DATA-DRIVEN MODELING OF BATCH PROCESSES

Dennis Bonné * **Sten Bay Jørgensen** *

* *Department of Chemical Engineering, Technical University of Denmark, DK-2800 Lyngby, Denmark, Email: (db or sbj)@kt.dtu.dk, Fax: +45 4593 2906*

Abstract: A one dimensional grid of interdependent linear models obtained from operation data is proposed for modeling repeated finite horizon, nonlinear and non-stationary process operations. Such finite horizon process operations include start-ups, grade transitions, shut-downs, and of course batch, semi-batch and periodic processes. The model grid is identified from data using a novel interpretation of generalized ridge regression that penalizes weighted discrepancies between one linear model and the models in its neighborhood. It is furthermore outlined how different representations of such a model grid may be used off-line as well as on-line, for prediction, monitoring, control, and optimization. Among these representations is a linear time-varying state space model which may be used for design in established linear monitoring and control methodologies.

Keywords: Batch, Non-linear systems, Time-varying systems, Parameter estimation

1. INTRODUCTION

Batch processes are experiencing a renaissance as products-on-demand and first-to-marked strategies impel the need for flexible and specialized production methods. Furthermore, industries such as food, biochemical, and pharmaceutical depend on the confinement of faults and contaminations to single batches. This renaissance propels the need for modeling and control tools, which can facilitate optimal and reliable operation of batch processes. However, the traditional linear modeling and control tools are inadequate, when applied to the often highly nonlinear and time-varying behavior of batch processes.

In section 2 it is thus proposed to model batch processes with a time-varying grid of linear models and it is demonstrated how such model grids may be applied to both off-line and on-line monitoring, prediction, control, and optimization. Identification of these model grids is addressed in section 3 and it is proposed to use model property based regularization to overcome excessive variance. The methods proposed in sections 2 and 3 are applied to an industrial case study in section 4 and finally conclusions are given.

2. TIME-VARYING MODELS

Most often the complex and nonlinear dynamics of continuously operated processes can be approximated with a moderate set of local Linear Time-Invariant (LTI) models, each of which describes a characteristic region in the operation window. These regions described by local models will often be characterized by a set of active constraints. For batch and semi-batch processes (from here on, batch will cover both batch and semi-batch processes) however, the set of active constraints will change as the batch progresses. In fact, to

operate a batch process in an optimal fashion, a specific sequence of constraints is tracked during operation. This means that local approximations of characteristic regions are not sufficient to describe batch operation. The transitions between these locally approximated characteristic regions are also needed to provide a complete description of batch operation. Furthermore, even if specific sets of constraints were active for longer periods; local LTI models can not be expected to describe the time variation due to changing holdups and/or compositions.

The periodic nature and the finite horizon of batch processes however, make it possible to model the evolution from each sample point to the next in a batch with one *grid-point* LTI model. In this fashion, both the time variation within the characteristic regions and the transitions between these may be approximated with a grid of grid-point models. Thus, such a model set gives a complete description of a batch. The finite horizon of batch processes means that the model set will be finite. The periodic way in which the same recipe is repeated batch after batch means that several measurements from the individual sample points are available for identification. That is, the time evolution of a process variable is measured/sampled at specific sample points during the batch operation and as the batch operation is repeated, several measurements are collected from every sample point. With multiple data points/measurements from one specific sample point a grid-point model can be identified for this sample point. Explicitly, in addition to the time dimension, data from batch processes also evolve in a batch index dimension.

2.1 Model Parameterization

Given the discussion above, batch processes are modeled with sets of dynamic grid-point LTI models. Such a set of grid-point LTI models could also be referred to as a Linear Time Varying (LTV) batch model. These grid-point LTI models can be parameterized in a number of ways – e.g. as Output Error (OE) models, AutoRegressive models with eXogenous inputs (ARX), State Space (SS) models, etc. In the present contribution the ARX model parameterization was chosen. This choice of parameterization offers a relatively good multivariable system description with a moderate number of model parameters.

