Optimization in Systems and Control (SC42055)

Non Linear optimization Assignment

Authors:	*Student Number:*
Kwabena Ofori	4031601
Koen Bakker	4389018

October 31, 2018

Contents

1 | Non linear programming assignment

In this assignment the traffic model METANET is used to describe the management of the density and velocity of vehicles on the highway and entering the highway. By using a speed limit and an ramp metering, traffic queing is managed.

1.1 State Space formulation

The METANET can be formulated as a state space system by defining x as:

$$x(k) = [\rho_1(k), \rho_2(k), \rho_3(k), \rho_4(k), v_1(k), v_2(k), v_3(k), v_4(k), W_r(k)] \tag{1.1}$$

Where $\rho_i(k)$ is the trafic density of each segment, $v_i(k)$ is the mean speed of each segment and $w_r(k$ is the number of vehicles waiting on the on-ramp. Consider the following figure

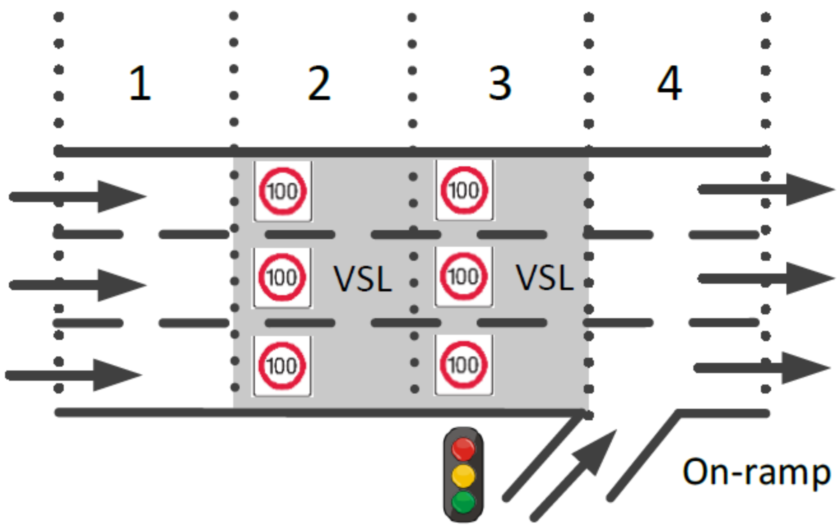

Figure 1.1: METANET

The dynamics of the METANET can be expressed in a state space as

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \\ x_5(k+1) \\ x_6(k+1) \\ x_7(k+1) \\ x_8(k+1) \\ x_9(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) + \frac{T}{\lambda_1 L_1}(\lambda(_0(k)x_5(k) - x_1(k)x_5(k)) \\ x_2(k) + \frac{T}{\lambda_2 L_2}(\lambda(_1(k)x_5(k) - x_2(k)x_6(k)) \\ x_3(k) + \frac{T}{\lambda_3 L_3}(\lambda(2(k)x_6(k) - x_3(k)x_7(k)) \\ x_4(k) + \frac{T}{\lambda_4 L_4}(\lambda(3(k)x_7(k) - x_4(k)x_8(k)) \\ x_5(k) + \frac{T}{\tau}(vfe^{\frac{-1}{a}\frac{x_1(k)}{\rho_c}^{a}} - x_5(k)) + \frac{T}{L}x_5(k)(x_5(k) - x_5(k)) - \frac{\mu T}{\tau L}\frac{x_2(k)-x_1(k)}{x_1(k)+K} \\ x_6(k) - \frac{T}{\tau}x_6(k) + \frac{T}{L}x_6(k)(x_5(k) - x_6(k)) - \frac{\mu T}{\tau L}\frac{x_3(k)-x_2(k)}{x_2(k)+K} \\ x_7(k) - \frac{T}{\tau}x_7(k) + \frac{T}{L}x_7(k)(x_6(k) - x_7(k)) - \frac{\mu T}{\tau L}\frac{x_4(k)-x_3(k)}{x_3(k)+K} \\ x_8(k) + \frac{T}{\tau}(vfe^{\frac{-1}{a}\frac{x_4(k)}{\rho_c}^{a}} - x_8(k) + \frac{T}{L}x_8(k)(x_7(k) - x_8(k)) - \frac{\mu T}{\tau L}\frac{x_5(k)-x_4(k)}{x_4(k)+K} \\ x_9(k) \end{bmatrix} + \begin{bmatrix} q_0 + q_{r,1}(k) \\ 0 \\ 0 \\ r(k)C_r \\ 0 \\ (1+\alpha)V_{SL}(k) \\ (1+\alpha)V_{SL}(k) \\ 0 \\ T(D_r(k) - q_r(k)) \end{bmatrix}$$

