Lesson 9

Application to Dynamic Models

All the above discussions assumed static dependency between the input and the output, so that the value of y should be directly determined by the value x. In systems engineering, the models are usually dynamic — there is some "inertia" in the system, or "memory" between inputs and responses. Past affects the future, and successive measurements become interlinked. In principle, this makes the models considerably more complicated. However, the derived models can readily be extended to dynamic domains.

9.1 Representing dynamics

It turns out that basically static multivariate methods can be extended also to the determination of dynamic model parameters. The procedure is as follows: First, determine the state sequence, then compress the state space (using the familiar methods), and, finally, solve for the system matrices. To accomplish this, some basic theory is needed.

9.1.1 Capturing history

In the chapter 3, it was explained how structural complexity can be changed into dimensional complexity. A specially interesting form of structural complexity is caused by dynamic nature in the system being studied. How to determine the features to represent the dynamic phenomena as data?

In dynamic systems, there is *inertia* — it is not only the input but also the history that affects the current and future behaviors of the system. This infinite history can be captured in a finite set of time-series samples: System theory says that behaviors of a d'th order discrete-time dynamic system can be represented in terms of d past samples:

$$y(\kappa) = f\left(y(\kappa-1), \ldots, y(\kappa-d_{\max}), u(\kappa), \ldots, u(\kappa-d_{\max})\right). \tag{9.1}$$

This means that the dynamic features in the data vector should be more or less delayed measurements of the signals. Assuming that the system input u goes through the system causing the output y, one can express the information in the input sample vectors and in the output sample vectors collectively as the *state vector*

$$x(\kappa) = \begin{pmatrix} u(\kappa) \\ \hline \vdots \\ \hline u(\kappa - d_{\max}) \\ \hline y(\kappa - 1) \\ \hline \vdots \\ \hline y(\kappa - d_{\max}) \end{pmatrix}. \tag{9.2}$$

That is, there are overlapping windows over the past time-series data. If the dynamics is assumed linear, there holds

$$y(\kappa) = F^T x(\kappa) \tag{9.3}$$

for some parameter matrix F. Here, it is assumed that the system dimension is not known beforehand, it is only assumed that dimension cannot exceed $d_{\max}$. Note that high dimensionality and excessive redundant variables is not assumed to be a problem — now it can be assumed that (preliminary) state x contains all information back to the maximum length of system memory.

Indeed, high dimensionality is not a problem if appropriate latent variable methods are applied when the regression model between x and y is constructed. Again, assume that the latent variable, or the reduced *minimum state,* is denoted $z(\kappa)$. The preliminary state $x(\kappa)$ is first projected onto this minimum state $z(\kappa)$ and from there to output $y(\kappa)$.

So, nothing new here. In principle, the above scheme is already a working solution for implementing determination of dynamic models. The problem here is that $x(\kappa)$ needs to be reconstructed at each time point κ separately — a more streamlined approach would be beneficial. What is more, more tailored formulations make it possible to employ the very powerful theory of linear dynamic systems. The rest of this chapter devoted to the challenges of writing the above model in the standard dynamic model form.

9.1.2 State-space models

There are various ways to represent dynamic systems in a mathematically compact way. In this context we only study *state-space models.* The basic idea here is that the history of the system is coded in the state vector z, as explained above, and the time-domain behavior of this state is coded in the model. Together with the future inputs the state determines the future outputs. The linear discrete-time state-space model can be written in the following general form:

$$\begin{cases} z(\kappa + 1) = Az(\kappa) + Bu(\kappa) + \epsilon(\kappa) \\ y(\kappa) = Cz(\kappa) + Du(\kappa) + e(\kappa). \end{cases} \tag{9.4}$$

Vector $u(\kappa)$ is the system input (dimension $n_u \times 1$), $z(\kappa)$ is the state vector (dimension $N \times 1$), and $y(\kappa)$ is the output (dimension $m \times 1$); matrices A, B, C, and D have compatible dimensions. Even if the model is written here using the minimum-dimensional state vector, the state sequence is by no means unique, and the state model representation is by no means optimal in terms of available parameters. Vectors $e(\kappa)$ and $\epsilon(\kappa)$ represent white noise sequences so that

$$\mathrm{E}\{\epsilon(\kappa_1)\epsilon^T(\kappa_2)\} = \begin{cases} R_{zz}, & \text{if } \kappa_1 = \kappa_2 \\ 0, & \text{otherwise,} \end{cases} \tag{9.5}$$

and

$$\mathrm{E}\{e(\kappa_1)e^T(\kappa_2)\} = \begin{cases} R_{yy}, & \text{if } \kappa_1 = \kappa_2 \\ 0, & \text{otherwise,} \end{cases} \tag{9.6}$$

are the *state noise* and the *measurement noise* covariances matrices, respectively. Additionally, assume that $\epsilon(\kappa)$ and $e(\kappa)$ are mutually correlated:

$$\mathrm{E}\{\epsilon(\kappa_1)e^T(\kappa_2)\} = \begin{cases} R_{zy}, & \text{if } \kappa_1 = \kappa_2 \\ 0, & \text{otherwise.} \end{cases} \tag{9.7}$$

Note that this system model is a generalization of what has been studied this far: If matrices A, B, and C are ignored, matrix D directly corresponds to the static regression model F^T. From this it is easy to see that there are much more degrees of freedom in the dynamic model as compared to the static one.