As operation of a batch progresses, different inputs and outputs may be used depending on the current phase of the batch and hence in order to model batch operation it is convenient to define the following variables and references for each time step t: Input variable $u_t \in \mathbb{R}^{n_u(t)}$ with reference $\bar{u}_t \in \mathbb{R}^{n_u(t)}$, output variable $y_t \in \mathbb{R}^{n_y(t)}$ with reference $\bar{y}_t \in \mathbb{R}^{n_y(t)}$, and disturbance variable $w_t \in \mathbb{R}^{n_y(t)}$. Using an ARX model parameterization, the output deviation $\bar{y}_t - y_t$ at time t may be given as a weighted sum of $n_A(t)$ past output deviations and $n_B(t)$ past input deviations

$$\begin{aligned}\bar{y}_t - y_t = &- a_{t,t-1}\Big(\bar{y}_{t-1} - y_{t-1}\Big) - \dots \\ &- a_{t,t-n_A(t)}\Big(\bar{y}_{t-n_A(t)} - y_{t-n_A(t)}\Big) \\ &+ b_{t,t-1}\Big(\bar{u}_{t-1} - u_{t-1}\Big) + \dots \\ &+ b_{t,t-n_B(t)}\Big(\bar{u}_{t-n_B(t)} - u_{t-n_B(t)}\Big) \\ &+ w_t\end{aligned} \tag{1}$$

where $n_A(t), n_B(t) \in [1, \dots, t]$ are the grid-point ARX model orders and $a_{i,j} \in \mathbb{R}^{n_y(i),n_y(j)}$ and $b_{i,j} \in \mathbb{R}^{n_y(i),n_u(j)}$ are the grid-point ARX model parameter matrices. Note, as the grid points are modeled with individual grid-point models, the sample points t do not have to be equidistantly spaced in time. Let N be the batch length(/number of samples) and define the input $\boldsymbol{u}$, output $\boldsymbol{y}$, shifted output $\boldsymbol{y}^0$, and disturbance $\boldsymbol{w}$ profiles as

$$\begin{aligned}\boldsymbol{u} &= \begin{bmatrix} u_0' & u_1' & \dots & u_{N-1}' \end{bmatrix}' \\ \boldsymbol{y} &= \begin{bmatrix} y_1' & y_2' & \dots & y_N' \end{bmatrix}' \\ \boldsymbol{y}^0 &= \begin{bmatrix} y_0' & y_1' & \dots & y_{N-1}' \end{bmatrix}' \\ \boldsymbol{w} &= \begin{bmatrix} w_1' & w_2' & \dots & w_N' \end{bmatrix}'\end{aligned} \tag{2}$$

Note, not all initial conditions y_0 are measurable and/or physically meaningful — e.g. off-gas measurements. Thus the ARX model set may be expressed in matrix form

$$\bar{\boldsymbol{y}} - \boldsymbol{y} = -\boldsymbol{A}(\bar{\boldsymbol{y}}^0 - \boldsymbol{y}^0) + \boldsymbol{B}(\bar{\boldsymbol{u}} - \boldsymbol{u}) + \boldsymbol{w} \tag{3}$$

where $\boldsymbol{A}, \boldsymbol{B}$ are structured lower block triangular matrices. The profile $\boldsymbol{w}$ is a sequence of disturbance terms caused by bias in the reference input profile $\bar{\boldsymbol{u}}$, the effect of process upsets, and the modeling errors from linear approximations. This means that the disturbance $\boldsymbol{w}$ contains contributions from both batch wise persistent disturbances, such as recipe/input bias, model bias, and erroneous sensor readings, as well as from random disturbances, which occur with no batch wise correlation. It thus seems resoanble to model the disturbance profile $\boldsymbol{w}$ with a random walk model with respect to the batch index k

