Proceeding of the 2004 American Control Conference
Boston, Massachusetts June 30 - July 2, 2004

Model Predictive Control – New tools for design and evaluation

Alberto Bemporad
Dip. Information Engineering
University of Siena
bemporad@dii.unisi.it

N. Lawrence Ricker
Department of Chemical Engineering
University of Washington
ricker@u.washington.edu

James Gareth Owen
The MathWorks, Inc.
jowen@mathworks.com

Abstract— **A new version of the Model Predictive Control Toolbox for MATLAB is described. Major improvements include more flexible modeling of plant and disturbance characteristics, and support for design and simulation involving nonlinear (Simulink) models.**

I. Introduction

Model Predictive Control (MPC) was popularized in the 1970s for control of petroleum refinery operations, which often operate at constraints on manipulated variables (e.g., valve saturation) or controlled variables (e.g., maximum catalyst temperature). Since then, MPC has become the benchmark for complex constrained multivariable control problems in the process industries.

Briefly, at each sampling time, – starting at the current state – an open-loop optimal control problem is solved over a finite horizon. At the next time step, the computation is repeated starting from the new state and over a shifted horizon, leading to a moving horizon policy. The solution relies on a linear dynamic model, respects specified constraints, and optimizes a quadratic performance index. Thus, provided that the model is accurate and a quadratic performance index and linear inequality constraints express true performance objectives, MPC provides near-optimal performance.

Over the last decade, a solid theoretical foundation for MPC has emerged so that large-scale MIMO controllers with non-conservative stability guarantees can be designed routinely [1]. Recent developments facilitate applications requiring high sampling rates [2].

Software for MPC design and implementation has developed as the technology has matured. There are two "flavors": 1) tools that work in conjunction with a proprietary real-time industrial control system such as DMCPlus [1] 2) tools intended primarily for analysis and prototyping. An example of the latter is the MPC Toolbox for MATLAB, which appeared in 1995 [3], and currently has over 1000 licensees worldwide. The original (V1.0) MPC Toolbox was command-driven, difficult to use in conjunction with nonlinear process simulations, and could not be applied to an experimental system. This paper describes the V2.0 MPC Tools release, which provides substantial additional capabilities and better ease-of-use.

[1]Trade Mark of AspenTech

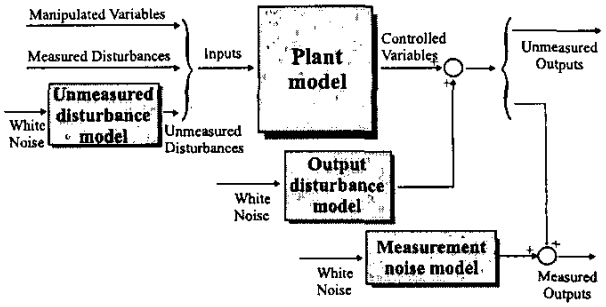

Fig. 1. Model used for prediction, optimization, and estimation

II. MPC Algorithm

The core MPC Toolbox algorithm is based on a model of the system to be controlled, a performance index driving the selection of the decision variables, a set of constraints to be fulfilled, and a state estimator to reconstruct the model's internal states.

The model used in the MPC Toolbox for prediction / optimization, and for state estimation is depicted in Fig.1. The model used for prediction / optimization is a sampled linear time invariant (LTI) system, which need not be open-loop stable, consisting of the model of the plant to be controlled, whose inputs are the manipulated variables, the measured disturbances, and the unmeasured disturbances, and a model generating the unmeasured disturbances.

A. Prediction Model

The model of the plant is an LTI discrete-time system described by the equations

$$\begin{cases} x(k+1) &= Ax(k) + B_u u(k) + B_v v(k) + B_d d(k) \\ y_m(k) &= C_m x(k) + D_{vm} v(k) + D_{dm} d(k) \\ y_u(k) &= C_u x(k) + D_{vu} v(k) + D_{du} d(k) + D_{uu} u(k) \end{cases} \quad (1)$$

