Nonlinear Model Predictive Control Using First-Principles Models

R. Russell Rhinehart

Oklahoma State University, Stillwater, OK 74074, USA
russ@r3eda.com, rrr@okstate.edu

Abstract. The common attribute in horizon predictive control is the use of a dynamic model of the process and optimization to plan a sequence of future control actions to best make the model match a desired future path, while avoiding constraints. This paper describes a structure to use first-principles process models in horizon predictive control, which contrasts the conventional use of linear empirical models derived from step-testing the process.

Keywords: Process Control, Nonlinear, Horizon Predictive.

1 Introduction

Model Predictive Control (MPC) has many alternate names and acronyms. Some are DMC (Dynamic Matrix Control), HPC (Horizon Predictive Control), IdCom (Identify and Command), and APC (Advanced Process Control). The common attribute is that they use a dynamic model of the process and use optimization to plan a sequence of future control actions to best make the model match a desired future path, while avoiding constraints. The first control action (manipulated variable value, MV), in the plan is implemented. The entire plan is not executed, just the first step. Then after one control interval, the future MV sequence plan is recalculated to accommodate changes in set points, disturbances, and process-model mismatch. And, again, the entire plan is not executed, just the first step.

The models in MPC can reveal both disturbance and interaction, as well as deadtime and other ill-behaved dynamics. The optimization in MPC compensates for the model features, and also has override action and can choose best MVs to manage the process when there are extra options. Although such features could be implemented in classic ARC (Advanced Regulatory Control) with feedforward, decouplers, ratio, override, etc.; once an implementation exceeds about 2 inputs and 2 outputs, the complexity and number of tuning and model coefficients becomes very complicated to manage. MPC provides a unified structure, with only one tuning coefficient for each Controlled Variable (CV).

A rule of thumb is that MPC (or any control strategy with equivalent features) will halve the CV deviations associated with conventional regulatory control. This reduction in variance permits assigning set points that are closer to specification limits, which translates to any of many economic benefits – higher throughput, less waste, lower energy and raw material use, etc. Although relatively expensive to license an MPC product, and although the process step testing to develop empirical models takes time and moderately upsets the process, MPC has become accepted as a best control approach in continuous chemical processing.

Normally, the MPC models of the process response are Finite Impulse Response (FIR) models, a vector of normalized responses to the steps in both control action and disturbances, a linear response. Alternately, models could be second-order plus dead-time (SOPDT), also linear. And commonly, the optimization used to plan a future sequence of Manipulated Variable (MV) moves is Linear Programming (LP). Although, many commercial products offer empirical models with nonlinear gain (such as neural networks based on steady state responses, or even heuristically set gains from interviewing operators and engineers), most models remain linear in the dynamics (lags and delays). Because LP seeks to place the control action on a constraint, an extreme boundary, and requires linear objective function and constraint models, many products use alternate optimizers, such as Successive Quadratic.

One issue is that chemical processes are nonlinear and non-stationary. Both gains (sensitivities and interactions) and dynamics (time-constants and delays) change with operating conditions, product specification, internal tank levels, piping reconfigurations, equipment switching, etc. When the process operating conditions or process attributes change, linear models become significantly mismatched to the process and require model recalibration (recurring process step testing). It would be desirable to use models that remain consistent with the process.

Another issue is that the mathematics and style of the classic controller models are substantially different from the phenomenological models that process engineers commonly use for process design, analysis, trouble shooting, constraint forecasting, optimization, and training. The mathematics of the linear models obscures process understanding. It would be desirable if the models in control were consistent with those used in all the process management actions.

This article is about the use of first-principles models in MPC. Since the models are nonlinear, it is Nonlinear Model-Predictive Control (NMPC).

Phenomenological models are mechanistic, cause-and-effect models. For process control, start by applying fundamental mass, momentum, and energy balances on a control volume, to get a model of an element of the process. Then integrate (sequentially combine) the individual models to describe the unit. If the models are elementary in their description of the process, if they use ideal relationships, they are termed first-principles models. If the models seek to fully describe all phenomena, they are rigorous models. First-principles models are characteristic of undergraduate courses, product bulletins, training courses, etc. First-principles models are relatively easy to develop, understand, and adapt; and are fully adequate for control.

Since about 1980, there have been many approaches to nonlinear process control. It seems that the industrial acceptance of digital control devices and distributed control systems (DCS) of the 70s, revealed a vision to many investigators that model-based control could be practicable. There have been many concepts as to how best to implement it. For examples in increasing complexity: Generic Process Control (GMC) provides a structure to use steady state models in an output characterization of a PI controller. If the process is SISO and the problem is nonlinear gains, this is an excellent solution. Process Model-Based Control (PMBC) is a single step-ahead controller (it does not forecast a future plan or handle ill-behaved dynamics), but can handle MIMO processes with constraints, disturbances, and nonlinear and non-stationary features. It was developed to adapt the model to changing process attributes such as catalyst reactivity, tray efficiency, heat exchanger fouling, drag coefficients, etc. If delay, high-order response, or inverse action is strong, PMBC would not be a good solution. Predictive Functional Control (PFC) was developed to handle both nonlinearity and ill-behaved dynamics (delays, high order, inverse action). It uses a process model to calculate a step-and-hold MV action, if implemented now, that would make the model match its set point at some future time (the coincidence point) well past the influence of the ill-behaved dynamics. These are all practical, industrially-proven techniques. Complying with the K.I.S.S. principle (Keep it Simple and Safe) that guides applications, those strategies should be considered prior to the topic of this article, first-principles models for multivariable, constraint handling, model-predictive control.

2 What Are First-Principles Models?

First-Principles models are the process engineer’s models which are used for process design, analysis, optimization, data reconciliation, and trouble shooting. They are based on elementary material and energy balances and ideal constitutive relations, and are characteristic of college classroom teaching models for heat exchange, fluid flow, reaction engineering, separations, etc. Representing process mechanisms, they are typically nonlinear.

Alternately, one could use rigorous models attempting to perfectly explain every nuance. But balancing sufficiency with perfection, first-principles models are fully adequate

2.1 How Can Elementary Models be Adequate for Control?

Contrasting modeling perfection, consider that humans can catch a fly ball or steer a car through a curve, without rigorous models, with just intuitive continual correction. And consider that PI controllers, based on linear, stationary, simplistic (FOPDT), inexact models, are also adequate for control. Here is a qualitative explanation: If there are 30 or so control intervals during a process transient, then even if the model is only 70% correct, leaving a 30% error in its first action, at the next control action, the remaining error is 70% corrected, leaving a 0.3*0.3=0.09 error. By the time a few control actions have been taken, the error is smaller than the discrimination interval for digital values.

2.2 Automobile Speed Illustration

As an example of first-principles models, I'll choose an automobile speed illustration. Although not a process model (such as reaction of distillation or heat exchange) the concepts are more universally familiar.