The above state-space system structure is used, for example, by the Kalman filter, and a wide variety of analysis and control design methods are readily available for models that are presented in this form. If one were able to find all the system matrices directly from data, such method would be very useful — and finding such a technique is our objective now.

9.2 Subspace identification

The solution to the above problem is given by *subspace identification* (complete coverage can be found in [**?**]). This branch of research is rather new, developed mainly in the 1990's (yet having its roots in *realization theory* dating back to 1960's), and the texts discussing this material are typically difficult to follow for non-experts. However, the ideas are again very simple. It needs to be kept in mind that this presentation only explains the bare bones, and various enhancements could be (and have been) implemented.

9.2.1 Stochastic models

There exist various modifications of the basic model (9.4) that can be useful in different applications. For example, discarding the input u, assuming that the process is run exclusively by the noise, one faces the so called *stochastic realization problem:* What is the underlying succession of unmeasurable states z, together with the model structure, that best can explain the observed outputs y? This problem setting is characteristic to filtering problems, where measurements are corrupted by noise and the original variables should be recovered.

State sequence

The simplified system model that is exclusively driven by the noise processes becomes

$$\begin{cases} z(\kappa+1) = Az(\kappa) + \epsilon(\kappa) \\ y(\kappa) = Cz(\kappa) + e(\kappa). \end{cases} \tag{9.8}$$

Now the available data only consists of output time series samples. Assume that the block of k successive data samples extends from time point $k_0 = 1$ to time k. Define vector of past outputs as follows:

$$\underset{d_{\max}m\times 1}{y_{\text{past}}}(\kappa) = \left(\begin{array}{c} y(\kappa-1) \\ \hline \vdots \\ \hline y(\kappa-d_{\max}) \end{array}\right). \tag{9.9}$$

The future signals are, correspondingly,

$$\underset{d_{\max}m\times 1}{y_{\text{future}}}(\kappa) = \left(\begin{array}{c} y(\kappa+d_{\max}-1) \\ \hline \vdots \\ \hline y(\kappa) \end{array}\right).$$

The data matrices Y_{past} and Y_{future} are constructed as before, stacking transposed sample vectors on top of each other, the indexes running from $d_{\max}+1$ to $k-d_{\max}+1$. Note that the matrices have $k-2d_{\max}+1$ rows rather than the original k. The preliminary state variable can be constructed immediately:

$$X = \left(\begin{array}{c} Y_{\text{past}} \end{array}\right). \tag{9.10}$$

The dimension of the preliminary system state is high and the states are highly redundant — but the main thing here is that the originally dynamic problem has been transformed into a static one, and all of the tools that have been presented in the previous chapters for dealing with static models can be utilized for dimension reduction, or for determining the latent vector matrix Z containing the N dimensional minimum state vectors. According to system theoretic understanding, one should select $N = d$.

Compressing the state

There is still plenty of freedom. One can compress the state space utilizing the state sequence X exclusively, or one can take into account the fact that there should be a mapping from the state to the future states and outputs. These alternative viewpoints give rise to approaches of PCA/PLS/CCA type. Further, applying some independent component approach to the state selection should be something new in this field (see Exercises). Here, the PLS option is briefly studied.

One should determine the compressed state so that mapping from this state to next state and to the output could be accomplished.

Define X^- as a submatrix of X, where the *last* row is eliminated (the newest state vector), and, similarly, define X^+ as a submatrix, where the *first* row is eliminated (the oldest state vector):

$$X^- = \begin{pmatrix} x^T(1) \\ \vdots \\ x^T(k-2d_{\max}) \end{pmatrix} \quad \text{and} \quad X^+ = \begin{pmatrix} x^T(2) \\ \vdots \\ x^T(k-2d_{\max}+1) \end{pmatrix}. \tag{9.11}$$

These matrices stand for the succession of states, that is, elements in X^+ are the next state variables corresponding to the state variables in X^-. Further, define (dimensionally and causally matching) output matrix

$$Y = \begin{pmatrix} y^T(d_{\max}) \\ \vdots \\ y^T(k-d_{\max}-1) \end{pmatrix}, \tag{9.12}$$

consisting of altogether $k-2d_{\max}$ rows. Now the intended mapping can be expressed so that the virtual input data and the virtual output data, respectively, are

$$\mathcal{X} = \left(\begin{array}{c} X^- \end{array} \right) \quad \text{and} \quad \mathcal{Y} = \left(\begin{array}{c|c} X^+ & Y \end{array} \right). \tag{9.13}$$

When the PLS model is constructed for this problem, and the latent variables are determined, it is this sequence of latent variable vectors (in the input-oriented subspace) that can be selected as the compressed state sequence, so that $Z = \mathcal{Z}$.

