

# Infeasibility Handling in MPC with Prioritized Constraints

Willy Wojsznis, Terry Blevins, Mark Nixon and Peter Wojsznis  
Emerson Process Management, Austin, TX  
[Willy.Wojsznis@EmersonProcess.com](mailto:Willy.Wojsznis@EmersonProcess.com)

## KEYWORDS

Optimization, Constraint Model Predictive Control, LP

## ABSTRACT

Optimization is an integral part of multivariable Model Predictive Control (MPC). This extremely successful merge of a two major technologies is due to a great extent to the fact that MPC in normal operation provides future prediction of the process outputs up to the steady state, creating the required conditions for reliable optimizer operation. The standard optimization approach which finds a solution within acceptable limits, does not work when some of the process outputs or predicted process outputs are out of limits. On the other hand the optimizer applied with the MPC controller should always find a solution and thus extend the original optimization formulation. This paper presents robust and reliable ways of handling optimized process outputs that are out of limits. The techniques are based on the priority structure, penalized slack variables, and redefining the constraint model. The paper presents background information on the techniques, and an example illustrating optimization and constraints handling.

## INTRODUCTION

Initial optimizer control applications for defining optimal economic set points has been extended in optimizers applied with MPC for economic optimization and constraints handling [1]. In normal operating conditions, an optimizer provides an optimal economical solution within acceptable ranges and limits. If a solution does not exist within the predefined ranges and limits, the optimizer should have the means to recover from the infeasibility. The existing recovery techniques are based on the priorities of the constrained and controlled variables. A simple approach is to drop the lowest priority constraints. More rational ways however are required to deal with constraints. There are several known approaches to this problem. In [2], integer variables are used to cope with prioritization. The minimization of the size of the violation is performed by solving a sequence of mixed integer optimization problems. In [3], an algorithm solves a sequence of LP or QP problems in the case of infeasibility. This algorithm, like the previous one, minimizes the violations of the constraints which cannot be fulfilled.

Both approaches are demanding computationally and inadequate particularly for fast real time applications.

In [4], an algorithm has been developed for off-line LP weights design in such a way that the computed constraint violations are optimal. It takes the burden of excess computations off-line but presents an additional off-line optimization problem.

In this paper, simple techniques suitable for on-line implementation are presented. The generalized approach with penalized slack variables is discussed as a primary solution. Redefining a constraint model in such a way as to drive constraints with higher priorities toward limits with no further constraint violations for lower priorities is presented in detail as an alternative for satisfying more application objectives. Finally, merging the two approaches of penalized slack variables and constraint model redefinition provides the most flexible and efficient constraint handling technique. For the reader's convenience, MPC optimization as outlined in the paper [5] precedes the infeasibility handling technique presentation.

## BASICS OF MPC OPTIMIZATION

For processes with several manipulated or controlled variables, optimization techniques are an essential component of Model Predictive Control technology. One of the proven approaches uses linear programming (LP) with steady state models. Linear programming is a mathematical technique for solving a set of linear equations and inequalities that maximizes or minimizes a certain additional function. This additional function is called the *objective function*. Usually objective functions express economic value like cost or profit.

Specific to MPC optimization is the use of incremental Manipulated Values (MV) at the present time or sum of increments of MV over control horizon and incremental values of Controlled and Constrained Values (CV) at the end of the prediction horizon, instead of positional current values as in typical LP applications. The LP technique uses a steady state model and therefore steady state condition is required for its application. With prediction horizon normally used in MPC design, future steady state is guaranteed for self-regulating processes. Predicted process steady state equation for an  $m$  by  $n$  input-output process, with prediction horizon  $p$ , control horizon  $c$ , in the incremental form is:

$$\Delta CV(t+p) = A * \Delta MV(t+c) \tag{1}$$

where

$\Delta CV(t+p) = [\Delta cv_1, \dots, \Delta cv_n]^T$  denotes the vector of the predicted changes in outputs at the end of the prediction horizon,

$$A = \begin{bmatrix} a_{11} & \dots & a_{1m} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{nm} \end{bmatrix} \quad \text{the process steady state } m \times n \text{ gains matrix,}$$

$\Delta MV(t+c) = [\Delta mv_1, \dots, \Delta mv_m]^T$  denotes the vector of changes in manipulating variables at the end of the control horizon

Vector  $\Delta MV(t+c)$  represents the sum of the changes over the control horizon made by every controller output  $mv_i$ .

$$\Delta mv_i = \sum_{j=1}^c mv_i(t+j) \quad i=1,2,\dots,m$$

The changes should satisfy limits on both MVs and CVs.

$$MV_{\min} \leq MV_{\text{current}} + \Delta MV(t+c) \leq MV_{\max} \quad (2)$$

$$CV_{\min} \leq CV_{\text{predicted}} + \Delta CV(t+p) = CV_{\text{predicted}} + A * \Delta MV(t+c) \leq CV_{\max} \quad (3)$$