Start with Newton's Law of motion then expand the acceleration term to the rate of change of velocity.

$$F = \frac{ma}{g_c} = \frac{m}{g_c}\frac{dv}{dt} \tag{1}$$

An elementary representation of the forces on the car are from: 1) The engine, represented by the accelerator pedal position, u, which will be modeled as a linear response. 2) Aerodynamic drag, ideally represented as proportional to velocity squared. And 3) gravitational impact of going up or down hill with an angle, θ, from the horizontal.

$$F = ku - bv^2 - sin(\theta)mg/g_c \tag{2}$$

Combine Equations (1) and (2), to obtain the elementary model.

$$\frac{m}{g_c}\frac{dv}{dt} = ku - bv^2 - sin(\theta)mg/g_c \tag{3}$$

This can be converted into the classic control form of a 1st-order process by dividing by bv then rearranging terms.

$$\frac{m}{bvg_c}\frac{dv}{dt} + v = \frac{k}{bv}u - \frac{sin(\theta)mg}{bvg_c} \tag{4}$$

And, now, it can be represented in generic terms for a first-order process with a locally linearized disturbance model.

$$\tau\frac{dv}{dt} + v = K_p u + K_d\theta \tag{5}$$

Compare Equations (4) and (5) and notice that the time-constant, τ, is nonlinear, depending on the velocity. The time-constant is also non-stationary because the drag coefficient, b, will depend on air density; and because the mass, m, will depend on fuel in the tank and other loading changes. The process gain is also nonlinear and nonstationary in spite of the linear force model in Equation (2). And the sine function nonlinearity of the road angle impact, is now also confounded by speed, mass, and drag coefficient.

For model-based control we will use an alternate rearrangement of Equation (3) which isolates the rate of change.

$$\frac{dv}{dt} = \frac{g_c}{m}(ku - av^2 - sin(\theta)mg/g_c) \tag{6a}$$

In generic terms, with y as the state variable, u as the MV value, $\underline{d}$ the vector of disturbance variables, and $\underline{p}$ as the vector of model parameter values, Equation (6a) can be represented as

$$\frac{dy}{dt} = f(y, u, \underline{d}, \underline{p}) \tag{6b}$$

This represents a first-order process, a process stage with one inventory location of mass, momentum, or energy. It could represent how fluid flow rate responds to the controller signal to the valve actuator, how temperature in a well-mixed tank depends on a heat source, how composition in a well-mixed tank depends on influent flow rate or composition, how pressure in a vessel depends on in- or out-flow or internal generation of gas, etc.

Some one-inventory-stage processes have more than one state variable. For instance, in a continuous flow reactor with well-mixed contents, temperature, composition, and volume would be several interacting states, which would be dependent on several influences input flow rates, heating/cooling rates, and discharge rates. Here each CV would have a first-order model such as Equation (6a), but each state variable would have an influence on the rate of change of the others.

In generic terms for a MIMO set of coupled first-order processes, with $\underline{y}$ as the vector of state variables, $\underline{u}$ as the vector of MV values, $\underline{d}$ the vector of disturbance variables, and $\underline{p}$ as the vector of model parameter values:

$$\frac{d\underline{y}}{dt} = \underline{f}(\underline{y}, \underline{u}, \underline{d}, \underline{p}) \tag{7}$$

However, the process might have several stages that sequentially influence each other. For example, in a tray-to-tray distillation column, the liquid contents on each tray have several interacting states (temperature and compositions), which are influenced by the trays above and below. In a 10-tray binary distillation there would be 20 states on the trays (temperature and x_1) (the other liquid and vapor compositions would be related to the states). But there would also be column pressure and reboiler and condensate receiver as states. Although there may be 23 states, only 5 would be controlled (top and bottom composition, column pressure, and reboiler and receiver levels). Equation (7) could represent 23 first-order, nonlinear, interactive models, but there may only be 5 MVs (reboiler heat, bottom's flow rate, cooling flow rate to the condenser, reflux flow rate, and distillate flow rate).

As a note, regardless of the form of the model presentation, do not solve the model with calculus for several reasons. The nonlinearity and non-stationarities may preclude analytical solution. Any solution will be predicated on an idealized time-dependent behavior of the MV and disturbances. And any solution will be predicated on fixed values (or at least elementary models) of how the model coefficients might change in time.

Alternately, to determine what the model predicts, use a simple numerical solution. Here is a Euler's explicit method, by converting the differential to a forward finite difference.

$$\underline{y_t} = \underline{y_{t-\Delta t}} + \Delta t \cdot \underline{f}\left(\underline{y}, \underline{u}, \underline{d}, \underline{p}\right)_{t-\Delta t} \tag{8}$$

This is a recursive relation, the new values of the states, $\underline{y_t}$, are calculated from the prior values of the states and of all other variables, $\underline{y_{t-\Delta t}} + \Delta t \cdot \underline{f}\left(\underline{y}, \underline{u}, \underline{d}, \underline{p}\right)_{t-\Delta t}$. Initialize this calculation with the prior modeled y-value.

In this car speed example, the model was a single variable, and could be coded on a single line. It appears that Equation (8) could be coded on N lines, one for each of the N state variables. Alternately, a model may have IF-THEN conditionals such as representing disparate mechanisms such as due to the transition from laminar to turbulent flow, or from turbulent to choked flow. And there may be many other constitutive relations, such as thermodynamic models that relate state variables, even requiring root-finding methods to solve them. Or steady-state or equilibrium relations for actions that have very fast dynamics. Also, for numerical stability the model Δt might be 1/10th of the control interval, requiring that Equation (8) be recursively calculated 10 times within each controller interval. Alternately, a preferred numerical solution might be a 4th-order Runge-Kutta approach. For any of many reasons then, the future modeled values might be coded as a separate procedure (a function or a subroutine). There is no need to have a single line of code (or N single lines) as suggested by Equation (8).

Note: The model is not the truth about the process. This model presumed that the engine force is linear with pedal position and independent of engine speed, gear, and intake air composition. This model presumed that the engine force instantaneously responds to pedal position, that there are no lags in getting fuel-air mixture into the engine or back pressure on exhaust as engine speed increases. This model ignores rolling friction of the tires on the pavement. This model uses the ideal Bernoulli exponent of 2 for the aerodynamic drag force, and the model also ignores that the drag force would be a function of the differential velocity between wind and car speeds, not just car speed.

First-principles models are fully adequate for control. Use ideal relations. Do not seek fully rigorous models.

3 MPC Controller Structure

There are three functions within model-based control – predict, correct, and act. The structure can be represented by the operations in Figure 1.

