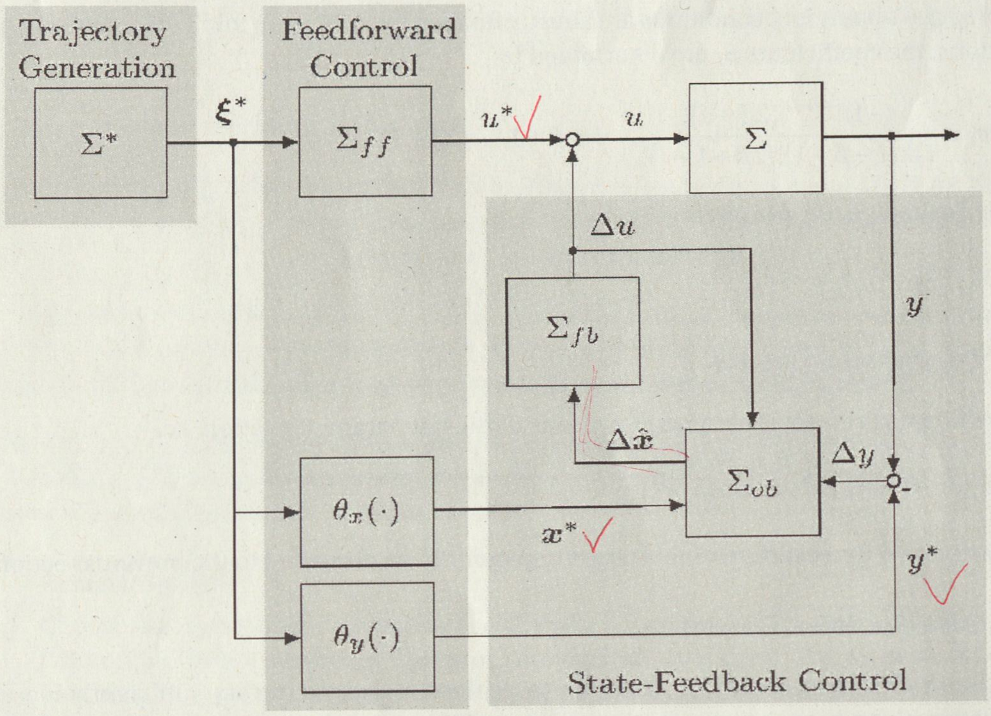


Figure 5.4: Schematic structure of the two degrees of freedom control.

5.2 Flatness-Based 2DOF Control

5.2.1 Trajectory Generation

The objective of this exercise is to implement a flatness-based trajectory-tracking controller to stabilize the imposed trajectory y^* in Simulink. Our main goal is to achieve a transition between two steady states, i.e., from $(x_R(t_0), u_R(t_0))$ to $(x_R(t_1), u_R(t_1))$ with $0 \leq t_0 < t_1$ of the measured output $y = \varphi_P + \varphi_B$. The target trajectory for the flat output is designed to fulfill the boundary conditions

$$\lambda^*(t_0) = \lambda_R(t_0), \qquad \dot{\lambda}^*(t_0) = \cdots = \lambda^{*(\beta)}(t_0) = 0, \tag{5.15a}$$

$$\lambda^*(t_1) = \lambda_R(t_1), \qquad \dot{\lambda}^*(t_1) = \cdots = \lambda^{*(\beta)}(t_1) = 0. \tag{5.15b}$$

A suitable β times differentiable trajectory for the flat output $\lambda^* \in C^\beta([t_0, t_1])$ is defined by [2]

$$\lambda^*(t) = \lambda_R(t_0) + (\lambda_R(t_1) - \lambda_R(t_0)) \sum_{i=\beta+1}^{2\beta+1} p_i \left(\frac{t - t_0}{t_1 - t_0}\right)^i$$

so that

$$\Sigma^* : \xi^* = \begin{bmatrix} \lambda^* \\ \dot{\lambda}^* \\ \vdots \\ \frac{\mathrm{d}^\beta}{\mathrm{d}t^\beta}\lambda^* \end{bmatrix}, \quad t \in [t_0, t_1]. \tag{5.16}$$