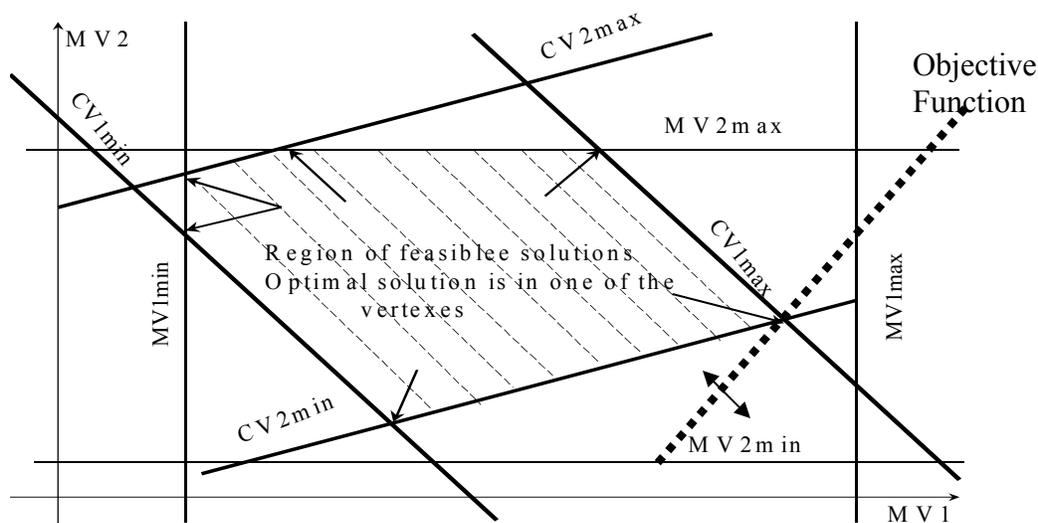
The objective function for maximizing product value and minimizing raw material cost can be defined jointly in the following way:

$$Q = -UCV^T * \Delta CV(t+p) + UMV^T * \Delta MV(t+c) \quad (4)$$

where  $UCV$  – cost vector for a one unit change in CV process value and  $UMV$  – cost vector for a one unit change in MV process value. Applying (1), the objective function expressed in terms of MV only is:

$$Q = -UCV^T * A * \Delta MV(t+c) + UMV^T * \Delta MV(t+c)$$

The LP solution is always located at one of the vertices of the region of feasible solutions. This is illustrated in Figure 1 for a two dimensional problem.



**Figure 1. Optimization Problem of a Two Dimensional System**

The region of feasible solutions for two controlled/constrained variables and two manipulated variables is an area contained within MV1 and MV2 limits represented by the vertical and horizontal lines and CV1 and CV2 limits represented by the straight lines (5) and (6).

$$CV1_{\min} = a_{11}MV1 + a_{12}MV2 \quad CV1_{\max} = a_{11}MV1 + a_{12}MV2 \quad (5)$$

$$CV2_{\min} = a_{21}MV1 + a_{22}MV2 \quad CV2_{\max} = a_{21}MV1 + a_{22}MV2 \quad (6)$$

The optimal solution is located at one of the vertices marked by arrows. To find this solution, the LP algorithm calculates the objective function for an initial vertex and improves the solution every next step until it determines the vertex with maximum (or minimum) value of the objective function as an optimal solution.

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

$$\Delta CV^c = A^c * \Delta MVT$$

$\Delta MVT$  - optimal target change of MV

$CV^c$  - subset of control and constraint variables redefined as control variables and included in MPC squared controller

$\Delta CV^c$  -  $CV^c$  change to achieve optimal MV.  $CV^c$  change is implemented by managing  $CV^c$  set points.

In normal situations, the target set points are within acceptable ranges and manipulated and constraint variables are within limits.

However when disturbances are too severe to be compensated within constraint limits, some constraints are violated. Formally the optimizer couldn't find a solution in such situations and some mechanism was needed for handling infeasibilities. The possible objectives of handling optimization may include:

1. Abandoning any optimization action
2. Dropping lower priority constraints
3. Dealing with all constraint variables by assigning penalties for constraints violation dependant on its priority
4. Dealing with all constraints by assigning penalties for constraints violation dependant on its priority, and redefining the model in such a way as to prevent any further constraints violation

Abandoning optimization is the simplest action. However, it is not the best method for constraint handling. When there is no solution within limits, it is still possible in most cases to make constraint violations smaller. Dropping lower priority constraints is an extreme action to help higher priority constraints. This may cause excessive uncontrolled departure from the limits of the constraints that are dropped off. Another problem with dropping off constraints is estimating how many lower priority constraints should be dropped to get a solution. An estimate is based on degree of freedom calculation, which is done before optimization. In the process of developing an optimal solution, dropping more constraints may be required until a solution is found. It may take several trials of dropping constraints and trying to find an optimal solution, which is not desirable in any real time application. More effective techniques are detailed in the following sections.

