Modeling, Identification and Control, Vol. 41, No. 3, 2020, pp. 223–239, ISSN 1890–1328

An analysis of Model Predictive Control with Integral Action applied to Digital Displacement Cylinders

Viktor Donkov[1] Torben Ole Andersen[1] Morten Kjeld Ebbesen[2]

[1]*Department of Energy Technology, Aalborg University, Pontoppidanstraede 111, 9220 Aalborg East, Denmark. E-mail:* vhd@et.aau.dk

[2]*Department of Engineering Sciences, University of Agder, Jon Lilletuns vei 9, 4879 Grimstad, Norway.*

Abstract

This article aims to analyze Model Predictive Control (MPC) for the control of multi-chamber cylinders. MPC with and without integral action has been introduced. Three different algorithms have been used to solve the optimization problem in the MPC. The different algorithms have been compared with an industrial solver. The influence of changing mass, choosing a different middle line pressure, system delays, signal noise, velocity estimation, and changing pressure levels has been investigated. It is concluded that for the small prediction horizon used in the paper a simple algorithm such as A^* can produce results as good as the previously used Differential Evolution algorithm in less than half the time. It is further concluded that unknown software delays and unknown changes in mass have the largest effect on system performance.

Keywords: Model Predictive Control, Digital Displacement Cylinders, Optimization

1 Introduction

Digital displacement cylinders are cylinders with more than two chambers. The differently sized areas allow the cylinder to change its force output, by connecting them to low or high pressure lines Linjama et al. (2009). Switching between these forces can allow the cylinder to follow a desired trajectory in an energy efficient way. The switching also causes undesirable vibrations in the cylinder's movement. According to Linjama et al. (2009) multi-chamber cylinders are only good for moving large masses, which help to filter out the vibrations. While this is true the minimum mass requirements have not clearly been investigated. The cylinders force resolution is defined by the number of chambers, their relative size, and the number of pressures, which it can be connected to. The authors have been using a middle pressure line to reduce the losses due to switching Hansen et al. (2011), Hansen et al. (2017). Usually this pressure is selected as the midpoint between the minimum and maximum pressures in the system. In the case of Hansen et al. (2011) a system with four pressure-rails is discussed but their magnitudes are still equally distributed. In Donkov et al. (2017) the value of the middle pressure rail is selected through a parameter sweep and it is shown that the value of the middle pressure value can vastly improve energy efficiency. A simulation of the entire trajectory run is used for each evaluation, which makes the method slow. Two algorithms are used in the control of multi-chamber cylinders - Force Selection Algorithm (FSA) used in Huova et al. (2010) and Hansen et al. (2011), and Model Predictive Control (MPC) used in Donkov et al. (2018), Hansen et al. (2018). In Heybroek and Sjöberg (2018) both algorithms are used in a sense as a FSA is used to generate a pressure reference and then a MPC is used to control the valves during the switching event. In Hansen et al. (2017) and Donkov et al. (2018) it was shown that MPC can perform better than FSA for position tracking problems, but in these papers and others the computational burden of the algorithm is mentioned as a significant issue. In Donkov et al. (2018) the MPC could not deal with the changing load. In Donkov et al. (2019) MPC with integral action was applied to deal with this, but since the focus in Donkov et al. (2019) was on fault tolerance the effects were not really properly investigated. Furthermore the delay compensation proposed in Cortes et al. (2012) was introduced to deal with the considerable computation delay and the long time between a given command to switch pressures and the actual event occurring. This was also not discussed. This paper will attempt to address some of these problems. First a system will be described in Sec. 2. This system will be used for all of the tests. Then the MPC used in Donkov et al. (2018) and the addition of integral action as in Donkov et al. (2019) will be presented in Sec. 3. Furthermore, the properties of the cost functions used in Donkov et al. (2018) and Donkov et al. (2019) will be analysed. Then three different optimization algorithms will be presented in Sec. 4. One of the algorithms will be chosen for further study of the system in Sec. 5, where the influence of changing mass, changes in middle pressure line, reference changes and system delay will be discussed.

2 Model

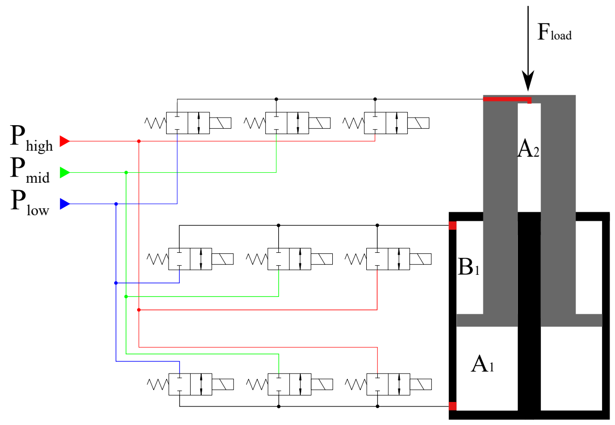

Figure 1: Representation of the multi-chamber cylinder used in the paper

The model used to investigate the system is a three chamber multi-chamber cylinder with a constant mass. The equations are as follows.

$$\dot{p}_i = \frac{\beta_i}{V_i}\left(A_i\dot{x}_p + Q_{v,i,n}(u)\right) \tag{1}$$

$$Q_{v,i,n}(u) = k_q u S(p_i - p_n)\sqrt{(|p_i - p_n|} \tag{2}$$

$$\ddot{x}_p = \frac{1}{m}\left(F_{cyl} - F_g - F_{fric}\right) \tag{3}$$

where $\dot{p}_i$, β_i, V_i, and A_i are the pressure gradient, bulk modulus, volume, and area of chamber i. x_p, $\dot{x}_p$, and $\ddot{x}_p$ are the position, velocity and the acceleration of the piston. $Q_{v,i,n}(u)$ is the flow to chamber i, through valve n. The flow is defined by the orifice equation with k_q being the valve specific coefficient, p_n being the n^{th} pressure line and u being the normalized valve opening. $S(*)$ stands for the sign function. m is the mass of the system, F_{cyl} is the force provided by the cylinder, F_g is the gravitational load, and F_{fric} is the frictional force modelled by the LuGre model. The system parameters are taken from Donkov et al. (2018) and Ho Cho et al. (2016), with a major difference that the cylinder is considered to push a constant mass instead of a changing inertia. This is done in order to simplify the analysis. The LuGre friction model is described by

$$F_{fric} = \sigma_0 z + \sigma_1 \dot{z} + \sigma_2 \dot{x} \tag{4}$$

$$\dot{z} = \dot{x} - \frac{|\dot{x}|}{g(\dot{x})} z \tag{5}$$

$$g(\dot{x}) = \frac{1}{\sigma_0}\left[F_c + (F_s - F_c)e^{-(\dot{x}/v_{str})}\right] \tag{6}$$

where z is the average deflection of the bristles, σ_0, σ_1, and σ_2 are friction parameters. $g(\dot{x})$ is a non-linear function describing the effects of the different friction forces, where F_c is the Coulomb friction, F_s is the static friction, and v_{str} is the Stribeck velocity. The parameters for this friction model have been obtained experimentaly for this specific cylinder by Ho et al. in Ho Cho et al. (2016). All parameters are collected in Tab. 1.

2.1 Force level number and density

The operation of the controller is heavily influenced by the possible force levels. The equation for the number of possible force levels is:

$$F_{num} = {n_c}^{n_p} \tag{7}$$

where F_{num} is the number of force levels that are available, n_c is the number of chambers and n_p is the number of pressure lines. Some researchers have chosen to use a four-chamber cylinder with two pressure lines - this gives 16 force levels. Others have chosen to use a three-chamber cylinder with three pressure lines - this gives 27 force levels. Others have chosen to use a normal differential cylinder with 7 pressure lines - this

Table 1: System parameters

A_1	$0.0051\ m^2$	F_s	$1214/\text{-}1646\ N$
A_2	$0.0026\ m^2$	F_c	$500/\text{-}600\ N$
A_3	$0.0013\ m^2$	v_{str}	$0.026/\text{-}0.035\ m/s$
k_q	$2.3570 \cdot 10^{-7}\ m^3/Pa$	σ_0	$8 \cdot 10^6/-6 \cdot 10^6\ N/m$
m	$50000\ kg$	σ_1	$7 \cdot 10^2/-7 \cdot 10^2\ N/ms^{-1}$
F_g	$9000\ N$	σ_2	$1 \cdot 10^4/-9 \cdot 10^3\ N/ms^{-1}$

