*(I0[p+1][i-1][N3-1]-I0[p+1][i][N3-1])+(Dm[i][N3-1][i+1][N3-1]/D[i]
570 [N3-1])*(I0[p+1][i+1][N3-1]-I0[p+1][i][N3-1])+(Dm[i][N3-2][i][N3-1]/D[i]
571 [N3-1])*(I0[p+1][i][N3-2]-I0[p+1][i][N3-1])
572 for i in range(N3) :
573     for j in range(N3) :
574         S0i[p+1][i][j]=S0[p+1][i][j]
575         I0i[p+1][i][j]=I0[p+1][i][j]
576         Si[p+1][i][j]=S[p+1][i][j]
577         Ii[p+1][i][j]=I[p+1][i][j]
578         Si[p][i][j]=S[p][i][j]
579         Ii[p][i][j]=I[p][i][j]
580         p=p+1
581         Ki=[[0 for i in range(N3)] for j in range(N3)]
582         Li=[[0 for i in range(N3)] for j in range(N3)]
583         for i in range(N3) :
584             for j in range(N3) :
585                 if Q[i][j]==1 :
586                     Ki[i][j]=K[i][j]*(1-λ)
587                     Li[i][j]=L[i][j]/(1-μ)
588                 else :
589                     Ki[i][j]=K[i][j]
590                     Li[i][j]=L[i][j]
591         while p<N :
592             for i in range(N3) :
593                 for j in range(N3) :
594                     k=K[i][j]
595                     l=L[i][j]
596                     d=D[i][j]/1000000
597                     s1=-k*d*s*S[p][i][j]*I[p][i][j]
598                     i1=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
599                     S2=S[p][i][j]+(h/2)*s1
600                     I2=I[p][i][j]+(h/2)*i1
601                     s2=-k*d*s*S2*I2
602                     i2=k*d*s*S2*I2-l*I2
603                     S3=S[p][i][j]+(h/2)*s2
604                     I3=I[p][i][j]+(h/2)*i2
605                     s3=-k*d*s*S3*I3
606                     i3=k*d*s*S3*I3-l*I3
607                     S4=S[p][i][j]+h*s3
608                     I4=I[p][i][j]+h*i3
609                     s4=-k*d*s*S4*I4
610                     i4=k*d*s*S4*I4-l*I4

```

```

598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
      S0[p+1][i][j]=S[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
      I0[p+1][i][j]=I[p][i][j]+(h/6)*(i1+2*i2+2*i3+i4)
      S[p+1][i][j]=S0[p+1][i][j]
      I[p+1][i][j]=I0[p+1][i][j]
      S[p][i][j]=D[i][j]*S[p][i][j]
      I[p][i][j]=D[i][j]*I[p][i][j]
      S[p+1][0][0]=S[p+1][0][0]+(Dm[0][0][1]/D[0][0])* (S0[p+1][1][0]-
      S0[p+1][0][0])+(Dm[0][0][0][1]/D[0][0])* (S0[p+1][0][1]-S0[p+1][0][0])
      I[p+1][0][0]=I[p+1][0][0]+(Dm[0][0][1]/D[0][0])* (I0[p+1][1][0]-
      I0[p+1][0][0])+(Dm[0][0][0][1]/D[0][0])* (I0[p+1][0][1]-I0[p+1][0][0])
      S[p+1][0][N3-1]=S[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0]-
      [N3-1])* (S0[p+1][1][N3-1]-S0[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0]-
      [N3-1])* (S0[p+1][0][N3-2]-S0[p+1][0][N3-1])
      I[p+1][0][N3-1]=I[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0]-
      [N3-1])* (I0[p+1][1][N3-1]-I0[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0]-
      [N3-1])* (I0[p+1][0][N3-2]-I0[p+1][0][N3-1])
      S[p+1][N3-1][0]=S[p+1][N3-1][0]+(Dm[N3-1][0][N3-1][1]/D[N3-1]-
      [0])* (S0[p+1][N3-1][1]-S0[p+1][N3-1][0])+(Dm[N3-2][0][N3-1][0]/D[N3-1]-
      [0])* (S0[p+1][N3-2][0]-S0[p+1][N3-1][0])
      I[p+1][N3-1][0]=I[p+1][N3-1][0]+(Dm[N3-1][0][N3-1][1]/D[N3-1]-
      [0])* (I0[p+1][N3-1][1]-I0[p+1][N3-1][0])+(Dm[N3-2][0][N3-1][0]/D[N3-1]-
      [0])* (I0[p+1][N3-2][0]-I0[p+1][N3-1][0])
      S[p+1][N3-1][N3-1]=S[p+1][N3-1][N3-1]+(Dm[N3-1][N3-2][N3-1]-
      [N3-1]/D[N3-1][N3-1])* (S0[p+1][N3-1][N3-2]-S0[p+1][N3-1][N3-1])+(Dm[N3-2][N3-1]-
      [N3-1]/D[N3-1][N3-1])* (S0[p+1][N3-2][N3-1]-S0[p+1][N3-1][N3-1])
      I[p+1][N3-1][N3-1]=I[p+1][N3-1][N3-1]+(Dm[N3-1][N3-2][N3-1]-
      [N3-1]/D[N3-1][N3-1])* (I0[p+1][N3-1][N3-2]-I0[p+1][N3-1][N3-1])+(Dm[N3-2][N3-1]-
      [N3-1]/D[N3-1][N3-1])* (I0[p+1][N3-2][N3-1]-I0[p+1][N3-1][N3-1])
      for j in range(1,N3-1) :
          S[p+1][0][j]=S[p+1][0][j]+(Dm[0][j-1][0][j]/D[0][j])* (S0[p+1][0]-
          [j-1]-S0[p+1][0][j])+(Dm[0][j][0][j+1]/D[0][j])* (S0[p+1][0][j+1]-S0[p+1][0]-
          [j])+(Dm[0][j][1][j]/D[0][j])* (S0[p+1][1][j]-S0[p+1][0][j])
          I[p+1][0][j]=I[p+1][0][j]+(Dm[0][j-1][0][j]/D[0][j])* (I0[p+1][0]-
          [j-1]-I0[p+1][0][j])+(Dm[0][j][0][j+1]/D[0][j])* (I0[p+1][0][j+1]-I0[p+1][0]-
          [j])+(Dm[0][j][1][j]/D[0][j])* (I0[p+1][1][j]-I0[p+1][0][j])
          S0[p+1][N3-1][j-1]-S0[p+1][N3-1][j])+(Dm[N3-1][j][N3-1][j+1]/D[N3-1]-
          [j])* (S0[p+1][N3-1][j+1]-S0[p+1][N3-1][j])+(Dm[N3-2][j][N3-1][j]/D[N3-1]-
          [j])* (S0[p+1][N3-2][j]-S0[p+1][N3-1][j])
          I[p+1][N3-1][j]=I[p+1][N3-1][j]+(Dm[N3-1][j-1][N3-1][j]/D[N3-1]-
          [j])* (I0[p+1][N3-1][j-1]-I0[p+1][N3-1][j])+(Dm[N3-1][j][N3-1][j+1]/D[N3-1]-
          [j])* (I0[p+1][N3-1][j+1]-I0[p+1][N3-1][j])+(Dm[N3-2][j][N3-1][j]/D[N3-1]-
          [j])* (I0[p+1][N3-2][j]-I0[p+1][N3-1][j])
          for i in range(1,N3-1) :
              S[p+1][i][0]=S[p+1][i][0]+(Dm[i-1][0][i][0]/D[i][0])* (S0[p+1][i-1]-
              [0]-S0[p+1][i][0])+(Dm[i][0][i+1][0]/D[i][0])* (S0[p+1][i+1][0]-S0[p+1][i]-
              [0])+(Dm[i][0][i][1]/D[i][0])* (S0[p+1][i][1]-S0[p+1][0])
              I[p+1][i][0]=I[p+1][i][0]+(Dm[i-1][0][i][0]/D[i][0])* (I0[p+1][i-1]-
              [0]-I0[p+1][i][0])+(Dm[i][0][i+1][0]/D[i][0])* (I0[p+1][i+1][0]-I0[p+1][i]-
              [0])+(Dm[i][0][i][1]/D[i][0])* (I0[p+1][i][1]-I0[p+1][0])
              S[p+1][i][N3-1]=S[p+1][i][N3-1]+(Dm[i-1][N3-1][i][N3-1]/D[i]-
              [N3-1])* (S0[p+1][i-1][N3-1]-S0[p+1][i][N3-1])+(Dm[i][N3-1][i+1][N3-1]/D[i]-
              [N3-1])* (S0[p+1][i+1][N3-1]-S0[p+1][i][N3-1])+(Dm[i][N3-2][i][N3-1]/D[i]-
              [N3-1])* (S0[p+1][i][N3-2]-S0[p+1][i][N3-1])
              I[p+1][i][N3-1]=I[p+1][i][N3-1]+(Dm[i-1][N3-1][i][N3-1]/D[i]-
              [N3-1])* (I0[p+1][i-1][N3-1]-I0[p+1][i][N3-1])+(Dm[i][N3-1][i+1][N3-1]/D[i]-
              [N3-1])* (I0[p+1][i+1][N3-1]-I0[p+1][i][N3-1])+(Dm[i][N3-2][i][N3-1]/D[i]-
              [N3-1])* (I0[p+1][i][N3-2]-I0[p+1][i][N3-1])
              for i in range(1,N3-1) :
                  for j in range(1,N3-1) :
                      S[p+1][i][j]=S[p+1][i][j]+(Dm[i-1][j][i][j]/D[i][j])* (S0[p+1]-
                      [i-1][j]-S0[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])* (S0[p+1][i+1][j]-S0[p+1][i]-
                      [j])+(Dm[i][j-1][i][j]/D[i][j])* (S0[p+1][i][j-1]-S0[p+1][i][j])+(Dm[i][j][i]-
                      [j+1]/D[i][j])* (S0[p+1][i][j+1]-S0[p+1][i][j])
                      I[p+1][i][j]=I[p+1][i][j]+(Dm[i-1][j][i][j]/D[i][j])* (I0[p+1]-
                      [i-1][j]-I0[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])* (I0[p+1][i+1][j]-I0[p+1][i]-
                      [j])+(Dm[i][j-1][i][j]/D[i][j])* (I0[p+1][i][j-1]-I0[p+1][i][j])+(Dm[i][j][i]-
                      [j+1]/D[i][j])* (I0[p+1][i][j+1]-I0[p+1][i][j])
                      for i in range(N3) :
                          for j in range(N3) :
                              ki=Ki[i][j]
                              li=Li[i][j]
                              d=D[i][j]/1000000

```

```

631     sl=-ki*d*s*Si[p][i][j]*Ii[p][i][j]
632     il=ki*d*s*Si[p][i][j]*Ii[p][i][j]-li*Ii[p][i][j]
633     S2=Si[p][i][j]+(h/2)*sl
634     I2=Ii[p][i][j]+(h/2)*il
635     s2=-ki*d*s*S2*I2
636     i2=ki*d*s*S2*I2-li*I2
637     S3=Si[p][i][j]+(h/2)*s2
638     I3=Ii[p][i][j]+(h/2)*i2
639     s3=-ki*d*s*S3*I3
640     i3=ki*d*s*S3*I3-li*I3
641     S4=Si[p][i][j]+h*s3
642     I4=Ii[p][i][j]+h*i3
643     s4=-ki*d*s*S4*I4
644     i4=ki*d*s*S4*I4-li*I4
645     S0i[p+1][i][j]=Si[p][i][j]+(h/6)*(sl+2*s2+2*s3+s4)
646     I0i[p+1][i][j]=Ii[p][i][j]+(h/6)*(il+2*i2+2*i3+i4)
647     Si[p+1][i][j]=S0i[p+1][i][j]
648     Ii[p+1][i][j]=I0i[p+1][i][j]
649     Si[p][i][j]=D[i][j]*Si[p][i][j]
650     Ii[p][i][j]=D[i][j]*Ii[p][i][j]
651     Si[p+1][0][0]=Si[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])*(S0i[p+1][1][0]-
652     S0i[p+1][0][0])+ (Dm[0][0][0][1]/D[0][0])*(S0i[p+1][0][1]-S0i[p+1][0][0])
653     Ii[p+1][0][0]=Ii[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])*(I0i[p+1][1][0]-
654     I0i[p+1][0][0])+ (Dm[0][0][0][1]/D[0][0])*(I0i[p+1][0][1]-I0i[p+1][0][0])
655     Si[p+1][0][N3-1]=Si[p+1][0][N3-1]+(Dm[0][0][N3-1]/D[0][0])*(S0i[p+1][1][N3-1]-
656     S0i[p+1][0][N3-1])*(S0i[p+1][1][N3-1]-S0i[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0][
657     N3-1])*(S0i[p+1][0][N3-2]-S0i[p+1][0][N3-1])
658     Ii[p+1][0][N3-1]=Ii[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0][N3-1])
659     for j in range(1,N3-1) :
660         Si[p+1][0][j]=Si[p+1][0][j]+(Dm[0][j-1][0][j]/D[0][j])*(S0i[p+1][0][
661     [j-1]-S0i[p+1][0][j])+ (Dm[0][j][0][j+1]/D[0][j])*(S0i[p+1][0][j+1]-S0i[p+1][0][j])
662     Ii[p+1][0][j]=Ii[p+1][0][j]+(Dm[0][j-1][0][j]/D[0][j])*(I0i[p+1][0][j+1]-I0i[p+1][0][j])
663     Si[p+1][N3-1][j]=Si[p+1][N3-1][j]+(Dm[N3-1][j-1][N3-1][j]/D[N3-1][j])
664     for i in range(1,N3-1) :
665         Si[p+1][i][0]=Si[p+1][i][0]+(Dm[i-1][0][i]/D[i][0])*(S0i[p+1][
666     [i-1][0]-S0i[p+1][i][0])+ (Dm[i][0][i+1][0]/D[i][0])*(S0i[p+1][i+1][0]-S0i[p+1][i][0])
667     Ii[p+1][i][0]=Ii[p+1][i][0]+(Dm[i-1][0][i]/D[i][0])*(I0i[p+1][i+1][0]-I0i[p+1][i][0])
668     Si[p+1][i][N3-1]=Si[p+1][i][N3-1]+(Dm[i-1][N3-1][i][N3-1]/D[i][N3-1])
669     for i in range(1,N3-1) :

```