## **INFEASIBILITY HANDLING BY SLACK VARIABLES AND PRIORITIES**

The concept is based on a special use of slack variables. In linear programming slack variable vectors  $S_{\max} \geq 0$  and  $S_{\min} \geq 0$  are used to transform inequalities (3) into equalities:

$$CV_{predicted} + A * \Delta MV(t + c) = CV_{min} + S_{min} \quad (7)$$

$$CV_{predicted} + A * \Delta MV(t + c) = CV_{max} - S_{max} \quad (8)$$

The equality is required for the linear programming model thus slack variables serve only as supporting formal parameters with no specific application meaning. Adding slack variables can be interpreted as an increment of degrees of freedom. Therefore adding yet another slack variable for each equation will increase degree of freedom and allow finding a solution in situations when the solution may not exist. In MPC applications, slack variables are used to extend limits range on the slack vector  $S^+ \geq 0$  for the high limit increase and  $S^- \geq 0$  applied for the low limit decrease.

Effectively the equations (7) and (8) are:

$$CV_{predicted} + A * \Delta MV(t + c) = CV_{min} + S_{min} - S^- \quad (9)$$

$$CV_{predicted} + A * \Delta MV(t + c) = CV_{max} - S_{max} + S^+ \quad (10)$$

To get the LP solution within ranges or to minimally exceed the ranges, the new slack variables should be penalized and penalties should be significantly higher than economic costs or profits.

Therefore the objective function (4) should be extended by adding the terms  $PS_-^T * S^-$  and  $PS_+^T * S^+$ .

$$Q_{min} = -UCV^T * \Delta CV(t + p) + UMV^T * \Delta MV(t + c) + PS_-^T * S^- + PS_+^T * S^+ \quad (11)$$

where

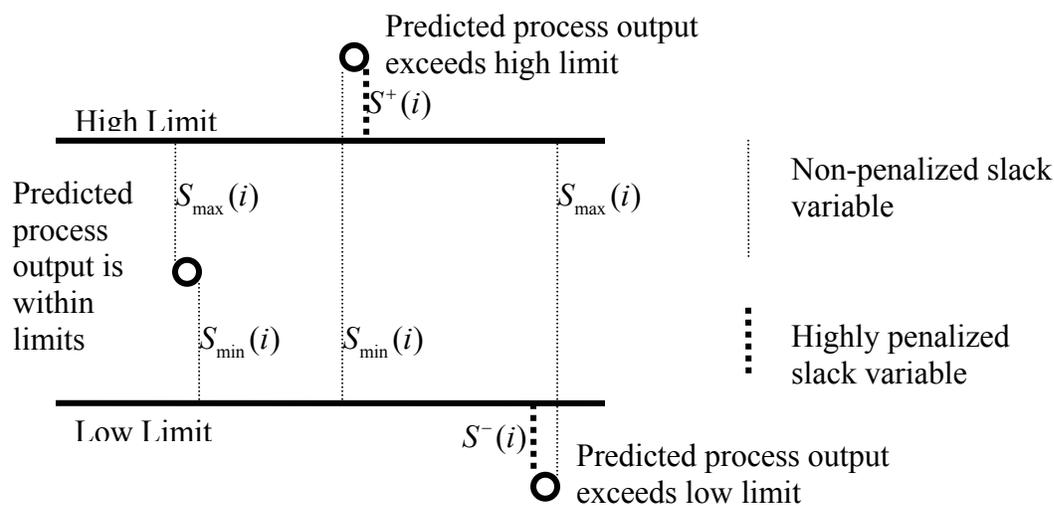
$PS_-$  is the penalty vector for violating low limits

$PS_+$  is the penalty vector of violating high limits

$PS_- \square UCV$  and  $PS_+ \square UCV$

Vector  $PS_-$  and  $PS_+$  should have all components significantly larger than economic cost/profit vectors. It is reasonable to assume in general that the smallest component of the vectors  $PS_-$  and  $PS_+$  should be several times greater than the largest component of the  $UCV$  vector.

Figure 2 illustrates the slack variables concept for the constraint variables (with no set point).  $S(i)$  is the component  $i$  of the relevant slack vector.