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \boldsymbol{v}_k \tag{4}$$

where $\boldsymbol{v}_k$ represents a sequence of batch wise non-persistent disturbances that are assumed to be zero-mean, independent and identically distributed. The assumption of $\boldsymbol{v}_k$ being white is rather crude, but necessary if one whishes to keep the parameter estimation problem linear. Considering the difference between two successive batches

$$\Delta \boldsymbol{y}_k = \boldsymbol{y}_k - \boldsymbol{y}_{k-1} = \boldsymbol{A}\Delta \boldsymbol{y}_k^0 - \boldsymbol{B}\Delta \boldsymbol{u}_k + \boldsymbol{v}_k \quad (5)$$

A batch ARX model (5) that is independent of the reference profiles $(\bar{\boldsymbol{y}}, \bar{\boldsymbol{u}})$ and batch wise persistent disturbances has been obtained. With such a batch ARX model the path is prepared for multivariable, model-based monitoring, control, optimization, and of course simulation.

During the model derivation above it was assumed that the outputs are known. This is however not the case in practice, where only a sequence $\boldsymbol{z}_k$ of noisy observations of the outputs is available

$$\boldsymbol{z}_k = \left[\, y'_{k,0} \;\; \boldsymbol{y}'_k \, \right]' + \boldsymbol{\epsilon}_k \quad (6)$$

where $\boldsymbol{\epsilon}_k$ is a sequence of measurement noise terms that are assumed to be zero-mean, independent and identically distributed.

2.2 Application Specific Models

Depending on the task the batch ARX model (5) is to be applied to, it is convenient to convert the batch ARX model into different representations. If the task at hand is to predict (or simulate) the behavior of a batch before it is started the following form is convenient

$$\Delta \boldsymbol{y}_k = \boldsymbol{H}\Delta y_{k,0} - \boldsymbol{G}\Delta \boldsymbol{u}_k + \boldsymbol{E}\boldsymbol{v}_k \quad (7)$$

Note that the disturbance matrix $\boldsymbol{E}$ models the propagation of batch wise non-persistent disturbances — including batch wise non-persistent model-plant mismatch.

The form (7) above is also convenient for the task of classification/monitoring (e.g. normal or not) of a batch after it has been completed. Furthermore, the form (7) can be used to determine open-loop optimal recipes in the sense of optimizing an objective for the batch. If such an objective is to minimize the deviations $\boldsymbol{e}$ from a desired trajectory $\bar{\boldsymbol{y}}$, then (7) can be modified into

$$\boldsymbol{e}_k = \bar{\boldsymbol{y}} - \boldsymbol{y}_k = \boldsymbol{e}_{k-1} - \boldsymbol{H}\Delta y_{k,0} + \boldsymbol{G}\Delta \boldsymbol{u}_k - \boldsymbol{E}\boldsymbol{v}_k \quad (8)$$

There are two important points to be made about the trajectory tracking model form (8). First of all, as the error profile $\boldsymbol{e}_k$ in batch k depends on the error profile $\boldsymbol{e}_{k-1}$ from batch $k-1$, the effects of the batch wise persistent disturbances are integrated with respect to batch index. This means that a properly designed controller can reject the effects of the batch wise persistent disturbances asymptotically with respect to batch index — e.g. removing the effects of recipe and model bias. Secondly, given the above mentioned asymptotic behavior and as the control moves/actions generated by such a controller are deviations from the control/input profile realized in the previous batch, the control actions due to batch wise persistent disturbances will converge asymptotically to zero with respect to batch index. In literature it is said that the controller learns to reject the batch wise persistent disturbances — i.e. the resulting controller is an Iterative Learning Control (ILC) scheme. A more accurate formulation would be that both output and input errors are modeled using integrators with respect to batch index. The trajectory tracking model representation (8) is similar to that of Lee *et al.* (2000), but the representations differ significantly since (8) includes the effect of initial conditions ($\boldsymbol{H}\Delta y_{k,0}$) and disturbance propagation ($\boldsymbol{E}\boldsymbol{v}_k$). Another important difference is that (8) does not have double dependence on the batch wise persistent disturbances — i.e., the trajectory tracking model representation (8) only include the batch wise persistent disturbances as represented by $\boldsymbol{e}_{k-1}$ and not as both the part of $\boldsymbol{e}_{k-1}$ caused by the batch wise persistent disturbances and the batch wise persistent disturbances themselves.

