2.17 Model Predictive Control and Optimization

W. K. WOJSZNIS (2005)

Software Products:
A. HIECON—Hierarchical constraint control
B. PFC—Predictive functional control
C. GLIDE—Identification package
D. DMCplus—Dynamic matrix control
E. DMCplus Model—Identification package
F. Aspen Target—Nonlinear model predictive control (MPC) package
G. Nova—NLC—Nonlinear controller
H. DeltaV Predict and DeltaV PredictPro—MPC controller and optimizer embedded into scalable DCS and integrated with identification and operator control applications
I. RMPCT—Robust model predictive control
J. Connoisseur—Control and identification package
K. Process Perfecter—Nonlinear controller
L. SMOC II—Shell multivariable optimizing control

Partial List of Suppliers:
Adersa—www.adersa.asso.fr/ (A, B, C)
AspenTech—www.aspentech.com/ (D, E, F)
DOT Products—www.dot-products.com/ (G)
Emerson Process Management—www.easydeltav.com/ (H)
Honeywell—www.honeywell.com/ (I)
Invensys—www.invensys.com/ (J)
Pavilion Technologies—www.pavtech.com/ (K)
Shell Global Solutions—www.shellglobalsolutions.com/ (L)

Costs:
The price for MPC usually based on: 1) Controller size, defined by the number of process inputs and outputs a given controller is capable of handling; 2) Number of controllers purchased; 3) Site wide or corporate licensing; 4) Run-time licenses vs. configuration licenses. In some cases the configuration tools are available at little or no cost; run-time licenses are always required; 5) Some DCS include embedded limited-size MPC software with no extra cost.

Model predictive control (MPC) was developed in the 1970s and 1980s to meet control challenges of refineries. Since then it has become the most effective advanced control technique for a wide range of industries. The advantages of MPC are most evident when it is used as a multivariable controller integrated with an optimizer.

The greatest MPC benefits are realized in applications with dead time dominance, interactions, constraints, and the need for optimization. Model predictive control takes into account the effect of past actions of manipulated and disturbance variables on the future profile of controlled and constraint variables and computes a sequence of controller moves to achieve the desired future profile of controlled variables.

The majority of MPC applications assume linear process models and apply linear programming (LP) for optimization. Techniques have been developed to also include nonlinear modeling and optimization.

The refining and petrochemical industries lead in MPC applications. MPC is also widely used in other industries including pulp and paper, chemicals, cement, power, food processing, automotive, metallurgy, and pharmaceuticals.

This section provides the basics of MPC operation and its theoretical background. It outlines also the following steps required for MPC implementation: process analysis and MPC configuration, process testing and model development, MPC simulation, and commissioning.
2.17 Model Predictive Control and Optimization

MODEL PREDICTIVE CONTROL PRINCIPLES

One way to understand MPC is to develop an analogy with a traditional feedback control loop (Figure 2.17a). As opposed to a traditional control loop, where the controller applies the difference (error) between the set point and the recent values of the measurement as its input, the predictive controller uses the difference between the future trajectory of the set point and the predicted trajectory of the controlled variable as its input. The difference is expressed not by a single value as in a traditional feedback loop, but as a vector of error values from target for a time period from the present to some set time in the future, usually defined to cover the settling time of the process.

Figure 2.17a shows an MPC controller for a process with two inputs and one output that allows one to see the analogy to a typical feedback control loop. The process has a manipulated variable (MV) and a disturbance variable (DV) on the input and a controlled variable (CV) on the output. A simple MPC controller used in this configuration has three basic components:

- A process model that predicts the process output in the future up to the prediction horizon (typically, 120 or more scans)
- A future trajectory of the set point for the same number of scans as the trajectory of the predicted process output
- A control algorithm for computing a control action based on the error vector as the difference between the future trajectories of the set point and the predicted process output

The controller output is an MV that is applied to the inputs of the process and the process model, which is a part of every MPC controller. A measured load upset to the process input is also applied as a DV to the model input. The process model computes a predicted trajectory of the CV that is the process output. After correction of this trajectory for any mismatch between the predicted and an actual measured value of the controlled variable, the predicted trajectory is subtracted from the future trajectory of the set point to form an error vector (Figure 2.17b) applied on the predictive controller input. The predictive control algorithm develops outputs to minimize the sum of squared errors over the prediction horizon, taking into account several future MV moves in its calculations.