where $x(k) \in \mathbb{R}^{n_x}$ is the plant state, $u(k) \in \mathbb{R}^{n_u}$ is the vector of manipulated variables (MV), i.e., the command inputs, $v(k) \in \mathbb{R}^{n_v}$ is the vector of measured disturbances (MD), $d(k) \in \mathbb{R}^{n_d}$ is the vector of unmeasured disturbances (UD) entering the plant, $y_m(k) \in \mathbb{R}^{n_{ym}}$ is the vector of measured outputs (MO), and $y_u(k) \in \mathbb{R}^{n_{yu}}$ is the vector of unmeasured outputs (UO). The overall output vector $y(k) \in \mathbb{R}^{n_y}$ includes $y_m(k)$ and $y_u(k)$. In the above equations $d(k)$ includes both state disturbances ($B_d \neq 0$)

0-7803-8335-4/04/$17.00 ©2004 AACC

and output disturbances ($D_d \neq 0$). A valid plant model for the MPC Toolbox cannot have direct feed-through of MVs on MOs, an hypothesis satisfied in most applications. The unmeasured disturbance $d(k)$ is modeled as the output of the LTI system

$$\begin{cases} x_d(k+1) &= \tilde{A}x_d(k) + \tilde{B}n_d(k) \\ d(k) &= \tilde{C}x_d(k) + \tilde{D}n_d(k) \end{cases} \tag{2}$$

System (2) is driven by the random Gaussian noise $n_d(k)$, having zero mean and a unit covariance matrix. For instance, a step-like unmeasured disturbance is modeled as the output of an integrator.

In many practical applications, the plant model matrices A, B, C, D are obtained by linearizing the nonlinear dynamics

$$\begin{cases} x' &= f(x,u,v,d) \\ y &= h(x,u,v,d) \end{cases} \tag{3}$$

at some nominal value $x = x_0$, $u = u_0$, $v = v_0$, $d = d_0$. In (3) x' denotes either the time derivative (continuous time model) or the successor $x(k+1)$ (discrete time model). The linearized model has an affine form

$$\begin{cases} x' &\approx f(x_0,u_0,v_0,d_0) + \nabla_x f(x-x_0) \\ & \quad + \nabla_u f(u-u_0) + \nabla_v f(v-v_0) + \nabla_d f(d-d_0) \\ y &\approx h(x_0,u_0,v_0,d_0) + \nabla_x h(x-x_0) \\ & \quad + \nabla_u h(u-u_0) + \nabla_v h(v-v_0) + \nabla_v h(d-d_0) \end{cases} \tag{4}$$

where Jacobians are evaluated at x_0, u_0, v_0, d_0. The matrices A, B, C, D of the model are readily obtained from the Jacobian matrices appearing in (4). The linearized dynamics are affected by the constant terms $F = f(x_0,u_0,v_0,d_0)$ and $H = h(x_0,u_0,v_0,d_0)$. For this reason the MPC algorithm internally adds a measured disturbance $v = 1$, so that F and H can be embedded into B_v and D_v, respectively, as additional columns.

B. Performance and Constraint Specifications

Assume that estimates of $x(k)$, $x_d(k)$ are available at time k (cf. Section II-C). The MPC control action at time k is obtained by solving the optimization problem

$$\min_{\substack{\Delta u(k|k) \\ \vdots \\ \Delta u(m-1+k|k) \\ \epsilon}} \sum_{i=0}^{p-1} \left(\sum_{j=1}^{n_y} |w^y_{i+1,j} [y_j(k+i+1|k) - r_j(k+i+1)]|^2 + \sum_{j=1}^{n_u} |w^{\Delta u}_{i,j} \Delta u_j(k+i|k)|^2 + \sum_{j=1}^{n_u} |w^u_{i,j} \cdot [u_j(k+i|k) - u_{\text{target},j}(k+i)]|^2 \right) + \rho_\epsilon \epsilon^2 \tag{5a}$$

where the subscript "$()_j$" denotes the jth component of a vector, "$(k+i|k)$" denotes the value predicted for time $k+i$ based on the information available at time k; $r(k)$ is the current sample of the output reference, subject to

