BIBLIOGRAPHY

Abdulmajid, S., Hassan, A. S. 2021. Analysis of Time Delayed Rabies Model in Human and Dog Populations with Controls. *Afrika Matematika*, 2021(32), 1067-1085. https://doi.org/10.1007/s13370-021-00882-w

Asamoah, J. K. K., Oduro, F. T., Bonyah, E., Seidu, B. 2017. Modelling of Rabies Transmission Dynamics Using Optimal Control Analysis. *Hindawi Journal of Applied Mathematics*, 2017. https://doi.org/10.1155/2017/2451237

Bornaa, C. S., Seidu, B., Daabo, M. I. 2020. Mathematical Analysis of Rabies Infection. *Hindawi Journal of Applied Mathematics*, 2020. https://doi.org/10.1155/2020/1804270

Brauer, F., Chaves, C. C. 2012. *Mathematical Models in Population Biology and Epidemiology, Second Edition*. New York: Springer.

CDC. 2022. "Animals and Rabies", https://www.cdc.gov/rabies/animals/index.html, accessed on January 21, 2023 at 6:25 pm.

Chen, J., Zou, L., Jin, Z., Ruan, S. 2015. Modeling the Geographic Spread of Rabies in China. *PLOS Neglected Tropical Diseases*, 9(5). https://doi.org/10.1371/journal.pntd.0003772

Clinic, C. 2022. "Rabies: Causes, Symptoms, Treatment & Prevention", https://my.clevelandclinic.org/health/diseases/13848-rabies, accessed on January 22, 2023 at 5:39 pm.

Corless, R. M., Gonnet, G. H., Hare, D. E. G., Jeffery, D. J., Knuth, D. E. 1996. On the Lambert W Function. *Adv. Comput. Math.*, 5(1996), 329-359.

Diekmann, O., Heesterbeek, J. A. P., Roberts, M. G. 2009. The Construction of Next-Generation Matrices for Compartmental Epidemic Models. *J. R. Soc. Interface*, 2010(7), 873-885. https://doi.org/10.1098/rsif.2009.0386

Haberman, R. 1998. *Mathematical Models (Mechanical Vibrations, Population Dynamics, and Traffic Flow)*. New Jersey: SIAM.

Huang, Y., Li, M. 2020. Application of a Mathematical Model in Determining the Spread of the Rabies Virus: Simulation Study. *JMIR Med Inform*, 8(5). https://doi.org/10.2196/18627

Jao, Y. 2019. *Analisis Kestabilan Model Waktu Tunda Respon Patogen terhadap Penambahan Protein Terapeutik Pada Sistem Imun Inang* [Bachelor Thesis]. Makassar: Universitas Hasanuddin.

Kasbawati, Gunawan, A. Y., Hertadi, R., Sidarto, K. A. 2014. Effects of Time Delay on the Dynamics of a Kinetic Model of a Microbial Fermentation Process. *ANZIAM J.*, 55(2014), 336-356. https://doi.org/10.1017/S1446181114000194

Kemenkes. 2020. "8 dari 34 Provinsi di Indonesia Bebas Rabies", https://www.kemkes.go.id/article/view/20092900001/8-dari-34-provinsi-di-indonesia-bebas-rabies.html, accessed on January 24, 2023 at 12:04 am.

Ndii, M. Z., Amarti, Z., Wiraningsih, E. D., Supriatna, A. K. 2018. Rabies Epidemic Model with Uncertainty in Parameters: Crisp and Fuzzy Approaches. *IOP Conf. Ser.: Mater. Sci. Eng.*, 332 012031. https://doi.org/10.1088/1757-899X/332/1/012031

Olsder, G. J. 1998. *Mathematical Systems Theory, Second Edition*. Delft: Delft University Press.

Pantha, B., Giri, S., Joshi, H. R., Vaidya, N. K. 2021. Modeling Transmission Dynamics of Rabies in Nepal. *Infectious Disease Modelling*, 2021(6), 284-301. https://doi.org/10.1016/j.idm.2020.12.009

Renald, E. K., Kuznetsov, D., Kreppel, K. 2020. Desirable Dog-Rabies Control Methods in an Urban Setting in Africa – A Mathematical Model. *I.J. Mathematical Sciences and Computing*, 2020(1), 49-67. https://doi.org/10.5815/ijmsc.2020.01.05

Song, Z.G., Xu, J. 2013. Stability Switches and Double Hopf Bifurcation in a Two-Neural Network System with Multiple Delays. Cogn. Neurodyn., 2013(7), 505-521. https://doi.org/10.1007/s11571-013-9254-0

Tohma, K., Saito, M., Demetria, C. S., Manalo, D. L., Quiambao, B. P., Kamigaki, T., Oshitani, H. 2015. Molecular and Mathematical Modeling Analyses of Inter-Island Transmission of Rabies into a Previously Rabies-Free Island in Philippines. *Infection, Genetics and Evolution*, 38, 22-28. https://doi.org/10.1016/j.meegid.2015.12.001

WHO. 2023. "Rabies", https://www.who.int/news-room/fact-sheets/detail/rabies, accessed on January 21, 2023 at 6:26 pm.

