International Journal of Engineering and Technology Innovation, Vol. 5, No. 1, 2015, pp. 19-32

A Simplified Approach to Multivariable Model Predictive Control

Michael Short*

Electronics & Control group, Teesside University, Middlesbrough, UK.

Received 27 May 2014; received in revised form 05 December 2014; accepted 09 December 2014.

Abstract

The benefits of applying the range of technologies generally known as Model Predictive Control (MPC) to the control of industrial processes have been well documented in recent years. One of the principal drawbacks to MPC schemes are the relatively high on-line computational burdens when used with adaptive, constrained and/or multivariable processes, which has warranted some researchers and practitioners to seek simplified approaches for its implementation. To date, several schemes have been proposed based around a simplified 1-norm formulation of multivariable MPC, which is solved online using the simplex algorithm in both the unconstrained and constrained cases. In this paper a 2-norm approach to simplified multivariable MPC is formulated, which is solved online using a vector-matrix product or a simple iterative coordinate descent algorithm for the unconstrained and constrained cases respectively. A CARIMA model is employed to ensure offset-free control, and a simple scheme to produce the optimal predictions is described. A small simulation study and further discussions help to illustrate that this quadratic formulation performs well and can be considered a useful adjunct to its linear counterpart, and still retains the beneficial features such as ease of computer-based implementation.

Keywords: Real-time and embedded control, predictive control, multivariable control.

1. Introduction

Model Predictive Control (MPC) schemes employ dynamic models of a process in conjunction with on-line optimization schemes to compute an optimal sequence of input moves which minimizes the predicted future values of an objective function [1][2]. The principal components of most MPC implementations are as shown in Fig. 1. The potential benefits of applying MPC strategies to complex industrial processes has been well documented in recent years, and includes the ability to deal with large time delays, non-minimum phase (inverse response) behaviors, and tightly-coupled multivariable systems [1]. MPC is a form of receding-horizon optimal control, and provides a large amount of flexibility when compared to traditional control schemes such as pole-placement [16][20]. However one of its principal drawbacks has been the relatively high on-line computational burden when used with adaptive and/or multivariable schemes [1][2]. This has warranted some researchers and practitioners to seek simplified approaches for its implementation. Although most modern MPC schemes utilize quadratic (2-norm) objective functions which are solved on-line by Quadratic Programming (QP) software, much of the original work on MPC algorithms utilized linear (1-norm) objective functions which were solved on-line by Linear Programming (LP) software such as the well-known Simplex algorithm [1][3]. This transition is due to several factors, principally due to advances in QP solving techniques and a number of well-documented drawbacks to the use of LP formulations in MPC. The drawbacks include possible idle/deadbeat dichotomous behaviors [3], the need to use iterative schemes to obtain solutions even in the unconstrained case [1][3], and potentially poor scaling in the size of the MPC problem to the corresponding LP [4]. Despite this, it has been argued that in some situations LP-based MPC may still be effective [3].

* Corresponding author. E-mail address: m.short@tees.ac.uk
Tel.: +44 (0)1642 342 528

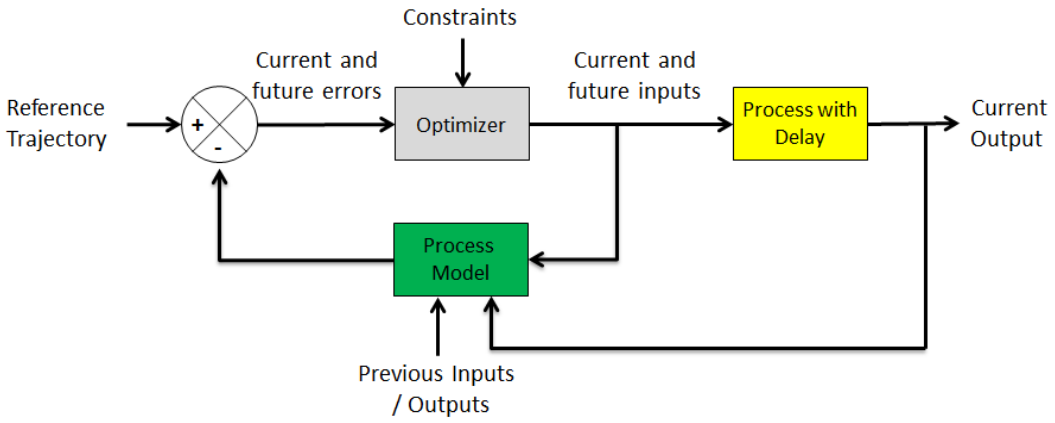

Fig. 1 Main elements of an MPC strategy

One such situation has received considerable attention in recent years, this is the Simplified Predictive Control (SPC) technique first described by Gupta [4]. In this approach, a number of simplifications are made to an LP formulation of an MPC instance with the main aim of reducing the run-time complexity, and also to allow an easier implementation. The simplifications proposed in SPC are such that the impact on control loop performance is minimal. Although principally aimed at multivariable MPC, the SPC technique can also be applied to single variable control; as was shown in [8], the performance of single-variable SPC can be made identical to a well-conditioned Dynamic Matrix Control (DMC) implementation. The SPC approach has been successfully applied in industry [5] and various extensions have been reported [6], along with favorable experimental comparisons to other MPC approaches [7] and analytical studies [8]. However, SPC also has several notable drawbacks. In this paper it will be argued that as with other MPC strategies, the adoption of a 2-norm objective function in SPC retains much of its beneficial properties and also has the ability to overcome some of the drawbacks. A small simulation study and further discussions help to illustrate that this quadratic formulation performs well and can be considered a useful adjunct to its linear counterpart, and still retains the beneficial features such as ease of computer-based implementation. The remainder of this paper is organized as follows. Section 2 describes the SPC approach in more detail and outlines the nature of its potential drawbacks. In Section 3, the QP formulation - SPC_2 - is introduced and a simple algorithm is presented to obtain optimal output predictions. In Section 4, a simple iterative algorithm is presented to solve constrained SPC_2 problems. Section 5 presents a small simulation study to provide an initial validation of the behavior of the proposed approach, and the paper is concluded in Section 6.

