



OPTIMIZATION IN SYSTEMS AND CONTROL (SC42055)

Non Linear optimization Assignment

Authors:

Kwabena Ofori

Koen Bakker

Student Number:

4031601

4389018

October 31, 2018

Contents

1	Non linear programming assignment	1
1.1	State Space formulation	1
1.2	Minimize Total Time Spent	2
1.3	Different starting points	2
1.4	Multiple control measures	3
1.5	Results	4
1.6	Discrete V_{SL} values	5
	Appendices	7
A	Matlab code	8

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:

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

Where $\rho_i(k)$ is the traffic 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

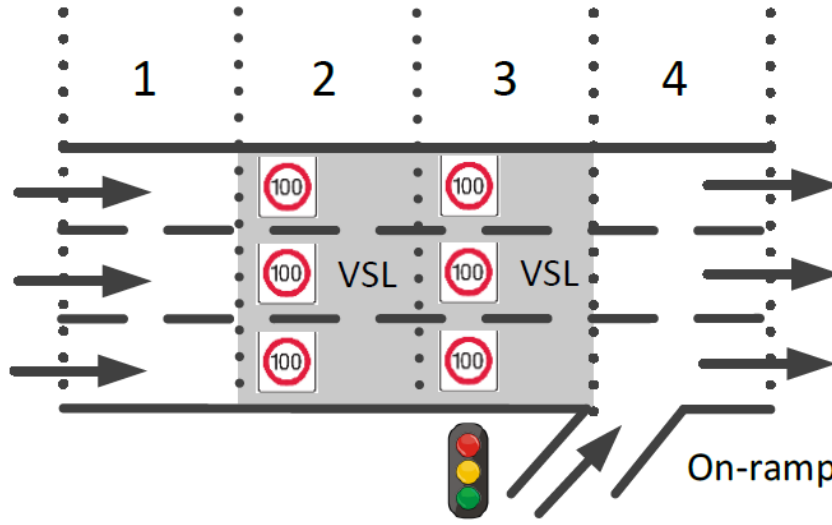


Figure 1.1: METANET

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

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \\ x_5(k+1) \\ x_6(k+1) \\ x_7(k+1) \\ x_8(k+1) \\ x_9(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) + \frac{T}{\lambda_1 L_1} (\lambda_0(k) x_5(k) - x_1(k) x_5(k)) \\ x_2(k) + \frac{T}{\lambda_2 L_2} (\lambda_1(k) x_5(k) - x_2(k) x_6(k)) \\ x_3(k) + \frac{T}{\lambda_3 L_3} (\lambda_2(k) x_6(k) - x_3(k) x_7(k)) \\ x_4(k) + \frac{T}{\lambda_4 L_4} (\lambda_3(k) x_7(k) - x_4(k) x_8(k)) \\ x_5(k) + \frac{T}{\tau} (v f e^{\frac{-1}{a} \frac{x_1(k)}{\rho_c}} - x_5(k)) + \frac{T}{L} x_5(k) (x_5(k) - x_5(k)) - \frac{\mu T}{\tau L} \frac{x_2(k) - x_1(k)}{x_1(k) + K} \\ x_6(k) - \frac{T}{\tau} x_6(k) + \frac{T}{L} x_6(k) (x_5(k) - x_6(k)) - \frac{\mu T}{\tau L} \frac{x_3(k) - x_2(k)}{x_2(k) + K} \\ x_7(k) - \frac{T}{\tau} x_7(k) + \frac{T}{L} x_7(k) (x_6(k) - x_7(k)) - \frac{\mu T}{\tau L} \frac{x_4(k) - x_3(k)}{x_3(k) + K} \\ x_8(k) + \frac{T}{\tau} (v f e^{\frac{-1}{a} \frac{x_4(k)}{\rho_c}} - x_8(k)) + \frac{T}{L} x_8(k) (x_7(k) - x_8(k)) - \frac{\mu T}{\tau L} \frac{x_5(k) - x_4(k)}{x_4(k) + K} \\ x_9(k) \end{bmatrix} + \begin{bmatrix} q_0 + q_{r,1}(k) \\ 0 \\ 0 \\ r(k) C_r \\ 0 \\ (1 + \alpha) V_{SL}(k) \\ (1 + \alpha) V_{SL}(k) \\ 0 \\ T(D_r(k) - q_r(k)) \end{bmatrix}$$