```

670     for j in range(1,N3-1) :
671         Si[p+1][i][j]=Si[p+1][i][j]+(Dm[i-1][j][i][j]/D[i]
672         [j])*(S0i[p+1][i-1][j]-S0i[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])*(S0i[p+1]
673         [i+1][j]-S0i[p+1][i][j])+(Dm[i][j-1][i][j]/D[i][j])*(S0i[p+1][i][j-1]-S0i[p+1]
674         [i][j])+(Dm[i][j][j+1]/D[i][j])*(S0i[p+1][i][j+1]-S0i[p+1][i][j])
675         Ii[p+1][i][j]=Ii[p+1][i][j]+(Dm[i-1][j][i][j]/D[i]
676         [j])*(I0i[p+1][i-1][j]-I0i[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])*(I0i[p+1]
677         [i+1][j]-I0i[p+1][i][j])+(Dm[i][j-1][i][j]/D[i][j])*(I0i[p+1][i][j-1]-I0i[p+1]
678         [i][j])+(Dm[i][j][j+1]/D[i][j])*(I0i[p+1][i][j+1]-I0i[p+1][i][j])
679         p=p+1
680     for i in range(N3) :
681         for j in range(N3) :
682             S[N][i][j]=D[i][j]*S[N][i][j]
683             I[N][i][j]=D[i][j]*I[N][i][j]
684             Si[N][i][j]=D[i][j]*Si[N][i][j]
685             Ii[N][i][j]=D[i][j]*Ii[N][i][j]
686     n=len(T)
687     for q in range(n) :
688         if T[q]<=t :
689             p=int(T[q]/h)
690             espace_sub(I,X,Y,p,h,N3,zlim)
691         else :
692             p=int(T[q]/h)
693             espace_sub_2(I,Ii,X,Y,p,h,N3,zlim)
694
695 def espace_sub_2(I,Ii,X,Y,p,h,N3,zlim) :
696     Z=[0]*(N3**2)
697     Zi=[0]*(N3**2)
698     for i in range(N3) :
699         for j in range(N3) :
700             Z[N3*i+j]=I[p][i][j]
701             Zi[N3*i+j]=Ii[p][i][j]
702     fig=figure()
703     ax=fig.gca(projection='3d')
704     ax.set_zlim3d(0,zlim)
705     ax.set_title('t='+(str(int(p*h*100)/100))+' ut')
706     ax.set_xlabel('x')
707     ax.set_ylabel('y')
708     ax.text(N3*0.95,N3*0.75,zlim*1.2,'d(I) (hab/km2)',color=(0,0,0),size=12)
709     ax.plot(X,Y,Zi,'o',color=(0,0.75,0))
710     ax.plot(X,Y,Z,'o',color=(0,0,0.75))
711
712 def espace_uni_impact(k,l,s,d,P,dm,T,N,x,N3,t,lambda,mu,Q,zlim) :
713     d=d/1000000
714     dm=dm/1000000
715     Id=[[0 for i in range(N3)] for j in range(N3)]
716     a=int((N3-1)/2)
717     Id[a][a]=(N3**2)/P
718     u=linspace(0,N3-1,N3)
719     h=x/N
720     X=[0]*(N3**2)
721     Y=[0]*(N3**2)
722     S=[[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
723     I=[Id]+[[[0 for i in range(N3)] for j in range(N3)] for i in range(N)]
724     S0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
725     I0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
726     Si=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
727     Ii=[[Id]+[[0 for i in range(N3)] for j in range(N3)] for i in range(N)]]
728     S0i=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
729     I0i=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
730     for i in range(N3) :
731         for j in range(N3) :
732             X[N3*i+j]=u[i]
733             Y[N3*i+j]=u[j]
734             S[0][i][j]=1-I[0][i][j]
735             Si[0][i][j]=1-I[0][i][j]
736
737     p=0
738     while p<t/h :
739         for i in range(N3) :
740             for j in range(N3) :
741                 sl=-k*d*s*S[p][i][j]*I[p][i][j]
742                 il=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
743                 S2=S[p][i][j]+(h/2)*sl
744                 I2=I[p][i][j]+(h/2)*il

```

```

738     s2=-k*d*s*S2*I2
739     i2=k*d*s*S2*I2-l*I2
740     S3=S[p][i][j]+(h/2)*s2
741     I3=I[p][i][j]+(h/2)*i2
742     s3=-k*d*s*S3*I3
743     i3=k*d*s*S3*I3-l*I3
744     S4=S[p][i][j]+h*s3
745     I4=I[p][i][j]+h*i3
746     s4=-k*d*s*S4*I4
747     i4=k*d*s*S4*I4-l*I4
748     S0[p+1][i][j]=S[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
749     I0[p+1][i][j]=I[p][i][j]+(h/6)*(i1+2*i2+2*i3+i4)
750     S[p+1][i][j]=S0[p+1][i][j]
751     I[p+1][i][j]=I0[p+1][i][j]
752     S[p][i][j]=d*S[p][i][j]*1000000
753     I[p][i][j]=d*I[p][i][j]*1000000
754     S[p+1][0][0]=S[p+1][0][0]+(dm/d)*(S0[p+1][1][0]-S0[p+1][0]
755     [0])+(dm/d)*(S0[p+1][0][1]-S0[p+1][0][0])
756     I[p+1][0][0]=I[p+1][0][0]+(dm/d)*(I0[p+1][1][0]-I0[p+1][0]
757     [0])+(dm/d)*(I0[p+1][0][1]-I0[p+1][0][0])
758     S[p+1][0][N3-1]=S[p+1][0][N3-1]+(dm/d)*(S0[p+1][1][N3-1]-S0[p+1][0]
759     [N3-1])+(dm/d)*(S0[p+1][0][N3-2]-S0[p+1][0][N3-1])
760     I[p+1][0][N3-1]=I[p+1][0][N3-1]+(dm/d)*(I0[p+1][1][N3-1]-I0[p+1][0]
761     [N3-1])+(dm/d)*(I0[p+1][0][N3-2]-I0[p+1][0][N3-1])
762     S[p+1][N3-1][0]=S[p+1][N3-1][0]+(dm/d)*(S0[p+1][N3-1][1]-S0[p+1][N3-1]
763     [0])+(dm/d)*(S0[p+1][N3-2][0]-S0[p+1][N3-1][0])
764     I[p+1][N3-1][0]=I[p+1][N3-1][0]+(dm/d)*(I0[p+1][N3-1][1]-I0[p+1][N3-1]
765     [0])+(dm/d)*(I0[p+1][N3-2][0]-I0[p+1][N3-1][0])
766     S[p+1][N3-1][N3-1]=S[p+1][N3-1][N3-1]+(dm/d)*(I0[p+1][N3-1][N3-2]-I0[p+1]
767     [N3-1][N3-1])+(dm/d)*(I0[p+1][N3-2][N3-1]-S0[p+1][N3-1][N3-1])
768     I[p+1][N3-1][N3-1]=I[p+1][N3-1][N3-1]+(dm/d)*(I0[p+1][N3-1][N3-2]-I0[p+1]
769     [N3-1][N3-1])+(dm/d)*(I0[p+1][N3-2][N3-1]-I0[p+1][N3-1][N3-1])
770     S[p+1][i][0]=S[p+1][i][0]+(dm/d)*(S0[p+1][i-1][0]-S0[p+1][i]
771     [0])+(dm/d)*(S0[p+1][i+1][0]-S0[p+1][i][0])+(dm/d)*(S0[p+1][i][1]-S0[p+1][i]
772     [0])
773     I[p+1][i][0]=I[p+1][i][0]+(dm/d)*(I0[p+1][i-1][0]-I0[p+1][i]
774     [0])+(dm/d)*(I0[p+1][i+1][0]-I0[p+1][i][0])+(dm/d)*(I0[p+1][i][1]-I0[p+1][i]
775     [0])
776     S[p+1][i][N3-1]=S[p+1][i][N3-1]+(dm/d)*(S0[p+1][i-1][N3-1]-S0[p+1]
777     [i][N3-1])+(dm/d)*(S0[p+1][i+1][N3-1]-S0[p+1][i][N3-1])+(dm/d)*(S0[p+1][i]
778     [N3-2]-S0[p+1][i][N3-1])
779     I[p+1][i][N3-1]=I[p+1][i][N3-1]+(dm/d)*(I0[p+1][i-1][N3-1]-I0[p+1]
780     [i][N3-1])+(dm/d)*(I0[p+1][i+1][N3-1]-I0[p+1][i][N3-1])+(dm/d)*(I0[p+1][i]
781     [N3-2]-I0[p+1][i][N3-1])
782     for i in range(1,N3-1) :
783         for j in range(1,N3-1) :
784             S[p+1][i][j]=S[p+1][i][j]+(dm/d)*(S0[p+1][i-1][j]-S0[p+1][i]
785             [j])+(dm/d)*(S0[p+1][i+1][j]-S0[p+1][i][j])+(dm/d)*(S0[p+1][i][j-1]-S0[p+1][i]
786             [j])+(dm/d)*(S0[p+1][i][j+1]-S0[p+1][i][j])
787             I[p+1][i][j]=I[p+1][i][j]+(dm/d)*(I0[p+1][i-1][j]-I0[p+1][i]
788             [j])+(dm/d)*(I0[p+1][i+1][j]-I0[p+1][i][j])+(dm/d)*(I0[p+1][i][j-1]-I0[p+1][i]
789             [j])+(dm/d)*(I0[p+1][i][j+1]-I0[p+1][i][j])
790             for i in range(N3) :
791                 for j in range(N3) :
792                     S0i[p+1][i][j]=S0[p+1][i][j]
793                     I0i[p+1][i][j]=I0[p+1][i][j]
794                     Si[p+1][i][j]=S[p+1][i][j]
795                     Ii[p+1][i][j]=I[p+1][i][j]
796                     Si[p][i][j]=S[p][i][j]

```