MPC vs. Feedback Control Summary

The major differences between the operation of an MPC controller and a feedback controller are:

- A predicted error vector is applied to the MPC controller algorithm instead of the scalar values of recent errors that are used in a feedback controller.
- The error vector for an MPC controller is computed as the corrected model prediction subtracted from the future set-point values; whereas the error in a feedback
controller is the measurement subtracted from the set-point value.

- The increments in the MPC controller output required to bring the process output trajectory as close as possible to the set point trajectory are spread over several moves into the future over the control horizon. However, only the first move is implemented, and the computation procedure is corrected and repeated for the next scan.
- An MPC controller includes disturbances with the proper dynamics based on identified models from process step responses in the prediction of the process output.

The operation of an MPC controller relies on a good process model. If the model is accurate, the MPC controller can demonstrate superior performance for processes with long dead times, inverse responses, and higher order dynamics that are difficult to control with a classical PID feedback controller.

For multivariable processes, the MPC controller takes into account process interactions. Additionally, the MPC controller handles constraints on the process output and the manipulated variables. Finally, MPC controllers are set up to integrate and manage optimization. Taken together, these capabilities make MPC the right choice for incorporating process knowledge and solving complex control and optimization problems on the unit operation control level.

**PROCESS MODELING**

A generic multivariable process controlled by MPC is presented as a black box in Figure 2.17c. In MPC, the process inputs are MVs and measured DVs. The process outputs are CVs and auxiliary or constraint variables (AVs).

Manipulated variables are managed by MPC controller outputs, predominantly by the manipulation of set points of the fast loops that stabilize the inputs to the process. In calculating the optimal solution, an MPC controller assumes in its calculations that it can make several future moves of the manipulated variables. Only the first move is implemented. The procedure for an optimal solution is repeated every scan. The control horizon is the number of future manipulated variable moves that are taken into account in developing the optimal MPC solution.

Manipulated variables have hard constraints, which means that no violations of the limits are allowed at any time. Both absolute and rate or incremental constraints are set for manipulated variables.

Measured disturbances are also inputs to the process; however, they are not managed by MPC.

 Controlled variables are process outputs kept at specific set points (targets) or within specified ranges. Process models predict the future values of controlled variables for a number of scans ahead. The prediction horizon is the range of process output prediction in scans.

Finally, constraint variables are a type of controlled variables with only range control and no set points. Constraint variables have so-called soft constraints. Violation of this type of constraint may occur temporarily to satisfy other criteria or constraints of the system.

Some use the term controlled variables both for controlled and constraint variables, differentiating them by using the terms controlled variables with set points for controlled variables and controlled variables with limits for constraint variables.

The process model is the basis for MPC technology. Most MPC implementations to date use step response models, which provide an explicitly available future prediction of process outputs. The future prediction is used to compute the predicted error vector as an input to the MPC controller.

The step response is represented by a number of coefficients (40 coefficients in our example). Every coefficient corresponds to the value of the model step response at a specific time instance (scan) in the future (Figure 2.17d). In MPC terms, the step response is a prediction of the process output up to the prediction horizon, for a unit step input that was applied at scan zero.

An MPC controller uses an incremental model. It means that real process input and output values are assigned to the model at model initialization. Later on, increments of the
inputs are accounted for and increments of the model outputs are calculated.