$$\begin{aligned} & u_j^{\min}(i) - \epsilon V_j^{\text{u,min}}(i) \leq u_j(k+i|k) \leq \\ & \qquad u_j^{\max}(i) + \epsilon V_j^{\text{u,max}}(i),\ j = 1,\ldots,n_u \\ & \Delta u_j^{\min}(i) - \epsilon V_j^{\Delta\text{u,min}}(i) \leq \Delta u_j(k+i|k) \leq \\ & \qquad \Delta u_j^{\max}(i) + \epsilon V_j^{\Delta\text{u,max}}(i),\ j = 1,\ldots,n_u \\ & y_j^{\min}(i) - \epsilon V_j^{\text{y,min}}(i) \leq y_j(k+i+1|k) \leq \\ & \qquad y_j^{\max}(i) + \epsilon V_j^{\text{y,max}}(i),\ j = 1,\ldots,n_y \\ & \Delta u(k+h|k) = 0 \\ & \epsilon \geq 0 \end{aligned} \tag{5b}$$

for all $i = 0,\ldots,p-1$, $h = m,\ldots,p$, with respect to the sequence of input increments $\{\Delta u(k|k),\ldots,\Delta u(m-1+k|k)\}$ and the slack variable ϵ. The MPC controller sets $u(k) = u(k-1) + \Delta u^*(k|k)$, where $\Delta u^*(k|k)$ is the first element of the optimal sequence. Note that although only the measured output vector $y_m(k)$ is fed back to the MPC controller, $r(k)$ is a reference for all the outputs (measured and unmeasured).

When the reference r is not known in advance, the current reference $r(k)$ is used over the whole prediction horizon, namely $r(k+i+1) \equiv r(k)$ in (5a). In model predictive control, the exploitation of future references in MPC is referred to as *anticipative action*. A similar anticipative action can be performed with respect to the measured disturbance $v(k)$, if this is known in advance. In the prediction, $d(k+i)$ is instead obtained by setting $n_d(k+i) \equiv 0$ in (2).

The scalars $w^y_{i,j}$, $w^u_{i,j}$, $w^{\Delta u}_{i,j}$ are nonnegative weights for the the corresponding variable. The smaller w, the less important is the behavior of the corresponding variable to the overall performance index. Instead, $u_j^{\min}$, $u_j^{\max}$, $\Delta u_j^{\min}$, $\Delta u_j^{\max}$, $y_j^{\min}$, $y_j^{\max}$ are lower/upper bounds on the corresponding variables. In (5b), the constraints on u, Δu, and y are relaxed by introducing the "slack" variable $\epsilon \geq 0$. The weight ρ_ϵ penalizes the violation of the constraints. The larger ρ_ϵ with respect to input and output weights, the more the constraint violation is penalized. The Equal Concern for the Relaxation (ECR) vectors $V^{\text{u,min}}$, $V^{\text{u,max}}$, $V^{\Delta\text{u,min}}$, $V^{\Delta\text{u,max}}$, $V^{\text{y,min}}$, and $V^{\text{y,max}}$ have nonnegative entries which represent the concern for relaxing the corresponding constraint; the larger V, the softer the constraint. $V = 0$ means that the constraint is hard and cannot be violated. Hard output constraints may cause infeasibility of the optimization problem (for instance, because of unpredicted disturbances, model mismatch). By default, all input constraints are hard, all output constraints are soft, and $\rho_\epsilon = 10^5 \max_{i,j}\{w^u_{i,j}, w^{\Delta u}_{i,j}, w^y_{i+1,j}\}$.

Vector $u_{\text{target}}(k+i)$ is a set-point for the input vector. One typically uses u_{target} if the number of inputs is greater than the number of outputs.

As mentioned earlier, only $\Delta u(k|k)$ is actually used to compute $u(k)$. The remaining samples $\Delta u(k+i|k)$ are discarded, and a new optimization problem based on $y_m(k+1)$ is solved at the next step, $k+1$.

The algorithm implemented in the MPC Toolbox uses different procedures depending on the presence of constraints.

If all the bounds are infinite, then the slack variable ϵ is removed, and problem (5) is solved analytically. Otherwise a Quadratic Programming (QP) solver is used. The matrices associated with the quadratic optimization problem are described in [3].

C. State Estimation

As the true states $x(k)$, $x_d(k)$ are not available to the controller, predictions are obtained from a state estimator. In order to provide maximum flexibility, the estimator is based on the model depicted in Fig. 1.

