Proceedings of the Canadian Society for Mechanical Engineering International Congress 2020
CSME Congress 2020
June 21-24, 2020, Charlottetown, PE, Canada

A Survey of Advanced Dynamic Matrix Control Algorithms for Improved Performance

Edward Parrott,* Brad Chevarie,† Kieran Ellis,‡ Josie Versloot,§ Rickey Dubay¶
*†‡§¶ Department of Mechanical Engineering, Intelligent Controls Laboratory
University of New Brunswick, Fredericton, Canada
*Email: eparrott@unb.ca †Email: brad.chevarie@unb.ca ‡Email: kieran.ellis@unb.ca
§Email: jversloo@unb.ca ¶Email: dubayr@unb.ca

***Abstract*—Model Predictive Controllers, specifically Dynamic Matrix Controllers (DMCs), have become widespread in industry. However, they have some drawbacks, particularly when it comes to their tuning and their application to non-linear systems. This paper explores several modifications to the canonical DMC algorithm which aim to improve its performance and applicability in these situations. The formulation of each modification is presented and contrasted to the canonical formulation, and then each is tested against the other in simulation on a variety of linear and non-linear systems.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Predictive control algorithms have been widely adopted in industry due to their ability to effectively control a wide array of real world plants and processes [1]. One of the most common of these is Dynamic Matrix Control, otherwise known as DMC [2]. This process, as the name implies, utilizes a dynamic matrix to capture system dynamics, and uses this knowledge to predict future system behaviours along with a cost function to determine optimal control actions to track its given setpoint. While this is a powerful control algorithm which is able to control a wide array of systems (including multi-variable systems), it does have some drawbacks, particularly when it comes to control of non-linear systems [3]. Additionally, the algorithm can require some expertise in order to tune properly for satisfactory performance. As such, many attempts have been made to address these shortcomings in performance and ease of effective tuning.

The following paper is organized such that the basic DMC algorithm will be introduced [3], and then the modifications that have been proposed in research will be presented. Once the standard DMC algorithm has been formulated, the research presented in [4] will be presented, as it is the most logical first extension to DMC to explore. It addresses the conditionality of the dynamic matrix in DMC in order to provide better performance, and to facilitate tuning of the controller. Next, [5] presents a multi-model adaptive modification which serves to improve the performance of the algorithm for non-linear processes. Finally, [6] introduces a more rigorous approach to control of non-linear processes using a non-linear regression to re-evaluate the estimated dynamic matrix at each time step based on the current system operating point. Once the theory of each modification has been presented, each method will be tested in simulation against one another, and these results will be presented and discussed.

II. Control Theory

A. Dynamic Matrix Control

Dynamic Matrix Control (DMC) is a popular Model Predictive Control (MPC) algorithm. MPC uses a dynamic model of the plant, along with the current state of the plant to predict the future response. The control algorithm then compares this predicted output to the desired setpoint trajectory to calculate the control action. The DMC algorithm uses a step response of the plant to predict the future response and minimizes the future error over a set prediction horizon [7]. DMC is characterized by its dynamic matrix, $\mathbf{A}$, which is defined in [3] as

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ a_2 & a_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_u} & a_{n_u-1} & \cdots & a_1 \\ \vdots & \vdots & \ddots & \vdots \\ a_N & a_{N-1} & \cdots & a_{N-n_u+1} \end{bmatrix}_{N\times n_u} \tag{1}$$

where N is the prediction horizon, and n_u is the control horizon. The dynamic matrix $\mathbf{A}$ is constructed using the vector, $\mathbf{a}$, of length N that contains the normalized open loop response of the system to a step input. The future output of the system, $\hat{\mathbf{y}}$, is then predicted as [8]

$$\hat{\mathbf{y}}|_t = \hat{\mathbf{y}}|_{t-1} + \mathbf{A}\Delta\mathbf{u}_c + \phi|_t \tag{2}$$

where $\hat{\mathbf{y}}$ is a vector of length N, the dynamic matrix $\mathbf{A}$ consists of N rows and n_u columns, and $\Delta\mathbf{u}_c$ is a vector of length n_u. ϕ is a model correction factor defined as:

$$\phi|_t = \mathbf{y}_m(1) - \hat{\mathbf{y}}(1)|_t \tag{3}$$

This model correction factor is calculated at each time step and is used so that the prediction, $\hat{\mathbf{y}}$, begins at the previously measured output. This is used to help with model mismatch and shifts the predicted output accordingly.