Wiggins, S. 2003. *Introduction to Applied Nonlinear Dynamical Systems and Chaos, Second Edition*. New York: Springer-Verlag.

Zhang, J., Zhen, J., Sun, G-S., Zhou, T., Ruan, S. 2011. Analysis of Rabies in China: Transmission Dynamics and Control. *PLoS ONE*, 6(7), e20891. https://doi.org/10.1371/journal.pone.0020891

Zhong, Q. C. 2006. *Robust Control of Time-delay Systems*. Liverpool: Springer.

Zill, D. G., Cullen, M. R. 2009. *Differential Equations with Boundary-Value Problems, Seventh Edition*. Belmont: Brooks/Cole Cengage Learning.

APPENDIX

Appendix 1 Relation of the roots of (4.33) and τ_2

```
import matplotlib.pyplot as plt
import numpy as np

mh=0.06
kh=0.5

xl=-10; xh=0; dx=(xh-xl)/1000
yl=0; yh=1; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)
eq1=x+mh+kh*np.exp(-y*(x+mh))

plt.style.use('classic')
plt.rc('axes',labelsize=16); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.contour(x,y,eq1,[0],colors='blue',linewidths=2)
plt.xlabel(r'$\lambda$'); plt.ylabel(r'$\tau_2$ (year)')
plt.show()
```

Appendix 2 Relation of the real part of the complex roots of (4.33) and τ_2

```
#Real Part Analysis of 4.33
import matplotlib.pyplot as plt
import numpy as np

mh=0.04
kh=0.5

xl=0; xh=2; dx=(xh-xl)/1000
yl=-10; yh=5; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)

p1=np.sqrt((kh*np.exp(-x*(x+mh)))**2-(y+mh)**2)
eq1=y+mh+kh*np.exp(-x*(x+mh))*np.cos(x*p1)

plt.style.use('classic')
plt.rc('axes',labelsize=14); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.contour(x,y,eq1,[0],colors='blue',linewidths=2)
plt.xlabel(r'$\tau_2$ (year)'); plt.ylabel('Real Part')
plt.show()
```

Appendix 3 Relation of the imaginary part of the complex roots of (4.33) and τ_2

```
#Imaginary Part Analysis of 4.33
import matplotlib.pyplot as plt
import numpy as np

mh=0.04
kh=0.5

xl=0; xh=2; dx=(xh-xl)/1000
yl=-10; yh=10; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)

p2=y/np.tan(x*y)
eq2=y-kh*np.exp(x*p2)*np.sin(x*y)

plt.style.use('classic')
plt.rc('axes',labelsize=14); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.contour(x,y,eq2,[0],colors='blue',linewidths=2)
plt.xlabel(r'$\tau_2$ (year)'); plt.ylabel('Real Part')
plt.show()
```

Appendix 4 Relation of the roots of (4.34) and τ_2

```
import matplotlib.pyplot as plt
import numpy as np

md=0.04
kd=0.5

xl=-10; xh=0; dx=(xh-xl)/1000
yl=0; yh=1; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)
eq2=x+md+kd*np.exp(-y*(x+md))

plt.style.use('classic')
plt.rc('axes',labelsize=16); plt.rc(('xtick','ytick'),labelsize=14)
plt.rc('figure',fc='white')
plt.contour(x,y,eq2,[0],colors='red',linewidths=2)
plt.xlabel(r'$\lambda$'); plt.ylabel(r'$\tau_2$ (year)')
plt.show()
```

Appendix 5 Relation of the real part of the complex roots of (4.34) and τ_2

```
#Real Part Analysis of 4.34
import matplotlib.pyplot as plt
import numpy as np

md=0.06
kd=0.5

xl=0; xh=2; dx=(xh-xl)/1000
yl=-10; yh=0; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)

p3=np.sqrt((kd*np.exp(-x*(y+md)))**2-(y+md)**2)
eq3=y+md+kd*np.exp(-x*(y+md))*np.cos(x*p3)

plt.style.use('classic')
plt.rc('axes',labelsize=14); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.contour(x,y,eq3,[0],colors='blue',linewidths=2)
plt.xlabel(r'$\tau_2$ (year)'); plt.ylabel('Real Part')
plt.show()
```

Appendix 6 Relation of the imaginary part of the complex roots of (4.34) and τ_2

```
#Imaginary Part Analysis of 4.34
import matplotlib.pyplot as plt
import numpy as np

md=0.06
kd=0.5

xl=0; xh=2; dx=(xh-xl)/1000
yl=-10; yh=0; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)

p4=y/np.tan(x*y)
eq4=y-kd*np.exp(x*p4)*np.sin(x*y)

plt.style.use('classic')
plt.rc('axes',labelsize=14); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.contour(x,y,eq4,[0],colors='blue',linewidths=2)
plt.xlabel(r'$\tau_2$ (year)'); plt.ylabel('Imaginary Part')
plt.show()
```