Typically, the selection of the state dimension will not be unique, so that the system order is more or less uncertain. Note that this uncertainty is an inherent property of real distributed parameter systems — the exact system dimension is just a mathematical abstraction that has been approved as there are no alternatives for representing the system behavior in such a compact form.

System matrices

To proceed, the resolved compressed state matrix Z needs to be restructured. This is done as in (9.11): Define Z^- as a submatrix of Z, where the *last* row is eliminated (the newest compresswed state vector), and, similarly, define Z^+ as a submatrix, where the *first* row is eliminated (the oldest compressed state vector):

$$Z^- = \begin{pmatrix} z^T(1) \\ \vdots \\ z^T(k-2d_{\max}) \end{pmatrix} \quad \text{and} \quad Z^+ = \begin{pmatrix} z^T(2) \\ \vdots \\ z^T(k-2d_{\max}+1) \end{pmatrix}. \tag{9.14}$$

These matrices stand for the succession of states. Also, define the output matrix as in (9.12).

When the state sequence is now known, the subsequent steps of subspace identification nicely illustrate the power of linear machinery — the final model (9.8)

can be constructed by matching the parameters against the now known state variables. Because the system model can be written as

$$\left(\frac{z(\kappa+1)}{y(\kappa)}\right) = \left(\frac{A}{C}\right)\left(\ z(\kappa)\ \right), \tag{9.15}$$

there holds

$$\left(\ Z^+ \ \middle|\ Y\ \right) = \left(\ Z^-\ \right)\cdot\left(\ A^T \ \middle|\ C^T\ \right). \tag{9.16}$$

The matrices A and C can be solved in the least squares sense:

$$\left(\frac{A}{C}\right) = \left(\ Z^+ \ \middle|\ Y\ \right)^T\cdot\left(\ Z^-\ \right)\Big(\left(\ Z^-\ \right)^T\cdot\left(\ Z^-\ \right)\Big)^{-1}, \tag{9.17}$$

where the individual system matrices can be identified as partitions of the resulting matrix the first $N \times N$ elements being allocated for A.

Noise covariances

The mismatch between data and the model gives the estimates for the noise. Because the state and output sequences estimated by the model can be written as

$$\left(\ \hat{Z}^+ \ \middle|\ \hat{Y}\ \right) = \left(\ Z^-\ \right)\cdot\left(\ A^T \ \middle|\ C^T\ \right), \tag{9.18}$$

the estimation error becomes

$$\begin{array}{rcl} E & = & \left(\ Z^+ \ \middle|\ Y\ \right) - \left(\ \hat{Z}^+ \ \middle|\ \hat{Y}\ \right) \\ & = & \left(\ Z^+ \ \middle|\ Y\ \right) - \left(\ Z^-\ \right)\cdot\left(\ A^T \ \middle|\ C^T\ \right), \end{array} \tag{9.19}$$

and the noise covariances can be mechanically solved from this. The approximations for the covariances are again found as partitions of

$$\left(\begin{array}{c|c} R_{zz} & R_{zy} \\ \hline R_{zy}^T & R_{yy} \end{array}\right) = \frac{1}{k'}\cdot E^T E. \tag{9.20}$$

Here, the normalization factor $k' = (k - 2d_{\max} + 1) - (N + m)$ is the number of data samples minus the overall data dimension. Again, in typical applications (like when doing Kalman filtering, see below), it is the ratio between R_{zz} and R_{yy} that is the most important information, not the absolute scalings.

Assuming that the input signal exactly can explain the output, the error covariances are zero matrices. In such cases one is facing the *deterministic realization problem;* it turns out that the realization problem is solved as a special case of the more general subspace identification problem. The above discussion can be further extended.

9.2.2 Stochastic-deterministic models

A more complicated case is faced when the complete model (9.4) is employed. Now define sequences of past inputs and outputs as follows:

$$u_{\text{past}}(\kappa) = \left(\begin{array}{c} u(\kappa-1) \\ \hline \vdots \\ \hline u(\kappa-d_{\max}) \end{array}\right) \quad \text{and} \quad y_{\text{past}}(\kappa) = \left(\begin{array}{c} y(\kappa-1) \\ \hline \vdots \\ \hline y(\kappa-d_{\max}) \end{array}\right) \tag{9.21}$$

The future signals are, correspondingly,

$$u_{\text{future}}(\kappa) = \left(\begin{array}{c} u(\kappa+d_{\max}-1) \\ \hline \vdots \\ \hline u(\kappa) \end{array}\right) \quad \text{and} \quad y_{\text{future}}(\kappa) = \left(\begin{array}{c} y(\kappa+d_{\max}-1) \\ \hline \vdots \\ \hline y(\kappa) \end{array}\right).$$

The data matrices U_{past}, U_{future}, Y_{past}, and Y_{future} are again constructed in the same way as above.