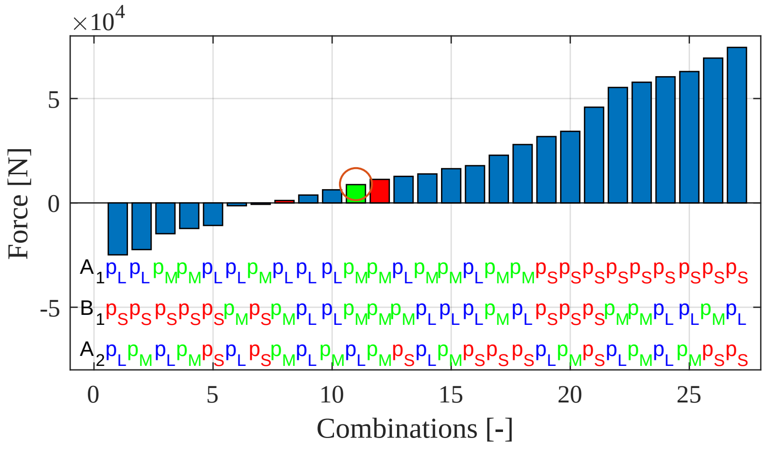

Figure 2: Force resolution of a three-chamber, three-pressure cylinder.

gives 128 force levels. The cost of switching between two pressures depends on the pressure and the volume as:

$$E = \frac{1}{2}\frac{V}{\beta}\left(p_1 - p_0\right)^2 \tag{8}$$

It stands to reason that having smaller differences between pressure levels improves efficiency more than having smaller chambers in which the switch can occur, because the pressure difference is squared. This would suggest that having more pressure lines is always better than having more chambers. An argument for increasing the number of chambers can be made, since all chambers are in use. On the contrary there might be 20 possible pressure lines, but a trajectory might require the use of only two of them. The number of force levels alone is not the only important thing. As mentioned above, it can be expected that trajectories will not utilise the entire force range of the cylinder. An example of this can be seen in Fig. 2. Here a constant force of 9000 N is needed (denoted with a red circle). Since it is very difficult to have a force which exactly matches the load the controller will have to switch constantly. In Fig. 2 the two forces with the smallest cost between them have been coloured in red. Changing pressure of the middle line value does not change the maximum and minimum forces of the cylinder, but it does change the distribution of possible forces. Here a mid pressure line of 30 bar has been chosen through a parameter sweep with a full model simulation for each point in the space (20 pressure values between 20 and 100 bar). A less time consuming method can be utilized to choose the value of the mid pressure line for a certain load by using Eq. (8). For each possible mid pressure value all combinations between a force above and a force below a target can be arranged. The cost of each combination can be found through Eq. (8) and the minimum can be selected. Fig. 3 shows the results for such an analysis when the target is the afore mentioned 9000 N. The analysis suggests that the mid pressure line should be even smaller than the previously selected 30 bar. This can be attributed to the fact that in the analysis only the average required force is considered. By doing the same sweep, but this time summing the cost for each force in the known force trajectory gives the result shown in Fig. 4. In the figure it can be seen that for this trajectory a mid pressure value of 30 bar gives the smallest amount of switching losses. This agrees with the results of the exhaustive search. The benefit is that the analysis can be conducted in a matter of seconds where as the parameter sweep can take up to several hours. The improved costs and force density around the load comes at a price of course - there is only one force level between 34000 N and 55000 N.

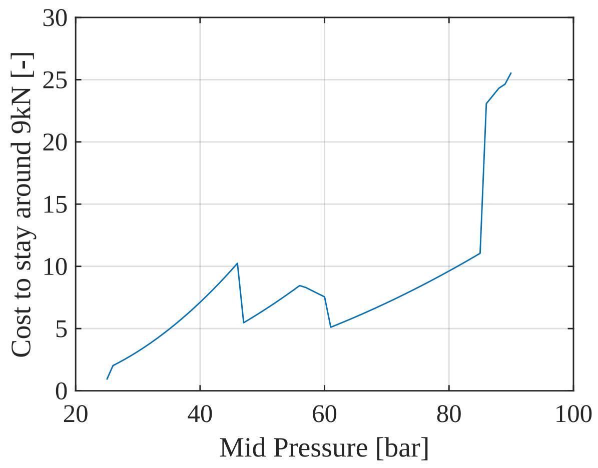

Figure 3: Cost to stay in one place with a load of 9 kN with different mid pressure levels

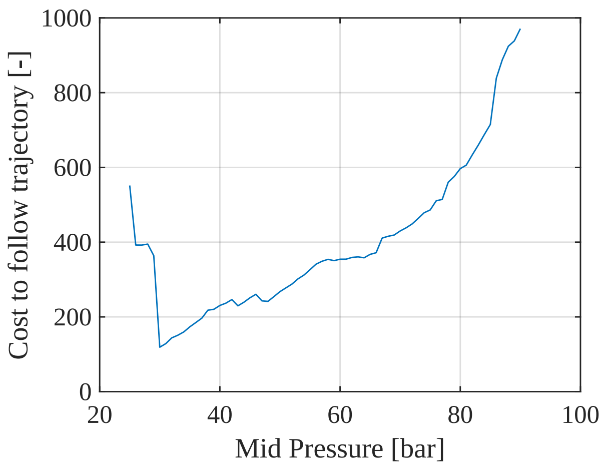

Figure 4: Cost to follow trajectory with different mid pressure levels

3 Control

The control structure will be a Model Predictive Controller (MPC). The controller has already been tested on a similar system in Donkov et al. (2018). Here it is repeated in brief terms for consistency in notation. The controller chooses a control signal based on an optimisation of a cost function J. The elements of the cost function are usually connected with the outputs of a physical representation of the system e.g a model. So in order to use MPC a model of the multi-chamber cylinder should be established. The one used in Donkov et al. (2018) was:

$$x(k+1) = Ax(k) + Bu(k) - A_{grav} \tag{9}$$
$$y(k+1) = Cx(k) \tag{10}$$

where $x(k)$ are the five internal states of the system at time step k which are p_A, p_B, p_C, v_p, x_p. These in turn are the three chamber pressures, the velocity of the piston, and the position of the piston respectively. $u(k)$ is the vector of valve openings. Once again from Donkov et al. (2018) A, B, and C can be defined as:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & Ts \\ \frac{A_1 T_s}{M_{eq}} & -\frac{B_1 T_s}{M_{eq}} & \frac{A_2 T_s}{M_{eq}} & 0 & 1 \end{bmatrix} \tag{11}$$

$$B = \begin{bmatrix} p_t & p_{mid} & p_{high} & 0\,0\,0 & 0\,0\,0 \\ 0\,0\,0 & p_t & p_{mid} & p_{high} & 0\,0\,0 \\ 0\,0\,0 & 0\,0\,0 & p_t & p_{mid} & p_{high} \\ 0 & & \cdots & & 0 \\ 0 & & \cdots & & 0 \end{bmatrix} \tag{12}$$

It can be noticed that no pressure dynamics are present in the model. The pressures in the system are directly defined by the valve vector and the matrix B. An equivalent mass M_{eq}, areas A_1, B_1, and A_2, and the sampling time T_s define the dynamics of the model. The magnitudes of these can be seen in Tab. 1, with the exception of T_s, which is 60 ms. Furthermore it was shown in Donkov et al. (2018), that the controller can work much better if the disturbance force is constant and known. In the referenced article the disturbance force was a load due to gravity, which is why it is denoted with A_{grav} in Eq. (9). The use of MPC with integral action can be a solution to problems with disturbances and modelling errors Stephens et al. (2013). According to Stephens et al. (2013) in order to introduce integral action the two things need to be introduced - the change in control $\Delta u(k) = u(k) - u(k-1)$ and change in system state $\Delta x(k) = x(k) - x(k-1)$. With these definitions Eq. (9) can be rewritten as

$$\Delta x(k+1) = A\Delta x_c(k) + B\Delta u(k) \tag{13}$$
$$y(k+1) - y(k) = CA\Delta x_c(k) + CB\Delta u(k) \tag{14}$$

Then a new state vector has to be introduced

$$\bar{x}(k) = \begin{bmatrix} \Delta x_c(k) \\ y(k) \end{bmatrix} \tag{15}$$

The system equations become

$$\bar{x}(k+1) = \bar{A}\bar{x}(k) + \bar{B}\Delta u(k) \tag{16}$$
$$y(k) = \bar{C}\bar{x}(k) \tag{17}$$

where

$$\bar{A} = \begin{bmatrix} A & 0^{n\times m} \\ CA & I^{m\times m} \end{bmatrix} \tag{18}$$

$$\bar{B} = \begin{bmatrix} B \\ CB \end{bmatrix} \tag{19}$$

$$\bar{C} = \begin{bmatrix} 0^{m\times n} & I^{m\times m} \end{bmatrix} \tag{20}$$

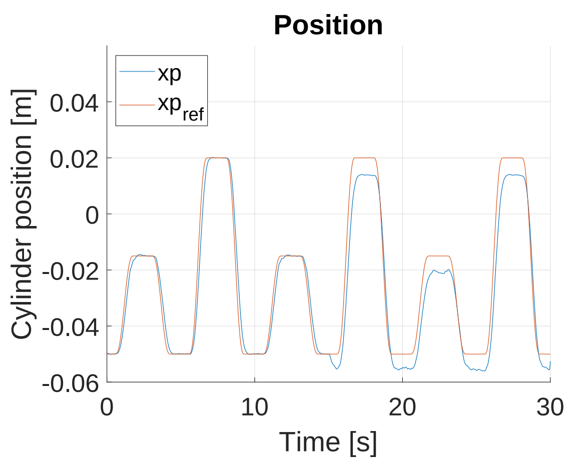

Figure 5: Position of cylinder without integral action MPC.

