Numerical Linear Algebra for Signals Systems and Control

Paul M. Van Dooren
University of Louvain, B-1348 Louvain-la-Neuve, Belgium

Draft notes prepared for the
Graduate School in Systems and Control

Spring 2003

Contents

Chapter 1

SCOPE AND INTRODUCTION

1.1 Scope of the Course

This course looks at numerical issues of algorithms for signals, systems and control. In doing that a clear choice is made to focus on *numerical linear algebra* techniques for *linear time-invariant, finite dimensional systems*. At first hand, this may look as narrowing down the subject quite a bit, but there are simple reasons for this.

The fact that we deal only with numerical methods for **linear time-invariant, finite dimensional systems** is not really as restrictive as it seems. One encounters in fact very much the same numerical problems when relaxing these constraints:

- problems in *time varying systems* are often based on the time-invariant counterpart because of the recursive use of time-invariant techniques or because one uses a reformulation into a time-invariant problem. There is e.g., hardly any difference between Kalman filtering for time-varying and time-invariant systems since they both lead to recursive least squares problems. Another typical example is that of periodic systems which can be rewritten as a time-invariant problem by considering the "lifted" system

- *nonlinear systems* are typically approximated by a sequence of linearized models for which standard techniques are then applied. Also the approximation is sometimes using techniques borrowed from linear systems theory

- *infinite dimensional systems* typically arise from partial differential equations. One then uses discretizations such as finite elements methods to represent the underlying operator by a (typically large and sparse) matrix. For both the approximation and the subsequent treatment of the finite dimensional problem one then uses matrix techniques.

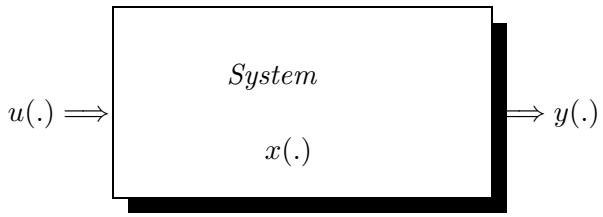
So in other words, this course deals with signals that can be modeled by sets of differential equations

$$P(\frac{d}{dt})y(t) = Q(\frac{d}{dt})u(t),$$

or difference equations

$$P(\mathcal{E})y(k) = Q(\mathcal{E})u(k),$$

where $P(.)$ and $Q(.)$ are polynomial matrices of appropriate dimensions and where $u(.)$ is an m-vector of controls or inputs, and $y(.)$ is an p-vector of outputs, and both are functions of time. When the time variable is the continuous time t, the operator is the differential operator $\frac{d}{dt}$, when the time variable is the discrete variable k, the operator is the advance operator $\mathcal{E}$ (often s and z are used instead of the above two). These models thus describe the following input-output behavior



Other "standard" linear models use the n-vector of states $x(.)$, leading to so-called state-space models:

$$\frac{d}{dt}x(t) = Ax(t) + Bu(t) \tag{1.1}$$
$$y(t) = Cx(t) + Du(t), \tag{1.2}$$

and

$$x_{k+1} = Ax_k + Bu_k \tag{1.3}$$
$$y_k = Cx_k + Du_k, \tag{1.4}$$

where the constant matrices A, B, C and D are of appropriate dimensions. Of course, other models are also possible, such as transfer functions and generalized state-space models, but we will focus on the above two.

The second restriction in the title of the course is the fact that we focus on **numerical linear algebra** techniques. The reasons for this are the following:

- once we decide to focus on linear time-invariant, finite dimensional systems, it is clear that we are dealing with numerical linear algebra problems since the models consist of polynomial and constant matrices
- there has been a lot of interaction between signals, systems and control and numerical linear algebra over the last 25 years and significant advances were made as a result of this
- due to the new developments in this interdisciplinary area, software tools like libraries (SLICOT) and interactive packages (MATLAB, Matrix$_x$, SCILAB) have been developed that have made these ideas ready to use.

Although we do not cover here methods and problems of optimization, approximation, ordinary differential equations, two point boundary value problems, and so on, we feel that this is not really a restriction. Indeed, numerical linear algebra methods are again at the heart of each of these other areas.

In view of all this, we believe the material in this course is a kind of greatest common denominator of what anybody interested in numerical methods for signals, systems and control, ought to know. As the reader will see, that already covers quite a lot of material. That is also the reason why this course avoids specialized topics as e.g., implementation aspects on specialized architectures, or software aspects. Since the course is meant to educate the reader rather than to provide him with a basic set of routines, we have chosen to base the course as much as possible on MATLAB for illustrating the numerical issues being discussed.