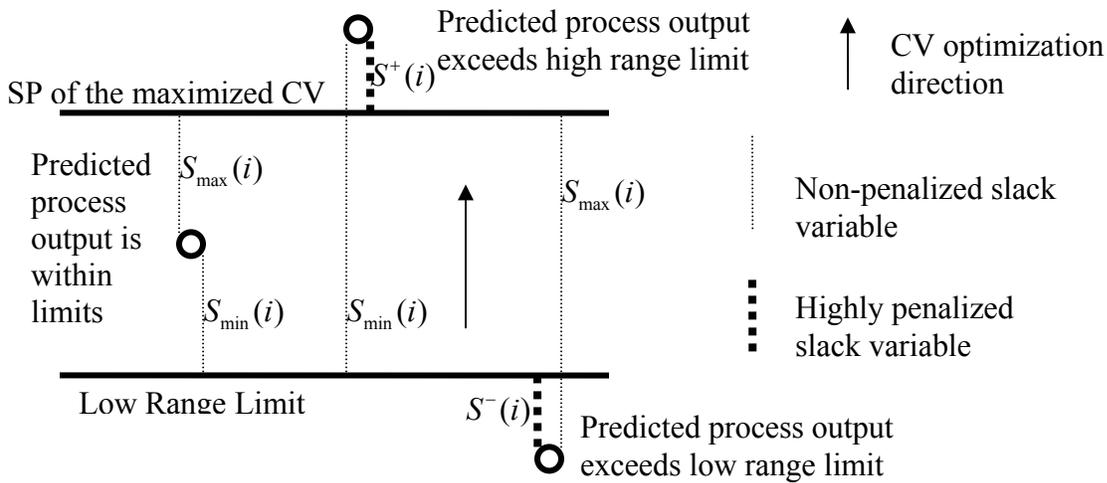


**Figure 2. Slack Variables Application for the Constraint Handling**

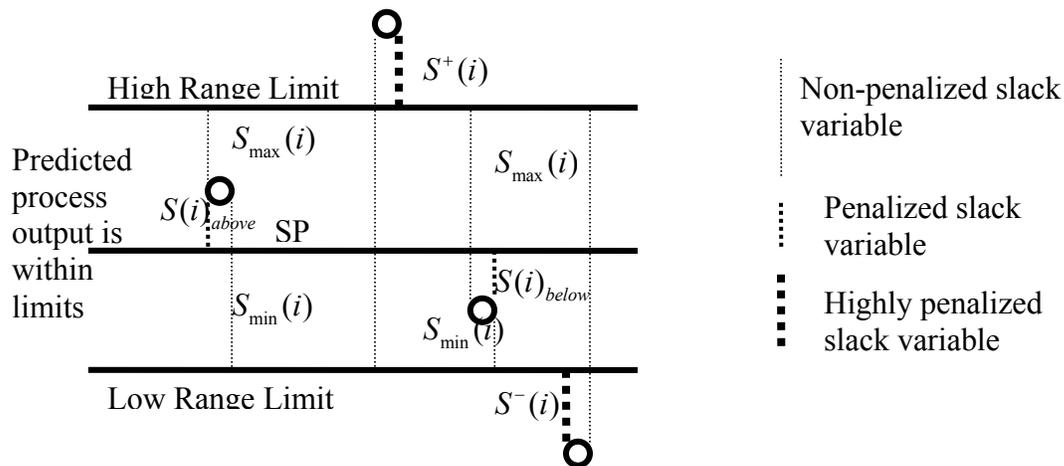
An extension of the known slack variable application for MPC optimization is achieved by applying slack variables to the set points optimization within acceptable ranges.

The ranges are defined around set points within high and low set point limits. Ranges can be single sided or two sided.

One sided ranges are associated with minimize and maximize objective functions (Figure 3), while two sided ranges have no economic objectives, other than obtaining the optimal solution as close as possible to the set point within the extended range on the penalized slack variables (Figure 4). If the range is equal to 0, then we have set point with the penalized slack variables around it.



**Figure 3. Slack Variables Application to the Maximized Single Sided Range Control**



**Figure 4. Slack Variables Application to the Two-Sided Range Control**

The equations for the setpoint control with the two-sided ranges are in the form:

$$CV_{predicted} + A * \Delta MV(t+c) = SP - S_{below} + S_{above}$$

$$CV_{predicted} + A * \Delta MV(t + c) = CV_{min} + S_{min} - S^-$$

$$CV_{predicted} + A * \Delta MV(t + c) = CV_{max} - S_{max} + S^+$$

$S_{below}$  and  $S_{above}$  are vectors of slack variables for the solutions below and above the setpoints. The terms  $PSP_{below}^T * S_{below} + PSP_{above}^T * S_{above}$  should be added to the objective function.

Where

$PSP_{below}^T$  is the unit penalty for the solution below the setpoint

$PSP_{above}^T$  is the unit penalty for the solution above the setpoint.

The outlined concept of the constraints handling with penalized slack variables provides significant flexibility in handling infeasible situations. Applying penalized slack variables, the following functionality is achieved:

1. The optimizer can always find a solution, even if the solution is outside the output limits.
2. Some outputs which are within limits prior to running the optimizer, may be out of limits as a result of the solution
3. The amount of limit excess for a particular output is not quantitatively defined prior to the solution.