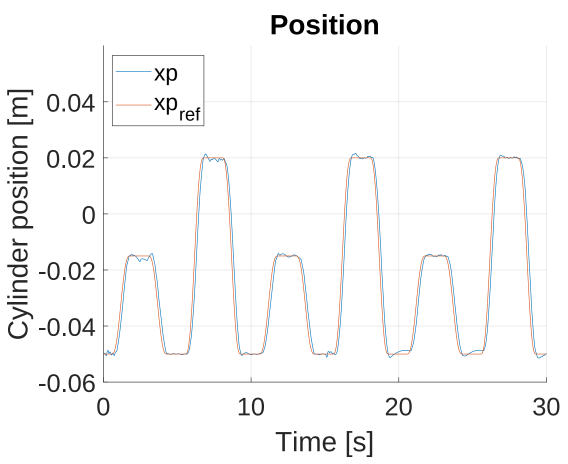

Figure 6: Position of cylinder with integral action MPC.

This can also be written as

$$\hat{y} = G\Delta\hat{u} + \hat{x}_o \tag{21}$$

where

$$G = \begin{bmatrix} \bar{B} & 0 & \cdots & 0 \\ \bar{A}\bar{B} & \bar{B} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{A}^{M-1}\bar{B} & \bar{A}^{M-2}\bar{B} & \cdots & \bar{B} \end{bmatrix} \tag{22}$$

$$\hat{x}_o = \begin{bmatrix} \bar{C}\bar{A} \\ \vdots \\ \bar{C}\bar{A}^{H_p} \end{bmatrix} \tag{23}$$

This is the model which will be used to predict the piston position of the multi-chamber cylinder for M steps ahead. The effect of a changing load on the original controller can be seen in Fig. 5. In this figure the load force on the cylinder is 0 N for the first 15 seconds and increases to 9000 N for the rest of the simulation. When the load increases an offset appears just as in Donkov et al. (2018). The effect of the integral action controller can be seen in Fig. 6.

3.1 Cost Function Analysis

The cost function used for the optimisation problem is

$$J = ||\hat{r} - (GQ\hat{u}_{full} + \hat{x}_o)||^2 + |F\hat{u}_{full}| \tag{24}$$

where $\hat{r}$ is the reference vector. The symbol $\hat{u}_{full}$ denotes the vector of valves which starts with the current valve combination followed by the vector to be found by optimisation. These variables can only take on values of 0 or 1. This denotes the valve being open or closed. The matrix Q is used to connect the vector $\hat{u}_{full} \in \{0, 1\}$ with the difference vector $\Delta\hat{u} \in \{-1, 0, 1\}$. One shows the actual valve opening, while the other shows the change in control action. By this definition of Q, $\Delta\hat{u}$ can be defined as $Q\hat{u}_{full} = \Delta\hat{u}$. In this case $(GQ\hat{u}_{full} + \hat{x}_o) = G\Delta\hat{u} + \hat{x}_o = \hat{y}$. The first part of the cost function is then $||\hat{r} - \hat{y}||^2$, which is the second norm squared of the position error of the cylinder. Throught this paper $||*||$ denotes the second norm of $*$ and $|*|$ denotes the first norm. The cost function can also be rewritten as:

$$J = ||-GQ\hat{u}_{full} + (\hat{r} - \hat{x}_o)||^2 + |F\hat{u}_{full}| \tag{25}$$

$$J = \left|\left|T\hat{u}_{full} + \hat{j}\right|\right|^2 + |F\hat{u}_{full}| \tag{26}$$

Putting the cost function on the form (26) makes it a well known optimization problem called classic lasso. F is a difference matrix which calculates the cost of switching from one pressure to another.

$$F(i,j) = \begin{cases} -V_{chamb_n} p_z, & if\ i = j \\ V_{chamb_n} p_z, & elseif\ j = i + 9 \\ 0, & otherwise \end{cases} \tag{27}$$

where V_{chamb_n} is the volume of the chamber connected to this valve, p_z is the normalized pressure in the pressure line connected to this valve.
In order for the optimization to agree with the model $\hat{u}$ has to be constrained to:

$$\hat{0} \leq \hat{u} \leq \hat{1} \tag{28}$$

where $\hat{0}$ and $\hat{1}$ are vectors of zeros and ones respectively, with the same size as $\hat{u}$. Furthermore, since only one pressure line should be connected to each chamber an

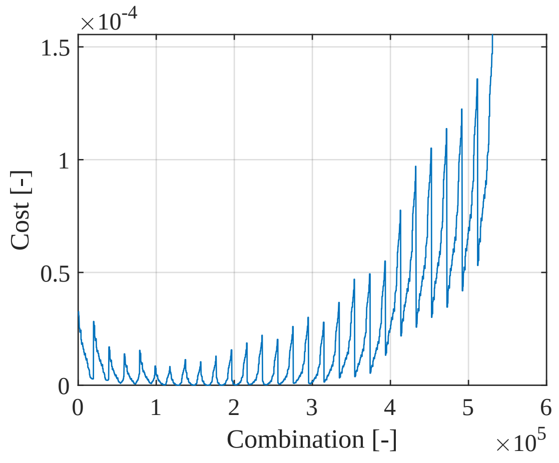

Figure 7: Cost of position accuracy.

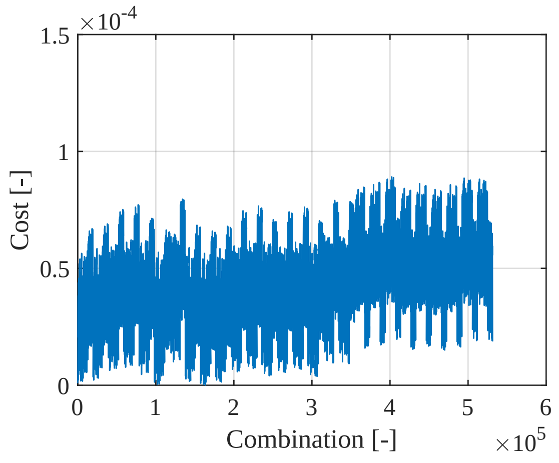

Figure 8: Cost of switching scaled with w.

additional constraint is added:

$$L\hat{u} = \begin{bmatrix} u_1(k) + u_2(k) + u_3(k) \\ \vdots \\ u_7(M) + u_8(M) + u_9(M) \end{bmatrix} \qquad L\hat{u} = \hat{1} \qquad (29)$$

Finally, because no throttling should occur with this controller, the input vector is constrained to only having the integer values $\hat{u} \in \{0, 1\}$.

The cost function is made of two parts - the cost of not following the reference, and the cost incurred due to switching between two pressure lines. The two costs for a prediction horizon of 4 steps can be seen in Fig. 7 and Fig. 8.

The switching cost function can be considered convex (if and only if the integrality constraint is dropped from $\hat{u}_{full}$), because the first one is constructed by taking an affine system $\hat{y}$ to the second power, and the second function is a norm, which makes it always affine.

In Fig. 9 the two parts of the cost function have been plotted against each other with blue points being specific combinations of valve openings. A Pareto optimal front can be identified by the points which are not dominated by any other points in at least one dimension. In this case the front has been denoted with red points. A closer view of the front is available in in Fig. 10.

The two cost functions can be added with the weighted sum method in order to find a preferred trade-off. This is illustrated in Fig. 11. In the figure the red star denotes the minimum, which is the the point with zero energy cost in Fig. 9.

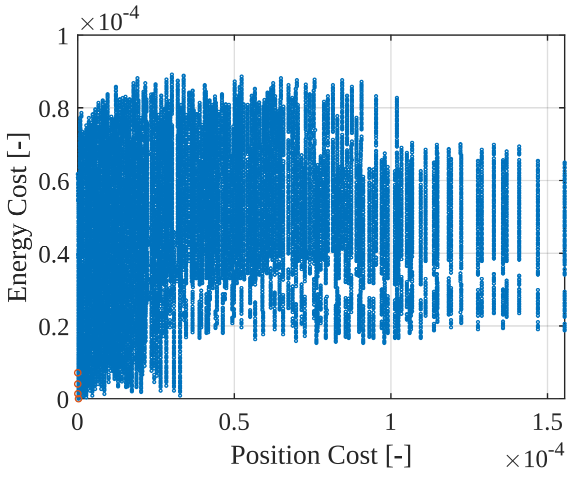

Figure 9: Pareto front.

4 Algorithms

Many different algorithms can be used to solve the problem defined in Eq. (26). Due to the integrality constraint on the inputs the problem is known as mixed-integer programming and is not convex. Some of the algorithms include stochastic optimization algorithms, branch and bound algorithms and enumeration. It is often difficult to determine which algorithm to use on a specific problem without testing it out. So far most DDC control papers have focused on stochastic algorithms (differential evolution in particular) and enumeration methods. In this section both differential evolution and branch and bound methods will be implemented and their performance will be investigated.

