Improved Predictions from Measured Disturbances in Linear Model Predictive Control

B.J.T. Binder[a,*], T.A. Johansen[a,b], L. Imsland[a]

[a] *Department of Engineering Cybernetics, Norwegian University of Science and Technology, O.S. Bragstads plass 2D, NO-7491 Trondheim, Norway*
[b] *Center for Autonomous Marine Operations and Systems, NTNU*

Abstract

Measured disturbances are often included in model predictive control (MPC) formulations to obtain better predictions of the future behavior of the controlled system, and thus improve the control performance. In the prediction model, a measured disturbance is in many ways treated like a control input to the system. However, while control inputs change only once per sampling interval as new control inputs are calculated, measured disturbances are typically sampled from continuous variables. While this difference is usually neglected, it is shown in this paper that taking this difference into account may improve the control performance. This is demonstrated through two simulation studies, including a realistic multivariable control problem from the petroleum industry. The proposed method requires only a minor modification in the implementation of the prediction model, and may thus improve the control performance with a minimal effort.

Keywords: Model Predictive Control, Disturbances, Prediction, Sampled Data

1. Introduction

The very foundation of model predictive control (MPC) is to predict the future behavior of a system based on a model [1]. In order to improve the control performance, feedforward from measured disturbances may also be included in this prediction model. This requires that the prediction model includes the dynamics from the measured disturbance to the output, in addition to the dynamics from the control input to the output. Predictions of the output response from measured disturbances may then be made in the same manner as with the inputs [2, 3]. However, there are two fundamental differences between control inputs and measured disturbances in the MPC framework:

1. While future control inputs are decision variables in the MPC formulation, and thus are known (predicted), future measured disturbances are unknown to the controller.
2. Control inputs typically change only once per sampling interval (as a new input is calculated and applied), while disturbances are typically sampled from variables that change continuously between samples.

*Corresponding author
Email addresses: `benjamin.binder@ntnu.no` (B.J.T. Binder), `tor.arne.johansen@ntnu.no` (T.A. Johansen), `lars.imsland@ntnu.no` (L. Imsland)

To cope with the first difference, common practice is to assume that future disturbances will remain constant at the last available measurement, though other assumptions may be more appropriate if a better knowledge of the disturbance is available [4]. A practical example of the latter can be found in [5], where a measured disturbance was extrapolated into the future using an autoregressive model. In [2], it is described how a model of disturbance dynamics, if available, may be included in the MPC predictions in a state-space formulation.

The second difference is usually ignored both in practical implementations of MPC and in the theoretical literature. This paper addresses implementation aspects and performance improvements from feedforward of measured disturbances with focus on this difference. We have not found any previous literature addressing this issue.

MPC arose from the industrial applications IDCOM [6] and DMC [7], where the prediction model is based on finite-impulse or step-response models. MPC with state-space models soon dominated academic research [8, 9]. In [4, p. 115], it is even stated that "There is really no good reason for working with step (or pulse) response models." However, step-response models have remained popular in industrial applications, the main reasons being that step-response models are intuitive, easy to maintain, and allows for easy and straight-forward system identification [10]. It should be noted, though, that there are standard algorithms that translate step-response models into state-space form (or transfer-function form, for that matter). The results in this paper do not rely on which model representation is used, and both state-space and step-response model representations are considered in this paper.

To achieve offset-free tracking of setpoints in MPC, and counteract the effect of various uncertainties, a feedback mechanism must be included in the prediction model. The most widely-used industrial implementations of (step-response) MPC use a constant output step disturbance model to achieve offset-free tracking [11]. The current measured output is then compared to the output of the prediction model, and the error (often denoted *bias*) is added to the future predictions. For state-space MPC formulations, there are many alternative methods for offset-free tracking, see e.g. [12, 13], and no particular method seems to have become "standard practice". The purpose in any case is to estimate and counteract the effect of uncertainties in the system, such as plant-model mismatch and unmeasured disturbances [8, 12]. The method proposed in [14] for systems with full state measurements is implemented for state-space systems in this paper.

The paper is organized as follows: First, in Section 2, the main issue considered in this paper (the second difference listed above) is discussed, and a method to address this issue is proposed. In Section 3, a typical MPC formulation for a SISO system with a measured disturbance is given, considering both state-space and step-response prediction model formulations. The implementation of both the conventional and the proposed method is described in Section 4, and some of the conceptual differences between the two methods are discussed analytically. How the assumption about future disturbances affect the control performance differently for the two methods is also discussed here. In Section 5, closed-loop simulation results for a SISO example system are presented comparing the method proposed in this paper to the conventional implementation method. Both the disturbance dynamics, measurement noise, prediction of the measured disturbance, and tuning of the controller is considered in this example. In Section 6, the two methods are implemented in simulations of a realistic industrial example; a petroleum production well with an electric submersible pump (ESP) installed. Finally, the main conclusions and results in this paper are summarized and discussed in Section 7.

2. Continuous Disturbance in a Discrete-Time System

Consider a linear time-invariant (LTI), continuous-time state-space model of a system with a measured disturbance:

$$\dot{x}(t) = A^c x(t) + B_u^c u(t) + B_d^c d(t) \tag{1a}$$

$$y(t) = C^c x(t) \tag{1b}$$

where $x(t)$ is the system state, $u(t)$ is the control input (also known as manipulated variable, MV), $d(t)$ is the measured disturbance (also known as disturbance variable, DV), $y(t)$ is the measured system output (also known as controlled variable, CV), and A^c, B_u^c, B_d^c and C^c are the system matrices, the superscript c denoting that these are for the continuous-time system.

To implement MPC for this system, the continuous-time model must be discretized to obtain a discrete-time equivalent:

$$x_{k+1} = Ax_k + B_u u_k + B_d d_k \tag{2a}$$

$$y_k = Cx_k \tag{2b}$$

The goal of the discretization is to obtain the discrete-time system matrices A, B_u, B_d and C, so that the dynamics of the discrete-time system matches the dynamics of the continuous-time system as closely as possible, as any model discrepancy may reduce the control performance.

2.1. Discretizing Using Zero-Order Hold

The standard approach for MPC implementations based on state-space models is to discretize the system model using zero-order-hold (ZOH). It is then assumed that all control inputs are piecewise constant, and only change at the exact time of the samples.

As discussed e.g. in [15, Section 4.1.2], by using ZOH, an exact discretization of the continuous-time system (1) is obtained, implying that the dynamics of the discrete-time system will coincide perfectly with the continuous-time (real/original) system at the sampled points in time, *given that the input signals are in fact applied using ZOH.*

In MPC, the applied control input is calculated once every sampling interval, and kept constant between samples, so the control input is in fact implemented using ZOH. Using ZOH in the discretization will thus provide a near perfect match between the discretized and the continuous-time system models.

The discretization is typically performed using ZOH also for measured disturbances, basically treating a measured disturbance just like another control input to the system. However, while ZOH is very suitable for a control input, an assumption that also a measured disturbance remains constant between samples is usually inaccurate, as measured disturbances typically are sampled from variables that change *continuously.* If ZOH is applied to a measured disturbance when discretizing the model, a sampled continuous disturbance signal will consequently be interpreted as a piecewise constant signal by the resulting prediction model, as illustrated in the top plot in Figure 1. As seen in this illustration, the “ZOH signal” does not match the continuous-time signal very accurately, but suffers from what could be considered a “time delay”, as the ZOH signal ignores any change in the continuous signal between samples, and is only updated when a new sample is taken. The consequence of introducing this time delay in the prediction model is that the effect of the measured disturbance is in fact predicted to occur later in time. This reduces the accuracy of the prediction model, and the dynamics of the discrete-time model does not match those of the continuous-time model, even if the exact same disturbance is applied to both systems. This is demonstrated later, in the illustrative example in Section 2.4.

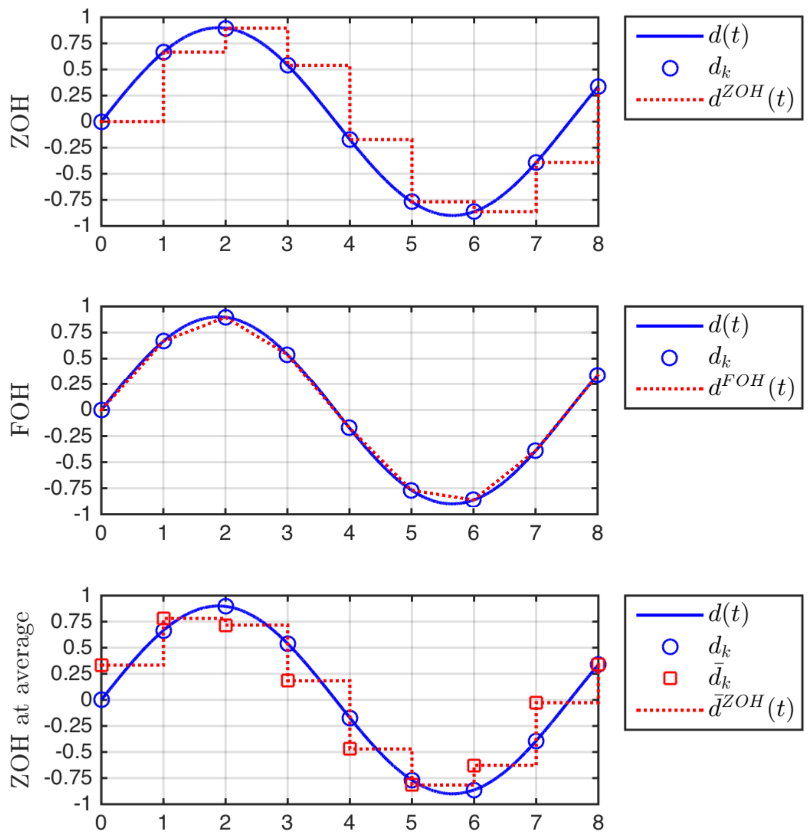


Figure 1: Illustration of different methods to discretize a continuous input variable

2.2. Discretizing Using First-Order Hold

As discretizing using ZOH provides a poor match for a continuous signal, one might instead consider discretizing the model using first-order hold (FOH) for the measured disturbance. This is equivalent to assuming that the measured disturbance changes linearly between samples, as illustrated in the second plot in Figure 1. According to [16], “The First-Order Hold (FOH) method provides an exact match between the continuous- and discrete-time systems in the time domain for piecewise linear inputs” and “is generally more accurate than ZOH for systems driven by smooth inputs.” This is quite clear from the illustrations in Figure 1. This implies that discretizing with FOH would generally provide a more accurate discrete-time model than with ZOH, which is also demonstrated later, in the example in Section 2.4.

Note that discretizing using ZOH results in the exact same output function as the continuous-time system, i.e. (2b) with $C = C^c$ (as shown in [15]), so that:

$$y_k = y(t) \Leftrightarrow x_k = x(t) \tag{3}$$

whereas discretizing using FOH alters the output function, and the above relationship no longer holds. In general, a continuous-time system on the form:

$$\begin{aligned} \dot{x} &= Ax(t) + Bd(t) && \text{(4a)} \\ y(t) &= Cx(t) && \text{(4b)} \end{aligned}$$

discretized using FOH will result in a discrete-time system on the form:

$$\begin{aligned} x^F_{k+1} &= A^F x^F_k + B^F d_k && \text{(5a)} \\ y^F_k &= C^F x^F_k + D^F d_k && \text{(5b)} \end{aligned}$$

with $D^F \neq 0$. (The superscript F denotes that this system is discretized using FOH.) This means that the output y^F_k of the system discretized using FOH will have a direct feedthrough from the measured disturbance (y^F_k depends directly on d_k), even if this is not the case for the continuous-time system. This complicates the MPC implementation in two ways.

First, most MPC formulations, both in practical implementations and in the literature, assume that there is no direct feedthrough from the control input or measured disturbance to the output, so that the prediction model is given in the form (2). Using a model discretized using FOH may thus in some cases make it difficult to implement known methods from the literature, such as methods for offset-free control. Even if a method can be reformulated to be used with a prediction model in the form (5), verifying the correctness of the new formulation may not always be straightforward.

Second, the output function takes a different form with FOH and ZOH, implying that the state vector is also different. Since ZOH should be used for the control input, using FOH for the measured disturbance entails that the system must be discretized separately for the control input and the disturbance, and the superposition theorem must be used to combine the resulting sub-systems to obtain the complete discrete-time prediction model. This results in an augmented state vector, with twice the number of states as in the original system, with all the implications this has for the complexity of the resulting MPC optimization problem. There may exist a minimal realization of the combined system with the same number of states as the original system in some cases, though it has not been investigated in this study whether it does so in general.

In conclusion, FOH is far more accurate than ZOH for continuous disturbances, but due to these potential complications, changing the discretization method for the disturbance may in practice prove to be more complicated than what can be justified from the possible advantage of addressing this issue in the first place. Thus, a much simpler solution that "approximates" FOH discretization is proposed in this paper, as discussed in the following section.

2.3. Proposed Solution

The basic idea of the proposed solution is not to change the discretization method, but instead to substitute the sampled disturbance signal d_k with another discrete-time disturbance signal $\bar{d}_k$ that provides a better match between the continuous- and discrete-time systems with $d(t)$ and $\bar{d}_k$ as inputs, compared to using $d(t)$ and the sampled d_k as inputs. This will not in any way alter the discrete-time system model or how this is derived.

The question is how to calculate $\bar{d}_k$. Ideally, $\bar{d}_k$ should be calculated to minimize the difference between the outputs $y(t)$ and y_k of the continuous- and discrete-time systems. In AppendixA, it is shown how an optimal $\bar{d}_k$ (denoted $\bar{d}^*_k$) can be calculated for a first-order SISO system in this way, and that implementing the derived expression for $\bar{d}^*_k$ in a system discretized using ZOH is in fact equivalent to discretizing the system using FOH. Further, it is shown that simply calculating $\bar{d}_k$ as the average of d_k and d_{k+1} is in fact a good approximation of the optimal $\bar{d}_k^*$, and thus of FOH, given that the sampling rate is relatively fast compared to the system dynamics. The expression for $\bar{d}_k$ is then simply given by:

$$\bar{d}_k = \frac{d_k + d_{k+1}}{2} \tag{6}$$

As the system dynamics are not taken into account, this is slightly less accurate than using the optimal $\bar{d}_k^*$, but for the same reason, this is a general solution (independent of the specific system), and thus directly applicable to any MPC implementation, even MPC based on step-response models. Given the results in AppendixA, and the simplicity of this method, it seems like a very appropriate solution for practical implementations, and is thus the proposed method in this paper. In the sequel, substituting d_k with $\bar{d}_k$ from (6) is referred to as method B, while the conventional approach (using the sampled d_k directly) is referred to as method A.

