Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
IDSC
Institut für Dynamische Systeme und Regelungstechnik

151-0310-00 **Problem Set 4**

Topic: **Unconstrained Linear MPC**

March 16, 2022

In today's exercise you will implement an unconstrained linear model predictive controller for the engine air-path control task. For a given prediction horizon and control horizon, your task is to find the optimal control trajectory $\boldsymbol{\Delta u}(\cdot|k)^*$, which minimizes the cost function

$$\begin{aligned} J\left(\boldsymbol{\Delta u}\left(\cdot|k\right)\right) &= \left\|\boldsymbol{r}\left(\cdot|k\right)-\boldsymbol{y}\left(\cdot|k\right)\right\|_{\boldsymbol{Q}}^{2}+\left\|\boldsymbol{\Delta u}\left(\cdot|k\right)\right\|_{\boldsymbol{R}}^{2}, \\ &= \Big(\boldsymbol{r}\left(\cdot|k\right)-\boldsymbol{y}\left(\cdot|k\right)\Big)^{T}\boldsymbol{Q}\,\Big(\boldsymbol{r}\left(\cdot|k\right)-\boldsymbol{y}\left(\cdot|k\right)\Big)+\boldsymbol{\Delta u}\left(\cdot|k\right)^{T}\boldsymbol{R}\,\boldsymbol{\Delta u}\left(\cdot|k\right)\,. \end{aligned} \tag{1}$$

As shown in the lecture, the optimal input rate trajectory $\boldsymbol{\Delta u}(k|k)^*$ for a linear model and an optimal control problem without constraints is

$$\boldsymbol{\Delta u}(k|k)^* = \boldsymbol{K}_{\mathrm{MPC}}\,\boldsymbol{e}\left(\cdot|k\right)\;. \tag{2}$$

In the following exercises, you will derive the constant control law $\boldsymbol{K}_{\mathrm{MPC}}$ and all the matrices required to calculate the free control error $\boldsymbol{e}\left(\cdot|k\right)$, implement them in Simulink, and use them to control the plant.

Exercise 1 (System Response Trajectory Matrices)

In order to find the optimal input rate $\boldsymbol{\Delta u}(k|k)^*$ trajectory, we set the derivative of $J\left(\boldsymbol{\Delta u}\left(\cdot|k\right)\right)$ to zero. Therefore, we have to calculate the system response trajectory $\boldsymbol{y}\left(\cdot|k\right)$ for a given prediction horizon and control horizon.

For a discrete-time linear system of the form

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\,\boldsymbol{x}(k)+\boldsymbol{B}\,\boldsymbol{u}(k-1)+\boldsymbol{B}\,\boldsymbol{\Delta u}(k|k) \qquad \boldsymbol{x}\in\mathbb{R}^n\,; \qquad \boldsymbol{u}\in\mathbb{R}^l \tag{3}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\,\boldsymbol{x}(k) \qquad \boldsymbol{y}\in\mathbb{R}^m\,, \tag{4}$$

the system response trajectory was derived in the lecture and is given by

$$\boldsymbol{y}\left(\cdot|k\right)=\boldsymbol{\Gamma}\,\boldsymbol{x}\left(\cdot|k\right)=\boldsymbol{\Gamma}\,\Big(\boldsymbol{\Psi}\,\boldsymbol{x}(k)+\boldsymbol{\Upsilon}\,\boldsymbol{u}(k-1)+\boldsymbol{\Theta}\,\boldsymbol{\Delta u}\left(\cdot|k\right)\Big) \qquad \boldsymbol{y}\left(\cdot|k\right)\in\mathbb{R}^{m\cdot(N_2-N_1+1)\,\times\,1}. \tag{5}$$

It depends on the initial conditions $\boldsymbol{x}(k)$, the initial input $\boldsymbol{u}(k-1)$ and the input rate trajectory $\boldsymbol{\Delta u}\left(\cdot|k\right)$. The matrices $\boldsymbol{\Gamma}$, $\boldsymbol{\Psi}$, $\boldsymbol{\Upsilon}$, and $\boldsymbol{\Theta}$ are defined as

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{C} & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{C} \end{bmatrix} \in\mathbb{R}^{m\cdot(N_2-N_1+1)\,\times\,n\cdot(N_2-N_1+1)} \tag{6}$$

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{A}^{N_1} \\ \vdots \\ \boldsymbol{A}^{N_2} \end{bmatrix} \in\mathbb{R}^{n\cdot(N_2-N_1+1)\,\times\,n} \tag{7}$$