Differential Evolution

The differential evolution (DE) algorithm is a stochastic algorithm inspired by natural processes Storn (1995). It can solve a large variety of problems including the mixed-integer non-linear problems of the

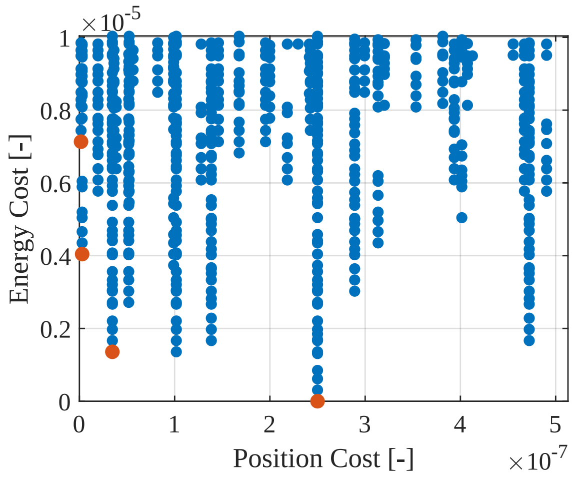

Figure 10: Zoomed in plot of the Pareto front.

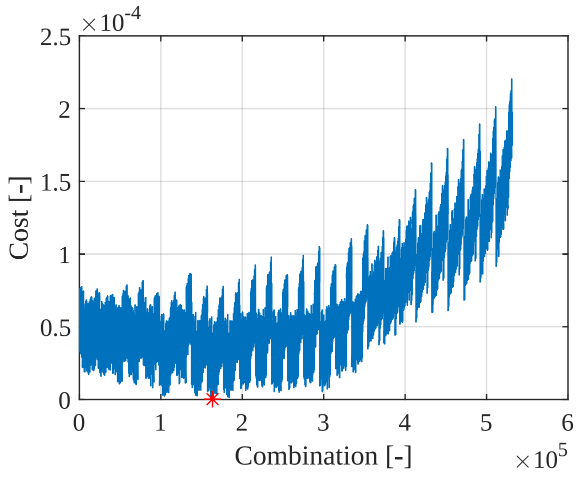

Figure 11: Combined cost for energy and position.

kind discussed in this paper. The algorithm has been proven to work for this specific controller in Hansen et al. (2017) and Donkov et al. (2019). The algorithm can be applied to the problem of controlling a DDC as described in Fig. 12. In it x stands for the current population, I_{ind} is a matrix of TRUE/FALSE values, rand is a random number generator outputting values between 0 and 1, and C_R is a number describing the crossover ratio. $\text{mod}(*, *)$ is the modulus function returning the remainder of the division of two numbers. Notation $x(min(f) == f)$ is taken from Matlab and stands for those members of x which produce the minimum value f. h is a vector holding the quality of the results of the last several generations. The size of the vector is defined by T_f. h_{indx} is the index in which the results of the current generation should be recorded in the vector h. The index cycles between 1 and T_f because of the modulus function. This cause the the vector h to act as a buffer getting continuously overwritten. The quality of the results of a generation is in this case the sum optimal values f for the entire population. If the algorithm is producing the same results multiple times in a row then it can be stopped. This is one of the stopping conditions for the algorithm, where the standard deviation of the history vector is compared with a tolerance T. In the algorithm a population of size $[NP, D]$ is initiated, where NP is the size of the population and D is the control horizon for the MPC. For each member in the population, a D-number of forces are selected based on a randomization of an initial seed. The seed can be a previous optimum or something else. In this implementation the seed was the force number 12 because that was a force level close to the constant load applied to the cylinder. It was found to be a good starting value. The randomization changed this force level within ± 4 forces i.e. the initial population had forces between force levels 8 and 16. The algorithm then uses the cost function J to find the fitness of each member. Each new generation is created based on the best members from the previous generations and a random mutation. The crossover ratio C_R determines how many members can be kept from the previous generation.

Figure 12: Differential Evolution (DE)

```
$x$ = Initiate(seed);
$G = 1$;
stop = 0;
$f = J(x)$;
**while** $stop == 0$ **do**
    $x_{new}$ = Mutate($x$, $min(f)$);
    $I_{ind}$ = rand > $C_R$;
    $x_{new}(I_{ind}) = x(I_{ind})$;
    $f$ = J($x_{new}$);
    $h_{indx}$ = mod($G$, $T_f$)+1;
    $h(h_{indx})$ = sum($f$);
    $G = G + 1$;
    $x_{best} = x(min(f) == f)$;
    $x = x_{new}$;
    **if** $std(h) < T$ $OR$ $G \geq G_{max}$ **then**
        stop = 1;
    **end**
**end**
```

The mutation was done according to:

$$x_{new} = x_{r_1} + F(x_{best} - x_{r_1}) + F(x_{r_2} - x_{r_3}) \qquad (30)$$

where r_1, r_2, and r_3 are randomly chosen vectors from the previous population, which are different from each other. The first part: $x_{r_1} + F(x_{best} - x_{r_1})$, moves the new population towards the best solution from the previous iteration, since if $F = 1$, the equation simplifies to x_{best}. The second part: $F(x_{r_2} - x_{r_3})$, prevents the algorithm from converging prematurely as it moves the answer in a random direction. As the algorithm converges, the population becomes more and more homogeneous and so the effect of the second part is reduced.

In theory the algorithm will converge to a global minimum provided the size of the population is large enough, the number of generations is large enough and the population is mutated in such a way as to explore the entire solution space. Generally speaking settings which allow the algorithm to converge faster are also more likely to result in local instead of global minima. The algorithm was implemented on a single-core 2 GHz dSpace microcontroller in Donkov et al. (2019). In that setup populations larger than 50 and with more than a 4 step prediction could not run in real time (<60 ms for the specific system). Even these numbers were not possible without first changing the cost function. Profiling the algorithm code showed the bottleneck in the evaluation of the cost function. Initially the cost function was selected for its convexity, but the DE method does not depend on convexity or the gradient of the cost function. The cost function J has a 2-norm squared. The squaring of matrices was identified as computationally intensive, so instead the sum($*$) and abs($*$) functions from Matlab were used:

$$f_x = \text{sum}(\text{abs}(\hat{e}_{np})) \tag{31}$$

The vector f_x has size $\Re^{NP,1}$ and represents the costs of following the reference of each individual member in the population of size NP. Similarly the cost of switching can be collected as:

$$f_E = \text{sum}(\text{abs}(Fx)) \tag{32}$$

where F is the difference matrix and x is the current population. The final cost vector is then:

$$f = f_x + f_E \tag{33}$$

It can be seen that for each individual member of the population the cost function was changed from Eq. (26) to

$$J = \left|T\hat{u}_{full} + \hat{j}\right| + |F\hat{u}_{full}| \tag{34}$$

The effects of this can be seen in Fig. 13. A red star denotes the minimum and it should be noticed that the same minimum as in Fig. 11 is found. This shows that changing the cost function has not changed the optimum. The algorithm was considerably sped up due to the reduction of mathematical operations. In fact the lines of code, which took the most time are the ones used to create the matrix x. Furthermore, the cost function can be sped up in Matlab by vectorizing

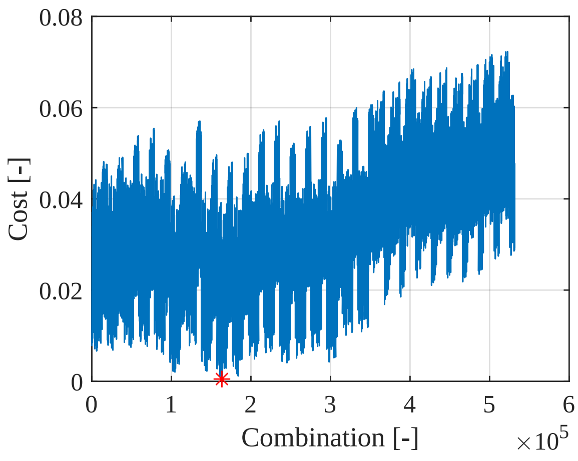

Figure 13: Position error and switching cost combined into one cost function.

the calculations. Instead of evaluating each member of the population in a FOR loop, the entire population can be evaluated at once by collecting all the control input vectors in a population of size NP to a matrix of size $\hat{u}_{pop} \in \Re^{D\times 9,NP}$.

$$\hat{u}_{pop} = [\hat{u}_{full,1}, \hat{u}_{full,2}, \cdots, \hat{u}_{full,np}] \tag{35}$$

$$\hat{j}_{np} = \left[\hat{j}, \hat{j}, \cdots, \hat{j}\right] \tag{36}$$

Then the entire population can be evaluated as

$$\hat{e}_{np} = T\hat{u}_{pop} + \hat{j}_{np} \tag{37}$$

After these changes the runtime of the code was reduced from $\approx$ 2 s to $\approx$ 30 ms. The small number of members in each population, the low number of populations and the changes to the cost function were expected to produce poor results, but in fact valve problems and measurement noise had a larger effect. For this article the function was written with both the flat cost function from Eq. 34 and the Lasso from Eq. 26 using Matlab 2019. The two functions were then compiled to MEX files with the Matlab coder application. Both were then tested on the same laptop. In the newer version of Matlab the two cost functions take the same time to complete. This can attribute to either improvements in Matlab's compiler or the difference in how the hardware handles the compiled code. The performance of the algorithm will be presented and compared after the other algorithms are also discussed. In those results DE refers to the Differential Evolution algorithm with the cost function Eq. (26), while DE-Flat refers to results with the cost function Eq. 34.