2. Simplified Multivariable MPC

Consider a potentially over-actuated Multi-Input Multi Output (MIMO) industrial process, such as the one depicted in Fig. 2, which has a number of interacting control loops featuring $m > 1$ controlled (output) variables and $n \geq m$ manipulated (input) variables. Such a process may be represented as an m-by-n matrix of transfer functions [1][2]. The SPC approach to control of this process is to select, for each of the m outputs under control, a single point p_i steps ahead on the prediction horizon at which the error is to be minimized. For each manipulated variable j, only a single control move u_j^k is calculated and applied at the current step discrete k, such that the vector of predicted future errors is minimized.

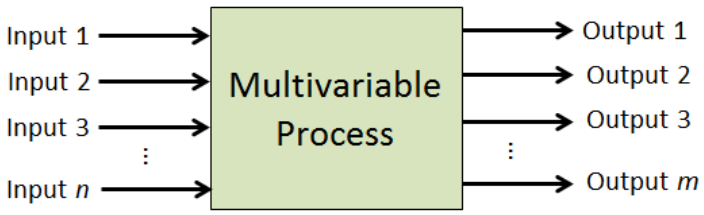

Fig. 2 Multivariable process with n inputs and m outputs.

The p_i values thus become tuning parameters, and by increasing (decreasing) their values, a slower (faster) response can be achieved in each loop in a similar fashion to the use of move suppression co-efficients in regular MPC strategies such as DMC and Generalized predictive Control (GPC) [4][5][7]. Choice of the p_i values cannot be made arbitrarily, they must be chosen such that they are greater than the time delay and/or inverse response of process loop i in order to prevent instability [4][8]. Prior to the calculation of the control moves at step k, the predicted error in each loop p_i steps into the future can be obtained by subtracting the predicted output of the process from the future reference:

$$e_i^{k+p_i} = r_i^{k+p_i} - \hat{y}_i^{k+p_i} \tag{1}$$

Where $\hat{y}_i^{k+pi}$ is an optimal prediction of the free response of output y_i, assuming the previously applied control signal vector u^{k-1} is held constant. If the future reference value r_i^{k+pi} is not known, then it can be assumed that the current set point signal r_i^k is held constant over the prediction horizon. In the original SPC approach, a step response model is used to obtain the predicted free response of the process in a similar fashion to the DMC algorithm [4]. Let the predicted change in output j at p_j steps ahead due to a step change in input u_i be given by the step co-efficient g_{ij}, which can be obtained with knowledge of the process transfer function matrix. Then by appropriate manipulations to the n process inputs, the predicted p–step ahead errors can be minimized for each loop. Thus, once the free errors for each loop have been obtained at step k, a constrained linear optimization problem is then required to be solved:

$$\begin{aligned} &\text{minimize}: \ \|G\Delta u - e\|_1 \\ &\text{with respect to}: \Delta u \\ &\text{subject to}: \ \Delta u_{\min} \le \Delta u \le \Delta u_{\max}; \\ &\qquad\qquad e_{\min} \le G\Delta u \le e_{\max}; \end{aligned} \tag{2}$$

Where Δu is the n-vector of applied incremental control moves (i.e. $u^k = u^{k-1} + \Delta u^k$), e is the m-vector of predicted future errors, Δu_{max} and Δu_{min} are n–vectors representing lower and upper bounds on the allowed control moves, e_{max} and e_{min} are m–vectors representing lower and upper bounds on the allowed errors, and G is an m-by-n matrix of p-step ahead coefficients:

$$G = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1m} \\ g_{21} & g_{22} & \cdots & g_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nm} \end{bmatrix} \tag{3}$$

Although input rate-of-change (velocity) limits and output error constraints are seemingly only present in the constraints defined in equation (2), due to the simplified nature of the problem position and output constraints can be explicitly enforced at each step as follows. Suppose the lower and upper actuator and output position constraints are denoted by u_{min}, u_{max} , y_{min} and y_{max} respectively. At iteration k, if the constraints for (2) are first updated according to (4), then these additional constraints will also be enforced by its solution:

$$\begin{aligned} \Delta u_{\max}^k &= \min(\Delta u_{\max}, u_{\max} - u^{k-1}) \\ \Delta u_{\min}^k &= \max(\Delta u_{\min}, u_{\min} - u^{k-1}) \\ e_{\max}^k &= y_{\max} - \hat{y}^{k+p} \\ e_{\min}^k &= y_{\min} - \hat{y}^{k+p} \end{aligned} \tag{4}$$

Note that as with all constrained MPC approaches, there may be situations in which both the input and error constraints may not be simultaneously satisfied, in other words (2) is infeasible [1][2]. To overcome this problem, it is common to drop the error constraints in (2) and consider only rate and position constraints on the input, and set warning alarms should the future output trajectory be predicted to deviate from its bounds [2][4][9]. This results in a simplified LP formulation with $V = 2m + 2n$ variables and C = $n + m$ constraints [3]. The SPC algorithm described above has many attractive features compared to regular multivariable MPC schemes, principally with reduced on-line computation times and easier implementation. In terms of potential performance reduction due to the minimization of only a single point on the future trajectory, as noted in [4] this loss of performance should be minimal, as each point on the trajectory is still optimized, but only one step at a time. However, it also has several notable drawbacks, including:

(i) Even in the absence of constraints, there is no analytical solution to (2) and an iterative solution such as the Simplex algorithm must therefore be employed; (ii) In order to achieve robust behavior, the p_i points may have to be projected a large distance into the future, which reduces the quality of the output predictions and (iii) Small changes in the predicted future errors can cause large changes in the applied control signals between iterations k and k+1, as the controls are discontinuously mapped to the errors due to the use of the least absolute deviations (1-norm) objective function.

