# Optimization in Systems and Control
Assignment 3

Luca de Laat, 4135040 Nathan Timmers, 4283449

## 1 Discrete-Time State-Space Model

In this system we have:

$$\begin{aligned} x(k) &= \begin{bmatrix} x_1(k) & x_2(k) & x_3(k) & x_4(k) & x_1(k) & x_2(k) & x_3(k) & x_4(k) & x_9(k) \end{bmatrix}^T \\ &= \begin{bmatrix} \rho_1(k) & \rho_2(k) & \rho_3(k) & \rho_4(k) & v_1(k) & v_2(k) & v_3(k) & v_4(k) & w_r(k) \end{bmatrix}^T \end{aligned} \tag{1}$$

$$u(k) = \begin{bmatrix} u_1(k) & u_2(k) \end{bmatrix}^T = \begin{bmatrix} VSL_{2,3}(k) & r(k) \end{bmatrix}^T \tag{2}$$

The derived discrete-time state-space model is as follows:

$$x(k+1) = \begin{bmatrix} x_1(k) + \frac{T}{\lambda L}\Big(q_0(k) - \lambda x_1(k) x_5(k)\Big) \\ x_2(k) + \frac{T}{L}\Big(x_1(k)x_5(k) - x_2(k)x_6(k)\Big) \\ x_3(k) + \frac{T}{L}\Big(x_2(k)x_6(k) - x_3(k)x_7(k)\Big) \\ x_4(k) + \frac{T}{\lambda L}\Big(\lambda x_3(k)x_7(k) - \lambda x_4(k)x_8(k) + q_r(k)\Big) \\ x_5(k) + \frac{T}{\tau}(V_1(k) - x_5(k)) - \frac{\mu T}{\tau L}\frac{x_2(k)-x_1(k)}{x_1(k)+K} \\ x_6(k) + \frac{T}{\tau}(V_2(k) - x_6(k)) + \frac{T}{3600\cdot L}x_6(k)(x_5(k) - x_6(k)) - \frac{\mu T}{\tau L}\frac{x_3(k)-x_2(k)}{x_2(k)+K} \\ x_7(k) + \frac{T}{\tau}(V_3(k) - x_7(k)) + \frac{T}{3600\cdot L}x_7(k)(x_6(k) - x_7(k)) - \frac{\mu T}{\tau L}\frac{x_4(k)-x_3(k)}{x_3(k)+K} \\ x_8(k) + \frac{T}{\tau}(V_4(k) - x_8(k)) + \frac{T}{3600\cdot L}x_8(k)(x_7(k) - x_8(k)) \\ x_9 + T(D_r(k) - q_r(k)) \end{bmatrix} \tag{3}$$

with:

$$V_{1,4}(k) = \min\Big((1+\alpha)\cdot 120, v_f e^{-\frac{1}{a}\frac{x_i(k)}{\rho_c}^a}\Big) \tag{4}$$

$$V_{2,3}(k) = \min\Big((1+\alpha)\cdot u_1(k), v_f e^{-\frac{1}{a}\frac{x_i(k)}{\rho_c}^a}\Big) \tag{5}$$

$$q_r(k) = \min\Big(u_2(k)C_r, D_r + \frac{x_9(k)}{T}\Big) \tag{6}$$

As output we have:

$$\begin{aligned} y(k) &= Tw_r(k) + TL\lambda \sum_{i=1,2,3,4} \rho_i(k) \\ &= Tx_9(k) + TL\lambda(x_1(k) + x_2(k) + x_3(k) + x_4(k)) \end{aligned} \tag{7}$$

Where necessary, all the parameters given were converted from hours to seconds. This was necessary with, for instance, $D_r$

## 2 TTS Optimization with varying VSL and constant r

For this problem the Sequential Quadratic Programming algorithm in the *fmincon* solver was used, together with multiple starting points. A non-linear constraint optimization algorithm and multiple starting points were used to increase the chance that the obtained local optimum is the global optimum (which can never be guaranteed). The starting points are $u_1(0) = \{60; 70; 80; 90; 100; 110; 120\}$. Figure 1 shows the result for one of the feasible solutions.