Method A is illustrated in the top plot in Figure 1, and method B is illustrated in the bottom plot. From this illustration, it should be quite obvious that $\bar{d}^{ZOH}(t)$ in the bottom plot, that results from applying ZOH to $\bar{d}_k$, is a much closer match to $d(t)$ than $d^{ZOH}(t)$ in the top plot, that results from applying ZOH to d_k. As shown in the appendix, method B is in fact a close approximation to discretizing the system using FOH (illustrated in the second plot). But compared to discretizing the system using FOH, method B has some major advantages:

- It will not in any way alter the discrete-time system model, or how it is derived, which implies that it can be applied directly to any existing/conventional discrete-time model, regardless of how it is obtained.
- It is completely independent of the prediction model representation, and can be applied directly to both state-space and step-response MPC, as shown in Section 4.
- It is very easy and intuitive to implement. This method could (to some extent) simply be considered a filter on the measured disturbance. Filtering measured disturbances using various filters is standard industrial practice, and in that sense, this method is similar to what is already done in industry.

Thus, method B may increase the accuracy of the predictions in the MPC controller, and thus the control performance, with a minimal effort.

It should be noted that with method B, $\bar{d}_k$ is not known at time k, as it depends on the measurement d_{k+1}. The implications of this are discussed in Section 2.5.

2.4. Illustrative Example

The effect of the discretization method for a system with a continuous disturbance is now demonstrated through an illustrative example. Consider the system:

$$\dot{x}(t) = -x(t) + d(t) \tag{7a}$$

$$y(t) = x(t) \tag{7b}$$

This is a simple first-order SISO system in continuous-time state-space form, with the system matrices:

$$A^c = -1,\ B_d^c = 1,\ C^c = 1,\ D^c = 0 \tag{8}$$

and the continuous disturbance $d(t)$ as the only input. Both a ZOH and a FOH discretization of this system with sampling time $T = 1$ is easily obtained in Matlab with the functions `ss` and c2d, where the discretization method is simply specified as a parameter in the c2d function call, as follows:

```
% Real system
A_c = -1; B_c = 1; C_c = 1; D_c = 0;
sys = ss(A_c,B_c,C_c,D_c);
% ZOH
sys_zoh = c2d(sys,T);
A_zoh=sys_zoh.a, B_zoh=sys_zoh.b, C_zoh=sys_zoh.c, D_zoh=sys_zoh.d
% FOH
sys_foh = c2d(sys,T,'foh');
A_foh=sys_foh.a, B_foh=sys_foh.b, C_foh=sys_foh.c, D_foh=sys_foh.d
```

This produces the following output from Matlab:

```
A_zoh = 0.3679, B_zoh = 0.6321, C_zoh = 1, D_zoh = 0
A_foh = 0.3679, B_foh = 0.3996, C_foh = 1, D_foh = 0.3679
```

Note how there is no direct feedthrough from the input to the output of the continuous-time system (7) in this example ($D^c = 0$), and how this is still the case for the system discretized using ZOH (D_zoh = 0), but not for the system discretized using FOH (D_foh = 0.3679).

The discrete-time systems can be simulated in Matlab, using both method A and method B for the ZOH-system, as outlined below:

```
% Initialize x_a, y_a, x_b, y_b, x_f, y_f
for k = (1:t_end)+1 % +1 due to 1-based indexing in Matlab
    % Obtain sampled measurement d(k)
    % ZOH - Method A
    x_a(k) = A_zoh * x_a(k-1) + B_zoh * d(k-1);
    y_a(k) = C_zoh * x_a(k);
    % ZOH - Method B
    x_b(k) = A_zoh * x_b(k-1) + B_zoh * (d(k-1) + d(k))/2;
    y_b(k) = C_zoh * x_b(k);
    % FOH
    x_f(k) = A_foh * x_f(k-1) + B_foh * d(k-1);
    y_f(k) = C_foh * x_f(k) + D_foh * d(k);
end
```

The systems are now simulated with the continuous disturbance $d(t) = \sin(t/2)$, and the discrete-time disturbance d_k obtained by sampling the disturbance $d(t)$ with the same sampling interval $T = 1$ that was used in the discretization. The result of these simulations are shown in Figure 2. The first plot shows the continuous disturbance $d(t)$, the sampled disturbance d_k, the calculated disturbance $\bar{d}_k$ for method B, as well as the continuous-time signal $d^{ZOH}(t)$ that is obtained by applying ZOH to the sampled disturbance d_k. The second plot shows five different outputs:

$y(t)$ - the output of the continuous-time system with $d(t)$ as input

$y^{ZOH}(t)$ - the output of the continuous-time system with $d^{ZOH}(t)$ as input

y_k^A - the output of the ZOH-discretized system with d_k as input (method A)

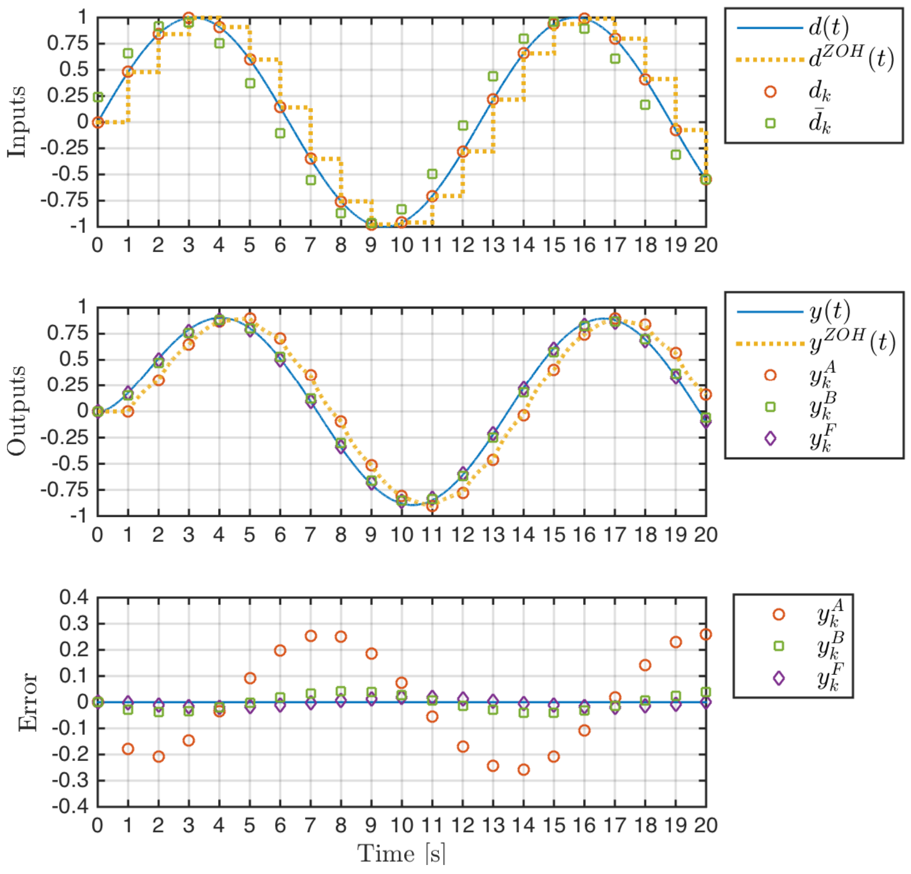


Figure 2: Example: Applying a continuous disturbance to the discretized systems

y_k^B - the output of the ZOH-discretized system with $\bar{d}_k$ as input (method B)

y_k^F - the output of the FOH-discretized system with d_k as input

Ideally, a discretized system with d_k applied as input should at every sample provide exactly the same output as the continuous-time system with $d(t)$ applied as input. However, as seen in the plot, y_k^A is far from identical to $y(t)$ at the samples, but the effect of the "time delay" introduced by the ZOH discretization is quite evident. As expected, y_k^A is instead identical to $y^{ZOH}(t)$ at every sample. On the other hand, the output y_k^F of the system discretized using FOH almost perfectly matches $y(t)$. The output y_k^B (method B) is not quite as accurate as FOH, but far more accurate than method A. These findings are confirmed in the last plot, which shows the error for each discretization method, i.e. the discrepancy between the discrete-time and the continuous-time output.

2.5. Dependency on Future Disturbance

The detailed implementation of method B for an MPC formulation is presented in Section 4, but a few observations should be made at this point in the discussion.

Given the continuous-time model (1), the discrete-time model discretized using ZOH will take the form (2). Thus, all the information required to calculate the *predicted* output y_{k+1} with method A is available already at time k, whereas with method B, y_{k+1} cannot be calculated before the measurement d_{k+1} is available. This is very similar to the system (5) discretized using FOH, where the output y_k depends directly on the current measurement d_k. This causes no problem when *simulating* the system (as was done in Section 2.4), as all the information required to calculate the *current* output y_k with method B is indeed available at time k (when d_k is available). But since the discrete-time model is used as a *prediction* model in the MPC controller, the fact that y_{k+1} no longer can be calculated at time k may at first seem like an obstacle to implement MPC with method B. However, due to the fact that MPC is predictive by nature, this does not actually change the implementation much, as shown in Section 4. The fact that d_{k+1} is not known at time k only means that this too must be predicted. This may sound like a big difference, but in MPC, the predictions of the system output will in fact *always* depend on some kind of prediction or extrapolation of the measured disturbance. The most common prediction is derived from simply assuming that the measured disturbance will remain constant at the last measured value [2, 4], in which case the prediction of d_{k+1} at time k is simply given by the measurement d_k:

$$\hat{d}_{k+1|k} = d_k \tag{9}$$

This is discussed in more detail in Section 4.3.

From a continuous-time perspective, it is obvious that the system output $y(t)$ at time $t = (k+1)T$ will depend on the disturbance $d(t)$ in the interval $kT \leq t \leq (k+1)T$. The fact that y_{k+1} with method B depends on d_{k+1} is thus only an indication that the goal to obtain a better match between the continuous- and discrete-time systems is achieved. As discussed later in Section 4 and demonstrated in the SISO example in Section 5, this dependency implies that method B will be able to take full advantage of any knowledge about the disturbance dynamics which can be used to predict/extrapolate the measured disturbance more accurately. On the other hand, the fact that y_{k+1} with method A does *not* depend on d_{k+1} implies that any such knowledge can only be exploited for the prediction of y_{k+2} onwards, which is in fact a major limitation for the conventional method A, as discussed in Section 4.4.

2.6. Step-Response Models

The results in this paper are presented with both state-space and step-response model representations. A step-response model is usually a sampled recording of the system response from a step in a control input or measured disturbance. Thus, when using step-response models in the prediction model, it is inherently assumed that a step occurs in each control input $u(t)$ and measured disturbance $d(t)$ at every time sample (and that they are constant between the samples), so that the predicted effect is given by the step-response models. In that sense, step-response models are equivalent to state-space models discretized using ZOH, and the proposed method B may be implemented for step-response models with the exact same approach as with state-space models, i.e. by replacing d_k with $\bar{d}_k$ from (6).

3. MPC Implementation

3.1. MPC Problem Formulation

A typical MPC formulation for a SISO system is given by the following optimization problem:

$$\min_{y,u,\Delta u} \sum_{k=1}^{H_p} \frac{1}{2} q y_k^2 + \sum_{k=0}^{H_u-1} \frac{1}{2} p \Delta u_k^2 \tag{10a}$$

subject to the constraints:

$$\begin{aligned} \Delta u^l \leq \quad & \Delta u_k && \leq \Delta u^h && (10b)\\ u^l \leq \quad & u_k && \leq u^h && (10c)\\ y^l \leq \quad & y_k && \leq y^h && (10d)\\ & u_k && = u_{k-1} + \Delta u_k && (10e)\\ & y_k && = \text{Prediction model} && (10f) \end{aligned}$$

where q and p are tuning parameters, H_p denotes the prediction horizon and H_u denotes the control horizon with $0 \leq H_u \leq H_p$. For simplicity of presentation, it is assumed that $H_u = H_p$ in the sequel.

The MPC control law is to solve the above optimization problem once every sampling interval, and implement the first control step Δu_k from the solution. Thus, the control input is updated each time step as follows:

$$u_k = u_{k-1} + \Delta u_k \tag{11}$$

The control input is then kept constant until the MPC problem is solved for the next time step, and the applied input is thus changed only once per sampling interval.

3.2. Prediction Model

The system model enters the MPC formulation as the prediction model in (10f). The implementation of the prediction model depends on which model representation is used. Predictions based on state-space models are most commonly found in the literature, while many industrial MPC applications still rely on step-response models. Implementation of the prediction model using each of these model representations are presented in this section. The following definitions are used in both representations:

$$\begin{aligned} \Delta x_k &= x_k - x_{k-1} && (12)\\ \Delta u_k &= u_k - u_{k-1} && (13)\\ \Delta d_k &= d_k - d_{k-1} && (14)\\ Y_k &= \begin{bmatrix} \hat{y}_{k+1|k} & \hat{y}_{k+2|k} & \cdots & \hat{y}_{k+H_p|k} \end{bmatrix}^T && (15)\\ \Delta U_k &= \begin{bmatrix} \Delta\hat{u}_{k|k} & \Delta\hat{u}_{k+1|k} & \cdots & \Delta\hat{u}_{k+H_u-1|k} \end{bmatrix}^T && (16)\\ \Delta D_k &= \begin{bmatrix} \Delta\hat{d}_{k|k} & \Delta\hat{d}_{k+1|k} & \cdots & \Delta\hat{d}_{k+H_p-1|k} \end{bmatrix}^T && (17) \end{aligned}$$

where $\hat{y}_{k+n|k}$ denotes the predicted system response, $\Delta\hat{u}_{k+n|k}$ denotes predicted control input steps, and $\Delta\hat{d}_{k+n|k}$ denotes predicted steps in the measured disturbance. (The definition of ΔD_k is discussed further in Section 4.)