```

783     Ii[p][i][j]=I[p][i][j]
784     p=p+1
785     Ki=[[0 for i in range(N3)] for j in range(N3)]
786     Li=[[0 for i in range(N3)] for j in range(N3)]
787     for i in range(N3) :
788         for j in range(N3) :
789             if Q[i][j]==1 :
790                 Ki[i][j]=k*(1-λ)
791                 Li[i][j]=l/(1-μ)
792             else :
793                 Ki[i][j]=k
794                 Li[i][j]=l
795     while p<N :
796         for i in range(N3) :
797             for j in range(N3) :
798                 s1=-k*d*s*S[p][i][j]*I[p][i][j]
799                 il=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
800                 S2=S[p][i][j]+(h/2)*s1
801                 I2=I[p][i][j]+(h/2)*il
802                 s2=-k*d*s*S2*I2
803                 i2=k*d*s*S2*I2-l*I2
804                 S3=S[p][i][j]+(h/2)*s2
805                 I3=I[p][i][j]+(h/2)*i2
806                 s3=-k*d*s*S3*I3
807                 i3=k*d*s*S3*I3-l*I3
808                 S4=S[p][i][j]+h*s3
809                 I4=I[p][i][j]+h*i3
810                 s4=-k*d*s*S4*I4
811                 i4=k*d*s*S4*I4-l*I4
812                 S0[p+1][i][j]=S[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
813                 I0[p+1][i][j]=I[p][i][j]+(h/6)*(il+2*i2+2*i3+i4)
814                 S[p+1][i][j]=S0[p+1][i][j]
815                 I[p+1][i][j]=I0[p+1][i][j]
816                 S[p][i][j]=d*S[p][i][j]*1000000
817                 I[p][i][j]=d*I[p][i][j]*1000000
818     S[p+1][0][0]=S[p+1][0][0]+(dm/d)*(S0[p+1][1][0]-S0[p+1][0]
819 [0])+ (dm/d)*(I0[p+1][0][1]-I0[p+1][0][0])
820     S[p+1][0][N3-1]=S[p+1][0][N3-1]+(dm/d)*(S0[p+1][1][N3-1]-S0[p+1][0]
821 [N3-1])+ (dm/d)*(S0[p+1][0][N3-2]-S0[p+1][0][N3-1])
822     I[p+1][0][N3-1]=I[p+1][0][N3-1]+(dm/d)*(I0[p+1][1][N3-1]-I0[p+1][0]
823 [N3-1])+ (dm/d)*(I0[p+1][0][N3-2]-I0[p+1][0][N3-1])
824     S[p+1][N3-1][0]=S[p+1][N3-1][0]+(dm/d)*(S0[p+1][N3-1][1]-S0[p+1][N3-1]
825 [0])+ (dm/d)*(S0[p+1][N3-2][0]-S0[p+1][N3-1][0])
826     I[p+1][N3-1][0]=I[p+1][N3-1][0]+(dm/d)*(I0[p+1][N3-1][1]-I0[p+1][N3-1]
827 [0])+ (dm/d)*(I0[p+1][N3-2][0]-I0[p+1][N3-1][0])
828     S[p+1][N3-1][N3-1]=S[p+1][N3-1][N3-1]+(dm/d)*(S0[p+1][N3-1][N3-2]-
829     S0[p+1][N3-1][N3-1])+ (dm/d)*(S0[p+1][N3-2][N3-1]-S0[p+1][N3-1][N3-1])
830     I[p+1][N3-1][N3-1]=I[p+1][N3-1][N3-1]+(dm/d)*(I0[p+1][N3-1][N3-2]-
831     I0[p+1][N3-1][N3-1])+ (dm/d)*(I0[p+1][N3-2][N3-1]-I0[p+1][N3-1][N3-1])
832     for j in range(1,N3-1) :
833         S[p+1][0][j]=S[p+1][0][j]+(dm/d)*(S0[p+1][0][j-1]-S0[p+1][0]
834 [j])+ (dm/d)*(S0[p+1][0][j+1]-S0[p+1][0][j])+ (dm/d)*(S0[p+1][1][j]-S0[p+1][0]
835 [j])
836         I[p+1][0][j]=I[p+1][0][j]+(dm/d)*(I0[p+1][0][j-1]-I0[p+1][0]
837 [j])+ (dm/d)*(I0[p+1][0][j+1]-I0[p+1][0][j])+ (dm/d)*(I0[p+1][1][j]-I0[p+1][0]
838 [j])
839         S[p+1][N3-1][j]=S[p+1][N3-1][j]+(dm/d)*(S0[p+1][N3-1][j-1]-S0[p+1]
840 [N3-1][j])+ (dm/d)*(S0[p+1][N3-1][j+1]-S0[p+1][N3-1][j])+ (dm/d)*(S0[p+1][N3-2]
841 [j]-S0[p+1][N3-1][j])
842         I[p+1][N3-1][j]=I[p+1][N3-1][j]+(dm/d)*(I0[p+1][N3-1][j-1]-I0[p+1]
843 [N3-1][j])+ (dm/d)*(I0[p+1][N3-1][j+1]-I0[p+1][N3-1][j])+ (dm/d)*(I0[p+1][N3-2]
844 [j]-I0[p+1][N3-1][j])
845         for i in range(1,N3-1) :
846             S[p+1][i][0]=S[p+1][i][0]+(dm/d)*(S0[p+1][i-1][0]-S0[p+1][i]
847 [0])+ (dm/d)*(S0[p+1][i+1][0]-S0[p+1][i][0])+ (dm/d)*(S0[p+1][i][1]-S0[p+1][i]
848 [0])
849             I[p+1][i][0]=I[p+1][i][0]+(dm/d)*(I0[p+1][i-1][0]-I0[p+1][i]
850 [0])+ (dm/d)*(I0[p+1][i+1][0]-I0[p+1][i][0])+ (dm/d)*(I0[p+1][i][1]-I0[p+1][i]
851 [0])
852             S[p+1][i][N3-1]=S[p+1][i][N3-1]+(dm/d)*(S0[p+1][i-1][N3-1]-S0[p+1]
853 [i][N3-1])+ (dm/d)*(S0[p+1][i+1][N3-1]-S0[p+1][i][N3-1])+ (dm/d)*(S0[p+1][i]
854 [N3-2]-S0[p+1][i][N3-1])

```

```

835     I[p+1][i][N3-1]=I[p+1][i][N3-1]+(dm/d)*(I0[p+1][i-1][N3-1]-I0[p+1]
836     [i][N3-1])+(dm/d)*(I0[p+1][i+1][N3-1]-I0[p+1][i][N3-1])+(dm/d)*(I0[p+1][i]
837     [N3-2]-I0[p+1][i][N3-1])
838     for i in range(1,N3-1) :
839         for j in range(1,N3-1) :
840             S[p+1][i][j]=S[p+1][i][j]+(dm/d)*(S0[p+1][i-1][j]-S0[p+1][i]
841             [j])+(dm/d)*(S0[p+1][i+1][j]-S0[p+1][i][j])+(dm/d)*(S0[p+1][i][j-1]-S0[p+1][i]
842             [j])+(dm/d)*(S0[p+1][i][j+1]-S0[p+1][i][j])
843             I[p+1][i][j]=I[p+1][i][j]+(dm/d)*(I0[p+1][i-1][j]-I0[p+1][i]
844             [j])+(dm/d)*(I0[p+1][i+1][j]-I0[p+1][i][j])+(dm/d)*(I0[p+1][i][j-1]-I0[p+1][i]
845             [j])+(dm/d)*(I0[p+1][i][j+1]-I0[p+1][i][j])
846             for i in range(N3) :
847                 for j in range(N3) :
848                     ki=Ki[i][j]
849                     li=Li[i][j]
850                     s1=-ki*d*s*Si[p][i][j]*Ii[p][i][j]
851                     il=ki*d*s*Si[p][i][j]*Ii[p][i][j]-li*Ii[p][i][j]
852                     S2=Si[p][i][j]+(h/2)*s1
853                     I2=Ii[p][i][j]+(h/2)*il
854                     s2=-ki*d*s*S2*I2
855                     i2=ki*d*s*S2*I2-li*I2
856                     S3=Si[p][i][j]+(h/2)*s2
857                     I3=Ii[p][i][j]+(h/2)*i2
858                     s3=-ki*d*s*S3*I3
859                     i3=ki*d*s*S3*I3-li*I3
860                     S4=Si[p][i][j]+h*s3
861                     I4=Ii[p][i][j]+h*i3
862                     s4=-ki*d*s*S4*I4
863                     i4=ki*d*s*S4*I4-li*I4
864                     S0i[p+1][i][j]=Si[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
865                     I0i[p+1][i][j]=Ii[p][i][j]+(h/6)*(il+2*i2+2*i3+i4)
866                     Si[p+1][i][j]=S0i[p+1][i][j]
867                     Ii[p+1][i][j]=I0i[p+1][i][j]
868                     Si[p][i][j]=d*Si[p][i][j]*1000000
869                     Ii[p][i][j]=d*Ii[p][i][j]*1000000
870                     Si[p+1][0][0]=Si[p+1][0][0]+(dm/d)*(S0i[p+1][1][0]-S0i[p+1][0]
871                     [0])+(dm/d)*(S0i[p+1][0][1]-S0i[p+1][0][0])
872                     Ii[p+1][0][0]=Ii[p+1][0][0]+(dm/d)*(I0i[p+1][1][0]-I0i[p+1][0]
873                     [0])+(dm/d)*(I0i[p+1][0][1]-I0i[p+1][0][0])
874                     Si[p+1][0][N3-1]=Si[p+1][0][N3-1]+(dm/d)*(S0i[p+1][1][N3-1]-S0i[p+1][0]
875                     [N3-1])+(dm/d)*(S0i[p+1][0][N3-2]-S0i[p+1][0][N3-1])
876                     Ii[p+1][0][N3-1]=Ii[p+1][0][N3-1]+(dm/d)*(I0i[p+1][1][N3-1]-I0i[p+1][0]
877                     [N3-1])+(dm/d)*(I0i[p+1][0][N3-2]-I0i[p+1][0][N3-1])
878                     Si[p+1][0][N3-1]=Si[p+1][0][N3-1]+(dm/d)*(S0i[p+1][N3-1][1]-S0i[p+1][N3-1][0])
879                     Ii[p+1][0][N3-1]=Ii[p+1][0][N3-1]+(dm/d)*(I0i[p+1][N3-1][1]-I0i[p+1][N3-1][0])
880                     for j in range(1,N3-1) :
881                         Si[p+1][0][j]=Si[p+1][0][j]+(dm/d)*(S0i[p+1][0][j-1]-S0i[p+1][0]
882                         [j])+(dm/d)*(S0i[p+1][0][j+1]-S0i[p+1][0][j])+(dm/d)*(S0i[p+1][1][j]-S0i[p+1]
883                         [0][j])
884                         Ii[p+1][0][j]=Ii[p+1][0][j]+(dm/d)*(I0i[p+1][0][j-1]-I0i[p+1][0]
885                         [j])+(dm/d)*(I0i[p+1][0][j+1]-I0i[p+1][0][j])+(dm/d)*(I0i[p+1][1][j]-I0i[p+1]
886                         [0][j])
887                         Si[p+1][N3-1][j]=Si[p+1][N3-1][j]+(dm/d)*(S0i[p+1][N3-1][j-1]-
888                         S0i[p+1][N3-1][j])+(dm/d)*(S0i[p+1][N3-1][j+1]-S0i[p+1][N3-1][j])
889                         Ii[p+1][N3-1][j]=Ii[p+1][N3-1][j]+(dm/d)*(I0i[p+1][N3-1][j-1]-I0i[p+1][N3-1]
890                         [j])+(dm/d)*(I0i[p+1][N3-1][j+1]-I0i[p+1][N3-1][j])
891                         for i in range(1,N3-1) :
892                             Si[p+1][i][0]=Si[p+1][i][0]+(dm/d)*(S0i[p+1][i-1][0]-S0i[p+1][i]
893                             [0])+(dm/d)*(S0i[p+1][i+1][0]-S0i[p+1][i][0])+(dm/d)*(S0i[p+1][i][1]-S0i[p+1]
894                             [i][0])
895                             Ii[p+1][i][0]=Ii[p+1][i][0]+(dm/d)*(I0i[p+1][i-1][0]-I0i[p+1][i]
896                             [0])+(dm/d)*(I0i[p+1][i+1][0]-I0i[p+1][i][0])+(dm/d)*(I0i[p+1][i][1]-I0i[p+1]
897                             [i][0])
898                             Si[p+1][i][N3-1]=Si[p+1][i][N3-1]+(dm/d)*(S0i[p+1][i-1][N3-1]-
899                             S0i[p+1][i][N3-1])+(dm/d)*(S0i[p+1][i+1][N3-1]-S0i[p+1][i][N3-1])
900                             Ii[p+1][i][N3-1]=Ii[p+1][i][N3-1]+(dm/d)*(I0i[p+1][i-1][N3-1]-I0i[p+1][i][N3-1])

```