The two forms (7) and (8) of the batch ARX model above are applicable to off-line or inter-batch type applications. For on-line estimation, monitoring, feedback control, and optimization however, it is convenient to use a state space realization of the batch ARX model. To achieve such a realization it is necessary to simplify the batch ARX model structure with the assumption that the number of outputs is constant $n_y(t) = n_y$ for $t = 1, \ldots, N$. In an observer canonical form the state space realization is given as

$$\begin{aligned} x_{k,t} &= \mathcal{A}_t x_{k,t-1} + \mathcal{B}_t \Delta u_{k,t-1} + \mathcal{E} v_{k,t} \\ \Delta y_{k,t} &= \mathcal{C} x_{k,t} \end{aligned} \quad (9)$$

with the initial condition $x_{k,0} = [\Delta y'_{k,0}, 0', \ldots, 0']'$. Just as (7), the SS model form (9) is convenient for prediction, monitoring, and optimization type applications, but also facilitates on-line implementations of these. Furthermore, the SS model form (9) is particularly well suited for closed-loop or feedback control applications. For tracking control applications the SS model form (9) can be modified into

$$\begin{aligned} x_{k,t} &= \mathcal{A}_t x_{k,t-1} + \mathcal{B}_t \Delta u_{k,t-1} + \mathcal{E} v_{k,t} \\ e_{k,t} &= e_{k-1,t} - \mathcal{C} x_{k,t} \end{aligned} \quad (10)$$

Following the discussion above, a multivariable feedback controller properly designed using the trajectory tracking SS model form (10), will reject the effects of the batch wise persistent disturbances asymptotically with respect to batch index. That is, due to the output and input error integration in the model framework, a controller designed to reject disturbances with respect to time in one batch at a time will also asymptotically reject the effects of batch wise persistent disturbances with respect to batch index.

3. MODEL IDENTIFICATION

With the batch ARX model (5) derived above, the parameterization of the batch model is in place, however the model orders and the model parameters still need to be determined from process data. One major drawback of the proposed parameterization is the immense dimensionality of the resulting set of models — in practice this immense dimensionality will render any standard Least Squares (LS) identification problem ill-conditioned. It turns out however, that as the grid-point models are progressively constrained by the smoothness of the model grid, the conditioning of the identification problem improves.

3.1 Data Pretreatment

In industry, the process variables $\tilde{z}_{k,\tilde{t}}(p) \in \mathbb{R}$ are most often logged individually at times $\tilde{T}(k,\tilde{t},p)$, giving $N_{\tilde{z}}(k,p)+1$ observations of variable p in batch k. What is needed however, is up to $N+1$ noise free observations of the variables at times $T(t)$ in the N_B batches available for identification. These noise free or expected observations can be estimated using local polynomial regression and If the profile of process variable p in batch k is defined as $\tilde{\boldsymbol{z}}_{k,p}$, then the estimation problem can be given explicitly (Hastie *et al.*, 2001) as

$$\hat{\tilde{z}}_{k,t}(p) = \boldsymbol{s}_{k,p,t}\tilde{\boldsymbol{z}}_{k,p} \tag{11}$$

where $\boldsymbol{s}_{k,p,t}$ is a smoothing vector. If it is further assumed that process variable p will be used throughout the batch, then the estimated profile of variable p in batch k is given as

$$\begin{aligned}\hat{\tilde{\boldsymbol{z}}}_k(p) &= \left[\begin{array}{cccc}\boldsymbol{s}'_{k,p,0} & \boldsymbol{s}'_{k,p,1} & \cdots & \boldsymbol{s}'_{k,p,N}\end{array}\right]' \tilde{\boldsymbol{z}}_k(p) \\ &= \boldsymbol{S}_{k,p}\tilde{\boldsymbol{z}}_k(p)\end{aligned} \tag{12}$$