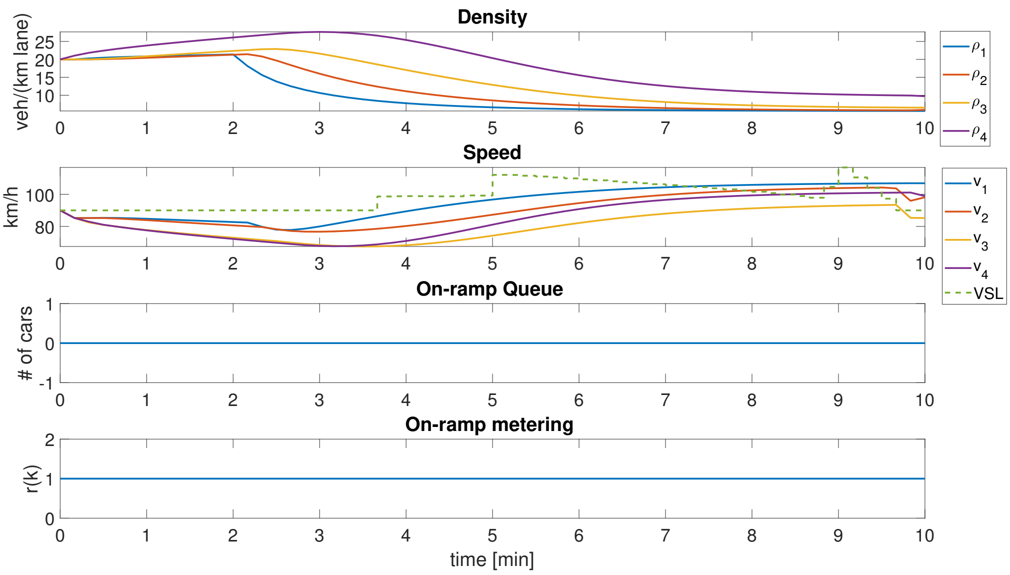


Figure 1: One of the optimization results with multiple starting point for varying $u_1(0)$ and $r = 1 \; \forall \; k$

# 3 Optimization results for different starting points

As can be expected the optimization yields different results for different starting points. This is shown in Figure 2 and 3. Some starting points even cause Matlab's *fmincon* not to converge fast enough and stop before the optimal solution is attained because of the defined iteration limit. This is mostly the case for initial VSL values between 60 and 87 km/h. Passed 87 km/h, the algorithm always converges to the same optimum: TTS = 36.98 hours. As shown in Figure 4, if the problem is optimized using a genetic algorithm, then for an initial VSL value between 60 and 87 km/h, a TTS value of 36.98 hours is not always found. It does however converge. Also, the genetic algorithm is slower. For one specific calculation its calculation time was 21 seconds as opposed to less than 1 second for the *fmincon*. Generally *fmincon* can be used, but the initial value should then be set between 88 and 120 km/h.

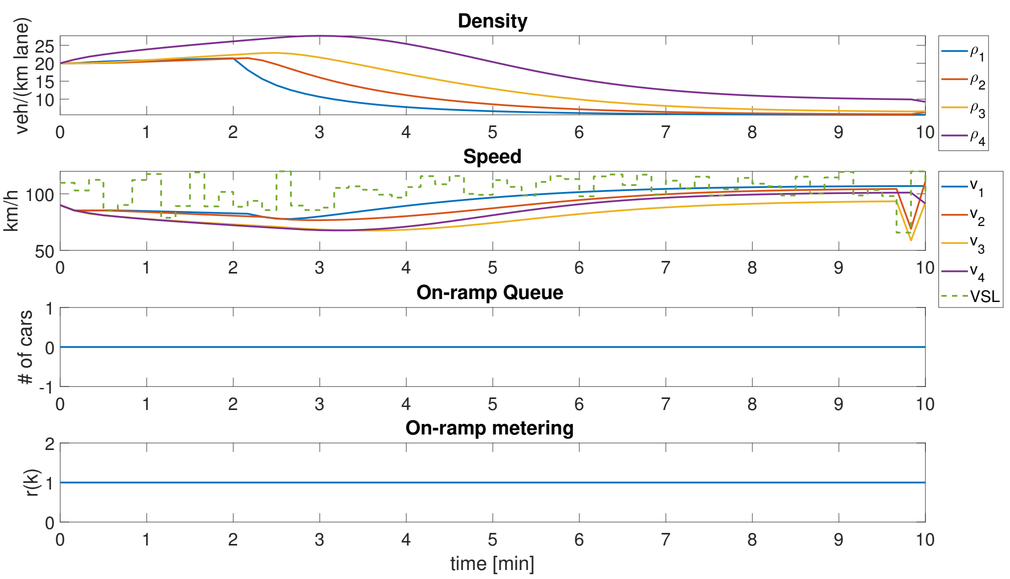


Figure 2: Optimization results for initial point $u_1(0) = 60$ km/h. TTS = 36.98 hours. This optimization was done with a genetic algorithm.