```

881 [N3-1])+(dm/d)*(I0i[p+1][i][N3-2]-I0i[p+1][i][N3-1])
882     for i in range(1,N3-1) :
883         for j in range(1,N3-1) :
884             Si[p+1][i][j]=Si[p+1][i][j]+(dm/d)*(S0i[p+1][i-1][j]-S0i[p+1]
885 [i][j])+(dm/d)*(S0i[p+1][i+1][j]-S0i[p+1][i][j])+(dm/d)*(S0i[p+1][i][j-1]-
886 S0i[p+1][i][j])+(dm/d)*(S0i[p+1][i][j+1]-S0i[p+1][i][j])
887             Ii[p+1][i][j]=Ii[p+1][i][j]+(dm/d)*(I0i[p+1][i-1][j]-I0i[p+1]
888 [i][j])+(dm/d)*(I0i[p+1][i+1][j]-I0i[p+1][i][j])+(dm/d)*(I0i[p+1][i][j-1]-
889 I0i[p+1][i][j])+(dm/d)*(I0i[p+1][i][j+1]-I0i[p+1][i][j])
890             p=p+1
891     for i in range(N3) :
892         for j in range(N3) :
893             S[N][i][j]=d*S[N][i][j]*1000000
894             I[N][i][j]=d*I[N][i][j]*1000000
895             Si[N][i][j]=d*Si[N][i][j]*1000000
896             Ii[N][i][j]=d*Ii[N][i][j]*1000000
897 n=len(T)
898 for q in range(n) :
899     if T[q]<=t :
900         p=int(T[q]/h)
901         espace_sub(I,X,Y,p,h,N3,zlim)
902     else :
903         p=int(T[q]/h)
904         espace_sub_2(I,Ii,X,Y,p,h,N3,zlim)
905
906 def q(N3,p,r0) :
907     Qres=[[0 for i in range(N3)] for j in range(N3)]
908     N4=int(p*N3**2)
909     R=[[0 for i in range(N3**2)]]
910     a=int((N3-1)/2)
911     for i in range(N3) :
912         for j in range(N3) :
913             R[i*N3+j]=[sqrt((i-a)**2+(j-a)**2),i,j]
914     R2=tri(R)
915     n=0
916     while R2[n][0]<r0 :
917         n=n+1
918     if n>=N3**2-N4+1 :
919         return(False)
920     else :
921         for m in range(n,n+N4) :
922             i=R2[m][1]
923             j=R2[m][2]
924             Qres[i][j]=1
925     return(Qres)
926
927 def espace_adj_e(K,L,s,D,Id,Dm,N,x,N3,t,λ,μ,Q,zlim) :
928     if Q==False :
929         return(0)
930     u=linspace(0,N3-1,N3)
931     h=x/N
932     X=[0]*(N3**2)
933     Y=[0]*(N3**2)
934     S=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
935     I=[Id]+[[0 for i in range(N3)] for j in range(N3)] for i in range(N)]
936     S0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
937     I0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
938     Si=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
939     Ii=[[0 for i in range(N3)] for j in range(N3)] for i in range(N)]
940     S0i=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
941     I0i=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
942     for i in range(N3) :
943         for j in range(N3) :
944             X[N3*i+j]=u[i]
945             Y[N3*i+j]=u[j]
946             S[0][i][j]=1-I[0][i][j]
947             Si[0][i][j]=1-I[0][i][j]
948             p=0
949             while p<t/h :
950                 for i in range(N3) :
951                     for j in range(N3) :
952                         k=K[i][j]
953                         l=L[i][j]
954                         d=D[i][j]/1000000

```

```

951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
999

```

```

990     for j in range(1,N3-1) :
991         S[p+1][i][j]=S[p+1][i][j]+(Dm[i-1][j][i][j]/D[i][j])*(S0[p+1]
992 [i-1][j]-S0[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])*(S0[p+1][i+1][j]-S0[p+1][i]
993 [j])+(Dm[i][j-1][i][j]/D[i][j])*(S0[p+1][i][j-1]-S0[p+1][i][j])+(Dm[i][j][i]
994 [j+1]/D[i][j])*(S0[p+1][i][j+1]-S0[p+1][i][j])
995         I[p+1][i][j]=I[p+1][i][j]+(Dm[i-1][j][i][j]/D[i][j])*(I0[p+1]
996 [i-1][j]-I0[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])*(I0[p+1][i+1][j]-I0[p+1][i]
997 [j])+(Dm[i][j-1][i][j]/D[i][j])*(I0[p+1][i][j-1]-I0[p+1][i][j])+(Dm[i][j][i]
998 [j+1]/D[i][j])*(I0[p+1][i][j+1]-I0[p+1][i][j])
999     for i in range(N3) :
1000         for j in range(N3) :
1001             S0i[p+1][i][j]=S0[p+1][i][j]
1002             I0i[p+1][i][j]=I0[p+1][i][j]
1003             Si[p+1][i][j]=S[p+1][i][j]
1004             Ii[p+1][i][j]=I[p+1][i][j]
1005             Sip[i][j]=S[p][i][j]
1006             Iip[i][j]=I[p][i][j]
1007             p=p+1
1008             Ki=[[0 for i in range(N3)] for j in range(N3)]
1009             Li=[[0 for i in range(N3)] for j in range(N3)]
1010             for i in range(N3) :
1011                 for j in range(N3) :
1012                     if Q[i][j]==1 :
1013                         Ki[i][j]=K[i][j]*(1-λ)
1014                         Li[i][j]=L[i][j]/(1-μ)
1015                     else :
1016                         Ki[i][j]=K[i][j]
1017                         Li[i][j]=L[i][j]
1018             while p<N :
1019                 for i in range(N3) :
1020                     for j in range(N3) :
1021                         k=K[i][j]
1022                         l=L[i][j]
1023                         d=D[i][j]/1000000
1024                         s1=-k*d*s*S[p][i][j]*I[p][i][j]
1025                         i1=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
1026                         S2=S[p][i][j]+(h/2)*s1
1027                         I2=I[p][i][j]+(h/2)*i1
1028                         s2=-k*d*s*S2*I2-l*I2
1029                         i2=k*d*s*S2*I2-l*I2
1030                         S3=S[p][i][j]+(h/2)*s2
1031                         I3=I[p][i][j]+(h/2)*i2
1032                         s3=-k*d*s*S3*I3-l*I3
1033                         i3=k*d*s*S3*I3-l*I3
1034                         S4=S[p][i][j]+h*s3
1035                         I4=I[p][i][j]+h*i3
1036                         s4=-k*d*s*S4*I4-l*I4
1037                         i4=k*d*s*S4*I4-l*I4
1038                         S0[p+1][i][j]=S[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
1039                         I0[p+1][i][j]=I[p][i][j]+(h/6)*(i1+2*i2+2*i3+i4)
1040                         S[p+1][i][j]=S0[p+1][i][j]
1041                         I[p+1][i][j]=I0[p+1][i][j]
1042                         S[p][i][j]=D[i][j]*S[p][i][j]
1043                         I[p][i][j]=D[i][j]*I[p][i][j]
1044                         S[p+1][0][0]=S[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])*(S0[p+1][1][0]-
S0[p+1][0][0])+(Dm[0][0][0][1]/D[0][0])*(S0[p+1][0][1]-S0[p+1][0][0])
I[p+1][0][0]=I[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])*(I0[p+1][1][0]-
I0[p+1][0][0])+(Dm[0][0][0][1]/D[0][0])*(I0[p+1][0][1]-I0[p+1][0][0])
S[p+1][0][N3-1]=S[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0]
[N3-1])*((S0[p+1][1][N3-1]-S0[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0]
[N3-1]))*(S0[p+1][0][N3-2]-S0[p+1][0][N3-1])
I[p+1][0][N3-1]=I[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0]
[N3-1])*((I0[p+1][1][N3-1]-I0[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0]
[N3-1]))*(I0[p+1][0][N3-2]-I0[p+1][0][N3-1])
S[p+1][N3-1][0]=S[p+1][N3-1][0]+(Dm[N3-1][0][N3-1][1]/D[N3-1]
[0])*((S0[p+1][N3-1][1]-S0[p+1][N3-1][0])+(Dm[N3-2][0][N3-1][0]/D[N3-1]
[0]))*(S0[p+1][N3-2][0]-S0[p+1][N3-1][0])
I[p+1][N3-1][0]=I[p+1][N3-1][0]+(Dm[N3-1][0][N3-1][1]/D[N3-1]
[0])*((I0[p+1][N3-1][1]-I0[p+1][N3-1][0])+(Dm[N3-2][0][N3-1][0]/D[N3-1]
[0]))*(I0[p+1][N3-2][0]-I0[p+1][N3-1][0])
S[p+1][N3-1][N3-1]=S[p+1][N3-1][N3-1]+(Dm[N3-1][N3-2][N3-1]
[N3-1]/D[N3-1][N3-1])*((S0[p+1][N3-1][N3-2]-S0[p+1][N3-1][N3-1])+(Dm[N3-2][N3-1]
[N3-1]/D[N3-1][N3-1]))*(S0[p+1][N3-2][N3-1]-S0[p+1][N3-1][N3-1])

```

```

1045     I[p+1][N3-1][N3-1]=I[p+1][N3-1][N3-1]+(Dm[N3-1][N3-2][N3-1]
1046     [N3-1]/D[N3-1][N3-1])* (I0[p+1][N3-1][N3-2]-I0[p+1][N3-1][N3-1])+(Dm[N3-2][N3-1]
1047     [N3-1]/D[N3-1][N3-1])* (I0[p+1][N3-2]-I0[p+1][N3-1])
1048     for j in range(1,N3-1) :
1049         S[p+1][0][j]=S[p+1][0][j]+(Dm[0][j-1][0][j]/D[0][j])* (S0[p+1][0]
1050         [j-1]-S0[p+1][0][j])+(Dm[0][j][0][j+1]/D[0][j])* (S0[p+1][0][j+1]-S0[p+1][0]
1051         [j])+(Dm[0][j][1][j]/D[0][j])* (S0[p+1][1][j]-S0[p+1][0][j])
1052         I[p+1][0][j]=I[p+1][0][j]+(Dm[0][j-1][0][j]/D[0][j])* (I0[p+1][0]
1053         [j-1]-I0[p+1][0][j])+(Dm[0][j][0][j+1]/D[0][j])* (I0[p+1][0][j+1]-I0[p+1][0]
1054         [j])+(Dm[0][j][1][j]/D[0][j])* (I0[p+1][1][j]-I0[p+1][0][j])
1055         S[p+1][N3-1][j]=S[p+1][N3-1][j]+(Dm[N3-1][j-1][N3-1]/D[N3-1]
1056         [j])* (S0[p+1][N3-1][j-1]-S0[p+1][N3-1][j])+(Dm[N3-1][j][N3-1][j+1]/D[N3-1]
1057         [j])* (S0[p+1][N3-2][j]-S0[p+1][N3-1][j])
1058         I[p+1][N3-1][j]=I[p+1][N3-1][j]+(Dm[N3-1][j-1][N3-1]/D[N3-1]
1059         [j])* (I0[p+1][N3-1]-I0[p+1][N3-1][j])+(Dm[N3-1][j][N3-1][j+1]/D[N3-1]
1060         [j])* (I0[p+1][N3-2]-I0[p+1][N3-1][j])
1061         for i in range(1,N3-1) :
1062             for j in range(1,N3-1) :
1063                 i1=S0[p+1][i][j]+(Dm[i][j][i+1][j]/D[i][j])* (S0[p+1][i+1][j]-S0[p+1][i]
1064                 [j])+(Dm[i][j-1][i][j]/D[i][j])* (S0[p+1][i][j-1]-S0[p+1][i][j])+(Dm[i][j][i]
1065                 [j+1]/D[i][j])* (S0[p+1][i][j+1]-S0[p+1][i][j])
1066                 I[p+1][i][j]=I[p+1][i][j]+(Dm[i-1][j][i][j]/D[i][j])* (I0[p+1]
1067                 [i-1][j]-I0[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])* (I0[p+1][i+1][j]-I0[p+1][i]
1068                 [j])+(Dm[i][j-1][j]/D[i][j])* (I0[p+1][i][j-1]-I0[p+1][i][j])+(Dm[i][j][i]
1069                 [j+1]/D[i][j])* (I0[p+1][i][j+1]-I0[p+1][i][j])
1070                 for i in range(N3) :
1071                     for j in range(N3) :
1072                         ki=Ki[i][j]
1073                         li=Li[i][j]
1074                         d=D[i][j]/1000000
1075                         s1=-ki*d*s*Si[p][i][j]*Ii[p][i][j]
1076                         il=ki*d*s*Si[p][i][j]*Ii[p][i][j]-li*Ii[p][i][j]
1077                         S2=Si[p][i][j]+(h/2)*s1
1078                         I2=Ii[p][i][j]+(h/2)*il
1079                         s2=-ki*d*s*S2*I2
1080                         i2=ki*d*s*S2*I2-li*I2
1081                         S3=Si[p][i][j]+(h/2)*s2
1082                         I3=Ii[p][i][j]+(h/2)*i2
1083                         s3=-ki*d*s*S3*I3
1084                         i3=ki*d*s*S3*I3-li*I3
1085                         S4=Si[p][i][j]+h*s3
1086                         I4=Ii[p][i][j]+h*i3
1087                         s4=-ki*d*s*S4*I4
1088                         i4=ki*d*s*S4*I4-li*I4
1089                         S0i[p+1][i][j]=Si[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
1090                         I0i[p+1][i][j]=Ii[p][i][j]+(h/6)*(i1+2*i2+2*i3+i4)
1091                         Si[p+1][i][j]=S0i[p+1][i][j]
1092                         Ii[p+1][i][j]=I0i[p+1][i][j]
1093                         Si[p][i][j]=D[i][j]*Si[p][i][j]
1094                         Ii[p][i][j]=D[i][j]*Ii[p][i][j]
1095                         Si[p+1][0][0]=Si[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])* (S0i[p+1][1][0]-
1096                         S0i[p+1][0][0])+ (Dm[0][0][0][1]/D[0][0])* (S0i[p+1][0][1]-S0i[p+1][0][0])
1097                         Ii[p+1][0][0]=Ii[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])* (I0i[p+1][1][0]-
1098                         I0i[p+1][0][0])+ (Dm[0][0][0][1]/D[0][0])* (I0i[p+1][0][1]-I0i[p+1][0][0])

```