**Process Modeling Equations**

MPC application assumes a linear process model. Mathematically we can consider a predicted output trajectory of a process as a process state and use a modified state space form for process modeling. For single-input, single-output (SISO) process prediction, the equations are in the form:

\[
x_{k} = Ax_{k-1} + b\Delta u_{k} + fw_{k}
\]

\[
y_{0} = Cx_{k-1}
\]

where \(x_{k-1} = [y_{0}, y_{1}, \ldots, y_{r}, y_{p+1}, \ldots, y_{p+1}]^{T}\) is the vector of process output prediction at a time \(k - 1, 0, 1, 2, \ldots, p + 1\) steps ahead.

Matrix \(A\) is the shift operator defined for a self-regulating process as:

\[
Ax_{k-1} = [y_{1}, y_{2}, \ldots, y_{r}, y_{p+1}]^{T}
\]

\(b = [b_{0}, b_{1}, \ldots, b_{p}]^{T}\) is the vector of \(p\) step response coefficients.

\(\Delta u_{i} = u_{k} - u_{k-1}\) is the change in the process input/controller output.

\(w_{k} = y^{p} - y^{m} = y^{p} - y_{0}\) is the process output measurement minus the model output (the mismatch between the process and the model that results from the noise, unmeasured disturbances, and model inaccuracy).

\(f\) is the \(p\) dimension filter vector with unity default values.

Matrix \(C\) is the operator for selecting the current model output defined as \(y_{i} = Cx_{i}.\)

The MPC controller updates prediction and control calculations every scan. This procedure is known as *receding horizon control*.

For \(n\) outputs and \(m\) inputs process, vector \(x_{i}\) has dimension \(n \times p\) and vector \(b\) becomes a matrix \(B\) with dimension \(n \times p\) rows and \(m\) columns. The graphical illustration of the equations in Figure 2.17e explains the prediction principles.

At any time instance \(k\), the process output prediction (bottom curve) is updated in three steps:

1. The prediction made at the time \(k - 1\) (the bottom dotted curve) is shifted one scan to the left.
2. The prediction curve is moved to the point to match the current measured process output, for filter coefficient \(= 1\), or in general, prediction shift is \(fw_{i}\).
3. A step response, scaled by the current change on the process input, is added to the output prediction.

**PROCESS MODEL IDENTIFICATION**

MPC is used mainly for multivariable and highly interactive processes. Processes are tested by applying a special pulse test sequence instead of a single step. Then, a process model is built from process test data using mathematics.

Using test data, we can build equations for every output that equal the number of collected samples. The number of collected samples in test data is normally significantly higher than the number of unknown model coefficients. Such equations are solved using the least squares technique. This technique finds coefficients that may not fit perfectly in any equation but fit optimally for all equations, in such a way that the total squared error for all equations is minimal.

**FIR and ARX Modeling**

The form of the equations used for process modeling is defined by the identification modeling technique. There are a number of identification techniques used. The most common identification techniques are Finite Impulse Response (FIR) and Auto Regressive with eXternal inputs (ARX). FIR identifies pulse response coefficients, as in Equation 2.17(2) for a SISO process:

\[
\Delta y_{k} = \sum_{i=1}^{p} h_{i} \Delta u_{k-i}
\]

where \(p\) is prediction horizon, with a typical default value for MPC model 120; \(\Delta y_{k}\) is change in the process output at the time \(k\); \(\Delta u_{k-i}\) is change in the process input at the time \(k - i\); and \(h_{i}\) is the pulse response coefficient of the model.

Step response coefficients for the MPC controller are calculated directly from the pulse response as follows:

\[
b_{i} = \sum_{j=1}^{i} h_{j} k = 1, 2, \ldots, p
\]

An advantage of FIR is that it does not require any preliminary knowledge about the process. However, identifying step responses with full prediction horizon with as many as 120 or more coefficients results in low confidence levels of identified coefficient values. Therefore, a shorter horizon with about 60 points is more suitable for a FIR model.
A FIR model with a shorter horizon provides the initial part of the step response that is adequate for defining process dead times using a heuristic approach. On the other hand, the ARX model in Equation 2.17(4) has fewer coefficients, which are defined with higher confidence, provided the process dead times are known.

\[ y_k = \sum_{i=1}^{a} a_i y_{k-i} + \sum_{j=1}^{v} b_j u_{k-d-j} \quad \text{for SISO process} \quad 2.17(4) \]

where \(a, v\) are autoregressive and moving average equation orders of ARX; \(a = 4, v = 4\) satisfy most applications; \(a_i, b_i\) are moving average and autoregressive coefficients of the ARX model; and \(d\) is dead time in scans.

