# Real-Time Substrate Feed Optimization of Anaerobic Co-Digestion Plants

Gaida, D.

# Chapter 2

# Multi-Objective Nonlinear Model Predictive Control

Consider a physical, time-dependent, real-world system showing deterministic behavior at any time $t \in \mathbb{R}_0^+$. Assume that the main influence on the system by its environment can be described by a finite number $n_\mathrm{u} \in \mathbb{N}_0$ of physical values. They are called the input values of the system. The nominal input values of the system are generated by a function of time $\boldsymbol{u} : \mathbb{R}_0^+ \to \mathcal{U}$, which, for each time $t \geq 0$, returns the input of the system at time $t$ symbolized by $\boldsymbol{u}(t) \in \mathcal{U}$. Each input function $u_{i_\mathrm{u}}$, with $\boldsymbol{u} := (u_1, \ldots, u_{i_\mathrm{u}}, \ldots, u_{n_\mathrm{u}})^T$, returns values out of the set $\mathcal{U}_{i_\mathrm{u}} \subseteq \mathbb{R}$, $i_\mathrm{u} = 1, \ldots, n_\mathrm{u}$. The set $\mathcal{U}$ then is defined as $\mathcal{U} := (\mathcal{U}_{i_\mathrm{u}})^{n_\mathrm{u}} := \mathcal{U}_1 \times \cdots \times \mathcal{U}_{n_\mathrm{u}}$. Note that the ith input of the system is symbolized by the iterator $i_\mathrm{u} \in \{1, \ldots, n_\mathrm{u}\}$.

Those physical values which are assumed to describe the inherent behavior of the system are put inside the state of the system $\boldsymbol{x} : \mathbb{R}_0^+ \to \mathcal{X}$, with the state space $\mathcal{X} \subseteq \mathbb{R}^{n_\mathrm{x}}$ and $n_\mathrm{x} \in \mathbb{N}_0$ representing the number of physical values in the system state vector $\boldsymbol{x} \in \mathcal{X}$. The idea of the system state is that if it is known for some time $t$, then the complete physical system description at that time is known. Examples of state vector components are the position of the system in space, the temperature inside the system or the concentration of fluids or species inside the system.

The sets $\mathcal{X}$ and $\mathcal{U}$ could be generated out of state and input constraints, respectively. If the state (input) constraints are linear, then $\mathcal{X}$ ($\mathcal{U}$) is a convex set (Boyd and Vandenberghe, 2004).

To be able to predict the future trajectory of the system state $\boldsymbol{x}$ for a given input trajectory $\boldsymbol{u}$ the real-world system is described as a system of continuous-time nonlinear stochastic differential equations:

$$ {}^o\boldsymbol{x}'(t) = \boldsymbol{f}\left({}^o\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\omega}(t)\right), \qquad {}^o\boldsymbol{x}(0) = \boldsymbol{x}(0). \tag{2.1} $$

This future state vector trajectory is symbolized by the vector valued function ${}^o\boldsymbol{x} : \mathbb{R}_0^+ \to \mathcal{X}$. As eq. (2.1) is only initialized at time $t = 0$ the calculated state ${}^o\boldsymbol{x}$ is

called the open loop state, whereas "open" is symbolized by the $^o$ in front of $\boldsymbol{x}$ in ${}^o\boldsymbol{x}$. The behavior of the real-world system is approximately modeled using the real-valued smooth vector field $\boldsymbol{f} : \mathcal{X} \times \mathcal{U} \times \mathbb{R}^{n_\omega} \to \mathcal{T}\mathcal{X}$, which maps the input space of the function onto the tangent space $\mathcal{T}\mathcal{X} \subseteq \mathbb{R}^{n_x}$. To emphasize that $\boldsymbol{f}$ is only an approximation of the real system the noise process $\boldsymbol{\omega} : \mathbb{R}_0^+ \to \mathbb{R}^{n_\omega}$ is introduced as input of the system function $\boldsymbol{f}$. This noise process is used to take account for the fact that $\boldsymbol{f}$ cannot describe exactly what is happening in the real world and for possibly noisy input values $\boldsymbol{u}$. This $n_\omega \in \mathbb{N}_0$ dimensional noise process $\boldsymbol{\omega}$ is modeled as a normal distribution with zero-mean and the covariance matrix $\boldsymbol{\Psi}_{\boldsymbol{\omega}} \in \mathbb{R}^{n_\omega \times n_\omega}$, symbolized by $\boldsymbol{\omega}(t) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Psi}_{\boldsymbol{\omega}})$. We assume stationary, white noise. That is, $E\left\langle \boldsymbol{\omega}(t) \cdot \boldsymbol{\omega}^T(\tau) \right\rangle = \boldsymbol{\Psi}_{\boldsymbol{\omega}} \cdot \delta_{\mathrm{D}}(t - \tau)$, where $\delta_{\mathrm{D}}$ is the Dirac delta "function" and $E\langle\cdot\rangle$ denotes the expected value.