```

1087     Si[p+1][0][N3-1]=Si[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0]
1088     [N3-1])* (S0i[p+1][1][N3-1]-S0i[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0]
1089     [N3-1])* (S0i[p+1][0][N3-2]-S0i[p+1][0][N3-1])
1090     Ii[p+1][0][N3-1]=Ii[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0]
1091     [N3-1])* (I0i[p+1][1][N3-1]-I0i[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0]
1092     [N3-1])* (I0i[p+1][0][N3-2]-I0i[p+1][0][N3-1])
1093     Si[p+1][N3-1][0]=Si[p+1][N3-1][0]+(Dm[N3-1][0][N3-1][1]/D[N3-1]
1094     [0])* (S0i[p+1][N3-1][1]-S0i[p+1][N3-1][0])+(Dm[N3-2][0][N3-1][0]/D[N3-1]
1095     [0])* (S0i[p+1][N3-2][0]-S0i[p+1][N3-1][0])
1096     Ii[p+1][N3-1][0]=Ii[p+1][N3-1][0]+(Dm[N3-1][0][N3-1][1]/D[N3-1]
1097     [0])* (I0i[p+1][N3-1][1]-I0i[p+1][N3-1][0])+(Dm[N3-2][0][N3-1][0]/D[N3-1]
1098     [0])* (I0i[p+1][N3-2][0]-I0i[p+1][N3-1][0])
1099     for j in range(1,N3-1) :
1100     Si[p+1][0][j]=Si[p+1][0][j]+(Dm[0][j-1][0][j]/D[0][j])* (S0i[p+1][0]
1101     [j-1]-S0i[p+1][0][j])+(Dm[0][j][0][j+1]/D[0][j])* (S0i[p+1][0][j+1]-S0i[p+1][0]
1102     [j])+(Dm[0][j][1][j]/D[0][j])* (S0i[p+1][1][j]-S0i[p+1][0][j])
1103     Ii[p+1][0][j]=Ii[p+1][0][j]+(Dm[0][j-1][0][j]/D[0][j])* (I0i[p+1][0]
1104     [j-1]-I0i[p+1][0][j])+(Dm[0][j][0][j+1]/D[0][j])* (I0i[p+1][0][j+1]-I0i[p+1][0]
1105     [j])+(Dm[0][j][1][j]/D[0][j])* (I0i[p+1][1][j]-I0i[p+1][0][j])
1106     Si[p+1][N3-1][j]=Si[p+1][N3-1][j]+(Dm[N3-1][j-1][N3-1][j]/D[N3-1]
1107     [j])* (S0i[p+1][N3-1][j-1]-S0i[p+1][N3-1][j])+(Dm[N3-1][j][N3-1][j+1]/D[N3-1]
1108     [j])* (S0i[p+1][N3-1][j+1]-S0i[p+1][N3-1][j])+(Dm[N3-2][j][N3-1][j]/D[N3-1]
1109     [j])* (S0i[p+1][N3-2][j]-S0i[p+1][N3-1][j])
1110     Ii[p+1][N3-1][j]=Ii[p+1][N3-1][j]+(Dm[N3-1][j-1][N3-1][j]/D[N3-1]
1111     [j])* (I0i[p+1][N3-1][j-1]-I0i[p+1][N3-1][j])+(Dm[N3-1][j][N3-1][j+1]/D[N3-1]
1112     [j])* (I0i[p+1][N3-1][j+1]-I0i[p+1][N3-1][j])+(Dm[N3-2][j][N3-1][j]/D[N3-1]
1113     [j])* (I0i[p+1][N3-2][j]-I0i[p+1][N3-1][j])
1114     for i in range(1,N3-1) :
1115     Si[p+1][i][0]=Si[p+1][i][0]+(Dm[i-1][0][i][0]/D[i][0])* (S0i[p+1]
1116     [i-1][0]-S0i[p+1][i][0])+(Dm[i][0][i+1][0]/D[i][0])* (S0i[p+1][i+1][0]-S0i[p+1]
1117     [i][0])+(Dm[i][0][i][1]/D[i][0])* (S0i[p+1][i][1]-S0i[p+1][i][0])
1118     Si[p+1][i][1]=Si[p+1][i][1]+(Dm[i-1][0][i][1]/D[i][1])* (I0i[p+1]
1119     [i-1][0]-I0i[p+1][i][0])+(Dm[i][0][i+1][1]/D[i][1])* (I0i[p+1][i+1][0]-I0i[p+1]
1120     [i][0])+(Dm[i][0][i][2]/D[i][1])* (I0i[p+1][i][2]-I0i[p+1][i][1])
1121     for i in range(N3) :
1122     for j in range(N3) :
1123     E=E+(Si[N][i][j]-S[N][i][j])/Dtot
1124     return(E)

```

```

1124 def espace_uni_e(k,l,s,d,P,dm,N,x,N3,t,λ,μ,Q,zlim) :
1125     if Q==False :
1126         return(0)
1127     d=d/1000000
1128     dm=dm/1000000
1129     Id=[[0 for i in range(N3)] for j in range(N3)]
1130     a=int((N3-1)/2)
1131     Id[a][a]=(N3**2)/P
1132     u=linspace(0,N3-1,N3)
1133     h=x/N
1134     X=[0]*(N3**2)
1135     Y=[0]*(N3**2)
1136     S=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1137     I=[Id]+[[[0 for i in range(N3)] for j in range(N3)] for i in range(N)]
1138     S0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1139     I0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1140     Si=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1141     Ii=[Id]+[[[0 for i in range(N3)] for j in range(N3)] for i in range(N)]
1142     S0i=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1143     I0i=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1144     for i in range(N3) :
1145         for j in range(N3) :
1146             X[N3*i+j]=u[i]
1147             Y[N3*i+j]=u[j]
1148             S[0][i][j]=1-I[0][i][j]
1149             Si[0][i][j]=1-I0[0][i][j]
1150     p=0
1151     while p<t/h :
1152         for i in range(N3) :
1153             for j in range(N3) :
1154                 sl=-k*d*s*S[p][i][j]*I[p][i][j]
1155                 il=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
1156                 S2=S[p][i][j]+(h/2)*sl
1157                 I2=I[p][i][j]+(h/2)*il
1158                 s2=-k*d*s*S2*I2
1159                 i2=k*d*s*S2*I2-l*I2
1160                 S3=S[p][i][j]+(h/2)*s2
1161                 I3=I[p][i][j]+(h/2)*i2
1162                 s3=-k*d*s*S3*I3
1163                 i3=k*d*s*S3*I3-l*I3
1164                 S4=S[p][i][j]+h*s3
1165                 I4=I[p][i][j]+h*i3
1166                 s4=-k*d*s*S4*I4
1167                 i4=k*d*s*S4*I4-l*I4
1168                 S0[p+1][i][j]=S[p][i][j]+(h/6)*(sl+2*s2+2*s3+s4)
1169                 I0[p+1][i][j]=I[p][i][j]+(h/6)*(il+2*i2+2*i3+i4)
1170                 S[p+1][i][j]=S0[p+1][i][j]
1171                 I[p+1][i][j]=I0[p+1][i][j]
1172                 S[p][i][j]=d*S[p][i][j]*1000000
1173                 I[p][i][j]=d*I[p][i][j]*1000000
1174                 S[p+1][0][0]=S[p+1][0][0]+(dm/d)*(S0[p+1][1][0]-S0[p+1][0]
1175 [0])+(dm/d)*(S0[p+1][0][1]-S0[p+1][0][0])
1176                 I[p+1][0][0]=I[p+1][0][0]+(dm/d)*(I0[p+1][1][0]-I0[p+1][0]
1177 [0])+(dm/d)*(I0[p+1][0][1]-I0[p+1][0][0])
1178                 S[p+1][0][N3-1]=S[p+1][0][N3-1]+(dm/d)*(S0[p+1][1][N3-1]-S0[p+1][0]
1179 [N3-1])+(dm/d)*(S0[p+1][0][N3-2]-S0[p+1][0][N3-1])
1180                 I[p+1][0][N3-1]=I[p+1][0][N3-1]+(dm/d)*(I0[p+1][1][N3-1]-I0[p+1][0]
1181 [N3-1])+(dm/d)*(I0[p+1][0][N3-2]-I0[p+1][0][N3-1])
1182                 S[p+1][N3-1][0]=S[p+1][N3-1][0]+(dm/d)*(S0[p+1][N3-1][1]-S0[p+1][N3-1]
1183 [0])+(dm/d)*(S0[p+1][N3-2][0]-S0[p+1][N3-1][0])
1184                 I[p+1][N3-1][0]=I[p+1][N3-1][0]+(dm/d)*(I0[p+1][N3-1][1]-I0[p+1][N3-1]
1185 [0])+(dm/d)*(I0[p+1][N3-2][0]-I0[p+1][N3-1][0])
1186                 S[p+1][N3-1][N3-1]=S[p+1][N3-1][N3-1]+(dm/d)*(S0[p+1][N3-1][N3-2]-S0[p+1][N3-1][N3-1])
1187                 I[p+1][N3-1][N3-1]=I[p+1][N3-1][N3-1]+(dm/d)*(I0[p+1][N3-1][N3-2]-I0[p+1][N3-1][N3-1])
1188                 I[p+1][N3-1][N3-1]=I[p+1][N3-1][N3-1]+(dm/d)*(I0[p+1][N3-2][N3-1]-I0[p+1][N3-1][N3-1])
1189                 for j in range(1,N3-1) :
1190                     S[p+1][0][j]=S[p+1][0][j]+(dm/d)*(S0[p+1][0][j-1]-S0[p+1][0]
1191 [j])+(dm/d)*(S0[p+1][0][j+1]-S0[p+1][0][j])+(dm/d)*(S0[p+1][1][j]-S0[p+1][0]
1192 [j])
1193                     I[p+1][0][j]=I[p+1][0][j]+(dm/d)*(I0[p+1][0][j-1]-I0[p+1][0]
1194 [j])+(dm/d)*(I0[p+1][0][j+1]-I0[p+1][0][j])+(dm/d)*(I0[p+1][1][j]-I0[p+1][0]
1195 [j])

```