Applying FIR first and defining dead times and then applying those dead times for ARX provide the best identification results. Step responses for the MPC controller with any prediction horizon are calculated directly from Equation 2.17(4).

For a MIMO process, superposition is applied from all inputs to every output both in FIR and ARX models. An identified model should be validated by applying real input data to the model inputs and comparing the model output with real output. An example of a validation plot is shown in Figure 2.17f.

The validation procedure also may include statistical techniques for calculating confidence intervals. Confidence intervals form an area around nominally defined step responses. The true value of the step response coefficients is found within confidence intervals with a predefined confidence (or probability). A 95% confidence interval is normally used.

The complete process model overview is presented in matrix form as in Figure 2.17g.

**MODEL PREDICTIVE CONTROLLER**

Dynamic matrix control (DMC) has historically been the most successful MPC implementation approach, based on a dynamic matrix. A dynamic matrix is used for developing an MPC controller. A dynamic matrix is built from step responses to predict the changes in the process outputs that result from moves of the manipulated variables over the control horizon. Dynamic matrix \( S \) as in Equation 2.17(5), calculates prediction vector \( \Delta x \) resulting from \( c \) future moves of MV, defined by the vector \( \Delta u(k) \).

\[
\Delta x = S \Delta u(k) = \begin{bmatrix} b_0 & 0 & \cdots & 0 \\ b_1 & b_0 & 0 & 0 \\ b_2 & b_1 & b_0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ b_1 & b_{i-1} & b_2 & b_{i-c+1} \\ \vdots & \vdots & \vdots & \vdots \\ b_{p-1} & b_{p-2} & b_{p-3} & b_p \end{bmatrix} \begin{bmatrix} \Delta u_e \\ \Delta u_{e+1} \\ \Delta u_{e+2} \\ \vdots \\ \Delta u_{e+c-1} \\ \vdots \\ \Delta u_{p-1} \end{bmatrix} = \begin{bmatrix} \Delta y_0 \\ \Delta y_1 \\ \Delta y_2 \\ \vdots \\ \Delta y_{1-c} \\ \vdots \\ \Delta y_{p-c} \end{bmatrix} 2.17(5)
\]

**MPC Controller Formulation**

A general formulation of the MPC controller includes minimization of both the squared sum of predicted control error and the squared sum of calculated controller moves.

---

**FIG. 2.17f**
Validation of identified model. (Courtesy of Emerson Process Management.)
To get the desired MPC controller performance, two tuning parameters called penalty on moves and penalty on error are used in the formulation.

The MPC controller objective for minimizing the squared error of the controlled variable includes the penalty on error over the prediction horizon. The objective for minimizing the squared changes in the controller output over the control horizon includes the penalty on moves in the following way:

\[
\min_{\Delta MV(k)} \left\{ \| \Gamma^y [CV(k) - R(k)] \|^2 + \| \Gamma^u \Delta MV(k) \|^2 \right\} \tag{2.17(6)}
\]

where CV(\(k\)) is the controlled output \(p\)-step ahead prediction vector; \(R(k)\) is the \(p\)-step ahead reference trajectory (set point) vector; \(\Delta MV(k)\) is the \(c\)-step ahead incremental control moves vector; \(\Gamma^y\) is a diagonal penalty matrix on the controlled output error; \(\Gamma^u\) is a diagonal penalty matrix on the control moves; \(p\) is the prediction horizon (number of scans); and \(c\) is the control horizon (number of scans).