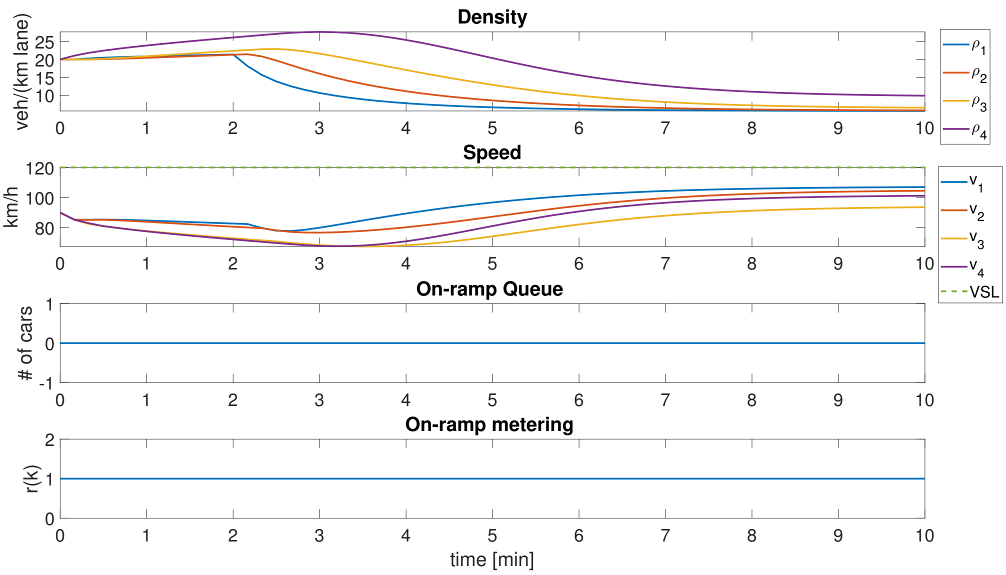


Figure 3: Optimization results for initial point $u_1(0) = 120$ km/h. TTS = 36.98 hours

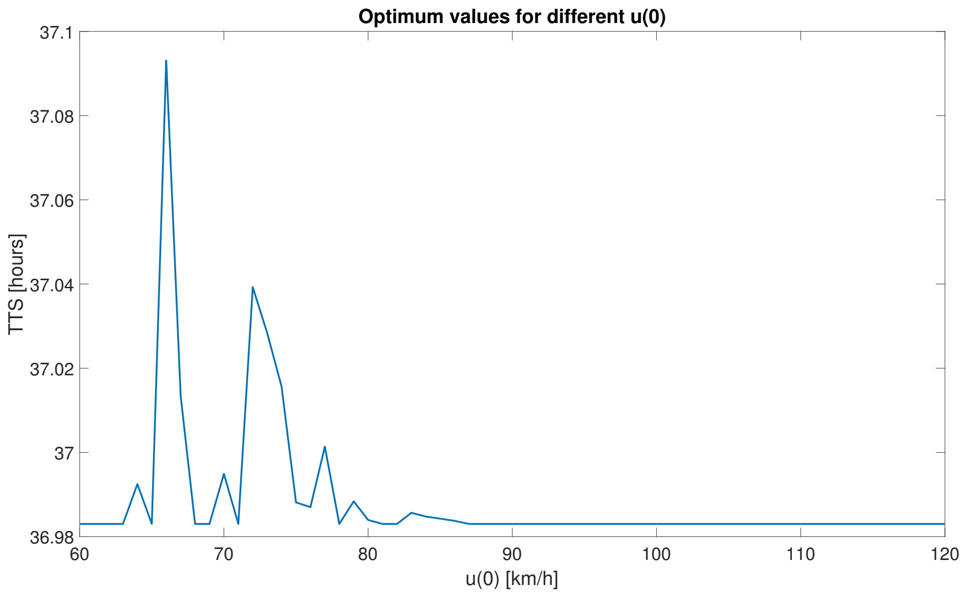


Figure 4: Optimization results for multiple initial points

# 4 TTS Optimization with varying VSL and r

In order to account for the constraint on the maximum queue length we use a penalty function:

$$y_{pen} = \sum^{k} \beta \cdot \max(0, g(x)) \tag{8}$$

with $\beta = 10^9$ and,

$$g(x) = w_r(k) - (20 - E3) \tag{9}$$

This penalty function is added to the objective function to obtain a new minimization problem:

$$\min_{x}(y(x) + y_{pen}(x)) \tag{10}$$

With this optimization the genetic algorithm was used first and the outcome of that was used as a starting point in *fmincon*. This was done to have multiple starting points and to ensure that the global minimum was found. Luckily, with the initial conditions, our limit of the on-ramp queue and our $q_0(0)$, the limit is never exceeded. This explains why the TTS (= 36.98 hours) remains the same as in the previous questions. Figure 5 shows the results for the optimization.

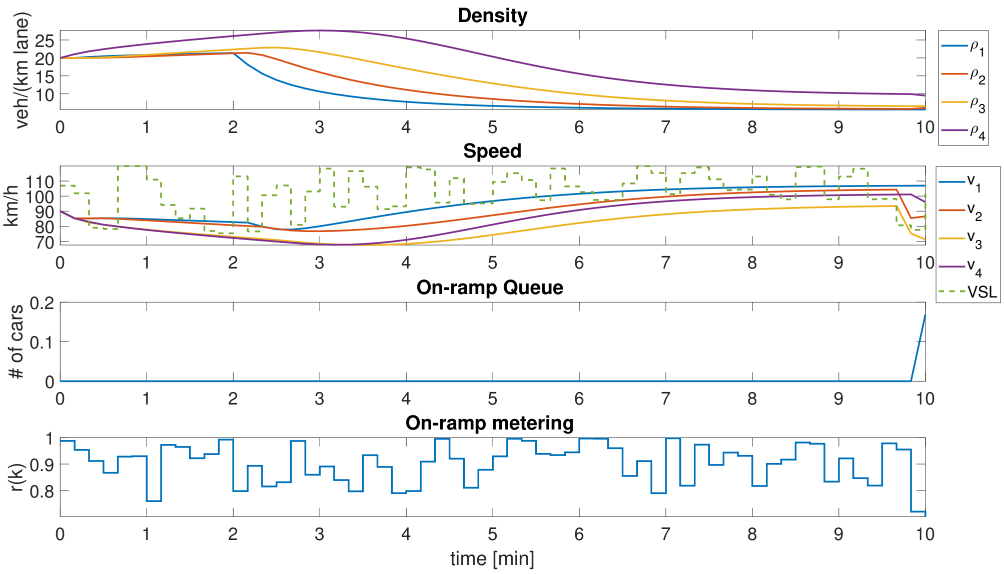


Figure 5: Optimization with first a genetic algorithm and then *fmincon*. TTS = 36.98 hours

# 5 Figures

All the figures can be found at the corresponding question.

# 6 Discrete VSL set

To make sure the VSL can only attain the values {60; 80; 100; 120}, multiple approaches can be used. Our first attempt was to add a nonlinear constraint to the optimization:

$$c_{eq} = u_1(k) \mod 20 \tag{11}$$

Since $u_1$ is bounded between 60 and 120, this would make sure the VSL would only attain the available values. However, using this method resulted in a long optimization time. This makes sense because the nonlinear constraint adds extra complexity. Our second attempt was based on the same principle as the previous method, but now it is not implemented via a nonlinear constraint. The method uses the option of the Matlab genetic algorithm function where you can set certain variables to be integers, so now $u_1$ was set to be an integer. To make sure the VSL could only be set to the allowed values, the optimization could only give the following values to the optimization function: {3; 4; 5; 6}. So:

$$3 \leq u_1 \leq 6$$
$$0 \leq u_2 \leq 1$$
$$u_1 \mod 1 = 0$$

(*This was implemented via the integer option in the genetic algorithm of Matlab*)

In the optimization function these values were multiplied by 20, resulting in the values {60; 80; 100; 120}.