Appendix 7 Relation of the roots of (4.77) and τ_2

```
import matplotlib.pyplot as plt
import numpy as np

Ah=2000; Ad=200
mh=0.04; md=0.06
ud=1
bh=0.001; bd=0.01
kh=0.25; kd=0.25
t1=0; t2=1/10
R0=bd*Ad*np.exp(-md*t1)/((md+ud)*(md+kd*np.exp(-md*t2)))

xl=-10; xh=0; dx=(xh-xl)/1000
yl=0; yh=1.75; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)
Ide=((md+kd*np.exp(-md*t2))/bd)*(R0-1)
eq3=x+mh+bh*Ide+kh*np.exp(-y*(x+mh))

plt.style.use('classic')
plt.rc('axes',labelsize=16); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.contour(x,y,eq3,[0],colors='blue',linewidths=2)
plt.xlabel(r'$\lambda$'); plt.ylabel(r'$\tau_2$ (year)')
plt.show()
```

Appendix 8 Relation of the real part of the complex roots of (4.77) and τ_2

```
#Real Part Analysis of 4.77
import matplotlib.pyplot as plt
import numpy as np

Ad=200
mh=0.04; md=0.06
ud=1
bh=0.001; bd=0.01
kh=0.25; kd=0.25
t1=0; t2=1/10

R0=bd*Ad*np.exp(-md*t1)/((md+ud)*(md+kd*np.exp(-md*t2)))
Ide=((md+kd*np.exp(-md*t2))/bd)*(R0-1)
B1=bh*Ide

xl=0; xh=2; dx=(xh-xl)/1000
yl=-10; yh=0; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)
```

```
u1=np.sqrt((kh*np.exp(-x*(y+mh)))**2-(y+mh+B1)**2)
eq5=y+mh+B1+kh*np.exp(-x*(y+mh))*np.cos(u1*x)

plt.style.use('classic')
plt.rc('axes',labelsize=14); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.contour(x,y,eq5,[0],colors='blue',linewidths=2)
plt.xlabel(r'$\tau_2$ (year)'); plt.ylabel('Real Part')
plt.show()
```

Appendix 9 Relation of the imaginary part of the complex roots of (4.77) and τ_2

```
#Imaginary Part Analysis of 4.77
import matplotlib.pyplot as plt
import numpy as np

Ad=200
mh=0.04; md=0.06
ud=1
bh=0.001; bd=0.01
kh=0.25; kd=0.25
t1=0; t2=1/10

R0=bd*Ad*np.exp(-md*t1)/((md+ud)*(md+kd*np.exp(-md*t2)))
Ide=((md+kd*np.exp(-md*t2))/bd)*(R0-1)
B1=bh*Ide

xl=0; xh=2; dx=(xh-xl)/1000
yl=-10; yh=10; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)

u2=y/np.tan(x*y)+B1
eq6=y+kh*np.exp(x*u2)*np.sin(x*y)

plt.style.use('classic')
plt.rc('axes',labelsize=14); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.contour(x,y,eq6,[0],colors='blue',linewidths=2)
plt.xlabel(r'$\tau_2$ (year)'); plt.ylabel('Imaginary Part')
plt.show()
```

Appendix 10 Determining the critical value of τ_2 based on equation (4.97)

```
#Importing necessary modules
import matplotlib.pyplot as plt
import numpy as np
```

```
Ad=200
md=0.06
ud=1
bd=0.01
kd=0.25
t1=0; t2=1/10
R0=bd*Ad*np.exp(-md*t1)/((md+ud)*(md+kd*np.exp(-md*t2)))

xl=0; xh=1; dx=(xh-xl)/1000
yl=0; yh=2; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)
eq4=(md+x*np.exp(-md*y))*R0**2-2*(md+ud)*(R0-1)-2*x*np.exp(-md*y)*R0

plt.style.use('classic')
plt.rc('axes',labelsize=14); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.contour(x,y,eq4,[0],colors='red',linewidths=2)
plt.xlabel(r'Dog vaccination rate ($k_D$) ($year^{-1}$)');
plt.ylabel(r'Vaccination latency ($\tau_2$) (year)')
plt.show()
```

Appendix 11 Determining the critical values of τ_1 and τ_2 based on equation (4.109)

```
import matplotlib.pyplot as plt
import numpy as np

Ad=200
md=0.06
bd=0.01
kd=0.25
ud=1
t1=1/6
t2=1/10
w=1
R0=bd*Ad*np.exp(-md*t1)/((md+ud)*(md+kd*np.exp(-md*t2)))

xl=0; xh=2; dx=(xh-xl)/1000
yl=0; yh=2; dy=(yh-yl)/1000
xRang=np.arange(xl,xh,dx)
yRang=np.arange(yl,yh,dy)
x,y=np.meshgrid(xRang,yRang)
r1=(md+ud)*np.sin(w*x); r2=w+(md+ud)*np.sin(w*x); r3=(md+ud)*(1-
np.cos(w*x))
r4=(md+ud)*(md+kd*np.exp(-md*y))*(R0-1); r5=md+kd*np.exp(-md*y)
s1=w**2+w*r1-md*r3-r4; s2=w*r5*R0+md*r1+w*r3+w*kd*np.exp(-md*y)
```

```
eq5=np.tan(w*y)-(r2*s1+r3*s2)/(r3*s1-r2*s2)

plt.style.use('classic')
plt.rc('axes',labelsize=14); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.contour(x,y,eq5,[0],colors='blue',linewidths=2)
plt.xlabel(r'Incubation period ($\tau_1$) (year)')
plt.ylabel(r'Vaccination latency ($\tau_2$) (year)')
plt.show()
```