**MPC Controller Equations**

The solution for the process with dynamic matrix \(S^u\) satisfying Equation 2.17 is in the form:

\[
\Delta MV(k) = (S^u HT \Gamma^u S^u + \Gamma^u HT \Gamma^u)^{-1} S^u HT \Gamma^u E_y(k) \tag{2.17(7)}
\]

where \(S^u\) is the \(p \times c\) process dynamic matrix built from the step responses of dimension \(p \times c\) for a SISO model and \(pn \times cm\) for a MIMO model with \(m\) manipulated inputs and \(n\) controlled outputs; and \(E_y(k)\) is the error vector over prediction horizon.

The performance of the control algorithm is modified by the adjustable parameters: \(p\), \(c\), \(\Gamma^u\), and \(\Gamma^y\). From an implementation point of view, it is inconvenient to use \(p\) and \(c\) as tuning parameters. \(\Gamma^u\) is a basic controller tuning parameter at the controller generation phase. Increasing \(\Gamma^u\) elements makes control less aggressive, and while decreasing \(\Gamma^u\) makes the control action more aggressive and the control response faster. It follows from experience that the dead time should be accounted for as a major factor in setting the penalty on moves.

The reference trajectory is applied for online tuning. One trajectory design acts in such a way that instead of penalizing any departure from the trajectory, only those deviations that are below the trajectory or above the set point value are penalized (area A and area C if range \(= 0\), Figure 2.17h). In addition, the control error is considered zero if the controlled variable is within range (area C if range > 0).
Presenting MPC control equation in the form

$$\Delta MV(k) = K_{mpe}E_j(k) \quad 2.17(8)$$

where

$$K_{mpe} = (S^\ast \Gamma \gamma \Gamma \gamma S^\ast + \Gamma \ast \Gamma \ast)^{-1}S^\ast \Gamma \gamma \Gamma \gamma$$  2.17(9)

is the MPC controller gain, one can see that MPC uses integral action. This, however, does not mean that there is a direct analogy between an MPC controller and a conventional, integral-only feedback controller. The major difference is that the MPC controller uses predicted errors and integrates over the prediction horizon.

INTEGRATING MPC, CONSTRAINTS, AND OPTIMIZATION

A typical MPC configuration includes both controlled variables and constrained variables. The unconstrained MPC controller does not manage constrained variables directly. A supervisory constraint-handling algorithm or optimizer performs the task.

The objectives of optimization are usually to maximize a product value and minimize a raw material cost. For processes with an objective function dependent on several manipulated or controlled variables, optimization techniques are essential components of model predictive control technology. The proven optimization technique is linear programming (LP) with steady-state models.

Linear programming is a mathematical technique for solving a set of linear equations and inequalities in order to maximize or minimize an additional function called an objective function. Usually, objective functions express economic value, such as cost or profit.

MPC optimization uses incremental values of MV at the present time or the sum of increments of MV over the control horizon and incremental values of CV at the end of the prediction horizon instead of positional current values, as in typical LP applications. With the prediction horizon normally used in MPC, this assumption guarantees a future steady state for self-regulating processes and perfectly satisfies requirements for proper optimizer operation.

Optimization Equations

The LP technique uses steady-state models and therefore a steady-state condition is required for its application. The basic steady-state process equation in the incremental form is:

$$\Delta CV(t + p) = A \Delta MV(t + c) \quad 2.17(10)$$

where

$$\Delta CV(t + p) = \begin{bmatrix} \Delta CV_1 \\ \vdots \\ \Delta CV_n \end{bmatrix}$$

changes in controlled and constraint variables up to the end of prediction horizon

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}$$

process steady-state gains matrix

$$\Delta MV(t + c) = \begin{bmatrix} \Delta MV_1 \\ \vdots \\ \Delta MV_n \end{bmatrix}$$

changes in manipulating variables to achieve the desired state at the end of control horizon.