Items 2 and 3 may not be desirable in many applications. Some applications may require that optimizer not drive lower priority outputs which are within range limits out of limits, to bring higher priority outputs within limits. Also it may be required as well to have well defined solution limits for the all the outputs. The technique outlined in the next section satisfies these requirements.

## INFEASIBILITY HANDLING BY REDEFINING CONSTRAINT MODEL

To handle infeasibilities with strictly defined degree of limits violation, the constraint model is redefined. Redefining takes place after the first optimization run if penalized slack variables are not used and there is no solution within original limits or penalized slack variables are used, but the solution with penalized slack variables is not acceptable.

If there is no solution and high limit  $CV^{HL}$  is exceeded, then new limits for CV is defined as:

$$CV^{HL'} = CV^{prediction}$$

$$CV^{LL'} = CV^{HL} - \Delta$$

$\Delta = 1 - 3\%$  to avoid solution exactly at the original limit.

Similarly when low limit  $CV^{LL}$  is exceeded the new limit will be:

$$CV^{LL'} = CV^{prediction}$$

$$CV^{HL'} = CV^{LL} + \Delta$$

Figure 5 illustrates the concept.

After redefining the constraint handling model, the penalty vector for all CVs out of the ranges or limits should be recalculated in such a way as to drive those CV in the direction of the exceeded limits. To achieve this objective, the penalty for limits violation should override economic criteria.

A general form of LP objective function:

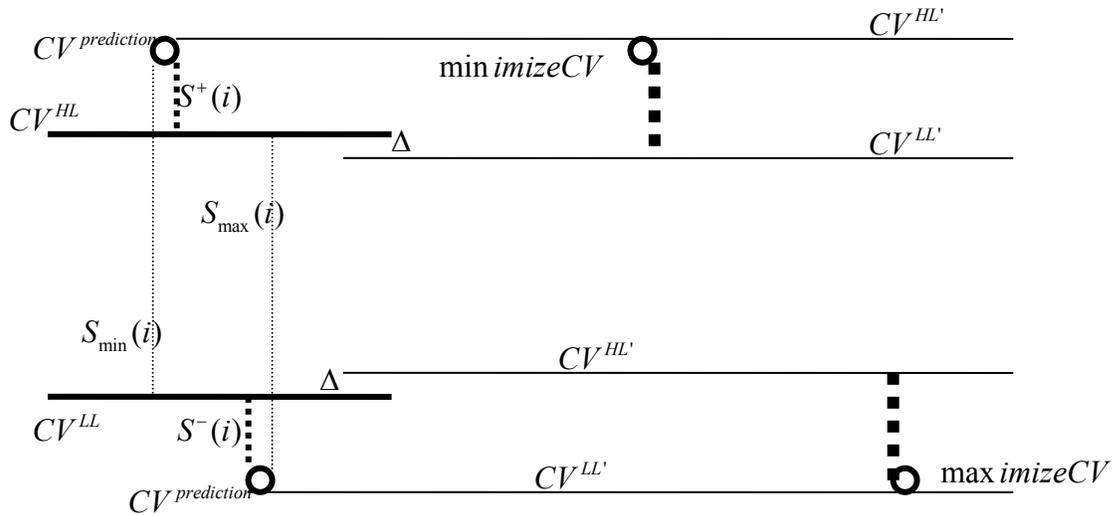
$$(P^T * A + C^T) * (MV^t - MV^{t-1}) \tag{12}$$

has the cost of the process outputs expressed by the vector:

$$P^T * A = [p_1, \dots, p_i, \dots, p_n] \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix} = \left[ \sum_{i=1}^n p_i a_{i1}, \sum_{i=1}^n p_i a_{i2}, \dots, \sum_{i=1}^n p_i a_{im} \right] \quad (13)$$

The resulting vector is:

$$CM^T = C^T + P^T * A = [C_1, C_2, \dots, C_m] + \left[ \sum_{i=1}^n p_i a_{i1}, \sum_{i=1}^n p_i a_{i2}, \dots, \sum_{i=1}^n p_i a_{im} \right] = [CM_1, CM_2, \dots, CM_m] \quad (14)$$



**Figure 5. Infeasibility Handling by Redefining Constraint Model**

To make constraint handling a priority, an additional penalty  $v_i$  for the constraint violate outputs should be defined as a negative value, if CV exceeds high limit, and a positive value, if CV exceeds low limit. A contribution of the additional penalty  $v_i$  to the cost vector will be:

$$[v_i a_{i1}, v_i a_{i2}, \dots, v_i a_{im}] = [V_1^i, V_2^i, \dots, V_m^i] \quad (15)$$