Given the initial state of the real system at time $t = 0$, $\boldsymbol{x}(0)$, for each $t \geq 0$ the approximate state of the system ${}^o\boldsymbol{x}(t)$ can be calculated using equation (2.1). As for $t > 0$ there is no further interaction with the real system (thus no feedback) this predictor could be very inaccurate, because it cannot be guaranteed that the predicted state values ${}^o\boldsymbol{x}(t)$ track the real state values $\boldsymbol{x}(t)$ for $t > 0$. The error between the two state vector trajectories is commonly measured by the root-mean-square error (RMSE):

$$\mathrm{RMSE}\left({}^o\boldsymbol{x}(t), \boldsymbol{x}(t)\right) := \left\| \left({}^o\boldsymbol{x}(t) - \boldsymbol{x}(t)\right) \cdot \left({}^o\boldsymbol{x}(t) - \boldsymbol{x}(t)\right)^T \right\|_2,$$

whose value must be kept arbitrarily small. Better predictors than eq. (2.1) are presented later in Chapter 4.

The task at hand is to find an optimal input function $\boldsymbol{u}^* : \mathbb{R}_0^+ \to \mathcal{U}$, such that an objective function

$$\widetilde{\boldsymbol{J}} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_\mathrm{o}} \tag{2.2}$$

gets minimized for all $t \in [0, \infty)$, with the number of objectives $n_\mathrm{o} \in \mathbb{N}_0$ and the constraint ${}^o\boldsymbol{x}(t) \in \mathcal{X}\ \forall t \in [0, \infty)$. The vector function $\widetilde{\boldsymbol{J}} := \left(\widetilde{J}_1, \ldots, \widetilde{J}_{n_\mathrm{o}}\right)^T$ consists out of $n_\mathrm{o}$ scalar-valued objective functions

$$\widetilde{J}_{i_\mathrm{o}} : \mathcal{X} \times \mathcal{U} \to \mathbb{R} \tag{2.3}$$

with $i_\mathrm{o} = 1, \ldots, n_\mathrm{o}$. The problem can be formulated as:

$$\begin{aligned} & \text{minimize}_{\boldsymbol{u}}\ \widetilde{\boldsymbol{J}}\left({}^o\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \\ \text{subject to}\quad & {}^o\boldsymbol{x}'(t) = \boldsymbol{f}\left({}^o\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{0}\right), && {}^o\boldsymbol{x}(0) = \boldsymbol{x}(0), \\ & {}^o\boldsymbol{x}(t) \in \mathcal{X}, && \forall t \geq 0, \\ & \boldsymbol{u} : [0, \infty) \to \mathcal{U}. \end{aligned} \tag{2.4}$$

The minimum of a vector function is not defined in general, such that it has to be defined what is meant by minimizing the objective function $\widetilde{\boldsymbol{J}}$. To minimize each objective

function $\widetilde{J}_{i_o}$ separately is often not well-suited, because most of the time the objective functions are conflicting. Two objective functions are conflicting, if and only if the set of optimal solutions of one objective function does not overlap with the set of optimal solutions of the other objective function. To simplify things, at first the optimal control problem for the case $n_o = 1$ is handled in Section 2.1 before the general case for $n_o > 1$ is solved in Section 2.3. To minimize the objective function $\widetilde{\boldsymbol{J}}$ properly, concepts from multi-objective optimization are used, which are recapped in Section 2.2.
Since minimizing $\widetilde{\boldsymbol{J}}$ in choosing the optimal input $\boldsymbol{u}$ for all $t \in [0, \infty)$ is in general a hard problem, in practice a heuristic technique named multi-objective nonlinear model predictive control (MONMPC) shall be used that will be introduced in Section 2.1.

## 2.1 Case I: Number of Objectives $n_o = 1$

For $n_o = 1$ the objective function reduces to the scalar-valued objective function $\widetilde{J}_1$, defined in equation (2.3), such that the minimum of the objective function is well defined. Thus, for this case problem (2.4) results in the problem formulation:

$$\begin{aligned} & \boldsymbol{u}^* := \arg\min_{\boldsymbol{u}} \widetilde{J}_1 \left( {}^o\boldsymbol{x}(t), \boldsymbol{u}(t) \right) \\ \text{subject to} \quad & {}^o\boldsymbol{x}'(t) = \boldsymbol{f} \left( {}^o\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{0} \right), && {}^o\boldsymbol{x}\left(0\right) = \boldsymbol{x}\left(0\right), \\ & {}^o\boldsymbol{x}(t) \in \mathcal{X}, && \forall t \geq 0, \\ & \boldsymbol{u} : [0, \infty) \to \mathcal{U}. \end{aligned} \tag{2.5}$$

Problem (2.5) states that we try to find the optimal input function $\boldsymbol{u}^*$ for system (2.1) which for all time $t \geq 0$ minimizes the objective function $\widetilde{J}_1$.
According to Diehl et al. (2006a) there are three basic approaches to address optimal control problems:

- Dynamic Programming Methods
- Indirect Methods
- Direct Methods

Direct methods can be divided into single shooting, collocation and multiple shooting. The approach followed in this thesis belongs to the method of single shooting. An example of multiple shooting can be found in Diehl et al. (2002, 2003). For collocation see Biegler (1984).
Finding a closed solution for this problem using dynamic programming or indirect methods can be very difficult or even impossible for some systems $\boldsymbol{f}$ and objective functions $\widetilde{J}_1$ (Findeisen et al., 2003, Diehl et al., 2006a). From a practical viewpoint a closed solution is often not needed, because model mismatch and disturbances acting on the real-world system (both modeled by the noise process $\boldsymbol{\omega}$) will make the solution for $t > t_0 > 0$, with $t_0 \in \mathbb{R}^+$ inaccurate.