All of these points may be problematic in terms of real-time control implementations with embedded processing systems. The first point may be problematic as control systems, by definition, are real-time in nature and many have hard timing constraints; in these situations, worst-case behavior in terms of execution time must be taken into account when deciding if the application software is schedulable on the current computing platform [10]. Although the average-case time complexity of the simplex algorithm (and its known variants) requires approximately $[C+V]^3$ operations - which is cubic in the number of variables and constraints - its worst-case run-time complexity is $O(2^{[C+V]})$ and hence exponential [11]. As this potential for occasional exponential behavior must surely be accounted for, in a dependable design the computing platform will be potentially under-utilized for much of its lifetime [10]. Note that the reliability and stability of the run-time computations required for most *constrained* MPC implementations is problematic generally, and is considered to be a major obstacle and an area of much-needed research [2]. In the regular SPC approach, this problem also applies equally to the *unconstrained* problem.

The second point can lead to a loss of performance, especially in cases when the system is subject to unmeasured disturbances and model-process mismatch, which is to be expected in most industrial implementations [1][2]. As noted in [1], it is impossible to completely prevent noise entering a system and affecting future process predictions, and hence there is no advantage to be gained in selecting the *start* of the prediction horizon beyond the process time delay d as the quality of the output predictions decreases. For SPC to be effective in many cases, it must be set well-beyond this lower bound in order to obtain a robust response [4][5][8]. However, when optimizing only a single error on the predicted output trajectory, as mentioned above care must also be taken for loops exhibiting inverse responses, regardless of the way the objective function is formulated: original formulations of the Minimum Variance (MV) control laws also suffered instability problems when optimizing only a single point d+1 steps into the future, despite the use of a quadratic loss function [19].

The final point can result in „chattering" of the actuators, i.e. large input changes occurring due to small changes in the predicted errors [3][11]. This can be especially problematic in systems with an ill-conditioned G matrix, and will require the addition of control signal penalties into the objective function, thus increasing the number of variables in the LP formulation and increasing computational demand further. The points listed above warrant the search for an SPC algorithm that retains much of the beneficial properties as the original, but also with the ability to overcome the highlighted problems. Such an algorithm may be obtained by adopting a 2-norm objective function, and will be described in the following Section.

3. A 2-Norm Approach to Simplified Multivariable MPC

As may be expected, many features of the proposed 2-norm approach to SPC deliberately remain identical to the original formulation. Again a single point p_i steps ahead on the prediction horizon is selected for error minimization, only a single control move for each input is calculated, and only rate constraints (plus position constraints via. (4)) on the input are considered. A discrete transfer function model with an input disturbance is used to obtain the predicted free responses (and hence the errors via equation (1)). A simple procedure for calculating predictions is described later in this Section. The main difference in the formulation of the 2-norm SPC technique (SPC_2) is the use of a quadratic objective function, which has many associated benefits which are recapped below:

(i) The use of 2-norm objective function allows for an analytical Least Squares (LS) solution to obtain the optimal control moves in the absence of constraints; (ii) Regularization (move suppression) parameters can be easily added into the LS objective function, loosening the dependence between loop robustness and the future prediction point (p_i) values, allowing them to be potentially reduced and temporarily nearer (improved) predictions used, and (iii) The LS approach is stable in that smooth changes in the future predicted errors result in only smooth changes in the control variables, which are continuous functions of the errors.

The constrained case reduces to a relatively simple „box" constrained LS problem with no more than n constraints, for which a simple iterative solver is developed in Section 4.

3.1. Unconstrained case

For the unconstrained case, replacing the objective function of (2) with its 2-norm equivalent and adding regularization parameters to provide a weighted penalty on the size of the output control moves gives the following simple optimization problem to be solved at each step k:

$$\begin{aligned} &\text{minimize}: \ \left\| G\Delta u - e \right\|_2^2 + \left\| diag(\lambda) \cdot \Delta u \right\|_2^2 \\ &\text{with respect to}: \Delta u \end{aligned} \tag{5}$$

Where Δu, l, u, e and G are as previously defined, and λ is an n-vector of regularization (move suppression) parameters such that each $\lambda_i \geq 0$, $1 \leq i \leq n$. Note that without loss of generality, a squared 2-norm objective is employed in (5) as the regularization parameters are dimensionless. Clearly, (5) is a simple regularized LS problem; it is a special case of the multivariable Generalized Predictive Control (GPC) scheme (described in, for example, [1]) featuring only a single point on the prediction horizon to be optimized per output. Minimization of (5) leads to the following analytical solution to obtain Δu:

$$\Delta u = \left((G^T G) + diag(\lambda) \right)^{-1} G^T e \tag{6}$$

The control matrix $C = (G^T G + \text{diag}(\lambda))$ may be easily inverted (or Cholesky factorized into $C = LL^T$) off-line during the design stage [13][14], once the move suppression parameters and p_i points have been chosen. At each time step k, once the predicted error vector e has been determined, the optimal control moves can be determined in $O(n^2)$ with a simple matrix-vector multiplication or solution via the Cholesky triangle [12][13][14]. Equation (6) may be interpreted as a state-feedforward control law, in which the current controls are determined as affine functions of the predicted future errors. Note that as mentioned above, the 1-norm objective function defined by equation (2) has no analytical (closed-form) solution that can be directly expressed in a form similar to equation (6). Online optimization is required for the solution of (2), and hence an analytical comparison of the control laws in the unconstrained cases is difficult to perform. In the case where λ is required to be tunable on-line, it may not be practical to re-compute the full matrix inverse or factorization of C in small

embedded systems, as in the worst-case if each element of the vector λ changes simultaneously, this a full-rank update of C and hence a full rank correction to C^{-1} is required. However, under the (not unreasonable) assumption that at every time step, only a single element of λ is likely to change, then the technique of [15] may be employed to efficiently update C^{-1}. Assuming that the k^{th} element of λ changes by an amount $\Delta\lambda_k$, then C^{-1} may be efficiently updated according to:

$$\Delta c_{ij}^{-1} = -\frac{c_{ik}^{-1} c_{kj}^{-1} \Delta\lambda_k}{1 + c_{kk}^{-1} \Delta\lambda_k} \tag{7}$$

Where c_{ij} (c_{ij}^{-1}) is the element corresponding to the i^{th} row and j^{th} column of C (C^{-1}). Computation of (7) may clearly be performed online with complexity $O(n^2)$.