Appendix 12 Program syntax for simulation of solutions around disease-free equilibrium

```
#Importing necessary modules
import numpy as np
from ddeint import ddeint
import matplotlib.pyplot as plt

Ah=2000 #Average birth of humans
Ad=200 #Average birth of puppies
mh=0.04 #Natural death rate of humans
md=0.06 #Natural death rate of dogs
uh=1 #Disease-related death rate of humans
ud=1 #Disease-related death rate of dogs
bh=0.0001 #Rate of transmission between susceptible humans and rabid
dogs
bd=0.001 #Rate of transmission between susceptible dogs and rabid dogs
kh=0.5 #Vaccination rate of humans
kd=0.5 #Vaccination rate of dogs

#Defining the model
def model2delays(x,t,t1,t2):
    Sh=x(t)[0]; Ih=x(t)[1]; Vh=x(t)[2]; Sd=x(t)[3]; Id=x(t)[4];
    Vd=x(t)[5]
    Sh1=x(t-t1)[0]; Ih1=x(t-t1)[1]; Vh1=x(t-t1)[2]; Sd1=x(t-t1)[3];
    Id1=x(t-t1)[4]; Vd1=x(t-t1)[5]
    Sh2=x(t-t2)[0]; Ih2=x(t-t2)[1]; Vh2=x(t-t2)[2]; Sd2=x(t-t2)[3];
    Id2=x(t-t2)[4]; Vd2=x(t-t2)[5]
    dSh=Ah-mh*Sh-bh*Sh*Id1*np.exp(-md*t1)-kh*Sh2*np.exp(-mh*t2)
    dIh=bh*Sh*Id1*np.exp(-md*t1)-(mh+uh)*Ih
    dVh=kh*Sh2*np.exp(-mh*t2)-mh*Vh
    dSd=Ad-md*Sd-bd*Sd*Id1*np.exp(-md*t1)-kd*Sd2*np.exp(-md*t2)
    dId=bd*Sd*Id1*np.exp(-md*t1)-(md+ud)*Id
    dVd=kd*Sd2*np.exp(-md*t2)-md*Vd
    return [dSh,dIh,dVh,dSd,dId,dVd]

psi=lambda t:[19500,500,0,1950,50,0] #Initial values and historical
functions
t1=1/6 #Incubation delay
```

```
t2=1/10 #Vaccination delay
tm=np.linspace(0,50,1000)
#Solving the delay system
solX=ddeint(model2delays,psi,tm,fargs=(t1,t2,))

#Plotting the results
plt.style.use('classic')
plt.rc('axes',labelsize=14); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.subplot(2,3,1)
plt.plot(tm,solX[:,0],linewidth=2)
plt.xlabel('Time (years)'); plt.ylabel(r'$S_H(t)$ (humans)')
plt.subplot(2,3,2)
plt.plot(tm,solX[:,1],linewidth=2,color='red')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_H(t)$ (humans)')
plt.subplot(2,3,3)
plt.plot(tm,solX[:,2],linewidth=2,color='green')
plt.xlabel('Time (years)'); plt.ylabel(r'$V_H(t)$ (humans)')
plt.subplot(2,3,4)
plt.plot(tm,solX[:,3],linewidth=2)
plt.xlabel('Time (years)'); plt.ylabel(r'$S_D(t)$ (dogs)')
plt.subplot(2,3,5)
plt.plot(tm,solX[:,4],linewidth=2,color='red')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_D(t)$ (dogs)')
plt.subplot(2,3,6)
plt.plot(tm,solX[:,5],linewidth=2,color='green')
plt.xlabel('Time (years)'); plt.ylabel(r'$V_D(t)$ (dogs)')

plt.show()
```