1.2 About Errors

The systems, control, and estimation literature is replete with *ad hoc* algorithms to solve the computational problems which arise in the various methodologies. Many of these algorithms work quite well on some problems (e.g., "small order" matrices) but encounter numerical difficulties, often severe, when "pushed" (e.g., on larger order matrices). The reason for this is that little or no attention has been paid to how the algorithms will perform in "finite arithmetic," i.e., on a finite-word-length digital computer.

A simple example due to Moler and Van Loan [95] will illustrate a typical pitfall. Suppose it is desired to compute the matrix e^A in single precision arithmetic on a computer which gives 6 decimal places of precision in the fraction part of floating-point numbers. Consider the case

$$A = \begin{bmatrix} -49 & 24 \\ -64 & 31 \end{bmatrix}$$

and suppose the computation is attempted using the Taylor series formula

$$e^A = \sum_{k=0}^{+\infty} \frac{1}{k!} A^k. \tag{1.5}$$

This is easily coded and it is determined that the first 60 terms in the series suffice for the computation, in the sense that terms for $k \geqslant 60$ are of the order of 10^{-7} and no longer add anything significant to the sum. The resulting answer is

$$\begin{bmatrix} -22.2588 & -1.43277 \\ -61.4993 & -3.47428 \end{bmatrix}.$$

Unfortunately, the true answer is (correctly rounded)

$$\begin{bmatrix} -0.735759 & 0.551819 \\ -1.47152 & 1.10364 \end{bmatrix}$$

and one sees a rather alarming disparity. What happened here was that the intermediate terms in the series got very large before the factorial began to dominate. In fact, the 17th and 18th terms, for example, are of the order of 10^7 but of opposite signs so that the less significant parts of these numbers—while significant for the final answer—are "lost" because of the finiteness of the arithmetic.

Now for this particular example various fixes and remedies are available. But in more realistic examples one seldom has the luxury of having the "true answer" available so that it is not always

easy simply to inspect or test an answer such as the one obtained above and determine it to be in error. Mathematical analysis (truncation of the series, in the example above) alone is simply not sufficient when a problem is analyzed or solved in finite arithmetic (truncation of the arithmetic). Clearly, a great deal of care must be taken.

When one thinks of the fact that the "response" of the system (1.3) is:

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-s)}Bu(s)ds$$

then it is clear that even more problems must be expected when approximating the exponential with truncated Taylor expansions in this expression.

The finiteness inherent in representing real or complex numbers as floating-point numbers on a digital computer manifests itself in two important ways: floating-point numbers have only finite precision and finite range. In fact, it is the degree of attention paid to these two considerations that distinguishes many reliable algorithms from more unreliable counterparts. Wilkinson [147] still provides the definitive introduction to the vagaries of floating-point computation while [90] and the references therein may be consulted to bring the interested reader up to date on roundoff analysis.

The development in systems, control, and estimation theory, of stable, efficient, and reliable algorithms which respect the constraints of finite arithmetic began in the 1970's and is ongoing. Much of the research in numerical analysis has been directly applicable, but there are many computational issues (e.g., the presence of hard or structural zeros) where numerical analysis does not provide a ready answer or guide. A symbiotic relationship has developed, particularly between numerical linear algebra and linear system and control theory, which is sure to provide a continuing source of challenging research areas.

The abundance of numerically fragile algorithms is partly explained by the following observation which will be emphasized by calling it a "folk theorem":

> If an algorithm is amenable to "easy" hand calculation, it is probably a poor method if implemented in the finite floating-point arithmetic of a digital computer.

For example, when confronted with finding the eigenvalues of a 2×2 matrix most people would find the characteristic polynomial and solve the resulting quadratic equation. But when extrapolated as a general method for computing eigenvalues and implemented on a digital computer, this turns out to be a very poor procedure indeed for a variety of reasons (such as roundoff and overflow/underflow). Of course the preferred method now would generally be the double Francis QR algorithm (see [35, 36], [120], and [148] for the messy details) but few would attempt that by hand—even for very small order problems.

In fact, it turns out that many algorithms which are now considered fairly reliable in the context of finite arithmetic are not amenable to hand calculations (e.g., various classes of orthogonal similarities). This is sort of a converse to the folk theorem. Particularly in linear system and control theory, we have been too easily seduced by the ready availability of closed-form solutions and numerically naive methods to implement those solutions. For example, in solving the initial value problem

$$\dot{x}(t) = Ax(t); \qquad x(0) = x_0 \tag{1.6}$$

it is not at all clear that one should explicitly want to compute the intermediate quantity e^{tA}. Rather, it is the vector $e^{tA}x_0$ that is desired, a quantity that may be computed more reasonably

by treating (1.6) as a system of (possibly stiff) differential equations and using, say, an implicit method for numerical integration of the differential equation. But such techniques are definitely not attractive for hand computation.