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

$$y(k) = [T x_9(k) + T L \lambda (x_1(k) + x_2(k) + x_3(k) + x_4(k))] \quad (1.2)$$

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:

$$y(k) = [Tx_9(k) + TL\lambda(x_1(k) + x_2(k) + x_3(k) + x_4(k))] \quad (1.3)$$

Where:

$$TTS = \sum_1^k y(k) \quad (1.4)$$

So:

$$\min TTS = \min \sum_1^k [Tx_9(k) + TL\lambda(x_1(k) + x_2(k) + x_3(k) + x_4(k))] \quad (1.5)$$

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

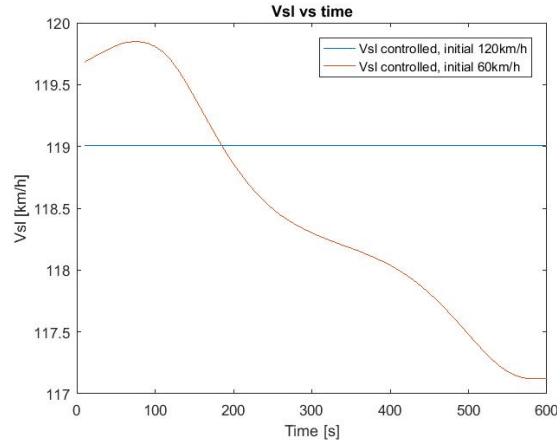


Figure 1.2: V_{SL} plot for different initial points

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:

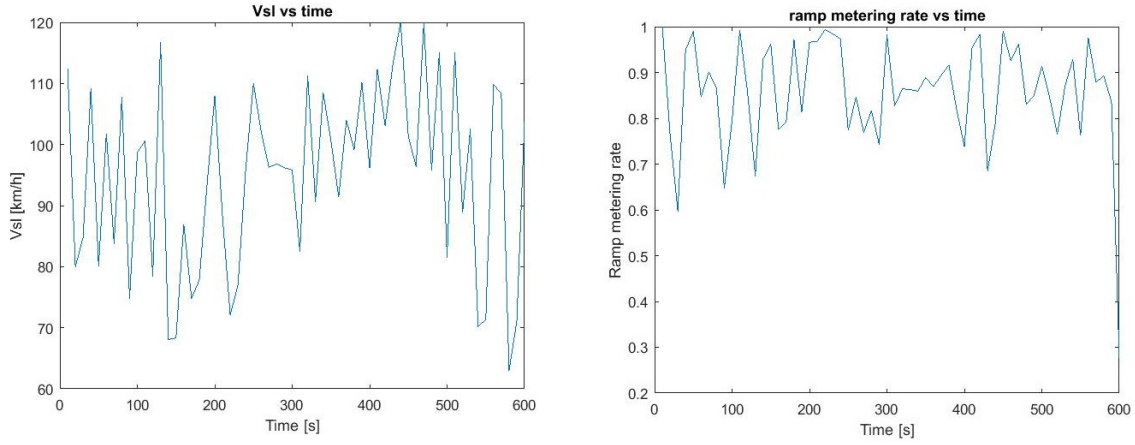


Figure 1.3: V_{SL} , $r(k)$ plot

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:

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

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:

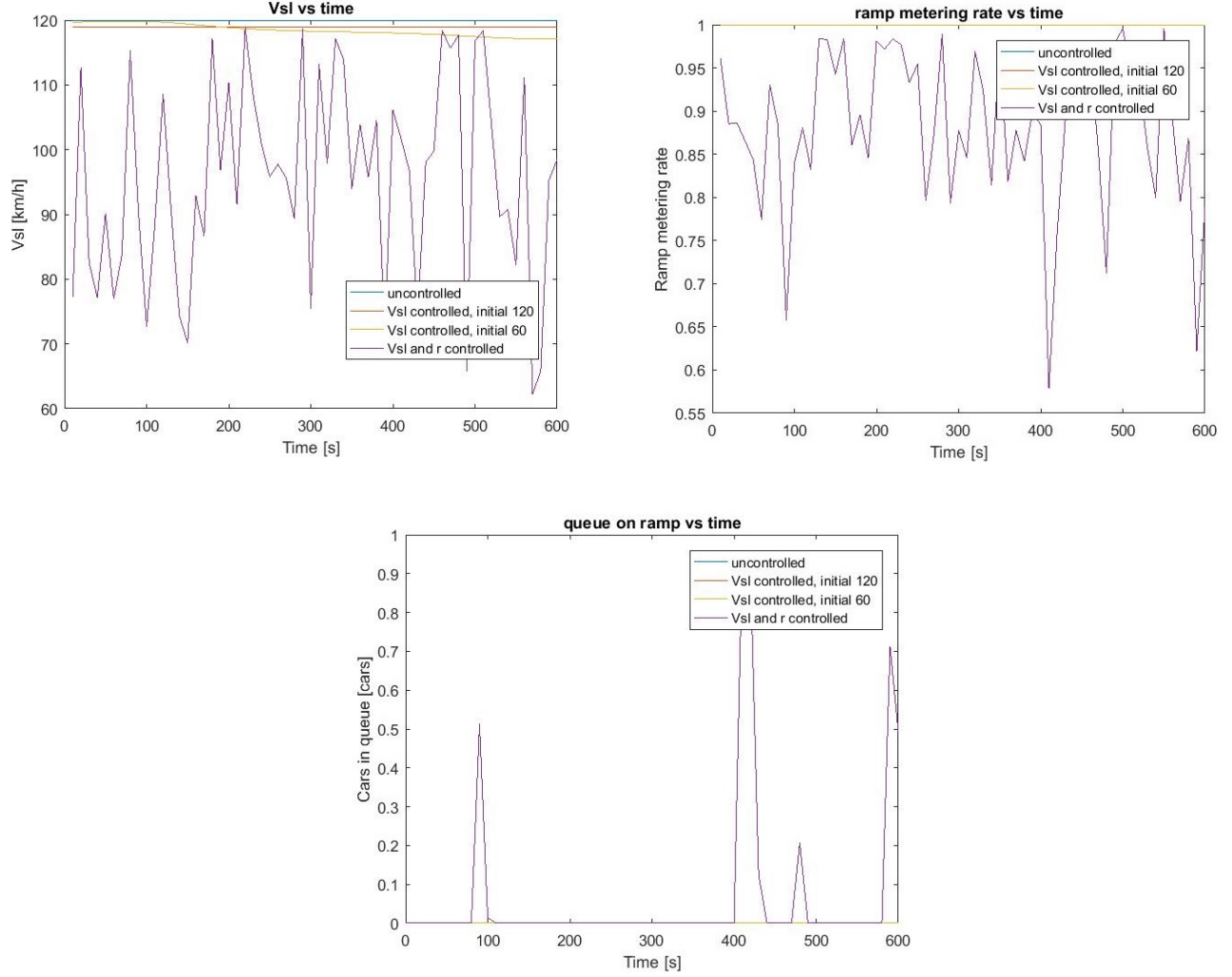


Figure 1.4: V_{SL} , $r(k)$ and queuing plot

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:

```
1 Vsl = X(1:60);
2
3
4 for j = 1:60
5     if Vsl(j) < 70
6         Vsld(j) = 60;
7     elseif Vsl(j) < 90
8         Vsld(j) = 80;
9     elseif Vsl(j) < 110
10        Vsld(j) = 100;
11    else
12        Vsld(j) = 120;
13    end
14 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