Therefore, using nonlinear model predictive control (NMPC), problem (2.5) is only solved over a finite horizon. This finite horizon is called the prediction horizon $T_\mathrm{p} \in \mathbb{R}^+$. Having solved problem (2.5) over the prediction horizon $T_\mathrm{p} > 0$, the optimal input is applied to the system for a short time period, named (control) sampling time $\delta$. After the sampling time $\delta \in \mathbb{R}^+$ has passed by, problem (2.5) is solved over the prediction horizon again. Therefore $T_\mathrm{p}$ is moved forward by time $\delta$ and the new solution is applied again for timespan $\delta$ and so on. Therefore, problem (2.5) is solved iteratively over the moving horizon $T_\mathrm{p}$, resulting in an approximate solution to problem (2.5).

For $T_\mathrm{p} \to \infty$ and $\delta \to 0$ the found approximate solution will converge towards the optimal solution $\boldsymbol{u}^*$, provided it exists.

The found optimal input functions at each iteration cannot be equal to the input values applied to the system, because they are only defined over the time period $T_\mathrm{p}$. Thus, the found optimal inputs are called open loop input functions. The applied input function to the system is named closed-loop input.

The NMPC approach has at least two justifications:

- As there may be no closed solution to problem (2.5), the approach using NMPC will at least return an approximate solution. The degree of approximation can be defined by the user in choosing appropriate values for the prediction horizon $T_\mathrm{p}$ and sampling time $\delta$.
- Because of model mismatch and disturbances a solution has to be calculated repeatedly, anyway. Therefore, there is no need to spend time in solving problem (2.5) over an infinite horizon.

Next to prediction horizon $T_\mathrm{p}$ and sampling time $\delta$, the term control horizon $T_\mathrm{c} \in \mathbb{R}^+$ with $T_\mathrm{p} \geq T_\mathrm{c} \geq \delta$ is used as well. Using these terms it can be stated that for each sampling instance $k = 0, 1, 2, \ldots$ at time

$$t_k := k \cdot \delta \tag{2.6}$$

NMPC tries to find the optimal open loop input function ${}^o\boldsymbol{u}_k^* : [t_k, t_k + T_\mathrm{p}] \to \mathcal{U}$ which minimizes the objective function $\widetilde{J}_1$ over the interval $[t_k, t_k + T_\mathrm{p}]$, defined by the prediction horizon $T_\mathrm{p}$. Open loop input functions are denoted by ${}^o\boldsymbol{u} : [t_k, t_k + T_\mathrm{p}] \to \mathcal{U}$. During the time period $[t_k, t_k + T_\mathrm{c}]$ the system input ${}^o\boldsymbol{u}$ may be changed, after that it is kept constant at the value ${}^o\boldsymbol{u}\,(t_k + T_\mathrm{c})$, see eq. (2.7). Using these terms, problem

(2.5) can be formulated approximately as:

For each $k = 0, 1, 2, \ldots$ set $t_k = k \cdot \delta$ and solve:

$$
\begin{aligned}
& {}^{o}\boldsymbol{u}_k^* := \arg\min_{{}^{o}\boldsymbol{u}} \widetilde{J}_1\left({}^{o}\boldsymbol{x}(\tau), {}^{o}\boldsymbol{u}(\tau)\right) \\
\text{subject to} \quad & {}^{o}\boldsymbol{x}'(\tau) = \boldsymbol{f}\left({}^{o}\boldsymbol{x}(\tau), {}^{o}\boldsymbol{u}(\tau), \boldsymbol{0}\right), && {}^{o}\boldsymbol{x}\left(t_k\right) = \boldsymbol{x}\left(t_k\right), \\
& {}^{o}\boldsymbol{x}(\tau) \in \mathcal{X}, && \forall \tau \in [t_k, t_k + T_\mathrm{p}], \\
& {}^{o}\boldsymbol{u} : [t_k, t_k + T_\mathrm{c}] \to \mathcal{U}, \\
& {}^{o}\boldsymbol{u}(\tau) = {}^{o}\boldsymbol{u}\left(t_k + T_\mathrm{c}\right), && \forall \tau \in (t_k + T_\mathrm{c}, t_k + T_\mathrm{p}].
\end{aligned}
\tag{2.7}
$$

Here it is assumed that for each discrete time $t_k$ the state $\boldsymbol{x}\left(t_k\right)$ of the real system can be observed. As the system state often can not be observed directly, it often has to be estimated for each time $t_k$, see Chapter 4.