Appendix 13 Program syntax for simulation of solutions around endemic equilibrium

```
#Importing necessary modules
import numpy as np
from ddeint import ddeint
import matplotlib.pyplot as plt

Ah=2000 #Average birth of humans
Ad=200 #Average birth of puppies
mh=0.04 #Natural death rate of humans
md=0.06 #Natural death rate of dogs
uh=1 #Disease-related death rate of humans
ud=1 #Disease-related death rate of dogs
bh=0.001 #Rate of transmission between susceptible humans and rabid dogs
bd=0.01 #Rate of transmission between susceptible dogs and rabid dogs
kh=0.25 #Vaccination rate of humans
kd=0.25 #Vaccination rate of dogs
```

```
#Defining the model
def model2delays(x,t,t1,t2):
    Sh=x(t)[0]; Ih=x(t)[1]; Vh=x(t)[2]; Sd=x(t)[3]; Id=x(t)[4];
    Vd=x(t)[5]
    Sh1=x(t-t1)[0]; Ih1=x(t-t1)[1]; Vh1=x(t-t1)[2]; Sd1=x(t-t1)[3];
    Id1=x(t-t1)[4]; Vd1=x(t-t1)[5]
    Sh2=x(t-t2)[0]; Ih2=x(t-t2)[1]; Vh2=x(t-t2)[2]; Sd2=x(t-t2)[3];
    Id2=x(t-t2)[4]; Vd2=x(t-t2)[5]
    dSh=Ah-mh*Sh-bh*Sh*Id1*np.exp(-md*t1)-kh*Sh2*np.exp(-mh*t2)
    dIh=bh*Sh*Id1*np.exp(-md*t1)-(mh+uh)*Ih
    dVh=kh*Sh2*np.exp(-mh*t2)-mh*Vh
    dSd=Ad-md*Sd-bd*Sd*Id1*np.exp(-md*t1)-kd*Sd2*np.exp(-md*t2)
    dId=bd*Sd*Id1*np.exp(-md*t1)-(md+ud)*Id
    dVd=kd*Sd2*np.exp(-md*t2)-md*Vd
    return [dSh,dIh,dVh,dSd,dId,dVd]

psi=lambda t:[19800,200,0,1950,50,0] #Initial values and historical
functions
t1=1/6 #Incubation delay
t2=1/10 #Vaccination delay
tm=np.linspace(0,50,1000)
#Solving the delay system
solX=ddeint(model2delays,psi,tm,fargs=(t1,t2,))

#Plotting the results
plt.style.use('classic')
plt.rc('axes',labelsize=14); plt.rc(('xtick','ytick'),labelsize=14);
plt.rc('figure',fc='white')
plt.subplot(2,3,1)
plt.plot(tm,solX[:,0],linewidth=2)
plt.xlabel('Time (years)'); plt.ylabel(r'$S_H(t)$ (humans)')
plt.subplot(2,3,2)
plt.plot(tm,solX[:,1],linewidth=2,color='red')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_H(t)$ (humans)')
plt.subplot(2,3,3)
plt.plot(tm,solX[:,2],linewidth=2,color='green')
plt.xlabel('Time (years)'); plt.ylabel(r'$V_H(t)$ (humans)')
plt.subplot(2,3,4)
plt.plot(tm,solX[:,3],linewidth=2)
plt.xlabel('Time (years)'); plt.ylabel(r'$S_D(t)$ (dogs)')
plt.subplot(2,3,5)
plt.plot(tm,solX[:,4],linewidth=2,color='red')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_D(t)$ (dogs)')
plt.subplot(2,3,6)
plt.plot(tm,solX[:,5],linewidth=2,color='green')
plt.xlabel('Time (years)'); plt.ylabel(r'$V_D(t)$ (dogs)')

plt.show()
```

Appendix 14 Program syntax for effects of variation of k_D

```
#Importing necessary modules
import numpy as np
from ddeint import ddeint
import matplotlib.pyplot as plt

Ah=2000 #Average birth of humans
Ad=200 #Average birth of puppies
mh=0.04 #Natural death rate of humans
md=0.06 #Natural death rate of dogs
uh=1 #Disease-related death rate of humans
ud=1 #Disease-related death rate of dogs
bh=0.001 #Rate of transmission between susceptible humans and rabid dogs
bd=0.01 #Rate of transmission between susceptible dogs and rabid dogs
kh=0.25 #Vaccination rate of humans

#Defining the model
def model2delays(x,t,t1,t2,kd):
    Sh=x(t)[0]; Ih=x(t)[1]; Vh=x(t)[2]; Sd=x(t)[3]; Id=x(t)[4];
    Vd=x(t)[5]
    Sh1=x(t-t1)[0]; Ih1=x(t-t1)[1]; Vh1=x(t-t1)[2]; Sd1=x(t-t1)[3];
    Id1=x(t-t1)[4]; Vd1=x(t-t1)[5]
    Sh2=x(t-t2)[0]; Ih2=x(t-t2)[1]; Vh2=x(t-t2)[2]; Sd2=x(t-t2)[3];
    Id2=x(t-t2)[4]; Vd2=x(t-t2)[5]
    dSh=Ah-mh*Sh-bh*Sh*Id1*np.exp(-md*t1)-kh*Sh2*np.exp(-mh*t2)
    dIh=bh*Sh*Id1*np.exp(-md*t1)-(mh+uh)*Ih
    dVh=kh*Sh2*np.exp(-mh*t2)-mh*Vh
    dSd=Ad-md*Sd-bd*Sd*Id1*np.exp(-md*t1)-kd*Sd2*np.exp(-md*t2)
    dId=bd*Sd*Id1*np.exp(-md*t1)-(md+ud)*Id
    dVd=kd*Sd2*np.exp(-md*t2)-md*Vd
    return [dSh,dIh,dVh,dSd,dId,dVd]

psi=lambda t:[19800,200,0,1950,50,0] #Initial values and historical
functions
t1=1/6 #Incubation delay
t2=1/10 #Vaccination delay
tm=np.linspace(0,10,1000)
#Variations of kd
kd1=0.25; kd2=0.5; kd3=0.75; kd4=1; kd5=1.5; kd6=2
#Solving the delay system with variations of kd
solX1=ddeint(model2delays,psi,tm,fargs=(t1,t2,kd1,))
solX2=ddeint(model2delays,psi,tm,fargs=(t1,t2,kd2,))
solX3=ddeint(model2delays,psi,tm,fargs=(t1,t2,kd3,))
solX4=ddeint(model2delays,psi,tm,fargs=(t1,t2,kd4,))
solX5=ddeint(model2delays,psi,tm,fargs=(t1,t2,kd5,))
solX6=ddeint(model2delays,psi,tm,fargs=(t1,t2,kd6,))

#Plotting the result
plt.style.use('classic')
```

```
plt.rc('font',size=14); plt.rc('figure',fc='white')
plt.subplot(2,1,1)
plt.title("Effects of Dog Vaccination on Infected Humans")
plt.plot(tm,solX1[:,1],linewidth=2,label=r'$k_D=0.25$')
plt.plot(tm,solX2[:,1],linewidth=2,label=r'$k_D=0.5$')
plt.plot(tm,solX3[:,1],linewidth=2,label=r'$k_D=0.75$')
plt.plot(tm,solX4[:,1],linewidth=2,label=r'$k_D=1$')
plt.plot(tm,solX5[:,1],linewidth=2,label=r'$k_D=1.5$')
plt.plot(tm,solX6[:,1],linewidth=2,label=r'$k_D=2$')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_H(t)$ (humans)')
plt.legend()

plt.subplot(2,1,2)
plt.title("Effects of Dog Vaccination on Infected Dogs")
plt.plot(tm,solX1[:,4],linewidth=2,label=r'$k_D=0.25$')
plt.plot(tm,solX2[:,4],linewidth=2,label=r'$k_D=0.5$')
plt.plot(tm,solX3[:,4],linewidth=2,label=r'$k_D=0.75$')
plt.plot(tm,solX4[:,4],linewidth=2,label=r'$k_D=1$')
plt.plot(tm,solX5[:,4],linewidth=2,label=r'$k_D=1.5$')
plt.plot(tm,solX6[:,4],linewidth=2,label=r'$k_D=2$')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_D(t)$ (dogs)')
plt.legend()
plt.show()
```