To find a good model for the data in this more complicated case, one should distribute the burden of explaining the future behaviors appropriately among the two sources of information — the known history, and the unknown future inputs. One has to apply the mathematical theory of *oblique projections*. To give an idea of what this means, construct matrices

$$\mathcal{X} = \left(\begin{array}{c|c|c} Y_{\text{past}} & U_{\text{past}} & U_{\text{future}} \end{array}\right) \tag{9.22}$$

and

$$\mathcal{Y} = \left(\begin{array}{c} Y_{\text{future}} \end{array}\right), \tag{9.23}$$

so that there should exist a mapping from $\mathcal{X}$ to $\mathcal{Y}$. Indeed, this can be constructed using least squares matching, giving a prediction model for the future:

$$\begin{array}{rcl} \mathcal{Y}_{\text{est}} & = & \mathcal{X}_{\text{est}} \cdot \left(\mathcal{X}^T\mathcal{X}\right)^{-1}\mathcal{X}^T\mathcal{Y} \\ & = & \left(\begin{array}{c|c|c} Y_{\text{past,est}} & U_{\text{past,est}} & U_{\text{future,est}} \end{array}\right) \cdot \left(\mathcal{X}^T\mathcal{X}\right)^{-1}\mathcal{X}^T\mathcal{Y}. \end{array} \tag{9.24}$$

In practice, the invertibility of $\mathcal{X}^T\mathcal{X}$ may be poor, specially id $d_{\max}$ has too high value as compared to the system dimension, and some dimension reduction technique may again be needed.

The values of $U_{\text{future,est}}$ are not known at time κ, and to make this model useful, it has to be divided in parts:

$$\begin{array}{rcl} \mathcal{Y}_{\text{past}} + \mathcal{Y}_{\text{future}} & = & \left(\begin{array}{c|c|c} Y_{\text{past,est}} & U_{\text{past,est}} & \mathbf{0} \end{array}\right) \cdot \left(\mathcal{X}^T\mathcal{X}\right)^{-1}\mathcal{X}^T\mathcal{Y} \\ & & + \left(\begin{array}{c|c|c} \mathbf{0} & \mathbf{0} & U_{\text{future,est}} \end{array}\right) \cdot \left(\mathcal{X}^T\mathcal{X}\right)^{-1}\mathcal{X}^T\mathcal{Y}. \end{array} \tag{9.25}$$

The variables in $\mathcal{Y}_{\text{past}}$ can be interpreted as containing all available information about the system past. Based on this, one can define the "refined" data matrix where the contribution of the future inputs is eliminated:

$$X = \mathcal{Y}_{\text{past}} = \left(\begin{array}{c|c|c} Y_{\text{past,est}} & U_{\text{past,est}} & \mathbf{0} \end{array}\right) \cdot \left(\mathcal{X}^T\mathcal{X}\right)^{-1}\mathcal{X}^T\mathcal{Y}. \tag{9.26}$$

It is possible to choose this data as constituting the preliminary system states. The determination of the reduced state matrix Z can be accomplished exactly as was done before.

Again, the resolved state matrix Z is restructured into Z^- and Z^+ parts. Further, define (dimensionally matching) input and output matrices

$$U = \begin{pmatrix} u^T(d_{\max}) \\ \vdots \\ u^T(k - d_{\max} - 1) \end{pmatrix} \quad \text{and} \quad Y = \begin{pmatrix} y^T(d_{\max}) \\ \vdots \\ y^T(k - d_{\max} - 1) \end{pmatrix}, \tag{9.27}$$

consisting of altogether $k - 2d_{\max}$ rows. Now, because the system model can be written as

$$\left(\frac{z(\kappa+1)}{y(\kappa)}\right) = \left(\frac{A \mid B}{C \mid D}\right)\left(\frac{z(\kappa)}{u(\kappa)}\right), \tag{9.28}$$

there holds

$$\left(\; Z^+ \;\middle|\; Y \;\right) = \left(\; Z^- \;\middle|\; U \;\right) \cdot \left(\frac{A^T \mid C^T}{B^T \mid D^T}\right). \tag{9.29}$$

The matrices A, B, C, and D can be solved in the least squares sense:

$$\begin{aligned} \left(\frac{A \mid B}{C \mid D}\right) \;\; = \;\; & \left(\; Z^+ \;\middle|\; Y \;\right)^T \cdot \left(\; Z^- \;\middle|\; U \;\right) \\ & \cdot \left(\left(\; Z^- \;\middle|\; U \;\right)^T \cdot \left(\; Z^- \;\middle|\; U \;\right)\right)^{-1}, \end{aligned} \tag{9.30}$$

where the individual system matrices can be identified as partitions of the resulting matrix (again, note the possible invertibility problems). The reconstruction errors give the estimates for the noise. Because the state and output sequences estimated by the model can be written as

