Lesson 10

Relations to Systems Engineering

This far we have been exclusively interested on data, and on models that are constructed directly from that data. It seems that the traditional control engineering practices that are usually based on first principles models would not have very much to do with these discussions. However, finally it is numeric data that flows through these physical models, too, even if the model construction had been based on qualitative studies; the structure of the model determines how the data is manipulated in the system, directly dictating how nice behavioral properties that system has. In this chapter we study how the multivariate ideas and approaches can be applied for analysis and manipulation of existing system models so that the *expected behavior* of the hypothetical data being processed in the system is taken into account.

In this context, some specially important aspects of modern control engineering will be concentrated on, among them *balanced model reduction* and *state estimation.*

10.1 MIMO vs. SISO systems

Traditional control design has been based on SISO (single input, single output) models, meaning that complex plants are reduced to simple control loops. This is natural, because the SISO systems are easily comprehensible for the system designers as well as for field operators. Yet, it is clear that local optimization of single control loops does not result in optimal behavior of the larger plant, and there is need to be able to design and analyze more complex models of multiple interconnected inputs and outputs (MIMO systems).

Because of the intuitive understandability and long tradition of SISO systems, the main paradigm in MIMO control design has been to somehow extend the SISO design ideas to the multivariate cases, or, rather, to make the MIMO systems *look like* sets of SISO systems.

One usually starts from the transfer function matrix

$$G(s) = \begin{pmatrix} G_{11}(s) & \cdots & G_{1,n}(s) \\ \vdots & \ddots & \\ G_{m,1}(s) & & G_{m,n}(s) \end{pmatrix}, \tag{10.1}$$

that characterizes a linear, dynamic multivariate system of n_u inputs and m outputs in Laplace (frequency) domain; the algebraic dependency $\mathcal{L}\{y(t)\}(s) = G(s) \cdot \mathcal{L}\{x(t)\}(s)$ governs the behavior of the Laplace transformed input and output signal vectors $\mathcal{L}\{u(t)\}(s)$ and $\mathcal{L}\{y(t)\}(s)$. Sustituting $s \to j\omega$ in (10.1), the transfer properties of sinusoidals of angular frequency ω are directly given, the absolute value revealing the amplitude and the angle of the complex-valued number revealing the phase of the output sinusoidal. Now, it is immediately clear that the interconnections between different inputs and outputs can be minimized if the matrix (10.1) is somehow *diagonalized,* and, indeed, constructing the singular-value decomposition, this can be done:

$$G(j\omega) = \Xi(\omega) \cdot \Sigma(\omega) \cdot \Psi^H(\omega), \tag{10.2}$$

the matrix $\Sigma(\omega)$ containing the Hankel singular values $\sigma_i(j\omega)$ (to be discussed more later) on the diagonal. This means that if the measured Laplace-domain output signals are multiplied by $\Psi^H(\omega)$ and the constructed control signals by $\Xi(\omega)$, the system looks like a set of separate SISO systems, and direct SISO design is possible for all input-output pairs. However, note that the matrices $\Xi(\omega)$ and $\Psi(\omega)$ are complex valued, and they are functions of ω; using constant (real) matrices, this diagonalization can only be approximate. To easily get rid of complex factors, the diagonalization is typically optimized for zero frequency.

The frequency domain stability and sensitivity analysis techniques (stability margins, etc.) are also originally developed for SISO systems. These techniques can easily be extended to the multivariate case, if one is only interested in the worst-case behavior: The above Hankel singular values $\sigma_i(j\omega)$ reveal the system gains, and studying the largest of them, the *principal gain* $\bar{\sigma}_i$ for all frequencies, gives information about the maximum possible system response, given the most pathological input signal. Constructing controllers concentrating on the open-loop principal gains, it is possible to assure system stability in all situations and for all inputs (for example, see [30]).

10.2 Dimension reduction

The traditional way to apply multivariate techniques (as discussed above) are somewhat crude, not really taking into account the dynamic nature beneath the numbers, but forcing the complex systems into the SISO framework. In what follows, more sophisticated approaches discussed.

10.2.1 About state-space systems

The basis of modern systems engineering is the state-space model; as compared to the models in the previous chapter, now we start with its deterministic version

of it:

$$\begin{cases} x(\kappa+1) = Ax(\kappa) + Bu(\kappa) \\ y(\kappa) = Cx(\kappa) + Du(\kappa). \end{cases} \tag{10.3}$$

Here, u and y are the system input and output of dimension n_u and m, respectively, and x is the state of dimension d. It needs to be noted that, from the input/output behavior point of view, the state realization is not unique: For example, if there is an invertible matrix L of size $d \times d$, one can define another state vector $x' = Lx$ so that the following state-space model fulfills the same relationship between $u(\kappa)$ and $y(\kappa)$:

$$\begin{cases} x'(\kappa+1) = LAL^{-1}x'(\kappa) + LBu(\kappa) \\ y(\kappa) = CL^{-1}x'(\kappa) + Du(\kappa). \end{cases} \tag{10.4}$$

Intuitively, the strength of the state-space model is that the real internal phenomena within the system can also be included in the model. In such cases, when the model has been constructed starting from first principles, the state representation for the system may be unique, and state transformations in the above way are meaningless. However, when the modern controller construction approaches like robust or optimal control theory are applied, the resulting controller is also given in the state-space form — but now the states are constructed by the mathematical machinery and they do no more have any physical relevance. The only thing that counts is the input-output behavior of the controller.

The question that arises is: How should we define the state vector of the model so that some of the model properties would be enhanced? Or, more specifically, how should the matrix L be selected so that the *essence* of the state components would be revealed[1]?

When the controller is constructed using robust or optimal control theory, the resulting controller is usually high-dimensional. There are practical difficulties when using this kind of controllers — it may take long time to calculate the control actions and it may take a long time before the controller reaches the correct operating state after startup. In many cases, the models need to be simplified for practical purposes.