Let the true observation $\bar{\boldsymbol{z}}_k \in \mathbb{R}^{(n_y(t)+n_u(t))}$ be given as

$$\bar{\boldsymbol{z}}_k = \hat{\bar{\boldsymbol{z}}}_k + \boldsymbol{\omega}_k \tag{13}$$

where $\hat{\bar{\boldsymbol{z}}}_k$ is the estimated observation and $\boldsymbol{\omega}_k$ is a sequence of estimation errors. The estimation error $\boldsymbol{\omega}_k$ will consist of both systematic errors such as the height of a characteristic peek being underestimated due to excessive smoothing and/or *trimming the hills and filling the valleys* due to too low local regression order, and random errors. Thus the estimation error $\boldsymbol{\omega}_k$ is modeled with a random walk with respect to the batch index k

$$\boldsymbol{\omega}_k = \boldsymbol{\omega}_{k-1} + \boldsymbol{\nu}_k \tag{14}$$

where $\boldsymbol{\nu}_k$ represents a sequence of batch wise non-persistent estimation errors that are assumed to be zero-mean. Consider the expected difference between two successive batches, then

$$E\{\Delta\bar{\boldsymbol{z}}_k\} = \hat{\bar{\boldsymbol{z}}}_k - \hat{\bar{\boldsymbol{z}}}_{k-1} + E\{\boldsymbol{\nu}_k\} = \Delta\hat{\bar{\boldsymbol{z}}}_k \tag{15}$$

is given as the difference between their respective estimates. The expected output and input difference profiles which are all contained in $\Delta\bar{\boldsymbol{z}}_k$, are thus given as

$$\begin{aligned}&E\{\Delta\boldsymbol{y}_k\} = \Delta\hat{\boldsymbol{y}}_k, \quad E\{\Delta\boldsymbol{y}^0_k\} = \Delta\hat{\boldsymbol{y}}^0_k \\ &E\{\Delta\boldsymbol{u}_k\} = \Delta\hat{\boldsymbol{u}}_k\end{aligned} \tag{16}$$

3.2 Parameter Estimation

Several suggestions to how (sets of) LTI or (periodic) LTV models should be identified from data can be found in literature. All these authors employ some or other coefficient shrinkage or subspace method to improve the conditioning of the identification problem and hence lower the variance of the model parameter estimates. Simoglou *et al.* (2002) suggested estimating a set of independent, overlapping local LTI SS models using Canonical Variant Analysis (CVA). Instead the present contribution proposes estimating a grid/set of interdependent grid-point LTI ARX models using a novel interpretation of generalized ridge regression.

The batch ARX model (5) can be formulated as linear regression

$$\Delta\boldsymbol{y}_k = \Delta\boldsymbol{x}_k\boldsymbol{\theta} + \boldsymbol{v}_k \tag{17}$$

where $\Delta\boldsymbol{x}_k = \Delta\boldsymbol{x}_k(\Delta\boldsymbol{y}^0_k, \Delta\boldsymbol{u}_k)$ is a structured regressor matrix with past outputs and inputs and $\boldsymbol{\theta} = \boldsymbol{\theta}(\boldsymbol{A},\boldsymbol{B})$ is a column parameter vector with the model parameters from the batch ARX model. Taking the expectation of the linear regression (17) and recalling (16) we find that

$$\Delta\hat{\boldsymbol{y}}_k = E\{\Delta\boldsymbol{x}_k\}\boldsymbol{\theta} + E\{\boldsymbol{v}_k\} = \Delta\hat{\boldsymbol{x}}_k\boldsymbol{\theta} \tag{18}$$

with $\Delta\hat{\boldsymbol{x}}_k = \Delta\boldsymbol{x}_k(\Delta\hat{\boldsymbol{y}}^0_k, \Delta\hat{\boldsymbol{u}}_k)$. This means that, if the process variable estimation error model (14) is a valid approximation, then estimation of model parameters from the pretreated data will give unbiased model parameter estimates. Although unbiased, model parameter estimates based on data from a single batch would have excessive variance. Thus to lower the variance of model parameter estimates, all available data should be used for the model parameter estimation