The control action for DMC control, $\Delta\mathbf{u_c}$, is optimized using the following cost function to minimize the future errors along the prediction horizon [8].

$$J = \sum_{j=1}^{N} [\hat{\mathbf{y}}(t+j|t) - \hat{\mathbf{y}}_{sp}(t+j)]^2 + \sum_{j=1}^{n_u} \lambda [\Delta \mathbf{u}_c(t+j-1)]^2 \quad (4)$$

where $\hat{\mathbf{y}}_{sp}$ is the setpoint, λ is a move suppression factor, and the error to be minimized, $\mathbf{e}$, is between the predicted output and the setpoint,

$$\mathbf{e} = \hat{\mathbf{y}} - \hat{\mathbf{y}}_{sp} \quad (5)$$

The solution to this unconstrained optimization problem gives the following control action, $\Delta \mathbf{u}_c$:

$$\Delta \mathbf{u}_c = (\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I})^{-1} \mathbf{A}^T \mathbf{e} \quad (6)$$

Though $\Delta \mathbf{u}_c$ is a vector that contains n_u future control actions, only the first would be implemented on a practical plant. At each time step a new control action, $\Delta \mathbf{u}_c$, is computed using equations 5 and 6 and a new predicted plant response, $\hat{\mathbf{y}}$, is computed using equations 2 and 3.

To obtain optimal results using a DMC controller, four parameters can be adjusted for tuning: the move suppression factor λ, the prediction horizon N, the control horizon n_u, and the time step, dt.

B. Well Conditioned Dynamic Matrix Control

Well-conditioned dynamic matrix control is designed to improve upon some of the inherent ill-conditioning of the matrix $(\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I})^{-1}$ that is present in the standard DMC formulation.

This ill-conditioning occurs when the inverse of the system matrix $\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I}$ is calculated due to excessive changes in the manipulated variables caused by the structure of dynamic matrix $\mathbf{A}$. Traditional DMC utilizes the move suppression factor λ with the intention of improving the conditionality of the system matrix when inverted. This results in smaller changes to the manipulated variable as λ increases. However, determining λ is somewhat ambiguous process that requires a more intimate knowledge of the process dynamics [9]

In traditional DMC, as sampling time gets smaller, the difference in corresponding columns in the dynamic matrix $\mathbf{A}$ gets smaller leading to the system matrix being severely ill-conditioned when evaluated [9].

In well conditioned DMC, the prediction and control time steps are decoupled. This is done by introducing a new tuning variable m where the prediction time step is $m-1$ samples longer than the control time step. To implement this shift, m needs to be defined and is applied to the dynamic matrix where the second and subsequent columns are shifted down by $m > 1$ (because of this shift, this DMC variant is often referred to as M-Shift DMC, or simply M-Shift). Thus the new dynamic matrix should take the form [4],

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 \\ \vdots & \vdots \\ a_m & 0 \\ a_{m+1} & a_1 \\ \vdots & \vdots \\ a_{m+N} & a_N \end{bmatrix}_{(N+m)\times n_u} \quad (7)$$

Compared to the dynamic matrix in standard DMC,

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 \\ a_2 & a_1 \\ a_3 & a_2 \\ \vdots & \vdots \\ a_N & a_{N-1} \end{bmatrix}_{N\times n_u} \quad (8)$$

This shifting factor removes the need for the move suppression factor λ and is very easy to implement. Shifting factor m is chosen based off the process open loop responses and the subsequent conditioning number of the system matrix. As the condition number approaches unity, the system matrix becomes well conditioned [4].

C. Multi-Model Adaptive Dynamic Matrix Control

The multi-model adaptive dynamic matrix control (MMAC) presented in [5] uses the basic DMC algorithm as its base, but in order to more effectively control non-linear processes it combines several DMC's tuned for varying operating sub-regions to more accurately account for the differences in plant dynamics over the entire operating region. A traditional DMC uses a single step test to formulate its dynamic matrix, which is adequate for a linear process whose dynamics are simply linearly scaled throughout its operating range, but is not sufficient for a non-linear process whose dynamics vary non-linearly throughout its operating range. As such, MMAC suggests instead splitting the operating range into an array of sub-regions such that a step test in each region reasonably approximates the actual non-linear dynamics throughout it [5]. This yields a set of controllers whose outputs are subjected to a weighted average according to the current operating point in order to yield a more suitable control move than would be possible with a single DMC.