3.2. Constrained case

In the constrained case, the following „box" constrained quadratic program is required to be solved at each step k:

$$\begin{aligned} &\text{minimize}: \ \left\| G\Delta u - e \right\|_2^2 + \left\| diag(\lambda) \cdot \Delta u \right\|_2^2 \\ &\text{with respect to}: \Delta u \\ &\text{subject to}: \Delta u_{\min} \le \Delta u \le \Delta u_{\max} \end{aligned} \tag{8}$$

There are several ways to obtain a valid vector of control moves satisfying the constraints in (8), but there is only one *optimal* vector [12]. The simplest way to obtain a solution is to „clip" the unconstrained solution; although this works well some of the time, it can on occasion be arbitrarily far from optimality. In Section IV, a simple iterative algorithm for the solution of (8) will be described.

3.3. Prediction Model

Consider again the multivariable process depicted in Fig. 2. The input/output relationships of the continuous-time dynamics may be expressed in matrix form as:

$$Y(s) = G_p(s) U(s) \tag{9}$$

where:

$$Y(s) = \begin{bmatrix} Y_1(s) \\ Y_2(s) \\ \vdots \\ Y_m(s) \end{bmatrix}, G_p(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) & \cdots & G_{1m}(s) \\ G_{21}(s) & G_{22}(s) & \cdots & G_{2m}(s) \\ \vdots & \vdots & \vdots & \vdots \\ G_{n1}(s) & G_{n2}(s) & \cdots & G_{nm}(s) \end{bmatrix}, U(s) = \begin{bmatrix} U_1(s) \\ U_2(s) \\ \vdots \\ U_n(s) \end{bmatrix} \tag{10}$$

Assuming a Zero-Order-Hold on the inputs and taking z-transforms gives:

$$Y(z) = G_p(z) U(z) \tag{11}$$

where:

$$Y(z)=\begin{bmatrix} Y_1(z) \\ Y_2(z) \\ \vdots \\ Y_m(z) \end{bmatrix}, G_p(z)=\begin{bmatrix} G_{11}(z) & G_{12}(z) & \cdots & G_{1m}(z) \\ G_{21}(z) & G_{22}(z) & \cdots & G_{2m}(z) \\ \vdots & \vdots & \vdots & \vdots \\ G_{n1}(z) & G_{n2}(z) & \cdots & G_{nm}(z) \end{bmatrix}, U(z)=\begin{bmatrix} U_1(z) \\ U_2(z) \\ \vdots \\ U_n(z) \end{bmatrix} \tag{12}$$

A matrix factorization (e.g. using a left co-prime representation) of $G_p(z)$ into $G_p(z) = A^{-1}(z)B(z)$ allows (11) to be re-written as:

$$A(z)Y(z)=B(z)U(z) \tag{13}$$

Adding a disturbance model consisting of an integrated white sequence $w(k)$ with zero mean and finite variance gives the multivariable Controlled Auto-Regressive Integrated Moving Average (CARIMA) model commonly used in GPC [1]:

$$A(z)Y(z)=B(z)U(z)+\frac{I(z)}{\Delta}W(z) \tag{14}$$

Where $\Delta = 1\text{-}z^{-1}$ is the difference (delta) operator and I is an n-by-n identity matrix. The advantage of using the model (14) is that the closed loop response will contain an integrator. To obtain an output predictor from (14), first multiply throughout by Δ to give:

$$A(z)\Delta Y(z)=B(z)\Delta U(z)+I(z)W(z) \tag{15}$$

The usual procedure for obtaining free response predictions using (15) is to employ Diophantine recursions [1]. However, a simplified recursive procedure for predicting the process free response is now described. At discrete time step k, it can be observed that the expected future values of the random disturbance term $w(k+i) = 0$ for all $i > 0$, and since the free response is required, it can be also be assumed that the future controls $\Delta u(k+i) = 0$ for all $i > 0$. Hence the predicted *change* in any output j, $\Delta\hat{y}_j(k+1)$, is trivially obtained from (15) using knowledge of the (known) previous input/output increments Δy and Δu. Similarly, one may easily obtain $\Delta\hat{y}_j(k+2)$ recursively, replacing $\Delta y_j(k+1)$ with $\Delta\hat{y}_j(k+1)$ as required, and then $\Delta\hat{y}_j(k+3)$, and so on until $\Delta\hat{y}_j(k+p_j)$ is reached. Obtaining the required p-step ahead output prediction $\hat{y}_j(k+p_j)$ is then trivially achieved through integration:

$$\hat{y}_j(k+p_j)=y_j(k)+\sum_{l=1}^{p_j}\Delta\hat{y}_j(k+l) \tag{16}$$

This can be easily programmed onto a microcomputer and does not require the use of Diophantine recursions or specialized software. Finally, note that if the disturbance terms are correlated, a coloring polynomial $C(z)$ may be used in (14) instead of the identity matrix I. In this case the terms of $C(z)$ may be absorbed into the (suitably expanded) $A(z)$ and $B(z)$ polynomial matrices [1], and the method outlined above applied thereafter. This is perhaps only useful in the adaptive case, since the terms of $C(z)$ are very difficult to identify and in most cases time-varying [16].

3.4. Parameter Tuning

As with most MPC formulations, the SPC_2 approach has several „tuning" parameters that can be used to modify the nature of the closed-loop behavior [1][2]. The main parameters in SPC_2 are the vector of prediction points P and the vector of move suppression coefficients λ. For tuning, it is suggested that the following rule-of-thumb may be applied for open-loop

stable processes. Suppose that the combined process dead-time and inverse response contribution for loop i is d_i sample times, and the open-loop 95% settling time for loop i is approximately s_i sample times. Initial simulations suggest that selection of $p_i \in (d_i, s_i/2]$ will give adequate results. With these values selected, then a value for each parameter λ_i is required. Initial experience suggests that they should be selected such that the control matrix C is well-conditioned, with a reciprocal condition number that is approximately ≥ 0.1 to ensure smooth control actions and less sensitivity to noise. The move suppression coefficients can then be adjusted through further simulation to produce the desired performance as needed before application to the real plant.