3.2.1. State-Space Prediction Model

As discussed in Section 2.1, in state-space MPC, the predicted response of the output(s) y from the applied and predicted control input(s) u and the measured disturbance(s) d is calculated using a discrete-time state-space model on the form (2). In addition, a method to achieve offset-free tracking should be implemented. There are many alternative approaches to achieve offset-free tracking with MPC, see e.g. [12]. For simplicity, it is assumed in the sequel that all states are measured, and the method proposed for such systems in [14] is implemented[1]. Following this method, the state-space model (2) is augmented by a state disturbance term d_k^x as follows:

$$\begin{aligned} x_{k+1} &= Ax_k + B_u u_k + B_d d_k + d_k^x && (18a) \\ \hat{y}_{u,k} &= Cx_k && (18b) \end{aligned}$$

The estimated state disturbance is calculated each time step as follows:

$$d_k^x = x_k - \left[Ax_{k-1} + B_u u_{k-1} + B_d d_{k-1}\right] \tag{19}$$

and a constant state disturbance prediction is used:

$$\hat{d}_{k+n|k}^x = d_k^x, \quad \forall n > 0 \tag{20}$$

Following the approach shown e.g. in [4], by iterating the state-space model (18), inserting (19) and using the definitions (12)-(17) and (20), the complete prediction model may be written in the following matrix-vector form:

$$\begin{aligned} Y_k = & \begin{bmatrix} C \\ C \\ \vdots \\ C \end{bmatrix} x_k + \begin{bmatrix} CA \\ C\left(A^2 + A\right) \\ \vdots \\ C\left(\sum_{i=1}^{H_p} A^i\right) \end{bmatrix} \Delta x_k \\ & + \begin{bmatrix} CB_u & 0 & \cdots & 0 \\ C\left(A+I\right)B_u & CB_u & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ C\left(\sum_{i=0}^{H_p-1} A^i\right)B_u & \cdots & & CB_u \end{bmatrix} \Delta U_k \\ & + \begin{bmatrix} CB_d & 0 & \cdots & 0 \\ C\left(A+I\right)B_d & CB_d & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ C\left(\sum_{i=0}^{H_p-1} A^i\right)B_d & \cdots & & CB_d \end{bmatrix} \Delta D_k \end{aligned} \tag{21}$$

where x_k is the current (measured) state, and Δx_k is the difference between the two most recent measurements, as defined by (12).

[1]According to [14], as long as the process outputs are elements of the state vector, or linear combinations of its elements, this formulation is sufficient to achieve offset-free control under deterministic constant-type unmeasured disturbances entering the process at any point, including both "modeling errors and outer step or piecewise-constant disturbances changing rarely".

3.2.2. Step-Response Prediction Model

In step-response MPC, the predicted response of the output y from the control input u is given by a step-response model as follows:

$$\hat{y}_{u,k} = \sum_{i=1}^{N-1} s_{u,i} \Delta u_{k-i} + s_{u,N}\, u_{k-N} \tag{22}$$

where $s_{u,k}$ denotes the step-response model coefficients for the model from u to y. It is assumed here that the step-response model reaches steady-state after at most N samples, so that $s_{u,N+n} = s_{u,N}$, $\forall\, n > 0$.

In the same way, assuming that a step-response model $s_{d,k}$ from the disturbance d to the output y is obtained, the response from the measured disturbance may be included in the predictions by adding the term:

$$\hat{y}_{d,k} = \sum_{i=1}^{N-1} s_{d,i} \Delta d_{k-i} + s_{d,N} d_{k-N} \tag{23}$$

Following the ideas of [17, 18], the prediction model based on step-response models may be implemented recursively, as outlined in [19]. A prediction state vector Y_k^p is then defined as:

$$Y_k^p = [\tilde{y}_{k|k}, \tilde{y}_{k+1|k}, \ldots, \tilde{y}_{k+N-1|k}]^T \tag{24}$$

where $N \geq H_p$, and $\tilde{y}_{k+n|k}$ denotes the open-loop output from the prediction model given the control inputs and measured disturbances recorded up to time k. Unlike $\hat{y}_{k+n|k}$ in (15), $\tilde{y}_{k+n|k}$ only depends on *past* inputs and disturbances, and not on any predictions. The recursive prediction model is initialized as follows:

- Set all elements of the prediction state vector Y_k^p equal to the current measurement of the output, y_k
- Set $\Delta u_{k-1} = 0$ and $\Delta d_{k-1} = 0$
- Set $d_{k-n} = d_k$, $\forall\, n > 0$, where d_k is the current measurement of the disturbance

Then, the prediction state vector is updated recursively using:

$$Y_k^p = A_N^p Y_{k-1}^p + B_u^p \Delta u_{k-1} + B_d^p \Delta d_{k-1} \tag{25}$$

where A_N^p, B_u^p and B_d^p are defined as follows:

$$A_N^p = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ & & & 0 & 1 \\ 0 & \cdots & & 0 & 1 \end{bmatrix} \tag{26}$$

$$B_u^p = \begin{bmatrix} s_u(1), & \cdots, & s_u(N) \end{bmatrix}^T \tag{27}$$

$$B_d^p = \begin{bmatrix} s_d(1), & \cdots, & s_d(N) \end{bmatrix}^T \tag{28}$$

The complete prediction model is then given by:

$$Y_k = \Theta_u \Delta U_k + \Theta_d \Delta D_k + A^p_{H_p} Y^p_k + v_k \tag{29}$$

where Y_k, ΔU_k and ΔD_k are defined by (15)-(17), $A^p_{H_p}$ contains the H_p first rows of A^p_N, and Θ_u and Θ_d contain the step-response model coefficients, as follows:

$$\Theta_u = \begin{bmatrix} s_u(1) & 0 & \cdots & 0 \\ s_u(2) & s_u(1) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ s_u(H_p) & \cdots & & s_u(1) \end{bmatrix} \tag{30}$$

$$\Theta_d = \begin{bmatrix} s_d(1) & 0 & \cdots & 0 \\ s_d(2) & s_d(1) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ s_d(H_p) & \cdots & & s_d(1) \end{bmatrix} \tag{31}$$

The last term of the prediction model, v_k, is a *bias* term, included to account for unmeasured disturbances and model-plant mismatch, introducing both output feedback and integral action in the MPC controller [4]. The bias is given by the difference between the measured output and the open-loop output from the recursive prediction model as follows:

$$v_k = y_k - \tilde{y}_{k|k} \tag{32}$$

where y_k is the measured output, and $\tilde{y}_{k|k}$ is the first element of the prediction state vector Y^p_k. The bias is a cumulative error, as it is not used to update the prediction state vector Y^p_k.

The recursive prediction model can easily be extended to the MIMO case using the superposition principle and block matrices/vectors.

4. Implementing the Proposed Method

The measured disturbance enters the MPC formulation presented in in Section 3 through the vector of disturbance steps defined by (17):

$$\Delta D_k = \begin{bmatrix} \Delta\hat{d}_{k|k} & \Delta\hat{d}_{k+1|k} & \cdots & \Delta\hat{d}_{k+H_p-1|k} \end{bmatrix}^T$$

where H_p is the prediction horizon. The only difference between method A and method B is how the steps $\Delta\hat{d}_{k|k}$ and thus the vector ΔD_k are defined. Note that ΔD_k contains a *prediction* of the disturbance steps from the current time k to the end of the prediction horizon H_p. To derive an expression for ΔD_k, a prediction of the measured disturbance in the prediction horizon is required:

$$\hat{d}_{k+n|k}, \quad 1 \le n \le H_p \tag{33}$$

where $\hat{d}_{k+n|k}$ denotes a prediction of the measured variable d_{k+n} based on information available at time k. Note that all past and current measurements are directly available, so that:

$$\hat{d}_{k+n|k} = d_{k+n}, \quad \forall\, n \le 0 \tag{34}$$

The prediction (33) can either be based on a simple assumption about the future disturbance, e.g. that the measured disturbance will remain constant at the last measurement, or on a more elaborate strategy, e.g. extrapolating the measured disturbance from available measurements. This is further discussed in Section 4.3.

The general idea behind method B is independent of how the measured disturbance is predicted. In Section 4.1, a general expression for ΔD_k is first derived for the conventional method A, and then the corresponding expression for the proposed method B is derived in Section 4.2. The resulting general expressions for ΔD_k for each method are applicable to any given prediction of the measured disturbance on the form (33). Implementing the methods for specific disturbance predictions is discussed in Section 4.3.

4.1. Method A

As the conventional method A uses the measured disturbance directly, the implementation is quite straightforward. A step in the disturbance for method A is given directly from the definition (14):

$$\begin{aligned} \Delta d_k^A &= \Delta d_k \\ &= d_k - d_{k-1} \end{aligned} \tag{35}$$

where the superscript A denotes that this only applies to method A. Given a prediction (33) of the measured disturbance, the predicted disturbance steps in ΔD_k for method A are thus given by:

$$\Delta \hat{d}^A_{k+n|k} = \Delta \hat{d}_{k+n|k} = \hat{d}_{k+n|k} - \hat{d}_{k+n-1|k} \tag{36}$$

Inserting (36) and (34) into (17), the general definition of ΔD_k for method A is given by:

$$\Delta D_k^A = \begin{bmatrix} d_k - d_{k-1} \\ \hat{d}_{k+1|k} - d_k \\ \hat{d}_{k+2|k} - \hat{d}_{k+1|k} \\ \vdots \\ \hat{d}_{k+H_p-1|k} - \hat{d}_{k+H_p-2|k} \end{bmatrix} \tag{37}$$

Note that the first element of ΔD_k^A, which is used to predict y_{k+1}, is calculated directly from the available measurements d_k and d_{k-1}, while the remaining elements (used to predict y_{k+2} onwards) depend on the prediction of the future disturbance.

4.2. Method B

With method B, d_k is replaced with $\bar{d}_k$ from (6), and the disturbance steps are thus calculated as follows:

$$\Delta d_k^B = \bar{d}_k - \bar{d}_{k-1} \tag{38}$$

where the superscript B denotes that this definition only applies to method B. By inserting (6), this is given in terms of the actual measured disturbance:

$$\Delta d_k^B = \frac{1}{2}\left(d_{k+1} - d_{k-1}\right) \tag{39}$$

The predicted disturbance steps with method B are thus given by:

$$\Delta\hat{d}^B_{k+n|k} = \frac{1}{2}\left(\hat{d}_{k+n+1|k} - \hat{d}_{k+n-1|k}\right) \tag{40}$$

Inserting (40) and (34) into (17), the general definition of ΔD_k for method B is given by:

$$\Delta D^B_k = \begin{bmatrix} \frac{1}{2}(\hat{d}_{k+1|k} - d_{k-1}) \\ \frac{1}{2}(\hat{d}_{k+2|k} - d_k) \\ \frac{1}{2}(\hat{d}_{k+3|k} - \hat{d}_{k+1|k}) \\ \vdots \\ \frac{1}{2}(\hat{d}_{k+H_p|k} - \hat{d}_{k+H_p-2|k}) \end{bmatrix} \tag{41}$$

Note that, while the first element of ΔD^A_k is calculated directly from available measurements, all elements of ΔD^B_k, and thus also the prediction of y_{k+1}, depend on the prediction (33) of the future disturbance.

For the step-response prediction model (29), one additional minor change is needed to implement method B; in the recursive prediction model update (25), Δd_{k-1} must be replaced by Δd^B_{k-1}, which is calculated using (39).

4.3. Prediction of the Measured Disturbance

Only the general definitions of ΔD^A_k and ΔD^B_k have been derived so far. For both methods, to implement ΔD_k, a prediction of the measured disturbance in the form (33) is required. Some possible approaches to derive this prediction are presented in this section. The objective here is to give examples of how different approaches to predict/extrapolate future disturbances may be implemented with the two different methods, an exhaustive investigation or a general discussion of all possible approaches is not attempted in this paper.

4.3.1. Constant Disturbance Assumption

The most common approach to predict/extrapolate the future measured disturbance is simply to assume that the measured disturbance will remain constant at the last measurement [2, 4]. This is referred to as the "constant disturbance assumption" in the sequel. Though this is the most commonly implemented assumption, it is often quite inaccurate, but this naturally depends on the disturbance dynamics. The constant disturbance assumption might be a suitable assumption if the measured disturbance is quite random and difficult to predict. An example of this is demonstrated in the SISO example in Section 5.

With the constant disturbance assumption, the prediction (33) is simply given by:

$$\hat{d}_{k+n|k} = d_k, \quad \forall\, n > 0 \tag{42}$$

Inserting this into (37) and (41), and using the definition (14), the specific definitions of ΔD^A_k and ΔD^B_k with the constant disturbance assumption are given by:

$$\Delta D^A_k = \begin{bmatrix} \Delta d_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{H_p} \qquad \Delta D^B_k = \begin{bmatrix} \frac{1}{2}\Delta d_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{H_p} \tag{43}$$

(Note the factor 1/2 in ΔD_k^B.) For both methods, only the first element of the vector ΔD_k is non-zero under this assumption, which simplifies the prediction model somewhat, though this does not reduce the complexity of the MPC optimization problem as ΔD_k is not a decision variable in the MPC formulation.

4.3.2. Linear Change Assumption

The constant disturbance assumption may be suitable for disturbances that change fast and/or randomly, but by design, the dynamics of the measured disturbance are often quite slow compared to the sampling frequency of the controller. For a disturbance that varies slowly and smoothly, the *change* in the measured disturbance is often quite similar for consecutive samples. Consider for example the sine plotted in Figure 2. It is quite clear that a linear extrapolation of this signal is a much better prediction than the constant disturbance assumption, at least for a few steps. It may thus be more accurate (at least for the first few steps in the prediction horizon) to assume that the measured disturbance steps Δd_k rather than the measured disturbance d_k will remain constant in the future. This assumption is denoted the "linear change assumption" in the sequel.

With the linear change assumption, the measured disturbance is predicted/extrapolated as follows:

$$\begin{aligned} \Delta\hat{d}_{k+n|k} &= \Delta\hat{d}_{k+n-1|k} \\ &\Updownarrow \\ \hat{d}_{k+n|k} &= \hat{d}_{k+n-1|k} + \Delta\hat{d}_{k+n-1|k} \\ &= d_k + n\Delta d_k, \quad \forall n > 0 \end{aligned} \tag{44}$$

Inserting (44) into (37) and (41), the specific definitions of ΔD_k^A and ΔD_k^B actually become identical, and are given by:

$$\Delta D_k^A = \Delta D_k^B = \begin{bmatrix} \Delta d_k \\ \Delta d_k \\ \vdots \\ \Delta d_k \end{bmatrix}_{H_p} \tag{45}$$

Although the linear change assumption might be suitable a few time steps into the prediction horizon, it is only bounded by the length of the prediction horizon, and might be quite inaccurate after a few steps, but again, this depends on the disturbance dynamics, as well as the sampling frequency and the length of the prediction horizon.