Appendix 15 Program syntax for effects of variation of A_D

```
#Importing necessary modules
import numpy as np
from ddeint import ddeint
import matplotlib.pyplot as plt

Ah=2000 #Average birth of humans
mh=0.04 #Natural death rate of humans
md=0.06 #Natural death rate of dogs
uh=1 #Disease-related death rate of humans
ud=1 #Disease-related death rate of dogs
bh=0.001 #Rate of transmission between susceptible humans and rabid dogs
bd=0.01 #Rate of transmission between susceptible dogs and rabid dogs
kh=0.25 #Vaccination rate of humans

#Defining the model
def model2delays(x,t,t1,t2,Ad):
Sh=x(t)[0]; Ih=x(t)[1]; Vh=x(t)[2]; Sd=x(t)[3]; Id=x(t)[4];
Vd=x(t)[5]
Sh1=x(t-t1)[0]; Ih1=x(t-t1)[1]; Vh1=x(t-t1)[2]; Sd1=x(t-t1)[3];
Id1=x(t-t1)[4]; Vd1=x(t-t1)[5]
Sh2=x(t-t2)[0]; Ih2=x(t-t2)[1]; Vh2=x(t-t2)[2]; Sd2=x(t-t2)[3];
Id2=x(t-t2)[4]; Vd2=x(t-t2)[5]
dSh=Ah-mh*Sh-bh*Sh*Id1*np.exp(-md*t1)-kh*Sh2*np.exp(-mh*t2)
```

```
    dIh=bh*Sh*Id1*np.exp(-md*t1)-(mh+uh)*Ih
    dVh=kh*Sh2*np.exp(-mh*t2)-mh*Vh
    dSd=Ad-md*Sd-bd*Sd*Id1*np.exp(-md*t1)-kd*Sd2*np.exp(-md*t2)
    dId=bd*Sd*Id1*np.exp(-md*t1)-(md+ud)*Id
    dVd=kd*Sd2*np.exp(-md*t2)-md*Vd
    return [dSh,dIh,dVh,dSd,dId,dVd]

psi=lambda t:[19800,200,0,1950,50,0] #Initial values and historical
functions
t1=1/6 #Incubation delay
t2=1/10 #Vaccination delay
tm=np.linspace(0,10,1000)
#Variations of Ad
Ad1=300; Ad2=200; Ad3=150; Ad4=100; Ad5=50; Ad6=30
#Solving the delay system with variations of Ad
solX1=ddeint(model2delays,psi,tm,fargs=(t1,t2,Ad1,))
solX2=ddeint(model2delays,psi,tm,fargs=(t1,t2,Ad2,))
solX3=ddeint(model2delays,psi,tm,fargs=(t1,t2,Ad3,))
solX4=ddeint(model2delays,psi,tm,fargs=(t1,t2,Ad4,))
solX5=ddeint(model2delays,psi,tm,fargs=(t1,t2,Ad5,))
solX6=ddeint(model2delays,psi,tm,fargs=(t1,t2,Ad6,))

#Plotting the result
plt.style.use('classic')
plt.rc('font',size=14); plt.rc('figure',fc='white')

plt.subplot(2,1,1)
plt.title("Effects of Puppies' Birth on Infected Humans")
plt.plot(tm,solX1[:,1],linewidth=2,label=r'$A_D=300$')
plt.plot(tm,solX2[:,1],linewidth=2,label=r'$A_D=200$')
plt.plot(tm,solX3[:,1],linewidth=2,label=r'$A_D=150$')
plt.plot(tm,solX4[:,1],linewidth=2,label=r'$A_D=100$')
plt.plot(tm,solX5[:,1],linewidth=2,label=r'$A_D=50$')
plt.plot(tm,solX6[:,1],linewidth=2,label=r'$A_D=30$')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_H(t)$ (humans)')
plt.legend()

plt.subplot(2,1,2)
plt.title("Effects of Puppies' Birth on Infected Dogs")
plt.plot(tm,solX1[:,4],linewidth=2,label=r'$A_D=300$')
plt.plot(tm,solX2[:,4],linewidth=2,label=r'$A_D=200$')
plt.plot(tm,solX3[:,4],linewidth=2,label=r'$A_D=150$')
plt.plot(tm,solX4[:,4],linewidth=2,label=r'$A_D=100$')
plt.plot(tm,solX5[:,4],linewidth=2,label=r'$A_D=50$')
plt.plot(tm,solX6[:,4],linewidth=2,label=r'$A_D=30$')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_D(t)$ (dogs)')
plt.legend()
plt.show()
```