Using this implementation an optimization with a genetic algorithm was done on the system with and without ramp metering. It ran much quicker than with the nonlinear constraint. In Figure 6 the results of the optimization without ramp metering can be found. For the interested reader, Figure 7 shows the results of the optimization with ramp metering. The local optimum found in this question with ramp metering is higher than the optimization in Question 4 with ramp metering. This is certainly due to the smaller solution space resulting from the limited discrete values VSL can adopt. The optimization for the problem with and without ramp metering was ran multiple times.

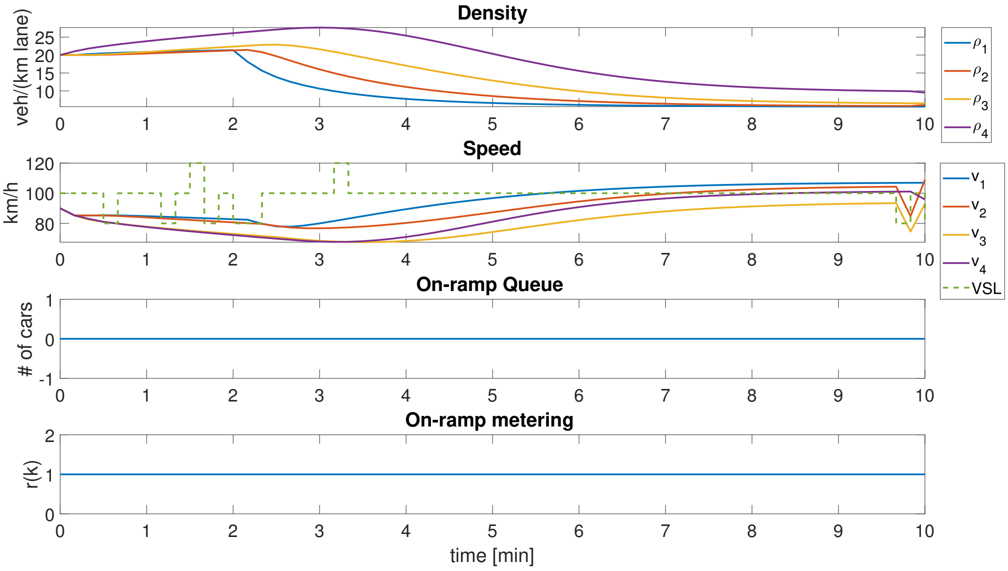


Figure 6: Optimization with discrete VSL and without ramp metering. TTS = 36.98 hours

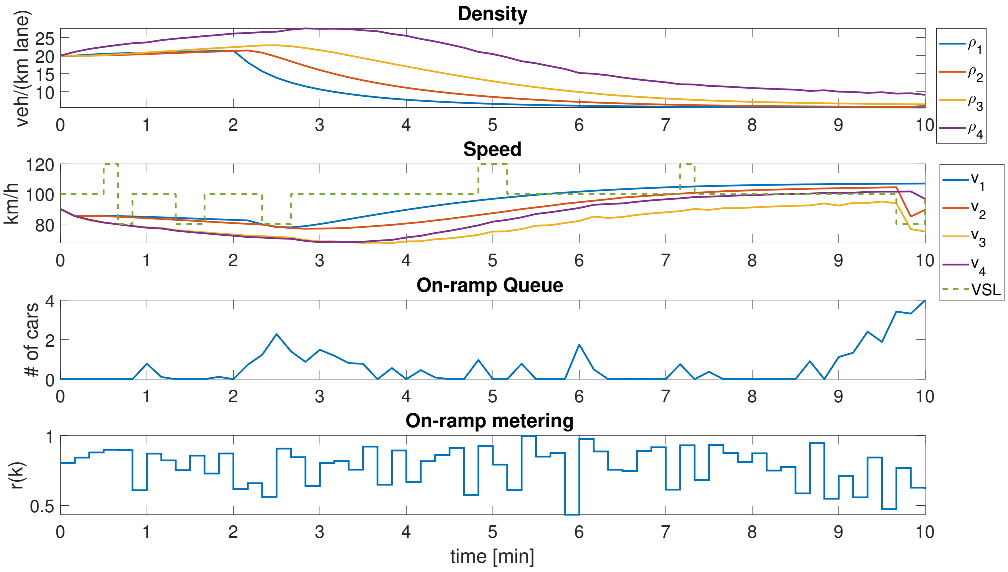


Figure 7: Optimization with discrete VSL and with ramp metering. TTS = 37.02 hours