The resulting optimal input ${}^{o}\boldsymbol{u}_k^*$ is applied for the interval $[t_k, t_k + \delta)$ to the system:

$$
\boldsymbol{u}(t) = {}^{o}\boldsymbol{u}_k^*(t), \qquad t \in [t_k, t_k + \delta) \tag{2.8}
$$

and the optimization problem in (2.7) is solved again for the next value of $k$. Note that we assume here that problem (2.7) can be solved in no time. If we would take into account, that a method solving problem (2.7) for one $k$ needs a certain runtime, then the application of the optimal input to the real system according to equation (2.8) will be delayed by the timespan of the runtime.

To simplify the NMPC problem (2.7) further, the open loop input ${}^{o}\boldsymbol{u}$ is often restricted to be a piecewise constant function. Therefore, given the open loop input function ${}^{o}\boldsymbol{u} := \left({}^{o}u_1, \ldots, {}^{o}u_{i_\mathrm{u}}, \ldots, {}^{o}u_{n_\mathrm{u}}\right)^T$, each component ${}^{o}u_{i_\mathrm{u}} : \mathbb{R}^+ \to \mathcal{U}_{i_\mathrm{u}}$ is a piecewise constant function. The duration of each constant period of the function ${}^{o}u_{i_\mathrm{u}}$ is given by the sampling time $\delta$. In problem (2.7) it is defined that the open loop input ${}^{o}\boldsymbol{u}$ should only be variable over the control horizon $T_\mathrm{c}$. Then, the number of steps of the piecewise constant function over the control horizon $T_\mathrm{c}$ is given by $s_\mathrm{c} := \frac{T_\mathrm{c}}{\delta} \in \mathbb{N}_0$. Thus, each such piecewise constant function ${}^{o}u_{i_\mathrm{u}}$ of the $i_\mathrm{u} = 1, \ldots, n_\mathrm{u}$ inputs can be described by $s_\mathrm{c}$ values given in the vector $\boldsymbol{u}_{i_\mathrm{u}} := \left(u_{i_\mathrm{u},1}, \ldots, u_{i_\mathrm{u},s_\mathrm{c}}\right)^T \in \left(\mathcal{U}_{i_\mathrm{u}}\right)^{s_\mathrm{c}}$ with the $i = 1, \ldots, s_\mathrm{c}$ amplitudes $u_{i_\mathrm{u},i} \in \mathcal{U}_{i_\mathrm{u}}$ as given in equation (2.9). This kind of parametrization is called control vector parametrization (Schlegel et al., 2005). An example of such a piecewise constant input function is depicted in Figure 2.1.

$$
\begin{aligned}
{}^{o}u_{i_\mathrm{u}}\left(t_k + \tau\right) &:= \begin{cases} \sum_{i=1}^{s_\mathrm{c}} u_{i_\mathrm{u},i} \cdot \mathrm{rect}\left(\tau - (i-1)\cdot\delta\right) & 0 \le \tau < T_\mathrm{c} \\ u_{i_\mathrm{u},s_\mathrm{c}} & T_\mathrm{c} \le \tau \le T_\mathrm{p} \end{cases} \\
\mathrm{rect}(\tau) &:= \begin{cases} 1 & 0 \le \tau < \delta \\ 0 & \text{else} \end{cases}
\end{aligned}
\tag{2.9}
$$

Furthermore, we define

$$\begin{aligned} \underline{\boldsymbol{u}} &:= \left(\boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_{i_\mathrm{u}}^T, \ldots, \boldsymbol{u}_{n_\mathrm{u}}^T\right)^T \in \mathcal{U}_\mathcal{F}, \text{with} \\ \mathcal{U}_\mathcal{F} &:= \left(\mathcal{U}_1\right)^{s_\mathrm{c}} \times \cdots \times \left(\mathcal{U}_{i_\mathrm{u}}\right)^{s_\mathrm{c}} \times \cdots \times \left(\mathcal{U}_{n_\mathrm{u}}\right)^{s_\mathrm{c}}, \end{aligned} \tag{2.10}$$

containing all $s_\mathrm{c}$ amplitudes of each of the $n_\mathrm{u}$ inputs, which therefore completely describes the piecewise constant open loop input function ${}^o\boldsymbol{u}$. Using this simplification

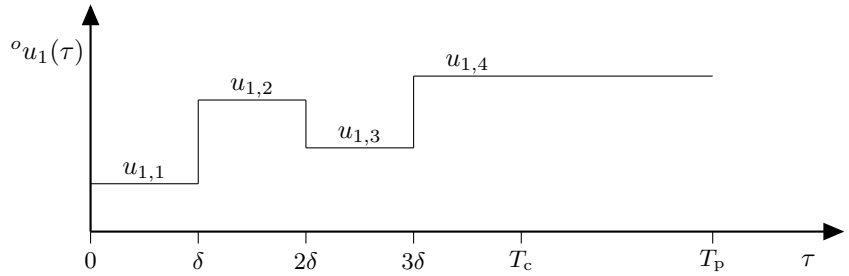


**Figure 2.1:** Example of a piecewise constant open loop input function ${}^o u_1$ for $i_\mathrm{u} = 1$ and number of steps $s_\mathrm{c} = 4$.