The formulation begins by dividing the operating range into n sub-regions with boundaries $y_1, y_2, ..., y_n$. Each sub-region then has a distinct DMC generated for it according to the standard methodology. During operation, at each time step each controller ($DMC_{1...n}$) receives the process output, y_m, and generates a control action $\Delta \mathbf{u}_n$. The outputs are then averaged according to the following rules, presented here for the specific case $n = 3$ [5].

The total control action $\Delta \mathbf{u}_{adap}$ is expressed as

$$\Delta \mathbf{u}_{adap} = \sum_{n=1}^{3} x_n \Delta \mathbf{u}_n \quad (9)$$

Where x_n is defined by the following rules [5]:

If $y_m \geq y_3$ then

$$x_1 = 0; x_2 = 0; x_3 = 1 \tag{10}$$

If $y_2 < y_m < y_3$ then

$$x_1 = 0; x_2 = 1 - x_3; x_3 = \frac{y_m - y_2}{y_3 - y_2} \tag{11}$$

If $y_1 < y_m < y_2$ then

$$x_1 = 1 - x_2; x_2 = \frac{y_m - y_1}{y_2 - y_1}; x_3 = 0 \tag{12}$$

If $y_m \leq y_1$ then

$$x_1 = 1; x_2 = 0; x_3 = 0 \tag{13}$$

The current time control action, $\mathbf{u}_t$ is then

$$\mathbf{u}_t = \mathbf{u}_{t-1} + \Delta \mathbf{u}_{adap} \tag{14}$$

which is analogous to the $\mathbf{u}_t$ formulation in the standard DMC.

D. Non-Linear Regression Dynamic Matrix Control

In order to react to nonlinear characteristics of a plant [6], modifications must be made to the formulation of the canonical DMC. As discussed above, the dynamic matrix **A** contains the step response of the system at an operating point. The assumption of a linear model in a specified operating range in the formulation of the DMC means that **A** is an appropriate representation of the dynamics of the system in this range. In most industrial applications, it is not possible to assume linearity over any substantial operating range. To represent a given operating range of a nonlinear system, an infinite set of piece-wise linear functions can instead be used. This allows the reevaluation of the dynamic matrix at each time step, enabling the use of the same techniques as noted for DMC with a nonlinear process.

The formulation of the non-linear regression predictive controller (NRPC) differs from the standard DMC primarily in the construction of the dynamic matrix **A** [6]. Where a single open loop test is sufficient to capture system dynamics for a linear process, a series of open loop tests are performed to capture the dynamics of a nonlinear plant over an operating range. A curve-fitted approximation of the step responses, $\mathbf{S}_k(\tilde{u})$, is built using the open loop input signals $\tilde{\mathbf{u}}$, which takes the form

$$\mathbf{S}_k(\tilde{u}) = \sum_{n=1}^{N} b_{k,n} \tilde{u}^{\mathbf{h}_n} \qquad k = 1...P \tag{15}$$

This equation represents a scalar value of the fitted normalized step response at time $k\Delta t$. The $b_{k,n}$ coefficients are the polynomial coefficients of the step inputs, and the **h** values represent the exponents which can be real or integer values. The importance of choosing proper **h** exponential values is not to be understated, as these will determine the accuracy of the curve fit of the responses, directly affecting performance of the controller [6]. This $\mathbf{S}_k(\tilde{u})$ is for a single step input $\tilde{\mathbf{u}}$.

In matrix form this appears as

$$\mathbf{S} = \tilde{\mathbf{B}} \tilde{\mathbf{U}}^T \tag{16}$$

The matrix $\tilde{\mathbf{B}}$ contains the aforementioned coefficients, while $\tilde{\mathbf{U}}$ contains the step inputs and **h** exponents. Both are expressed below as

$$\tilde{\mathbf{B}} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N} \\ b_{21} & b_{22} & \cdots & b_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ b_{P1} & b_{P2} & \cdots & b_{PN} \end{bmatrix} \tag{17}$$

$$\tilde{\mathbf{U}} = \begin{bmatrix} \tilde{u}^{h_1} & \tilde{u}^{h_2} & \cdots & \tilde{u}^{h_N} \end{bmatrix} \tag{18}$$

