

MATLAB PROGRAMS

1) Time series solution: shifted unit step function

```
% S is the number of Susceptible individuals
% I is the number of Infected individuals
% R is the number of Recovered individuals
clear all
close all
% %%-- initial conditions
S0=0.99;% initial condition for S
I0=0.01;% initial condition for I
N0=S0+I0;% initial population
S=[];
I=[];
R=[];
rho=[];
S(1)=S0;
I(1)=I0;
% %%--
% %%-- parameters
v=0.05;% vaccine rate
mu=1/(59*365);% natural death rate
Lambda=1/(59*365); % birth rate
beta=0.2;% infectious rate
alpha=0.0332;% recover rate
gamma=0.0135;% disease induced death rate
% % --
Lmax=100;% maximum delay
Tmax=200; %maximum time
maxt=20000;% numebr of steps for time
dt=Tmax/maxt;%
n=1000;% number of steps for delay
dx=Lmax/n; % time scince recovery step size
b=dt/(2*dx);% factor in the PDE that needs to be less than 1
rho1=0.03;% rate of losing immunity
% %---Define the unite step losing immunity rho(u)=rho u(tau-50)---
% %---Note that delay T=500/10=50
    for i=1:n+1;
if i<500
    rho(i)=0;
else
    rho(i)=rho1;
end
    end
%---- Initial values
S(1)=S0;
I(1)=I0;
%--delay-stepping loop defining initial condition
    for i=1:n+1;
R(1,i)=0;% initial condition for R
%ss1=ss1+rho(i).*R(1,i);
    end
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ---intm is a matrix to store values for dx*rho*R and together with ss will
be
% used to estimate the integral for rho*R
intm=[];
intm(1)= 0;
ss=0;
%Forward in time t, centered in time since recover tau, dx instead of dtau
for k=1:maxt;%time-stepping loop
    S(k+1)=S(k)+dt.*(Lambda-beta*S(k).*I(k)-(v+mu)*S(k)+intm(k));
    I(k+1)=I(k)+dt.*(beta*S(k).*I(k)-(alpha+gamma+mu)*I(k));
    R(k+1,1)=v*S(k+1)+alpha*I(k+1);
    ss=0;
    ss=dx*rho(1).*R(k+1,1);

    for i=2:n;%delay or tau-stepping loop

        R(k+1,i)=(1-dt.*rho(i)-dt.*mu).*0.5.*(R(k,i+1)+R(k,i-1))-
b.*(R(k,i+1)-R(k,i-1));
        ss=ss+dx*rho(i).*R(k+1,i);

    end

    intm(k+1)=ss;
end
%delay-stepping loop to evaluate  $Q=\int_0^{\infty} e^{-\mu \tau} P(\tau) d\tau$ .
Q=0;
for i=1:n+1;
    srho=0;
    for jj=1:i;%delay-stepping internal loop to evaluate the integral
P(\tau)
        srho=srho+dx*rho(jj);
    end
    Q=Q+ dx*exp(-mu*i)* exp(-srho);
    x(i)=i*dx; % needed for plotting R
end
for k=1:maxt+1;
    tt(k)=(k-1)*dt;
end
%--R_0 for the SIRS PDE model---
disp('For the PDEs R_0=')
R_0=(beta*Lambda)/(mu*(alpha+gamma+mu)*(1+v*Q))
%---EE for the SIRS PDE model-----
disp('For the PDEs the EE is (S,I,R)=')
Se=(alpha+gamma+mu)/beta
Ie=(mu*Se*(1+v*Q)*(R_0-1))./(gamma+mu+mu*alpha*Q)
Re=(v*Se+alpha*Ie)*Q
disp('For the PDEs the DFE is (S,I,R)=')
S00=Lambda/(mu*(1+v*Q))
I00=0
R00=(v*S00+alpha*I00)*Q
%----- plots from PDEs-----
figure(1)
plot(tt(1,:),I(1,:), '-','linewidth',2.5)
hold on

```

```

figure(2)
plot(tt(1,:),S(1,:), '-','linewidth',2.5)
hold on
%2d-graph for R
figure(3)
mesh(tt,x,R')
xlabel('Time (days)');
ylabel('Time since recovery (days)');
zlabel('Recovered')
set(gca,'FontSize',14);
% -----solving the classical ODE SIRS model-----
% S1 is the number of Susceptible individuals
% I1 is the number of Infected individuals
% R1 is the number of Recovered individuals
S1=[];
I1=[];
R1=[];
S1(1)=S0;
I1(1)=I0;
R1(1)=0;
for k=1:maxt;
    S1(k+1)=S1(k)+dt.*(Lambda-beta*S1(k).*I1(k)-(v+mu)*S1(k)+
rho1*R1(k));
    I1(k+1)=I1(k)+dt.*(beta*S1(k).*I1(k)-(alpha+gamma+mu)*I1(k));
    R1(k+1)=R1(k)+dt.*(v*S1(k)+alpha*I1(k)-(rho1+mu)*R1(k));
end

for k=1:maxt+1;
    tt(k)=(k-1)*dt;
end
%--R_0 for the SIRS ODEs model---
disp('For the ODEs R_0=')
R_01=beta*Lambda*(mu+rho1)/(mu*(alpha+gamma+mu)*(mu+rho1+v))
% note that % Q=1/(mu+rho1) in this classical case;
%--EE for the SIRS ODEs model---
disp('For the ODEs the EE is (S,I,R)=')
Q1=1/(mu+rho1);
Se1=(alpha+gamma+mu)/beta
Ie1=(mu*Se1*(1+v*Q1)*(R_01-1))./(gamma+mu+mu*alpha*Q1)
Re1=(v*Se1+alpha*Ie1)*Q1
%--DFE for the SIRS ODEs model---
disp('For the ODEs the DFE is (S,I,R)=')
S001=Lambda/(mu*(1+v*Q1))
I001=0
R001=(v*S001+alpha*I001)*Q1
%----- plots from form SIRS ODEs-----
figure(1)
plot(tt(1,:),I1(1,:), ':','linewidth',2.5)
ylabel('Infected','FontSize', 14);
xlabel('Time(days)', 'FontSize', 14);
legend('\rho(\tau)=0.03 u(t-50)', '\rho(\tau)=0.03')
set(gca,'FontSize',14);

figure(2)
plot(tt(1,:),S1(1,:), ':','linewidth',2.5)
ylabel('Susceptible','FontSize', 14);
xlabel('Time(days)', 'FontSize', 14);