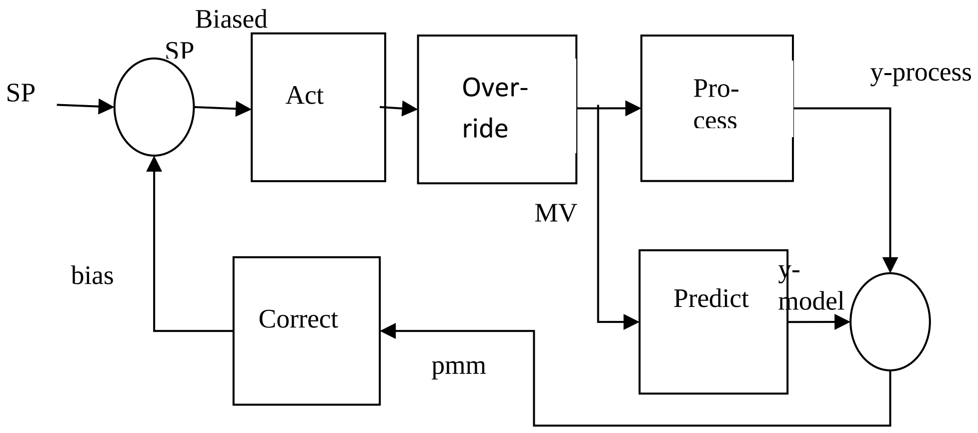

Figure 1 – MBC Representation

3.1 Predict

The first MBC function is to use the model to predict or mimic what the process output is expected to do. This is the lower right box in Figure 1. This updates the past model value (from the prediction) with past disturbance and control action, to predict the updated (current or new) modeled output. With conventional practice of 30 control samplings in an open loop transient (or 10 samplings within a time-constant), the control interval, Δt, is usually small enough to permit Euler's explicit finite difference method to solve the model numerically. Using Equation (8) to represent a SISO dynamic model, this model prediction calculation becomes:

$$y_{m,i} = y_{m,i-1} + \Delta t \cdot f(y_m, u, d, p)_{i-1} \tag{9}$$

Where "i" is the time increment counter, representing the present value, and "$i-1$" the immediate past values for each item in the function list. The subscript "m" refers to the modeled value, not the measured process value. However, "u" refers to the actually-implemented MV value, after any overrides change what the controller initially desired. The symbol "d" refers to measured disturbances or feedforward variables, and "p" represents parameter values in the model.

Note: The right-hand side of Equation (8) uses the prior modeled output value, $y_{m,i-1}$, not the prior process value, to update the modeled value, $y_{m,i}$. Because of process-model mismatch, noise, and such, the process measurement is not the right value to initiate the model update.

However, since computers use assignment statements, the i-subscript is not necessary, and there is no need to store all of the past model values. The new modeled y-value is based on the most recent prior y_m and possibly delayed input conditions. Here the symbol "$:=$" will indicate a computer assignment statement. A delay array with put and retrieve pointers would be needed to obtain the u_θ and d_θ values.

$$y_m := y_m + \Delta t \cdot f(y_m, u_\theta, d_\theta, p) \tag{10}$$

If the process model is high-order, then internally to a procedure representing "Predict" there would be a set of equations similar to Equation (10). Also, if the model has conditionals, or root-finding algorithms, then again, the model function would be a procedure, not just one equation.

3.2 Correct

The second function within model-based control is a correction for process-model mismatch. Process-model mismatch, pmm, is the difference between modeled prediction and process measurement, as represented by the circle difference operation on the lower right side of Figure 1. The modeled value will not exactly match the process value for a variety of reasons including model error (as discussed in Section 2.2), noise, or unmeasured disturbances. A simple correction approach is to bias the set point for the model with the process-model mismatch as follows:

$$pmm = y_p - y_m \tag{11}$$
$$y_{SPbias} = y_{SP} - pmm \tag{12}$$

The logic is illustrated in Figure 2. "If you aim at the target, but hit 3 cm low. Then next time aim 3 cm above the target."

The box labeled "correct' in the lower left of Figure 1, could provide other functions. One related to model coefficient adjustment has some managerial information benefits, but in this simple correction approach, it simply is a pass-through of the pmm value to bias the set point for the model.

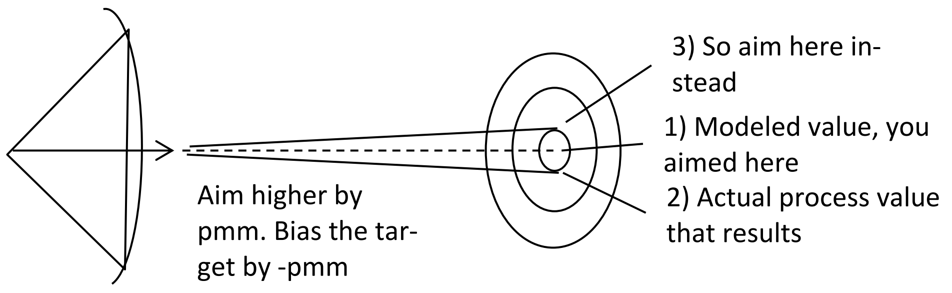
3) So aim here instead
1) Modeled value, you aimed here
2) Actual process value that results
Aim higher by pmm. Bias the target by -pmm

Figure 2 - The Concept of biasing setpoint for model by pmm.

Note: The biased set point is the set point for the model, not the set point for the process.

Note: pmm is not the actuating error. It is not the process deviation from the target. If at first you aim at the bull's eye, but hit 3 cm low. Next aim 3 cm high. If that shot (modeled to hit 3 cm above) has a process response that is 0.5 cm above the bull's eye, the pmm is the difference between where you aimed (+3 cm) and what happened (+0.5 cm). So, the pmm is 2.5 cm, next aim 2.5 cm above (not 0.5 cm below).

3.3 Act – An Elementary Choice – Simple PMBC

The third model-based control function is to calculate the control action, to determine the controller MV value plan. This is represented as the box labeled "Act" in Figure 1. This starts with a user-defined, desired performance objective for the controller to achieve. A simple desire, in the strategy termed Process-Model Based Control (PMBC) is to calculate a u-value that would push the model toward the biased set point at a rate proportional to the deviation from biased set point. (This is an effective strategy for low-order models with inconsequential delay. Section 3.5 discusses how to include constraints and ill-behaved dynamics in the MV plan.)

3.4 A Caution about the Triple Model Use

The control model is used in two places. In both the Predict and Act functions.

The Predict function updates the past modeled value with the actual implemented MV to predict what the CV is doing now. This is a one-time-step update using the first-step of past values. I use the name Past-to-Now which has the acronym P2N. Initialize the one-time-step update with the prior P2N modeled value.

The Action (Act) function uses a model to forecast what might happen into the future, associated with a trial solution for the sequence of MV values. Guess at an MV plan, see what the modeled CV does. Re-guess at an MV plan until the future modeled CV matches the desired rate path. I call this the Now-to-Future model, which has the acronym N2F. Initialize the N2F prediction with the current P2N modeled value. After each MV guess, the N2F model needs to be re-initialized with the current P2N value.

Note: Both the P2N and N2F models use the same Equation (10). But they need to be initialized with the correct y_m values. Use $y_{m,i-1}$ for the P2N model. Use $y_{m,i}$ for the N2F model.