For the constraints handling to take precedence over economics, this vector should have every component greater than vector  $CM^T$ . Therefore:

$$|v_i a_{ij}| \geq (|CM_j| + 1) \quad j=1, 2, \dots, m \quad (16)$$

$r_i$  is priority/rank number of redefined CV

$r_{\max}$  is the maximum priority/rank number for the lowest priority/rank

$r_{\min}$  is the minimum priority/rank number for the highest priority/rank

Calculations can be simplified by using the high estimate of  $|v_i|$  for the highest priority as:

$$|v_i| = \frac{\max_j (|CM_j| + 1) r_{\max}}{\min_{i,j} |a_{ij}| r_{\min}} = \nu \quad |a_{ij}| > .05 \quad i=1,2,\dots,n \quad (17)$$

For practical purposes it is assumed  $|a_{ij}| > .05$  to exclude extremely low process gains from the calculations. After computing  $\nu$ , the penalty for all CVs out of the limits or out of the ranges is adjusted depending on CV priority.

$$|v_i| = \nu \left( 1 - \frac{r_i - r_{\min}}{r_{\max}} \right)$$

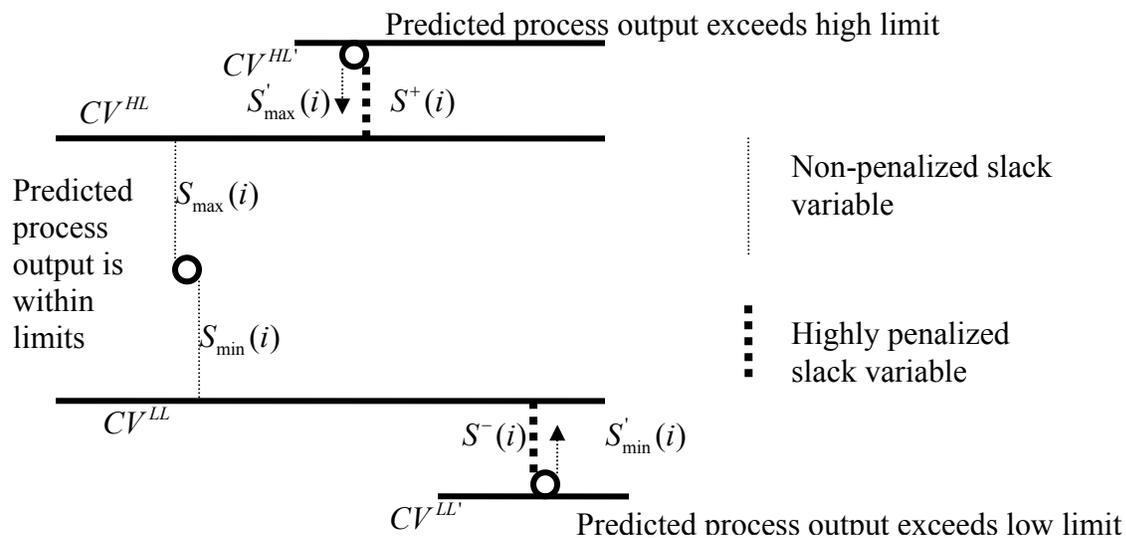
After calculating costs for a particular CV exceeding constraints according to equation (17), the total cost of all constraints for all manipulated variables is calculated from the equation (13).

The procedure for calculating penalty should be applied sequentially for all CVs violating constraints by applying equation (17) starting from the lowest priority violated constraint (greatest  $r_i$ ). For practical purposes the effective penalty vector should be normalized. One way to do this is to divide all vector components by a maximum component and multiply by 100.

## INTEGRATED CONCEPT OF CONSTRAINTS HANDLING

The concept of constraints handling can be extended by integrating two approaches of the penalized slack variables and model redefinition. When there are no constraints violations or optimal solutions with penalized slack variable it is acceptable that only the penalized slack variables be used. When constraints are violated and a solution with penalized slack variables is not accepted, then process output limits are redefined.

The new output limits become equal to the predictions for the predicted process outputs violating limits – Figure 6. The original limit is used for defining penalized slack variable as in the previously described slack variable application.



**Figure 6. Integrated Concept of Handling Constraints on the Process Output.**

The following equations are needed for this type of constraint handling.

$$CV_{predicted} + A * \Delta MV(t+c) = CV^{LL} + S_{min} - S^- \quad (18)$$

$$CV_{predicted} + A * \Delta MV(t+c) = CV^{HL} - S_{max} + S^+ \quad (19)$$

$$CV_{predicted} + A * \Delta MV(t+c) = CV^{LL'} + S'_{min} \quad (20)$$

$$CV_{predicted} + A * \Delta MV(t+c) = CV^{HL'} - S'_{max} \quad (21)$$

Values  $CV^{LL'}$  and  $CV^{HL'}$  are low and high limits which cannot be exceeded by the optimizer. The limit values are set as out of limit CV predictions or out of limit values with a wider range than current predictions violating limits.