1) Measurement Noise Model: We assume that the measured output vector $y_m(k)$ is corrupted by measurement noise $m(k)$. The measurement noise $m(t)$ is the output of the LTI system

$$\begin{cases} x_m(k+1) &= \tilde{A}x_d(k) + \tilde{B}n_m(k) \\ m(k) &= \tilde{C}x_m(k) + \tilde{D}n_m(k) \end{cases} \quad (6)$$

System (6) is driven by random Gaussian noise $n_m(k)$, having zero mean and unit covariance matrix.

Note that the objective of the MPC controller is to bring $y_u(k)$ and $y_m(k) - m(k)$ as close as possible to the reference vector $r(k)$. For this reason, the measurement noise model producing $m(k)$ is not needed in the prediction model used for the optimization in (5).

2) Output Disturbance Model: In order to guarantee asymptotic rejection of output disturbances, the overall model is augmented by an output disturbance model. By default, in order to reject constant disturbances due for instance to gain nonlinearities, the output disturbance model is a collection of integrators driven by white noise on measured outputs. Output integrators are added according to the following rule:

1) Measured outputs are ordered by decreasing output weight
2) By following such order, an output integrator is added per measured output unless there is a violation of observability, or the corresponding output weight is zero.

Alternatively, an arbitrary output disturbance model can be specified.

D. State Observer

The state observer is designed to provide estimates of $x(k)$, $x_d(k)$, $x_m(k)$, where where $x(k)$ is the state of the plant model, $x_d(k)$ is the overall state of the input and output disturbance model, $x_m(k)$ is the state of the measurement noise model. The estimates are computed from the measured output $y_m(k)$ by the linear state observer

$$\begin{bmatrix} \hat{x}(k|k) \\ \hat{x}_d(k|k) \\ \hat{x}_m(k|k) \end{bmatrix} = \begin{bmatrix} \hat{x}(k|k-1) \\ \hat{x}_d(k|k-1) \\ \hat{x}_d(k|k-1) \end{bmatrix} + M[y_m(k) - C_m\hat{y}(k)]$$

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \hat{x}_d(k+1|k) \\ \hat{x}_m(k+1|k) \end{bmatrix} = \begin{bmatrix} A\hat{x}(k|k)+B_u u(k)+B_v v(k)+B_d\tilde{C}\hat{x}_d(k|k) \\ \tilde{A}\hat{x}_d(k|k) \\ \tilde{A}\hat{x}_m(k|k) \end{bmatrix}$$

$$y_m(k) = C_m\hat{x}(k|k-1) + D_{vm}v(k) + D_{dm}\tilde{C}\hat{x}_d(k|k-1) + \tilde{C}\hat{x}_m(k|k-1)$$

where "$_m$" denotes the rows of C,D corresponding to measured outputs. To prevent numerical difficulties in the absence of unmeasured disturbances, the gain M is designed using Kalman filtering techniques on the extended model

$$\begin{bmatrix} x(k+1) \\ x_d(k+1) \\ x_m(k+1) \end{bmatrix} = \begin{bmatrix} A & B_d\tilde{C} & 0 \\ 0 & \tilde{A} & 0 \\ 0 & 0 & \tilde{A} \end{bmatrix} \begin{bmatrix} x(k) \\ x_d(k) \\ x_m(k) \end{bmatrix} + \begin{bmatrix} B_u \\ 0 \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} B_v \\ 0 \\ 0 \end{bmatrix} v(k) + \begin{bmatrix} B_d\tilde{D} & 0 & B_u & B_v \\ \tilde{B} & 0 & 0 & 0 \\ 0 & \tilde{B} & 0 & 0 \end{bmatrix} \begin{bmatrix} n_d(k) \\ n_m(k) \\ n_u(k) \\ n_v(k) \end{bmatrix} +$$

$$y_m(k) = \begin{bmatrix} C_m & D_{dm}\tilde{C} & \tilde{C} \end{bmatrix} \begin{bmatrix} x(k) \\ x_d(k) \\ x_m(k) \end{bmatrix} + D_{vm}v(k) + \begin{bmatrix} D_{dm} & \tilde{D} & 0 & 0 \end{bmatrix} \begin{bmatrix} n_d(k) \\ n_m(k) \\ n_u(k) \\ n_v(k) \end{bmatrix} \quad (7)$$

where $n_u(k)$ and $n_v(k)$ are additional unmeasured white noise disturbances having unit covariance matrix and zero mean, that are added on the vector of manipulated variables and the vector of measured disturbances, respectively, to ease the solvability of the Kalman filter design.