Note: The P2N and N2F models must be consistent with each other. Once when I was clever, seeking a way to avoid the iterative search for the MV, I modeled the inverse with empirical approaches. But at steady-state the inverse surrogate for the N2F prediction did not exactly match the P2N model, and accordingly control had a steady-state offset. The inverse could be represented as $u = M_{N2F}^{-1}(y_{SPbias})$, and the model could be represented as $y_m = M_{P2N}(u)$. Then $y_m = M_{P2N}(M_{N2F}^{-1}(y_{SPbias}))$ and the models must ensure that at steady-state, $y_{SPbias} = M_{P2N}(M_{N2F}^{-1}(y_{SPbias}))$.

In control simulations, the Process function will be simulated by a model. If the process model in the simulator is the same as the controller model, then there will be no steady-state offset, zero pmm, and perfect control. This is cheating. I do not know of any process that could possibly be exactly modeled. Be sure that the controller models are not identical to the process model in the process function. This includes both functionality and coefficient values. For example: If you use the Ideal Gas law in the controller model, use a different Equation of State in the process. If you use the ideal Bernoulli relation for fluid behavior in the controller model, use 1.8 rather than 2 as the exponent in the process. If you use a catalyst reactivity of 0.8 in the controller model, use a different, and ever-changing value in the process simulator. There should be many differences between the controller models and the process simulator model.

3.5 The Act Function in Horizon Predictive Constraint Handling Control

In Horizon-Predictive Control (HPC), the Act function forecasts several future MV moves (3 will be used in this article) to:

1. Shape the future modeled CV response to best fit a reference trajectory (here, first-order, from current model P2N value to the biased setpoint for the model).
2. Steer variables clear of constraints or limits, or to balance violations when a constraint is unavoidable.
3. Economically optimize which MVs to use when there are extra Degrees of Freedom (extra MVs).

Calculating the future MV sequence is an optimization application.

HPC Objective 1 – Following the Reference Trajectory - Figure 3 illustrates Item 1 in the list above. The vertical axis represents a PV or MV value, and the horizontal axis represents time. The time of 40, marked by the vertical line, is the current time. To the left are past events, to the right extending beyond a time of 50 is a possible future plan of three MV steps. The plan has not happened and might not happen. Like a trip plan, which is revised when you hit a detour or decide to take a side visit, the original might never exactly happen. In this example, the process started at an initial steady state. At the left of the time maker, the black line (the lower of the three lines) has been the CV, which is steady and at the set point. The red line (the upper of the three) has been the MV. The turquoise line (the middle line) has been the model predictions. The difference between the modeled and actual CV values is the pmm.

To the right of the time marker are the possible future MV outcomes. The lower horizontal line is the biased setpoint, the current target for the model. The black line, which makes a first-order transition from the current modeled CV to the biased set point is the reference trajectory, the desired path for the controller MV plan to move the modeled CV value. This was chosen to be a first-order path. The plan is the three steps in the red line, the possible MV sequence. The green line, the upper of the lines early in the future, just after the now marker, is the modeled CV value response to the three MV steps. It starts at the present modeled value, shows a bit of a delay, then a higher order lag, and a subsequent path which substantially follows the reference trajectory. To get this modeled CV path, the controller MV plan is to first make a large change to get the modeled CV to move quickly; but, if held there, the modeled CV would fall substantially lower than desired. Continuing the plan, raise the MV to prevent overshoot from the reference trajectory, but this would cause the modeled CV to remain above the reference trajectory longer into the future, so the third step in the MV plan is a bit lower.

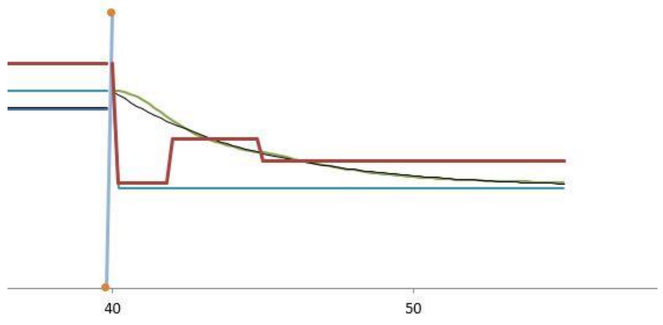

Figure 3 – Forecast MV Moves to make the Model Best Match a Desired Path

Note: The future forecast time is termed the control horizon. Hence the name Horizon Predictive Control. The horizon does not extend to the settling time, to steady state for either the modeled or process CV or for the reference trajectory, but typically to 70% to 90% of the settling time. Accordingly, the reference trajectory does not get to the biased set point, as Figure 3 illustrates. This does not matter, because only the first MV step will be implemented. Then at the next sampling, the future plan is recalculated with information about new disturbance values, new pmm evidence, and new set points.

Note: The control interval is not the length of the first MV step-and-hold. In this simulation there are about 30 control intervals in the future plan. The first MV step-and-hold in the current plan lasts for about 15% of the horizon (about 5 control intervals), the second step-and-hold for about 25% (about 7 control intervals), and the last for about 60% (about 24 intervals).

Note: This three-step forecast MV pattern is not essential. More steps would give more precise matching of model to reference trajectory, but in my experience three is fully adequate and minimizes the optimization burden. One could use alternate spacing patterns for the MV transitions. Since the first step is what will be implemented and it shapes the early CV response; and since subsequent MV steps trim the plan and are farther out into the future, I like a plan that has more MV moves earlier. This plan roughly spaces MV moves according to the golden ratio, but any of many options work just as well. You might be more comfortable with ¼ , ¼, and ½.

The optimization objective is not to force the model to match the reference trajectory exactly, but to minimize the forecast CV deviations from r, the reference trajectory. If the model dynamics are not simply first-order, then it would be impossible to make model match the reference exactly. The objective function is a least squares deviation from reference trajectory over the control horizon.

$$\underset{\{u_1, u_2, u_3\}}{Min} \quad J = \sum_{i=1}^{H} ({r_1}_i - {\tilde{Y}_1}_i)^2 \tag{13}$$

Here i is a counter for future intervals, and H is the number of intervals in the control horizon.

If the reference trajectory is first-order, the time-constant, tau, is the controller tuning factor:

$$\tau \frac{dr}{dt} + r = SP', \qquad t_{t=now} \leq t \leq H, \qquad r_{t=now} = \tilde{Y}_{P2N,\,now} \tag{14}$$

If Δt, the control interval is small relative to τ, the time-constant for the reference trajectory, as it should be, then using Euler's Explicit Method to determine r(t):

$$r_{t+\Delta t} = \frac{\Delta t}{\tau} SP' + \left(1 - \frac{\Delta t}{\tau}\right) r_t \tag{15}$$

For a multivariable process, each CV will have its own reference trajectory, and each MV will have several future values.