$$\begin{aligned}\boldsymbol{Y} &= \left[\begin{array}{cccc}\Delta\hat{\boldsymbol{y}}'_1 & \Delta\hat{\boldsymbol{y}}'_2 & \cdots & \Delta\hat{\boldsymbol{y}}'_{N_B}\end{array}\right]' \\ &= \left[\begin{array}{cccc}\Delta\hat{\boldsymbol{x}}'_1 & \Delta\hat{\boldsymbol{x}}'_2 & \cdots & \Delta\hat{\boldsymbol{x}}'_{N_B}\end{array}\right]'\boldsymbol{\theta} = \boldsymbol{X}\boldsymbol{\theta}\end{aligned} \tag{19}$$

The linear system (19) would however, most likely still be rank-deficient and solving it in a Least Squares (LS) sense would still produce estimates with low bias, but excessive variance. Such excessive model parameter variance would despite the low bias, yield poor model predictions (Larimore, 1996). Hence, to improve the predictive capabilities of an estimated model the variance of the estimated model parameters must be further reduced.

A possible approach to reducing the variance of model parameter estimates is to enforce that the estimated model possesses some desired model properties. One such model property could be that neighboring grid-point models are analogous in the sense that they exhibit similar behavior. In fact, without this property, the model would be a *set* of independent models and not a *grid* of interdependent models. Enforcing model properties however, inevitably introduce bias into the model parameter estimates. There will thus be a trade-off between the bias and variance of the model parameter estimates and this trade-off will determine the predictive capabilities of estimated models. A parameter estimation method that could incorporate model properties into LS estimates is generalized ridge regression, which also is referred to as Tikhonov regularization. We thus propose to estimate the model parameters by solving the extended LS problem

$$\begin{aligned}\hat{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) &= \arg\min_{\boldsymbol{\theta}}[(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\theta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\theta}) \\ &\qquad + (\boldsymbol{\Lambda}\boldsymbol{L}\boldsymbol{\theta})'(\boldsymbol{\Lambda}\boldsymbol{L}\boldsymbol{\theta})] \\ &= \left(\boldsymbol{X}'\boldsymbol{X}+\boldsymbol{L}'\boldsymbol{\Lambda}^2\boldsymbol{L}\right)^{-1}\boldsymbol{X}'\boldsymbol{Y}\end{aligned} \tag{20}$$

where the penalty $\boldsymbol{\Lambda}\boldsymbol{L}\boldsymbol{\theta}$ is a column vector of weighted differences between parameters in neighboring grid-point models. In this fashion, the structured penalty matrix $\boldsymbol{L}$ maps the parameter vector $\boldsymbol{\theta}$ into the desired parameter differences and the diagonal regularization matrix $\boldsymbol{\Lambda}$ weights the parameter differences. The estimated parameter vector $\hat{\boldsymbol{\theta}}(\boldsymbol{\Lambda})$ is a function of the regularization matrix $\boldsymbol{\Lambda}$, which determines the shrinkage and hence the trade-off between bias and variance. This means that the regularization matrix $\boldsymbol{\Lambda}$ can be used to tune the predictive capabilities of the model estimate. Through the particular choice of penalty matrix $\boldsymbol{L}$, the regularization matrix $\boldsymbol{\Lambda}$ also determines the interdependency between the grid-point models in the model grid.

3.3 Model Orders and Regularization Weights

Several methods for choosing (optimal) regularization weights can be found in literature (Hansen, 1996), but all of these consider either scalar regularization weights or diagonal penalties. In the present work it is proposed to simply select a regularization matrix from a finite set $\boldsymbol{\Lambda} \in \boldsymbol{\Omega}_{\boldsymbol{\Lambda}}$, that yield near minimum mean squared prediction error, when the estimated model is cross-validated through "pure-simulation". That is, given the "pure-simulation" prediction error profile $\boldsymbol{\zeta}_k(\boldsymbol{\Lambda})$ from cross-validation batch k