the problem in finding a continuous function ${}^o\boldsymbol{u}$ over the interval $[t_k, t_k + T_\mathrm{c}]$ was transformed into the simpler problem of finding a vector $\underline{\boldsymbol{u}}$ containing only $n_\mathrm{v} := s_\mathrm{c} \cdot n_\mathrm{u} \in \mathbb{N}_0$ components, i.e., the amplitudes of the piecewise constant inputs. This means, that the argument of the objective function $\widetilde{\boldsymbol{J}}$ is changed from a function ${}^o\boldsymbol{u}$ to a vector $\underline{\boldsymbol{u}}$ with $n_\mathrm{u}$ elements. From now on this vector $\underline{\boldsymbol{u}}$ is called the vector of optimization or decision variables, containing $n_\mathrm{u}$ optimization variables.
The transformation between the vector of optimization variables $\underline{\boldsymbol{u}}$ and the open loop input function ${}^o\boldsymbol{u}$ is given by the function

$$\boldsymbol{f}_\mathcal{U} : \mathcal{U}_\mathcal{F} \to \mathcal{U} \tag{2.11}$$

which returns the piecewise constant function

$${}^o\boldsymbol{u} : [t_k, t_k + T_\mathrm{p}] \mapsto \boldsymbol{f}_\mathcal{U}\left(\underline{\boldsymbol{u}}\right) \tag{2.12}$$

given the corresponding vector of optimization variables $\underline{\boldsymbol{u}}$ using equation (2.9).
To account for this transformation in optimization problem (2.7) a new objective function with a different domain $\boldsymbol{J} : \mathcal{X} \times \mathcal{U}_\mathcal{F} \to \mathbb{R}^{n_\mathrm{o}}$ has to be introduced. Using equation (2.11) the objective function $\boldsymbol{J}$ is defined by the following equation:

$$\widetilde{\boldsymbol{J}}\left({}^o\boldsymbol{x}(\tau), {}^o\boldsymbol{u}(\tau)\right) \overset{(2.11)}{=} \widetilde{\boldsymbol{J}}\left({}^o\boldsymbol{x}(\tau), \boldsymbol{f}_\mathcal{U}\left(\underline{\boldsymbol{u}}\right)\right) =: \boldsymbol{J}\left({}^o\boldsymbol{x}(\tau), \underline{\boldsymbol{u}}\right) \qquad \forall \tau \in [t_k, t_k + T_\mathrm{p}] \tag{2.13}$$

Using the new objective function $\boldsymbol{J} := \left(J_1, \ldots, J_{n_\mathrm{o}}\right)^T$, with $J_{i_\mathrm{o}} : \mathcal{X} \times \mathcal{U}_\mathcal{F} \to \mathbb{R}$ and introducing $\underline{\boldsymbol{u}}_k^* \in \mathcal{U}_\mathcal{F}$, with

$${}^o\boldsymbol{u}_k^* : [t_k, t_k + T_\mathrm{p}] \to \boldsymbol{f}_\mathcal{U}\left(\underline{\boldsymbol{u}}_k^*\right), \tag{2.14}$$

problem (2.7) can be reformulated as:

For each $k = 0, 1, 2, \ldots$ set $t_k = k \cdot \delta$ and solve:

$$\begin{aligned} & \underline{\boldsymbol{u}}_k^* := \arg\min_{\underline{\boldsymbol{u}} \in \mathcal{U}_{\mathcal{F}}} J_1\left({}^o\boldsymbol{x}(\tau), \underline{\boldsymbol{u}}\right) \\ \text{subject to} \quad & {}^o\boldsymbol{x}'(\tau) = \boldsymbol{f}\left({}^o\boldsymbol{x}(\tau), {}^o\boldsymbol{u}(\tau), \boldsymbol{0}\right), \qquad {}^o\boldsymbol{x}\left(t_k\right) = \boldsymbol{x}\left(t_k\right), \\ & {}^o\boldsymbol{x}(\tau) \in \mathcal{X}, \qquad \forall \tau \in \left[t_k, t_k + T_\mathrm{p}\right], \\ & {}^o\boldsymbol{u} : \left[t_k, t_k + T_\mathrm{p}\right] \to \boldsymbol{f}_{\mathcal{U}}\left(\underline{\boldsymbol{u}}\right). \end{aligned} \tag{2.15}$$

As ${}^o\boldsymbol{u}_k^* \overset{(2.14)}{=} \boldsymbol{f}_{\mathcal{U}}\left(\underline{\boldsymbol{u}}_k^*\right)$, equation (2.8) can be applied.
To stress that the open loop input $\underline{\boldsymbol{u}}$ is the vector of optimization variables, which therefore is the only grip to influence the values of the objective function, if necessary the following notation is used:

$$\boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}\right) := \boldsymbol{J}\left({}^o\boldsymbol{x}(\tau), \underline{\boldsymbol{u}}\right). \tag{2.16}$$