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \sum\limits_{i=0}^{N_1-1}\boldsymbol{A}^i\,\boldsymbol{B} \\ \vdots \\ \sum\limits_{i=0}^{N_2-1}\boldsymbol{A}^i\,\mathbf{B} \end{bmatrix} \in\mathbb{R}^{n\cdot(N_2-N_1+1)\,\times\,l} \tag{8}$$

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\Lambda}(N_1) & \boldsymbol{\Lambda}(N_1-1) & \dots & \boldsymbol{\Lambda}(N_1-N_\mathrm{u}+1) \\ \boldsymbol{\Lambda}(N_1+1) & \boldsymbol{\Lambda}(N_1) & \dots & \boldsymbol{\Lambda}(N_1-N_\mathrm{u}+2) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Lambda}(N_2) & \boldsymbol{\Lambda}(N_2-1) & \dots & \boldsymbol{\Lambda}(N_2-N_\mathrm{u}+1) \end{bmatrix} \in \mathbb{R}^{n\cdot(N_2-N_1+1)\,\times\, l\cdot N_\mathrm{u}} \tag{9}$$

$$\boldsymbol{\Lambda}(i) = \begin{cases} \sum\limits_{j=0}^{i-1} \boldsymbol{A}^j\,\boldsymbol{B} & ,i \geq 1 \\ \mathbf{0} & ,i < 1 \end{cases} \in \mathbb{R}^{n\,\times\, l}\,. \tag{10}$$

As a simplification, we introduce the horizon N, which defines both the prediction horizon and the control horizon. Neglect any influence of a system delay and set $N_1 = 1$. As a result use

$$N = N_2 - N_1 + 1 = N_2 = N_\mathrm{u}\,. \tag{11}$$

a) Complete the provided function template
`[Gamma,Psi,Upsilon,Theta] = setupPredictionMatrices(A,B,C,N)`
to calculate the four matrices $\boldsymbol{\Gamma}$, $\boldsymbol{\Psi}$, $\boldsymbol{\Upsilon}$, and $\boldsymbol{\Theta}$ for a generic discrete-time linear system with steate-space matrices $\boldsymbol{A}, \boldsymbol{B}$, and $\boldsymbol{C}$ and for a given horizon N. As a starting point, the implementation of $\boldsymbol{\Gamma}$ and $\boldsymbol{\Psi}$ are provided in the function template.

b) Use the provided function `[sysC, linearizationPt] = getLinearModel(u_vtg, u_egr)` to derive a continuous-time linearized model of the ROM at $[u_\mathrm{vtg}; u_\mathrm{egr}] = [0.5; 0.5]$. Name the resulting linearized model `sysC` and the resulting operating point `linearizationPt`.

c) Discretize the continuous-time model with a sampling time $T_s = 0.05$ s in order to derive the discrete system `sysD` with state-space matrices $\boldsymbol{A}, \boldsymbol{B}$, and $\boldsymbol{C}$. Use the Matlab function `c2d()` with the `'zoh'` method.

d) Use the derived discrete-time state-space matrices and a horizon $N = 50$ to calculate the prediction matrices with the function from Exercise 1 a). Make sure the dimensions of your matrices are correct. *Hint: For debugging, you can use generic, simple, low-dimensional state-space matrices and a small horizon, e.g., $N = 3$.*

Exercise 2 (MPC Control Law)

In this exercise, you derive the optimal control law K_MPC. In order to calculate the optimal control rate trajectory $\boldsymbol{\Delta u}(k|k)^*$, we insert Eq. (5) into Eq. (1), calculate its derivative with respect to $\boldsymbol{\Delta u}(k|k)$, and set it to zero. Solving the resulting equation for $\boldsymbol{\Delta u}(k|k)$ gives the optimal control rate trajectory

$$\boldsymbol{\Delta u}(\cdot|k)^* = \left(\boldsymbol{\Theta}^T\boldsymbol{\Gamma}^T\boldsymbol{Q}\boldsymbol{\Gamma}\boldsymbol{\Theta} + \boldsymbol{R}\right)^{-1}\boldsymbol{\Theta}^T\boldsymbol{\Gamma}^T\boldsymbol{Q}\,\boldsymbol{e}\,(\cdot|k)\ , \tag{12}$$

$$\boldsymbol{e}\,(\cdot|k) = \boldsymbol{r}(\cdot|k) - \boldsymbol{\Gamma}\Big(\boldsymbol{\Psi}\,\boldsymbol{x}(k) + \boldsymbol{\Upsilon}\,\boldsymbol{u}(k-1)\Big)\,. \tag{13}$$