4. Reduced Complexity QP Algorithm for SPC$_2$

The constrained QP technique described in the previous Section has several attractive features as the constraints are only applied to the inputs, leading to a „box-constrained" least squares optimization problem. Several techniques are known to be effective in this situation, the most efficient of which are based around „active set" techniques [1][17]. Active set algorithms operate around the same basic principle, in that constraints which are active at the current iteration are treated as equality constraints, and the remaining constraints are disregarded. The „free" variables are then solved using standard LS techniques. Once a minimum to this reduced problem is found, if the solution satisfies the constraints then the Karush-Kuhn-Tucker (KKT) conditions can be checked to test the optimality of the solution. If the solution does not satisfy the constraints, then one or more constraints are added into the „working set" and the next iteration is started. If the current solution satisfies the conditions of (9), then the solution is optimal and the search terminates; otherwise, an active constraint is removed from the working set and the next iteration starts. For the problem given in (8), if J is the value of the objective function at the current solution point, the KKT conditions reduce to checking the following conditions:

$$\begin{aligned} &\frac{\partial J}{\partial \Delta u_i} = 0, \quad \Delta u_{j\min} \leq \Delta u_j \leq \Delta u_{j\max} \\ &\frac{\partial J}{\partial \Delta u_i} \geq 0, \quad \Delta u_j = \Delta u_{j\min} \\ &\frac{\partial J}{\partial \Delta u_i} \leq 0, \quad \Delta u_j = \Delta u_{j\max} \end{aligned} \tag{17}$$

In other words, moving a variable away from its current position in a valid area of the solution space can only increase the value of the objective function. Given a candidate solution vector Δu", the vector of partial derivatives ΔJ required for checking (17) can be computed as $\Delta J = (G^T e) - C^{-1}\Delta u$", where C^{-1} is the inverse control matrix as described in the previous Section. Assuming one is willing to implement a full active set solver and solve up to $2n$-1 sets of linear equations at each sample step, then the active set algorithm developed by Schofield [17] is a good choice. However motivated by the need for implementation simplicity, in this paper an alternative simple iterative solution based upon coordinate descent will be suggested. For the problem described in (8), given an initial (feasible) solution Δu^0 and desired optimal solution Δu^*, one may proceed to iteratively generate a series of improved controls Δu^1, Δu^2, Δu^3, … which improve the objective function at each iteration. For the least squares problem defined as:

$$\begin{aligned} &C\Delta u = d \\ &where: C = \left(G^T G + diag(\lambda)\right)^{-1} \\ &d = G^T e \end{aligned} \tag{18}$$

A popular iterative scheme based upon coordinate descent is the Gauss-Siedel (GS) scheme [12]. GS proceeds by selecting the variable indices in a cyclic fashion and optimizing each variable (with the remaining variables assumed fixed) to

minimize the objective function. The minimization of the objective function is achieved by setting the gradient to zero. The gradient of (18) with respect to a single variable of index i is easily obtained as:

$$\frac{\partial J}{\Delta u_i} = \sum_{j=1}^{n} c_{ij}\Delta u_j - d_i \tag{19}$$

For C positive-definite, the sequence of controls thus obtained is guaranteed to converge, i.e. $\Delta u^k \to \Delta u^*$ as $k \to \infty$ [12][18], however it may be terminated at any point when a suitable convergence criteria is met or a pre-specified bound on the number of iterations is exceeded. Thus tight control over the worst-case execution time is obtained by appropriate choice of this bound. It has the advantage that for box constraints, after adjustment the variable under consideration may be simply „clipped" into its feasible box after solution [12]. Once the index of the variable with the lowest gradient that does not satisfy (17) is found, setting the left hand side of (19) to zero and „clipping" the result gives the simple update rule:

$$\Delta u_i = sat\left\{\frac{1}{c_{ii}}\cdot\left(d_i - \sum_{1\le j\le n \wedge j\ne i} c_{ij}\Delta u_j\right), \Delta u_i^{Min}, \Delta u_i^{Max}\right\} \tag{20}$$

Where $sat\{\}$ is the usual saturation function. Each inner iteration of the method requires computation of (20) requiring $O(n)$ steps, and the complexity of a full GS iteration is therefore $O(n^2)$ as each index must be cycled through. A suitable convergence criterion for a control application would be to iterate the method until the smallest change in a variable is less than some specified bound. In this paper, a simple but powerful adjustment to the described GS scheme is suggested to help speed up the identification of the active set and reduce the required number of iterations to converge upon a solution of specified accuracy. In this adjustment, the variable indices are not selected in a cyclic fashion per iteration, but are instead selected in an almost cyclic manner. At each step, the index of the variable whose absolute value changes the most following application of (20) is processed next. A scheme that carries this out method efficiently, such that each inner iteration costs no more than $O(n)$ steps as in the regular GS scheme, is given in pseudocode in Fig. 3 below. The method exploits the observation that the value of the summation term in (20), from a known point, can be updated following a change in a single variable in just n steps (step 5 in the code below). This routine can be easily programmed onto a microcomputer and does not require the use of specialized QP software. Note that saturation on line 1 of the code should be taken as element wise; also, the value of epsilon employed in line 5 of the code represents the required accuracy of the solution. Typically, since the required accuracy of the manipulated variables is limited by the DAC resolution in a computer-based control system, this dominates the criteria to be employed for convergence. For example, in a 16-bit DAC with +/- 10 VDC plant interface, the smallest change in an output is approximately 0.000305 V. Thus iteration until an epsilon below this level is achieved is sufficient.

1. Initial solution: compute $\Delta u = sat\{C^{-1}d, \Delta u_{min}, \Delta u_{max}\}$;
2. Initialization: compute, for all i, $1\le i\le n$: $\phi_i = \sum_{1\le j\le n \wedge j\ne i} c_{ij}\Delta u_j$;
3. Determine maximum: compute:

$$j = \arg\max_{1\le i\le n}\left[\Delta u_i - sat\left\{\frac{(d_i-\phi_i)}{c_{ii}}, \Delta u_i^{Min}, \Delta u_i^{Max}\right\}\right]$$

$$\Delta = \max_{1\le i\le n}\left[\Delta u_i - sat\left\{\frac{(d_i-\phi_i)}{c_{ii}}, \Delta u_i^{Min}, \Delta u_i^{Max}\right\}\right]$$

4. Update solution: compute $\Delta u_j = \Delta u_j + \Delta$;
5. Update phi: compute, for all i, $1\le i\le n$, $i\ne j$: $\phi_i = \phi_i + c_{ij}\Delta$;
6. Repeat: if $\Delta \ge \varepsilon$ and iteration limit not exceeded, goto step 3;

Fig. 3 Pseudo-code for constrained SPC_2 based upon coordinate descent.