The function $\boldsymbol{J}_{\boldsymbol{x}} : \mathcal{U}_{\mathcal{F}} \to \mathbb{R}^{n_\mathrm{o}}$, $\boldsymbol{J}_{\boldsymbol{x}} := \left(J_{\boldsymbol{x},1}, \ldots, J_{\boldsymbol{x},n_\mathrm{o}}\right)^T$, will be used in Section 2.2 to simplify the notation, $J_{\boldsymbol{x},i_\mathrm{o}} : \mathcal{U}_{\mathcal{F}} \to \mathbb{R}$ for $i_\mathrm{o} = 1, \ldots, n_\mathrm{o}$.

## 2.2 Multi-Objective Optimization

To be able to solve problem (2.4) for $n_\mathrm{o} > 1$ the concept of multi-objective optimization is introduced. In multi-objective optimization one tries to solve the following optimization problem:

$$\begin{aligned} & \text{minimize}_{\underline{\boldsymbol{u}}} \ \boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}\right) \\ \text{subject to} \quad & \underline{\boldsymbol{u}} \in \mathcal{U}_{\mathcal{F}} \end{aligned} \tag{2.17}$$

To solve (2.17) a couple of terms are defined to get an idea of how to minimize the vector function $\boldsymbol{J}_{\boldsymbol{x}}$.
In Definition 2.1 the notation $\boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}_1\right) \leq \boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}_2\right)$ is used, which is short for $J_{\boldsymbol{x},i_\mathrm{o}}\left(\underline{\boldsymbol{u}}_1\right) \leq J_{\boldsymbol{x},i_\mathrm{o}}\left(\underline{\boldsymbol{u}}_2\right) \forall i_\mathrm{o} \in \{1, \ldots, n_\mathrm{o}\}$, for $\underline{\boldsymbol{u}}_1, \underline{\boldsymbol{u}}_2 \in \mathcal{U}_{\mathcal{F}}$.

**Definition 2.1** (Custódio et al. (2012))**:** Given two vectors of optimization variables $\underline{\boldsymbol{u}}_1, \underline{\boldsymbol{u}}_2 \in \mathcal{U}_{\mathcal{F}}$, it is said that $\underline{\boldsymbol{u}}_1$ dominates $\underline{\boldsymbol{u}}_2$, being represented by $\underline{\boldsymbol{u}}_1 \prec \underline{\boldsymbol{u}}_2$, iff $\boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}_1\right) \leq \boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}_2\right)$ and $J_{\boldsymbol{x},i_\mathrm{o}}\left(\underline{\boldsymbol{u}}_1\right) < J_{\boldsymbol{x},i_\mathrm{o}}\left(\underline{\boldsymbol{u}}_2\right) \exists i_\mathrm{o} \in \{1, \ldots, n_\mathrm{o}\}$.

As $\boldsymbol{J}_{\boldsymbol{x}}$ shall be minimized, $\underline{\boldsymbol{u}}_1$ is always preferred over $\underline{\boldsymbol{u}}_2$, if $\underline{\boldsymbol{u}}_1 \prec \underline{\boldsymbol{u}}_2$. Definition 2.1 implies that $\underline{\boldsymbol{u}}_1 \prec \underline{\boldsymbol{u}}_2$ if and only if $\boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}_1\right) \leq \boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}_2\right)$ and $\boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}_1\right) \neq \boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}_2\right)$. Some authors define the dominance relation in the space of objective function vectors. In this meaning there exists the following equivalence, which is used in this thesis:

$$\underline{\boldsymbol{u}}_1 \prec \underline{\boldsymbol{u}}_2 \equiv \boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}_1\right) \prec \boldsymbol{J}_{\boldsymbol{x}}\left(\underline{\boldsymbol{u}}_2\right).$$

Special interest lies in vectors of optimization variables $\underline{\boldsymbol{u}}$ which are non-dominated within a given set. They are called Pareto optimal points, see Definition 2.2.

**Definition 2.2** (Coello Coello (2011))**:** It is said that a vector of optimization variables $\underline{\boldsymbol{u}}^* \in \mathcal{U}_{\mathcal{F}}$ is **Pareto optimal** iff there does not exist another $\underline{\boldsymbol{u}} \in \mathcal{U}_{\mathcal{F}}$ such that $\underline{\boldsymbol{u}} \prec \underline{\boldsymbol{u}}^*$.