It is typical that some of the states are more relevant than the others. Could we reduce the number of states without losing too much, without essentially changing the dynamic behavior of the model? It sounds plausible that the methodologies presented in the previous chapters could be applied for this purpose, and, indeed they can. Now it is the most relevant dynamic modes that we would like to retain, while ignoring the less significant ones. We would like to determine the transformation matrix L so that the relevance of different variables would be easily assessed in the resulting state vector[2].

[1] What is this "essence" in the data — again, it can be noted that mathematical machinery cannot judge the real relevance of the phenomena. But if we are less ambitious, useful results can be reached

[2] A traditional way to solve this model reduction problem was to concentrate on the most significant modes, those that are nearest to the unit circle, thus being slowest and decaying last, and simply "forget" about the less significant modes altogether. Often this approach works — the slowest decaying modes are visible longest and determine the overall system behavior. However, pole-zero studies miss one crucial point: The gains of the modes cannot

10.2.2 Preliminary experiments

For a moment, study the following simplified system model:

$$\left\{ \begin{array}{rcl} x(\kappa+1) & = & Ax(\kappa) \\ y(\kappa) & = & Cx(\kappa). \end{array} \right. \tag{10.5}$$

Assume that A can be diagonalized, so that there exists a matrix of eigenvecors Θ so that there holds

$$A = \Theta \cdot \Lambda \cdot \Theta^{-1}. \tag{10.6}$$

Defining $x' = \Theta^{-1}x$, the model (10.5) can be written as

$$\left\{ \begin{array}{rcl} x'(\kappa+1) & = & \Lambda x'(\kappa) = \left(\begin{array}{ccc} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{array} \right) \cdot x'(\kappa) \\ y(\kappa) & = & C\Theta \cdot x'(\kappa). \end{array} \right. \tag{10.7}$$

Now it is evident that all modes have been separated — each of the state elements $x_i(\kappa)$ only affects itself; furthermore, the eigenvalues λ_i determine their decay rates. However, there are problems. First, the matrix A may not be diagonalizable; what is more, when the general model (10.3) is studied, this diagonalization of the matrix A alone cannot guarantee independence of the modes: It is the matrix B that routes the exogenous input $u(\kappa)$ into the system, and depending on the connections between the state elements as determined by B, the modal structure gets blurred, state elements becoming mutually correlated; this approach clearly does not solve the problem.

Of course, one possibility would be to analyze again the state covariance matrix $\frac{1}{k} \cdot \sum_{\kappa=1}^{k} x'(\kappa)x'^T(\kappa) = L \cdot \frac{1}{k} \cdot \sum_{\kappa=1}^{k} x(\kappa)x^T(\kappa) \cdot L^T$ in the same way as in the previous chapters, defining the state transformation matrix L so that the covariance becomes identity matrix. This procedure separates the states — but there are various problems: First, the *output* is not taken care of in the construction of x'; second, this kind of transformation is not determined exclusively by the system structure but also by the input signal properties — this kind of dependency of the external conditions is not what is needed.

Something more sophisticated needs to be done. What one would like to have is a state reduction technique that would find a realization that is only dependent of the system structure, and that would be somehow *balanced* between input and output.

---

be seen if only the locations of the transfer function roots are studied. There may be a mode that is located far from the unit circle but having so high gain that — even though decaying fast — its transients start from such high values that they are still most significant. On the other hand, the qualitative pole-zero analyses cannot reveal the role of interactions between the modes: A nearby zero can essentially shadow the effects of some otherwise relevant pole. What is needed is a numerical methodology for reducing the system order

10.2.3 Balanced realizations

It turns out to be a good strategy to study how the signal *energy* is transferred through the system; how the *past inputs* affect the *future outputs* through the system state variables [10].

Let us study how the state x at time κ relays the signal *energy* from input to output in the model (10.3). First, calculate the contributions of the past inputs on the current state (assuming system stability):

$$x(\kappa) = Bu(\kappa-1) + ABu(\kappa-2) + A^2Bu(\kappa-3) + \cdots$$

This can be expressed as

$$\begin{aligned} x(\kappa) &= \left(\begin{array}{c|c|c|c} B & AB & A^2B & \cdots \end{array} \right) \cdot \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ \vdots \end{pmatrix} \\ &= M_C \cdot \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ \vdots \end{pmatrix}, \end{aligned} \tag{10.8}$$

where M_C is the (extended) *controllability matrix.* On the other hand, the effect of the current state $x(\kappa)$ on the future outputs can be written as

$$\begin{cases} y(\kappa) = Cx(\kappa) & \text{Output at time } \kappa \\ y(\kappa+1) = CAx(\kappa) & \text{Output at time } \kappa+1 \\ y(\kappa+2) = CA^2x(\kappa) & \text{Output at time } \kappa+2 \\ \quad \vdots \end{cases} \tag{10.9}$$

This can be written in a compact form using the (extended) *observability matrix* M_O:

$$\begin{pmatrix} y(\kappa) \\ y(\kappa+1) \\ y(\kappa+2) \\ \vdots \end{pmatrix} = \begin{pmatrix} C \\ \hline CA \\ \hline CA^2 \\ \hline \vdots \end{pmatrix} \cdot x(\kappa) = M_O \cdot x(\kappa). \tag{10.10}$$

These two expressions (10.8) and (10.10) can now be combined, resulting in the expression for signal that goes from input to output through the single state $x(\kappa)$:

$$\begin{pmatrix} y(\kappa) \\ y(\kappa+1) \\ y(\kappa+2) \\ \vdots \end{pmatrix} = M_O M_C \cdot \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ u(\kappa-3) \\ \vdots \end{pmatrix}. \tag{10.11}$$

The matrix $H = M_O M_C$ is known as the *Hankel matrix.* For a discrete system, the Hankel matrix has a close connection to the system properties, and it can

be constructed simply from its pulse response: The element on the i'th row and j'th column in the Hankel matrix is the $i+j-1$'th element of the system pulse response.