For $t = t_0$, $\beta + 1$ boundary conditions are already fulfilled by (5.15a). To fulfill the remaining boundary conditions, the coefficients p_i are determined by

$$p_i = \frac{(-1)^{i-\beta-1}(2\beta+1)!}{i\beta!(i-\beta-1)!(2\beta+1-i)!}, \quad i = \beta+1,\dots,2\beta+1 \tag{5.17}$$

and the desired system output is given by

$$y^* = \theta_y\left(\lambda^*, \dot{\lambda}^*, \dots, \lambda^{*(\gamma)}\right). \tag{5.18}$$

5.2.2 Feedforward Control

The feedforward control is obtained by means of the flat parameterization, i.e.

$$\Sigma_{ff}:\ u^* = \theta_u\left(\lambda^*, \dot{\lambda}^*, \dots, \lambda^{*(\beta)}\right) \tag{5.19}$$

along with the β times differentiable target trajectory $\boldsymbol{\xi}^*$ as discussed in the previous section.

5.2.3 State-Feedback Control

Since the feedforward control is assumed to achieve a state trajectory sufficiently close to $\boldsymbol{x}^*$ a linearization along this nominal trajectory yields an LTV system that constitutes a good approximation of the time evolution of the error $\Delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}^*$. Based on this LTV system description, i.e.,

$$\Delta\dot{\boldsymbol{x}} = A(t)\Delta\boldsymbol{x} + \boldsymbol{b}(t)\Delta u, \tag{5.20a}$$

$$\Delta y = \boldsymbol{c}^\top(t)\Delta\boldsymbol{x}, \tag{5.20b}$$

where $A(t) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}|_{(\boldsymbol{x}^*,u^*)}$, $\boldsymbol{b}(t) = \frac{\partial \boldsymbol{g}}{\partial u}|_{(\boldsymbol{x}^*,u^*)}$, $\boldsymbol{c}^\top(t) = \frac{\partial h}{\partial \boldsymbol{x}}|_{(\boldsymbol{x}^*,u^*)}$, the flatness-based feedforward control is extended with a trajectory-tracking state-feedback PI controller that compensates for model uncertainties and disturbances. To this end, the state (difference) vector is augmented with the integral of the relative tracking error the time derivative of which is given by $\Delta\dot{x}_i = -\boldsymbol{c}^\top(t)\Delta\boldsymbol{x}$. The augmented LTV system reads

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\Delta\boldsymbol{x}\\ \Delta x_i\end{bmatrix} = \underbrace{\begin{bmatrix}A(t) & \boldsymbol{0}\\ -\boldsymbol{c}^\top(t) & 0\end{bmatrix}}_{\bar{A}}\underbrace{\begin{bmatrix}\Delta\boldsymbol{x}\\ \Delta x_i\end{bmatrix}}_{\Delta\bar{x}} + \underbrace{\begin{bmatrix}\boldsymbol{b}(t)\\ 0\end{bmatrix}}_{\bar{\boldsymbol{b}}}\Delta u. \tag{5.21}$$

Based on this, the feedback controller is expressed mathematically as

$$\Sigma_{fb}:\ \Delta u = -\boldsymbol{k}^\top(t)\begin{bmatrix}\Delta\boldsymbol{x}\\ \Delta x_i\end{bmatrix}. \tag{5.22}$$

5.2.4 State (Difference) Observer

Since the state-feedback controller (5.22) needs all system states, the state information must be reconstructed by means of a nonlinear Luenberger-type observer of the form

$$\Sigma_{ob}:\begin{cases}\Delta\dot{\hat{\boldsymbol{x}}} = \boldsymbol{f}(\hat{\boldsymbol{x}}) - \boldsymbol{f}(\boldsymbol{x}^*) + \boldsymbol{g}(\hat{\boldsymbol{x}})u - \boldsymbol{g}(\boldsymbol{x}^*)u^*\\ \qquad + \boldsymbol{l}(t)(\Delta y - \Delta\hat{y}),\\ \Delta\hat{y} = h(\hat{\boldsymbol{x}}) - h(\boldsymbol{x}^*).\end{cases} \tag{5.23}$$

$\delta U = U \,\hat{\mp}\, \text{ref}.$

$U = \delta U + \text{ref}$

Similar reasoning as in the state feedback controller is applied to argue that the observer gain $\boldsymbol{l}(t)$ can be calculated based on the LTV system (5.20).

Exercise 5.2 (Flatness-Based 2DOF Control).