Note: Some MPC algorithms have as the objective to make the model match the biased set point, which would create a very aggressive control action. For these, the control action is tempered by a move suppression factor, a penalty for large MV moves. This is termed MV damping. What I have described with a reference trajectory target, is termed CV damping. In a MIMO process the move suppression factor on any one MV affects all CV responses. Since tuning with MV damping is interactive in its effect on CV responses, and since it does not have an explicit relation to any one of the CV dynamics, I prefer CV damping. CV damping choice of τ for each CV has a direct relation to that CV response speed.

HPC Objective 2 – Avoiding Constraints - Another desirable, Item 2 in the list above, is to plan the future MV sequence to avoid constraints. The constraint might be on a

CV, or any other process variable (PV), often termed an Auxiliary Variable (AuxV). The constraint might represent a safety or specification limit, or a desired resting value, or a rate of change. There are diverse ways to penalize either a constraint violation or proximity to the constraint. The "Soft" penalty method for a deviation penalizes a violation, proportional to the square of the violation.

$$Penalty_1 = \begin{Bmatrix} 0, & if\ AuxV_1\ is\ within\ the\ limit \\ (PV_1 - L_1)^2, & if\ AuxV_1\ exceeds\ the\ limit \end{Bmatrix} \tag{16}$$

But constraints could happen at any future time, so sum all of the penalties.

$$\underset{\{u_1, u_2, u_3\}}{Min} \sum_{i=1}^{H} Penalty_{1,i} \tag{17}$$

If several variables are subject to constraints, calculate the penalties for each.

Constraints on the MV, such as rate-of-change limits or values beyond the 0-100% range of feasible values, could be handled as either soft constraints in Equation (17) or as hard constraints, limiting MV choices in the optimization. I prefer to treat MV constraints as hard, limiting feasible values for the optimizer.

CV or AuxV constraints should be soft. Near a constraint, a disturbance, noise, or such could move the AuxV beyond the constraint. Then, being beyond now, it has happened, and no future control action can undo that. In such a case, all attempts to find the MV sequence will fail. To prevent such infeasible corrections, treat CV and AuxV constraints as soft.

HPC Objective 3 – Minimizing Utility Cost - The third objective in the list, Item 3, is to minimize costs of using MVs, when there are extra MVs available. If there are extra MVs, then an infinite number of MV actions can provide equivalent control. But usually, some MVs are more expensive than others. For instance, in heating, you may have available electrical resistance heating or low-pressure steam. You would prefer to use the less expensive steam until supplemental heat is necessary. To minimize the cost of MV use over the future horizon calculate the cost for each MV in the plan. A simple model for cost rate for an MV is $\dot{C} = c \cdot MV$. Summed over each future time interval, for one MV the objective is:

$$\underset{\{u_1, u_2, u_3\}}{Min} J = \sum_{t=now}^{t=H} \Delta t \cdot \dot{C}_t = \Delta t \sum_{t=now}^{t=H} c \cdot MV_t \tag{18}$$

For a process with multiple MVs, the projected cost over the control horizon is:

$$\underset{\{u_1, u_2, u_3\}}{Min} J = \Delta t \sum_{l=1}^{L} \sum_{t=now}^{t=H} c_l \cdot MV_{l,t} \tag{19}$$

Note: if there are extra MVs then, instead of this economic cost method, you could define a resting value of zero for the high cost MVs, and use a deviation from the resting value as a penalty as described in Section 3.5.2.

Combining all Optimization Considerations **-** Each of the three considerations represent undesirable aspects. The controller should determine the control action that minimizes all. Now that the three considerations are quantified, a standard way of combining all is to sum all the measures of badness, and seek the MV sequences that minimize the sum of all undesirables.

$$\underset{\{\underline{u}_1, \underline{u}_2, \underline{u}_3, \ldots\}}{Min} J = \frac{\sum_{i=1}^{H}\left(r_{1_i} - \tilde{Y}_{1_i}\right)^2}{EC_1^2} + \frac{\sum_{i=1}^{H}\left(r_{2_i} - \tilde{Y}_{2_i}\right)^2}{EC_2^2} + \cdots + \frac{\sum_{i=1}^{H} Penalty_{1,i}}{EC_3^2} + \frac{\sum_{i=1}^{H} Penalty_{2,i}}{EC_4^2} + \cdots + \frac{\Delta t \sum_{l=1}^{L} \sum_{i=1}^{H} c_l \cdot MV_{l,i}}{EC_{cost}} \qquad (20)$$

Note: The decision variables, the $\underline{u}_i$ factors, are underscored, representing vectors. In a multivariable process there are several MVs, and each would have its several future possible values (three MV values for one MV have been illustrated here).

Because each term in Equation (20) will have unique units, and because some aspects will be more meaningful than others, weighting factors are required to unify units, and to balance relative importance. This representation uses Equal Concern (EC) factors. They scale each quantity by a factor that has the same units as the numerator, and represent equal concerns for the violation of the desired value.

Note: If there are no constraint violations, then the penalty terms in Equation (20) are all zero. If the model CVs can exactly follow the reference trajectories (or if the model is at steady state at the biased setpoint) then the first set of terms is also zero. The MV cost term, however, will not be zero. If the optimizer changes MV values so that the MV cost term is reduced, the new MV values will cause the model CVs to deviate from the desired path or from the biased setpoint value. This would make the first set of terms non-zero. It may be possible to reduce cost by a greater amount than the penalty for not following the reference trajectory. This would mean that the process CVs will have a steady state offset. So, if the cost term is included in the optimization objective function, it is important that the weighting factors make the MV cost term much less important than either the CV deviation or the violation penalty terms. Choose EC factors so that any steady state offset is imperceptible. See Section 3.5.5. Alternately consider other ways to manage the economic factors. See Section 3.5.6.

Note: The EC weighting factors could be used for tuning aggressiveness, but that is not their role. The time-constants for the reference trajectories are the tuning factors. One for each CV.

HPC 2 – Setting the EC Factor Values **-** Here is a procedure for setting the EC factors.

1. Choose a most critical term. One term to minimize is $\sum_{i=1}^{H}(r_{1_i} - \tilde{Y}_{1_i})^2$, maybe this is of greatest concern.
2. Choose a moderate $\tilde{Y}_1$ deviation from perfection. It might be the deviation from the reference trajectory along the future time, or the deviation from the biased set point at steady-state. This value is an EC factor, EC_1 is equal to the $\tilde{Y}_1$ deviation considered to be moderate.
3. Feel the concern this deviation creates. Use both qualitative and quantitative aspects. Perhaps rate the concern level on a 1 to 10 basis. Concern aspects could include your boss's opinion, customer reaction consequences, the generation of waste product, etc.
4. Assign EC values to each term in Equation (20) representing each deviation from perfection that has the same level of concern. As an example:
    - Perhaps $EC_1 = 0.5 wt\%\, product\ deviation\ from\ SP$ has a 1-10 concern rating of a 3.
    - To have the same concern rating of 3, perhaps the deviation from a target flow rate is $15\ lb/min$. Then $EC_2 = 15\ lb/min$.
    - Also, to have the same concern rating of 3, perhaps the deviation from a process temperature limit is $10\ F$. Then $EC_3 = 10\ F$.