The total power of the output (summing the powers of all output signals together) can be expressed as

$$\begin{aligned} & y^T(\kappa)y(\kappa) + y^T(\kappa+1)y(\kappa+1) + y^T(\kappa+2)y(\kappa+2) + \cdots \\ & \quad = \begin{pmatrix} y(\kappa) \\ y(\kappa+1) \\ y(\kappa+2) \\ \vdots \end{pmatrix}^T \begin{pmatrix} y(\kappa) \\ y(\kappa+1) \\ y(\kappa+2) \\ \vdots \end{pmatrix} \\ & \quad = \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ u(\kappa-3) \\ \vdots \end{pmatrix}^T \cdot M_C^T M_O^T M_O M_C \cdot \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ u(\kappa-3) \\ \vdots \end{pmatrix}. \end{aligned} \tag{10.12}$$

In the static case, structuring of data was reached through maximization of (co)variance; similarly, it turns out that the power transfer properties can be structured through an optimization procedure: Now the goal is to maximize the power in output when the total input power is fixed (but the power in input may be arbitrarily distributed):

$$\begin{aligned} & u^T(\kappa-1)u(\kappa-1) + u^T(\kappa-2)u(\kappa-2) + \cdots \\ & \quad = \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ u(\kappa-3) \\ \vdots \end{pmatrix}^T \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ u(\kappa-3) \\ \vdots \end{pmatrix} \\ & \quad = 1. \end{aligned} \tag{10.13}$$

The criterion to be maximized also becomes

$$\begin{aligned} \text{Maximize} \quad & \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ u(\kappa-3) \\ \vdots \end{pmatrix}^T \cdot M_C^T M_O^T M_O M_C \cdot \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ u(\kappa-3) \\ \vdots \end{pmatrix}, \\ \text{when} \quad & \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ u(\kappa-3) \\ \vdots \end{pmatrix}^T \begin{pmatrix} u(\kappa-1) \\ u(\kappa-2) \\ u(\kappa-3) \\ \vdots \end{pmatrix} = 1. \end{aligned} \tag{10.14}$$

The method of Lagrangian multipliers can again be applied, and it turns out that, again, an eigenproblem is found:

$$M_C^T M_O^T M_O M_C \cdot u_i = \lambda_i \cdot u_i. \tag{10.15}$$

Elements of vector u_i have the same interpretation as the elements in the infinite dimensional input signal vectors above; the problem now is that the infinite

dimensional eigenproblem is slightly questionable! However, there is a nice trick available here: Multiply (10.15) from left by the matrix M_C, so that

$$M_C M_C^T M_O^T M_O \cdot M_C u_i = \lambda_i \cdot M_C u_i. \tag{10.16}$$

It turns out that the vector $M_C u_i$ must now be the eigenvector of the finite-dimensional matrix $M_C M_C^T M_O^T M_O$, the eigenvalues remaining the same as in the original eigenproblem. Note that this equality of eigenvalues holds only for the nonzero ones; the higher-dimensional problem of course has high number of additional eigenvalues, but they are all zeros. The matrix $M_C M_C^T \cdot M_O^T M_O$ consists of two low-dimensional parts:

- The *Controllability Gramian* contains only the input-related factors:

$$\begin{aligned} P_C &= M_C M_C^T = \sum_{\kappa=0}^{\infty} A^\kappa B B^T (A^T)^\kappa \\ &= BB^T + ABB^T A^T + A^2 BB^T A^{2T} + \cdots \end{aligned} \tag{10.17}$$

- The *Observability Gramian* contains only the output-related factors:

$$\begin{aligned} P_O &= M_O^T M_O = \sum_{\kappa=0}^{\infty} (A^T)^\kappa C^T C A^\kappa \\ &= C^T C + A^T C^T C A + A^{2T} C^T C A^2 + \cdots \end{aligned} \tag{10.18}$$

It is easy to show that the Gramians satisfy the linear matrix equations

$$\begin{aligned} AP_C A^T - P_C &= -BB^T \\ A^T P_O A - P_O &= -C^T C. \end{aligned} \tag{10.19}$$

Gramians are closely related to the controllability and observability properties of a system: If P_C is positive definite, the system is controllable, and if P_O is positive definite, the system is observable. Compared to the standard controllability and observability matrices, the Gramians offer a more quantitative method to studying the system properties.

If some similarity transform is applied to the system states, so that $A' = LAL^{-1}$, $B' = LB$, and $C' = CL^{-1}$, the Gramians will be modified as

$$\begin{aligned} P_C' &= LP_C L^T \qquad \text{and} \\ P_O' &= L^{-T} P_O L^{-1}. \end{aligned} \tag{10.20}$$

It is also possible to change the Gramians by selecting the state transformation matrix L appropriately. However, it turns out that the product of the Gramians

$$P_C' P_O' = L \cdot P_C P_O \cdot L^{-1} \tag{10.21}$$

is a similarity transtorm of the original Gramian product $P_C P_O$; regardless of the state transformation the eigenvalues of

$$P_C' P_O' \cdot M_C u_i = \lambda_i \cdot M_C u_i \tag{10.22}$$

remain invariant. It is possible to select L so that the new Gramian product $P'_C P'_O$ becomes diagonal by diagonalizing it using eigenvalue decomposition:

$$P'_C P'_O = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_n^2 \end{pmatrix}. \tag{10.23}$$

The parameters σ_i are very important system characterizing constants, and they are called *Hankel singular values,* being the singular values of the corresponding Hankel matrix. As was shown, Hankel singular values are invariant in a system under state-space representation. The Hankel singular values also determine, in a way, the "maximum gain" and the "minimum gain" of the system; the largest of the Hankel singular values is called the *Hankel norm* of the system.

The system realization where the state transformation has been selected in this way is said to be in the *balanced* form (the signals should also be appropriately normalized). In the balanced realization each of the state components is independent of the others; what is more, each state component is as well "visible" in the output as it is "excitable" from the input.

10.2.4 Eliminating states

The above discussion gives us concrete tools for state reduction: Drop those state components from x' that have the least importance in signal power transfer between input and output; these state components are exposed by the lowest Hankel singular values.