The free control error trajectory $\boldsymbol{e}\,(\cdot|k)$ represents the predicted control error for $\boldsymbol{\Delta u}(\cdot|k) = \mathbf{0}$. The extended error weighting matrix $\boldsymbol{Q}$ and the extended control weighting matrix $\boldsymbol{R}$ are defined as

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{N_1} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \boldsymbol{Q}_{N_2} \end{bmatrix} \in \mathbb{R}^{m\cdot(N_2-N_1+1)\,\times\, m\cdot(N_2-N_1+1)} \tag{14}$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \boldsymbol{R}_{N_\mathrm{u}-1} \end{bmatrix} \in \mathbb{R}^{l\cdot N_\mathrm{u}\,\times\, l\cdot N_\mathrm{u}}\,. \tag{15}$$

For the controller, we only need the first value of the optimal control rate trajectory. By comparing Eqs. (2) and (12) we get

$$\boldsymbol{K}_{\text{MPC}} = [\boldsymbol{I}\ \boldsymbol{0}\ \ldots \boldsymbol{0}]\ \left(\boldsymbol{\Theta}^T \boldsymbol{\Gamma}^T \boldsymbol{Q} \boldsymbol{\Gamma} \boldsymbol{\Theta} + \boldsymbol{R}\right)^{-1} \boldsymbol{\Theta}^T \boldsymbol{\Gamma}^T \boldsymbol{Q}\,, \tag{16}$$

where $\boldsymbol{I}$ is the identity matrix

$$\boldsymbol{I} \in \mathbb{R}^{l \times l}\,. \tag{17}$$

Exercise Tasks:

a) For given weighting matrices $\boldsymbol{Q}_{\text{i}}$ and $\boldsymbol{R}_{\text{i}}$ calculate the extended weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, required for the calculation of $\boldsymbol{K}_{\text{MPC}}$.

$$\boldsymbol{Q}_{\text{i}} = \begin{bmatrix} 10^{-10} & 0 \\ 0 & 10 \end{bmatrix} \forall\, i \in [N_1, N_2] \qquad \boldsymbol{R}_{\text{i}} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \forall\, i \in [0, N_{\text{u}} - 1] \tag{18}$$

Hint: Instead of for-loops, you can use the Matlab command `R = kron(eye(...),R_i);`.

b) Calculate $\boldsymbol{K}_{\text{MPC}}$.

Exercise 3 (MPC Implementation in Simulink)

Now we are ready to implement the derived matrices in Simulink and use them in a model predictive controller. The control output $\boldsymbol{u}(k)$ to the engine model is given by

$$\boldsymbol{u}(k)^* = \boldsymbol{u}(k-1) + \boldsymbol{K}_{\text{MPC}}\, \boldsymbol{e}\,(\cdot|k) = \boldsymbol{u}(k-1) + \boldsymbol{K}_{\text{MPC}}\, \Big(\boldsymbol{r}(\cdot|k) - \boldsymbol{\Gamma}(\boldsymbol{\Psi}\, \boldsymbol{x}(k) + \boldsymbol{\Upsilon}\, \boldsymbol{u}(k-1))\Big)\,. \tag{19}$$

The reference trajectory $\boldsymbol{r}(\cdot|k)$ is kept constant over the prediction horizon, i.e.,

$$\boldsymbol{r}(\cdot|k) = \begin{bmatrix} \boldsymbol{r}(k,k) \\ \vdots \\ \boldsymbol{r}(k,k) \end{bmatrix} \in \mathbb{R}^{m\cdot(N_2 - N_1 + 1)\,\times\,1}\,. \tag{20}$$

Exercise Tasks:

a) Open the Simulink template file `ps04_ex3_LinearModel.slx`. It contains the continuous-time linearized model `sysC` derived in Exercise 1 with full state feedback. Implement Eq. (19) in the subsystem `Linear Unconstrained MPC`.
Hints:

- In order to generate the reference trajectory $\boldsymbol{r}(\cdot|k)$ you can use the Simulink block `MATLAB Function` and the command `repmat`.
- To do matrix multiplications with the standard `Gain` block in Simulink, change its multiplication setting from `Element-wise` to `Matrix(K*u)`
- To check whether your implementation contains syntax errors or to see signal dimensions and visualize sampling times, you can use the `Update Diagram` command by pressing `Ctrl+D` .

b) Copy the subsystem `Linear Unconstrained MPC` into the Simulink file `ps04_ex3_ROM.slx`. This model contains the nonlinear ROM with full state feedback. Run both models using the Matlab script `ps04_run_ex3.m` and compare the results. Explain why the simulation with the ROM model has a steady-state error.