With the values of parameters $\tau, \mu, C_r, \rho_m, \alpha, K, a, v_f, \rho_c$ depicted in table 1.1:
The output Total time spent can be expressed as

$$y(k) = \left[Tx_9(k) + TL\lambda(x_1(k) + x_2(k) + x_3(k) + x_4(k))\right] \tag{1.2}$$

τ	μ	C_r	ρ	α	K	a	v_f	ρ_c
10/3600 h	80 km/h	2000 veh/hr	120	0.1	10	2	110 km/h	28

Table 1.1: parameters

with L the lenght of each segment, which is one kilometer, and T the simulation timestep which is 10 seconds.

1.2 Minimize Total Time Spent

The minimization problem of the Total Time Spent can be expressed as follows:

$$y(k) = \left[Tx_9(k) + TL\lambda(x_1(k) + x_2(k) + x_3(k) + x_4(k))\right] \tag{1.3}$$

Where:

$$TTS = \sum_{1}^{k} y(k) \tag{1.4}$$

So:

$$\min TTS = \min \sum_{1}^{k} \left[Tx_9(k) + TL\lambda(x_1(k) + x_2(k) + x_3(k) + x_4(k))\right] \tag{1.5}$$

It is a nonlinear objective function, that is only constraint by lower and upper bounds for V_{SL}. fmincon, a built-in Matlab function, is able to solve this type of optimization problem.

1.3 Different starting points

When V_{SL} = 120km/h is the starting point, the V_{SL} will remain constant for the entire duration at 119km/h. When V_{SL} = 60km/h, the average V_{SL} will also be around 119km/h, but not as a constant.

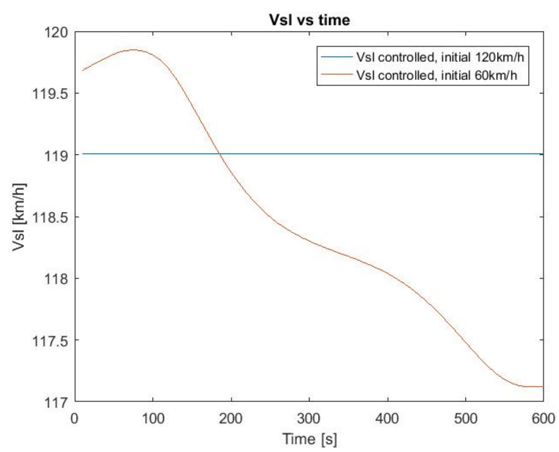

Figure 1.2: V_{SL} plot for different initial points

It is not desired that the starting point of the algorithm has this kind of influence on the outcome. To prevent this a multi-start algorithm can be used, like particle swarm optimization or a genetic algorithm. To improve our solution we will use the genetic algorithm, since it lies within the scope of the course.

1.4 Multiple control measures

The optimization problem is now a genetic algorithm with 120 variables: 60 for the V_{SL} values and 60 for the ramp metering values. The V_{SL} and ramp metering rate over time are:

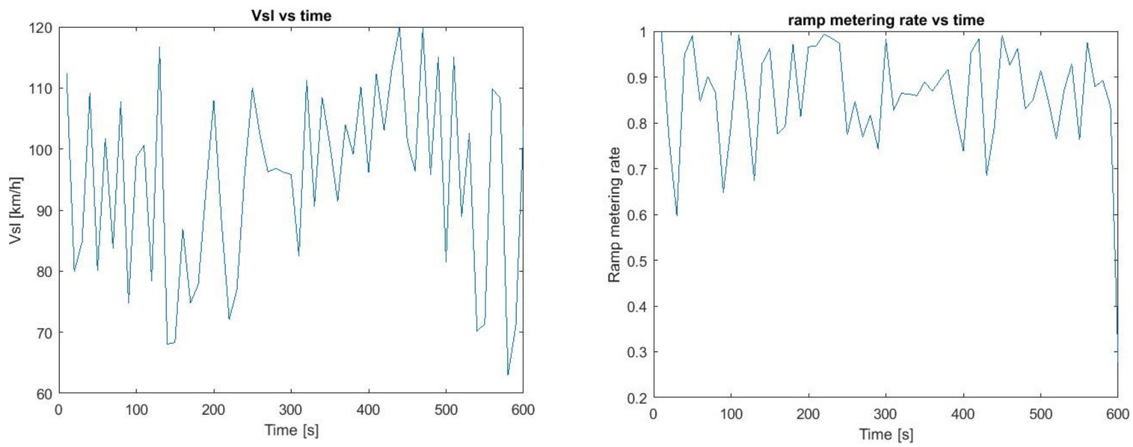

Figure 1.3: V_SL, $r(k)$ plot

These figures almost suggest that it does not matter what the values of the V_{SL} and the ramp metering are. This is partially true.
The reason the value for ramp metering can be anything, is because it appears that there is no queue formed on the ramp in our model. This is not what you would expect based on the assignment, so it is probably caused by a programming error. But without a queue, the ramp metering can be anything.
For the V_{SL} it is only important that:

$$(1+\alpha)V_{SL,i}(k) > v_f e^{-\frac{1}{a}(\frac{\rho_i(k)}{\rho_c})^a} \tag{1.6}$$

As long as this is the case, it does not matter what V_{SL} is. There is still somewhat of a trend visible in V_{SL} that supports this claim.
The extra constraint is implemented with the use of a penalty function, whenever there are more than 15.5 (20-E_3) cars waiting, a relatively very large number will be added to the objection function.

1.5 Results

Plots of the queues, the speed limits and the ramp metering rate are as followed:

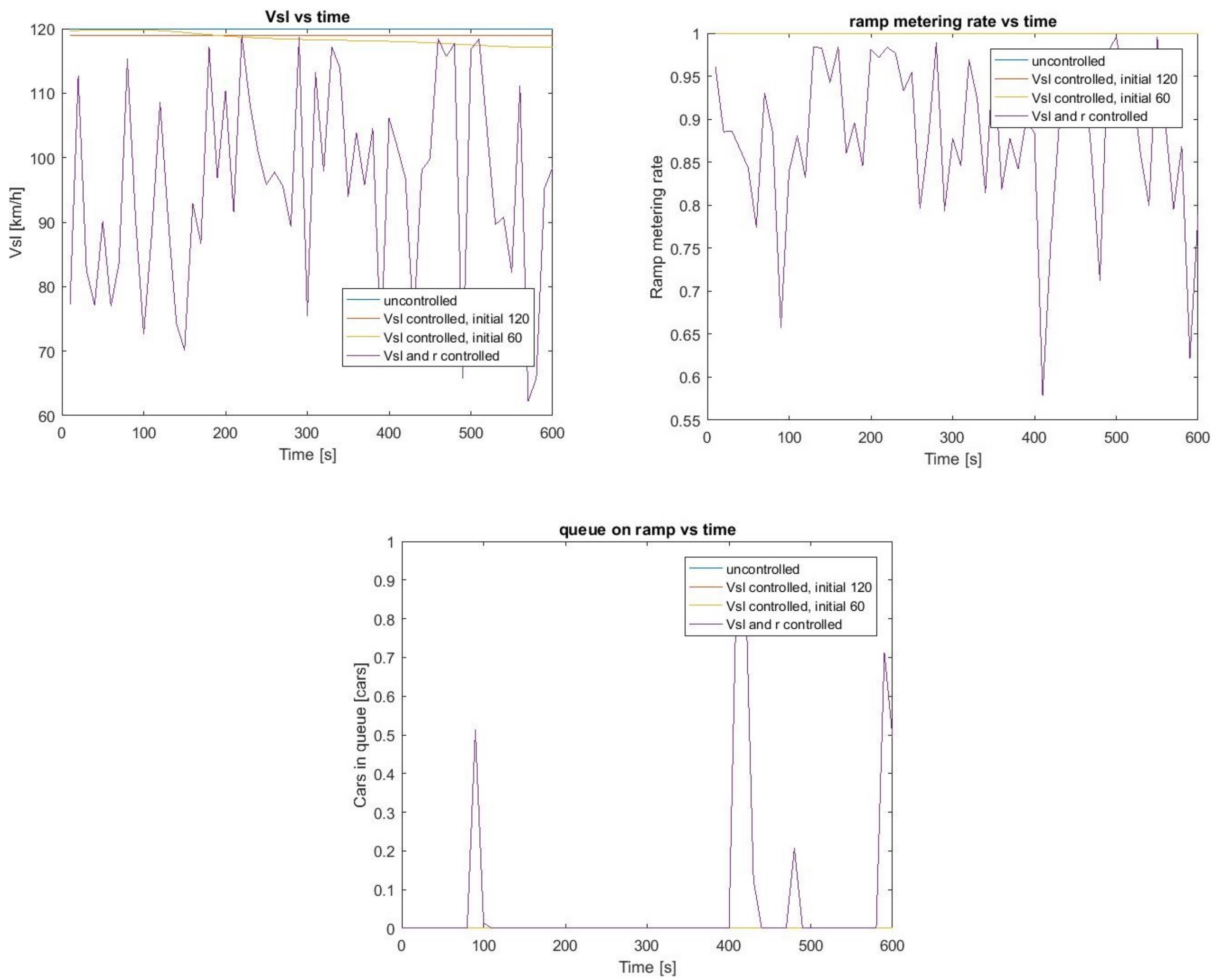

Figure 1.4: V_SL, $r(k)$ and queing plot

For the different cases the following can be concluded:

1. For the uncontrolled case, where V_{SL} is 120 km/h and the ramp metering value is 1, the TTS is 29.06 h.
2. For the case where only the V_{SL} is controlled and calculated by the Matlab function fmincon, with an initial value of 120km/h, the TTS is 29.06 h.
3. For the case where only the V_{SL} is controlled and calculated by the Matlab function fmincon, with an initial value of 60km/h, the TTS is 29.06 h.
4. For the case where both the V_{SL} and ramp metering are controlled by a genetic algorithm, the TTS is 29.06 h.

The reason that all cases have the same result, is most likely because there is no queue formed on the ramp.

1.6 Discrete V_{SL} values

The discrete set of V_{SL} can replace the V_{SL} vector in question 2 by implementing a new vector, in our case V_{SL}, where V_{SL} is implemented in the following way:

```

Vsl = X(1:60);

for j = 1:60
    if Vsl(j) < 70
        Vsld(j) = 60;
    elseif Vsl(j) < 90
        Vsld(j) = 80;
    elseif Vsl(j) < 110
        Vsld(j) = 100;
    else
        Vsld(j) = 120;
    end
end
```

References

[1] T. van den Boom and B. De Schutter, *Lecture notes Optimization in systems and control*, Delft, September 2018.