Noticing that $x = L^{-1}x'$, it can be recognized that the most of the input-output mapping is transferred through those states in x' that correspond to those rows of L^{-1} standing for the largest Hankel singular values. If dimension reduction is to be carried out using the balanced realization, the state mapping matrix θ^T is constructed from these rows of L^{-1}. Note that because $P_C P_O$ is not generally symmetric, the eigensystem is not orthogonal; that is why, the reduced system matrices cannot be constructed as $A' = \theta^T A\theta$, etc., but one first has to calculate the full matrices $A' = L^{-1}AL$, $B' = L^{-1}B$, and $C' = CL$, and only after that eliminate the rows and columns that correspond to the eliminated state elements in $z = x'$. The matrix D is not affected in the reduction process.

It needs to be recognized that there are some practical limitations what comes to balanced system truncation. First, the system has to be asymptotically stable (otherwise the Gramians do not remain bounded). Second, again, one fact needs to be kept in mind: Mathematical optimality does not always mean good design[3].

[3]For example, for physical reasons, we may know that the overall system gain should be unity; state truncation in the above way does not assure that this system property is maintained — see Exercises

10.3 State estimation

Another example of the surprises that one can attack using "multivariate thinking" is taken from the field of *state estimation:* Given the system structure and the measurement signals, the problem is to determine the system state. Now, it is assumed that the system model has the form of (9.4), also containing stochastic components. Further, assume that only the output $y(\kappa)$ can be measured, and, of course, $u(\kappa)$ is known. The goal is to find out $x(\kappa)$ using only these past system inputs and outputs:

$$\begin{cases} x(\kappa+1) = Ax(\kappa) + Bu(\kappa) + \epsilon(\kappa) \\ y(\kappa) = Cx(\kappa) + Du(\kappa) + e(\kappa). \end{cases} \tag{10.24}$$

The state estimators generally has the (recursive) form

$$\hat{x}(\kappa+1) = A\hat{x}(\kappa) + Bu(\kappa) + K(\kappa) \cdot (y(\kappa) - \hat{y}(\kappa)), \tag{10.25}$$

where

$$\hat{y}(\kappa) = C\hat{x}(\kappa) + Du(\kappa). \tag{10.26}$$

The expression $y(\kappa) - \hat{y}(\kappa)$ represents the error in the model output as compared to the real system output. The state estimate follows the assumed system model, but if there is error in the model output, the estimate is corrected appropriately. Our goal is to determine the matrix $K(\kappa)$ so that the actual state would be recovered as well as possible using the observed system behavior. It is reasonable to define this "goodness" in terms of the state estimation error

$$\tilde{x}(\kappa) = x(\kappa) - \hat{x}(\kappa). \tag{10.27}$$

The goal is now to minimize the covariance matrix $\mathrm{E}\{\tilde{x}(\kappa)\tilde{x}^T(\kappa)\}$. This is accomplished by the so called *Kalman filter.*

10.3.1 Kalman filter

The solution to the state estimation problem is based on induction: Assume that $P(\kappa)$ is the minimum error covariance having been found using the measurements that were available before the time instant κ. The matrix $K(\kappa)$ is now determined so that the covariance at the next time point also is minimal.

Subtract the state estimator, as given by (10.25), from the system state, as defined in (10.24):

$$\begin{aligned} \tilde{x}(\kappa+1) &= x(\kappa+1) - \hat{x}(\kappa+1) \\ &= (A - K(\kappa)C)\,\tilde{x}(\kappa) + \epsilon(\kappa) - K(\kappa)e(\kappa). \end{aligned} \tag{10.28}$$

Multiply both sides by their transposes and take the expectation values — on the left hand side, one has the next step estimation error covariance matrix to

be minimized:

$$\begin{aligned}
&P(\kappa+1)\\
&\quad = \mathrm{E}\{\tilde{x}(\kappa+1)\tilde{x}^T(\kappa+1)\}\\
&\quad = (A-K(\kappa)C)P(\kappa)(A-K(\kappa)C)^T\\
&\qquad + R_{xx} + K(\kappa)R_{xy}^T + R_{xy}K^T(\kappa) + K(\kappa)R_{yy}K^T(\kappa)\\
&\quad = AP(\kappa)A^T + R_{xx}\\
&\qquad - \left(AP(\kappa)C^T + R_{xy}\right)\left(CP(\kappa)C^T + R_{yy}\right)^{-1}\left(AP(\kappa)C^T + R_{xy}\right)^T\\
&\qquad + \Big(K(\kappa) - \left(AP(\kappa)C^T + R_{xy}\right)\left(CP(\kappa)C^T + R_{yy}\right)^{-1}\Big)\cdot\\
&\qquad\quad \left(CP(\kappa)C^T + R_{yy}\right)\cdot\\
&\qquad\qquad \Big(K(\kappa) - \left(AP(\kappa)C^T + R_{xy}\right)\left(CP(\kappa)C^T + R_{yy}\right)^{-1}\Big)^T.
\end{aligned}$$

The last part of the equation above (last three rows) is a quadratic form and the matrix in the middle $CP(\kappa)C^T + R_{yy}$ is positive semidefinite. This means that the minimum for the overall expression is reached if this last part is made zero, or if one selects

$$K(\kappa) = \left(AP(\kappa)C^T + R_{xy}\right)\left(CP(\kappa)C^T + R_{yy}\right)^{-1}. \tag{10.29}$$

In this case the minimum covariance becomes

$$\begin{aligned}
P(\kappa+1) &= AP(\kappa)A^T + R_{xx}\\
&\qquad - \left(AP(\kappa)C^T + R_{xy}\right)\left(CP(\kappa)C^T + R_{yy}\right)^{-1}\cdot\\
&\qquad\quad \left(AP(\kappa)C^T + R_{xy}\right)^T\\
&= AP(\kappa)A^T + R_{xx} - K(\kappa)\cdot\left(AP(\kappa)C^T + R_{xy}\right)^T.
\end{aligned} \tag{10.30}$$