4.3.3. Extrapolation with an Autoregressive Model

In many cases, some knowledge regarding the dynamics of the measured disturbance is readily available through historical measurement data. This data can be used to estimate a model for the disturbance dynamics, e.g. in the form of an autoregressive (AR) model:

$$A(z)d_k = e_k \tag{46}$$

where e_k is white noise. An example from the literature implementing disturbance predictions in MPC with this approach is given in [5]. The coefficients for the polynomial $A(z)$ are obtained by minimizing the least-squares error between the model and the historical data. This is quite easily done e.g. by using the `ar` function in Matlab. While obtaining an AR model is slightly more complicated than simply assuming that the measured disturbance will remain constant, the implementation is quite straightforward once the model is obtained. The predictions (33) are simply calculated using (46) with $e_k = 0$, and ΔD_k^A and ΔD_k^B are then calculated from (37) and (41). It should also be noted that the computational complexity of the MPC optimization problem is unaltered by this.

4.4. Implicit Linear Change Assumption of Method A

In this section, one of the most fundamental differences between the considered methods is discussed. First consider the prediction model (21) in the MPC formulation. As the MPC control action is calculated once every sampling interval and only the first input of the predicted input sequence is actually implemented, and since all predictions further into the prediction horizon through the prediction model rely on the first predicted output y_{k+1}, a prediction error in the first step has a much larger negative effect on the control performance than a prediction error later in the prediction horizon. Looking at the last term of (21), it is clear that only the first element of ΔD_k is used to predict y_{k+1}. The accuracy of the first element of ΔD_k may thus have a major impact on the control performance, and one of the main differences between method A and method B is in fact the first element of ΔD_k.

Consider the definitions of the vectors ΔD_k^A and ΔD_k^B given by (37) and (41). With method A, the first element of ΔD_k^A is not affected by the prediction of the measured disturbance, but is always given directly from the measured disturbance:

$$\begin{aligned}\Delta\hat{d}^A_{k|k} &= d_k - d_{k-1}\\ &= \Delta d_k \end{aligned}\tag{47}$$

Further, this actually corresponds to the first element of ΔD_k^B with the linear change assumption (44) implemented. In that sense, considering that the prediction model with method B is much more accurate, the linear change assumption is always *implicitly* implemented with method A for the first step in the prediction horizon, whereas the *explicitly* stated disturbance assumption will only affect the predictions from the *second* time step in the prediction horizon. With method B, the first element of ΔD_k^B depends on the predicted disturbance as follows:

$$\begin{aligned}\Delta\hat{d}^B_{k|k} &= \frac{1}{2}(\hat{d}_{k+1|k} - d_{k-1})\\ &= \frac{1}{2}(\Delta d_k + \Delta\hat{d}_{k+1|k}) \end{aligned}\tag{48}$$

Thus, the disturbance prediction will affect the prediction model already from the very important first step of the prediction horizon.

On the one hand, this is a major limitation of method A, as it is inherently restricted to the linear change assumption (44) for the first time step of the prediction horizon, even if a more precise prediction/extrapolation of the measured disturbance is available. As this is not the case for method B, which relies on the disturbance prediction also for the first time step, this implies that method B has a much bigger potential to exploit information about the future measured disturbance to obtain a more accurate prediction of y_{k+1}, and thus improve the control performance. This is demonstrated quite clearly in the SISO example, in Section 5.2.3.

But on the other hand, the most commonly implemented disturbance assumption is the constant disturbance assumption (42), even though this assumption is often quite inaccurate. In most practical implementations, the sampling rate is intentionally chosen very fast compared to the disturbance dynamics. For example, a well-known rule of thumb is to choose the sampling rate 5-10 times faster than the fastest time constant in the system. Then, as discussed in 4.3.2, the linear change assumption is often a much better assumption than the constant disturbance assumption, at least for the first few time steps in the prediction horizon. The fact that method A with the constant disturbance assumption actually implements the linear change assumption for the quite important first time step of the prediction horizon, however unintentional, might in practice improve the control performance quite significantly compared to implementing the constant disturbance assumption with the more precise method B, where also the first time step is based on the explicitly stated assumption. This is also demonstrated in the SISO example, in Section 5.2.1.

4.5. Theoretical Potential

Now consider the prediction of the measured disturbance as a degree of freedom when deriving the controller, completely independent of the actual disturbance dynamics, while keeping in mind that the vector ΔD_k is the only difference between method A and B. Then consider any given prediction of the measured disturbance on the form (33), denoted $\hat{d}^A$. If $\hat{d}^A$ is used with method A, there always exists another prediction $\hat{d}^B$ so that ΔD_k^B with $\hat{d}^B$ inserted is identical to ΔD_k^A with $\hat{d}^A$ inserted, easily obtained by solving $\Delta D_k^B = \Delta D_k^A$ recursively for each element:

$$\begin{bmatrix} \frac{1}{2}(\hat{d}^B_{k+1|k} - d_{k-1}) \\ \frac{1}{2}(\hat{d}^B_{k+2|k} - d_k) \\ \frac{1}{2}(\hat{d}^B_{k+3|k} - \hat{d}^B_{k+1|k}) \\ \vdots \\ \frac{1}{2}(\hat{d}^B_{k+H_p|k} - \hat{d}^B_{k+H_p-2|k}) \end{bmatrix} = \begin{bmatrix} d_k - d_{k-1} \\ \hat{d}^A_{k+1|k} - d_k \\ \hat{d}^A_{k+2|k} - \hat{d}^A_{k+1|k} \\ \vdots \\ \hat{d}^A_{k+H_p-1|k} - \hat{d}^A_{k+H_p-2|k} \end{bmatrix} \tag{49}$$

But since the first element of ΔD_k^A does not depend on the predicted disturbance, then for a given prediction $\hat{d}^B$, the equation (49) can only be solved for $\hat{d}^A$ in the special case where the first element of $\hat{d}^B$ is given by $\hat{d}^B_{k+1|k} = d_k + \Delta d_k$, corresponding to the linear change assumption (44). In other words, for any given disturbance prediction, the performance of method A can always be matched exactly by method B by using a different disturbance prediction, but not the other way around.

Further, if the disturbance prediction can be chosen freely, then all elements of ΔD_k^B can take any value, while the first element of ΔD_k^A is restricted to the measured Δd_k. Let the vector ΔD_k that provides the best possible control performance be denoted ΔD_k^*. While the equation $\Delta D_k^B = \Delta D_k^*$ always has a solution $\hat{d}^B$, the equation $\Delta D_k^A = \Delta D_k^*$ only has a solution $\hat{d}^A$ if the first element of ΔD_k^* is given by Δd_k. This means that only method B *can* be optimal if the first element of ΔD_k^* is not equal to Δd_k.

These results combined imply that, if the prediction of the measured disturbance can be chosen freely, it is always possible to obtain a performance with method B that is equal to or better than the best performance achievable with method A.

4.6. Practical Considerations

While method B has a greater theoretical potential than method A, a more interesting question is which of the methods that performs better in practice, i.e. when the disturbance prediction is based on practical considerations such as known disturbance dynamics, and not simply considered a degree of freedom when deriving the controller. As the prediction model with method B is more accurate for continuous disturbances (as discussed in Section 2), it may be natural to assume that method B also will provide a better control performance than method A when a continuous disturbance is applied. However, there are a number of factors that affect the control performance other than the accuracy of the prediction model formulation, inlcuding:

- The disturbance dynamics and the accuracy of the predicted future disturbance (33)
- The feedback mechanism in the MPC formulation
- The controller configuration and tuning (prediction horizon, sampling frequency, weights, etc.)
- Uncertainties (unmeasured disturbances, measurement noise, model-plant mismatch, etc.)

Given all these factors, and especially the many possible approaches to derive a prediction/extrapolation of the future measured disturbance, one cannot conclude that method B will perform better than method A in general, but it should be clear from the discussion so far that method B is fundamentally more precise than method A, and has a greater potential. This is sought to be demonstrated through the examples in the following sections.

5. Closed-Loop Simulations of a SISO Example System

In this section, closed-loop simulations are performed with the following first-order system in a continuous-time state-space formulation:

$$\begin{aligned} \dot{x}(t) &= -x(t) + u(t) + d(t) \\ y(t) &= x(t) \end{aligned} \tag{50}$$

The system is simulated with the control input $u(t)$ determined by the MPC controller as formulated in Section 3, implementing both the conventional method A and the proposed method B as discussed in Section 4.

Note that the simulations in this section are not intended to illustrate realistic control problems, but to clearly demonstrate the fundamental characteristics of the two considered methods and the theoretical results from Section 4.

5.1. Simulation Setup

5.1.1. Model representation

The results presented in this example are based on a state-space prediction model, as described in Section 3.2.1. Simulations with a step-response formulation show very similar results, and as the main findings are the same with both representations, the results with step-response models are omitted from this presentation.

5.1.2. Controller Tuning

In the initial simulations in this example, the MPC controller is tuned as an unconstrained dead-beat controller, i.e. there is only a weight on the output and no weight on the input moves ($q = 1$, $p = 0$ in (10a)), there are no constraints on the input or the output (the constraints (10b)-(10d) are omitted), and the prediction horizon is just one step ($H_p = 1$), as the controller will always predict a zero output error after the first step. This is not meant to imitate a practical or realistic controller tuning. A dead-beat tuning is a very aggressive tuning, and usually a less aggressive tuning is required in practical implementations due to uncertainties (such as measurement noise, modeling errors, unmeasured disturbances, etc.) and other considerations in the system (e.g. limitations on the actuators, wear and tear, etc.). However, in this simple example, with a perfect system model, and no unmeasured disturbances, a dead-beat tuning is quite reasonable. Also, with this tuning, *the system output will be equal to the prediction error* of the MPC controller, which is very convenient when discussing the results and comparing the considered methods.

5.2. Smooth Sine Disturbance

First the following measured disturbance is considered:

$$d(t) = \sin(\pi t) - \frac{1}{2}\sin(2\pi t) \tag{51}$$

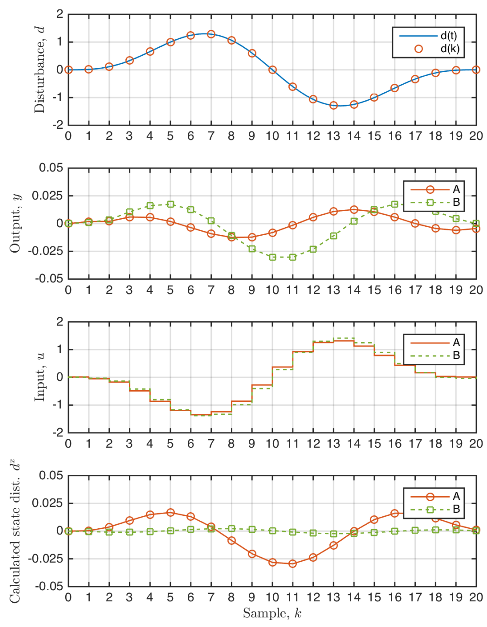
simulated with sampling interval $T = 0.1$ from $t = 0$ to $t = 2$ (one period, 20 samples). This disturbance is denoted the "smooth sine" disturbance in the sequel. This disturbance is very smooth and slowly varying, and thus relatively easy to extrapolate precisely, which makes it very suitable for the discussion in this paper.

5.2.1. Constant Disturbance Assumption

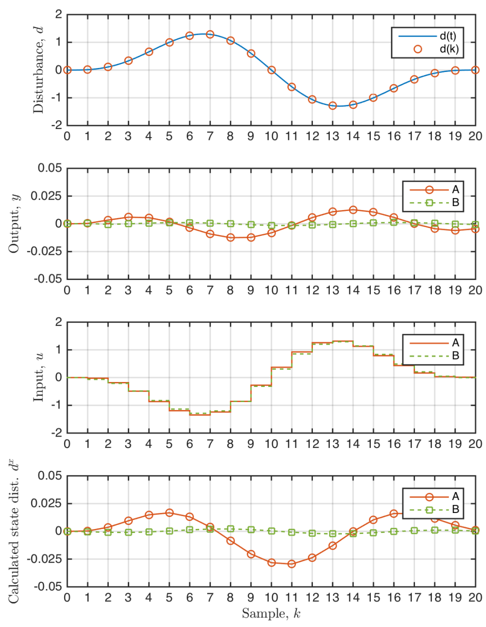
First, the SISO system (50) is simulated with the smooth sine disturbance (51) using the commonly implemented constant disturbance assumption (42) for disturbance prediction. ΔD_k^A and ΔD_k^B are then given by (43). These simulation results are shown in Figure 3a. The measured disturbance is shown in the top plot, the outputs (equal to the prediction error) for each of the two methods in the second plot, the inputs in the third plot, and the calculated state disturbance d^x defined in (19) in the bottom plot.

This simulation shows that, even though method B is based on a more accurate prediction model, the conventional method A actually provides a better control performance for this scenario. However, it is quite obvious that the constant disturbance assumption is quite inaccurate for the smooth sine disturbance, and the linear change assumption discussed in Section 4.3.2 would be a much better choice. As discussed in Section 4.4, method A actually ignores the explicitly stated disturbance assumption for the first time step of the prediction horizon, and instead implicitly implements the linear change assumption. As the linear change assumption is more accurate than the constant disturbance assumption, the result is that the predictions with method A are more accurate than with method B, even though the underlying prediction model is less accurate. That the prediction model is more accurate with method B is quite clear in the bottom plot of Figure 3a, where it is shown that method A operates with a much larger calculated state disturbance d^x (analogous to the plant-model mismatch) than method B.