Appendices

A | Matlab code

```
clear all
close all
clc

[TTS2] = funExercise2(120*ones(1,60));
[X31,Fval31,flag31] = fmincon(@funExercise2,120*ones(1,60),[],[],[],[],60*ones
    (1,60),120*ones(1,60));
[X32,Fval32,flag32] = fmincon(@funExercise2,60*ones(1,60),[],[],[],[],60*ones(1,60)
    ,120*ones(1,60));
[X4,Fval4,flag4] = ga(@funExercise4, 120, [],[],[],[],[60*ones(1,60) zeros(1,60)
    ],[120*ones(1,60), ones(1,60)]);
[X6,Fval6,flag6] = ga(@funExercise6, 120, [],[],[],[],[60*ones(1,60) zeros(1,60)
    ],[120*ones(1,60), ones(1,60)]);

[k2,wr2,r2,Vsl2] = dataVsl(120*ones(1,60));
[k31,wr31,r31,Vsl31] = dataVsl(X31);
[k32,wr32,r32,Vsl32] = dataVsl(X32);
[k4,wr4,r4,Vsl4] = dataVslAndR(X4);

T = 10;
figure
plot((0:k2)*T,wr2,'-',(0:k31)*T,wr31,'-',(0:k32)*T,wr32,'-',(0:k4)*T,wr4,'-')
title('queue on ramp vs time')
xlabel('Time [s]')
ylabel('Cars in queue [cars]')
legend('uncontrolled','Vsl controlled, initial 120', 'Vsl controlled, initial 60',
    'Vsl and r controlled')
figure
plot((1:k2)*T,r2,'-',(1:k31)*T,r31,'-',(1:k32)*T,r32,'-',(1:k4)*T,r4,'-')
title('ramp metering rate vs time')
xlabel('Time [s]')
ylabel('Ramp metering rate')
legend('uncontrolled','Vsl controlled, initial 120', 'Vsl controlled, initial 60',
    'Vsl and r controlled')
figure
plot((1:k2)*T,Vsl2,'-',(1:k31)*T,Vsl31,'-',(1:k32)*T,Vsl32,'-',(1:k4)*T,Vsl4,'-')
title('Vsl vs time')
xlabel('Time [s]')
ylabel('Vsl [km/h]')
legend('uncontrolled','Vsl controlled, initial 120', 'Vsl controlled, initial 60',
    'Vsl and r controlled')

function [TTS] = funExercise2(Vsl)
r=ones(1,60);

E1 = 3;
E2 = 0.5;
E3 = 4.5;

tau = 10;
mu = 80;
Cr = 2000;
rhom = 120;
```

```
alfa = 0.1;
K = 10;
a = 2;
vf = 110;
rhoc = 28;
T = 10/3600;
lambda = 3;
L = 1;


xr(:,1) = [20; 20; 20; 20];
xv(:,1) = [90; 90; 90; 90];
wr(1) = 0;

for k=1:60

    Dr(k) = 1500;
    qr(k) = min([r(k)*Cr Dr(k)+wr(k)/T Cr*(rhom - xr(4,k))/(rhom - rhoc)]);
    if k<12
        q0(k) = (7000 + 100*E1);
    else
        q0(k) = (2000 + 100*E2);
    end

    for i=1:4
        q(i,k) = lambda * xr(i,k) * xv(i,k);
        if i == 1
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q0(k) - q(i,k));
        elseif i == 4
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k) + qr(k));
        else
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k));
        end
    end

    for i=1:4
        if i == 1
            V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K));
        elseif i == 4
            V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i,k)-xr(i,k))/(xr(i,k)+K));
        else
            V(i,k) = min([(1+alfa)*Vsl(k), vf*exp((-1/a)*(xr(i,k)/rhoc)^a)]);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K));
        end
    end

    wr(k+1) = wr(k) + T*(Dr(k)-qr(k));
    y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k));
end
TTS = sum(y);
```

```
end

function [k,wr,r,Vsl] = dataVsl(Vsl)
r=ones(1,60);

E1 = 3;
E2 = 0.5;
E3 = 4.5;

tau = 10;
mu = 80;
Cr = 2000;
rhom = 120;
alfa = 0.1;
K = 10;
a = 2;
vf = 110;
rhoc = 28;
T = 10/3600;
lambda = 3;
L = 1;


xr(:,1) = [20; 20; 20; 20];
xv(:,1) = [90; 90; 90; 90];
wr(1) = 0;

for k=1:60

    Dr(k) = 1500;
    qr(k) = min([r(k)*Cr Dr(k)+wr(k)/T Cr*(rhom - xr(4,k))/(rhom - rhoc)]);
    if k<12
        q0(k) = (7000 + 100*E1);
    else
        q0(k) = (2000 + 100*E2);
    end

    for i=1:4
        q(i,k) = lambda * xr(i,k) * xv(i,k);
        if i == 1
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q0(k) - q(i,k));
        elseif i == 4
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k) + qr(k));
        else
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k));
        end
    end

    for i=1:4
        if i == 1
            V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(
                i,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K));
        elseif i == 4
            V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
```

```
                xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i,k)-xr(i,k))/(xr(i,k)+K));
           else
                V(i,k) = min([(1+alfa)*Vsl(k), vf*exp((-1/a)*(xr(i,k)/rhoc)^a)]);
                xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K));
           end
     end

     wr(k+1) = wr(k) + T*(Dr(k)-qr(k));
     if wr(k) > 20-E3
           y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k))+1e10;
     else
           y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k));
     end
end
TTS = sum(y);
end

function [TTS] = funExercise4(X)
Vsl = X(1:60);
r = X(61:120);

E1 = 3;
E2 = 0.5;
E3 = 4.5;

tau = 10;
mu = 80;
Cr = 2000;
rhom = 120;
alfa = 0.1;
K = 10;
a = 2;
vf = 110;
rhoc = 28;
T = 10/3600;
lambda = 3;
L = 1;


xr(:,1) = [20; 20; 20; 20];
xv(:,1) = [90; 90; 90; 90];
wr(1) = 0;

for k=1:60

     Dr(k) = 1500;
     qr(k) = min([r(k)*Cr Dr(k)+wr(k)/T Cr*(rhom - xr(4,k))/(rhom - rhoc)]);
     if k<12
           q0(k) = (7000 + 100*E1);
     else
           q0(k) = (2000 + 100*E2);
     end