$$\boldsymbol{\zeta}_k(\boldsymbol{\Lambda}) = \Delta\hat{\boldsymbol{y}}_k^{val} - \hat{\boldsymbol{H}}(\boldsymbol{\Lambda})\Delta\hat{y}_{k,0}^{val} + \hat{\boldsymbol{G}}(\boldsymbol{\Lambda})\Delta\hat{\boldsymbol{u}}_k^{val} \tag{21}$$

the regularization matrix $\boldsymbol{\Lambda}$ is the solution to the discrete optimization problem

$$\begin{aligned}\boldsymbol{\Lambda} = \arg\min_{\boldsymbol{\Lambda}} &\left[\gamma(\boldsymbol{\Lambda}) = \sum_{k=1}^{N_B^{val}} \boldsymbol{\zeta}_k(\boldsymbol{\Lambda})'\boldsymbol{\zeta}_k(\boldsymbol{\Lambda})\right] \\ s.t.\quad &\boldsymbol{\Lambda} \in \boldsymbol{\Omega}_{\boldsymbol{\Lambda}} \\ &(\boldsymbol{X}'\boldsymbol{X}+\boldsymbol{L}'\boldsymbol{\Lambda}^2\boldsymbol{L}) \text{ nonsingular}\end{aligned} \tag{22}$$

where N_B^{val} is the number of batch difference profiles available for cross-validation. In this fashion, the computational burden of solving (22) is determined by the number of elements in the finite set $\boldsymbol{\Omega}_{\boldsymbol{\Lambda}}$.

Thus far only estimation of a specific batch ARX parameterization, i.e., a batch ARX model with model orders $n_A(t)$ and $n_B(t)$ for $t = 1, \ldots, N$ has been considered. These model orders are however unknown and will also have to be identified from data. This means that in addition to the regularization weighting matrix $\boldsymbol{\Lambda}$ also the model orders can be used to tune the predictive capabilities of the model estimate. Traditionally, ARX model orders are selected based on minimization of measures such as Final Prediction Error (FPE) or Akaike's Information Criterion (AIC) both of which are proportional to the optimal value of the LS objective being minimized as part of the identification, to prevent modeling noise/disturbance characteristics, i.e., overfit. Overfit is however also prevented if the ARX model orders are selected based on minimization of the mean squared prediction errors from cross-validation of the estimated models. This means that the ARX model orders can be selected based on minimization of

$$\begin{aligned}\begin{bmatrix}\{n_A(t)\}_{t=1}^N \\ \{n_B(t)\}_{t=1}^N\end{bmatrix} = \min_{\substack{\{n_A(t)\}_{t=1}^N \\ \{n_B(t)\}_{t=1}^N}} &[\gamma(\boldsymbol{\Lambda})] \\ s.t.\quad &\boldsymbol{\Lambda} \text{ given by (22)}\end{aligned} \tag{23}$$

If the ARX model orders are assumed constant throughout the batch $n_A = n_A(t)$ and $n_B = n_B(t)$ for $t = 1, \ldots, N$, then (23) simplifies to

$$\begin{aligned}(n_A, n_B) = \min_{n_A, n_B} &[\gamma(\boldsymbol{\Lambda})] \\ s.t.\quad &\boldsymbol{\Lambda} \text{ given by (22)}\end{aligned} \tag{24}$$

4. APPLICATION

To demonstrate the capability of the proposed data-driven models, an industrial, production scale *Bacillus protease* fermentation has been modeled from historical data (Novozymes A/S). The modeling objective was prediction of the on-line measured variables used to supervise the fermentation as well as the product activity which is measured sparsely off-line. That is, the objective was to obtain a model that can predict the course and outcome of a batch given the batch recipe and its initial conditions. For this *Bacillus protease* fermentation the batch recipe consists (essentially) of reference profiles for two

