A Appendices

A.1 Comparison of Feedforward and Feedback Control for Reference Tracking

We consider a discrete-time, linear system of first order with sampling period $T = 0.02$ seconds, output $y \in \mathbb{R}$, and input $u \in \mathbb{R}$. The system is affected by an input delay of one sample leading to state-space dynamics

$$\forall n \in \mathbb{N}, \quad x(n+1) = \begin{bmatrix} 0.5 & 1 \\ 0 & 0 \end{bmatrix} x(n) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(n),$$

where $x \in \mathbb{R}^2$ is the state vector. The system’s output is affected by measurement noise, i.e.,

$$\forall n \in \mathbb{N} \geq 0, \quad y(n) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(n) + w(n),$$

where $w(n) \in \mathbb{R}$ is measurement noise that is drawn from a normal distribution with zero-mean and variance $10^{-4}$, i.e.,

$$\forall n \in \mathbb{N} \geq 0, \quad w(n) \sim \mathcal{N}(0, 10^{-4}).$$

The task consists of having the output $y$ follow the reference $r \in \mathbb{R}$ over a finite horizon of $N = 100$ samples with

$$\forall n \in [1, N], \quad r(n) = \sin(2\pi T n).$$

The feedforward control strategy consists of applying an input trajectory

$$u_{\text{FF}} := [u_{\text{FF}}(0) \quad u_{\text{FF}}(1) \ldots \quad u_{\text{FF}}(N-1)]^T.$$

The input values are determined by optimization such that the squared tracking error is minimized, i.e.,

$$u_{\text{FF}} = \arg\min_u \sum_{n=1}^{N} [r(n) - y(n)]^2.$$

The feedback control strategy consists of a generic, non-linear function to ensure that performance is not limited by the structure of the feedback law. In particular, the input values $u_{\text{FB}}$ are computed as the sum of ten polynomials of tenth order, to which the current and nine previous error samples serve as inputs, i.e., $\forall n \in [1, N],$

$$u_{\text{FB}}(n) = \sum_{i=1}^{10} \sum_{j=1}^{10} k_{ij} [r(n) - y(n)]^j.$$

The set of feedback parameters $\mathcal{K} = \{k_{ij} \mid i, j \in [1, 10]\}$ is determined via optimization such that the squared tracking error is minimized, i.e.,

$$\mathcal{K} = \arg\min_{\mathcal{K}} \sum_{n=1}^{N} [r(n) - y(n)]^2.$$
A.2 Reference Trajectories

The first reference \( r_1 \in \mathbb{R}^{25} \) is given by, \( \forall n \in [1, 25] \),
\[
[r_1]_n = 75 \sin(2\pi T n) .
\] (9)

The second reference \( r_2 \in \mathbb{R}^{50} \) is given by, \( \forall n \in [1, 50] \),
\[
[r_2]_n = \begin{cases} 
57 \sin(1.38\pi T n) & \forall n \leq 17 \\
57 & \forall 17 < n \leq 31 \ \text{[°]}.
\end{cases}
\] (10)

The third reference \( r_3 \in \mathbb{R}^{71} \) is given by, \( \forall n \in [1, 71] \),
\[
[r_3]_n = \begin{cases} 
-20 \sin(1.5\pi T n) & \forall n \leq 16 \\
-20 & \forall 16 < n \leq 31 \\
-20 \sin(1.5\pi T (n - 11)) & \forall 31 < n \leq 43 \ \text{[°]}.
\end{cases}
\] (11)

A.3 Feedback Control of the TWIPR

Consider the dynamics of the TWIPR moving along a straight line. The robot has two degrees of freedom, namely, the pitch angle \( \Theta \in \mathbb{R} \) and the position \( s \in \mathbb{R} \). The state vector follows with
\[
x = [\Theta \ \dot{\Theta} \ s \ \dot{s}]^T .
\] (12)

The motor torque serves as input variable and is denoted by \( u \in \mathbb{R} \). To stabilize the TWIPR in its upright equilibrium, the nonlinear dynamics are approximated by a linear, discrete-time model with state vector \( x \in \mathbb{R}^4 \) of the form
\[
\forall n \in \mathbb{N}, \ x(n + 1) = Ax(n) + Bu(n)
\] (13)

using a sampling period of \( T = 0.02 \) seconds. The stabilizing control input \( u_C \in \mathbb{R} \) is computed by linear state feedback of the form
\[
\forall n \in \mathbb{N}, \ u_C(n) = -Kx(n) ,
\] (14)

where the feedback matrix \( K \) is designed by LQR [1].

To track the desired reference maneuvers, the feedback input \( u_C \) is superposed by a learned feedforward input \( u_L \) leading to the overall input
\[
\forall n \in \mathbb{N}, \ u(n) = u_C(n) + u_L(n) .
\] (15)

A.4 Policy Gradient Implementation

In this section, we briefly outline the implementation details of the finite-difference policy gradient method that was used as a baseline comparison in Section 4.2. For a detailed discussion of the method and its implementation,
see [2]. The finite-difference gradient estimation was chosen because this method is expected to be highly efficient due to the deterministic nature of the simulations, see [2].

In order to apply the policy gradient scheme to the learning task of Section 4.2, the policy is defined as the input trajectory $u_j$, and the reward of a trial reward is defined as

$$ R_t := \frac{1}{2} (r - y_j)^\top (r - y_j) . $$

On each iteration, the policy is updated by

$$ u_{j+1} = u_j + \alpha \nabla R_j , $$

where $\nabla R_j$ is an estimate of the reward’s gradient with respect to the input trajectory, and $\alpha \in \mathbb{R}$ is a step-size. To estimate the gradient, $W \in \mathbb{N}$ roll-out trials with the perturbed policies $u_j + \Delta w$ are performed, and the gradient is determined by least-squares estimation, as detailed in [2]. In the simulations, the step-size was chosen as $\alpha = 50$, one roll-out per trial, i.e., $W = 1$, was used, and the policy permutations were drawn according to

$$ \Delta w \sim \mathcal{N}(0, 0.001I) . $$

In contrast to the method proposed in this paper, the parameters of the policy gradient scheme had to be tuned manually and were chosen to yield a satisfying trade-off between a fast speed of learning and robust convergence for all three reference trajectories.

References

[1] Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos. *Optimal control*. John Wiley & Sons, 2012.

[2] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, October 2006.