```

```
    for i=1:4
        q(i,k) = lambda * xr(i,k) * xv(i,k);
        if i == 1
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q0(k) - q(i,k));
        elseif i == 4
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k) + qr(k));
        else
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k));
        end
    end

    for i=1:4
        if i == 1
            V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(
                i,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K));
        elseif  i == 4
            V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(
                i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i,k)-xr(i,k))/(xr(i,k)+K));
        else
            V(i,k) = min([(1+alfa)*Vsl(k), vf*exp((-1/a)*(xr(i,k)/rhoc)^a)]);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(
                i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K))
                ;
        end
    end

    wr(k+1) = wr(k) + T*(Dr(k)-qr(k));
    if wr(k) > 20-E3
        y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k))+1e10;
    else
        y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k));
    end
end

TTS = sum(y);
end

function [k,wr,r,Vsl] = dataVslAndR(X)
Vsl = X(1:60);
r = X(61:120);

E1 = 3;
E2 = 0.5;
E3 = 4.5;

tau = 10;
mu = 80;
Cr = 2000;
rhom = 120;
alfa = 0.1;
K = 10;
a = 2;
vf = 110;
rhoc = 28;
```

```
T = 10/3600;
lambda = 3;
L = 1;


xr(:,1) = [20; 20; 20; 20];
xv(:,1) = [90; 90; 90; 90];
wr(1) = 0;

for k=1:60

    Dr(k) = 1500;
    qr(k) = min([r(k)*Cr Dr(k)+wr(k)/T Cr*(rhom - xr(4,k))/(rhom - rhoc)]);
    if k<12
        q0(k) = (7000 + 100*E1);
    else
        q0(k) = (2000 + 100*E2);
    end

    for i=1:4
        q(i,k) = lambda * xr(i,k) * xv(i,k);
        if i == 1
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q0(k) - q(i,k));
        elseif i == 4
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k) + qr(k));
        else
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k));
        end
    end

    for i=1:4
        if i == 1
            V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K));
        elseif i == 4
            V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i,k)-xr(i,k))/(xr(i,k)+K));
        else
            V(i,k) = min([(1+alfa)*Vsl(k), vf*exp((-1/a)*(xr(i,k)/rhoc)^a)]);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K));
        end
    end

    wr(k+1) = wr(k) + T*(Dr(k)-qr(k));
    if wr(k) > 20-E3
        y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k))+1e10;
    else
        y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k));
    end
end

TTS = sum(y);
```

```
end

function [TTS] = funExercise6(X)
Vsl = X(1:60);

for j = 1:60
    if Vsl(j) < 70
        Vsld(j) = 60;
    elseif Vsl(j) < 90
        Vsld(j) = 80;
    elseif Vsl(j) < 110
        Vsld(j) = 100;
    else
        Vsld(j) = 120;
    end
end

r = X(61:120);

E1 = 3;
E2 = 0.5;
E3 = 4.5;

tau = 10;
mu = 80;
Cr = 2000;
rhom = 120;
alfa = 0.1;
K = 10;
a = 2;
vf = 110;
rhoc = 28;
T = 10/3600;
lambda = 3;
L = 1;


xr(:,1) = [20; 20; 20; 20];
xv(:,1) = [90; 90; 90; 90];
wr(1) = 0;

for k=1:60

    Dr(k) = 1500;
    qr(k) = min([r(k)*Cr Dr(k)+wr(k)/T Cr*(rhom - xr(4,k))/(rhom - rhoc)]);
    if k<12
        q0(k) = (7000 + 100*E1);
    else
        q0(k) = (2000 + 100*E2);
    end

    for i=1:4
        q(i,k) = lambda * xr(i,k) * xv(i,k);
        if i == 1
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q0(k) - q(i,k));
        elseif i == 4
```

```
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k) + qr(k));
        else
            xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k));
        end
    end

    for i=1:4
        if i == 1
            V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K));
        elseif i == 4
            V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i,k)-xr(i,k))/(xr(i,k)+K));
        else
            V(i,k) = min([(1+alfa)*Vsld(k), vf*exp((-1/a)*(xr(i,k)/rhoc)^a)]);
            xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K))
            ;
        end
    end

    wr(k+1) = wr(k) + T*(Dr(k)-qr(k));
    if wr(k) > 20-E3
        y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k))+1e10;
    else
        y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k));
    end
end

TTS = sum(y);
end
```