```

1185     S[p+1][N3-1][j]=S[p+1][N3-1][j]+(dm/d)*(S0[p+1][N3-1][j-1]-S0[p+1]
1186 [N3-1][j])+(dm/d)*(S0[p+1][N3-1][j+1]-S0[p+1][N3-1][j])+(dm/d)*(S0[p+1][N3-2]
1187 [j]-S0[p+1][N3-1][j])
1188     I[p+1][N3-1][j]=I[p+1][N3-1][j]+(dm/d)*(I0[p+1][N3-1][j-1]-I0[p+1]
1189 [N3-1][j])+(dm/d)*(I0[p+1][N3-1][j+1]-I0[p+1][N3-1][j])+(dm/d)*(I0[p+1][N3-2]
1190 [j]-I0[p+1][N3-1][j])
1191     for i in range(1,N3-1) :
1192         S[p+1][i][0]=S[p+1][i][0]+(dm/d)*(S0[p+1][i-1][0]-S0[p+1][i]
1193 [0])+(dm/d)*(S0[p+1][i+1][0]-S0[p+1][i][0])+(dm/d)*(S0[p+1][i][1]-S0[p+1][i]
1194 [0])
1195         I[p+1][i][0]=I[p+1][i][0]+(dm/d)*(I0[p+1][i-1][0]-I0[p+1][i]
1196 [0])+(dm/d)*(I0[p+1][i+1][0]-I0[p+1][i][0])+(dm/d)*(I0[p+1][i][1]-I0[p+1][i]
1197 [0])
1198         S[p+1][i][N3-1]=S[p+1][i][N3-1]+(dm/d)*(S0[p+1][i-1][N3-1]-S0[p+1]
1199 [i][N3-1])+(dm/d)*(S0[p+1][i+1][N3-1]-S0[p+1][i][N3-1])+(dm/d)*(S0[p+1][i]
1200 [N3-2]-S0[p+1][i][N3-1])
1201         I[p+1][i][N3-1]=I[p+1][i][N3-1]+(dm/d)*(I0[p+1][i-1][N3-1]-I0[p+1]
1202 [i][N3-1])+(dm/d)*(I0[p+1][i+1][N3-1]-I0[p+1][i][N3-1])+(dm/d)*(I0[p+1][i]
1203 [N3-2]-I0[p+1][i][N3-1])
1204         for i in range(1,N3-1) :
1205             for j in range(1,N3-1) :
1206                 S[p+1][i][j]=S[p+1][i][j]+(dm/d)*(S0[p+1][i-1][j]-S0[p+1][i]
1207 [j])+(dm/d)*(S0[p+1][i+1][j]-S0[p+1][i][j])+(dm/d)*(S0[p+1][i][j-1]-S0[p+1][i]
1208 [j])+(dm/d)*(S0[p+1][i][j+1]-S0[p+1][i][j])
1209             I[p+1][i][j]=I[p+1][i][j]+(dm/d)*(I0[p+1][i-1][j]-I0[p+1][i]
1210 [j])+(dm/d)*(I0[p+1][i+1][j]-I0[p+1][i][j])+(dm/d)*(I0[p+1][i][j-1]-I0[p+1][i]
1211 [j])+(dm/d)*(I0[p+1][i][j+1]-I0[p+1][i][j])
1212             for i in range(N3) :
1213                 for j in range(N3) :
1214                     S0i[p+1][i][j]=S0[p+1][i][j]
1215                     I0i[p+1][i][j]=I0[p+1][i][j]
1216                     Si[p+1][i][j]=S[p+1][i][j]
1217                     Ii[p+1][i][j]=I[p+1][i][j]
1218                     Si[p][i][j]=S[p][i][j]
1219                     Ii[p][i][j]=I[p][i][j]
1220             p=p+1
1221     Ki=[[0 for i in range(N3)] for j in range(N3)]
1222     Li=[[0 for i in range(N3)] for j in range(N3)]
1223     for i in range(N3) :
1224         for j in range(N3) :
1225             if Q[i][j]==1 :
1226                 Ki[i][j]=k*(1-lambda)
1227                 Li[i][j]=l/(1-mu)
1228             else :
1229                 Ki[i][j]=k
1230                 Li[i][j]=l
1231     while p<N :
1232         for i in range(N3) :
1233             for j in range(N3) :
1234                 s1=-k*d*s*S[p][i][j]*I[p][i][j]
1235                 il=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
1236                 S2=S[p][i][j]+(h/2)*s1
1237                 I2=I[p][i][j]+(h/2)*il
1238                 s2=-k*d*s*S2*I2
1239                 i2=k*d*s*S2*I2-l*I2
1240                 S3=S[p][i][j]+(h/2)*s2
1241                 I3=I[p][i][j]+(h/2)*i2
1242                 s3=-k*d*s*S3*I3
1243                 i3=k*d*s*S3*I3-l*I3
1244                 S4=S[p][i][j]+h*s3
1245                 I4=I[p][i][j]+h*i3
1246                 s4=-k*d*s*S4*I4
1247                 i4=k*d*s*S4*I4-l*I4
1248                 S0[p+1][i][j]=S[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
1249                 I0[p+1][i][j]=I[p][i][j]+(h/6)*(il+2*i2+2*i3+i4)
1250                 S[p+1][i][j]=S0[p+1][i][j]
1251                 I[p+1][i][j]=I0[p+1][i][j]
1252                 S[p][i][j]=d*S[p][i][j]*1000000
1253                 I[p][i][j]=d*I[p][i][j]*1000000
1254                 S[p+1][0][0]=S[p+1][0][0]+(dm/d)*(S0[p+1][1][0]-S0[p+1][0]
1255 [0])+(dm/d)*(S0[p+1][0][1]-S0[p+1][0][0])
1256                 I[p+1][0][0]=I[p+1][0][0]+(dm/d)*(I0[p+1][1][0]-I0[p+1][0]
1257 [0])+(dm/d)*(I0[p+1][0][1]-I0[p+1][0][0])

```

```

1240     S[p+1][0][N3-1]=S[p+1][0][N3-1]+(dm/d)*(S0[p+1][1][N3-1]-S0[p+1][0]
1241     [N3-1])+(dm/d)*(S0[p+1][0][N3-2]-S0[p+1][0][N3-1])
1242     I[p+1][0][N3-1]=I[p+1][0][N3-1]+(dm/d)*(I0[p+1][1][N3-1]-I0[p+1][0]
1243     [N3-1])+(dm/d)*(I0[p+1][0][N3-2]-I0[p+1][0][N3-1])
1244     S[p+1][N3-1][0]=S[p+1][N3-1][0]+(dm/d)*(S0[p+1][N3-1][1]-S0[p+1][N3-1]
1245     [0])+(dm/d)*(S0[p+1][N3-2][0]-S0[p+1][N3-1][0])
1246     I[p+1][N3-1][0]=I[p+1][N3-1][0]+(dm/d)*(I0[p+1][N3-1][1]-I0[p+1][N3-1]
1247     [0])+(dm/d)*(I0[p+1][N3-2][0]-I0[p+1][N3-1][0])
1248     S[p+1][N3-1][0]=S[p+1][N3-1][0]+(dm/d)*(S0[p+1][N3-1][1]-S0[p+1][N3-2]
1249     [0])+(dm/d)*(S0[p+1][N3-1][0]-S0[p+1][N3-1][0])+(dm/d)*(S0[p+1][1][j]-S0[p+1][0]
1250     [j])
1251     I[p+1][0][j]=I[p+1][0][j]+(dm/d)*(I0[p+1][0][j-1]-I0[p+1][0]
1252     [j])+(dm/d)*(I0[p+1][0][j+1]-I0[p+1][0][j])+(dm/d)*(I0[p+1][1][j]-I0[p+1][0]
1253     [j])
1254     S[p+1][N3-1][j]=S[p+1][N3-1][j]+(dm/d)*(S0[p+1][N3-1][j-1]-S0[p+1]
1255     [N3-1][j])+(dm/d)*(S0[p+1][N3-1][j+1]-S0[p+1][N3-1][j])+(dm/d)*(S0[p+1][N3-2]
1256     [j]-S0[p+1][N3-1][j])
1257     I[p+1][N3-1][j]=I[p+1][N3-1][j]+(dm/d)*(I0[p+1][N3-1][j-1]-I0[p+1]
1258     [N3-1][j])+(dm/d)*(I0[p+1][N3-1][j+1]-I0[p+1][N3-1][j])+(dm/d)*(I0[p+1][N3-2]
1259     [j]-I0[p+1][N3-1][j])
1260     for i in range(1,N3-1) :
1261         S[p+1][i][0]=S[p+1][i][0]+(dm/d)*(S0[p+1][i-1][0]-S0[p+1][i]
1262         [0])+(dm/d)*(S0[p+1][i+1][0]-S0[p+1][i][0])+(dm/d)*(S0[p+1][i][1]-S0[p+1][i]
1263         [0])
1264         I[p+1][i][0]=I[p+1][i][0]+(dm/d)*(I0[p+1][i-1][0]-I0[p+1][i]
1265         [0])+(dm/d)*(I0[p+1][i+1][0]-I0[p+1][i][0])+(dm/d)*(I0[p+1][i][1]-I0[p+1][i]
1266         [0])
1267         S[p+1][i][N3-1]=S[p+1][i][N3-1]+(dm/d)*(S0[p+1][i-1][N3-1]-S0[p+1]
1268     [i][N3-1])+(dm/d)*(S0[p+1][i+1][N3-1]-S0[p+1][i][N3-1])+(dm/d)*(S0[p+1][i]
1269     [i][N3-2]-S0[p+1][i][N3-1])
1270         I[p+1][i][N3-1]=I[p+1][i][N3-1]+(dm/d)*(I0[p+1][i-1][N3-1]-I0[p+1][i]
1271     [N3-1])+(dm/d)*(I0[p+1][i+1][N3-1]-I0[p+1][i][N3-1])+(dm/d)*(I0[p+1][i][N3-2]
1272     -I0[p+1][i][N3-1])
1273         for i in range(1,N3-1) :
1274             for j in range(1,N3-1) :
1275                 S[p+1][i][j]=S[p+1][i][j]+(dm/d)*(S0[p+1][i-1][j]-S0[p+1][i]
1276     [j])+(dm/d)*(S0[p+1][i+1][j]-S0[p+1][i][j])+(dm/d)*(S0[p+1][i][j-1]-S0[p+1][i]
1277     [j])+(dm/d)*(S0[p+1][i][j+1]-S0[p+1][i][j])
1278                 I[p+1][i][j]=I[p+1][i][j]+(dm/d)*(I0[p+1][i-1][j]-I0[p+1][i]
1279     [j])+(dm/d)*(I0[p+1][i+1][j]-I0[p+1][i][j])+(dm/d)*(I0[p+1][i][j-1]-I0[p+1][i]
1280     [j])+(dm/d)*(I0[p+1][i][j+1]-I0[p+1][i][j])
1281                 for i in range(N3) :
1282                     for j in range(N3) :
1283                         ki=Ki[i][j]
1284                         li=Li[i][j]
1285                         s1=-ki*d*s*Si[p][i][j]*Ii[p][i][j]
1286                         i1=ki*d*s*Si[p][i][j]*Ii[p][i][j]-li*Ii[p][i][j]
1287                         S2=Si[p][i][j]+(h/2)*s1
1288                         I2=Ii[p][i][j]+(h/2)*i1
1289                         s2=-ki*d*s*S2*I2
1290                         i2=ki*d*s*S2*I2-li*I2
1291                         S3=Si[p][i][j]+(h/2)*s2
1292                         I3=Ii[p][i][j]+(h/2)*i2
1293                         s3=-ki*d*s*S3*I3
1294                         i3=ki*d*s*S3*I3-li*I3
1295                         S4=Si[p][i][j]+h*s3
1296                         I4=Ii[p][i][j]+h*i3
1297                         s4=-ki*d*s*S4*I4
1298                         i4=ki*d*s*S4*I4-li*I4
1299                         S0i[p+1][i][j]=Si[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
1300                         I0i[p+1][i][j]=Ii[p][i][j]+(h/6)*(i1+2*i2+2*i3+i4)
1301                         Si[p+1][i][j]=S0i[p+1][i][j]
1302                         Ii[p+1][i][j]=I0i[p+1][i][j]
1303                         Si[p][i][j]=d*Si[p][i][j]*1000000
1304                         Ii[p][i][j]=d*Ii[p][i][j]*1000000
1305                         Si[p+1][0][0]=Si[p+1][0][0]+(dm/d)*(S0i[p+1][1][0]-S0i[p+1][0]
1306     [0])+(dm/d)*(S0i[p+1][0][1]-S0i[p+1][0][0])

```