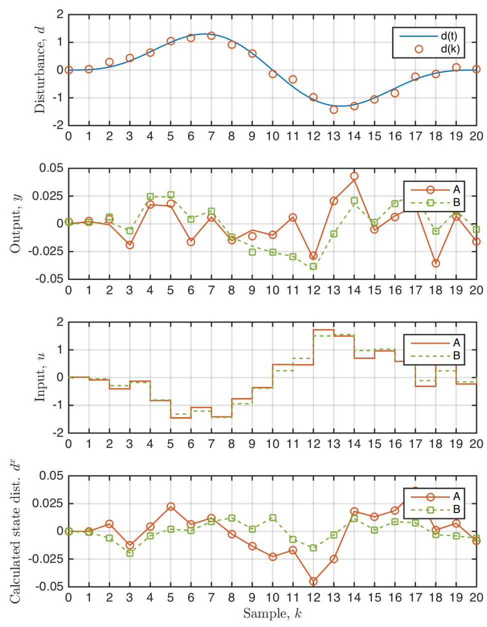
Taking a closer look at the simulation results in Figure 3a, keeping in mind that the output is equal to the prediction error with the dead-beat tuning implemented, it can be seen quite clearly that the output (and thus the prediction error) with method B is directly proportional to the disturbance steps Δd. Method B provides very good predictions (output close to zero) when the disturbance is nearly constant, e.g. at samples 7 and 14, and poor predictions when the disturbance is changing rapidly, e.g. around sample 11. This is exactly the behavior one would expect from implementing the constant disturbance assumption. On the other hand, the prediction error with method A is directly proportional to the *change* in the disturbance steps. The predictions are most accurate when the disturbance is changing steadily in one direction, e.g. at samples 5, 11 and 17, which is exactly the behavior one would expect from implementing the linear change assumption.



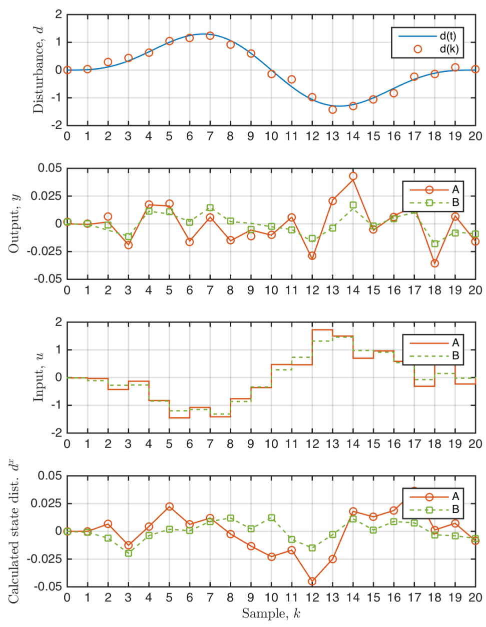
(a) Constant disturbance assumption, noise free



(b) Extrapolation based on AR-model, noise free



(c) Constant disturbance assumption, with measurement noise



(d) Extrapolation based on AR-model, with measurement noise

Figure 3: Simulation results with the “smooth sine” disturbance

This confirms that only method B is actually true to the explicitly stated constant disturbance assumption, while method A instead implicitly implements the linear change assumption.

It should also be noted that the state disturbance with method A is in fact nearly identical to the prediction error with method B. This shows that with method A, the mismatch between the explicitly stated disturbance assumption and the implicit linear change assumption is interpreted as a plant-model mismatch by the controller. On the other hand, even though the prediction error with method B in this scenario is larger than with method A, the state disturbance with method B is nearly negligible. This shows that the prediction error with method B is correctly interpreted as a mismatch between the predicted and the actual disturbance.

5.2.2. Linear Change Assumption

Given the results in Section 4.3.2, it is to no surprise that simulation results with the linear change assumption are identical for the two methods, and the same as the results with method A in Figure 3a.

5.2.3. Accurate Disturbance Prediction

The linear change assumption is clearly more accurate than the constant disturbance assumption for the smooth sine disturbance. But as discussed in Section 4.3.3, an even more accurate extrapolation of the measured disturbance may be obtained using an identified AR model. An AR-model for the smooth sine disturbance is now obtained in Matlab using the function `ar`, based on the measurement data from one period of the smooth sine disturbance. The resulting AR-model is given by:

$$d(k) = 3.5201 \cdot d(k-1) - 5.0777 \cdot d(k-2) + 3.5201 \cdot d(k-3) - 1.0000 \cdot d(k-4) \tag{52}$$

The dynamics of this 4th order AR-model nearly perfectly matches the smooth sine disturbance (as long as at least 4 previous measurements are available). To minimize transient effects in the simulation, it is assumed that previous measurements are available, so that the AR model is initialized correctly. The simulation results are shown in Figure 3b.

These results show that with an accurate prediction of the measured disturbance, method B is indeed more precise and by far outperforms the conventional method A. Method B now provides a near perfect control, while method A does not benefit at all from the precise prediction of the measured disturbance, which is due to the fact that the first element of ΔD_k^A does not depend on the predicted disturbance, in combination with the dead-beat tuning and the one-step prediction horizon ($H_p = 1$). This is clearly a major limitation of method A. (It is shown later, in Section 5.4, that also method A benefits from a more precise extrapolation of the measured disturbance with a less aggressive tuning and $H_p > 1$, though not as much as method B.)

5.2.4. Measurement Noise

The simulations in Sections 5.2.1 and 5.2.3 are now repeated, but this time measurement noise in the form of white Gaussian noise is added both to the measured disturbance and the measured output. In these simulations, the noise on the measured disturbance has standard deviation $\sigma_d = 0.1$, while the noise on the output has standard deviation $\sigma_y = 0.002$. The Matlab function `ar` is again used to obtain an AR-model to extrapolate the measured disturbance, but this time the model is estimated based on noisy measurements from 200 samples (corresponding to 10 periods of the smooth sine disturbance), and a model order of 10 is chosen to provide better noise filtering.

			Control error (MSE of y, 10^{-6})			**Aggressiveness** (Mean $\|\Delta u\|$)		
Scenario	**AR**	**Noise**	**A**	**B**	**%**	**A**	**B**	**%**
Smooth	-	-	51.8	231.8	+347.4	0.27	0.28	+6.3
Sine	ON	-	51.8	0.7	-98.7	0.27	0.26	-3.0
	-	ON	326.2	341.1	+4.6	0.51	0.38	-26.0
	ON	ON	326.2	84.8	-74.0	0.51	0.31	-39.7
Random	-	-	767	379	-50.5	0.73	0.44	-40.0
	-	ON	917	491	-46.5	0.82	0.51	-37.6

Table 1: Measures of performance

The results of the simulations with measurement noise are shown in Figures 3c and 3d. Due to the noise introduced in the system, the results are not as easily compared in the figures, but some measures of performance are shown in Table 1. The column “Control error” shows the Mean Square Error (MSE) on the output for each of the simulations, and the performance increase/decrease for method B relative to method A, in percent. The column “Aggressiveness” shows how actively the control input is used in each simulation, i.e. the mean absolute value of the input steps Δu.

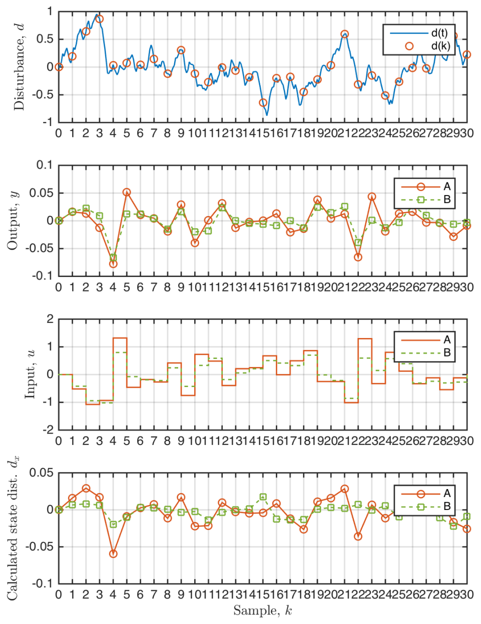
The results with the constant disturbance assumption (Figure 3c and the third row in Table 1) show that the performance of the two methods are now quite comparable, with only a 4.6% difference. Comparing the control error with and without measurement noise, it is clear that method A is affected a lot more severely by the measurement noise than method B. This may be attributed to the implicit linear change assumption, as method A will always predict that a change in the measured disturbance will be repeated in the next time step, and thus overestimate the effect of any measured change that is simply due to measurement noise. The result is a too aggressive controller, with large steps on the input. As seen in the last column of Table 1, the input in this case is used 26% less actively with method B than method A when measurement noise is added, even though the control error is relatively similar.

Further, the results show that when the measured disturbance is extrapolated using the identified AR model, method B by far outperforms method A, with a 74% smaller MSE on the output, while method A again does not benefit at all from the more precise extrapolation. The aggressiveness of the controller with method B is also further reduced, this time by 39.7% compared to method A.

These results indicate that in a realistic scenario with noisy measurements, it may be possible to achieve a quite decent control performance with method B, even with an inaccurate disturbance prediction, and a less aggressive controller may be expected with method B than with method A. Further, as in the noise free case, when the measured disturbance is extrapolated more accurately, method B may be expected to outperform method A both with respect to control error and aggressiveness.

5.3. Random Disturbance

To contrast the smooth and slowly varying smooth sine disturbance, the system is now simulated with a lot more random disturbance based on filtered white noise. Due to the highly random


(a) Noise free

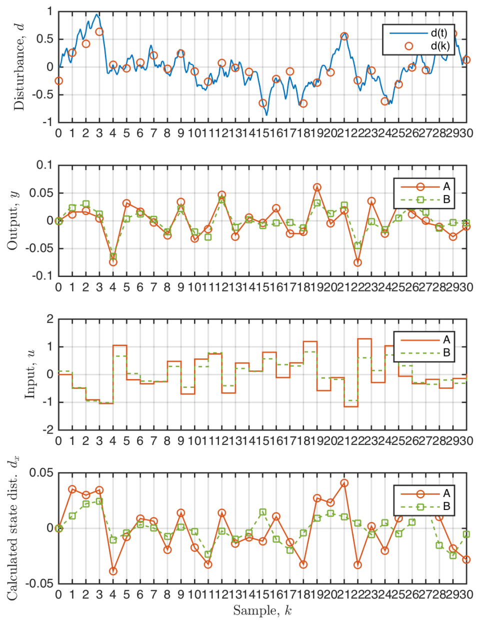

(b) With measurement noise

Figure 4: Simulation results with the “random” disturbance

dynamics of this disturbance, the constant disturbance assumption is presumably a very suitable disturbance prediction, and is thus the only disturbance prediction considered in this scenario. Noise free simulation results with this disturbance are shown in Figure 6a, while measurement noise is added in the simulations shown in Figure 6b. (Measurement noise with standard deviation $\sigma_d = 0.1$ is added to the measured disturbance, and measurement noise with standard deviation $\sigma_y = 0.002$ is added to the measured output.) The measures of performance are given in the last two rows of Table 1.

As this disturbance changes rapidly and randomly, the implicit linear change assumption in method A does not work well in this scenario, and excessively large spikes are experienced on the output. Both with and without measurement noise, the control error is about halved with method B, and the aggressiveness of the controller is reduced by about 40%. These results confirm that method B outperforms method A when the disturbance assumption matches well with the real disturbance dynamics.

5.4. Tuning

So far, only a dead-beat tuning has been considered, i.e. without any move penalty and with a one-step prediction horizon ($p = 0$ and $H_p = 1$ in (10a)). The dead-beat tuning is generally considered too aggressive for practical implementations, but was convenient when comparing the two methods. The effect of tuning with the two considered methods is now investigated. For this purpose, the simulations in Section 5.2 and 5.3 are repeated with a move penalty p ranging from 10^{-5} to 100. A prediction horizon $H_p = 10$ is used in these simulations.

5.4.1. Tuning with the Smooth Sine Disturbance

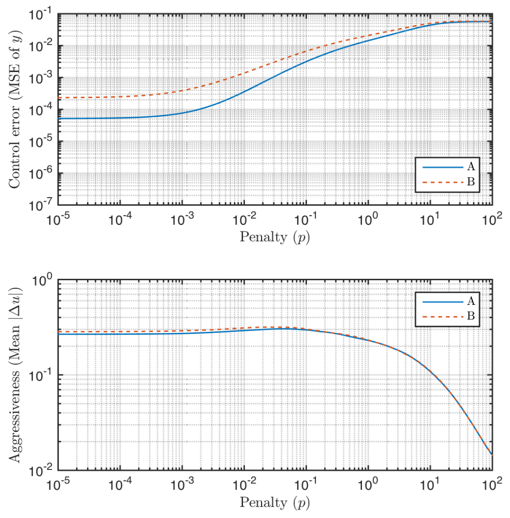
Considering again the smooth sine disturbance from Section 5.2, the control error (MSE of the output y) and the aggressiveness of the controller (mean $|\Delta u|$) with a varying move penalty p are shown in Figure 5. (Note that a logarithmic scale has been used for all axes.)

The results in Figure 5a and 5c show that with the constant disturbance assumption, method A performs better than method B regardless of the move penalty. Without measurement noise, the aggressiveness is quite similar with the two methods, but when measurement noise is introduced, method A is clearly more aggressive than method B, at least for relatively small move penalties, in which case the performance of method A is also just slightly better than method B.

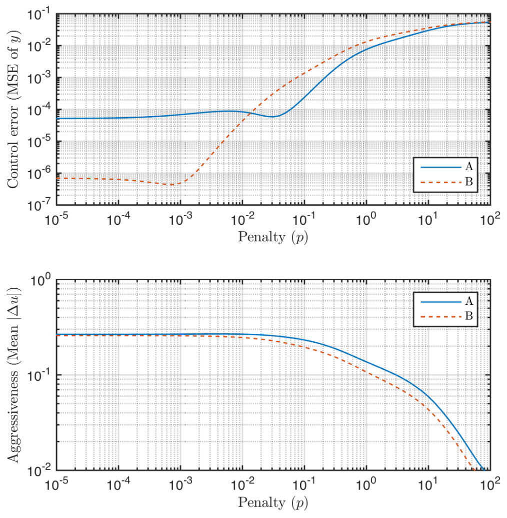
The results in Figure 5b and 5d show that while method A does not benefit from a more accurate extrapolation of the measured disturbance with a dead-beat tuning, it does benefit from this when $p > 0$. However, the optimal performance of method B is still far better than the optimal performance of method A. On the other hand, method B only has a better control performance then method A when the move penalty is relatively small, and method A performs better for large p. But it should also be noted that when method A performs better, method A is also more aggressive than method B. Considering the discussion in Section 4.4, this is quite reasonable. Due to the implicit linear change assumption, method A always overestimates the effect of the measured disturbance, and thus makes more aggressive moves to counteract this. When p is restrictive enough to reduce the control performance of method B, the increased aggressiveness of method A counteracts some of the restrictiveness of the move penalty, which results in a reduced control error.