$$\left(\; \hat{Z}^+ \;\middle|\; \hat{Y} \;\right) = \left(\; Z^- \;\middle|\; U \;\right) \cdot \left(\frac{A^T \mid C^T}{B^T \mid D^T}\right), \tag{9.31}$$

the estimation error becomes

$$\begin{aligned} E \;\; &= \;\; \left(\; Z^+ \;\middle|\; Y \;\right) - \left(\; \hat{Z}^+ \;\middle|\; \hat{Y} \;\right) \\ &= \;\; \left(\; Z^+ \;\middle|\; Y \;\right) - \left(\; Z^- \;\middle|\; U \;\right) \cdot \left(\frac{A^T \mid C^T}{B^T \mid D^T}\right), \end{aligned} \tag{9.32}$$

and one has

$$\left(\frac{R_{zz} \mid R_{zy}}{R_{zy}^T \mid R_{yy}}\right) = \frac{1}{k'} \cdot E^T E. \tag{9.33}$$

9.3 Practical aspects

Subspace identification differs much from traditional identification methods, and its properties are also different from what one is familiar with. Understanding these boundary conditions is necessary to effriciently exploit the techniques.

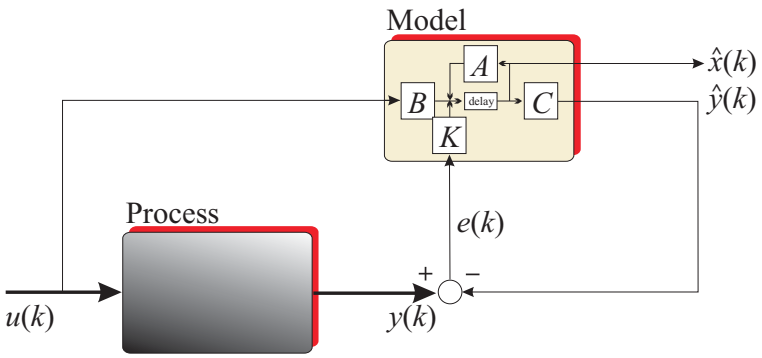

Figure 9.1: Making the process behaviors transparent

9.3.1 Comparisons

As this subspace identification approach is compared to the standard black-box identification procedures, some clear benefits can be seen:

- The selection of the system dimension needs not be done beforehand as in traditional black-box identification; the algorithms give tools for appropriate on-line determination of the dimension.
- The models are naturally applicable to multi-input/multi-output (MIMO) systems, whereas traditional identification only studies systems of one input and one output.
- Coloured noise becomes modeled without additional efforts, whereas standard techniques have to rely on complicated nonlinear, iterative methods.

The dynamical phenomena that are visible in the data are all integrated in the model structure itself, no matter where those phenomena originate from. Indeed, the properties of observed noise are also characteristic to the system and it is natural that this information is also coded in the model in a consistent way. Some state variables may be allocated for presenting the noise — if this dynamics is relevant enough.

However, there are also some drawbacks as the dynamic modeling approach is compared to the static approach. The model is more sophisticated, and there are more quality requirements concerning the data: Longer sequences of valid data are needed, as no outlier or missing data samples can be dropped from within the sequence. For the same reason, also model validation becomes problematic, because longer continuous sequences of validation data are needed; the leave-one-out cross-validation (with one sample being dropped at a time) cannot be implemented.

9.3.2 Emulating the process

A natural way to exploit the models given by subspace identification is reached through *state estimation:* When the process state is known, all state-based analysis and design methods are available. Typically, one can only measure the

inputs and outputs of a complex process; now, one can implement a *process simulator,* and this simulator, rather than being a black-box system, is a *white-box system* with explicitly known structure and measurable state. When simulator has the same dynamics as the process, and when it is given the same inputs as the actual process has, the behaviors of the simulator and the process itself should be equal (see Fig. 9.1). The feedback matrix K is used for correcting the state according to the observed error $e(k) = y(k) - \hat{y}(k)$, or difference between process and model outputs.

The *Kalman filter* is a natural companion of models derived by subspace identification: The feedback matrix K can be optimally determined based on the observed system and noise properties [**?**]. The model structure with the matrices A, B, C, and D are explicitly given by subspace identification, and even the noise properties R_{xx}, R_{yy}, and R_{xy}, are solved, making the determination of the Kalman filter a straightforward task.

Often, specially when the system dimension is high, determining all the system matrices is a difficult task using traditional techniques — and the matrices R_{zz} and R_{yy} are typically used as tuning parameters only, typically chosen to be simply diagonal. Now, using subspace identification, all these data structures are determined to optimally match the data. However, as it turns out in the next chapter, optimality and robustness are different things also in this case. Understanding the nature of the problem — again caused by the multivariate nature and collinearity — makes it possible to attack the problems.