Appendix 16 Program syntax for effects of the variation of τ_1

```
#Importing necessary modules
import numpy as np
from ddeint import ddeint
import matplotlib.pyplot as plt

Ah=2000 #Average birth of humans
Ad=200 #Average birth of puppies
mh=0.04 #Natural death rate of humans
md=0.06 #Natural death rate of dogs
uh=1 #Disease-related death rate of humans
ud=1 #Disease-related death rate of dogs
bh=0.001 #Rate of transmission between susceptible humans and rabid dogs
bd=0.01 #Rate of transmission between susceptible dogs and rabid dogs
kh=0.25 #Vaccination rate of humans
kd=0.25 #Vaccination rate of dogs

#Defining the model
def model2delays(x,t,t1,t2):
    Sh=x(t)[0]; Ih=x(t)[1]; Vh=x(t)[2]; Sd=x(t)[3]; Id=x(t)[4];
    Vd=x(t)[5]
    Sh1=x(t-t1)[0]; Ih1=x(t-t1)[1]; Vh1=x(t-t1)[2]; Sd1=x(t-t1)[3];
    Id1=x(t-t1)[4]; Vd1=x(t-t1)[5]
    Sh2=x(t-t2)[0]; Ih2=x(t-t2)[1]; Vh2=x(t-t2)[2]; Sd2=x(t-t2)[3];
    Id2=x(t-t2)[4]; Vd2=x(t-t2)[5]
    dSh=Ah-mh*Sh-bh*Sh*Id1*np.exp(-md*t1)-kh*Sh2*np.exp(-mh*t2)
    dIh=bh*Sh*Id1*np.exp(-md*t1)-(mh+uh)*Ih
    dVh=kh*Sh2*np.exp(-mh*t2)-mh*Vh
    dSd=Ad-md*Sd-bd*Sd*Id1*np.exp(-md*t1)-kd*Sd2*np.exp(-md*t2)
    dId=bd*Sd*Id1*np.exp(-md*t1)-(md+ud)*Id
    dVd=kd*Sd2*np.exp(-md*t2)-md*Vd
    return [dSh,dIh,dVh,dSd,dId,dVd]

psi=lambda t:[19800,200,0,1950,50,0] #Initial values and historical
functions
t2=1.571 #Vaccination delay
tm=np.linspace(0,10,1000)
#Variations of tau1
t11=0.16; t12=0.32; t13=0.5; t14=0.67; t15=0.83; t16=1
#Solving the delay system with variations of tau1
solX1=ddeint(model2delays,psi,tm,fargs=(t11,t2,))
solX2=ddeint(model2delays,psi,tm,fargs=(t12,t2,))
solX3=ddeint(model2delays,psi,tm,fargs=(t13,t2,))
solX4=ddeint(model2delays,psi,tm,fargs=(t14,t2,))
solX5=ddeint(model2delays,psi,tm,fargs=(t15,t2,))
solX6=ddeint(model2delays,psi,tm,fargs=(t16,t2,))

#Plotting the result
plt.style.use('classic')
```

```
plt.rc('font',size=14); plt.rc('figure',fc='white')
plt.subplot(2,1,1)
plt.title("Effects of Incubation Period on Infected Humans")
plt.plot(tm,solX1[:,1],linewidth=2,label=r'$\tau_1=0.16$')
plt.plot(tm,solX2[:,1],linewidth=2,label=r'$\tau_1=0.32$')
plt.plot(tm,solX3[:,1],linewidth=2,label=r'$\tau_1=0.5$')
plt.plot(tm,solX4[:,1],linewidth=2,label=r'$\tau_1=0.67$')
plt.plot(tm,solX5[:,1],linewidth=2,label=r'$\tau_1=0.83$')
plt.plot(tm,solX6[:,1],linewidth=2,label=r'$\tau_1=1$')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_H(t)$ (humans)')
plt.legend()

plt.subplot(2,1,2)
plt.title("Effects of Incubation Period on Infected Dogs")
plt.plot(tm,solX1[:,4],linewidth=2,label=r'$\tau_1=0.16$')
plt.plot(tm,solX2[:,4],linewidth=2,label=r'$\tau_1=0.32$')
plt.plot(tm,solX3[:,4],linewidth=2,label=r'$\tau_1=0.5$')
plt.plot(tm,solX4[:,4],linewidth=2,label=r'$\tau_1=0.67$')
plt.plot(tm,solX5[:,4],linewidth=2,label=r'$\tau_1=0.83$')
plt.plot(tm,solX6[:,4],linewidth=2,label=r'$\tau_1=1$')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_D(t)$ (dogs)')
plt.legend()
plt.show()
```