Another quite interesting side effect of the increased aggressiveness with method A is seen in Figure 5d. Here the performance with method A actually improves by increasing the move penalty, and the optimal performance is achieved with a with a penalty $p = 0.0316$. This indicates that method A is inherently too aggressive and thus very sensitive to measurement noise, and actually benefits from being restricted by a move penalty $p > 0$, even though this is quite counter-intuitive. Similar tendencies are also seen in Figure 5b and Figure 5c. On the other hand, the results show that method B (with a couple minor exceptions) consistently shows a better performance with a smaller move penalty, which is a lot more intuitive. One exception is in Figure 5b, where also method B shows a better performance with $p > 0$. This is, however, on a lot smaller scale, with a control error less than 10^{-6}, which is really negligible in a realistic scenario with measurement noise, model-plant mismatch, etc. Also in Figure 5d, the optimal performance is achieved with $p > 0$ (with $p \approx 10^{-4}$), but the performance difference compared to $p = 0$ is negligible.

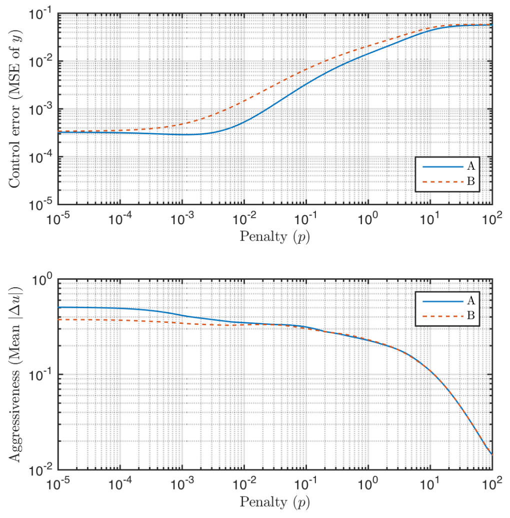
The control error and aggressiveness of the methods with optimal tuning are shown in Table 2. These results show that, while the dead-beat tuning is near optimal with method B, method A benefits a lot from an increased move penalty when measurement noise is added. Interestingly, the aggressiveness of the methods is a lot more similar with the optimal tuning than with the dead-beat tuning shown in Table 1, which again indicates that method A is inherently too aggressive with a dead-beat tuning. However, even with an optimal tuning, when measurement noise is added and the disturbance is extrapolated using the AR model, the control error is still nearly halved with method B. On the other hand, method B performs 17.8% worse than method A when the much less accurate constant disturbance assumption is implemented, even with measurement noise. Once again, this confirms that method B outperforms method A when an accurate extrapolation of the disturbance is implemented, while method A may benefit from the implicit linear change



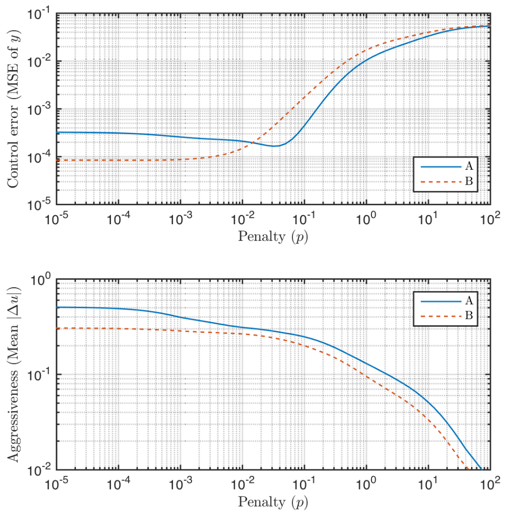
(a) Constant disturbance assumption, noise free



(b) Prediction based on AR model, noise free



(c) Constant disturbance assumption, with measurement noise



(d) Prediction based on AR model, with measurement noise

Figure 5: Control error and aggressiveness versus move penalty with the “smooth sine” disturbance

			Move penalty (p, 10^{-3})		**Control error** (MSE of y, 10^{-6})			**Aggressiveness** (Mean $\|\Delta u\|$)		
Sce.	**AR**	**Noise**	**A**	**B**	**A**	**B**	**%**	**A**	**B**	**%**
Sine	-	-	0	0	51.8	231.8	+347.4	0.27	0.28	+6.2
	ON	-	0	0.6	51.8	0.4	-99.2	0.27	0.26	-3.3
	-	ON	1.3	0	289.1	340.5	+17.8	0.40	0.38	-7.1
	ON	ON	31.6	0.2	165.6	84.4	-49.0	0.28	0.30	+5.2
Rand	-	-	20.0	0.4	447	374	-16.3	0.35	0.40	+15.6
	-	ON	15.8	0.5	550	481	-12.6	0.40	0.46	+14.2

Table 2: Measures of performance with optimal tuning

assumption if this is better than the explicitly stated disturbance prediction.

5.4.2. Tuning with the Random Disturbance

With the random disturbance from Section 5.3, the control error and the aggressiveness of the controller with a varying move penalty are shown in Figure 6, and the measures of performance are given in the last two rows of Table 2. The main findings are:

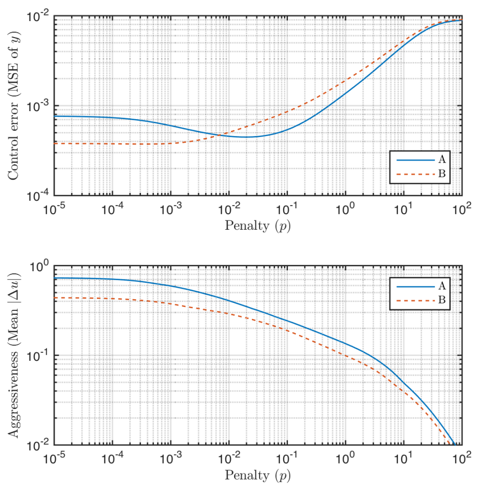
- Method A is always more aggressive than method B if the same move penalty is used
- The optimal performance with method A is achieved with $p > 0$, while with method B, the optimal performance is achieved with p close to 0.
- The optimal performance with method B is better than the optimal performance with method A
- The tuning of the controller is again a lot more intuitive with method B

These results are the same both with and without measurement noise. Again, it may be concluded that method B performs better than method A when the disturbance assumption matches well with reality.

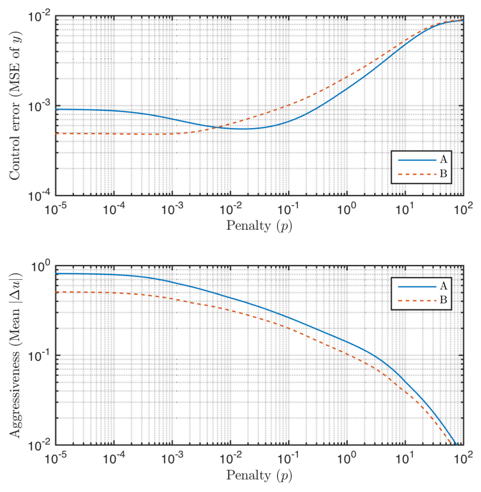
It should be noted, though, that in this case, method A with the optimal tuning is actually less aggressive than method B with optimal tuning. But then again, in the simulations with measurement noise, with $p = 1.633 \cdot 10^{-3}$, the aggressiveness with method B is the same as the aggressiveness of method A with the optimal tuning, while the control error is $497 \cdot 10^{-3}$, which is 9.7% better than the optimal performance with method A. The results without measurement noise are similar. This shows that when the aggressiveness is identical, method B still performs better than method A in this scenario.

6. MIMO Example

In this section, the considered methods are compared in closed-loop simulations of a realistic industrial multivariable system. The system and the control problem were thoroughly presented and discussed in [20] and [21], and only a brief summary is presented here.



(a) Noise free



(b) With measurement noise

Figure 6: Control error and aggressiveness versus move penalty with the "random" disturbance

6.1. System Description

The system considered in this example is an oil production well with an electric submersible pump (ESP) installed, as shown in Figure 7. The ESP is installed inside the well to create artificial lift, in order to boost the production from the well, and improve recovery from the reservoir. There are many control challenges related to an ESP installation. Failure of an ESP installation has a huge economic impact, both due to production loss and the cost of replacing the pump. The main priority in this system is thus to maintain acceptable operating conditions for the ESP, to prevent failure or reduced life-time of the pump. There are many variables that affect the life-time of an ESP, but this example focuses on thrust forces acting on the pump shaft, and the power consumption of the pump motor.

6.2. Control Problem

An outline of the control problem considered in this example is given below. More details regarding the system, modeling and associated control concerns may be found in [20] and [21].

6.2.1. Control Inputs

The control inputs in the system are the pump frequency (or speed), denoted f, and the production choke valve opening, denoted z.

6.2.2. Outputs and Control Objectives

The main control objective is to sustain a given production rate from the well. As the inflow into the well (and thus the production rate) is determined by the difference between the reservoir pressure (p_r) and the bottomhole pressure inside the well (p_{bh}), this may be achieved by keeping the bottomhole pressure at a desired setpoint. Under the assumption that a constant pressure at the

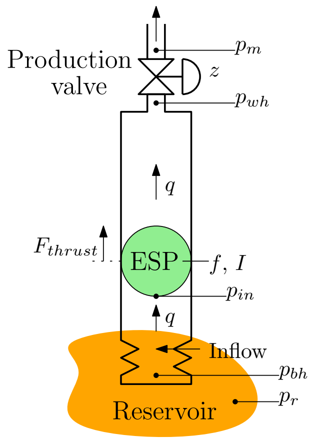


Figure 7: MIMO example: Oil production well with an ESP installed

inlet of the ESP will ensure a constant bottomhole pressure, this is achieved indirectly by keeping the ESP inlet pressure, denoted p_{in}, at a desired setpoint.

At the same time, thrust forces acting on the ESP shaft (denoted F_{thrust} in Figure 7) must be limited to prevent excessive wear and tear on the pump, which could lead to premature failure of the ESP installation. This may be achieved indirectly by controlling the relative pump flow q_0, which was defined in [20] as:

$$q_0 = q\frac{f_0}{f} \tag{53}$$

where f_0 is the reference frequency for the pump characteristics assumed to be available. (See [20] for details.) The relative pump flow should be kept within certain bounds, and preferably close to an optimal setpoint.

Finally, the power consumption of the pump must be limited with regard to the life-time of the ESP motor, and preferably minimized to reduce operation costs. This is achieved by limiting and minimizing the electric current through the ESP motor, denoted I, which is directly proportional to the power consumption.

6.2.3. Disturbance

The main disturbance in this example is the manifold pressure, i.e. the pressure at the outlet of the well (denoted p_m in Figure 7), which is a measured disturbance. This pressure may vary considerably due to other components in the production system, such as booster pumps, separators and other wells producing to the same manifold, especially when such components are started up or shut down.

6.2.4. Measurement Noise

Measurement noise (normally distributed white noise with standard deviation 0.1 bar) is added to the measured disturbance (p_m), while perfect measurements of the outputs are used in this example for an easier comparison of the results.

Variable	**Unit**	**Low limit**	**Setpoint**	**High limit**	**Weight**	
					Method A	Method B
p_{in}	bar	40	50	70	1	1
q_0	m^3/h	40	50	60	0.1	0.1
I	A	-	0	65	$1{\cdot}10^{-4}$	$1.8{\cdot}10^{-4}$
Δf	Hz	-0.5	-	0.5	2	1.85
Δz	$\%$	-0.5	-	0.5	0.2	0.185
f	Hz	35	-	65	-	-
z	$\%$	0	-	100	-	-

Table 3: Controller settings and tuning

6.3. Simulator and Prediction Model

Based on a model of the system derived in [20] and [21], a simulator for the considered system is implemented in Matlab. An MPC controller based on a step-response prediction model, as described in Section 3.2.2, is also implemented in Matlab. The step-response models are obtained from the simulator by applying steps in the inputs with the system at a steady state close to the desired operating point. The simulator model and the prediction model in this example are thus not the same.

6.4. Controller Configuration and Tuning

As shown in Section 5.4, the effect of tuning is quite different for the two methods. Specifically, due to the implicit linear change assumption in method A, the controller is usually more aggressive with method A than with method B. As reducing wear and tear on the installation is vital in this system, the aggressiveness of the controller is very important when evaluating the performance. To make the performance of the considered methods as comparable as possible, the controller is tuned differently for the two methods in this example, so that the level of aggressiveness is similar for the two methods. The difference in performance is then seen mainly on the outputs of the system, and the performance is thus more directly comparable. To achieve this, a tuning that provided a decent performance[2] with respect to the control objectives defined above was first found for method A, and then the move penalties p on the control inputs were slightly reduced with method B to obtain a comparable aggressiveness. In addition, the weight on the output I was increased with method B to achieve a similar power consumption. The implemented control targets (constraints and setpoints) and tuning parameters for each method are given in Table 3. The sampling interval is set to $T_s = 1$ second, the prediction horizon is set to $H_p = 10$, and the measured disturbance is extrapolated using the constant disturbance assumption.

[2]Tuning of a multivariable MPC configuration is not a straightforward task, and we do not claim to have found the best possible tuning for neither method in this example. The main concern was rather to make the performance of the two methods easily comparable.

	Constant dist. assumption			**Extrapolated disturbance**		
	A	B	%	A	B	%
p_{in} (MSE)	$36.4{\cdot}10^{-3}$	$29.6{\cdot}10^{-3}$	-18.5	$32.7{\cdot}10^{-3}$	$24.3{\cdot}10^{-3}$	-25.7
q_0 (MSE)	0.659	0.671	+1.7	0.661	0.662	+0.1
I (Mean)	24.1	24.1	-0.0	24.1	24.1	+0.0
$\lvert\Delta f\rvert$ (Mean)	0.111	0.111	-0.1	0.113	0.115	+2.3
$\lvert\Delta z\rvert$ (Mean)	0.350	0.349	-0.0	0.350	0.351	+0.3
Pred. error, p_{in} (MSE)	$16.9{\cdot}10^{-3}$	$13.1{\cdot}10^{-3}$	-22.8	$15.8{\cdot}10^{-3}$	$12.2{\cdot}10^{-3}$	-22.7

Table 4: MIMO simulation results

6.5. Simulation Results

Simulation results with the above controller configuration are shown in Figure 8a. The top plot shows the measured disturbance, i.e. the manifold pressure p_m. The disturbance is the same for both methods, the real disturbance is plotted with a solid black line, and the measurements (including measurement noise) are plotted with red dots. In the remaining plots, method A is plotted with a solid blue line and method B with a dotted red line. The outputs are shown in the next three plots, i.e. the ESP inlet pressure p_{in}, the relative pump flow q_0 and the electric current of the ESP I. The outputs are plotted relative to their setpoints, so that the setpoint is at zero in the plots. The next two plots show the inputs, i.e. the pump frequency f and the choke opening z. The bottom plot shows the prediction error for the ESP inlet pressure.