In order to determine the $\tilde{\mathbf{B}}$, a matrix containing m open loop input tests and **h** exponents is defined as

$$\Phi = \begin{bmatrix} \tilde{U}_1 \\ \tilde{U}_2 \\ \vdots \\ \tilde{U}_m \end{bmatrix} = \begin{bmatrix} \tilde{u}_1^{\mathbf{h}_1} & \tilde{u}_1^{\mathbf{h}_2} & \cdots & \tilde{u}_1^{\mathbf{h}_N} \\ \tilde{u}_2^{\mathbf{h}_1} & \tilde{u}_2^{\mathbf{h}_2} & \cdots & \tilde{u}_2^{\mathbf{h}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{u}_m^{\mathbf{h}_1} & \tilde{u}_m^{\mathbf{h}_2} & \cdots & \tilde{u}_m^{\mathbf{h}_N} \end{bmatrix}_{m \times N} \tag{19}$$

A matrix containing the m open loop normalized test responses is also constructed as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 & \cdots & \mathbf{Q}_m \end{bmatrix}_{P \times m}^T \tag{20}$$

The least squares optimization method is then used [10] to construct $\tilde{\mathbf{B}}$ as

$$\tilde{\mathbf{B}}^T = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{Q} \tag{21}$$

Using the resulting $\mathbf{S}_k(\tilde{u})$, a dynamic matrix **A** can be evaluated at each time step [6] using the input $\tilde{\mathbf{u}}$

$$\mathbf{A} = \begin{bmatrix} \mathbf{S}_1(\tilde{u}) & 0 & \cdots & 0 \\ \mathbf{S}_2(\tilde{u}) & \mathbf{S}_1(\tilde{u}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_P(\tilde{u}) & \mathbf{S}_{P-1}(\tilde{u}) & \cdots & \mathbf{S}_{P-n_u+1}(\tilde{u}) \end{bmatrix}_{P \times n_u} \tag{22}$$

As in the canonical DMC formulation, the calculation for the control moves is performed by evaluating the change in control moves $\Delta \mathbf{u}_c$ for the current time step within the control loop. This is once again performed by minimizing the objective function

$$min_{\Delta \mathbf{u}_c} J = [\mathbf{e} - \mathbf{A}\Delta \mathbf{u}_c]^T [\mathbf{e} - \mathbf{A}\Delta \mathbf{u}_c] + \Delta \mathbf{u}_c^T \lambda \Delta \mathbf{u}_c \tag{23}$$

Where λ is an additional term for move suppression, also utilized in the standard DMC. The vector **e** represents the future errors (difference between setpoint and prediction), where the predicted output is calculated using:

$$\hat{\mathbf{y}}(t+k|t) = y_0 + \sum_{v=(k-n_u+1)}^{k} \mathbf{S}_v(\tilde{u})\Delta\mathbf{u}_c(t+k-v|t) + \sum_{v=(k+1)}^{k+P-1} \mathbf{S}_v(\tilde{u})\Delta\mathbf{u}_c(t+k-v|t) \quad (24)$$

The result of the minimization of the objective function is the change in control move

$$\Delta\mathbf{u}_c = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{e} \quad (25)$$

This is then added to the previous control action to acquire the new control action, of which only the first term of each manipulated variable is relevant [6].

III. Results

Several tests were performed in order to compare the performance of each controller on several different plants (both linear and non-linear) and to highlight their specific advantages and drawbacks. The test plants were modeled as first order plus deadtime systems, as is standard practice when taking step responses for formulation of dynamic matrices, and are presented in their discrete forms.

Plant 1, from [4], is a linear plant described as

$$\mathbf{y}_k = 0.9777\mathbf{y}_{k-1} + 0.02362\mathbf{u}_{k-1} \quad (26)$$

with $dt = 16$ seconds and $\lambda = 0.14$. Additionally, its m factor for the M-Shift controller is m = 2. Plant 2, also from [4] is a linear plant described as

$$\mathbf{y}_k = 0.9842\mathbf{y}_{k-1} + 0.01886\mathbf{u}_{k-1} \quad (27)$$

with $dt = 16$ seconds and $\lambda = 0.15$. Additionally, its m factor for the M-Shift controller is m = 3. The process for choosing m, λ, and dt are explained in greater detail in [4]. When applying the non-linear regression dynamic matrix control, both plants 1 and 2 used $h = [0.5, 1, 1.5, 2, 2.5]$ as their set of fitted exponents. This was determined by testing various combinations of exponents against the set of step test results until the combination resulting in the best overall fit throughout the operating range was found.