Equations for the CV range control and two sided CV range control can be developed in a similar fashion. Equations for single sided range control are identical to (18) - (21). Two sided range control is amended by the equation:

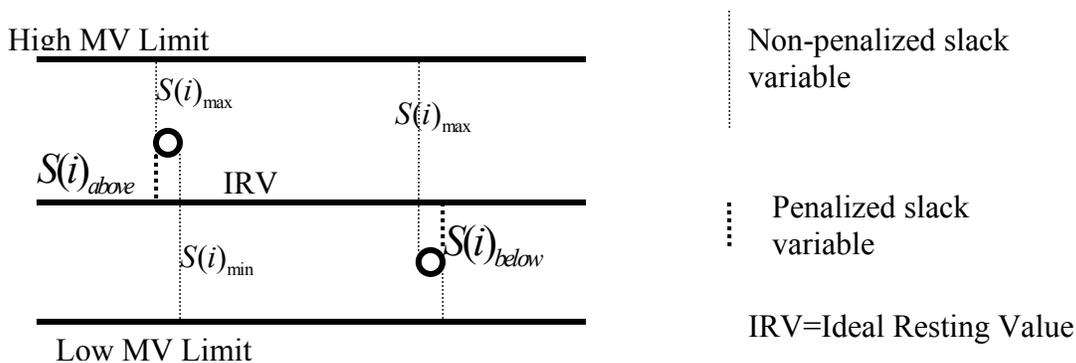
$$CV_{predicted} + A * \Delta MV(t+c) = SP - S_{below} + S_{above} \quad (22)$$

In addition the approach allows handling input constraints in a more flexible way. Currently, inputs have only hard constraints defined. Using the same approach, it is possible to define soft constraints for some inputs, contained within hard constraints. Introducing penalized slack variables for the soft constraints, it is easy to define penalized range for the MV. The additional equations to the equation (2) should be included into the model:

$$MV_{min}^{soft} - S_{soft\ min}^- = MV_{current} + \Delta MV(t+c) \quad (23)$$

$$MV_{max}^{soft} + S_{soft\ max}^+ = MV_{current} + \Delta MV(t+c) \quad (24)$$

Finally the same approach may be used to the so called Ideal Resting Value (IRV) - the optimizer preferred input value, as shown in Figure 7. IRV can be a value defined by the user or it can be the last MV value in MPC Manual mode, prior to entering Auto mode.



**Figure 7. Using penalized slack variables to account for the Ideal Resting Value**

The equations for MV with IRV are:

$$IRV - S_{below} + S_{above} = MV_{current} + \Delta MV(t+c) \quad (25)$$

$$MV_{\min} + S_{\min} = MV_{\text{current}} + \Delta MV(t+c) \quad (26)$$

$$MV_{\max} - S_{\max} = MV_{\text{current}} + \Delta MV(t+c) \quad (27)$$

The components of the penalized slack vectors  $S^-$ ,  $S^+$ ,  $S_{\text{below}}$ ,  $S_{\text{above}}$  are set based on variable priorities or variable unit costs in the objective function or other factors like degree of limit violation.

## OPTIMIZER USER INTERFACE

The fractionator model was used in simulation. This is a commonly used reference model for MPC testing known as the Shell model. The tested optimizer applied the model redefinition technique. The screen capture in Figure 8 presents the operator screen and Figure 9 the optimizer window.