Some measures of performance are given in the column “Constant dist. assumption” in Table 4, which shows the results for each of the methods, and the difference in percent. The first row shows the mean square error (MSE) for tracking of the setpoint for the ESP inlet pressure p_{in}, the second row shows the MSE for the setpoint for the relative flow q_0, the third row shows the average current I, the next two rows show the aggressiveness of the controller, i.e. the average step size of the inputs f and z, and the last row shows the MSE for the prediction error for the ESP inlet pressure p_{in}.

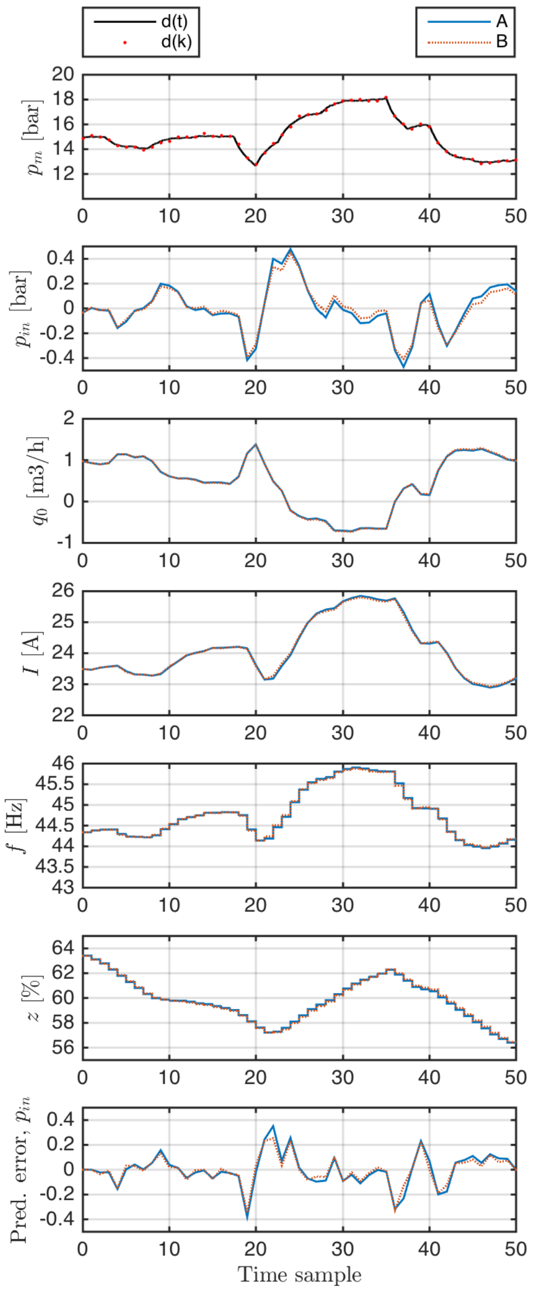
As seen in the table, the aggressiveness of the two methods as well as the power consumption is nearly identical with this tuning, but tracking of the inlet pressure setpoint (the main control target) is improved by 18.5% with method B compared to method A, though tracking of the relative pump flow is 1.7% less accurate.

6.6. Extrapolated Disturbance

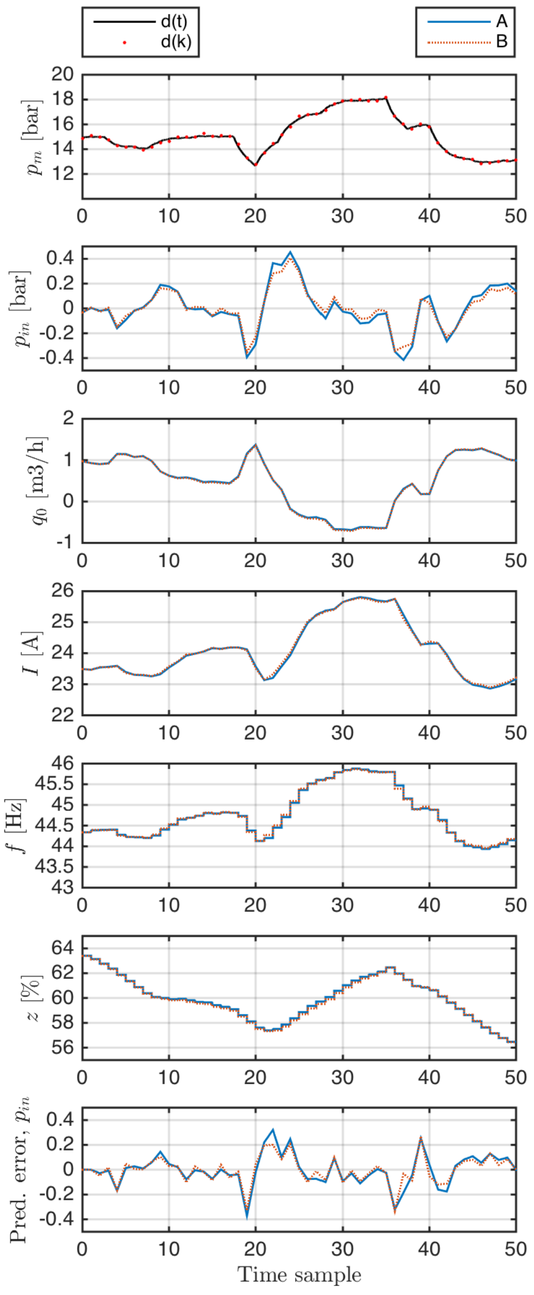
Next, as in the SISO example, a more accurate extrapolation of the measured disturbance is implemented in the prediction model, based on an AR model derived from logged measurement data. A 3rd order model is used in this example, given by:

$$\hat{d}_{k+1|k} = 1.5382\,d_k - 0.6571\,d_{k-1} + 0.1182\,d_{k-2} \tag{54}$$

Simulation results with this disturbance extrapolation are shown in Figure 8b, and some measures of performance are given in the column “Extrapolated disturbance” in Table 4. The results show that the control performance is improved by this extrapolation with both methods, but method B benefits more from the extrapolation than method A, and tracking of the pressure setpoint is now 25.7% better with method B than with method A. The tracking of q_0 has also improved slightly, but the aggressiveness is also slightly increased.



(a) Constant disturbance assumption



(b) Extrapolated disturbance

Figure 8: Simulation results of the MIMO system

	Original results			**Extrapolated disturbance**		
	Method A	Method B	%	Method A	Method B	%
p_{in} (MSE)	$6.16 \cdot 10^{-3}$	$5.16 \cdot 10^{-3}$	-16.2	$5.02 \cdot 10^{-3}$	$3.61 \cdot 10^{-3}$	-28.0
q_0 (MSE)	0.417	0.450	+7.9	0.410	0.418	+1.8
I (Mean)	24.4	24.3	-0.2	24.3	24.3	-0.2
$\|\Delta f\|$ (Mean)	0.0578	0.0579	+0.2	0.0569	0.0586	+3.1
$\|\Delta z\|$ (Mean)	0.272	0.272	+0.1	0.287	0.284	-1.1
Pred. error, p_{in} (MSE)	$1.86 \cdot 10^{-3}$	$1.09 \cdot 10^{-3}$	-41.6	$1.71 \cdot 10^{-3}$	$1.00 \cdot 10^{-3}$	-41.2

Table 5: MIMO simulation results with increased sampling rate

6.7. *Sampling Rate*

The performance of MPC is generally increased with a higher control update rate, but the update rate is often restricted by the computationally demanding MPC algorithm, which usually involves numerically solving a quadratic programming (QP) problem online (in real-time). Depending on the application, the sampling rate of the measurements may also be a limitation. For example, instruments installed inside oil production wells often have a very slow sampling rate.

So far in this paper, the control update rate has been considered fixed, but from the discussion in Section 2, it is natural to assume that the inaccuracies addressed in this paper might be reduced by increasing the sampling rate. Considering this, the MIMO system is now simulated with a doubled sampling rate ($Ts = 0.5$). A new AR model for extrapolation is found for this sampling rate, given by:

$$\hat{d}_{k+1|k} = 1.4006\, d_k - 0.2607\, d_{k-1} - 0.1402\, d_{k-2} \tag{55}$$

The simulation results (in numbers) are given in Table 5. Compared to Table 4, these results show that the control performance is indeed significantly improved by doubling the sampling rate, but interestingly, the performance improvement from implementing method B compared to method A is fairly similar for both sampling rates. This indicates that the issues addressed in this paper are quite independent of the sampling rate, and increasing the sampling rate is thus not a valid reason to not implement method B.

If the measured disturbance is sampled more often than the control update rate, this could be exploited for better noise filtering and/or more accurate extrapolation of the measured disturbance, without altering the control update rate. Considering the results in this paper, method B would probably benefit more from such extra information than method A.

7. Conclusions

The results in this paper show that a system discretized using ZOH is quite inaccurate when a measured disturbances sampled from a continuous variable is applied as an input. This is nevertheless the standard method for MPC implementations based on state-space models, and the equivalent to step-response models. It was shown that discretizing the system using FOH would be more precise, but quite impractical in the MPC framework. Thus a much simpler approach that approximates FOH discretization, and is very easy to implement in the MPC framework, was proposed, denoted method B. It was shown that method B is much more precise for continuous inputs than the conventional method, denoted method A.

In MPC, future values of the measured disturbance must also be predicted. This is often done simply by assuming that the measured disturbance remains constant in the future, denoted the "constant disturbance assumption", though more accurate extrapolations may be more appropriate. It was shown through an analytical comparison of the methods that method B has a greater theoretical potential than method A, but simulation results show that the performance in practice depends on how accurately the measured disturbance is predicted. The comparison also revealed that due to the inaccuracy of method A, it does not actually implement the explicitly stated prediction of the measured disturbance in the first step of the prediction horizon, but instead implicitly predicts that a change in the measured disturbance always will be repeated in the first step, denoted the "implicit linear change assumption". The proposed method B, on the other hand, implements the explicitly stated disturbance prediction correctly.

It was shown in the SISO example that the implicit linear change assumption embedded in method A can in some cases actually be a benefit, if the implicit linear change assumption is a better match with the actual disturbance dynamics than the explicitly stated disturbance prediction. For example, the implicit linear change assumption is often much better than the constant disturbance assumption for smooth and slowly varying disturbances. But it was also shown that the linear change assumption may also be implemented with method B, if stated explicitly, and the performance of the two methods will then be identical. And further, it was shown that if the explicitly stated disturbance assumption is better than the linear change assumption, the more precise method B provides a much better control performance than the conventional method A, which benefits very little from a more accurate prediction. These results confirm that method B is conceptually more precise than method A, and that method B has a greater potential than method A.

The simulation results also indicate that due to the implicit linear change assumption in method A, method A will usually result in a controller that is more aggressive than method B with the same tuning, and also more sensitive to measurement noise. Due to this, method A relies on a more restrictive controller tuning than method B to achieve optimal performance, while the controller tuning with method B is a lot more intuitive. The performance of the two methods is thus not directly comparable with the same tuning.

Very promising results for method B were shown also in the more realistic MIMO example. The control error for the main control objective was reduced significantly by implementing method B, even when the standard constant disturbance assumption was implemented, and even more when a more accurate extrapolation was considered, though the controller with method A also benefited some from the extrapolation.

Extrapolating the measured disturbance does not increase the complexity of the MPC optimization problem that is solved online, and very little effort is often required to obtain a more accurate extrapolation of a measured disturbance than the constant disturbance assumption. But as shown in this paper, due to the fact that the conventional method A to some extent ignores the explicitly stated disturbance prediction, the effort to derive and implement a more accurate extrapolation is barely rewarded in a conventional MPC implementation. This might be one of the main reasons that the constant disturbance assumption has remained so popular, while a more accurate extrapolation of the measured disturbance is rarely implemented in practice, and very few examples of this exist in the literature. On the other hand, with method B, the MPC controller benefits a lot from a more accurate prediction of the measured disturbance, as it should. Implementing method B may thus enable significant improvements of the control performance through better disturbance predictions.

Implementing the proposed method B only requires a minor modification in the prediction model, comparable to filtering the measured disturbance, and does not rely on which model representation is implemented in the prediction model. Both state-space models and step-response models were considered in this paper, with very similar results. Method B may thus improve the control performance significantly with a minimal effort.

Acknowledgments

This work was supported by the Research Council of Norway and Statoil through the PETROMAKS project 215684. The main author is also grateful for the considerable amount of understanding and support from his wife when continuing the work after the funds were depleted.

AppendixA. Deriving Method B for a First-Order System

In this appendix, it is shown how $\bar{d}_k$ which is used in the proposed method B is derived. First, in AppendixA.1, an expression for an optimal $\bar{d}_k$ (denoted $\bar{d}_k^*$) is found that under certain assumptions provides an accurate match between the continuous system with $d(t)$ as input and the discrete-time system with $\bar{d}_k^*$ as input. Then, in AppendixA.2, it is shown that this is in fact equivalent to FOH-discretization, but leads to a system on a different form. In AppendixA.3, it is shown that the simple average of d_k and d_{k+1}, which is the proposed solution in this paper, is a good approximation of the optimal $\bar{d}_k^*$ if the sampling rate is relatively fast compared to the system dynamics.

For pedagogical reasons, the discussion in this appendix is based on a first-order system, as this produces simple analytical expressions.