The overall state-space realization of the combination of plant and disturbance models must be observable for the state estimation design to succeed.

III. Case Study: MPC Supervisory Control of a Two Stage Thermo-mechanical Pulping Process

A. Process Background

Thermo-mechanical pulping (TMP) is the most common process in North America for producing mechanical pulp for newsprint. Fig. 2 shows the typical process arrangement for a two stage TMP operation. Two pressured refiners operate in sequence. The primary refiner produces a coarse pulp from a feed of wood chips and water. The secondary refiner further develops the pulp bonding properties so that it is suitable for paper making. The refiners consist of two disks (either contra-rotating or one static and the other rotating) with overlaid grooved surfaces. These surfaces impact on a three phase flow of wood fibers, steam and water that passes from the center of the refiner disks to their periphery. The impact of the disk surfaces on the wood fibers: i.) breaks rigid chemical and physical bonds between them; ii.) microscopically roughens the surface of individual fibers enabling them to mesh together on the paper sheet. The primary objective of controlling the TMP plant is to apply sufficient energy to derive pulp with good physical properties without incurring excess energy costs or fiber damage. In practice, this reduces to control of the total electrical energy applied per mass of dry wood fibers, i.e., the specific energy applied to the pulp. A secondary control objective is to hold the ratio of dry mass flow rate (fibers) to overall mass flow rate (water & fibers) (i.e., pulp consistency) at a desired value. The process schematic shown in Fig. 2 illustrates the I/O for a TMP supervisory control system:

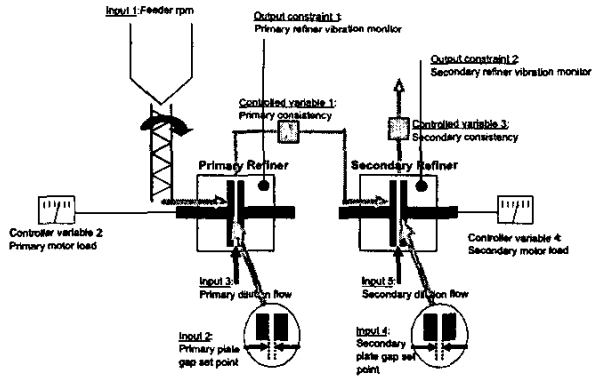

Fig. 2. Process schematic illustrating TMP control application

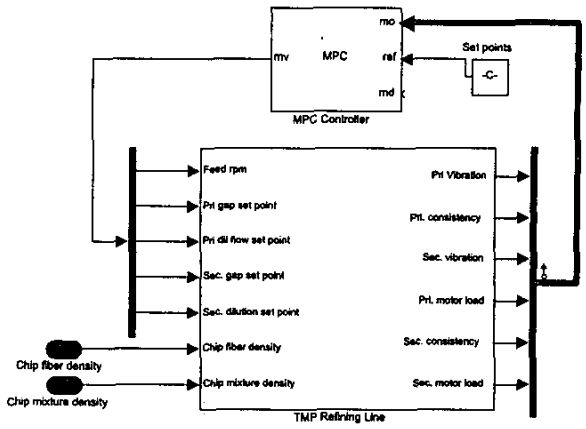

Fig. 3. Simulink model of the TMP control application

manipulated variables (5): feed rate of chips (screw feeder rpm), dilution water flow to each of the refiners, set points to two regulatory controllers that control the gap between the rotating disks in each set of refiners

unmeasured disturbance inputs (2): fiber filling factor and fiber-water filling factor

measured outputs (6): primary and secondary refiner consistencies, primary and secondary refiner motor loads and vibration monitor measurements on the two refiners (included for output constraint purposes only).