As the objective function is convex, linear convergence for this almost cyclic coordinate descent scheme follows from the results in [18]; convergence depends upon the conditioning of the control matrix C. However, the condition number can be made arbitrarily low through appropriate choice of the vector of move suppression coefficient vector λ. In order to explore the potential improvement it may have over GS, a series of numerical tests were undertaken on a standard IBM PC using double-precision floating point math and C++ implementations of the proposed method and regular GS. Instances of dense, well-conditioned box-constrained least squares problems of dimension n between 2 and 10 were generated using the procedure outlined in [19]. Three instances for each value of n were generated and solved. The findings indicated that the proposed coordinate decent algorithm outperformed regular GS to compute solutions to an accuracy $\varepsilon = 10^{-4}$, reducing the required number of inner iterations by 20% in the average case and the average computation time by almost as much (19%). The GS method did not achieve any performance improvement in terms of less iteration over the proposed method in any trial. However due to the slight increase in the amount of inner loop operations per iteration, on a very small number of occasions (3) it outperformed the proposed method in terms of run-time, but never by more than 5%. Finally, note also that although the unconstrained controls are obtained with a simple vector/matrix product in SPC_2, a simple variation to the Simplex method could also be used to solve problem (8) through its Linear Complimentary Problem (LCP). Lemke"s algorithm can be used to achieve this, a good description is provided in [1].

5. Simulation-Based Example

In order to provide an initial study of the behavior of the proposed SPC_2 algorithm, a simple example is considered in this Section. The example is based upon a two-input, two-output (n=m=2) process described in [1], p.139. The Matlab® Simulink® Environment was employed to develop the process and controllers described in this Section.

5.1. Process Description and Model

The process consists of a stirred tank reactor as depicted in Fig. 4, where the manipulated variables are the main feed inflow rate u_1 and the jacket coolant inflow rate u_2. The two controlled outputs are the effluent concentration y_1 and the reactor temperature y_2.

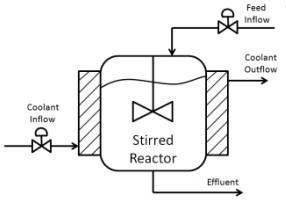

Fig. 4 Stirred tank reactor.

With the process time constants expressed in seconds, the transfer function matrix for this process can be obtained as:

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{1}{1+42s} & \dfrac{5}{1+18s} \\ \dfrac{1}{1+30s} & \dfrac{2}{1+24s} \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} \tag{21}$$

Assuming a ZOH on the inputs, the transfer function matrix for the process can be discretized with a sampling interval of 0.5 seconds as follows:

$$\begin{bmatrix} Y_1(z) \\ Y_2(z) \end{bmatrix} = \begin{bmatrix} \dfrac{0.01183}{z-0.9882} & \dfrac{0.1370}{z-0.9726} \\ \dfrac{0.0165}{z-0.9835} & \dfrac{0.0412}{z-0.9794} \end{bmatrix} \begin{bmatrix} U_1(z) \\ U_2(z) \end{bmatrix} \tag{22}$$

As described in Section 3.3, assuming that disturbances can be modeled as an integrated white noise sequence $w(k)$ a matrix fraction expansion allows (22) to be written in a direct I/O form allowing the optimal predictions of y_1 and y_2:

$$\begin{aligned}\Delta\hat{y}_1(k+l) &= 1.9608\Delta\hat{y}_1(k+l-1) - 0.9611\Delta\hat{y}_1(k+l-2) \\ &+ 0.0118\Delta u_1(k+l-1) - 0.0115\Delta u_1(k+l-2) \\ &+ 0.1370\Delta u_2(k+l-1) - 0.1354\Delta u_2(k+l-2)\end{aligned} \tag{23}$$

$$\begin{aligned}\Delta\hat{y}_2(k+l) &= 1.9629\Delta\hat{y}_1(k+l-1) - 0.9632\Delta\hat{y}_1(k+l-2) \\ &+ 0.0165\Delta u_1(k+l-1) - 0.0162\Delta u_1(k+l-2) \\ &+ 0.0412\Delta u_2(k+l-1) - 0.0406\Delta u_2(k+l-2)\end{aligned} \tag{24}$$

Where the optimal p-step ahead predictions of the outputs - and hence errors - may be obtained by recursion upon the equations above followed by integration, as detailed in Section 3.3.

5.2. Simulation Results

As an initial validation study, an unconstrained controller was designed and compared with dual PI controllers. Equations (23) and (24) were used to obtain the process predictions, and the proposed SPC_2 technique was then designed using $P = [10, 10]$ and $\lambda = [0.1, 0.5]$. Note that smaller values of p could have been employed without undue alteration of the results that follow. Although the process is highly interacting, to provide a comparative benchmark a baseline level of control can be achieved using continuous PI controllers. Analysis of the process relative gain array indicates that best results will be achieved by closing the control loops as $\{u_1, y_2\}$ and $\{u_2, y_1\}$ [19]. The direct synthesis (or „lambda-tuning") method was then employed for each PI loop individually, with a target time constant of $\lambda = 5$s [20]. In the simulations, which each lasted 200 seconds, an initial step change in setpoints to $r_1 = 6$ and $r_2 = 5$ was issued. After 120 seconds, r_2 was decreased to 4. In each case, a zero-mean band-limited white noise sequence with variance 0.005 and sampling time 0.05 seconds was injected into each process output; the same random number seeds were employed across both experiments. In order to measure the quality of the resulting control, the Integral of Squared Error (ISE) between the process outputs and the desired reference trajectory (with time constant 5 seconds) was also measured in each case. Figs. 5 and 6 show the results obtained for the dual PI and SPC_2 controllers, respectively. In the figures, the blue line shows the response of effluent concentration y_1 and red line that of the reactor temperature y_2.