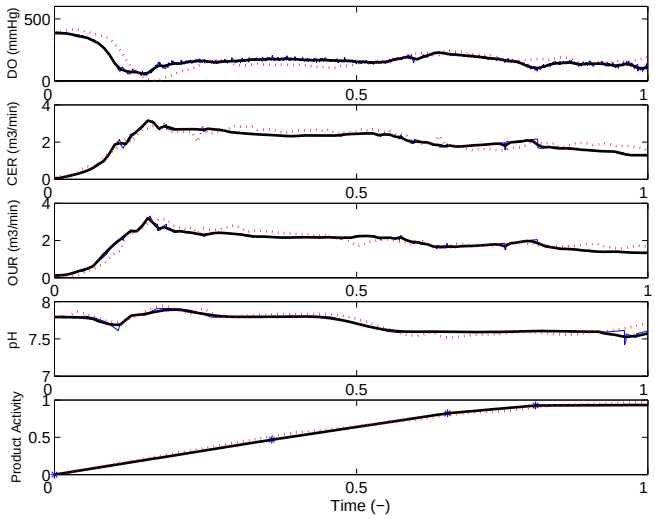

Fig. 1. Example of cross-validation of an industrial *Bacillus protease* fermentation model. The five outputs, Dissolved Oxygen (DO), Carbon dioxide Evolution Rate (CER), Oxygen Uptake Rate (OUR), pH, and product activity, are predicted given information about their initial conditions and the batch recipe (assuming perfect control) — i.e., *pure-simulation* prediction. The thin solid lines (or '$*$' for the product activity) are the historical measurements as logged, the bold solid lines are the pretreated data (as the data from which the model was identified, but not used in the identification), and the dotted lines are the model predictions.

substrate feeds, an alkaline feed, pressure, and temperature — i.e., these reference profiles are the inputs of the model. For most of these inputs the current practice is however, that the reference profile is either not logged or not manipulated from batch to batch. As a temporary workaround, perfect control was assumed and the reference profiles were replaced by the realized profiles. As is common practice in supervision of fermenters, Dissolved Oxygen (DO), Carbon dioxide Evolution Rate (CER), Oxygen Uptake Rate (OUR), and pH were chosen as process indicators. Along with the product activity these process indicators makeup the outputs of the model.

The historical data was smoothened and re-sampled to 30 minutes intervals. The product activity was re-sampled using linear interpolation, while the remaining inputs and outputs were smoothened using local constant regression and bandwidths ranging from 6 to 73 nearest neighbors. Before identification the batch difference profiles were normalized. The model was identified using data from 29 batches and cross-validated using data from 9 batches, one of which is shown in figure 1. The identified model orders ranged from 0 to 20 and the total number of model parameters estimated was 12,135. The mean cross-validation prediction error was 0.13.

5. CONCLUSION

In the present paper it is proposed to model finite horizon, time-varying and nonlinear process operations with 1-dimensional grids of interdependent ARX models. Such model grids can be used for both off- and on-line monitoring, prediction, control, and optimization applications. It is further proposed that these model grids are identified from historical process data using ridge regression. By identifying all the ARX models in a model grid simultaneously, the interdependency of the ARX models can be used to reduce the variance of their estimates and thereby improve the predictive capabilities of the estimated model grid. The proposed data-driven modeling scheme has been demonstrated through modeling of an industrial fermentation process.

REFERENCES

Hansen, Per Christian (1996). *Rank-Deficient and Discrete Ill-Posed Problems*. Polyteknisk Forlag. Lyngby.

Hastie, Travor, Robert Tibshirani and Jerome Friedman (2001). *The Elements of Statistical Learning*. Springer. New York.

Larimore, Wallace E. (1996). Statistical optimality and canonical variate analysis system identification. *Signal Processing* **52**(2), 131–144.

Lee, Jay H., Kwang S. Lee and Won C. Kim (2000). Model-based iterative learning control with a quadratic criterion for time-varying linear systems. *Automatica* **36**(5), 641–657.

Simoglou, A., E. B. Martin and A. J. Morris (2002). Statistical performance monitoring of dynamic multivariate processes using state space modelling. *Computers and Chemical Engineering* **26**(6), 909 – 920.