Awareness of such numerical issues in the mathematics and engineering community has increased significantly in the last fifteen years or so. In fact, some of the background material that is well known to numerical analysts, has already filtered down to undergraduate and graduate curricula in these disciplines. A number of introductory textbooks currently available (e.g., [63, 21, 33, 62, 111, 112]) also reflect a strong software component. The effect of this awareness and education has been particularly noticeable in the area of system and control theory, especially in linear system theory. A number of numerical analysts were attracted by the wealth of interesting numerical linear algebra problems in linear system theory. At the same time, several researchers in the area of linear system theory turned their attention to various methods and concepts from numerical linear algebra and attempted to modify and use them in developing reliable algorithms and software for specific problems in linear system theory. This cross-fertilization has been greatly enhanced by the widespread use of software packages and by recent developments in numerical linear algebra. This process has already begun to have a significant impact on the future directions and development of system and control theory, and applications, as is evident from the growth of computer-aided control system design as an intrinsic tool. Algorithms implemented as mathematical software are a critical "inner" component of such a system.

Before proceeding further we shall list here some notation to be used in the sequel:

$\mathbb{F}^{n\times m}$	the set of all $n \times m$ matrices with coefficients in the field $\mathbb{F}$ ($\mathbb{F}$ will generally be $\mathbb{R}$ or $\mathbb{C}$)
A^T	the transpose of $A \in \mathbb{R}^{n\times m}$
A^H	the complex-conjugate transpose of $A \in \mathbb{C}^{n\times m}$
A^+	the Moore-Penrose pseudoinverse of A
diag $(a_1, \cdots, a_n)$	the diagonal matrix $\begin{bmatrix} a_1 & & 0 \\ & \ddots & \\ 0 & & a_n \end{bmatrix}$
$\Lambda(A)$	the set of eigenvalues $\lambda_1, \cdots, \lambda_n$ (not necessarily distinct) of $A \in \mathbb{F}^{n\times n}$
$\lambda_i(A)$	the ith eigenvalue of A
$\Sigma(A)$	the set of singular values $\sigma_1, \cdots, \sigma_m$ (not necessarily distinct) of $A \in \mathbb{F}^{n\times m}$
$\sigma_i(A)$	the ith singular value of A.

Finally, let us define a particular number to which we shall make frequent reference in the sequel. The *machine epsilon* or *relative machine precision* can be defined, roughly speaking, as the smallest positive number ϵ which, when added to 1 on our computing machine, gives a number greater than 1. In other words, any machine representable number δ less than ϵ gets " rounded off" when (floating-point) added to 1 to give exactly 1 again as the rounded sum. The number ϵ, of course, varies depending on the kind of computer being used and the precision with which the computations are being done (single precision, double precision, etc.). But the fact that there exists such a positive number ϵ is entirely a consequence of finite word length.

1.3 Numerical Background

In this section we give a very brief discussion of two concepts of fundamental importance in numerical analysis: *numerical stability* and *conditioning*. While this material is standard in textbooks such as [46, 54, 122, 128] it is presented here both for completeness and because the two concepts are frequently confused in the systems, control, and estimation literature.

Suppose we have some mathematically defined problem represented by f which acts on data x belonging to some set of data $\mathcal{D}$, to produce a solution $y = f(x)$ in a solution set $\mathcal{S}$. These notions are kept deliberately vague for expository purposes. Given $x \in \mathcal{D}$ we desire to compute $f(x)$. Suppose x^* is some approximation to x. If $f(x^*)$ is "near" $f(x)$ the problem is said to be well-conditioned. If $f(x^*)$ may potentially differ greatly from $f(x)$ even when x^* is near x, the problem is said to be ill-conditioned. The concept of "near" can be made precise by introducing norms in the appropriate spaces.