Pareto optimal points are so-called trade-off solutions. There is no solution which is better (viz. smaller) in all components, but there could be solutions which are at least better in some component(s) and in each case worse in other components.
If, for $\underline{\boldsymbol{u}}_1, \underline{\boldsymbol{u}}_2 \in \mathcal{U}_{\mathcal{F}}$, $\underline{\boldsymbol{u}}_1 \not\prec \underline{\boldsymbol{u}}_2$ and $\underline{\boldsymbol{u}}_2 \not\prec \underline{\boldsymbol{u}}_1$ then $\underline{\boldsymbol{u}}_1$ and $\underline{\boldsymbol{u}}_2$ are said to be nondominated points. A subset of $\mathcal{U}_{\mathcal{F}}$ is said to be nondominated when any pair of points in this subset is nondominated (Custódio et al., 2012).

**Definition 2.3** (Coello Coello (2011))**:** The **Pareto optimal set** $\mathcal{P}^*$ is defined by:

$$\mathcal{P}^* := \{\underline{\boldsymbol{u}} \in \mathcal{U}_{\mathcal{F}} \,|\underline{\boldsymbol{u}} \text{ is Pareto optimal}\}$$

Out of definition the Pareto optimal set is a nondominated set. The Pareto optimal set contains all Pareto optimal points in the feasible set $\mathcal{U}_{\mathcal{F}}$. As for each vector in the Pareto optimal set, there does not exist a better (in the sense of domination) solution candidate with respect to problem (2.17), each Pareto optimal point will minimize the objective function $\boldsymbol{J}_{\boldsymbol{x}}$. In other words, all Pareto optimal points are equally good, thus there is no ranking of Pareto optimal points at the moment. This is why we want to know the Pareto optimal set for the given problem (2.17). A set which contains only a subset of all Pareto optimal points is called approximate Pareto set or just Pareto set. As will be shown later in Chapter 3, it usually is not possible to find the Pareto optimal set but only an approximate Pareto set, because only a finite number of Pareto optimal points can be found.
To the Pareto optimal set there does also exist the corresponding Pareto front, see Figure 2.2, defined as:

**Definition 2.4** (Coello Coello (2011))**:** The **Pareto front** $\mathcal{PF}^*$ is defined by:

$$\mathcal{PF}^* := \{\boldsymbol{J}_{\boldsymbol{x}}(\underline{\boldsymbol{u}}) \in \mathbb{R}^{n_\mathrm{o}} | \underline{\boldsymbol{u}} \in \mathcal{P}^*\}$$

To each approximate Pareto set there does also exist an approximate Pareto front. So later the question will be to find the best finite set of solutions, which approximates the Pareto front best possibly.
Multi-objective optimization algorithms, see Chapter 3, then have the task to find the Pareto optimal set and therefore the Pareto front. Using both terms the general multi-objective nonlinear model predictive control problem with $n_\mathrm{o} > 1$ is studied in Section 2.3.

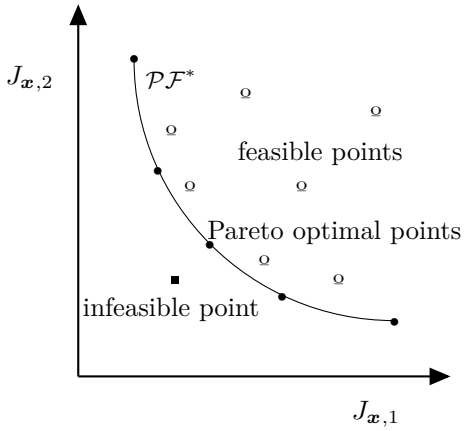


**Figure 2.2:** Example of a two-dimensional Pareto front. The Pareto front is depicted as a line. Feasible points lie on the right side of the line, infeasible points on the left side. Pareto optimal points are feasible points which lie directly on the Pareto front.

## 2.3 Case II: Number of Objectives $n_\mathrm{o} > 1$

Knowing that the minima of the objective function $\boldsymbol{J}$ lie on the Pareto front, problem (2.4) is approximately solved by:

For each $k = 0, 1, 2, \ldots$ set $t_k = k \cdot \delta$ and solve:

$$\begin{aligned} & & \mathcal{PF}_k^* &:= \min_{\underline{\boldsymbol{u}} \in \mathcal{U}_\mathcal{F}} \boldsymbol{J}\left({}^o\boldsymbol{x}(\tau), \underline{\boldsymbol{u}}\right) & & \\ \text{subject to} \quad & & {}^o\boldsymbol{x}'(\tau) &= \boldsymbol{f}\left({}^o\boldsymbol{x}(\tau), {}^o\boldsymbol{u}(\tau), \boldsymbol{0}\right), & & {}^o\boldsymbol{x}\left(t_k\right) = \boldsymbol{x}\left(t_k\right), \\ & & {}^o\boldsymbol{x}(\tau) &\in \mathcal{X}, & & \forall \tau \in [t_k, t_k + T_\mathrm{p}], \\ & & {}^o\boldsymbol{u} &: [t_k, t_k + T_\mathrm{p}] \to \boldsymbol{f}_\mathcal{U}\left(\underline{\boldsymbol{u}}\right). \end{aligned} \tag{2.18}$$