Plant 3, from [11], is a non-linear plant described as

$$\mathbf{y}_k = \mathbf{y}_{k-1}^3 - 0.2|\mathbf{y}_{k-1}|\mathbf{u}_{k-1} + 0.08\mathbf{u}_{k-1}^2 \quad (28)$$

with $dt = 0.003$ and $\lambda = 1.5$. The set of fitted exponents for the non-linear controller, from [6], is $h = [0.5, 1, 2, 3, 4]$. Finally, plant 4, from [12], is a non-linear plant described as

$$\mathbf{y}_k = \frac{\mathbf{y}_{k-1}\mathbf{y}_{k-2}[\mathbf{y}_{k-1} + 2.5]}{1 + [\mathbf{y}_{k-1}]^2 + [\mathbf{y}_{k-2}]^2} \quad (29)$$

with $dt = 0.003$ and $\lambda = 2$. The set of fitted exponents for the non-linear controller, determined similarly to those for plants 1 and 2, is $h = [0.5, 2, 2.5, 3, 3.5]$.

The first test was implemented on plant 1, using a step setpoint $\mathbf{y}_{sp} = 5$. The results of this test are presented in figure 1 and table I. Table I compares percent overshoot, mean squared error, and settling time (in seconds) for each controller, as will all further performance tables.

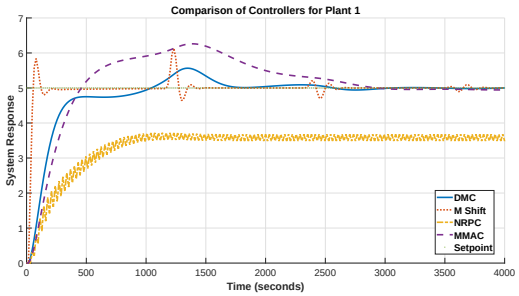

Fig. 1. Plant 1 results

TABLE I
Plant 1 Performance Summary

Controller	%OS	MSE	S.T.
DMC	11.2937	0.5296	1664
M-Shift	22.7254	0.1747	3648
MMAC	25.2240	0.7189	2752
NRPC	-25.5654	2.9199	Inf

In this scenario, the two linear controllers (DMC and M-Shift) drastically outperform the non-linear ones (NRPC and MMAC). This is reasonable, as they are explicitly formulated for linear plants and thus lack the additional overhead computations present in the non-linear formulations which lead to reduced performance in this scenario. Because it is essentially a combination of linear controllers, MMAC was still able to reach the setpoint, but because of its more general formulation to allow for non-linear control, it is not able to reach the desired setpoint as quickly as DMC or M-Shift. It should be noted that in all of these experiments, the MMAC formulation used three component DMCs. Additionally, M-Shift performs better than standard DMC, a fact that is most likely due to the improved conditionality of its dynamic matrix. However, M-Shift does exhibit some odd behaviors which are almost reminiscent of what would be expected from a disturbance rejection once it has settled. This could be a result of either the controller or the plant. This shows in its longer settling time. Additionally, the NRPC's settling time is listed as infinite in this case as it never converges closely enough to the desired setpoint to meet the settling criteria set out for this study (this study considers a process to have settled once the process output is within $\pm 2\%$ of the desired output for all future time steps).

Tests on plant 2, whose results are presented in figure 2 and table II, were conducted with a step setpoint $\mathbf{y}_{sp} = 5$.