Branch and bound

A branch and bound algorithm was also developed. The algorithm consists of several steps as can be seen in Fig. 14.

Figure 14: Branch and bound based on Alternating Directions Method of Multipliers

```
while $stop == 0$ do
    $x_{opt}, f_{opt}$ = ADMM($J$, $A$) ;
    if $Binary(x_{opt})$ OR $I > I_{max}$ then
        stop = 1 ;
    end
    $x_b$ = BranchVar($x_{opt}$) ;
    $A_1$ = AddConstaintOne($x_b$) ;
    $x_1, f_1$ = ADMM($J$, $A_1$) ;
    $A_0$ = AddConstaintZero($x_b$) ;
    $x_0, f_0$ = ADMM($J$, $A_0$) ;
    if $f_0 > f_1$ then
        $A = A_1$ ;
    else
        $A = A_0$
    end
    $I = I + 1$ ;
end
```

The algorithm removes the integrality constraint and solves the resulting convex problem using a normal convex solver. In this case the convex solver used is the Alternating Directions Method of Multipliers (ADMM) algorithm. The solver finds a convex solution x_{opt} with a cost of f_{opt} according to the cost function J and the constraint matrix A. Then a variable x_b is chosen on which to branch by the function BranchVar(). The branching function evaluates three values of x_{opt} at a time. According to Eq. 29 the sum of these should be equal to one. If all three values are integers the function continues to the next three. If they are not, the function chooses the largest value to be branched on. In this case the two branches are - "the variable is constrained to be 0" and "the variable is constrained to be 1". This is done by adding the appropriate row to matrix A, which creates the matrices A_0 and A_1 respectively. The two minimums f_0 and f_1 are then compared and the smaller one is selected, provided all the constraints are satisfied. If f_1 is smaller and the value is constrained to be one, then the other two values have to be equal to zero for the original constraints to be satisfied. Because there is a switching cost associated with changing a variable, constraining one variable can affect the other values in the vector x_{opt}, resulting in faster convergence to an integer solution. If at this point all the variables are integers the algorithm is stopped. If not, then another variable to branch on is selected. In practice it was found that f_0 is more often smaller, so the same three variables have to be evaluated again.

The ADMM algorithm was chosen because it shows good properties for solving the lasso problem Boyd et al. (2011), Gaines et al. (2018). The method consists of separating the cost function into two separate functions

$$f(x) = 0.5\left|\left|Tx + \hat{j}\right|\right|^2 + |Fx| \tag{38}$$

$$g(z) = I(z) \tag{39}$$

where $f(x)$ is the original cost function and $g(z)$ is an indication function connected with the constraints as

$$I(z) = \begin{cases} 0 & x \in C \\ \text{inf} & \text{otherwise} \end{cases} \tag{40}$$

In this case C is the set $Ax = b$ satisfying the constraints Eq. (29) and $x \geq 0$. Then the problem can be described as:

$$\min(f(x) + g(z)) \tag{41}$$

$$\textit{subject to} \;\; x - z = 0 \tag{42}$$

The ADMM iterations to solve this problem are:

$$x^{k+1} = \min(f(x) + 0.5\rho\left|\left|x^{k+1} - z^k + u^k\right|\right|^2) \tag{43}$$

$$z^{k+1} = \min(g(z) + 0.5\rho\left|\left|x^{k+1} - z^{k+1} + u^k\right|\right|^2) \tag{44}$$

$$u^{k+1} = u^k + x^{k+1} + z^{k+1} \tag{45}$$

For this type of problem the proximity functions of the x and z iteration are known Gaines et al. (2018).

A* search

Another algorithm which will be tested is the A* search algorithm. The algorithm can be seen in Fig. 15. At each step, the algorithm chooses the node n_{branch} with the smallest cost n_{cost} and explores its branches n_{new} using the cost function J. The costs and names of the new branches are added to the list of branches N with the function AddNodes(). Then the already explored branch is removed from the list with the function RemoveNode(). Each branch deeper represents one more simulation step. For this reason the algorithm is searching for a node number above 20440 which represents nodes in the 5th simulation step. For the specific case here each node branches into 27 possible new combinations. The algorithm stops when the goal is reached and the cost to reach it is smaller than the cost to reach any of the unexplored nodes. Longer time horizons represent bigger vectors. Summing bigger vectors can result in larger results, which causes the algorithm to choose to branches with smaller numbers

Figure 15: A* search

```
$N$ = [1, 0, 0] ;
stop = 0 ;
while $stop$ ==$0$ do
    $n_{branch}, n_{cost} = \min(N(:,2))$;
    $n_{new} = \text{Branch}(n_{branch})$;
    $n_{new,costs} = J(n_{new})$;
    $f = \text{J}(x_new)$;
    $N = \text{AddNodes}([\ n_{new},\ n_{new,costs},$
     $n_{branch}])$ ;
    $N = \text{RemoveNode}(n_{branch})$ ;
    if  $n_{new} > Goal\ AND\ n_{new,costs} < N(:,2)$
     then
        stop = 1 ;
    end
end
```

i.e. branches earlier in the tree. This type of search is called a width first search and can result in a large number of explored branches. An example of this can be seen in Fig. 16. In order to help the algorithm to converge faster an heuristic is implemented. The heuristic is based on the distance of a certain combination to an oracle. In order to obtain an oracle the integrality constraint is dropped and the much simpler problem is solved using ADMM. Unfortunately the heuristic only works if the ADMM produces results above 0.5 for a specific valve which can then be rounded to 1. An example where the heuristic is rounded and where it is not can be seen in Fig. 18 and Fig. 17 respectively. Without a heuristic 229 nodes need to be explored, while with one the number drops to 84. It can be observed that at each following time step the number of summed elements is increased. This can lead to a situation in which every node on a level is explored before the next level is explored. The heuristic and the tuning need to take this into account. It can also be observed that during the first steps the chosen pressure levels do not affect the position error directly. This leads to a situation where the cost of changing force has a larger effect on the cost function.

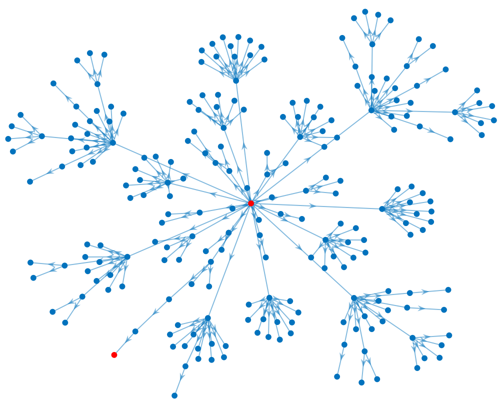
Figure 16: Tree exploration without heuristic. Explored nodes 229.

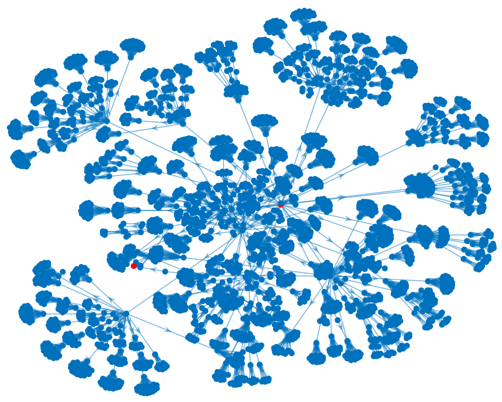
Figure 17: Tree exploration with heuristic but no rounding. Explored nodes = 3423.

The algorithm was still slow to complete in this form so a modification was introduced. Instead of predicting only until the current step in each node, the prediction for the entire horizon with the current node's force level is conducted. For example for node 10, which corresponds to the choice of force 9 at the first time step, the prediction is for the full 5 time steps with force 9 being kept constant. This prevents the optimization algorithm to find combinations in which a more costly step is taken first in order to use a much cheaper step later in the prediction horizon. This greatly increased performance time-wise, while not considerably degrading the accuracy of the controller, due to the nature of the system.