```
1 clear all
2 close all
3 clc
4
5 [TTS2] = funExercise2(120*ones(1,60));
6 [X31,Fval31,flag31] = fmincon(@funExercise2,120*ones(1,60),[],[],[],[],60*ones
   (1,60),120*ones(1,60));
7 [X32,Fval32,flag32] = fmincon(@funExercise2,60*ones(1,60),[],[],[],[],60*ones(1,60)
   ,120*ones(1,60));
8 [X4,Fval4,flag4] = ga(@funExercise4, 120, [],[],[],[],[60*ones(1,60) zeros(1,60)
   ],[120*ones(1,60), ones(1,60)]);
9 [X6,Fval6,flag6] = ga(@funExercise6, 120, [],[],[],[],[60*ones(1,60) zeros(1,60)
   ],[120*ones(1,60), ones(1,60)]);
10
11 [k2,wr2,r2,Vsl2] = dataVsl(120*ones(1,60));
12 [k31,wr31,r31,Vsl31] = dataVsl(X31);
13 [k32,wr32,r32,Vsl32] = dataVsl(X32);
14 [k4,wr4,r4,Vsl4] = dataVslAndR(X4);
15
16 T = 10;
17 figure
18 plot((0:k2)*T,wr2,'-',(0:k31)*T,wr31,'-',(0:k32)*T,wr32,'-',(0:k4)*T,wr4,'-')
19 title('queue on ramp vs time')
20 xlabel('Time [s]')
21 ylabel('Cars in queue [cars]')
22 legend('uncontrolled','Vsl controlled, initial 120','Vsl controlled, initial 60',
   'Vsl and r controlled')
23 figure
24 plot((1:k2)*T,r2,'-',(1:k31)*T,r31,'-',(1:k32)*T,r32,'-',(1:k4)*T,r4,'-')
25 title('ramp metering rate vs time')
26 xlabel('Time [s]')
27 ylabel('Ramp metering rate')
28 legend('uncontrolled','Vsl controlled, initial 120','Vsl controlled, initial 60',
   'Vsl and r controlled')
29 figure
30 plot((1:k2)*T,Vsl2,'-',(1:k31)*T,Vsl31,'-',(1:k32)*T,Vsl32,'-',(1:k4)*T,Vsl4,'-')
31 title('Vsl vs time')
32 xlabel('Time [s]')
33 ylabel('Vsl [km/h]')
34 legend('uncontrolled','Vsl controlled, initial 120','Vsl controlled, initial 60',
   'Vsl and r controlled')
35
36 function [TTS] = funExercise2(Vsl)
37 r=ones(1,60);
38
39 E1 = 3;
40 E2 = 0.5;
41 E3 = 4.5;
42
43 tau = 10;
44 mu = 80;
45 Cr = 2000;
46 rhom = 120;
```

```

47 | alfa = 0.1;
48 | K = 10;
49 | a = 2;
50 | vf = 110;
51 | rhoc = 28;
52 | T = 10/3600;
53 | lambda = 3;
54 | L = 1;
55 |
56 |
57 | xr(:,1) = [20; 20; 20; 20];
58 | xv(:,1) = [90; 90; 90; 90];
59 | wr(1) = 0;
60 |
61 | for k=1:60
62 |
63 |     Dr(k) = 1500;
64 |     qr(k) = min([r(k)*Cr Dr(k)+wr(k)/T Cr*(rhom - xr(4,k))/(rhom - rhoc)]);
65 |     if k<12
66 |         q0(k) = (7000 + 100*E1);
67 |     else
68 |         q0(k) = (2000 + 100*E2);
69 |     end
70 |
71 |     for i=1:4
72 |         q(i,k) = lambda * xr(i,k) * xv(i,k);
73 |         if i == 1
74 |             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q0(k) - q(i,k));
75 |         elseif i == 4
76 |             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k) + qr(k));
77 |         else
78 |             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k));
79 |         end
80 |     end
81 |
82 |     for i=1:4
83 |         if i == 1
84 |             V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
85 |             xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(
86 |                 i,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K));
87 |         elseif i == 4
88 |             V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
89 |             xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(
90 |                 i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i,k)-xr(i,k))/(xr(i,k)+K));
91 |         else
92 |             V(i,k) = min([(1+alfa)*Vsl(k), vf*exp((-1/a)*(xr(i,k)/rhoc)^a)]);
93 |             xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(
94 |                 i-1,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K))
95 |             ;
96 |         end
97 |     end
98 |     wr(k+1) = wr(k) + T*(Dr(k)-qr(k));
99 |     y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k));
100 | end
101 | TTS = sum(y);

```