The TMP process is difficult to control using the classical multi loop method due to the interaction between the variables [4]. For example, the following actions will cause similar effects in differing ratios: Increasing the feed rate increases the power applied on both refiners at constant gap (due to the higher stresses between the refiner plates) and elevates the pulp consistency due to increased flow rate of bone dry mass of fibers; Closing the gap increases refiner power due to higher stresses and elevates consistency as a result of increased water evaporation to steam. Dilution flow changes also cause multiple effects. Moreover, the process is usually operated under multiple constraints: manipulated variables are often subject to active physical constraints, especially when the process is run close to its maximum capacity. Depending on the ratings of the refiner motors, there may also be active output constraints on the power applied by each. A key issue is the need to avoid refiner plate clash, an event that causes plate surface wear and damages pulp fibers. In this case study, we shall assume that plate clashing is prevented by maintaining the measured vibration levels on each refiner (measured outputs) below some critical value.

B. Building a Process Model from Simulink [2]

The MPC toolbox requires the model used in controller design to be affine, i.e., a linear, time-invariant (LTI) system describing deviations from a nominal condition. A model derived from a system identification experiment is usually affine by construction. However, models derived from physical principles (such as the Simulink TMP process in this case study) are frequently non-affine. Such a Simulink model must be linearized before it can be used as the basis for an MPC design. Conceptually this is a two step process: The operating condition represented by x_0, u_0, v_0, d_0 in (3) must be determined for the non linear state equations represented by the Simulink model; The Jacobian matrices $\nabla_x f, \nabla_d f, \nabla_u f, \nabla_v f$ describing perturbations from that operating condition must be extracted in the form of an LTI model. The Simulink/MPC Toolbox combination automates these procedures.

1) Finding the operating condition: When performing linearization, Simulink requires knowledge of the entire Simulink model state (block interconnections prevent isolation of sub-models in general). There are two ways to specify the nominal model state: i.) Simulation to steady state followed by extraction of the model's terminal conditions; or, ii.) Specification of target values for a subset of the model's variables followed by calculation of the corresponding steady state. Fig. 4 shows an annotated screen shot of the MPC Toolbox user interface displaying the nominal input and output values determined using method (ii).

2) Plant model derivation: Once a model operating condition has been specified, Simulink requires the user to delineate the subsection to be linearized. If the goal is to create a plant model for use in the MPC Toolbox, the linearization tool needs to map each component of the vectors u (plant input) and y (plant output) to signals in the Simulink model.

When there is an MPC block (as depicted in Fig. 3), the natural choice is to associate plant inputs u with the MPC block output port and plant outputs y with the MPC block's 'measured outputs' port. The MPC Toolbox linearization

[2]The Simulink model of the TMP process shown in Fig. 3 was based on a model provided by Sylvain Gendron at the Pulp and Paper Research Institute of Canada (PAPRICAN). The model is available as a demo in the MPC Toolbox or directly from the authors

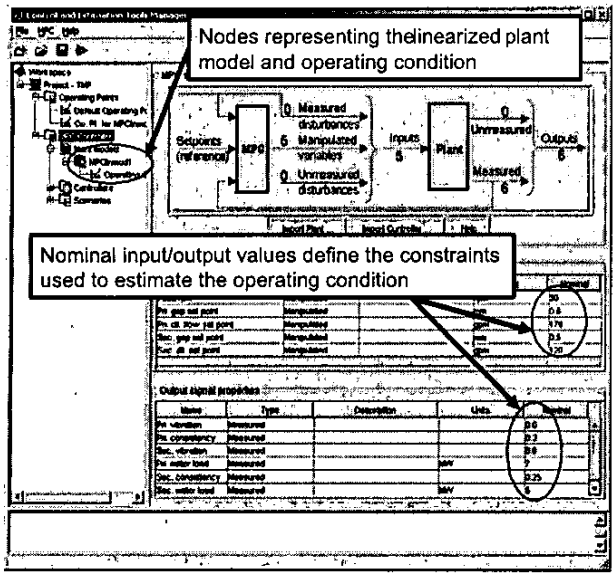

Fig. 4. Linearization of a Simulink plant model connected to the MPC block

tool assumes this correspondence. The user must manually identify d and v inputs and unmeasured plant outputs (if any) by tagging the signals on the block diagram.