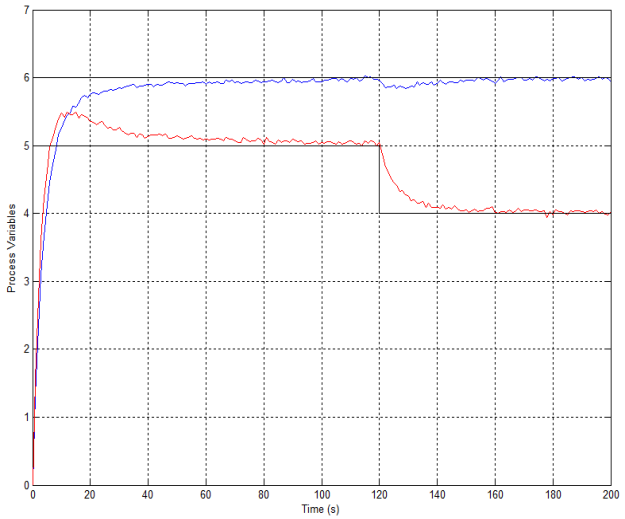

Fig. 5 Simulation results for double PI controllers.

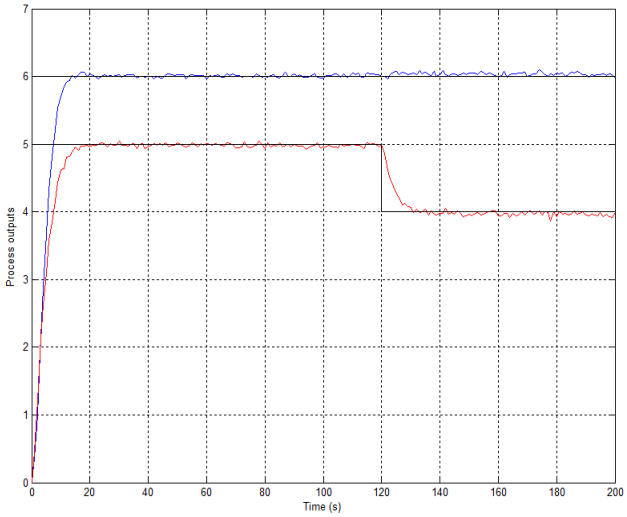

Fig. 6 Simulation results for SPC_2 controller with $\lambda = [0.1, 0.5]$.

From the figures obtained, it can be seen that the PI controllers take some time to settle to their corresponding setpoints (approximately 100 seconds in both cases), with undershoot in r_1 and overshoot in r_2. Following the change in r_2 after 120 seconds, the loop interaction is evident with a disturbance in r_1. Settling times in this case are of the order 60 seconds. The ISE measured for this case was 25.73. For the SPC_2 controller, it can be observed that the process variables make a smooth transition to their setpoints, and minimal loop interaction is evident, even following the change in r_2 after 120 seconds. Settling times are of the order 20 seconds in all cases. The ISE measured for this case was 5.83, giving a measure of the improved performance over the dual PI approach. In order to illustrate that in the SPC_2 approach, the link between the choice of P values and system robustness is weakened, a further experiment was carried out. The experiment was repeated again using $P = [10, 10]$ but with λ increased to [0.4, 2.0]. The simulation result is shown in Fig. 7, where again the blue line shows the response of effluent concentration y_1 and red line that of the reactor temperature y_2. A comparison between Fig. 6 and Fig. 7 clearly shows that as expected, the robustness and hence speed of response in the control loop may be altered by appropriate adjustment of λ, without the need to modify the P points. This is reflected in the increased ISE in this case, which was 31.19. However, comparison of the behaviors shown in Figs. 5 and 7, even the detuned SPC_2 approach seems preferable to the dual PI as the setpoints are reached more quickly, despite the overshoot; setting times of around 60 seconds are observed. Finally, note that in the original SPC approach, P would need to be set to ≥ 60 to achieve a similar response to that shown in Fig. 6. This increase has a marked effect on the quality of the predictions, and a corresponding increase in the ISA value.

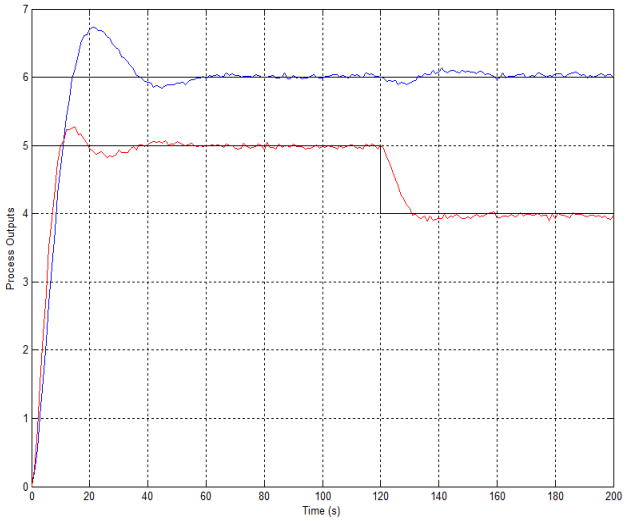

Fig. 7 Simulation results for SPC_2 controller with $\lambda = [0.4, 2]$.

5.3. Simulation Results: Constrained Control

As a second validation study, rate constraints of +/- 1.0 unit per 0.5 second were assumed to be present on the process manipulated variables u_1 and u_2. The proposed SPC_2 technique was then applied using the previous configuration of P = [10, 10] and λ = [0.1, 0.5]. Constraints were handled using the coordinate descent method described in the previous Section, with an epsilon of 10^{-4} employed. Figs. 8 and 9 display the obtained responses and the applied manipulated variables respectively. As may be seen in Fig.8 the response is actually very close to that of Fig. 7 (detuning of the controller), with an associated increase in ISE which was recorded as 35.88. Not more than 4 simple inner loop iterations were required to handle the constraints at each sample (in most cases when the constraints were violated, the saturated solution was quickly detected as optimal in 1 iteration). This gives an illustration that for the relatively small dimensions typical of many industrial multivariable control loops, the proposed optimization technique is computationally feasible as well as being trivial to implement.