Often in time-invariant environments a constant gain matrix is used instead:

$$\bar{K} = \left(A\bar{P}C^T + R_{xy}\right)\left(C\bar{P}C^T + R_{yy}\right)^{-1}, \tag{10.31}$$

where $\bar{P}$ is the positive semidefinite solution to the Riccati equation

$$\begin{aligned}
\bar{P} &= A\bar{P}A^T + R_{xx}\\
&\qquad - \left(A\bar{P}C^T + R_{xy}\right)\left(C\bar{P}C^T + R_{yy}\right)^{-1}\left(A\bar{P}C^T + R_{xy}\right)^T.
\end{aligned} \tag{10.32}$$

10.3.2 Optimality vs. reality

The above solution to the state estimation problem is also optimal. However, let us study what may happen in practice — assume that the system is one-dimensional, with only one input and two outputs as

$$\begin{cases}
x(\kappa+1) = ax(\kappa) + bu(\kappa) + \epsilon(\kappa)\\
\begin{pmatrix} y_1(\kappa)\\ y_2(\kappa)\end{pmatrix} = \begin{pmatrix} 1\\ 1\end{pmatrix}\cdot x(\kappa) + \begin{pmatrix} e_1(\kappa)\\ e_2(\kappa)\end{pmatrix},
\end{cases} \tag{10.33}$$

so that essentially the scalar state is measured two times. Intuitively, this should of course enhance the estimate, or, at least, it should not have any catastrophic effects. However, study the resulting steady-state gain matrix:

$$\bar{K} = \left(a\bar{p} \cdot \begin{pmatrix} 1 & 1 \end{pmatrix} + R_{xy}\right) \cdot \left(\bar{p} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} \mathrm{E}\{e_1^2(\kappa)\} & \mathrm{E}\{e_1(\kappa)e_2(\kappa)\} \\ \mathrm{E}\{e_1(\kappa)e_2(\kappa)\} & \mathrm{E}\{e_2^2(\kappa)\} \end{pmatrix}\right)^{-1}, \tag{10.34}$$

so that the properties of the estimator are essentially dictated by the invertibility of the matrix

$$\bar{p} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} \mathrm{E}\{e_1^2(\kappa)\} & \mathrm{E}\{e_1(\kappa)e_2(\kappa)\} \\ \mathrm{E}\{e_1(\kappa)e_2(\kappa)\} & \mathrm{E}\{e_2^2(\kappa)\} \end{pmatrix}. \tag{10.35}$$

Clearly, the first term is singular regardless of the numeric value of the scalar $\bar{p}$ — the whole sum becomes uninvertible, at least, if there holds $e_1(\kappa) = e_2(\kappa)$. If the same variable is measured, most probably the different measurements are correlated — if the measurements happen to be exactly identical, the whole estimator explodes, and even if they are not, the matrix may still become arbitrarily badly conditioned.

Consider some sensor fusion tasks, for example: The Kalman filter is often used for combining more or less reliable measurements, and often the number of measurements is very high — for example, take the weather models, where thousands of measurements are used to gain information about atmospheric phenomena. Blindly trusting the Kalman filter is dangerous: Even though it is optimal it may sometimes work against intuition[4].

10.3.3 Reducing the number of measurements

The above uninvertibility problem was caused again by the collinearity of the measurements. It is not a surprise that the multivariate analysis techniques turn out to offer valuable tools for attacking this kind of problems. So, assume that we want to reduce the output dimension so that redundancies are eliminated. We search for the directions where the measurements are most informative as determined by the symmetric, positive semidefinite matrix

$$C\bar{P}C^T + R_{yy}. \tag{10.36}$$

What is "informative" is again a matter of taste; if the PCA type approach is chosen, the task is to find the eigenvectors corresponding to the largest eigenvalues in

$$\left(C\bar{P}C^T + R_{yy}\right) \cdot \theta_i = \lambda_i \cdot \theta_i. \tag{10.37}$$

Assume that the dimension is reduced by, say, the PCA technique. The reduced basis is assumed to be θ and the corresponding eigenvalues are on the diagonal of Λ; then one can write

$$C\bar{P}C^T + R_{yy} \approx \theta \cdot \Lambda \cdot \theta^T. \tag{10.38}$$

[4]In practice, there is no exact information about the noise properties, and to avoid problems, the covariance matrices are usually assumed diagonal ... but the optimality of the estimator is of course ruined when the system model is incorrect!

Now Λ is low-dimensional and well-conditioned, so that its inverse is easily calculated; approximately there holds

$$\left(C\bar{P}C^T + R_{yy}\right)^{-1} \approx \theta \cdot \Lambda^{-1} \cdot \theta^T. \tag{10.39}$$

Substituting this in (10.31) gives

$$\bar{K} \approx \left(A\bar{P}C^T + R_{xy}\right) \cdot \theta\Lambda^{-1}\theta^T \left(A\bar{P}C^T\theta + R_{xy}\theta\right) \Lambda^{-1}\theta^T, \tag{10.40}$$

so that the estimator becomes

$$\begin{aligned}&\hat{x}(\kappa+1) = \\ &\quad A\hat{x}(\kappa) + Bu(\kappa) + \left(A\bar{P}C^T\theta + R_{xy}\theta\right) \Lambda^{-1}\theta^T \cdot \left(y(\kappa) - C\hat{x}(\kappa) - Du(\kappa)\right).\end{aligned}$$

This formulation efficiently helps to avoid anomalies caused by the measurement redundancy.

10.4 SISO identification

Finally, yet another systems engineering application field is studied where the multivariate problem setting becomes relevant. We will study the prediction error methods for black-box identification (see [29], [26]). The traditional approaches that are still the mainstream technology (for example, see the `System Identification Toolbox` for `Matlab`) suffer from the problems that have been demonstrated in previous chapters, and analogous solutions to the problems can be proposed. Note that for practical parameter estimation purposes in dynamic systems, subspace identification (as explained in Chapter 9) is recommended.