Algorithm results

The performance of the various algorithms has been tested and presented in Fig. 20 and Fig. 19. Two algorithms are added for comparison. The results denoted with Mosek are obtained using a commercial solver of the same name. Mosek can solve a variety of problems including mixed interger problems. The Yalmip toolbox for optimization in Matlab was used to interact with the solver Löfberg (2004). The algorithm is

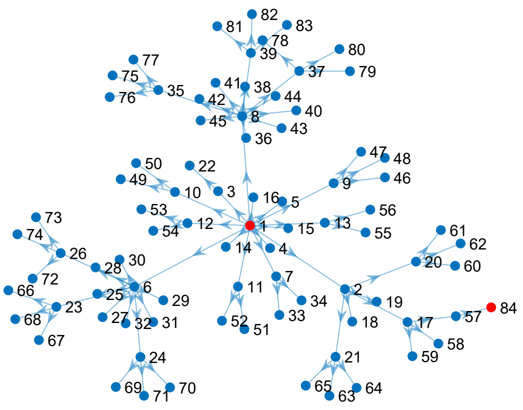

Figure 18: Tree exploration with heuristic with rounding. Explored nodes = 84.

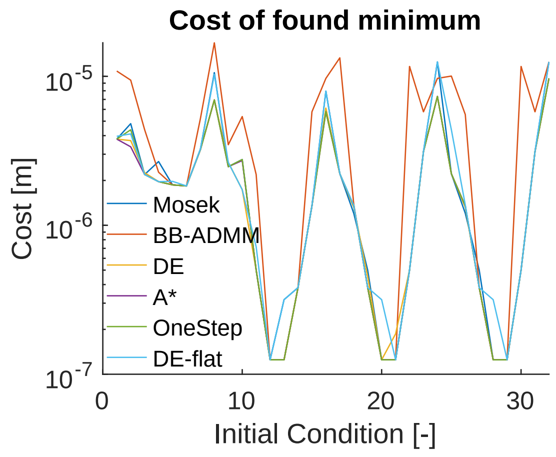

Figure 19: The value of the minimums found by the algorithms

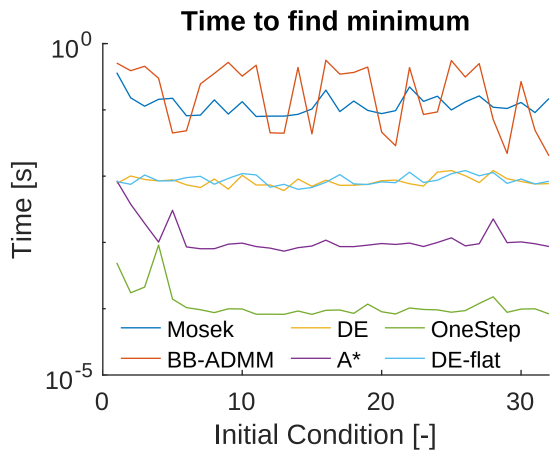

Figure 20: Time needed to find the minimum with different algorithms

included as a baseline. The second algorithm is called OneStep. It takes the first step of the A* algorithm and stops. The algorithm is included because it was noticed that the A* algorithm takes very few iterations to converge, which could mean that it converges prematurely. The initial conditions for the tests consist of 32 different initial conditions. These are created from all the combinations of two parameters. One parameter is the current force level. The second parameter is the position reference. This corresponds to a change in the position reference for the final step from -5 mm to +5 mm. In the table the notation (1,1) then corresponds to a situation where all the valves are closed and the desired movement is in the negative direction.

The different algorithms find similar optimum values in most cases. The only exception is the branch and bound algorithm which in most cases gives a poorer result. In order to show how these small variations affect the accuracy of the controller full simulation runs are conducted with each algorithm. Each test is repeated ten times and the values in Tab. 2 are the average and the standard deviation. Instead of RMS sum of position error, only the absolute sum of errors is used for the Accuracy measure. This is done, because when these numbers are divided by the large number of sample the variations become very small numbers which are difficult to compare. All the tests presented here have the same number of samples.

It can be seen that the branch and bound algorithm has the worst results in terms of accuracy, energy use and simulation time. It is also important to notice that the A* and OneStep algorithm have the same accuracy and energy use. This shows that in its current implementation the A* cannot find the global optimum. On the other hand it can also be noticed that the accuracy of the controller compared with the DE controller is not much different. Furthermore the OneStep algorithm completes the computation in ≈ 1/3 of the time. It can also be noticed that while DE and DE-Flat have variations in their results, the other algorithms do not. This is, because these two algorithms are stochastic based. The variation of the DE-Flat algorithm is quite large. In one of the ten tests the algorithm fails to find an optimum and gives an incorrect force command. The cost to switching back to a better force level is quite high and by the time the cylinder is following the trajectory again the results of that run are poor. This illustrates the issue with using stochastic algorithms to solve time sensitive problems.

Table 2: Result for full simulation

	BB-ADMM	DE	DE-Flat	A*	OneStep
Accuracy [m]	1260.99±0.00	843.90±8.29	916.30±119.49	859.62±0.00	859.62±0.00
Energy use [J/mm]	21.15±0.00	5.03±0.69	4.48±1.04	7.44±0.00	7.44±0.00
Sim Time [s]	197.50±12.85	131.01±17.49	121.75±7.43	64.87±34.84	41.50±8.19

5 Parameter variation

The effect of different algorithms has been explored in the previous section. In this section one of the algorithms is chosen and the effects of the variation of different system parameters will be examined. The performance of the algorithm under varying conditions will be examined according to the root-mean-square sum of error over time, the energy used over time and the total harmonic distortion. All test results are normalized with a so called base run where mass is 50 ton, software delays are 30 ms, no position or velocity noise is present, the pressure lines are constant and all signal are available.

5.1 Total Harmonic Distortion analysis and reference variation

In order to analyse the effect of parameter variation it was decided that the sum of error and the energy use of the simulation might not be enough. The Total Harmonic Distortion (THD) analysis is based on Fast Fourier Transforms (FFT) and can show the ratio between the main amplitude present in a signal and other frequencies such as harmonics or noise. Compared with the RMS sum of error the analysis shows whether position error comes from switching activity or if it comes from a phase shift between the reference and the actual position. In order to calculate the THD, the FFT of the position, velocity, and force output signals of the cylinder were collected. FFT produces a single-sided amplitude spectrum of the analysed signal. For instance the position signal was analysed and the results can be seen in Fig. 21. The FFT has broken down the original position signal into a large number of smaller composite sine-wave signals. In the figure the y-axis shows the amplitude of these components and the x-axis shows their frequency. The following formula was used in order to calculate THD based on the signal P1:

$$THD = \frac{\sqrt{(\sum_{f=1}^{L} P_{pos}(f)^2) - max(P_{pos}(f))^2}}{A_{ref,rms}} \tag{46}$$

In the equation L is the length of the amplitude spectrum signal. This is determined by the sampling frequency. Since the signal was obtained from a simulation a very high sampling frequency of 10000 Hz could

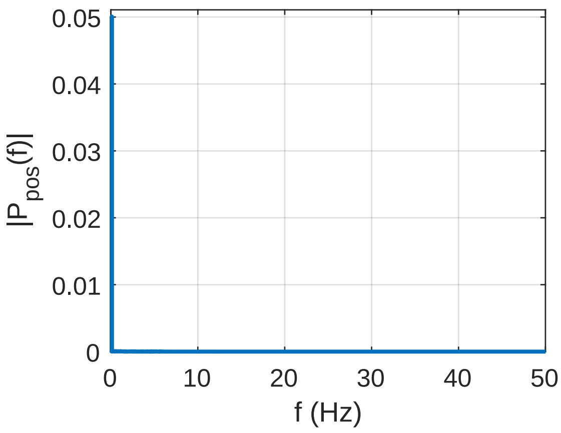

Figure 21: The single-sided amplitude spectrum of the position signal.

be chosen. $P_{pos}(f)$ is the value of the signal at frequency f i.e. the amplitude of the sine-wave with this frequency. In the figure it can be seen that the reference trajectory with a frequency of 0.2 Hz and an amplitude of 0.05 m is clearly seen. $max(P_{pos}(f))$ is the maximum value of the signal. In this case it will correspond to the previously mentioned spike due to the reference. It is important to notice that the spike does not necessarily equal the reference, as the controller might over or undershoot. $A_{ref,rms}$ is the RMS value of the reference signal. The noise due to switching is difficult to see due to the scale of the plot in Fig. 21. For this reason a zoomed in version is provided in Fig. 22. In Tab. 3 the results of five test are shown. The position reference is a sine wave with the indicated frequency. The length of the trajectory is always the same - 3 periods of the sine wave. THD results closer to 0 are better. It can be seen that position error, energy use and THD do not vary the same way. The trajectory with a frequency of 0.2 Hz has two times larger error, but uses three times less energy. At the same time the smoothness of motion has not degraded considerably as is indicated by the THD. It can be concluded that the error is largely due to phase shift between position and trajectory. It can be seen that at faster frequencies the controller is no longer able to follow the trajectory with the same accuracy until at 0.8 Hz the error and

Table 3: Sine wave testing

Sine freq [Hz]	Error sum [m]	Energy measure[J/mm]	THD [%]
0.1	0.0024	6.36	1.25
0.2	0.0047	2.66	1.79
0.4	0.0102	4.40	1.43
0.6	0.0171	8.02	4.99
0.8	0.0356	8.36	48.04

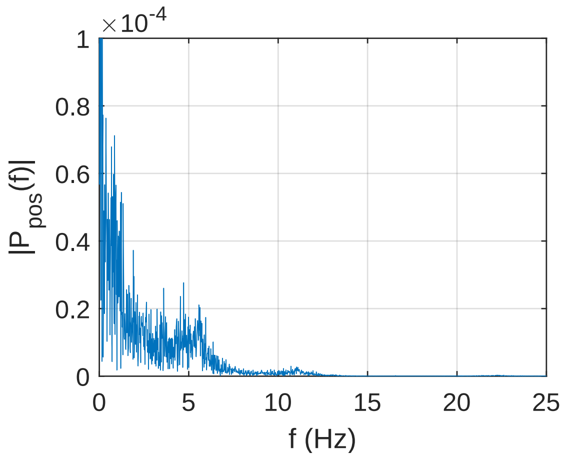

Figure 22: A zoomed in plot of the single-sided amplitude spectrum of the position signal.