```

1286     Si[p+1][0][N3-1]=Si[p+1][0][N3-1]+(dm/d)*(S0i[p+1][1][N3-1]-S0i[p+1][0]
1287     [N3-1])+(dm/d)*(S0i[p+1][0][N3-2]-S0i[p+1][0][N3-1])
1288     Ii[p+1][0][N3-1]=Ii[p+1][0][N3-1]+(dm/d)*(I0i[p+1][1][N3-1]-I0i[p+1][0]
1289     [N3-1])+(dm/d)*(I0i[p+1][0][N3-2]-I0i[p+1][0][N3-1])
1290     Si[p+1][N3-1][0]=Si[p+1][N3-1][0]+(dm/d)*(S0i[p+1][N3-1][1]-S0i[p+1]
1291     [N3-1][0])
1292     Ii[p+1][N3-1][0]=Ii[p+1][N3-1][0]+(dm/d)*(I0i[p+1][N3-1][1]-I0i[p+1]
1293     [N3-1][0])
1294     Si[p+1][N3-1][N3-1]=Si[p+1][N3-1][N3-1]+(dm/d)*(S0i[p+1][N3-1][N3-2]-S0i[p+1][N3-1][N3-1])
1295     Ii[p+1][N3-1][N3-1]=Ii[p+1][N3-1][N3-1]+(dm/d)*(I0i[p+1][N3-1][N3-2]-I0i[p+1][N3-1][N3-1])
1296     for j in range(1,N3-1) :
1297         Si[p+1][0][j]=Si[p+1][0][j]+(dm/d)*(S0i[p+1][0][j-1]-S0i[p+1][0]
1298         [j])+(dm/d)*(S0i[p+1][0][j+1]-S0i[p+1][0][j])+(dm/d)*(S0i[p+1][1][j]-S0i[p+1]
1299         [0][j])
1300         Ii[p+1][0][j]=Ii[p+1][0][j]+(dm/d)*(I0i[p+1][0][j-1]-I0i[p+1][0]
1301         [j])+(dm/d)*(I0i[p+1][0][j+1]-I0i[p+1][0][j])+(dm/d)*(I0i[p+1][1][j]-I0i[p+1]
1302         [0][j])
1303         Si[p+1][N3-1][j]=Si[p+1][N3-1][j]+(dm/d)*(S0i[p+1][N3-1][j-1]-S0i[p+1][N3-1]
1304         [j])+(dm/d)*(S0i[p+1][N3-1][j+1]-S0i[p+1][N3-1][j])
1305         Ii[p+1][N3-1][j]=Ii[p+1][N3-1][j]+(dm/d)*(I0i[p+1][N3-1][j-1]-I0i[p+1][N3-1]
1306         [j])+(dm/d)*(I0i[p+1][N3-1][j+1]-I0i[p+1][N3-1][j])
1307         for i in range(1,N3-1) :
1308             for j in range(1,N3-1) :
1309                 Si[p+1][i][j]=Si[p+1][i][j]+(dm/d)*(S0i[p+1][i-1][j]-S0i[p+1]
1310                 [i][j])+(dm/d)*(S0i[p+1][i+1][j]-S0i[p+1][i][j])+(dm/d)*(S0i[p+1][i][j-1]-S0i[p+1]
1311                 [i][j])+(dm/d)*(S0i[p+1][i][j+1]-S0i[p+1][i][j])
1312                 Ii[p+1][i][j]=Ii[p+1][i][j]+(dm/d)*(I0i[p+1][i-1][j]-I0i[p+1]
1313                 [i][j])+(dm/d)*(I0i[p+1][i+1][j]-I0i[p+1][i][j])+(dm/d)*(I0i[p+1][i][j-1]-I0i[p+1]
1314                 [i][j])+(dm/d)*(I0i[p+1][i][j+1]-I0i[p+1][i][j])
1315                 p=p+1
1316             for i in range(N3) :
1317                 for j in range(N3) :
1318                     E=E+(Si[N][i][j]-S[N][i][j])/Dtot
1319             return(E)
1320
1321 def espace_adj_t(K,L,s,D,Id,Dm,t,N,x,N3,zlim) :
1322     h=x/N
1323     S=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1324     I=[[Id]+[[0 for i in range(N3)] for j in range(N3)] for t in range(N)] for t in range(N+1)]
1325     S0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1326     I0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1327     for i in range(N3) :
1328         for j in range(N3) :
1329             S[0][i][j]=1-I[0][i][j]
1330     p=0
1331     while p<t/h :
1332         for i in range(N3) :
1333             for j in range(N3) :

```

```

1333     k=K[i][j]
1334     l=L[i][j]
1335     d=D[i][j]/1000000
1336     s1=-k*d*s*S[p][i][j]*I[p][i][j]
1337     i1=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
1338     S2=S[p][i][j]+(h/2)*s1
1339     I2=I[p][i][j]+(h/2)*i1
1340     s2=-k*d*s*S2*I2
1341     i2=k*d*s*S2*I2-l*I2
1342     S3=S[p][i][j]+(h/2)*s2
1343     I3=I[p][i][j]+(h/2)*i2
1344     s3=-k*d*s*S3*I3
1345     i3=k*d*s*S3*I3-l*I3
1346     S4=S[p][i][j]+h*s3
1347     I4=I[p][i][j]+h*i3
1348     s4=-k*d*s*S4*I4
1349     i4=k*d*s*S4*I4-l*I4
1350     S0[p+1][i][j]=S[p][i][j]+(h/6)*(s1+2*s2+2*s3+s4)
1351     I0[p+1][i][j]=I[p][i][j]+(h/6)*(i1+2*i2+2*i3+i4)
1352     S[p+1][i][j]=S0[p+1][i][j]
1353     I[p+1][i][j]=I0[p+1][i][j]
1354     S[p][i][j]=D[i][j]*S[p][i][j]
1355     I[p][i][j]=D[i][j]*I[p][i][j]
1356     S[p+1][0][0]=S[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])* (S0[p+1][1][0]-
1357     S0[p+1][0][0])+ (Dm[0][0][0][1]/D[0][0])* (S0[p+1][0][1]-S0[p+1][0][0])
1358     I[p+1][0][0]=I[p+1][0][0]+(Dm[0][0][1][0]/D[0][0])* (I0[p+1][1][0]-
1359     I0[p+1][0][0])+ (Dm[0][0][0][1]/D[0][0])* (I0[p+1][0][1]-I0[p+1][0][0])
1360     S[p+1][0][N3-1]=S[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0]
1361     [N3-1])* (S0[p+1][1][N3-1]-S0[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0]
1362     [N3-1])* (S0[p+1][0][N3-2]-S0[p+1][0][N3-1])
1363     I[p+1][0][N3-1]=I[p+1][0][N3-1]+(Dm[0][N3-1][1][N3-1]/D[0]
1364     [N3-1])* (I0[p+1][1][N3-1]-I0[p+1][0][N3-1])+(Dm[0][N3-2][0][N3-1]/D[0]
1365     [N3-1])* (I0[p+1][0][N3-2]-I0[p+1][0][N3-1])
1366     S[p+1][N3-1][0]=S[p+1][N3-1][0]+(Dm[N3-1][0][N3-1][1]/D[N3-1]
1367     [0])* (S0[p+1][N3-1][1]-S0[p+1][N3-1][0])+(Dm[N3-2][0][N3-1][0]/D[N3-1]
1368     [0])* (S0[p+1][N3-2][0]-S0[p+1][N3-1][0])
1369     I[p+1][N3-1][0]=I[p+1][N3-1][0]+(Dm[N3-1][0][N3-1][1]/D[N3-1]
1370     [0])* (I0[p+1][N3-1][1]-I0[p+1][N3-1][0])+(Dm[N3-2][0][N3-1][0]/D[N3-1]
1371     [0])* (I0[p+1][N3-2][0]-I0[p+1][N3-1][0])
1372     for i in range(1,N3-1) :
1373         S[p+1][i][0]=S[p+1][0][0]+(Dm[i-1][0][i][0]/D[i][0])* (S0[p+1][i-1]
1374     [0]-S0[p+1][i][0])+ (Dm[i][0][i+1][0]/D[i][0])* (S0[p+1][i+1][0]-S0[p+1][i]
1375     [0])+ (Dm[i][0][i][1]/D[i][0])* (S0[p+1][i][1]-S0[p+1][i][0])
1376     I[p+1][i][0]=I[p+1][0][0]+(Dm[i-1][0][i][0]/D[i][0])* (I0[p+1][i-1]
1377     [0]-I0[p+1][i][0])+ (Dm[i][0][i+1][0]/D[i][0])* (I0[p+1][i+1][0]-I0[p+1][i]
1378     [0])+ (Dm[i][0][i][1]/D[i][0])* (I0[p+1][i][1]-I0[p+1][i][0])
1379     S[p+1][i][N3-1]=S[p+1][i][N3-1]+(Dm[i-1][N3-1][i][N3-1]/D[i]
1380     [N3-1])* (S0[p+1][i-1][N3-1]-S0[p+1][i][N3-1])+(Dm[i][N3-1][i+1][N3-1]/D[i]
1381     [N3-1])* (S0[p+1][i+1][N3-1]-S0[p+1][i][N3-1])+(Dm[i][N3-2][i][N3-1]/D[i]
1382     [N3-1])* (S0[p+1][i][N3-2]-S0[p+1][i][N3-1])

```

```

1373 [N3-1])*(I0[p+1][i-1][N3-1]-I0[p+1][i][N3-1])+(Dm[i][N3-1][i+1][N3-1]/D[i]
1374 [N3-1])*(I0[p+1][i+1][N3-1]-I0[p+1][i][N3-1])+(Dm[i][N3-2][i][N3-1]/D[i]
1375 [N3-1])*(I0[p+1][i][N3-2]-I0[p+1][i][N3-1])
1376         for i in range(1,N3-1) :
1377             for j in range(1,N3-1) :
1378                 S[p+1][i][j]=S[p+1][i][j]+(Dm[i-1][j][i][j]/D[i][j])* (S0[p+1]
1379 [i-1][j]-S0[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])* (S0[p+1][i+1][j]-S0[p+1][i]
1380 [j])+(Dm[i][j-1][i][j]/D[i][j])* (S0[p+1][i][j-1]-S0[p+1][i][j])+(Dm[i][j][i]
1381 [j+1]/D[i][j])* (S0[p+1][i][j+1]-S0[p+1][i][j])
1382                 I[p+1][i][j]=I[p+1][i][j]+(Dm[i-1][j][i][j]/D[i][j])* (I0[p+1]
1383 [i-1][j]-I0[p+1][i][j])+(Dm[i][j][i+1][j]/D[i][j])* (I0[p+1][i+1][j]-I0[p+1][i]
1384 [j])+(Dm[i][j-1][i][j]/D[i][j])* (I0[p+1][i][j-1]-I0[p+1][i][j])+(Dm[i][j][i]
1385 [j+1]/D[i][j])* (I0[p+1][i][j+1]-I0[p+1][i][j])
1386             p=p+1
1387         for i in range(N3) :
1388             for j in range(N3) :
1389                 S[p][i][j]=D[i][j]*S[p][i][j]
1390                 I[p][i][j]=D[i][j]*I[p][i][j]
1391     return(S[p],I[p])
1392
1393 def espace_uni_t(k,l,s,d,P,dm,t,N,x,N3,zlim) :
1394     d=d/1000000
1395     dm=dm/1000000
1396     Id=[[0 for i in range(N3)] for j in range(N3)]
1397     a=int((N3-1)/2)
1398     Id[a][a]=(N3**2)/P
1399     h=x/N
1400     S=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1401     I=[Id]+[[[0 for i in range(N3)] for j in range(N3)] for t in range(N)]
1402     S0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1403     I0=[[0 for i in range(N3)] for j in range(N3)] for t in range(N+1)]
1404     for i in range(N3) :
1405         for j in range(N3) :
1406             S[0][i][j]=1-I[0][i][j]
1407     p=0
1408     while p<t/h :
1409         for i in range(N3) :
1410             for j in range(N3) :
1411                 sl=-k*d*s*S[p][i][j]*I[p][i][j]
1412                 il=k*d*s*S[p][i][j]*I[p][i][j]-l*I[p][i][j]
1413                 S2=S[p][i][j]+(h/2)*sl
1414                 I2=I[p][i][j]+(h/2)*il
1415                 s2=-k*d*s*S2*I2
1416                 i2=k*d*s*S2*I2-l*I2
1417                 S3=S[p][i][j]+(h/2)*s2
1418                 I3=I[p][i][j]+(h/2)*i2
1419                 s3=-k*d*s*S3*I3
1420                 i3=k*d*s*S3*I3-l*I3
1421                 S4=S[p][i][j]+h*s3
1422                 I4=I[p][i][j]+h*i3
1423                 s4=-k*d*s*S4*I4
1424                 i4=k*d*s*S4*I4-l*I4
1425                 S0[p+1][i][j]=S[p][i][j]+(h/6)*(sl+2*s2+2*s3+s4)
1426                 I0[p+1][i][j]=I[p][i][j]+(h/6)*(il+2*i2+2*i3+i4)
1427                 S[p+1][i][j]=S0[p+1][i][j]
1428                 I[p+1][i][j]=I0[p+1][i][j]
1429                 S[p][i][j]=d*S[p][i][j]*1000000
1430                 I[p][i][j]=d*I[p][i][j]*1000000
1431                 S[p+1][0][0]=S[p+1][0][0]+(dm/d)*(S0[p+1][1][0]+S0[p+1][0][1]-2*S0[p+1]
1432 [0][0])
1433                 I[p+1][0][0]=I[p+1][0][0]+(dm/d)*(I0[p+1][1][0]+I0[p+1][0][1]-2*I0[p+1]
1434 [0][0])
1435                 S[p+1][0][N3-1]=S[p+1][0][N3-1]+(dm/d)*(S0[p+1][1][N3-1]+S0[p+1][0]
1436 [N3-2]-2*S0[p+1][0][N3-1])
1437                 I[p+1][0][N3-1]=I[p+1][0][N3-1]+(dm/d)*(I0[p+1][1][N3-1]+I0[p+1][0]
1438 [N3-2]-2*I0[p+1][0][N3-1])
1439                 S[p+1][N3-1][0]=S[p+1][N3-1][0]+(dm/d)*(S0[p+1][N3-1][1]+S0[p+1][N3-2]
1440 [0]-2*S0[p+1][N3-1][0])
1441                 I[p+1][N3-1][0]=I[p+1][N3-1][0]+(dm/d)*(I0[p+1][N3-1][1]+I0[p+1][N3-2]
1442 [0]-2*I0[p+1][N3-1][0])
1443                 S[p+1][N3-1][N3-1]=S[p+1][N3-1][N3-1]+(dm/d)*(S0[p+1][N3-1]
1444 [N3-2]+S0[p+1][N3-2][N3-1]-2*S0[p+1][N3-1][N3-1])
1445                 I[p+1][N3-1][N3-1]=I[p+1][N3-1][N3-1]+(dm/d)*(I0[p+1][N3-1]
1446 [N3-2]+I0[p+1][N3-2][N3-1]-2*I0[p+1][N3-1][N3-1])

```