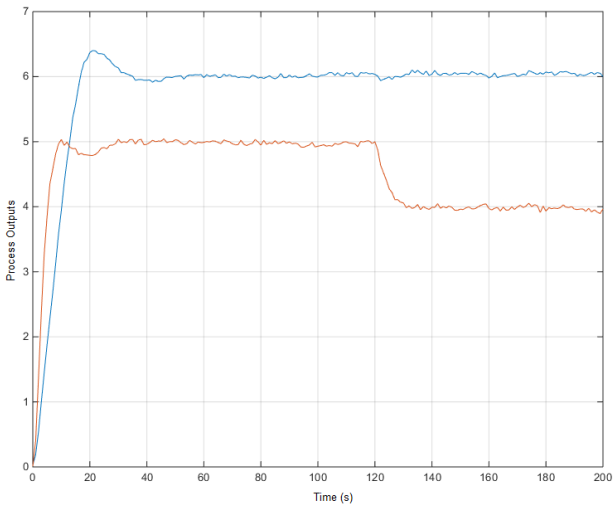

Fig. 8 Simulation results for constrained SPC_2 controller with λ = [0.1, 0.5].

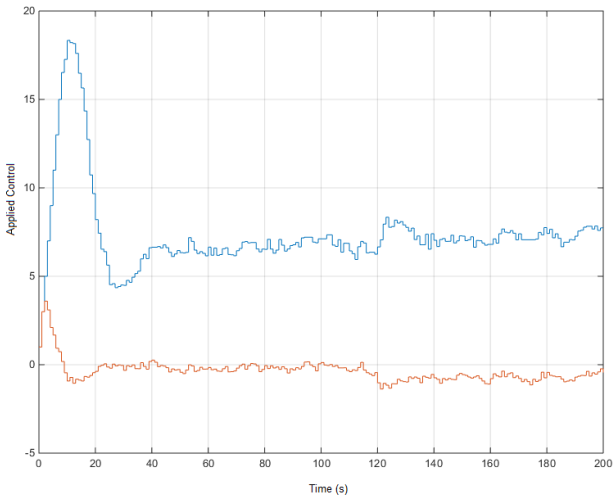

Fig. 9 Applied controls for the constrained SPC_2 controller simulation.

6. Conclusion

It has been argued in this paper that whilst the linear-programming approach to simplified multivariable MPC has many attractive features, it also has several drawbacks. The paper has then gone on to present the 2-norm approach to simplified multivariable MPC, and shown that it retains many of the beneficial features of the original whilst overcoming some of these drawbacks. Preliminary results have been described which provide evidence of the suitability of the proposed technique, and in conclusion it seems to be a useful adjunct to its linear counterpart. Future work will concentrate upon a more thorough experimental comparison between the linear and quadratic algorithms.

References

[1] E.F. Camacho and C. Bordons, Model Predictive Control: 2nd Edition. Springer-Verlag London, 2004.
[2] M. Morari and J. H. Lee, "Model predictive control: past, present and future," Computers and Chemical Engineering, vol. 23, no. 4/5, pp. 667-682, 1999.
[3] D.R. Saffer II and F.J. Doyle III, "Analysis of linear programming in model predictive control," Computers and Chemical Engineering, vol. 28, pp. 2749-2763, 2004.
[4] Y.P. Gupta, "A simplified predictive control approach for handling constraints through linear programming," Computers in Industry, vol. 21, no. 3, pp. 255-265, 1993.
[5] R.A. Abou-Jeyab, Y.P. Gupta, J.R. Gervais, P.A. Branchi and S.S. Woo, "Constrained multivariable control of a distillation column using a simplified model predictive control algorithm," Journal of Process Control, vol. 11, pp. 509-517.
[6] F. Zhao and Y.P. Gupta, "A simplified predictive control algorithm for disturbance rejection," ISA Transactions, vol. 44, pp. 187-198, 2005.
[7] M. Abu-Ayyad and R. Dubay, "Real-time comparison of a number of predictive controllers," ISA Transactions, vol. 46, pp. 411-418, 2007.
[8] G.C. Kember, R. Dubay and S.E. Mansour, "On simplified predictive control as a generalization of least-squares dynamic matrix control," ISA Transactions, vol. 44, 345-352.
[9] Y.P. Gupta, "Solution of low-dimensional constrained model predictive control problems," ISA Transactions, vol. 43, pp. 499-508.
[10] G.C. Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, Spinger-Verlag, New York, 2005.
[11] C.H. Papadimitriou and K. Stieglitz, Combinatorial Optimization: Algorithms and Complexity, Dover Publications Inc., England, 2000.
[12] A. Bjorck, Numerical Methods for Least Squares Problems, SIAM Publishing, Philadelphia, USA, 1996.
[13] G.H. Golub and C.F. Van Loan, Matrix Computations: 3rd edition, Baltimore: Johns Hopkins University Press, 1996.
[14] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, 1992.
[15] J. Sherman and W. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," Annals of Mathematical Statistics, vol. 21, no. 1, pp. 124-127, 1950.
[16] K.J. Astrom and B. Wittenmark, Adaptive Control: 2nd Edition, Addison Wesley, 1995.
[17] B. Schofield, "On Active Set Algorithms for Solving Bound-Constrained Least Squares Control Allocation Problems," In: Proceedings of the 2008 American Control Conference, Seattle, Washington, USA, June 2008.
[18] Z. Luo and P. Tseng, "On the linear convergence of descent methods for convex essentially smooth minimization," SIAM Journal of Control and Optimization, vol. 19, no. 3, pp. 368–400, 1992.
[19] M. Bierlaire, Ph.L. Toint and D. Tuyttens, "On iterative algorithms for linear least squares problems with bound constraints," Linear Algebra and its Applications, vol. 143, pp. 111–143, 1991.
[20] B. Roefel and B.H. Betlem, Advanced practical process control, Springer-Verlag, Berlin, 2004.
[21] D. Chen and D.E. Seborg, "PI/PID Controller Design Based on Direct Synthesis and Disturbance Rejection," Industrial Engineering Chemistry Research, vol. 41, no. 19, pp. 4807-4822, 2002.