Appendix 17 Program syntax for effects of the variation of τ_2

```
#Importing necessary modules
import numpy as np
from ddeint import ddeint
import matplotlib.pyplot as plt

Ah=2000 #Average birth of humans
Ad=200 #Average birth of puppies
mh=0.04 #Natural death rate of humans
md=0.06 #Natural death rate of dogs
uh=1 #Disease-related death rate of humans
ud=1 #Disease-related death rate of dogs
bh=0.001 #Rate of transmission between susceptible humans and rabid dogs
bd=0.01 #Rate of transmission between susceptible dogs and rabid dogs
kh=0.25 #Vaccination rate of humans
kd=0.25 #Vaccination rate of dogs

#Defining the model
def model2delays(x,t,t1,t2):
    Sh=x(t)[0]; Ih=x(t)[1]; Vh=x(t)[2]; Sd=x(t)[3]; Id=x(t)[4];
    Vd=x(t)[5]
    Sh1=x(t-t1)[0]; Ih1=x(t-t1)[1]; Vh1=x(t-t1)[2]; Sd1=x(t-t1)[3];
    Id1=x(t-t1)[4]; Vd1=x(t-t1)[5]
```

```
Sh2=x(t-t2)[0]; Ih2=x(t-t2)[1]; Vh2=x(t-t2)[2]; Sd2=x(t-t2)[3];
Id2=x(t-t2)[4]; Vd2=x(t-t2)[5]
dSh=Ah-mh*Sh-bh*Sh*Id1*np.exp(-md*t1)-kh*Sh2*np.exp(-mh*t2)
dIh=bh*Sh*Id1*np.exp(-md*t1)-(mh+uh)*Ih
dVh=kh*Sh2*np.exp(-mh*t2)-mh*Vh
dSd=Ad-md*Sd-bd*Sd*Id1*np.exp(-md*t1)-kd*Sd2*np.exp(-md*t2)
dId=bd*Sd*Id1*np.exp(-md*t1)-(md+ud)*Id
dVd=kd*Sd2*np.exp(-md*t2)-md*Vd
return [dSh,dIh,dVh,dSd,dId,dVd]

psi=lambda t:[19800,200,0,1950,50,0] #Initial values and historical
functions
t1=1 #Incubation delay
tm=np.linspace(0,20,1000)
#Variations of tau2
t21=1.57; t22=0.79; t23=0.4; t24=0.31; t25=0.16; t26=0.07
#Solving the delay system with variations of tau2
solX1=ddeint(model2delays,psi,tm,fargs=(t1,t21,))
solX2=ddeint(model2delays,psi,tm,fargs=(t1,t22,))
solX3=ddeint(model2delays,psi,tm,fargs=(t1,t23,))
solX4=ddeint(model2delays,psi,tm,fargs=(t1,t24,))
solX5=ddeint(model2delays,psi,tm,fargs=(t1,t25,))
solX6=ddeint(model2delays,psi,tm,fargs=(t1,t26,))

#Plotting the result
plt.style.use('classic')
plt.rc('font',size=14); plt.rc('figure',fc='white')
plt.subplot(2,1,1)
plt.title("Effects of Vaccination Latency on Infected Humans")
plt.plot(tm,solX1[:,1],linewidth=2,label=r'$\tau_2=1.57$')
plt.plot(tm,solX2[:,1],linewidth=2,label=r'$\tau_2=0.79$')
plt.plot(tm,solX3[:,1],linewidth=2,label=r'$\tau_2=0.4$')
plt.plot(tm,solX4[:,1],linewidth=2,label=r'$\tau_2=0.31$')
plt.plot(tm,solX5[:,1],linewidth=2,label=r'$\tau_2=0.16$')
plt.plot(tm,solX6[:,1],linewidth=2,label=r'$\tau_2=0.07$')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_H(t)$ (humans)')
plt.legend()

plt.subplot(2,1,2)
plt.title("Effects of Vaccination Latency on Infected Dogs")
plt.plot(tm,solX1[:,4],linewidth=2,label=r'$\tau_2=1.57$')
plt.plot(tm,solX2[:,4],linewidth=2,label=r'$\tau_2=0.79$')
plt.plot(tm,solX3[:,4],linewidth=2,label=r'$\tau_2=0.4$')
plt.plot(tm,solX4[:,4],linewidth=2,label=r'$\tau_2=0.31$')
plt.plot(tm,solX5[:,4],linewidth=2,label=r'$\tau_2=0.16$')
plt.plot(tm,solX6[:,4],linewidth=2,label=r'$\tau_2=0.07$')
plt.xlabel('Time (years)'); plt.ylabel(r'$I_D(t)$ (dogs)')
plt.legend()
plt.show()
```