The controller's presence in the Simulink diagram requires special handling during linearization. The MPC Toolbox uses a newly introduced feature of the Simulink Control Design product that automatically opens the feedback loop during linearization. If this were not done, the numerical perturbations introduced during linearization would determine the closed-loop system dynamics rather than the plant dynamics.

C. Controller Design

Once the internal plant model has been defined the remaining design decisions comprise:

i.) Characterization of each plant model input and output signal (e.g., manipulated variable, measured disturbance, etc.) and assignment of their nominal values. (See Fig. 4.)

ii.) Description of unmeasured and additive disturbance characteristics. (See section II.C) In the case study we initially accept the defaults of negligible measurement noise and additive integrated white noise on each measured output.

iii.) Assignment of horizons and weights. We assign the two vibration outputs zero weights since their only purpose is to represent plate clash constraints. Consequently, the application has 5 manipulated variables (MVs) and 4 outputs to be controlled at setpoints. The feeder rpm MV is assigned a setpoint, representing the plant production rate. The other MV weights are left at their defaults, i.e., a small penalty on rate-of-change. The prediction horizon is set to 20 samples and the control horizon to 5 samples.

iv.) Definition of input and output constraints and assignment of their ECR vectors. The vibration and motor load output have upper bounds. All other outputs are unconstrained. The MVs have hard upper and lower bounds (ECR values are zero).

The model and detailed design specifications are available from the authors.

D. Simulation

Simulation can be performed either in the MPC Toolbox GUI or in Simulink using the MPC block. GUI simulation enables rapid visualization of the effect of changing design parameters, but is limited to linear plant models. (Mismatch between the controller model and the plant is an option.) Simulink enables simulation to be performed on nonlinear plant models. When a nonlinear plant model is available, as it is in this case study, it can be used to assess the controller behavior under more realistic conditions. For example, consider the effect of a change in primary motor load from 7 to 8.5 MW, which is large enough to make the first output constrain active and cause significant nonlinear effects. Fig. 5 compares the results of a linear simulation in the GUI with no plant/model mismatch to a nonlinear Simulink simulation. Both predict that the controller's attempt to reach the higher motor load target (by closing the primary plates) will elevate the primary vibration signal to its upper constraint at 0.1. The oscillations (and constraint violations) in the Simulink case are a result of prediction errors in the controller's linear model. Also, in the linear simulation the primary motor load attains its new target. The nonlinear simulation predicts that the primary load target must be sacrificed in order to satisfy the vibration constraint. If these results were unacceptable, the designer could easily modify the controller specifications in the GUI and rerun the Simulink simulation. The model obtained by the default linearization procedure does not account for the effect of unmeasured chip packing density disturbances, which are the last two inputs in Fig. 1. They can be explicitly modeled by manually assigning linearization 'input points' to these inputs, and then re-running the linearization step to obtain an expanded plant model. This additional step may be warranted if packing density changes are a major disturbance source and specific information is available about the power spectrum of such disturbances. Consider the case where the screw feeder is fed by a rotary valve, which often introduces random 'periodic' packing density variation. In these cases the power spectrum of the packing density disturbances will concentrate in a narrow frequency range around the rotational frequency of the rotary valve. Consequently, an LTI model of such disturbances must include one or more under-damped poles. If these resonant disturbance characteristics can be accurately modeled within the MPC controller, its state estimates will be improved, resulting in improved disturbance rejection.

Fig. 6 shows a comparison of the closed loop control of primary consistency with the default disturbance model

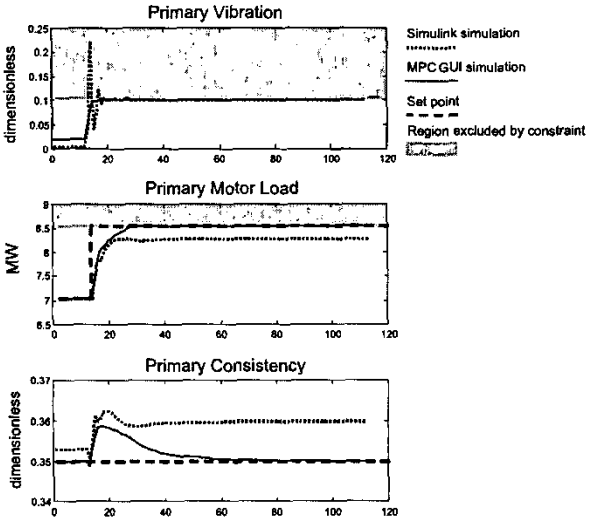

Fig. 5. Simulated reaction of primary vibration signals, load and consistency to a primary motor load set point change