```

```

legend('\rho(\tau)=0.03 u(t-50)', '\rho(\tau)=0.03')
set(gca, 'FontSize', 14);
%----- plots from int_0^100 R dtau from PDEs and R from the classical ODEs-
-----
figure(4)
plot(tt(1,:), sum((dx*R(:, :))'), 'linewidth', 2.5)
    hold on
plot(tt(1,:), R1(1,:), ':', 'linewidth', 2.5)

    ylabel('Recovered', 'FontSize', 14);
xlabel('Time(days)', 'FontSize', 14);
legend('\rho(\tau)=0.03 u(t-50)', '\rho(\tau)=0.03')
set(gca, 'FontSize', 14);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

2) SI-Phase plane for shift losing immunity rate

```

clear all
close all
% %-- initial conditions
S0=0.99;% initial condition for S
I0=0.01;% initial condition for I
N0=S0+I0;% initial population
S=[];
I=[];
R=[];
rho=[];
S(1)=S0;
I(1)=I0;
% %--
% %-- parameters
v=0*0.05;% vaccine rate
mu=1/(59*365);% natural death rate
Lambda=1/(59*365); % birth rate
beta=0.2;% infectious rate
alpha=0.0332;% recover rate
gamma=0.0135;% disease induced death rate
% % --
Lmax=100;% maximum delay
Tmax=200;% maximum time
maxt=20000;% numebr of steps for time
dt=Tmax/maxt;%
n=1000;% number of steps for delay
dx=Lmax/n; % time scince recovery step size
b=dt/(2*dx);% factor in the PDE that needs to be less than 1
rho1=0.03;% rate of losing immunity
for kk=[250 500 750];
    T=kk/10;
    for i=1:n+1;
        if i<=kk
            rho(i)=0;
        else

```

```

        rho(i)=rho1;
    end
    end
    S(1)=S0;
    I(1)=I0;
    for i=1:n+1;
        R(1,i)=0;
    end
    intm=[];
    intm(1)= 0;
    ss=0;

    for k=1:maxt;
        S(k+1)=S(k)+dt.*(Lambda-beta*S(k).*I(k)-(v+mu)*S(k)+intm(k));
        I(k+1)=I(k)+dt.*(beta*S(k).*I(k)-(alpha+gamma+mu)*I(k));
        R(k+1,1)=v*S(k)+alpha*I(k);
        ss=0;
        ss=dx*rho(1).*R(k+1,1);

        for i=2:n;
            R(k+1,i)=(1-dt.*rho(i)-dt.*mu).*0.5.*(R(k,i+1)+R(k,i-1))-
b*(R(k,i+1)-R(k,i-1));
            ss=ss+dx*rho(i).*R(k+1,i);

        end

        intm(k+1)=ss;
    end
    if T==25;
    plot(S(1,:),I(1:,:),'-k','linewidth',2.5,'DisplayName', sprintf('T=
%d',T))
    hold on
    elseif T==50;
        plot(S(1,:),I(1:),'--b','linewidth',2.5,'DisplayName',
sprintf('T= %d',T))
        hold on
    else
        plot(S(1,:),I(1:),'-.r','linewidth',2.5,'DisplayName',
sprintf('T= %d',T))
        hold on
    end
end
end
xlabel('S(t)','FontSize', 14);
ylabel('I(t)','FontSize', 14);

%---the classical SIRS model
S1=[];
I1=[];
R1=[];

```

```

    S1(1)=S0;
    I1(1)=I0;
    R1(1)=0;
    for k=1:maxt;
        S1(k+1)=S1(k)+dt.*(Lambda-beta*S1(k).*I1(k)-(v+mu)*S1(k)+
        rho1*R1(k));
        I1(k+1)=I1(k)+dt.*(beta*S1(k).*I1(k)-
        (alpha+gamma+mu)*I1(k));
        R1(k+1)=R1(k)+dt.*(v*S1(k)+alpha*I1(k)- rho1*R1(k));
    end

    for k=1:maxt+1;
        tt(k)=(k-1)*dt;
    end
    figure(1)
    plot(S1(1,:),I1(1,:),':m','linewidth',2.5,'DisplayName', sprintf('T=
    0'));

```