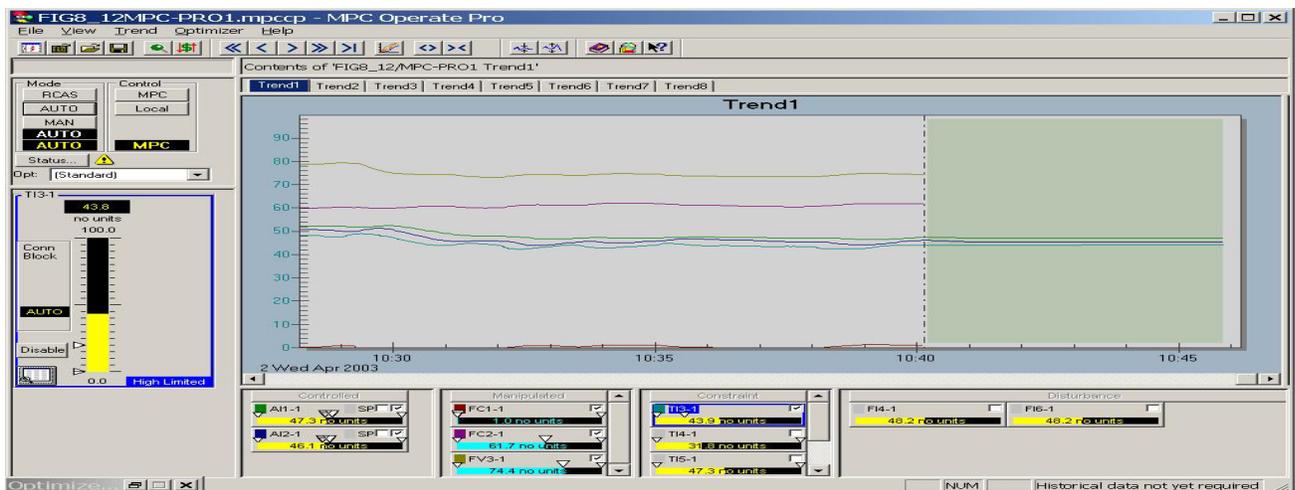


Figure 8. Operator Screen for MPC Integrated with Optimizer

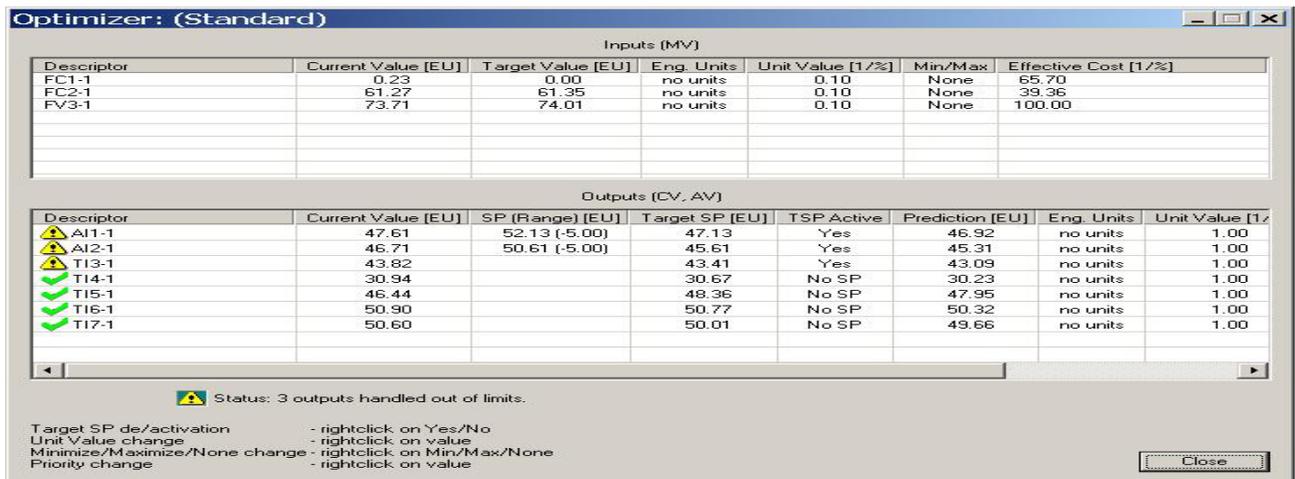


Figure 9. Optimizer Window

Constraints violation was achieved by applying high level disturbances or by setting low and high limits on the inputs and outputs violated by their actual values. The tests confirmed the assumed property of the technique, in particular the optimizer acted properly and effectively to improve the high priority constraint violation with no further violation of lower priority constraints.

## CONCLUSIONS

Handling infeasible LP solutions by applying slack variables and model redefinition is an extremely flexible and effective approach.

Applying the presented principles of constraints handling is an easy way to develop a number of modifications of the constraint models to satisfy more specific requirements. The approach may be used for other real time optimization applications that do not use MPC such as gasoline blending.

## REFERENCES

1. Qin, S. J. and Badgwell, T. A., "An Overview of Industrial Model Predictive Control Technology," *Fifth International Conference on Chemical Process control*, pages 232-256, AIChE and CACHE, 1997.
2. Tyler, M.L. and Morari M., "Propositional Logic in Control and Monitoring Problems," In *Proceedings of European Control Conference '97*, pages 623-628, Bruxelles, Belgium, June 1997.
3. Vada, J., Slupphaug, O. and Foss, B.A., "Infeasibility Handling in Linear MPC subject to Prioritized Constraints," In *PreprintsIFAC'99 14<sup>th</sup> World Congress*, Beijing, China, July 1999.
4. Vada, J., Slupphaug, O. and Johansen, T.A., "Efficient Infeasibility Handling in Linear MPC subject to Prioritized Constraints," In *ACC2002 Proceedings*, Anchorage, Alaska, May 2002.
5. Wojsznis, W., T., Thiele D., Wojsznis, P., and Ashish Mehta, "Integration of Real Time Process Optimizer with a Model Predictive Function Block," *ISA Conference*, Chicago, October 2002.
6. William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, "Numerical Recipes in C," *Cambridge University Press*, 1997.
7. Wojsznis, W., Gudaz, J., Blevins, D., and Mehta, A., "Practical Approach to Tuning MPC," *ISA Transactions*, April 2002.
8. <http://easydeltav.com/keytechnologies/index.asp>