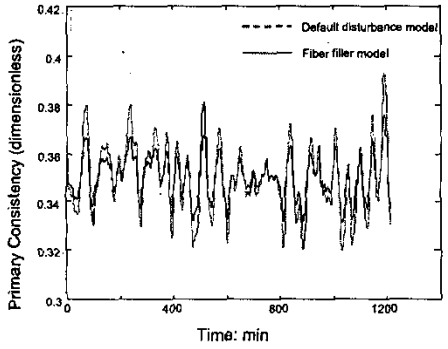

Fig. 6. Comparison of controller performance using a custom LTI input disturbance model versus using the default disturbance model

versus an accurate under-damped LTI model describing the density disturbance random process. For clarity all disturbances were assumed to be introduced by these density variations.

E. Implementation

Once designed, the controller can be implemented in a real-time application in several ways. An approach suitable for an educational environment is described in [5]. It uses Simulink and the Data Acquisition Toolbox. Implementation as C code using MathWorks Real Time Workshop is another possibility. Yet another is the combination of MATLAB and the OLE for Process Control (OPC) Toolbox. The following steps set up the connection to an OPC server that allow the deplotment of the application from MATLAB.

1) Extract the MPC design by exporting the controller from the MPC GUI as a MATLAB object
2) Connect to the OPC server

```
h = opcda(hostipaddress,ProgID);
connect(h);
```

3) Build one OPC group for the measured plant outputs and another for the manipulated variables

```
g_measured_op = ...
   addgroup(h,'Measured vars');
additem(g_measured_op,'feedrate_rpm');
additem(g_measured_op,'primarygap');
... (remaining measured variables)
g_manipulated_vars = ...
   addgroup(h,'Manipulated vars');
additem(g_manipulated_vars,'primary_vib');
additem(g_manipulated_vars,'primary_cons');
... (remaining measured disturbances)
```

4) Trigger MPC execution from a clock tag on the OPC server whose value changes each time a control interval has elapsed.

```
g_timer = addgroup(h,'timeer');
time_tag = additem(g_timer,'Time_tag');
```

5) Set the "DataChangeFcn" callback which will execute each time the clock tag changes. The callback function "localMPCStep" has arguments defined by the OPC measured variable & manipulated variable OPC groups and the MPC object representing the controller design.

```
gtime.DataChangeFcn = ...
   {@localMPCStep g_measured_op ...
   g_manipulated_vars mpcobj};
```

The callback calls the MPCMOVE method of the MPC object to re-compute target values for the manipulated variables each step.

```
[manipulated_variables,thismpcstate] = ...
   mpcmove(mpcobject, measured_outputs, ...
   set_points,[],thismpcstate);
```

Measured output data is accessed in the callback by calling:

```
measured_outputs = g_measured_op.Item.Value;
```

The new manipulated variable targets are written back to the OPC server using:

```
writeasync(g_manipulated_vars, ...
    num2cell(manipulated_variables));
```

IV. Conclusions

The TMP case study illustrates the importance of testing an MPC design in realistic nonlinear simulations. To facilitate such testing, one needs automated linearization tools and a convenient way to implement the controller in a nonlinear system. We have also outlined one method for implementing the final design in a real-time application.

References

[1] S. Qin and T. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, pp. 733–764, 2003.

[2] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[3] A. Bemporad, N. Ricker, and M. Morari, *Model Predictive Control Toolbox for Matlab*. The Mathworks, Inc., 2004.

[4] R. S. Roger Evans and K. Saarinen, "Refiner control effectively accomplished through adaptive control," *Control Systems '92 (Published by IFAC and the Canadian Pulp and Paper Association)*, 1992.