9.3.3 Preprocessing and postprocessing

When applying subspace identification, model construction can be affected again by appropriately scaling of the data samples. In addition to the normal scalings, etc., there also exist other ways to affect the modeling results. Specially, because of the dynamic nature of the problem, or because there is a time-domain succession of the samples, one can apply *frequency weighting* to emphasize (or attenuate) variations in some frequency ranges. Here, one would not like to affect the model construction process itself (compare to [**?**]) — the goal is to implement the frequency weighting directly in the data.

Assume that there exists some linear dynamic model between inputs and outputs, so that the mapping matrix $F(q^{-1})$ contains delay operator polynomials

$$y(k) = F^T(q^{-1})\, u(k). \tag{9.34}$$

Now, the model remains valid if the left-hand side and the right-hand side are further filtered by the operator polynomial $h(q^{-1})$; because of the linearity assumption, the operators are commutative:

$$h(q^{-1})y(k) = h(q^{-1})F^T(q^{-1})\, u(k) = F^T(q^{-1})\, h(q^{-1})u(k). \tag{9.35}$$

If one defines the new data as $u'(\kappa) = h(q^{-1})u(\kappa)$ and $y'(\kappa) = h(q^{-1})y(\kappa)$, for this new data the original model still applies:

$$y'(k) = F^T(q) \cdot u'(k). \tag{9.36}$$

The system structure, and also the state-space model, can be determined for the modified data as well as for the original data. Nominally there is no change in the system — however, as always when modeling is based on the variance properties, the results change as errors are weighted in different ways in different frequency bands, depending on the ratios between information and noise along the frequency axis. In practice, one can apply low-pass, band-pass, band-stop, or high-pass filtering, as studied below.

It is very common in real processes that the levels of the signals can vary, even though the dynamic properties essentially remain intact. If one applies the standard mean-centering for training data, this mean does not necessarily remain constant, and the biased model can become invalid. A simple way to avoid this problem is to apply "on-line centering", or high-pass filtering, so that the zero-frequency properties are eliminated altogether:

$$h(q^{-1}) = 1 - q^{-1}, \tag{9.37}$$

meaning that the differentiated variables (variable vectors) in time domain are defined as

$$\Delta y(\kappa) = y(\kappa) - y(\kappa - 1). \tag{9.38}$$

However, assuming that there exists high-frequency noise in the system, differentiation emphasizes such noise signals. It can be reasonable to define some upper limit frequency for the filter, for example, by further filtering

$$y'(\kappa) = \lambda\, y'(\kappa - 1) + (1 - \lambda)\ \Delta y(\kappa). \tag{9.39}$$

This lead-lag compensator can also be applied for inputs and outputs alike, and subspace identification is applied for those signals. However, when the model is used for estimation, data preprocessing has to be ripped off by applying inverse postprocessing to reach the actual signal estimates. First, the differentiated signal estimates are found by inverting (9.39):

$$\Delta\hat{y}(\kappa) = \frac{1}{1-\lambda}\,\hat{y}'(\kappa) - \frac{\lambda}{1-\lambda}\,\hat{y}'(\kappa - 1). \tag{9.40}$$

In principle, the differentiation within these signals can be eliminated by integration — however, no pure integrators should be applied, as biases in signals would increase during this process. It is motivated to introduce a "leaking integrator" that eventually tends towards the actual measurements:

$$\hat{y}(\kappa) = \mu\ (\hat{y}(\kappa - 1) + \Delta\hat{y}(\kappa)) + (1 - \mu)\ y(\kappa). \tag{9.41}$$

Here, the parameters λ and μ are forgetting factors to be adjusted appropriately to fit the signal and process properties. The pure discrete-time derivator in (**??**) should also be modified to match the inverse operation (9.41).

9.4 Case study: Towards "smart devices"

Here, an example is presented where the above tools are being experimented and exploited. This research and development work is currently being carried

out in the industrial scale. It is clear that the full potential of the new methods cannot yet be seen.

9.4.1 Data based data reconciliation

In process industries knowing the contents of processed materials is of utmost importance. An *X-ray fluorescence analyzer* is an efficient tool when doing such analyses in mineral processing. The X-ray fluorescence analyzer excites the atoms and measures the emitted photons: The emission spectra are unique to the atoms, and, in principle, the atom contents can be solved by analyzing the spectral peaks. However, because of the environmental conditions and because of the physical reasons, the results are corrupted by noise and other stochastic phenomena. To enhance the estimates, one needs *calibration models* to map between the measured spectra to actual concentration estimates. The superposition of individual spectra is a linear process, and it is plausible that linear multivartiate methods can efficiently be used for this calibration purpose. And, indeed, such developments have been taking place recently (for example, see [**?**]).

Even though the static mapping models from the intensities to concentrations could be appropriately constructed, there is need for closer studies. It does not help if the sample could be analyzed exactly, if that sample is not representative. The random samples do not necessarily reflect the overall contents of the slurry — this is due to the changes in slurry densities and grain size distributions. To dampen the variability in the measurements, different kinds of filtering techniques have been applied. The traditional approach is exponential filtering with some forgetting factor: That is, one only partly trusts the measurements, keeping the prior average level of estimates as the starting point. If there are some consistent changes in the slurry properties, such filtered estimate becomes delayed, as the average level only slowly follows the changes. There is a trade-off between accuracy and smoothness of estimates. It is not the variability that should be dampened, if it reclects the reality; but there is no way to locally determine whether some change in the measurement values is relevant or not. Are there other ways to enhance the estimates?