A norm (usually denoted by $\|.\|$) is a real scalar function defined on a vector space $\mathcal{V}$, with the following properties

- $\|x\| \geqslant 0,\ \forall x \in \mathcal{V}$
- $\|x\| = 0 \Longleftrightarrow x = 0$
- $\|\alpha x\| = |\alpha|\|x\|,\ \forall x \in \mathcal{V}, \alpha \in \mathbb{C}$
- $\|x + y\| \leqslant \|x\| + \|y\|,\ \forall x, y \in \mathcal{V}$

For vectors in $\mathbb{C}^n$ one typically uses the 2-norm:

$$\|x\|_2 \doteq \sqrt{\sum_{i=1}^{n} |x_i|^2}$$

and for matrices in $\mathbb{C}^{m\times n}$ one typically uses the induced 2-norm:

$$\|\mathbf{A}\|_2 \doteq \sup_{\|x\|\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sup_{\|x\|_2=1} \|Ax\|_2$$

or the Frobenius norm:

$$\|\mathbf{A}\|_F \doteq \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}$$

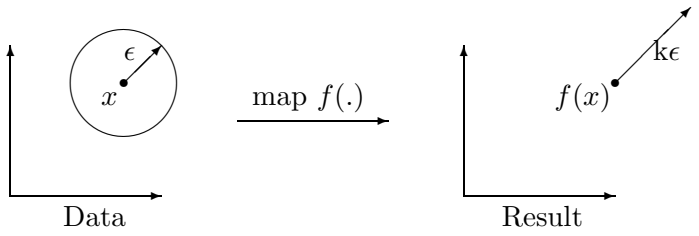
These norms have the following useful properties:

- $\|Ux\|_2 = \|x\|_2$ if $U^H U = I_n$
- $\|UAV^H\|_{2,F} = \|A\|_{2,F}$ if $U^H U = I_m$, $V^H V = I_n$
- $\|AB\|_{2,F} \leqslant \|A\|_{2,F} \cdot \|B\|_{2,F}$

which are used a lot in the sequel. We can then define the condition of the problem f with respect to these norms as

$$\kappa(f,x) = \lim_{\delta \to 0} \sup_{d_2(x,x^*)=\delta} \left[\frac{d_1(f(x), f(x^*))}{\delta} \right]$$

where d_i (.,.) are distance functions in the appropriate spaces. This can be "visualized" by the following picture



We see that an ϵ ball around x here is mapped by the function $f(.)$ to a region around $f(x)$ of smallest radius $k\epsilon$. For infinitesimally small ϵ, the ratio k of both radii tends to the condition number $\kappa(f,x)$. When $f(.)$ has an expansion around x one can write

$$f(x+\delta x) = f(x) + \nabla_x f.\delta x + O(\|\delta x\|^2)$$

where $\nabla_x f$ is the Fréchet derivative of f at x. The condition number $\kappa(f,x)$ is then also the norm $\|\nabla_x f\|$ of the Fréchet derivative. When $\kappa(f,x)$ is infinite, the problem of determining *f(x)* from x is *ill-posed* – as opposed to *well-posed* – and the Fréchet derivative is thus not bounded. When $\kappa(f,x)$ is finite and *relatively large* (or *relatively small*), the problem is said to be *ill-conditioned* (or *well-conditioned*). Further details can be found in [110].

We now give two simple examples to illustrate these concepts.

Example 1.1.

Consider the $n \times n$ matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & & & \cdot & \cdot & \cdot & 0 \\ \cdot & & & & \cdot & \cdot & 1 \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \end{bmatrix}$$

with n eigenvalues at 0. The characteristic polynomial of this matrix is

$$\chi(\lambda) \doteq det.(\lambda I - A) = \lambda^n.$$

Now consider the perturbed matrix A_δ with a small perturbation of the data (the n^2 elements of A) consisting of adding δ to the first element in the last (nth) row of A. This perturbed matrix then has the following characteristic polynomial:

$$\chi_\delta(\lambda) \doteq det.(\lambda I - A_\delta) = \lambda^n - \delta$$

which has n distinct roots $\lambda_1, \cdots, \lambda_n$ with $\lambda_k = \delta^{1/n} \exp(2k\pi j/n)$. Denoting the vector of eigenvalues by Λ we have thus

$$\|A - A_\delta\|_2 = \delta, \;\; \|\Lambda - \Lambda_\delta\|_2 = \sqrt{n}\delta^{1/n}.$$

Related examples can be found in [148], where it is also shown that this is the worst possible perturbations that can be imposed on A. For this case we thus have:

$$\kappa_\Lambda(A) = \lim_{\delta \to 0} \frac{\|A - A_\delta\|}{\|\Lambda - \Lambda_\delta\|} = \lim_{\delta \to 0} \frac{\delta}{\sqrt{n}\delta^{1/n}} = \infty$$

and the problem is thus ill-conditioned.

Example 1.2.