The EC values are situation-dependent, and they change with the situation.

Note: Traditional optimization uses Lagrange-type multipliers for the terms in a sum. Here, those values are the reciprocal of the denominator values. I think that the weighting method using EC factors is intuitive and easier to assign weighting values, which provide the relative importance.

Choosing the economic EC factor may be a bit difficult because it must also balance the consequential steady state offset. To help see the way to include this aspect, consider a single CV. Ideally the deviation from set point is zero with some particular MV value. If the optimizer chooses a less expensive MV value perturbed by Δu, there will be a corresponding $\Delta\tilde{y} = K_m \Delta u$, and the CV deviation factor will be $n\left(\frac{\Delta\tilde{y}}{EC_y}\right)^2$, where n is the number of samplings in the horizon. The cost term improvement will be $\frac{\Delta t \cdot n \cdot c_l \cdot \Delta u}{EC_{cost}} = \frac{\Delta t \cdot n \cdot c_l \cdot \Delta\tilde{y}/K_m}{EC_{cost}}$. We would like the CV deviation to be imperceptible, perhaps small relative to CV noise or relative to the width of the image on the display. We desire $\Delta\tilde{y} \le \varepsilon$. Equating the two factors and solving for EC_{cost}, $EC_{cost} = \frac{\Delta t \cdot c_l \cdot {EC_y}^2}{\varepsilon \cdot K_m}$.

Alternate Methods to Account for MV Costs - One option is to not include the last term in the Equation (20) objective. It is not in standard linear empirical model-based controllers.

Another option is to recognize that two MVs may provide the same benefit, but one is more expensive than the other. This may be two possible heating mechanisms steam or the more expensive electrical resistance. In such a case use the steam until its capacity is inadequate, then add the electrical resistance to provide the extra needed. As another example there may be two sources of a reagent, dilute and the more expensive concentrate. Use the dilute first, until a flow rate limit is reached or until the dilution causes alternate issues, then trim with the concentrate. There are several ways to handle this logic, one is to split the 200% range of the optimizer MV. If 0-100% use the less expensive source. If 101-200% also use the more expensive source. Another method is to have a resting value for the more expensive source (it actually may be non-zero to ensure it is immediately available) and add a penalty for deviations from the resting value.

A third case might be one in which the extra MV does not have a cost, but affects the values of the other MVs. As an example, pressure on a distillation column can be adjusted by a throttling valve on the vapor leaving the column or on adjusting the flow rate of coolant (process water, or air) in the distillate condenser. Pressure is normally specified in an optimization exercise to stay within limits, maximize column efficiency, and balance performance over a range of expected duties. Then not included in the column composition control. However, pressure could be considered an MV for control, and the optimization may determine the pressure that results in minimizing the cost of reboiler power and distillate condenser cooling.

***Optimization to Determine the Possible Future MV values* -** The optimization statement is

$$\min_{\{\underline{U}_{l,m}\}} J = \sum_{j=1}^{N} \frac{\sum_{i=1}^{H}(r_{j_i} - \tilde{Y}_{j_i})^2}{EC_j^2} + \sum_{k=1}^{M} \frac{\sum_{i=1}^{H} Penalty_{k,i}}{EC_k^2} + \frac{\Delta t \sum_{l=1}^{L} \sum_{i=1}^{H} c_l \cdot \dot{MV}_{l,i}}{EC_{cost}} \tag{21}$$

S.T: $hard\ constraints\ on\ the\ MVs$ (Value or Rate-of-Change)

The U Subscripts are l=# MVs, m=# future moves for each MV
N=# CVs, M=# soft constraints, L=# MVs
H= control horizon – the number of Δt calculations in the future time projection. This should go to about 85% of open loop settling time.

One could include MV constraints, such as values or rate-of-change (RoC) in the soft penalty terms. These are input variables to the process. However, one cannot include process response variables (CVs and AuxVs) in the list of hard constraints, because a disturbance may cause a response to exceed a limit, and the controller will be unable to immediately remove it from the constraint.

Classic MPC products use Linear Programming or Successive Quadratic. However, there are a number of other constraint-handling gradient-based techniques that could be used. Contrasting these, my preference is direct search, multi-player optimization methods. Multi-player methods are “global”, and direct search methods do not get confounded by surface aberrations (non-idealities, discontinuities, numerical striations, etc.), can handle hard constraints, and nonlinear responses. Leapfrogging is my choice, considering effectiveness, speed, simplicity, robustness [1].

DDC or not? - Direct Digital Control (DDC) sends signals from the controller directly to the final control element (valves, heaters, etc.). If you have a single loop controller, SISO, and the model-based control calculation is within the PLC or DCS that then this may be permissible.

However, model-based controllers are typically implemented external to the PLC or DCS where greater computational power is available. Although, the model-based controller decisions regarding the signal to the final control element could be transferred to the DCS to be relayed to the process, experience has shown that it is best to transfer set points for the process primitive variables (flow rates, temperatures, levels, etc.) and let the PID algorithms in the DCS or PLC determine the signals to the final elements. This is a cascade type of implementation. The primitive, or lower-level loops typically respond faster than the primary variables that need to be managed by constraint-handling horizon-predictive control. So, this transfer of set points does not realistically slow control response. The advantages are that the DCS or PLC has many auxiliary functions related to signal processing and validation that should be used, and it is designed to be robust to power and communication failures. If for any reason the computer in which MPC is implemented gets confounded, or fails, the DCS or PLC will continue process operation with the last set of set point values. The process will not shut down, if the MPC computer does.

4 MPC Example

This example will discuss nonlinear, horizon-predictive, constraint-avoiding model-based control for the car example in Section 2.2. The process, representing the real car, is a simulation. To make the illustration legitimate, the simulation equations for the process are not the same of those for the controller model. The simulation is more complex, has alternate functionalities for effects, and coefficient values that are different from the controller model. One cannot perfectly model the reality of a process. Simulations of control should reflect that reality.

For control this is a SISO process with the controller output to the final control element. However, the process also has an auxiliary variable, intended to represent cabin noise, which has an upper limit constraint.

Figure 4 represents a deterministic and disturbance-free trend in simulated process and controller model, subject to ideal step changes in the MV.