```

1431     for j in range(1,N3-1) :
1432         S[p+1][0][j]=S[p+1][0][j]+(dm/d)*(S0[p+1][0][j-1]+S0[p+1][0]
1433 [j+1]+S0[p+1][1][j]-3*S0[p+1][0][j])
1434         I[p+1][0][j]=I[p+1][0][j]+(dm/d)*(I0[p+1][0][j-1]+I0[p+1][0]
1435 [j+1]+I0[p+1][1][j]-3*I0[p+1][0][j])
1436         S[p+1][N3-1][j]=S[p+1][N3-1][j]+(dm/d)*(S0[p+1][N3-1][j-1]+S0[p+1]
1437 [N3-1][j+1]+S0[p+1][N3-2][j]-3*S0[p+1][N3-1][j])
1438         I[p+1][N3-1][j]=I[p+1][N3-1][j]+(dm/d)*(I0[p+1][N3-1][j-1]+I0[p+1]
1439 [N3-1][j+1]+I0[p+1][N3-2][j]-3*I0[p+1][N3-1][j])
1440         for i in range(1,N3-1) :
1441             S[p+1][i][0]=S[p+1][i][0]+(dm/d)*(S0[p+1][i-1][0]+S0[p+1][i+1]
1442 [0]+S0[p+1][i][1]-3*S0[p+1][i][0])
1443             I[p+1][i][0]=I[p+1][i][0]+(dm/d)*(I0[p+1][i-1][0]+I0[p+1][i+1]
1444 [0]+I0[p+1][i][1]-3*I0[p+1][i][0])
1445             S[p+1][i][N3-1]=S[p+1][i][N3-1]+(dm/d)*(S0[p+1][i-1][N3-1]+S0[p+1]
1446 [i+1][N3-1]+S0[p+1][i][N3-2]-3*S0[p+1][i][N3-1])
1447             I[p+1][i][N3-1]=I[p+1][i][N3-1]+(dm/d)*(I0[p+1][i-1][N3-1]+I0[p+1]
1448 [i+1][N3-1]+I0[p+1][i][N3-2]-3*I0[p+1][i][N3-1])
1449             for i in range(1,N3-1) :
1450                 for j in range(1,N3-1) :
1451                     S[p+1][i][j]=S[p+1][i][j]+(dm/d)*(S0[p+1][i-1][j]+S0[p+1][i+1]
1452 [j]+S0[p+1][i][j-1]+S0[p+1][i][j+1]-4*S0[p+1][i][j])
1453                     I[p+1][i][j]=I[p+1][i][j]+(dm/d)*(I0[p+1][i-1][j]+I0[p+1][i+1]
1454 [j]+I0[p+1][i][j-1]+I0[p+1][i][j+1]-4*I0[p+1][i][j])
1455                     p=p+1
1456                 for i in range(N3) :
1457                     for j in range(N3) :
1458                         S[p][i][j]=d*S[p][i][j]*1000000
1459                         I[p][i][j]=d*I[p][i][j]*1000000
1460             return(S[p],I[p])
1461
1462 def espace_adj_elieu(K,L,s,D,Id,Dm,N,x,N3,t,λ,μ,p,R0,zlim) :
1463     E=[0 for i in range(len(R0))]
1464     Q=[[0 for i in range(N3)] for j in range(N3)]
1465     I=espace_adj_t(K,L,s,D,Id,Dm,t,N,x,N3,zlim)[1]
1466     for r in range(len(R0)) :
1467         r0=R0[r]
1468         Q=q(N3,p,r0)
1469         E[r]=espace_adj_e(K,L,s,D,Id,Dm,N,x,N3,t,λ,μ,Q,zlim)
1470     Sf=espace_adj_t(K,L,s,D,Id,Dm,x,N,x,N3,zlim)[0]
1471     a=0
1472     Dtot=0
1473     for i in range(N3) :
1474         for j in range(N3) :
1475             a=a+Sf[i][j]
1476             Dtot=Dtot+D[i][j]
1477     Rfinal=1-a/Dtot
1478     M=0
1479     R=0
1480     for r in range(len(R0)) :
1481         if E[r]>=M :
1482             M=E[r]
1483             R=R0[r]
1484     Q=q(N3,p,R)
1485     espace_sub_3(I,Q,t,M,Rfinal,N3,zlim)
1486
1487 def espace_sub_3(I,Q,t,M,Rfinal,N3,zlim) :
1488     X=[0]*(N3**2)
1489     Y=[0]*(N3**2)
1490     Z=[0]*(N3**2)
1491     u=linspace(0,N3-1,N3)
1492     for i in range(N3) :
1493         for j in range(N3) :
1494             X[N3*i+j]=u[i]
1495             Y[N3*i+j]=u[j]
1496     Xq=[]
1497     Yq=[]
1498     Zq=[]
1499     for i in range(N3) :
1500         for j in range(N3) :
1501             Z[N3*i+j]=I[i][j]
1502             if Q[i][j]==1 :
1503                 Xq=Xq+[i]
1504                 Yq=Yq+[j]

```

```

1495     |     |     Zq=Zq+[zlim]
1496     |     |     fig=figure()
1497     |     |     ax=fig.gca(projection='3d')
1498     |     |     ax.set_zlim3d(0,zlim)
1499     |     |     ax.set_title('t=' + str(int(t*100)/100) + ' ut' + '
1500     |     |     e = ' + str(int(M*10**3-ordredegrandeur(M)))/10**3-ordredegrandeur(M)) + '
1501     |     |     Rfinal = ' + str(int(Rfinal*10**3-ordredegrandeur(Rfinal)))/10**3-
1502     |     |     ordredegrandeur(Rfinal)))
1503     |     |     ax.set_xlabel('x')
1504     |     |     ax.set_ylabel('y')
1505     |     |     ax.text(N3*0.95,N3*0.75,zlim*1.2,'d(I) (hab/km2)',color=(0,0,0),size=12)
1506     |     |     ax.plot(X,Y,Z,'o',color=(0,0,0.75))
1507     |     |     ax.plot(Xq,Yq,Zq,'o',color=(0,0.75,0))
1508
1509     |     def ordredegrandeur(M) :
1510         c=0
1511         while M<1 :
1512             M=10*M
1513             c=c-1
1514             return(c)
1515
1516     |     def espace_uni_elieu(k,l,s,d,P,dm,N,x,N3,t,λ,μ,p,R0,zlim) :
1517         E=[0 for i in range(len(R0))]
1518         Q=[[0 for i in range(N3)] for j in range(N3)]
1519         I=espace_uni_t(k,l,s,d,P,dm,t,N,x,N3,zlim)[1]
1520         for r in range(len(R0)) :
1521             r0=R0[r]
1522             Q=q(N3,p,r0)
1523             E[r]=espace_uni_e(k,l,s,d,P,dm,N,x,N3,t,λ,μ,Q,zlim)
1524             Sf=espace_uni_t(k,l,s,d,P,dm,x,N,x,N3,zlim)[0]
1525             a=0
1526             Dtot=d*N3**2
1527             for i in range(N3) :
1528                 for j in range(N3) :
1529                     a=a+Sf[i][j]
1530             Rfinal=1-a/Dtot
1531             M=0
1532             R=0
1533             for r in range(len(R0)) :
1534                 if E[r]>=M :
1535                     M=E[r]
1536                     R=R0[r]
1537             Q=q(N3,p,R)
1538             espace_sub_3(I,Q,t,M,Rfinal,N3,zlim)
1539
1540     |     def dm(D,m) :
1541         N3=len(D)
1542         Dm=[[[[0 for i in range(N3)] for j in range(N3)] for t in range(N3)] for u
1543         in range(N3)]
1544         Dm[0][0][0][1]=m*max(D[0][0],D[0][1])
1545         Dm[0][0][1][0]=m*max(D[0][0],D[1][0])
1546         Dm[0][N3-1][1][N3-1]=m*max(D[0][N3-1],D[1][N3-1])
1547         Dm[N3-1][0][N3-1][1]=m*max(D[N3-1][0],D[N3-1][1])
1548         for j in range(1,N3-1) :
1549             Dm[0][j][0][j+1]=m*max(D[0][j],D[0][j+1])
1550             Dm[0][j][1][j]=m*max(D[0][j],D[1][j])
1551             Dm[N3-1][j][N3-1][j+1]=m*max(D[N3-1][j],D[N3-1][j+1])
1552         for i in range(1,N3-1) :
1553             Dm[i][0][i+1][0]=m*max(D[i][0],D[i+1][0])
1554             Dm[i][0][i][1]=m*max(D[i][0],D[i][1])
1555             Dm[i][N3-1][i+1][N3-1]=m*max(D[i][N3-1],D[i+1][N3-1])
1556         for i in range(1,N3-1) :
1557             for j in range(1,N3-1) :
1558                 Dm[i][j][i+1][j]=m*max(D[i][j],D[i+1][j])
1559                 Dm[i][j][i][j+1]=m*max(D[i][j],D[i][j+1])
1560
1561     |     def gauss_ex(A,B,C) :
1562         D=[[0 for i in range(41)] for j in range(41)]
1563         for i in range(41) :
1564             for j in range(41) :
1565                 if j<15 :
1566                     if i<12 :
1567                         D[i][j]=A*exp(-(((i-3)**2)/B)-(((j-7)**2)/B))+C

```

```

1565     elif i<29 :
1566         D[i][j]=A*exp(-(((i-20)**2)/B)-(((j-7)**2)/B))+C
1567     else :
1568         D[i][j]=A*exp(-(((i-37)**2)/B)-(((j-7)**2)/B))+C
1569     elif j<30 :
1570         if i<21 :
1571             D[i][j]=A*exp(-(((i-12)**2)/B)-(((j-22)**2)/B))+C
1572         else :
1573             D[i][j]=A*exp(-(((i-28)**2)/B)-(((j-22)**2)/B))+C
1574     else :
1575         if i<12 :
1576             D[i][j]=A*exp(-(((i-3)**2)/B)-(((j-37)**2)/B))+C
1577         elif i<29 :
1578             D[i][j]=A*exp(-(((i-20)**2)/B)-(((j-37)**2)/B))+C
1579         else :
1580             D[i][j]=A*exp(-(((i-37)**2)/B)-(((j-37)**2)/B))+C
1581     return(D)
1582
1583 def tri(R) :
1584     n=len(R)
1585     if n==1 :
1586         return(R)
1587     else :
1588         a=int((n+1)/2)
1589         A=R[:a]
1590         B=R[a:]
1591         A2=tri(A)
1592         B2=tri(B)
1593         C=[]
1594         p=0
1595         q=0
1596         for i in range(n) :
1597             if p==a :
1598                 C=C+[B2[q]]
1599                 q=q+1
1600             elif q==n-a :
1601                 C=C+[A2[p]]
1602                 p=p+1
1603             else :
1604                 u=A2[p][0]
1605                 v=B2[q][0]
1606                 if u<=v :
1607                     C=C+[A2[p]]
1608                     p=p+1
1609                 else :
1610                     C=C+[B2[q]]
1611                 q=q+1
1612     return(C)

```

Fonctions non détaillées précédemment :

espace_sub, **espace_sub2** et **espace_sub3** sont des fonctions subordonnées pour tracer les graphiques.

La fonction **q** permet de créer la matrice Q associée à l'anneau de rayon intérieur r_0 et contenant $p \cdot N^3$ cases.

espace_adj_t et **espace_uni_t** renvoient le couple de matrice (S, I) à l'instant t fixé.

ordredegrandeur renvoie la puissance de 10 significative de M .

dm permet de créer facilement une matrice D_m pour une matrice de densité D non uniforme, m caractérisant le mouvement (plus m est grand, plus il y a de mouvement).

gauss_ex est la fonction dont je me suis servi pour créer la matrice de densité D constitué de pics gaussiens en dimension 2 pour le second exemple.

tri est une fonction de tri de listes utilisée dans la fonction q .