10.4.1 Black-box model

The behavior of a linear, strictly proper, d'th order discrete time system can be expressed as a difference equation

$$\begin{aligned}y(\kappa) = a_1 y(\kappa-1) + \cdots + a_d y(\kappa-d) \\ + b_1 u(\kappa-1) + \cdots + b_d u(\kappa-d),\end{aligned} \tag{10.41}$$

where $u(\kappa)$ denotes the (centered) scalar process input and $y(\kappa)$ the scalar output at time instant κ. Using vector formulation, this can be written (following the earlier notations) as

$$y(\kappa) = x^T(\kappa) \cdot f, \tag{10.42}$$

where

$$x(\kappa) = \begin{pmatrix} y(\kappa-1) \\ \vdots \\ y(\kappa-d) \\ u(\kappa-1) \\ \vdots \\ u(\kappa-d) \end{pmatrix} \quad \text{and} \quad f = \begin{pmatrix} a_1 \\ \vdots \\ a_d \\ b_1 \\ \vdots \\ b_d \end{pmatrix}. \tag{10.43}$$

This means that the dynamic nature of the process has been transformed into a representation where virtually static time series samples are used; the dynamic complexity has been changed to dimensional complexity.

The structure of the linear dynamic system is assumed to be extremely simple, consisting of one input and one output signals; further, it is assumed that the dynamic dimension of the system is exactly known. If it is still assumed that the signals are persistently exciting, and no unmodeled noise is present, identifying the parameters of the model should be a trivial task. This is what standard theory says; however, in practice, problems often emerge. These problems can again be studied in the framework of statistical data analysis.

10.4.2 Recursive least-squares algorithm

The parameter vector f can be solved off-line, as a batch for some set of data; in this way, the methods presented in earlier chapters can directly be utilized (in practice, it seems to be customary to stick to the basic MLR or its derivations). However, in many cases measurements are obtained one at a time, and it is reasonable to rearrange the calculations so that the computational load would be minimized. To derive the *on-line recursive identification algorithm,* define the exponentially weighted cost criterion as

$$J(k) = \sum_{\kappa=0}^{k} \lambda^{k-\kappa} \cdot e^2(\kappa), \tag{10.44}$$

where the prediction error is defined as

$$e(\kappa) = y(\kappa) - x^T(\kappa) f. \tag{10.45}$$

The exponential weighting emphasizes the newest measurements, that is, if the *forgetting factor* λ has value less than one, old measurements are gradually forgotten. Note that the so called ARX system structure is chosen, again essentially assuming that the error is summed only to the output; otherwise the noise becomes *colored* and algorithms give biased estimates. Minimizing the criterion can be carried out as follows:

$$\begin{aligned} \tfrac{d\,J(k)}{d\,f} &= -2 \cdot \textstyle\sum_{\kappa=0}^{k} \lambda^{k-\kappa} \cdot x(\kappa) \cdot \big(y(\kappa) - x^T(\kappa) f\big) \\ &= -2 \cdot \textstyle\sum_{\kappa=0}^{k} \lambda^{k-\kappa} x(\kappa) \cdot y(\kappa) + 2 \cdot \sum_{\kappa=0}^{k} \lambda^{k-\kappa} x(\kappa) x^T(\kappa) \cdot f \\ &= \mathbf{0}, \end{aligned}$$

or

$$\sum_{\kappa=0}^{k} \lambda^{k-\kappa} x(\kappa) x^T(\kappa) \cdot f = \sum_{\kappa=0}^{k} \lambda^{k-\kappa} x(\kappa) \cdot y(\kappa), \tag{10.46}$$

so that the parameter estimate can be solved as

$$\hat{f} = \left(\sum_{\kappa=0}^{k} \lambda^{k-\kappa} x(\kappa) x^T(\kappa) \right)^{-1} \cdot \sum_{\kappa=0}^{k} \lambda^{k-\kappa} x(\kappa) \cdot y(\kappa). \tag{10.47}$$

However, this is not yet in the recursive form, so that the new parameter estimate $\hat{f}(k)$ would be received from the old estimate $\hat{f}(k-1)$ by updating it using some fresh information. To reach such a formulation, define

$$\begin{aligned} R_{xx}(k) &= \sum_{\kappa=0}^{k} \lambda^{k-\kappa} x(\kappa) x^T(\kappa) \\ &= x(k)x^T(k) + \lambda \cdot \sum_{\kappa=0}^{k-1} \lambda^{k-\kappa} x(\kappa) x^T(\kappa) \\ &= x(k)x^T(k) + \lambda \cdot R_{xx}(k-1) \end{aligned} \tag{10.48}$$

and

$$\begin{aligned} R_{xy}(k) &= \sum_{\kappa=0}^{k} \lambda^{k-\kappa} x(\kappa) y(\kappa) \\ &= x(k)y(k) + \lambda \cdot \sum_{\kappa=0}^{k-1} \lambda^{k-\kappa} x(\kappa) y(\kappa) \\ &= x(k)y(k) + \lambda \cdot R_{xy}(k-1). \end{aligned} \tag{10.49}$$

Formula (10.46) can be expressed using these matrices as $R_{xx}(k) \cdot \hat{f}(k) = R_{xy}(k)$, so that the new parameter estimate can be written as

$$\begin{aligned} \hat{f}(k) &= R_{xx}^{-1}(k) \cdot R_{xy}(k) \\ &= R_{xx}^{-1}(k) \cdot (x(k)y(k) + \lambda \cdot R_{xy}(k-1)) \\ &= R_{xx}^{-1}(k) \cdot \Big(x(k)y(k) + \lambda \cdot R_{xx}(k-1) \cdot \hat{f}(k-1) \Big) \\ &= R_{xx}^{-1}(k) \cdot \Big(x(k)y(k) + \big(R_{xx}(k) - x(k)x^T(k) \big) \cdot \hat{f}(k-1) \Big) \\ &= \hat{f}(k-1) + R_{xx}^{-1}(k) \cdot \Big(x(k)y(k) - x(k)x^T(k) \cdot \hat{f}(k-1) \Big) \\ &= \hat{f}(k-1) + R_{xx}^{-1}(k) \cdot x(k) \cdot \Big(y(k) - x^T(k) \cdot \hat{f}(k-1) \Big). \end{aligned}$$