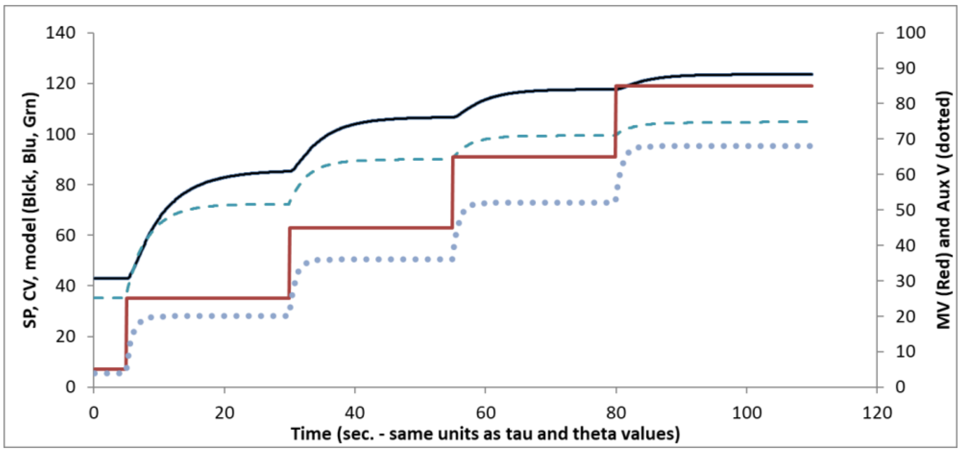

Figure 4 – Trends in MAN Mode

The red trace that makes step changes is the car accelerator pedal position, the MV. It is attached to right-hand vertical axis. Each step is an increase of 20%. The upper curve is both the car speed and the set point, which are attached to the left-hand vertical axis. In the MAN mode, the set point tracks the CV for bumpless transfer to AUTO. Note, the first MV step increases the CV by about 42 kph, but the last equivalent MV step only by about 6 kph. The gain change ratio is about 7:1, being one evidence of nonlinearity. Also note that at the higher speed the settling time is about 15 seconds, while at the first MV step at the lower speed, even after 25 sec., the process is still rising. This nearly 2:1 change in settling time is a second evidence of nonlinearity of the process.

The model is represented by the blue dashed line, also attached to the left-hand vertical axis. It too, is nonlinear, with dynamics that change with operating speed. However, the model features do not match the process, exactly. The model is usually lower than the process and the pmm progressively increases with speed. But because the dynamics are different at the first MV step the model actually rises, temporarily, above the process.

The lowest curve, the blue dots, is an Auxiliary Variable, in this case it is a very simple representation of noise volume associated with the engine and car movement.

Figure 5 illustrates several features in AUTO mode. Like Figure 3, the vertical line toward the right of the graph marks the time now, separating the past and future. Initially, at time=0 the controller is in MAN mode, and placed in AUTO at a time of 2 sec. Because of set point tracking and P2N model calculation in MAN mode, there is no bump in the transfer to AUTO. Set point changes occur at times of 5, 40, 60, and 80 sec. indicated by the black line that makes steps. The solid blue curve represents the process CV (car speed); and in spite of the 7:1 gain nonlinearity, the 2:1 settling time ratio, and the changing pmm, the overshoot magnitude and rise time in the first and third set point changes are very similar.

The second and fourth set point changes represent constraint conditions. The AuxV (blue dots) has a soft constraint of 50 (right vertical axis); so even though Figure 4 shows that the speed set point of 120 kph is feasible at MV=85%; at that MV value, the AuxV has a value of about 70, exceeding the constraint. In Figure 5, the EC factors make constraint violations more important than meeting the speed set point, and the optimizer limits the MV to about 60% preventing the speed from reaching its set point. When the setpoint is dropped to a speed of 100 kph, there is no delay in the MV action (there is no wind-up in the controller). That was a soft constraint. A second constraint happens at the final set point change, where the controller would like to place the MV below zero. The MV stays at the constrained value limit, until responding (again without wind-up). That is a hard constraint. The third constraint is a rate-of-change constraint on the controller, another hard constraint, which is evidenced by the linear ramp change in the MV immediately after each SP change.

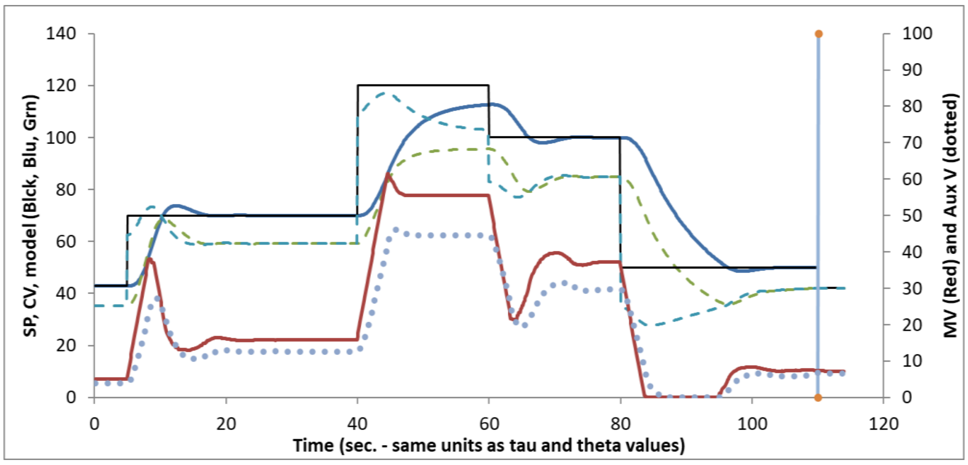

Figure 5 – Events in AUTO Mode – 1

Figure 6, Illustrates the same events as in Figure 5, except that the rate-of-change constraint is removed, and the EC factor for the AuxV violation is increased to place greater importance on the CV to SP response. Without the rate-of-change constraints, note the spikes in the MV action just after the set point changes and the MV limit of 100% at a time of 40 sec. Also notice the lesser deviation of the CV to the set point and associated larger deviation of the AuxV from the limit of 50 during the 40-60 sec. period.

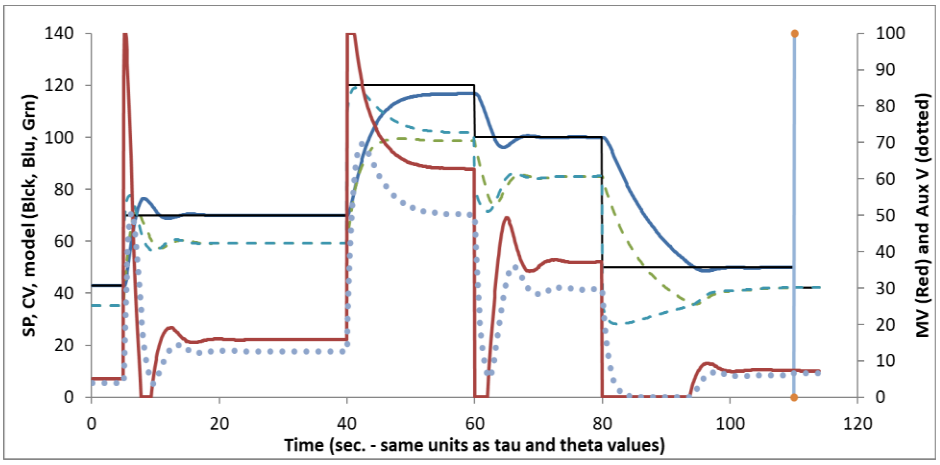

Figure 6 – Events in AUTO Mode – 2

There is only one tuning factor for each CV, the time-constant for the reference trajectory. In Figure 7 the time-constant is significantly larger, making the controller less aggressive. Compare to Figure 6, also with no rate-of-change constraint and the duplicate EC factors. Notice the smaller jumps in the MV action and nearly overdamped response of the CV to the SP.