Data reconciliation is a bunch of techniques for employing the system structure for enhancing the noisy measurements. Typically, data reconciliation is based on mass or volume balances: If the inputs and outputs of some process structure are recorded, the mass/volume flows should compensate each other. For example, a flotation cell (or a bang of cells) is such a vessel: The two outgoing flows (concentrate and gangue flows) have to balance the incoming slurry flow. Because of the practical challenges, data reconciliation is often carried out statically, for steady-state levels, so that the flows compensate each other only in the long run.

However, often the structures or dependencies within a complex process are not known, and there exist no explicit constraint equations. And even if the constraints were known, there may be no measurements: For example, an X-ray analyzer is such an expensive device that measurements are carried out only in the most informative locations in the process. But, after all, data reconciliation is based on the redundancy among data, and if the measurements are cut to

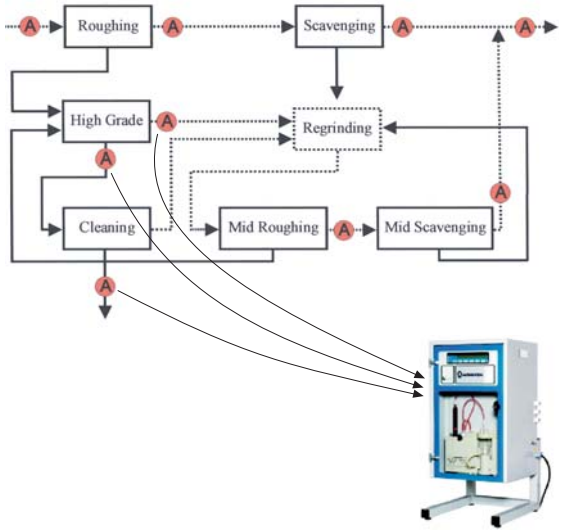

Figure 9.2: The analyzer is already the "heart" of the process — make it also the "brain"

minimum, the crucial measurements closing the logical loops may be missing. Still, there most probably exist some interdependencies between signals. How to implement the idea of data reconciliation as a robust enough scheme, applicable in such a complex environment — and, what is more, how to extend data reconciliation to dynamic cases?

The presented ideas of statistical multivariate modeling, and specially the ideas of subspace identification, make it possible to implement "data based data reconciliation". As an example case, again study the Pyhäsalmi concentrator plant (see chapter 3).

The zinc circuit alone is a rather complicated network (see Fig. 9.2), different recirculations having been implemented to enhance the recovery rate and purity of the concentrate. The current structures in the process are a result of an evolutionary process, and they are characteristic to this unique process plant. In addition to the physical feedforward and feedback flows, there is yet another level of information flows, being caused by the control structures, making the overall system fully connected and "pancausal". Indeed, the degrees of freedom in the system are reduced by the interconnected variables, and the remaining variation assumedly takes place only in a rather low-dimensional subspace. Even if all the constraints cannot be explicitly tracked, it can be assumed that in the cybernetic system the balance around the steady-state average is maintained, and it suffices to capture the properties of the "emergent model" (see chapter 11). Because of the balance and assumed minimization of signal variations, local linearity can be assumed, at least if no structural changes take place in the process.

The concentrations in the most relevant flows are measured: Slurry samples are taken to the centralized analyzer unit, where the samples are analyzed one after the other in a sequence, the whole cycle lasting some 20 minutes. However, significant changes can take place in the process during these time intervals, and to implement efficient control and monitoring applications, it would be of utmost importance to be capable of reliably estimating the signal behaviors during the

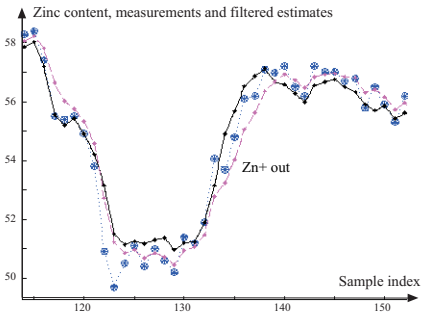

Figure 9.3: Original unfiltered measurements are shown as dots; the "50% smoothened", exponentially filtered estimates are shown along dashed lines, and the model-based estimates are along the solid lines. The new filtering scheme is considerably faster

periods of "blindness". One should implement *soft sensors* based on process models; to construct the models that utilize the available measurements, one has to implement *sensor fusion.* It is clear the slurry concentrations in different parts of the process are interrelated, and utilizing these relationships, behaviors "downstream" can be estimated. Rather than concentrating on the individual signals, one should find the overall dependencies — this global-level state of the whole zinc circuit can be exploited for estimation. The correlations between measurements are rather low, but if many pieces of evidence are combined in a clever way, useful models can still be found.