```

99 end
100
101 function [k,wr,r,Vsl] = dataVsl(Vsl)
102 r=ones(1,60);
103
104 E1 = 3;
105 E2 = 0.5;
106 E3 = 4.5;
107
108 tau = 10;
109 mu = 80;
110 Cr = 2000;
111 rhom = 120;
112 alfa = 0.1;
113 K = 10;
114 a = 2;
115 vf = 110;
116 rhoc = 28;
117 T = 10/3600;
118 lambda = 3;
119 L = 1;
120
121
122 xr(:,1) = [20; 20; 20; 20];
123 xv(:,1) = [90; 90; 90; 90];
124 wr(1) = 0;
125
126 for k=1:60
127
128     Dr(k) = 1500;
129     qr(k) = min([r(k)*Cr Dr(k)+wr(k)/T Cr*(rhom - xr(4,k))/(rhom - rhoc)]);
130     if k<12
131         q0(k) = (7000 + 100*E1);
132     else
133         q0(k) = (2000 + 100*E2);
134     end
135
136     for i=1:4
137         q(i,k) = lambda * xr(i,k) * xv(i,k);
138         if i == 1
139             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q0(k) - q(i,k));
140         elseif i == 4
141             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k) + qr(k));
142         else
143             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k));
144         end
145     end
146
147     for i=1:4
148         if i == 1
149             V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
150             xv(i,k+1) = xv(i,k) + (T/tau) * (V(i,k)-xv(i,k)) + (T/L)*xv(i,k) * (xv(
151                 i,k)-xv(i,k)) - ((mu*T)/(tau*L))*((xr(i+1,k)-xr(i,k))/(xr(i,k)+K));
152             elseif i == 4
153                 V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);

```

```

153         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));
154     else
155         V(i,k) = min([(1+alfa)*Vsl(k), vf*exp((-1/a)*(xr(i,k)/rhoc)^a)]);
156         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))
            ;
157     end
158 end
159
160 wr(k+1) = wr(k) + T*(Dr(k)-qr(k));
161 if wr(k) > 20-E3
162     y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k))+1e10;
163 else
164     y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k));
165 end
166 end
167 TTS = sum(y);
168 end
169
170 function [TTS] = funExercise4(X)
171 Vsl = X(1:60);
172 r = X(61:120);
173
174 E1 = 3;
175 E2 = 0.5;
176 E3 = 4.5;
177
178 tau = 10;
179 mu = 80;
180 Cr = 2000;
181 rhom = 120;
182 alfa = 0.1;
183 K = 10;
184 a = 2;
185 vf = 110;
186 rhoc = 28;
187 T = 10/3600;
188 lambda = 3;
189 L = 1;
190
191
192 xr(:,1) = [20; 20; 20; 20];
193 xv(:,1) = [90; 90; 90; 90];
194 wr(1) = 0;
195
196 for k=1:60
197
198     Dr(k) = 1500;
199     qr(k) = min([r(k)*Cr Dr(k)+wr(k)/T Cr*(rhom - xr(4,k))/(rhom - rhoc)]);
200     if k<12
201         q0(k) = (7000 + 100*E1);
202     else
203         q0(k) = (2000 + 100*E2);
204     end
205

```

```

206     for i=1:4
207         q(i,k) = lambda * xr(i,k) * xv(i,k);
208         if i == 1
209             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q0(k) - q(i,k));
210         elseif i == 4
211             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k) + qr(k));
212         else
213             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k));
214         end
215     end
216
217     for i=1:4
218         if i == 1
219             V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
220             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));
221         elseif i == 4
222             V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
223             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));
224         else
225             V(i,k) = min([(1+alfa)*Vsl(k), vf*exp((-1/a)*(xr(i,k)/rhoc)^a)]);
226             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))
                ;
227         end
228     end
229
230     wr(k+1) = wr(k) + T*(Dr(k)-qr(k));
231     if wr(k) > 20-E3
232         y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k))+1e10;
233     else
234         y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k));
235     end
236 end
237
238 TTS = sum(y);
239 end
240
241 function [k,wr,r,Vsl] = dataVslAndR(X)
242 Vsl = X(1:60);
243 r = X(61:120);
244
245 E1 = 3;
246 E2 = 0.5;
247 E3 = 4.5;
248
249 tau = 10;
250 mu = 80;
251 Cr = 2000;
252 rhom = 120;
253 alfa = 0.1;
254 K = 10;
255 a = 2;
256 vf = 110;
257 rhoc = 28;

```