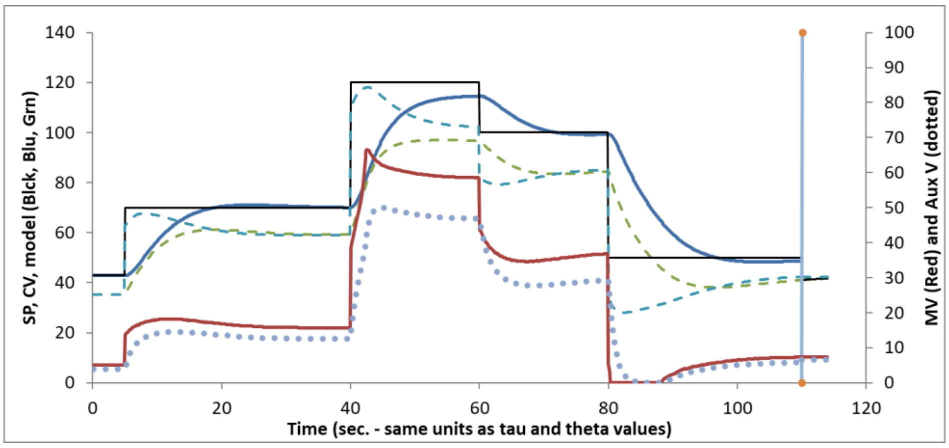

Figure 7 – Events in AUTO Mode – 3

Finally, in Figure 8 both measurement noise and random disturbance patterns are included in the process. Compare this to Figure 6, which has the same tuning and EC values. The noise is evident in the perturbations on the CV value throughout. The disturbance is evident on the trend in the MV during the 20-40 sec. period and the up-and-down action in the MV during the 60-80 sec. period, even though the CV remains about at the set point. The disturbance is unmeasured, not included in the controller model, but affecting the process only. This additional source of the process-model mismatch is also evident in the 20-40 sec. period where the pmm changes during the 20-40 time period.

The pmm is filtered to temper the impact of measurement noise. The noisy CV is not filtered, which has a benefit of being able to display the noise pattern and the real state of the process. If the process CV were filtered creating a significant lag, then the modeled CV should be equivalently filtered also.

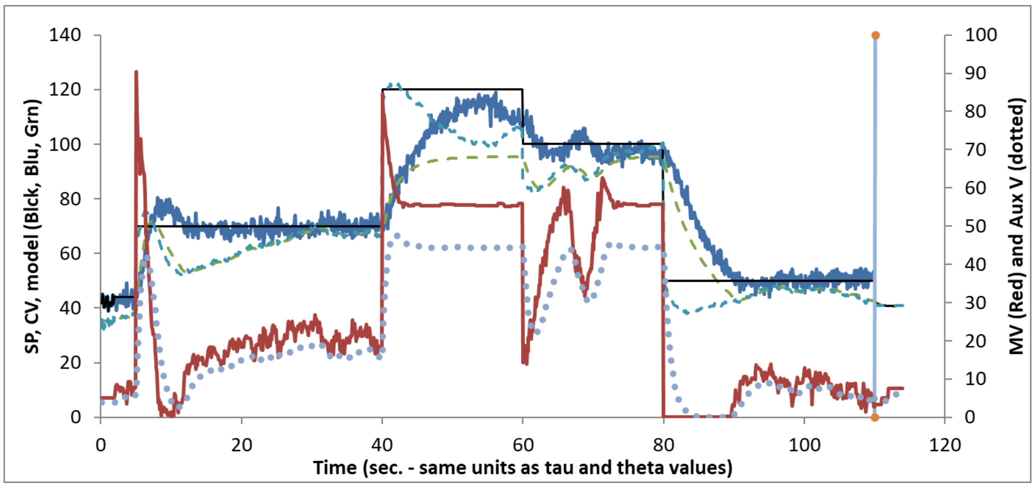

Figure 8 – Events in AUTO Mode – 4

5 End Notes

The model does not have to be perfect, great, very good, or even good. A fair model works as this simulation reveals. As additional evidence that imperfection works: We use linear constant FOPDT models for nonlinear high-order processes. We use linear FIR models in APC (MPC, DMC). We steer cars through curves and catch fly balls by continually correcting our mental intuitive "models".

Implement advanced control in stages. Start with the model shadowing the process to affirm model validity. Test bumpless transfer. Then observe the controller suggesting action. If confident that it is making good decisions, let the controller take action while being supervised. Finally, implement autonomous control, but displaying goodness of control messages.

Implement HPC showing the operator the horizon-predictive constraint-avoidance predictions. Otherwise, the decoupling and predictive consideration might lead to unexpected decisions and operator discomfort.

There may be MVs removed for temporary maintenance. These should be removed from optimization. Include on-off switches so that the controller understands the temporary condition.

Be sure that measurements are valid. Faulty inputs lead to bad decisions. This is no different for any level of complexity in control. You may need sensor redundancy or data reconciliation for critical measurements.

When the process changes (perhaps piping or units are reconfigured), you may need to change the model.

Provide a trap for any possible infeasible calculations and a Plan B action. Cascading MPC outputs as set points to lower-level PLC or DCS controllers is one strategy to keep the process running even if the supervisor MPC gets stuck.

K.I.S.S. Keep it Simple and Safe. If gain-scheduled PI works use it, if classic advanced regulatory control solves the problem use it, if GMC-SS solves the nonlinearity use it.

Consider incremental model parameter adjustment on-line to make the model adapt to the process. Observation of the adapting model coefficient can be a clue for predictive maintenance, and the model will be better at forecasting the constrained limits of process operation.

The justifications for first-principles models in control include:

- One-model consistency for many uses.
- Preserving process mechanistic knowledge.
- Process analysis, monitoring, and constraint forecasting.
- Handling nonlinearity, non-stationarity, disturbances that invalidate linear models.
- Ease of tuning.
- Ease and reduced process testing for model calibration (adjustment).
- Wider valid modeling range than with linear models.

How to economically justify using a nonlinear model? APC (linear multi-variable control) experience shows a halving (or better) of process variance, which permits operating closer to constraints and specifications. This permits higher throughput, reduced waste, higher quality, lower operating costs, etc. which provides the economic benefit. In my assessment, any control scheme (ARC, Fuzzy, APC, NMBC) that has equivalent disturbance-rejecting, feedforward, and decoupling features has the same benefits as any other controller with those same features. The nonlinear model-based controllers will have those benefits. As well it will work over a wider operating range, require fewer recalibrations, and be less expensive to recalibrate than controllers with empirical models.

References

1. Rhinehart, R. R., Engineering Optimization: Applications, Methods, and Analysis, John Wiley & Sons, New York, NY, (2018).