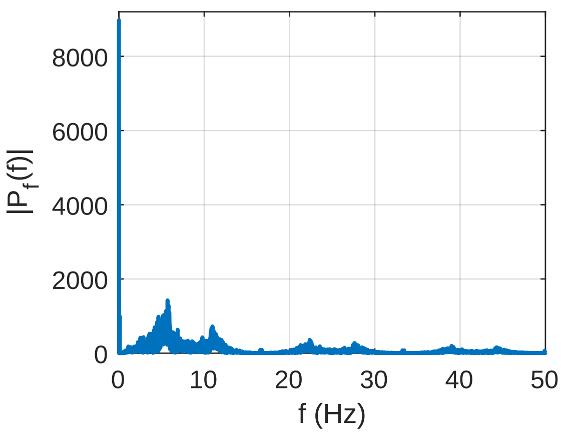

Figure 23: The single-sided amplitude spectrum of the force output signal.

THD become extremely large.

A standard cylinder with a proportional control valve, which can employ throttling control would produce THD values very close to zero. Outside of simulation studies it can be expected that the multi-chamber cylinder would produce results with higher THD. The same THD analysis can be applied to the force output of the cylinder and the results can be seen in Fig. 23. The constant load can be seen as a spike at 0 Hz. The result of the force switching can be seen as two mountains on either side of the 8 Hz mark, which repeat periodically. This plot can give an indication of the force spikes in the cylinder. Comparing the results in Fig. 22 and Fig. 23 shows that distortions at 5.5 Hz are visible in both plots, but in the position signal more noise is present in the frequency below 5.5 Hz. The higher frequency distortions have been filtered out.

The THD value is well established in electrical and audio engineering and multiple standards define acceptable limits. The field of digital hydraulics does not have such standards, but perhaps as the field expands it can establish them in conjunction with industry.

5.2 Changing mass

In order to test how the controller reacts to a change in load mass, this parameter has been varied from 25 ton to 100 ton. In Fig. 24 the mass is varied but the controller uses only 50 ton in the model of the system. It can be seen that reducing the mass below 40 ton increases error significantly. It can be seen that THD increases much faster. At 25 ton the RMS sum of position error is 5 times larger compared with the base run. In Fig. 25 when the mass is changed the new value is given to the controller and the prediction matrices are recalculated. This reduces both error and THD.

When inertia is reduced the system's frequency increases. In order to investigate if a multi-chamber cylinder can operate with low mass - the mass and switching frequency of the controller have been swept individually. The accurate values are supplied to the controller, so the model agrees with the parameters. The sweep has been extended in one direction until the controller is no longer accurate due to the large mass. With different tuning(increased penalty on position error) this error can be reduced. Increasing the switching frequency helps with this problem, but very quickly a lower bound is found. One valve requires 15

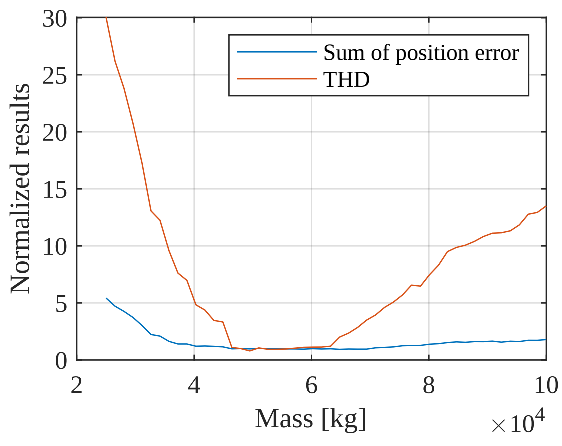

Figure 24: Result from varying mass without providing the controller the correct value.

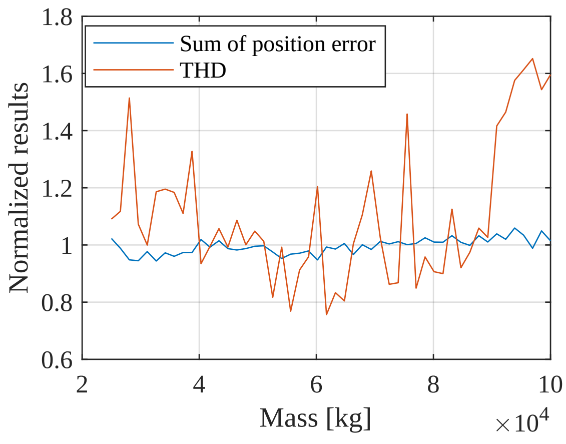

Figure 25: Result from varying mass and providing the controller the correct value.

ms to close and then the other requires 15 ms to open. It is assumed that only 5 ms are actual spool travel time while the rest is software delay and force build up. Using the transition can be done in 20 ms. After the transition is done it takes between 18 and 20 ms for the pressure in the chamber to change. According to this the controller cannot run faster than 40 ms. In this case a time step of 60 ms was used to ensure that the transition is complete. Changing tuning parameters does not help with the instability due to very low system inertia. It was found that with this size of cylinder and valves with these properties the system needs at least 15 ton of equivalent mass for the controller to be stable.

5.3 System delays

The system delays have a large effect on the performance of the controller. The delays are of a significant size compared with the sampling time of the model. In order to deal with this, the output of the controller is further delayed as discussed previously. If the added delay together with the system delays perfectly matches one sample of the model, it can be said that the delay can be cancelled out. This is of course never the case, since the delay compensation is static and user chosen and the real delays are variable and potentially unknown. In this study the representation of the system delay is varied, while the user chosen delay is kept constant at 30 ms. The results in Fig. 26 illustrate how much does the performance degrade depending on difference in delay.

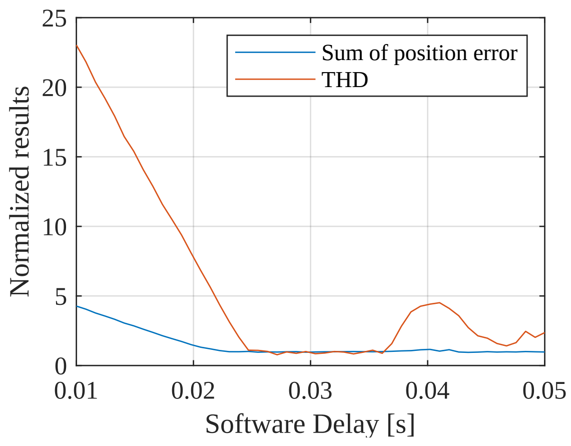

Figure 26: Result from varying system delays.

When the user selected delay is 30 ms the algorithm gives the best results. Between 25 ms and 35 ms there is no noticeable effect. Above 35 ms position error is still not affected, but THD increases. Between 10 ms and 25 ms position error and THD grow. THD is affected worse than the position error. Reducing the user selected delay to zero is equivalent to removing the delay compensation completely, which is why the results for 10 ms are the worst.

5.4 Noise and velocity estimation

In this study the difference in performance when noise is present on the position measurement and velocity measurement are presented. The level of the white noise is varied with a gain. A low pass filter with a cut-off frequency of 160 Hz is used to filter out the noise, because it can be expected that both noise and filtering would be present on any implementation of the system. Because of the gain the maximum noise level varies from 0 mm in the first test to 1.25 mm in the last. As a reference the real system has a noise level of 0.3 mm. The effect of noise on the position measurement can be seen in Fig. 27. The effect of noise on the velocity measurement can be seen in Fig. 28. Finally, it cannot be certain that a velocity measurement is available on all systems. A test was conducted were the velocity is estimated from the position measurement using the same method as in Donkov et al. (2019). The frequency of this estimation process was varied by changing the sampling time t_s and the results can be seen in Fig. 29.