\Delta MV change should satisfy limits both on MV and CV.

$$MV_{MIN} \leq MV_{CURRENT} + \Delta MV(t + c) \leq MV_{MAX} \quad 2.17(11)$$

$$CV_{MIN} \leq CV_{PREDICTED} + \Delta CV(t + p) \leq CV_{MAX} \quad 2.17(12)$$

The objective function for both maximizing output product value and minimizing input raw material cost is defined in the following way:

$$Q_{\text{min}} = -UCV^T \Delta CV(t + p) + UMV^T \Delta MV(t + c) \quad 2.17(13)$$

where UCV is the cost vector for a CV change of unit value and UMV is the cost vector for an MV change of unit value. Applying 2.17(10), we can express the objective function through MV only:

$$Q_{\text{min}} = -UCV^T A \Delta MV(t + c) + UMV^T \Delta MV(t + c) \quad 2.17(14)$$

The LP solution is always located at one of the vertexes of the region of feasible solutions. An illustration for a two-dimensional problem explains the point (Figure 2.17i).

The region of feasible solutions for two controlled variables and two manipulated variables is an area contained within MV1 and MV2 limits, represented by vertical and horizontal lines, and CV1 and CV2 limits, represented by straight lines (see Equations 2.17[15] and 2.17[16]):

$$CV_{1MIN} = a_{11} MV1 + a_{12} MV2 \quad CV_{1MAX} = a_{11} MV1 + a_{12} MV2 \quad 2.17(15)$$

$$CV_{2MIN} = a_{21} MV1 + a_{22} MV2 \quad CV_{2MAX} = a_{21} MV1 + a_{22} MV2 \quad 2.17(16)$$
2.17 Model Predictive Control and Optimization

The optimal solution is located at one of the vertexes marked by arrows. To find this solution, an LP algorithm calculates an objective function for an arbitrary vertex and improves the solution sequentially until the vertex with the maximum (or minimum) value of the objective function as an optimal solution is found.

**MPC Optimal Controller**

Optimal MV values are applied to the MPC algorithm as target values of MV to be achieved within the control horizon. If the MPC controller is squared, i.e., the number of MV is equal to the number of CV, then MV targets can be effectively achieved by a change in CV value.

\[
\Delta CV_T = A^* \Delta MV_T 
\]

where \( \Delta MV_T \) is the optimal target change of MVs and \( \Delta CV_T \) is the CVs target change to achieve the optimal MV. The CVs change is implemented by managing CV set points.

The MPC algorithm working with an optimizer therefore has two main objectives:

- Minimize CV control error with minimal MV moves within operational constraints.
- Achieve optimal steady-state MV values set up by the optimizer and target CV values calculated directly from MV values.

To satisfy these objectives, the original unconstrained MPC algorithm has been extended to include MV targets into the least square solution. The objective function for this MPC controller is

\[
\min_{\Delta MV_T(v)} \left\{ \|C^*[CV(k) - R(k)]\|^2 + \|C^* \Delta MV(k)\|^2 \right. \\
+ \left. \|C^* \sum \Delta MV(k) - \Delta MV_T\|^2 \right\} 
\]

where \( C^* \) is a penalty on error of the sum of controller output moves over control horizon relative to the target optimal change of MV defined by the optimizer and \( \sum \Delta MV(k) \) is the sum of the MV moves over control horizon. For simplicity of notation, the objective function is shown for the SISO control.

The first two terms are the objective function for the unconstrained MPC controller. The third term sets up an additional condition to make the sum of the controller output moves equal to the optimal targets. In other words, the first two terms set up objectives for controller dynamic operation, and the third term sets up steady-state optimization objectives. Graphically optimized MPC objectives are illustrated in Figure 2.17j.

The general solution for this controller, similar to that for the unconstrained MPC controller, can be expressed as

\[
\Delta MV(k) = (S^T C^T G^T S^* + C^* C^*)^{-1} S^T C^T G E_p(k) \\
= K_{\text{MPC}} E_p(k) 
\]

where \( \Delta MV(k) \) is the change in MPC controller output at the time \( k \) and \( K_{\text{MPC}} \) is the optimized MPC controller gain.
Matrix $\Gamma$ combines matrices $\Gamma^u$ and $\Gamma^o$. It is a square matrix of dimension $(p + 1)$ for the SISO controller and $\lfloor n(p + m) \rfloor$ for the multivariable controller. Superscript $T$ denotes a transposed matrix.