Rewriting this and collecting the results together (and defining $R = R_{xx}$), the final Gauss-Newton type identification algorithm becomes

$$\begin{aligned} \hat{f}(k) &= \hat{f}(k-1) + R^{-1}(k) x(k) \cdot \Big(y(k) - x^T(k) \hat{f}(k-1) \Big) \\ R(k) &= \lambda R(k-1) + x(k)x^T(k). \end{aligned} \tag{10.50}$$

The *matrix inversion lemma* could be applied here to make the algorithm more efficient in practice; however, in this context overall comprehensibility of the algorithm is preferred. On the first line, the parameter estimate vector is updated; the size of the update step is determined by the prediction error, whereas the update direction is determined by the matrix $R(k)$. What is this matrix, then — this can be seen if one studies the expectation values:

$$E\{R(k)\} = \lambda \cdot E\{R(k-1)\} + E\{x(k)x^T(k)\}, \tag{10.51}$$

where $E\{R(k)\} = E\{R(k-1)\} = E\{R\}$, so that

$$E\{R\} = \frac{1}{1-\lambda} \cdot E\left\{ xx^T \right\}. \tag{10.52}$$

This means that R is the (scaled) data covariance matrix estimate — sounds familiar ...!

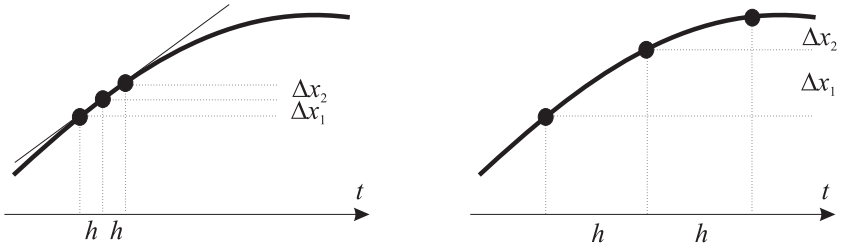

Figure 10.1: A figure illustrating the fact that short sampling intervals (shown on the left) make the successive samples mutually dependent: They have almost equal values, or, at least, assuming smooth signal behavior, they are almost on the same line, thus being collinear

10.4.3 Structure of dynamic data

The data $x(\kappa)$ was constructed from successive signal measurements. However, there are peculiar dependencies between the delayed signals that are not taken into account by the standard SISO identification algorithms. The problems are (as in the MLR case) concentrated on the invertibility of the data covariance matrix R.

The successive samples are most probably highly correlated because they should represent continuous dynamic evolution. This redundancy between the signal samples is the main reason for the structural identifiability problems — the data is collinear, and this linear dependency between variables becomes more and more dominating when the sampling interval is made smaller (see Fig. 10.1).

The numerical problems inherent in the data are emphasized by the recursive "forgetting" of the algorithms: Older information gets ignored as time evolves, and it may be that the numerical properties of the data covariance matrix are gradually ruined, the identification process becoming badly behaving.

Because of the relevance of the robustness issues in practical approaches, various more or less heuristic approaches have been proposed, including different kinds of *constant trace, regularization,* or *variable forgetting* algorithms (see [19] and [21], for example). A close relative of Ridge Regression (as written in recursive form) is the so called *Levenberg-Marquardt* identification algorithm that keeps R invertible by adding a minor positive definite factor to it during each step:

$$\begin{aligned} \hat{f}(k) &= \hat{f}(k-1) + R^{-1}(k)x(k) \cdot \Big(y(k) - x^T(k)\hat{f}(k-1)\Big) \\ R(k) &= \lambda R(k-1) + x(k)x^T(k) + q \cdot I_{2d}. \end{aligned} \tag{10.53}$$

It can be shown that as q varies from zero to higher values, the continuum from Gauss-Newton and simple gradient method is spanned.

Another family of identification methods (or, actually, data preprocessing methods) is found when the system *parameterizations* are studied. The dynamic nature of a system can be captured in an infinite number of ways; even though the above time series approach is rather natural, it is only one alternative — and not a very good choice, as it has turned out. As an example, study Fig. 10.1:

Using the traditional shift operator q formalism (or, equivalently, using delay operators q^{-1}) all systems finally become integrators as the sampling period h goes towards zero! A very simple alternative to the standard parameterization is the δ parameter formalism [12]; the idea is that rather than taking the measurements themselves as a basis, the *differentiated* signals are used: Differences between successive samples are analyzed rather than the original signal values. It has been shown that this parameterization enhances the numerical properties of many algorithms (not only identification algorithms). More sophisticated parameterizations are studied, for example, in [9] and in [20][5].

10.4.4 Further analysis: System identifiability*

The properties of the data covariance matrix are, of course, determined by the data properties themselves — but not exclusively. As was seen above, the system dynamics dictates what is the relation between successive measurements, and this is reflected also in the data covariance. Now study the *structural identifiability properties* of a dynamic system. This analysis was carried out originally in [22].

Differentiating (10.42) with respect to f, one has

$$\frac{d\,y}{d\,f}(\kappa) = x(\kappa), \tag{10.54}$$

so that the data covariance matrix can be written as

$$\mathrm{E}\left\{x(\kappa)x^T(\kappa)\right\} = \mathrm{E}\left\{\left(\frac{d\,y}{d\,f}(\kappa)\right)\left(\frac{d\,y}{d\,f}(\kappa)\right)^T\right\}. \tag{10.55}$$

What are these signal derivatives? One can differentiate the response $y(\kappa)$ in (10.41) with respect to all the parameters, so that a set of difference equations is found:

$$\left\{\begin{array}{rcl} \dfrac{\partial y}{\partial a_i}(\kappa) & = & \displaystyle\sum_{j=1}^{d} a_j q^{-j} \cdot \frac{\partial y}{\partial a_i}(\kappa) + q^{-i} \cdot y(\kappa) \\ \dfrac{\partial y}{\partial b_i}(\kappa) & = & \displaystyle\sum_{j=1}^{d} a_j q^{-j} \cdot \frac{\partial y}{\partial b_i}(\kappa) + q^{-i} \cdot u(\kappa). \end{array}\right. \tag{10.56}$$