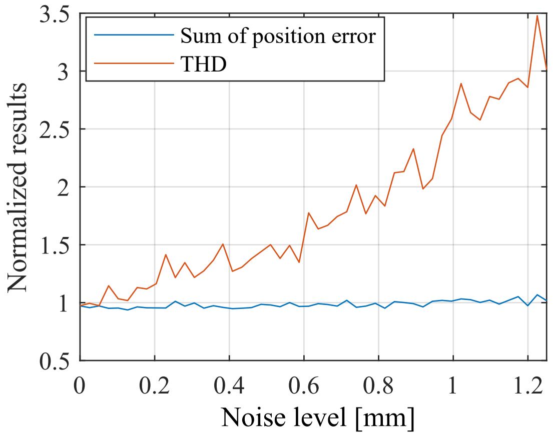

Figure 27: Result from varying position noise level.

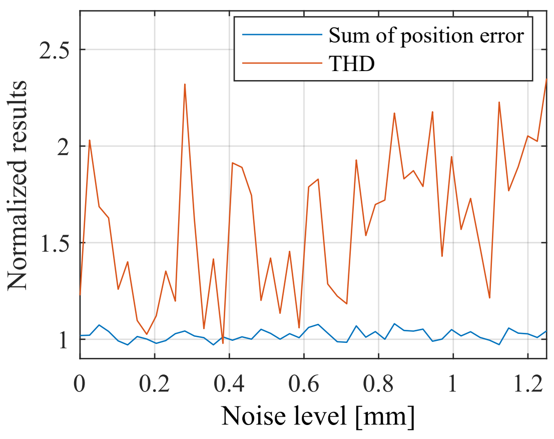

Figure 28: Result from varying velocity noise level.

These tests show that noise has little effect on the position error, but it increases THD. It can also be seen that at 0 mm velocity noise THD has increased by 20 % compared with the base run. The only difference between the two is that in this test the velocity signal is filtered even though the noise gain is zero. The same

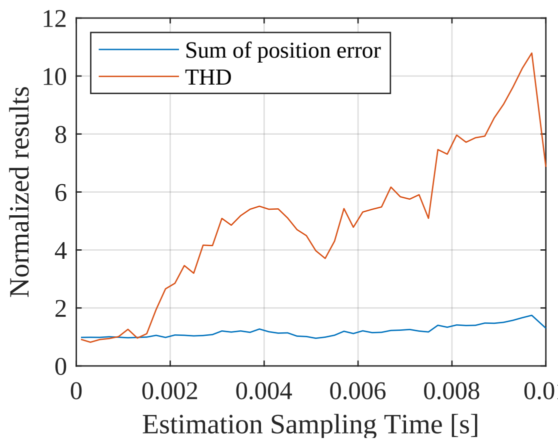

Figure 29: Result from varying the frequency of velocity estimation.

is true for the position noise test, but the THD there has not increased. It can be concluded that the phase shift introduced by the filter has increased THD, but the controller is only sensitive to phase shifts in velocity and not position. The results also show that if velocity is estimated with a high enough frequency it does not effect performance. In Donkov et al. (2019) the sampling rate of the velocity estimation was 5 ms. It can be concluded that this contributed to the poor laboratory results in that study.

5.5 Supply pressure change

It can be expected that the pressure of the supply lines will not be perfectly constant. In this test the value of the middle pressure line is changed, but the controller is not updated. The results can be seen in Fig. 30.

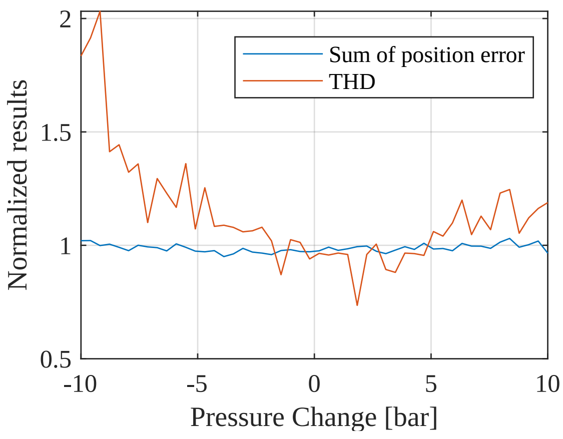

Figure 30: Result from varying the

In this test also the position error is not significantly changed by the variation of the parameter, but the THD raises to twice its original value when the pressure line magnitude is reduced by nine bar.

6 Conclusion

In this paper, a multi-chamber cylinder with MPC with integral action has been investigated through a sensitivity study. Several different optimization algorithms have been tested for the specific problem. The DE algorithm produces the best results regarding time and accuracy at larger time horizons, but at smaller time horizons of 4 steps a simpler iterative algorithm or the A* search can find similar results faster. Another benefit of the A* algorithm is that it is not based on stochastic methods, so it delivers the same performance every time. The systems performance depends on several factors. It can be seen that the integral action of the MPC can overcome some parameter variations, but this usually comes at the cost of increased control effort. The factors, which have the largest impact on the system performance are changes in mass, changes in system delays and the frequency of velocity estimation. If the exact mass is known and provided to the controller, multi-chamber cylinders can work with masses as low as 15 ton. In order to drive systems with lower inertia - smaller cylinders, faster valves and faster controllers need to be used. The THD number can give a good indication of the vibrations the controller will introduce due to switching, but the number is meaningless without some well established standards.

Acknowledgments

This research was funded by the Research Council of Norway, SFI Offshore Mechatronics, project number 237896/O30.

References

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 2011. 3(1):1–122. doi:10.1561/2200000016.

Cortes, P., Rodriguez, J., Silva, C., and Flores, A. Delay compensation in model predictive current control of a three-phase inverter. *IEEE Transactions on Industrial Electronics*, 2012. 59(2):1323–1325. doi:10.1109/TIE.2011.2157284.

Donkov, V., Andersen, T., Ebbesen, M. K., Linjama, M., and Paloniitty, M. Investigation of the fault tolerance of digital hydraulic cylinders. In *The 16th Scandinavian International Conference on Fluid Power, SICFP*. 2019.

Donkov, V., Andersen, T. O., Ebbesen, M. K., and Pedersen, H. C. Applying digital hydraulic technology on a knuckle boom crane. In *The Ninth Workshop on Digital Fluid Power*. 2017.

Donkov, V. H., Andersen, T. O., Pedersen, H. C., and Ebbesen, M. K. Application of model predictive control in discrete displacement cylinders to drive a knuckle boom crane. In *2018 Global Fluid Power Society PhD Symposium (GFPS)*. IEEE, pages 408–413, 2018. doi:10.1109/GFPS.2018.8472363.

Gaines, B. R., Kim, J., and Zhou, H. Algorithms for fitting the constrained lasso. *Journal of Computational and Graphical Statistics*, 2018. 27(4):861–871. doi:10.1080/10618600.2018.1473777.

Hansen, A. H., Asmussen, M. F., and Bech, M. M. Energy optimal tracking control with discrete fluid power systems using model predictive control. In *Proceedings of the Ninth Workshop on Digital Fluid Power, Aalborg, Denmark*. pages 7–8, 2017.

Hansen, A. H., Asmussen, M. F., and Bech, M. M. Model predictive control of a wave energy converter with discrete fluid power power take-off system. *Energies*, 2018. 11(3):635. doi:10.3390/en11030635.

Hansen, R. H., Andersen, T. O., and Perdersen, H. C. Analysis of discrete pressure level systems for wave energy converters. In *Fluid Power and Mechatronics (FPM), 2011 International Conference on*. IEEE, pages 552–558, 2011. doi:10.1109/FPM.2011.6045825.

Heybroek, K. and Sjöberg, J. Model predictive control of a hydraulic multichamber actuator: A feasibility study. *IEEE/ASME Transactions on Mechatronics*, 2018. 23(3):1393–1403. doi:10.1109/TMECH.2018.2823695.

Ho Cho, S., Niemi-Pynttäri, O., and Linjama, M. Friction characteristics of a multi-chamber cylinder for digital hydraulics. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 2016. 230(5):685–698. doi:10.1177/0954406215575414.

Huova, M., Laamanen, A., and Linjama, M. Energy efficiency of three-chamber cylinder with digital valve system. *International Journal of Fluid Power*, 2010. 11(3):15–22. doi:10.1080/14399776.2010.10781011.

Linjama, M., Vihtanen, H., Sipola, A., and Vilenius, M. Secondary controlled multi-chamber hydraulic cylinder. In *The 11th Scandinavian International Conference on Fluid Power, SICFP*, volume 9. pages 2–4, 2009.

Löfberg, J. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004. doi:10.1109/CACSD.2004.1393890.

Stephens, M. A., Manzie, C., and Good, M. C. Model predictive control for reference tracking on an industrial machine tool servo drive. *IEEE Transactions on Industrial Informatics*, 2013. 9(2):808–816. doi:10.1109/TII.2012.2223222.

Storn, R. Differrential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical report, International Computer Science Institute*, 1995. 11. doi:10.1023/A:1008202821328.