**MPC and Optimizer Integration**

In operation, the optimizer sets up and updates the steady-state targets for the MPC unconstrained controller at every scan; thus, the MPC controller executes the unconstrained algorithm. Since the targets are set in a manner that accounts for constraints as long as a feasible solution exists, the controller works within the constraint limits. Optimization, therefore, is an integral part of the MPC controller. The integrated MPC controller and optimizer performs the following sequence of operations each scan:

1. Update CV and DV measurements.
2. Update CV predictions.
3. Determine optimal MV steady-state targets and calculate CV targets.
4. Provide MV outputs accounting for CV and MV targets.
5. Update MV.

Figure 2.17k illustrates communication and integrated operation of the optimizer and MPC algorithm. There is a typical situation when the optimizer cannot find an optimal solution because of too tight constraint limits, too many set points, or too severe disturbances. In these cases, the system constraints are relaxed by changing set points to range control and/or abandoning some constraints. This could be done by minimizing the squared error over a number of constraints or by abandoning constraints with the lowest priority sequentially.

Ill conditioning is another typical problem an optimizer has to deal with. As with the MPC algorithm, ill conditioning manifests itself as excessive changes in calculated MV targets even for minor corrections of constraints. Ill conditioning is removed dynamically by changing the configuration of active constraints (abandoning some constraints) or by removing the association between constraint or controlled variables and manipulated variables that have excessive moves.

**MPC APPLICATION DEVELOPMENT**

Figure 2.17l illustrates a typical procedure of MPC application development consisting of the following basic steps:

1. Process analysis
2. MPC configuration development
3. Process testing
4. Process model development and controller generation
5. MPC simulation and tuning validation
6. MPC control evaluation and tuning adjustment

Process analysis should deliver a clear formulation of process control objectives and limitations as well as an understanding of how MPC control can achieve those objectives. Process analysis should result in MPC configuration. The process inputs and outputs are grouped into four different categories based on how they are utilized in the control of a process.

---

**FIG. 2.17k**

MPC and optimizer integration. (Copyright © 2002, ISA—The Instrumentation, Systems, and Automation Society. All rights reserved. Used with permission of ISA—The Instrumentation, Systems, and Automation Society.)

**FIG. 2.17l**

A typical procedure for MPC application. (Copyright © 2002, ISA—The Instrumentation, Systems, and Automation Society. All rights reserved. Used with permission of ISA—The Instrumentation, Systems, and Automation Society.)
The most common process testing procedure used today is pseudo random binary sequence (PRBS).

Process model development consists of test data review and model generation and validation. Controller generation computes the controller matrix defined by the process model and controller tuning parameters: penalty on move and penalty on error.

Penalty on move—Control sensitivity to changes in a process is determined by the controller robustness. The parameter used in controller generation that most impacts robustness is the penalty-on-move (PM) parameter. The PM defines how much the MPC controller penalizes changes in the manipulated output (MV). The penalty-on-move parameter is typically defined independently for every MV. High PM values result in a slow controller with a wide stability margin. With such settings, the control is relatively insensitive to a change in either the process or model errors. Low PM values result in a fast controller with a narrow stability margin. The penalty on move is also known as “move suppression” or an MV tuning weight.

Penalty on error—The penalty-on-error (PE) factor allows more importance to be placed on a specific controlled variable. In some systems an “equal concern” factor is used to account for differences in scale spans and engineering units, where a smaller equal concern places a greater importance on the error. The penalty on error is also known as a CV or AV tuning weight.

Testing in simulation—The user can influence the control behavior and robustness in online operation. A simulation environment permits operation in manual and automatic mode and allows for the introduction of noise and disturbance inputs and the acceleration of control execution.