AppendixA.1. Optimal Constant Input

Consider a first-order SISO system on the form:

$$\begin{aligned} \dot{x} &= Ax(t) + Bd(t) && \text{(A.1a)} \\ y(t) &= Cx(t) && \text{(A.1b)} \end{aligned}$$

Given a sampling time T and an initial state $x(0) = x_k$, the solution y_{k+1} of this system at time T is given by:

$$\begin{aligned} x_{k+1} &= e^{AT}x_k + B\int_0^T e^{A(T-t)}d(t)\,dt && \text{(A.2a)} \\ y_k &= Cx_k && \text{(A.2b)} \end{aligned}$$

From [15, eq. (4.17)], the system (A.1) discretized using ZOH for the disturbance, and with $\bar{d}_k$ as input (equivalent to inserting $d(t) = \bar{d}_k$, $0 \le t \le T$ in (A.2)) is given by:

$$\begin{aligned} x_{k+1}^Z &= e^{AT}x_k^Z + \frac{B}{A}(e^{AT} - 1)\bar{d}_k && \text{(A.3a)} \\ y_k^Z &= Cx_k^Z && \text{(A.3b)} \end{aligned}$$

The optimal $\bar{d}_k$ is the one that minimizes the difference between the outputs of the continuous- and discrete-time systems, and may thus be found by solving $y^Z_{k+1} = y_{k+1}$ for $\bar{d}_k$. From (A.2) and (A.3), and with $x^Z_k = x_k$, the optimal $\bar{d}_k$ is given by:

$$\bar{d}^*_k = \frac{\int_0^T e^{A(T-t)} d(t)\, dt}{\frac{1}{A}\left(e^{AT} - 1\right)} \tag{A.4}$$

Given that only the sampled measurements d_k are available, the continuous-time disturbance $d(t)$ is not known, and this equation cannot be solved exactly. Without any knowledge about the disturbance dynamics, the best possible approximation of $d(t)$ is presumably a linear interpolation between d_k and d_{k+1}:

$$d(t) = d_k + (d_{k+1} - d_k)\frac{t}{T}, \quad 0 \le t \le T \tag{A.5}$$

By solving the integral:

$$\begin{aligned} &\int_0^T e^{A(T-t)} \left(d_k + (d_{k+1} - d_k)\frac{t}{T} \right) dt \\ &= \frac{1}{A^2 T}\left((AT-1)e^{AT} + 1 \right) d_k + \left(e^{AT} - AT - 1\right) d_{k+1} \big) \end{aligned} \tag{A.6}$$

and inserting this into (A.4), the solution for $\bar{d}^*_k$ is given by:

$$\bar{d}^*_k = \frac{(AT-1)e^{AT}+1}{AT(e^{AT}-1)}\, d_k + \frac{e^{AT} - AT - 1}{AT(e^{AT}-1)}\, d_{k+1} \tag{A.7}$$

Under the assumption that the linear interpolation (A.5) is indeed the best possible approximation of $d(t)$, this is the optimal $\bar{d}_k$ for the first-order SISO system (A.1).

AppendixA.2. Comparing Method B and FOH

Since (A.7) is derived from inserting the linear interpolation (A.5), applying $\bar{d}^*_k$ from (A.7) as input to the system (A.3) discretized using ZOH will result in an exact match with the continuous system (A.1) with a piecewise linear input. Thus, using ZOH and substituting d_k with $\bar{d}^*_k$ is in fact equivalent to discretizing the system (A.1) using FOH. However, inserting $\bar{d}^*_k$ into the system (A.3) results in a system on the form:

$$x_{k+1} = A^Z x_k + B^Z d_k + D^Z d_{k+1} \tag{A.8a}$$
$$y_k = C x_k \tag{A.8b}$$

whereas the system obtained in Section 2.4 using FOH-discretization in Matlab was on the form:

$$x^F_{k+1} = A^F x^F_k + B^F d_k \tag{A.9a}$$
$$y^F_k = C^F x^F_k + D^F d_k \tag{A.9b}$$

But an equivalent system on the form (A.9) may in fact be derived from the system on the form (A.8) as follows.

Inserting $\bar{d}_k^*$ into the system (A.3) results in the following discrete-time system:

$$\begin{aligned} x_{k+1} &= e^{AT}x_k + \frac{B}{A^2T}\left((AT-1)e^{AT}+1\right)d_k + \frac{B}{A^2T}\left(e^{AT}-AT-1\right)d_{k+1} && \text{(A.10a)} \\ y_k &= Cx_k && \text{(A.10b)} \end{aligned}$$

This may be rewritten without the state x as follows:

$$y_{k+1} = e^{AT}y_k + \frac{BC}{A^2T}\left((AT-1)e^{AT}+1\right)d_k + \frac{BC}{A^2T}\left(e^{AT}-AT-1\right)d_{k+1} \qquad \text{(A.11)}$$

The $\mathcal{Z}$-transform of this system produces the discrete-time transfer function:

$$\frac{y}{d}(z) = \frac{BC}{A^2T}\,\frac{(e^{AT}-AT-1)z+(AT-1)e^{AT}+1}{z-e^{AT}} \qquad \text{(A.12)}$$

Using polynomial division, this becomes:

$$\frac{y}{d}(z) = \frac{BC}{A^2T}\,\frac{(e^{AT}-1)^2}{z-e^{AT}} + \frac{BC}{A^2T}(e^{AT}-AT-1) \qquad \text{(A.13)}$$

A new state variable x^F may now be defined from the first term of this transfer function:

$$x^F(z) = \frac{BC}{A^2T}\,\frac{(e^{AT}-1)^2}{z-e^{AT}}\,d(z) \qquad \text{(A.14)}$$

so that:

$$y(z) = x^F(z) + \frac{BC}{A^2T}(e^{AT}-AT-1)\,d(z) \qquad \text{(A.15)}$$

The $\mathcal{Z}^{-1}$-transform of (A.14) and (A.15) leads to the system:

$$\begin{aligned} x_{k+1}^F &= e^{AT}x_k^F + \frac{BC(e^{AT}-1)^2}{A^2T}\,d_k && \text{(A.16a)} \\ y_k &= x_k^F + \frac{BC}{A^2T}(e^{AT}-AT-1)\,d_k && \text{(A.16b)} \end{aligned}$$

which is on the form (A.9). That (A.16) is indeed identical to the system obtained using FOH-discretization in Matlab is easily confirmed for particular values of A and T e.g. using the Matlab function `tf`.

Note that the change of state variable from x to x^F implies that (3) no longer holds for the system (A.16), which may complicate the implementation as discussed in Section 2.2.

AppendixA.3. Approximation of the Optimal Constant Input

The proposed solution in this paper, denoted method B, is not to replace d_k with the optimal $\bar{d}_k^*$ from (A.7), but with $\bar{d}_k$ from (6), which is simply the average of d_k and d_{k+1}. It is shown in this section that this is in fact a good approximation of the optimal $\bar{d}_k^*$.

The optimal $\bar{d}_k^*$ depends on only two variables; the system parameter A and the sampling time T. By calculating the limit $\lim_{T\to 0}\bar{d}_k^*$, it can be shown that $\bar{d}_k^*$ actually converges to the average of d_k and d_{k+1} when $T \to 0$:

$$\begin{aligned}
\lim_{T\to 0} \bar{d}_k^* &= \lim_{T\to 0} \left(\frac{(AT-1)e^{AT}+1}{AT(e^{AT}-1)}\, d_k + \frac{e^{AT}-AT-1}{AT(e^{AT}-1)}\, d_{k+1} \right) \\
&= \lim_{T\to 0} \left(\frac{A^2Te^{AT}}{(A^2T+A)e^{AT}-A}\, d_k + \frac{A(e^{AT}-1)}{(A^2T+A)e^{AT}-A}\, d_{k+1} \right) \\
&= \lim_{T\to 0} \left(\frac{(A^3T+A^2)e^{AT}}{(A^3T+2A^2)e^{AT}}\, d_k + \frac{A^2e^{AT}}{(A^3T+2A^2)e^{AT}}\, d_{k+1} \right) \\
&= \frac{A^2e^0}{2A^2e^0}\, d_k + \frac{A^2e^0}{2A^2e^0}\, d_{k+1} \\
&= \frac{1}{2}(d_k + d_{k+1}) \qquad \text{(A.17)}
\end{aligned}$$

(This limit was calculated using L'Hôpital's rule twice.)

Now obviously, the sampling rate cannot be chosen infinitely fast, but a well-known rule of thumb is to choose a sampling rate that is 5-10 times faster than the fastest time constant in the system. Consider for example the following first-order SISO system with time constant τ:

$$\begin{aligned}
\tau\dot{x}(t) &= -x(t) + d(t) \qquad \text{(A.18a)} \\
y(t) &= x(t) \qquad \text{(A.18b)}
\end{aligned}$$

If the sampling rate is chosen to be 10 times faster than the time constant τ, the sampling time is given by $T = \tau/10$. Inserting this and $A = -1/\tau = -1/(10T)$ into (A.7), $\bar{d}_k^*$ for this system is then given by:

$$\begin{aligned}
\bar{d}_k^* &= \frac{\left(-\frac{1}{10}-1\right)e^{-\frac{1}{10}}+1}{-\frac{1}{10}\left(e^{-\frac{1}{10}}-1\right)}\, d_k + \frac{e^{-\frac{1}{10}}+\frac{1}{10}-1}{-\frac{1}{10}(e^{-\frac{1}{10}}-1)} d_{k+1} \qquad \text{(A.19)} \\
&\approx 0.4917\, d_k + 0.5083\, d_{k+1}
\end{aligned}$$

This shows that method B ($\bar{d}_k = 0.5\, d_k + 0.5\, d_{k+1}$) is a very good approximation of the optimal $\bar{d}_k^*$ (and thus FOH discretization) if the sampling rate is sufficiently fast compared to the system dynamics.

However, due to e.g. limited computational capacity in embedded implementations, or instruments with a slow measurement update rate, the sampling rate cannot always be chosen "sufficiently fast". Now consider that the sampling rate for the system (A.18) for some reason cannot be faster than the time constant τ, so that $T = \tau$, then $\bar{d}_k^*$ is given by:

$$\begin{aligned}
\bar{d}_k^* &= \frac{1-2e^{-1}}{1-e^{-1}}\, d_k + \frac{e^{-1}}{1-e^{-1}}\, d_{k+1} \qquad \text{(A.20)} \\
&\approx 0.4180\, d_k + 0.5820\, d_{k+1}
\end{aligned}$$

From this, it is clear that even with this slow sampling rate, method B ($\bar{d} = 0.5\, d_k + 0.5\, d_{k+1}$) is still a quite decent approximation of $\bar{d}_k^*$, and obviously a much better approximation than method A (which is equivalent to $\bar{d}_k = 1\, d_k + 0\, d_{k+1}$). But if the issue addressed in this paper is vital for the control performance, the performance might be improved slightly by using the optimal $\bar{d}_k^*$, compared to simply using the average from method B, if the sampling rate is relatively slow. While deriving the optimal $\bar{d}_k^*$ may seem complicated, once it is derived, it is just as easy to implement as method B, at least for a first-order SISO system as considered in this appendix.

For higher-order/MIMO systems, deriving and implementing an optimal $\bar{d}_k$ may not be quite as straightforward, as the optimal expression for $\bar{d}_k$ might be different for each disturbance/output pair. Given the simplicity of method B, and the improvement already achieved compared to method A, it is hard to imagine a scenario where the effort required to derive and implement the optimal $\bar{d}_k$ in a complex system may be justified by the marginal improvement that can be expected from doing so, compared to simply implementing method B.

---

[1] J. B. Rawlings, Tutorial overview of model predictive control, IEEE Control Systems Magazine 20 (3) (2000) 38–52.

[2] K. R. Muske, J. B. Rawlings, Model predictive control with linear models, AIChE Journal 39 (2) (1993) 262–287.

[3] A. Bemporad, Model predictive control design: New trends and tools, in: Proc. 45th IEEE Conference on Decision & Control (CDC 2006), San Diego, CA, USA, 2006.

[4] J. M. Maciejowski, Predictive Control: with constraints, Pearson and Prentice Hall, 2002.

[5] C. Bordons, J. R. Cueli, Predictive controller with estimation of measurable disturbances. application to an olive oil mill, Journal of Process Control 14 (3) (2004) 305–315.

[6] J. Richalet, A. Rault, J. L. Testud, J. Papon, Model predictive heuristic control: Applications to industrial processes, Automatica 14 (5) (1978) 413–428.

[7] C. R. Cutler, B. L. Ramaker, Dynamic Matrix Control–a computer control algorithm, in: AIChE meeting, Houston, TX, USA, 1979.

[8] G. Pannocchia, J. B. Rawlings, Disturbance models for offset-free model-predictive control, AIChE Journal 49 (2) (2003) 426–437.

[9] M. Morari, J. H. Lee, Model predictive control: past, present and future, Computers and Chemical Engineering 23 (4-5) (1999) 667–682.

[10] S. Strand, J. R. Sagli, MPC in Statoil – advantages with in-house technology, in: International Symposium on Advanced Control of Chemical Processes (ADCHEM), Hong Kong, 2003, pp. 97–103.

[11] K. R. Muske, T. A. Badgwell, Disturbance modeling for offset-free linear model predictive control, Journal of Process Control 12 (5) (2002) 617–632.

[12] G. Pannocchia, Offset-free tracking mpc: A tutorial review and comparison of different formulations, in: 2015 European Control Conference (ECC), Linz, Austria, 2015.

[13] U. Maeder, F. Borrelli, M. Morari, Linear offset-free Model Predictive Control, Automatica 45 (10) (2009) 2214–2222.

[14] P. Tatjewski, Disturbance modeling and state estimation for offset-free predictive control with state-space process models, Int. J. Appl. Math. Comput. Sci. 24 (2) (2014) 313–323.

[15] C.-T. Chen, Linear System Theory and Design, 3rd Edition, The Oxford series in electrical and computer engineering, Oxford University Press, 1999.

[16] Continuous-discrete conversion methods - matlab & simulink [cited 2018-02-25].
URL `www.mathworks.com/help/control/ug/continuous-discrete-conversion-methods.html`

[17] S. Li, K. Y. Lim, D. G. Fisher, A state space formulation for model predictive control, AIChE Journal 35 (2) (1989) 241–249.

[18] J. H. Lee, M. Morari, C. E. Garcia, State-space interpretation of model predictive control, Automatica 30 (4) (1994) 707–717.

[19] D. K. M. Kufoalor, L. Imsland, T. A. Johansen, High-performance embedded model predictive control using step response models, in: 16th IFAC workshop on Control Applications of Optimization, 2015.

[20] B. J. T. Binder, D. K. M. Kufoalor, A. Pavlov, T. A. Johansen, Embedded Model Predictive Control for an Electric Submersible Pump on a Programmable Logic Controller, in: 2014 IEEE Conference on Control Applications (CCA), 2014.

[21] A. Pavlov, D. Krishnamoorthy, K. Fjalestad, E. Aske, M. Fredriksen, Modelling and model predictive control of oil wells with electric submersible pumps, in: Proc. 2014 IEEE Conference on Control Applications (CCA), 2014.