Let $\mathcal{P}_k^*$ be the corresponding Pareto optimal set to the Pareto front $\mathcal{PF}_k^*$ for each $k = 0, 1, 2, \ldots$. Then the optimal input $\underline{\boldsymbol{u}}_k^* \in \mathcal{U}_\mathcal{F}$ has to be picked out of the solutions inside the Pareto optimal set $\mathcal{P}_k^*$. One possible approach would be the use of a weighted sum:

$$\underline{\boldsymbol{u}}_k^* := \arg\min_{\underline{\boldsymbol{u}} \in \mathcal{P}_k^*} \sum_{i_\mathrm{o}=1}^{n_\mathrm{o}} \varpi_{i_\mathrm{o}} \cdot J_{\boldsymbol{x},i_\mathrm{o}}\left(\underline{\boldsymbol{u}}\right) \tag{2.19}$$

with $\varpi_{i_\mathrm{o}} \in (0,1)$, $i_\mathrm{o} = 1, \ldots, n_\mathrm{o}$ and $\sum_{i_\mathrm{o}=1}^{n_\mathrm{o}} \varpi_{i_\mathrm{o}} = 1$. The weights $\varpi_{i_\mathrm{o}}$ could also be made dependent on the current state of the system $\boldsymbol{x}\left(t_k\right)$.

Other possibilities to determine the optimal input $\underline{\boldsymbol{u}}_k^*$ out of the Pareto optimal set $\mathcal{P}_k^*$ can be found in Bemporad and Muñoz de la Peña (2009), Valera García et al. (2012) and Flores-Tlacuahuac et al. (2012).

## 2.4 Summary and Discussion

In this chapter an optimization problem was defined, which states that an objective function $\widetilde{\boldsymbol{J}}$ shall be minimized which depends on state trajectories of a dynamic system ${}^{o}\boldsymbol{x}(t)$ and inputs $\boldsymbol{u}(t)$, eq. (2.4). It was proposed to approximately solve this optimal control problem using multi-objective nonlinear model predictive control. Applying NMPC resulted in the problem formulation (2.18):

For each $k = 0, 1, 2, \ldots$ set $t_k = k \cdot \delta$ and solve:

$$\begin{aligned} & \mathcal{PF}_k^* := \min_{\underline{\boldsymbol{u}} \in \mathcal{U}_{\mathcal{F}}} \boldsymbol{J}\left({}^{o}\boldsymbol{x}(\tau), \underline{\boldsymbol{u}}\right) \\ \text{subject to} \quad & {}^{o}\boldsymbol{x}'(\tau) = \boldsymbol{f}\left({}^{o}\boldsymbol{x}(\tau), {}^{o}\boldsymbol{u}(\tau), \boldsymbol{0}\right), \qquad {}^{o}\boldsymbol{x}\left(t_k\right) = \boldsymbol{x}\left(t_k\right), \\ & {}^{o}\boldsymbol{x}(\tau) \in \mathcal{X}, \qquad \forall \tau \in \left[t_k, t_k + T_{\mathrm{p}}\right], \\ & {}^{o}\boldsymbol{u} : \left[t_k, t_k + T_{\mathrm{p}}\right] \to \boldsymbol{f}_{\mathcal{U}}\left(\underline{\boldsymbol{u}}\right). \end{aligned} \tag{2.20}$$

with the optimal input vector $\underline{\boldsymbol{u}}_k^*$ in equation (2.19)

$$\underline{\boldsymbol{u}}_k^* := \arg\min_{\underline{\boldsymbol{u}} \in \mathcal{P}_k^*} \sum_{i_{\mathrm{o}}=1}^{n_{\mathrm{o}}} \varpi_{i_{\mathrm{o}}} \cdot J_{\boldsymbol{x}, i_{\mathrm{o}}}\left(\underline{\boldsymbol{u}}\right) \tag{2.21}$$

and application of equation (2.8) which gives the optimal input in the interval $t \in [t_k, t_k + \delta)$

$$\boldsymbol{u}(t) = {}^{o}\boldsymbol{u}_k^*(t) = \boldsymbol{f}_{\mathcal{U}}\left(\underline{\boldsymbol{u}}_k^*\right), \qquad t \in \left[t_k, t_k + \delta\right). \tag{2.22}$$

As open questions remained how to solve the optimization problem in eq. (2.20) and how to get the state of the system at time $t_k$, $\boldsymbol{x}\left(t_k\right)$, in case it is not directly observable. The first question will be tackled in the next Chapter 3 and the latter question will be answered in Chapter 4.

In model-based control offset-free control in case of plant-model mismatch is an important issue, because there the control error usually is not directly fed back to the control as it is done in conventional control (e.g. Huang et al. (2010), Tian et al. (2012)). There are different approaches to handle this problem. For example it can be solved by introducing a disturbance model that models the plant-model mismatch and persistent disturbances acting on the plant (Maeder et al., 2009, Morari and Maeder, 2012). However, in this thesis it is tackled using RTO, thus a master/slave (or two-layer) approach, whereas the master (upper layer) control is the model-based and the slave (lower layer) is a conventional control. This approach is explained in detail in Chapter 9.