Neglecting the initial conditions, these $2d$ difference equations of order d can be written in a $2d$-dimensional state space form

$$x'(\kappa+1) = A'x'(\kappa) + B'u(\kappa) \tag{10.57}$$

[5] In the multivariate framework, one straightforward approach to enhancing the matrix invertibility properties would be to reduce the dimension of R, just as has been done so many times this far. However, now it cannot be assumed that the properties of data remain invariant — the original reason to use recursive algorithms was to be able to react to changing system properties — and the matrix R does not remain constant: The eigenvalue decomposition should be computed, in principle, during each step, and the computational burden would become excessive

defined by the matrices

$$A' = \left(\begin{array}{cccc|cccc} a_1 & a_2 & \cdots & a_d & & & & \\ 1 & & & & & & 0 & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ \hline b_1 & b_2 & \cdots & b_d & a_1 & a_2 & \cdots & a_d \\ & & & & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \end{array}\right) \quad \text{and} \quad B' = \left(\begin{array}{c} 1 \\ 0 \\ \vdots \\ 0 \\ \hline 0 \\ 0 \\ \vdots \\ 0 \end{array}\right).$$

This state-space system will here be called the "sensitivity system" corresponding to the model (10.41). The control signal in this system is the original input $u(k)$, but the state vector is

$$x'(\kappa) = \left(\begin{array}{c} \frac{\partial y}{\partial b_1}(\kappa) \\ \vdots \\ \frac{\partial y}{\partial b_d}(\kappa) \\ \hline \frac{\partial y}{\partial a_1}(\kappa) \\ \vdots \\ \frac{\partial y}{\partial a_d}(\kappa) \end{array}\right) = \frac{d\,y}{d\,f}(\kappa). \tag{10.58}$$

The motivation for these definitions is that the covariance structures for the data vector x and the state vector x' must be identical; and the behavior of the dynamic system state can be easily analyzed. The data covariance matrix (10.55) can also be written as

$$\begin{aligned} &\mathrm{E}\left\{\left(\frac{dy}{df}(\kappa)\right)\left(\frac{dy}{df}(\kappa)\right)^T\right\} \\ &\quad = \mathrm{E}\Big\{(B'u(\kappa-1) + A'B'u(\kappa-2) + \cdots)\cdot \\ &\qquad\qquad (B'u(\kappa-1) + A'B'u(\kappa-2) + \cdots)^T\Big\} \\ &\quad = r_u(0)\cdot\left(B'B'^T + A'B'B'^TA'^T + A'^2B'B'^TA'^{2T} + \cdots\right) \\ &\qquad + r_u(1)\cdot\left(A'B'B'^T + B'B'^TA'^T + \cdots\right) \\ &\qquad\quad + r_u(2)\cdot\left(A'^2B'B'^T + B'B'^TA'^{2T} + \cdots\right) \\ &\qquad\qquad + \cdots \\ &\quad = r_u(0)\cdot M'_C \\ &\qquad + r_u(1)\cdot\left(A'M'_C + M'_CA'^T\right) \\ &\qquad\quad + r_u(2)\cdot\left(A'^2M'_C + M'_CA'^{2T}\right) \\ &\qquad\qquad + \cdots \end{aligned} \tag{10.59}$$

It turns out that the matrix M'_C equals the Controllability Gramian for the sensitivity system; additionally, the input signal autocorrelation function values

are involved here:

$$\begin{aligned}
r_u(0) &= \mathrm{E}\{u^2(\kappa-1)\} = \mathrm{E}\{u^2(\kappa-2)\} = \cdots \\
r_u(1) &= \mathrm{E}\{u(\kappa-1)u(\kappa-2)\} = \mathrm{E}\{u(\kappa-2)u(\kappa-3)\} = \cdots \\
&= \mathrm{E}\{u(\kappa-2)u(\kappa-1)\} = \mathrm{E}\{u(\kappa-3)u(\kappa-2)\} = \cdots \\
r_u(2) &= \mathrm{E}\{u(\kappa-1)u(\kappa-3)\} = \mathrm{E}\{u(\kappa-2)u(\kappa-4)\} = \cdots \\
&= \mathrm{E}\{u(\kappa-3)u(\kappa-1)\} = \mathrm{E}\{u(\kappa-4)u(\kappa-2)\} = \cdots \\
&\vdots
\end{aligned}$$

This gives us a possibility of estimating the "efficiency" of the input signal what comes to its capability of helping in the parameter identification. On the other hand, optimization of the input can also be carried out: One can determine the autocorrelation function values so that (10.59) becomes easily invertible, and after that construct realizations of such a signal applying spectral factorization. However, note that there are physical constraints what comes to the autocorrelation function — for example, $r_u(0)$ must always be the largest of all $r_u(i)$ for the signal to be realizable.

More anqalysis is needed here: Formal identifiability differs from actual identifiability. To avoid the problems that are inherent to traditional model structures, new model structures need to be introduced. Such efforts are illustrated in the last chapter.

Computer exercises

1. Study the power of the dimension reduction technique; run the following command sequence various times. Select (interactively) the number of states in different ways — what kind of non-physicalities are seen in the approximations?

```
d = 10;
A = randn(d,d); A = A/(1.1*norm(A));
B = randn(d,1);
C = randn(1,d);
regrBal(A,B,C);
```

   Construct a discrete-time system for implementing a *pure delay* of d time steps, and try to reduce the model:

```
d = 10;
A = zeros(d,d); A(2:d,1:d-1) = eye(d-1);
B = zeros(d,1); B(1,1) = 1;
C = zeros(1,d); C(1,d) = 1;
[Ared,Bred,Cred] = regrBal(A,B,C);
```

   What is the problem? In *this* special case, that specific problem can be circumvented by the following modifications without altering the input/output behavior. However, what can you say about all possible "optimal" model reductions now?

```
A = 0.9*A;
B = B/0.9^d;
```

2. Study the robustness of recursive identification: Check how much the behavior of the parameter estimates changes as the underlying system structure varies by running the following commands various times. Also try the effects of the forgetting factor λ.

```
lambda = 0.99;
u = randn(100,1);
[u,y] = dataDyn(3,u,1);
regrIdent(u,y,3,lambda);
```