These tests returned similar results to the tests on plant 1, as expected. The only notable difference is in the performance of the non-linear plants. The NRPC seemed to more closely track the setpoint (although it was still subject to small oscillations

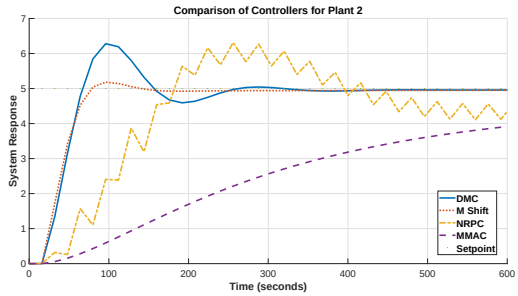

Fig. 2. Plant 2 results

TABLE II
PLANT 2 PERFORMANCE SUMMARY

Controller	%OS	MSE	S.T.
DMC	25.6063	1.7999	272
M-Shift	3.6389	1.5933	144
MMAC	0.5318	0.8657	4112
NRPC	26.2279	3.8456	Inf

about its operating point, causing it to once again fail to meet settling criteria), and the MMAC took much longer to reach the setpoint. These differences are most likely due to the differences between the plants, though, as opposed to the formulations of the given controllers. M-shift also did not display the same odd oscillations after settling it showed on plant 1 (allowing it to achieve a much faster settling time than all other controllers), which suggest that these were more a result of the system dynamics than the controller dynamics. This result is also almost analogous to what could be expected from an unstable plant with a persistent excitation condition, which could be an interesting exploration for future work.

Figure 3 and table III show the results for the first of the non-linear plants, plant 3 with a setpoint $\mathbf{y}_{sp} = 0.5$. It should be noted that the results presented therein only show the performance of the NRPC and the MMAC, as neither standard DMC or M-Shift methodologies were able to achieve stable performance. This instability is most likely due to the fact that the non-linear system dynamics over its operating range make it such that control moves calculated at operating points far from that at which the step response was calculated would be inappropriate, resulting in unpredictable behavior.

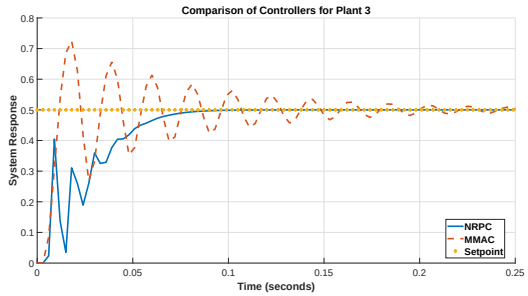

Fig. 3. Plant 3 results

TABLE III
PLANT 3 PERFORMANCE SUMMARY

Controller	%OS	MSE	S.T.
MMAC	44.9558	0.0016	0.2430
NRPC	0	0.0054	0.0810

In this instance, NRPC quite clearly outperforms the MMAC controller. This is most likely due to its method of handling non-linearities being much more rigorous than the MMAC's strategy of simply using a combination of models at different operating points to approximate the non-linearities. While MMAC does eventually settle to the desired setpoint, the oscillations it shows about this operating point would almost certainly prove undesirable in a real application.

Finally, the results for plant 4 (using a setpoint $\mathbf{y}_{sp} = 0.75$, presented in figure 4 and table IV, show an interesting set of results for non-linear control.

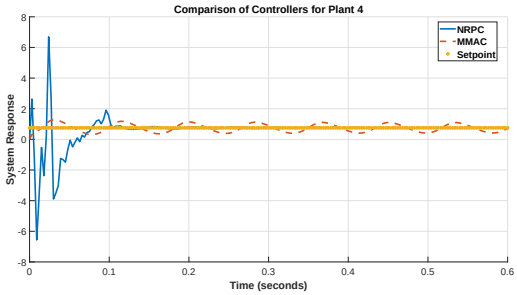

Fig. 4. Plant 4 results

TABLE IV
PLANT 4 PERFORMANCE SUMMARY

Controller	%OS	MSE	S.T.
MMAC	69.9039	0.0609	Inf
NRPC	791.7361	0.7783	0.4170

For this specific plant, NRPC converges almost exactly to the desired setpoint, resulting in zero steady state error, as opposed to MMAC which does reach the desired setpoint but exhibits significant oscillations about. However, in converging to the setpoint, the NRPC experiences a very large and dramatic transient. While it was able to maintain system stability in this scenario, this behavior could prove dangerous when applied to real-world system. It is worth noting though that this behavior is most likely due to a poor non-linear regression fit in the control formulation due to a sub-optimal choice of $\mathbf{h}$ exponents. This could be addressed by using a more rigorous method of $\mathbf{h}$ selection, perhaps using a local or global optimization method such as genetic algorithms [13]. These results make it difficult to say conclusively that one controller is better than the other in this scenario, as both could most likely achieve better performance with more rigorous tuning.