Commissioning MPC Application

Before the new control is tried on the real process, the user must observe how well the control responds to set-point changes and load disturbances in simulation. If the process responds slowly to input changes, then it may take many hours in an operating plant to see the full control response. In a simulated environment, the process and control simulation can support faster than real-time execution to allow the response of even the slowest processes to be seen in a few minutes. Similarly, the ability to simulate the control and process response slower than real time is valuable when working with a very fast process. The ability to adjust the speed of execution allows the control response to be verified quickly for a variety of operating conditions. Through such testing, the user can gain confidence in the control performance.

In the commissioning phase of the MPC application, the user must first perform the obvious step of checking all communications between the process instrumentation and the MPC controller. Once the controller is operating in a stable manner, the performance elements that must be checked out during the commissioning phase include:

- Set-point changes
- Constraint control
- Steady-state giveaway
- Manipulated variable limitations
- Feedforward disturbances handling
- Optimizer functionality

Many issues of MPC application development have been addressed in modern MPC design. To provide a consistent environment for a configuration definition, MPC is implemented as a function block, and a simulation environment is automatically created based on the block definition, the identified process model, and the controller generated.

Once the controller has been generated, the MPC block is downloaded to the DCS controller to place the control online. Predefined applications are provided to allow each control, constraint, manipulated, or disturbance input to be placed on operator graphics. The trend display shows past plots and future predicted process outputs.

The tuning parameters associated with the generation of the MPC controller are automatically set based on the process environment permits operation in manual and automatic mode and allows for the introduction of noise and disturbance inputs and the acceleration of control execution.

CONCLUSIONS

Model predictive control and optimization are the primary techniques for achieving high performance unit operations. Modern MPC products, especially those integrated with DCS, are easy to apply and use. Good process understanding, however, is a major factor in setting the control objectives and in designing and commissioning an MPC application.

Bibliography


**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARX</td>
<td>Auto regressive with eXternal inputs (model)</td>
</tr>
<tr>
<td>AV</td>
<td>Auxiliary (constraint) variable</td>
</tr>
<tr>
<td>CV</td>
<td>Controlled variable</td>
</tr>
<tr>
<td>DCS</td>
<td>Distributed control system</td>
</tr>
<tr>
<td>DMC</td>
<td>Dynamic matrix control</td>
</tr>
<tr>
<td>DV</td>
<td>Disturbance variable</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite impulse response (model)</td>
</tr>
<tr>
<td>LP</td>
<td>Linear programming</td>
</tr>
<tr>
<td>MIMO</td>
<td>Multiple-input multiple-output</td>
</tr>
<tr>
<td>MPC</td>
<td>Model predictive control</td>
</tr>
<tr>
<td>MV</td>
<td>Manipulated variable</td>
</tr>
<tr>
<td>PE</td>
<td>Penalty on error</td>
</tr>
<tr>
<td>PM</td>
<td>Penalty on moves</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional, integral, derivative (control)</td>
</tr>
<tr>
<td>PRBS</td>
<td>Pseudo random binary sequence</td>
</tr>
</tbody>
</table>

**Definitions**

**Control horizon** The number of future manipulated variable moves that are taken into account in developing the optimal MPC solution.

**Dynamic matrix** A matrix built from step responses to predict the changes in the process output that results from the moves of the manipulated variable over the control horizon.

**Linear programming** A mathematical technique for solving a set of linear equations and inequalities in order to maximize or minimize an additional function called an objective function.

**Model predictive control (MPC)** A model-based control technique that uses process output prediction and calculates several consecutive controller moves in order to satisfy control objectives.

**Moving horizon control** See Receding horizon control.

**Penalty on error** A parameter that ranks the importance of a specific controlled variable.

**Penalty on move** A parameter that defines how much the MPC controller penalizes changes in the manipulated variable.

**Prediction horizon** The range of process output prediction defined in scans.

**Receding horizon control** Control that is based on updating every scan output prediction over the prediction horizon.

**Reference trajectory** The future desired projection of the process output.

**Set point trajectory** See Reference trajectory.