```

258 T = 10/3600;
259 lambda = 3;
260 L = 1;
261
262
263 xr(:,1) = [20; 20; 20; 20];
264 xv(:,1) = [90; 90; 90; 90];
265 wr(1) = 0;
266
267 for k=1:60
268
269     Dr(k) = 1500;
270     qr(k) = min([r(k)*Cr Dr(k)+wr(k)/T Cr*(rho_m - xr(4,k))/(rho_m - rho_c)]);
271     if k<12
272         q0(k) = (7000 + 100*E1);
273     else
274         q0(k) = (2000 + 100*E2);
275     end
276
277     for i=1:4
278         q(i,k) = lambda * xr(i,k) * xv(i,k);
279         if i == 1
280             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q0(k) - q(i,k));
281         elseif i == 4
282             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k) + qr(k));
283         else
284             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k));
285         end
286     end
287
288     for i=1:4
289         if i == 1
290             V(i,k) = vf*exp((-1/a)*(xr(i,k)/rho_c)^a);
291             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));
292         elseif i == 4
293             V(i,k) = vf*exp((-1/a)*(xr(i,k)/rho_c)^a);
294             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));
295         else
296             V(i,k) = min([(1+alfa)*Vsl(k), vf*exp((-1/a)*(xr(i,k)/rho_c)^a)]);
297             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))
                ;
298         end
299     end
300
301     wr(k+1) = wr(k) + T*(Dr(k)-qr(k));
302     if wr(k) > 20-E3
303         y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k))+1e10;
304     else
305         y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k));
306     end
307 end
308
309 TTS = sum(y);

```

```

310 end
311
312 function [TTS] = funExercise6(X)
313 Vsl = X(1:60);
314
315 for j = 1:60
316     if Vsl(j) < 70
317         Vsld(j) = 60;
318     elseif Vsl(j) < 90
319         Vsld(j) = 80;
320     elseif Vsl(j) < 110
321         Vsld(j) = 100;
322     else
323         Vsld(j) = 120;
324     end
325 end
326
327 r = X(61:120);
328
329 E1 = 3;
330 E2 = 0.5;
331 E3 = 4.5;
332
333 tau = 10;
334 mu = 80;
335 Cr = 2000;
336 rhom = 120;
337 alfa = 0.1;
338 K = 10;
339 a = 2;
340 vf = 110;
341 rhoc = 28;
342 T = 10/3600;
343 lambda = 3;
344 L = 1;
345
346
347 xr(:,1) = [20; 20; 20; 20];
348 xv(:,1) = [90; 90; 90; 90];
349 wr(1) = 0;
350
351 for k=1:60
352
353     Dr(k) = 1500;
354     qr(k) = min([r(k)*Cr Dr(k)+wr(k)/T Cr*(rhom - xr(4,k))/(rhom - rhoc)]);
355     if k<12
356         q0(k) = (7000 + 100*E1);
357     else
358         q0(k) = (2000 + 100*E2);
359     end
360
361     for i=1:4
362         q(i,k) = lambda * xr(i,k) * xv(i,k);
363         if i == 1
364             xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q0(k) - q(i,k));
365         elseif i == 4

```



```

366         xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k) + qr(k));
367     else
368         xr(i,k+1) = xr(i,k) + (T/(lambda*L)) * (q(i-1,k) - q(i,k));
369     end
370 end
371
372 for i=1:4
373     if i == 1
374         V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
375         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));
376     elseif i == 4
377         V(i,k) = vf*exp((-1/a)*(xr(i,k)/rhoc)^a);
378         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));
379     else
380         V(i,k) = min([(1+alfa)*Vsld(k), vf*exp((-1/a)*(xr(i,k)/rhoc)^a)]);
381         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))
            ;
382     end
383 end
384
385 wr(k+1) = wr(k) + T*(Dr(k)-qr(k));
386 if wr(k) > 20-E3
387     y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k))+1e10;
388 else
389     y(k) = T*wr(k) + T*L*lambda*sum(xr(:,k));
390 end
391 end
392
393 TTS = sum(y);
394 end

```