IV. Conclusions

The aim of this paper was to show various improvements to the canonical DMC algorithm that have been developed, and compare them against each other in order to determine their individual advantages and disadvantages. It is clear that for linear plants, M-Shift is the superior choice. It settled most quickly and accurately, and exhibited less overshoot and oscillations than any of the other controllers. MMAC was able to to also effectively control the system, however it did so much more slowly than any others. Thus, for a linear system, although MMAC may be a functional option, it is not necessarily the optimal choice. For linear systems, the NRPC was easily the least effective. This could be due to it attempting to to account for perceived non-linearities which could appear in the regression analysis that may not actually be present in the system dynamics. Thus, not only is it not an effective choice for control of linear plants, due to its extensive regression analysis required in its formulation it is also an overly complex solution.

For non-linear plants, canonical DMC and M-Shift algorithms failed outright. This could be addressed in practice by using them in a much smaller operating range for which the system dynamics can be approximately expressed a linear, or by using some manner of feedback linearization. While MMAC and NRPC were able to control the systems, neither of them was able to do it perfectly. NRPC displayed aggressive transients while it moved through the smallest operating ranges. This is most likely due to the non-linear regression fits not being as good at small ranges (analysis of the fits does usually show this), but could also be a result of sub-optimal choices for the $\mathbf{h}$ exponents. This could be addressed in practice with better methods for selecting $\mathbf{h}$, or by increasing the resolution for the range of open loop test inputs, $\tilde{u}$. MMAC, meanwhile, was able to reach the setpoint but exhibited significant oscillations in both cases. This is most likely a flaw in the tuning of its individual component DMCs, as well as their inability to truly effectively capture non-linear dynamics. This could be remedied with more careful tuning of the component DMCs, as well as by increasing the number of component DMCs such that the linear approximations of each of their respective operating ranges are improved.

Acknowledgment

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada for their support on this project.

References

[1] J. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[2] C. Cutler and B. Ramaker, "Dynamic matrix control – a computer control algorithm," in *Proceedings of the Joint Automatic Control Conference, San Francisco*, 1980.

[3] E. Camacho and C. Bordons, *Model Predictive Control*, M. A. J. Michael J. Grimble, Ed. Springer, 2004, vol. 13.

[4] R. Dubay, G. Kember, and B. Pramujati, "Well-conditioned model predictive control," *ISA Transactions*, vol. 43, no. 1, pp. 23–32, Jan. 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019057807600172

[5] D. Dougherty and D. Cooper, "A practical multiple model adaptive strategy for single-loop MPC," *Control Engineering Practice*, vol. 11, no. 2, pp. 141–159, Feb. 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066102001065

[6] R. Dubay, M. Abu-Ayyad, and J. Hernandez, "A nonlinear regression model-based predictive control algorithm," *ISA Transactions*, vol. 48, no. 2, pp. 180–189, Apr. 2009.

[7] A. Abbas, "Dynamic matrix control (dmc) of rolling mills," *Materials and Manufacturing Processes*, vol. 22, no. 7-8, pp. 909–915, 2007. [Online]. Available: https://doi.org/10.1080/10426910701451788

[8] J. Wilson, M. Charest, and R. Dubay, "Non-linear model predictive control schemes with application on a 2 link vertical robot manipulator," *Robotics and Computer-Integrated Manufacturing*, vol. 41, pp. 23–30, Oct. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584516300564

[9] B. Pramujati, "Development of shifted dynamic matrix control," Ph.D. dissertation, University of New Brunswick, 2004. [Online]. Available: https://books.google.ca/books?id=GuJ1PwAACAAJ

[10] K. Astrom and B. Wittenmark, *Computer controlled systems*. Prentice Hall, 1997.

[11] J. Zhao, V. Wertz, and R. Gorez, "A fuzzy clustering method for the identification of fuzzy models for dynamic systems," in *Proceeding of the 9th IEEE international symposium on intelligent control*, 1994, pp. 172–179.

[12] M. Abu-Ayyad and R. Dubay, "Improving the performance of generalized predictive control for nonlinear processes," *Industrial & Engineering Chemistry Research*, vol. 49, no. 10, pp. 4809–4816, 2010.

[13] K. Kristinsson and G. A. Dumont, "System identification and control using genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 5, pp. 1033–1046, 1992.