What is this *clever way,* then? It is evident that subspace identification, as accompanied by the appropriate Kalman filter, is the method of choice: As the high dimensionality is no problem, the various measurent channels can be efficiently exploited, and also the minor pieces of correlation information can be extracted. Depending on the available information, different model structures are possible:

1. **Stochastic model.** The measurements constitute the output vector y, and the task is to determine the system state vector z, assuming that the changes in the system state are driven by noise. The internal system model makes it possible to enhance the "downstream" estimates by exploiting the "upstream" observations.

2. **Stochastic-deterministic model.** As explained in chapter 3, there are also cameras installed on top of the flotation cells: The features extracted from the camera images can be employed as causally preceding information, giving delayless data of the froth outlook (color, "thickness", etc.) that is assumedly related with the concentrate properties.

It is a closed-loop system that is being implicitly identified. The causalities are blurred, one cannot distinguish between the process and the controls. If further feedbacks or process re-design is to be implemented, the derived models have to be interpreted by a domain-area expert.

In Fig. 9.3, results are shown when the zinc concentrate from the "Cleaning" cell is estimated utilizing also the measurement information from the "High Grade" cell (see Fig. 9.2). The preliminary experiments with the subspace identification schemes are promising, and faster reactions to changes in concetrate properties can be reached. When the estimates are "calibrated" after each measurement, the signals can be estimated during the inter-sample periods using the model. However, it turns out that the data quality needs to be

emphasized: Just as global information is exploited for local estimation, local problems become global, perhaps ruining the whole plant operation. The outlier values need to be detected and fixed reliably on-line, so that new challenges are faced.

9.4.2 Connections to AI*

The discussion of "clever" data analyses can be extended.

In Artificial intelligence (AI) one tries to implement "intelligent-looking" functionalities in computers. Traditionally, AI approaches are symbolic, meaning that the constructs, declarative or procedural, need to be explicitly programmed: There is then no connection to data, and there is no possibility of adaptation. Human is needed as an interface between the system and its environment. Real intelligence can be defined as ability of appropriate reacting to its environment and adapting according to it, and fixed structures are necessarily deficient.

The neural netwroks, etc., have been exploited to reach data-orientedness and associative reasoning, and they can be applied to accomplish complicated pattern recognition tasks. However, something essential seems to be missing: The structures themselves are still fixed, non-adaptive. How to find a good compromise, a structural framework where adaptation is possible?

Structures that characterize human cognition are causal, meaning that human mind naturally organizes observations in cause/effect hierarchies. Such causality structures cannot be seen in the data — but, again, one can do assumptions of how the structures are reflected in data. If it is assumed that correlation structures can be used to represent causalities, temporal ordering among data can be found applying the techniques presented above. When the data dimension is high and redundant, the challenge of the modeling method is to cope with the high dimensionality, and detect the appropriate connections — getting "wiser" is about ignoring irrelevant connections. After adaptation the dynamic state models become "numeric inference models" — when given the current state, the system can predict what happens afterwards. Many philosophical problem settings become very concrete: Questions concerning *ontologies* and *semantics* are wiped away, as everything is based on *contextual semantics* defined in terms of similarities, or correlation structures among variables. As the "numeric concepts", or the state variables, have continuous values, the problems of "hermeneutic circles", or the convergence properties of infinite recursions can be studied mathematically. In this way, it seems that multivariate modeling can offer new possibilities for research in AI.

On the other hand, models derived in the field of AI can perhaps give new tools for extending the dynamic model structure beyond linearity. For example, the sparse coding schemes derived for cognitive tasks can perhaps be extended to dynamic applications.

Computer exercises

1. Study the properties of the subspace identification algorithm; try different parameter values in the following:

```
[U,Y] = dataDyn(3,2,1,100,0.001);
regrSSI(U,Y,5);
```

2. Analyze the different compression techniques in state dimension reduction. First, define time series data as

```
[u,y] = dataDyn(1,0,1,100,0.1);
X = [y(1:96),y(2:97),y(3:98),y(4:99),y(5:100)];
Xminus = X(1:size(X,1)-1,:);
Xplus = X(2:size(X,1),:);
```

   Compare the first principal component basis vector to the system transient behavior:

```
thetaPCA = regrPCA(X,1)
```

   Compare PLS and CCA. How can you explain the differences in the eigenvalue behavior? How about the latent vectors?

```
[thetaPLS,phiPLS] = regrPLS(Xminus,Xplus)
[thetaCCA,phiCCA] = regrCCA(Xminus,Xplus)
```

   What happens to the independent components in a dynamic system? Can you explain the results?

```
U = dataIndep(1000,'f1','f2');
[U,Y] = dataDyn(2,U,3);
X = [Y(1:999,:),Y(2:1000,:)];
thetaICA = regrICA(regrWhiten(X),-1)
```