(i) *Implement the trajectory generator* Σ^* *according to* (5.16) *in Fig. 5.4 as a MATLAB function block in Simulink with inputs* t *and* `param` *to calculate* y^*. *The block's output is the vector* $\boldsymbol{\xi}^* = [y^*, \dot{y}^*, \dots, y^{*(\beta)}]^\top$. *Make sure that* $\boldsymbol{\xi}^*$ *is well-defined for* $t > T$.

(ii) *Implement the feedforward controller* Σ_{ff} *according to* (5.19) *in another MATLAB function block, which calculates the feedforward control to approximately realize the desired target trajectory. Simulate the open-loop system with the simulation model as plant* Σ.

(iii) *Implement the flatness-based state-feedback controller* Σ_{fb} *according to* (5.22) *in a MATLAB function block using the Ackerman formula to obtain* $\boldsymbol{k}^\top(t)$ *[2]. Simulate the closed-loop system with the simulation model as plant* Σ.

> **Remark 5.1: (iii)**
>
> Choose the eigenvalues imposed by the Hurwitz polynomial so that the controller acts as fast as possible but also keeps the input bounded by $-10\,\text{V} \le u \le 10\,\text{V}$. Furthermore, since the flatness-based controller assumes knowledge of the entire state vector use the states from the simulation model.

(iv) *Implement the flatness-based nonlinear Luenberger-type observer* Σ_{ob} *according to* (5.23) *in a MATLAB function block using the Ackerman formula to obtain* $\boldsymbol{l}(t)$ *[2]. Simulate the closed-loop system with the simulation model as plant* Σ *and replace* $\boldsymbol{x}$ *with* $\hat{\boldsymbol{x}}$ *in* Σ_{fb}.

(v) *Evaluate the flatness-based 2DOF controller and observer at the real rotary flexible joint setup.*

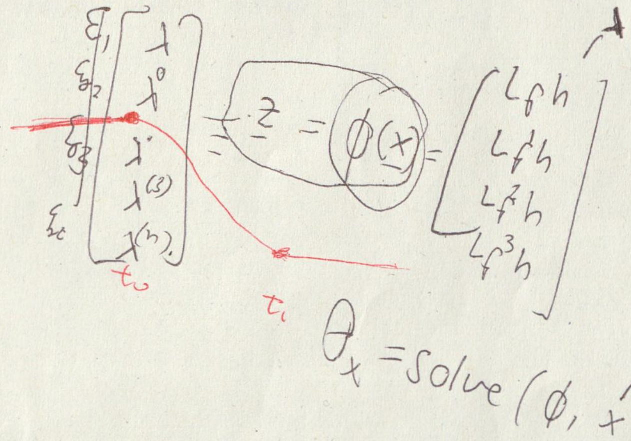
References

[1] Petar V. Kokotovic, Hassan K. Khalil, and John O'Reilly. *Singular Perturbation Methods in Control: Analysis and Design.* Classics in Applied Mathematics 25. SIAM, 1986 (cit. on p. 5).

[2] T. Meurer. „Nonlinear Control Systems". In: `https://www.control.tf.uni-kiel.de/en/teaching/summer-term/nonlinear-control-systems` (2020) (cit. on pp. 6, 7, 9).

[3] T. Meurer. „Rigid Body Dynamics and Robotics". In: `https://www.control.tf.uni-kiel.de/en/teaching/winter-term/rigid-body-dynamics-robotics` (2020) (cit. on p. 5).

[4] Quanser. *Quansers's Rotary Flexible Joint.* `http://www.quanser.com/Images/Products/494-200-600.jpg`. [Online; accessed 02-December-2016]. 2016 (cit. on p. 2).



$$\begin{bmatrix} y \\ y^{0} \\ \dot{y} \\ y^{(3)} \\ y^{(3)} \end{bmatrix} = z = \phi(x) = \begin{bmatrix} L_f h \\ L_f^1 h \\ L_f^2 h \\ L_f^3 h \end{bmatrix}^{-1}$$

$$\theta_x = \text{solve}(\phi, x) \quad \dot{\phi}_p \quad \phi_p \quad \dot{\phi}_B \quad \phi_B$$

$$z = \phi(x)$$

$$\nexists \; x = \phi^{-1}(z)$$

$$\theta_x$$