Take now a symmetric diagonal matrix A=diag $(\lambda_1, \cdots, \lambda_n)$. For such matrices it is known that the worst place to put a δ perturbation in order to move an eigenvalue as much as possible is actually on the diagonal. Without loss of generality, let us perturb the first diagonal entry by δ, then clearly it is only λ_1 that is perturbed to $\lambda_1 + \delta$ and hence

$$\|A - A_\delta\|_2 = \delta, \;\; \|\Lambda - \Lambda_\delta\|_2 = \delta.$$

Therefore we have

$$\kappa_\Lambda(A) = \lim_{\delta \to 0} \frac{\|A - A_\delta\|}{\|\Lambda - \Lambda_\delta\|} = \lim_{\delta \to 0} \frac{\delta}{\delta} = 1$$

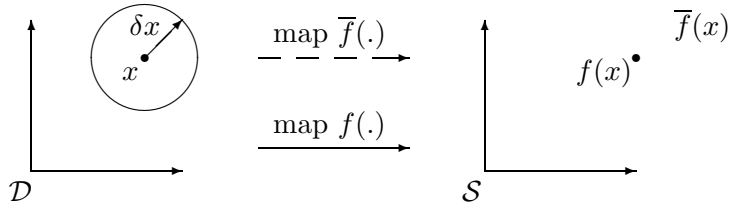
and the problem is thus very well conditioned.

These two examples illustrate well that the conditioning of a problem (the eigenvalue problem of a $n \times n$ matrix, in our case) depends as well on the data point x of our function $f(.)$ (the matrix A, in our case). Note that we have thus far made no mention of how the problem f above (computing $\Lambda(A)$ in the example) was to be solved. Conditioning is a function solely of the problem itself. We now turn our attention the the stability of an algorithm. Each algorithm performs little errors all along the computational process. Hence we do not implement the function $f(x)$ but rather the function $\overline{f}(x)$. Thus, given x, $\overline{f}(x)$ represents the result of applying the algorithm to x.

If we want to characterize the amount of errors incurred by this erroneous function, we need to relate it to $f(x)$. A standard way to do that in numerical linear algebra is to write $\overline{f}(x)$ as the effect of $f(.)$ on perturbed data $\overline{x} = x + \delta x$, i.e.,

$$\overline{f}(x) = f(\overline{x}) = f(x + \delta x). \tag{1.7a}$$

Notice that rigorously speaking, such a rewrite may not always be possible, especially when the data space $\mathcal{D}$ has smaller dimension than the solution space $\mathcal{S}$, but we will not dwell here on these aspects. An algorithm is now said to be (backward) stable when for all data points x one can guarantee that δx in the above rewrite, will be of the order of the machine accuracy ϵ. In pictures this means the following :



Here the computed $\overline{f}(x)$ also corresponds to the mapping $f(\overline{x}) = f(x + \delta x)$ of some nearby point $x + \delta x$. In words this means that what we computed is actually the *exact result of slightly perturbed data*. In practice this is about as much as one could hope, since we can also assume that the collected data are usually not of full accuracy anyway. Roughly speaking, one has then the following property:

$$\|\overline{f}(x) - f(x)\| = \|f(\overline{x}) - f(x)\| \approx \|x - \overline{x}\|\kappa(f, x). \tag{1.7b}$$

The first equality comes from the rewrite of $\overline{f}(x)$ as $f(\overline{x})$, the second approximation comes from the Taylor expansion of $f(\overline{x}) = f(x + \delta x)$, provided it exists of course. So, the algorithm $\overline{f}$ is said to be numerically (backward) stable if, for all $x \in \mathcal{D}$, there exists $\overline{x} \in \mathcal{D}$ near x such that $\overline{f}(x)$ equals $f(\overline{x})$ (= the exact solution of a nearby problem). *Near* in this context should be interpreted as $\|x - \overline{x}\| \leqslant \epsilon\|x\|$, i.e., of the order of the machine precision times the norm of the data point x. If the problem is (backward) stable, we thus have

$$\|\overline{f}(x) - f(x)\| = \|f(\overline{x}) - f(x)\| \approx \epsilon\|x\|\kappa(f, x). \tag{1.7c}$$

This implies that a backward stable algorithm does not introduce any more errors in the result than what is expected from the sensitivity of the problem. If moreover the problem is well-conditioned, this implies that then $f(\overline{x})$ will be near $f(x)$.

Of course, one cannot expect a stable algorithm to solve an ill-conditioned problem any more accurately than the data warrant but an unstable algorithm can produce poor solutions even to well-conditioned problems. Example 1.3, below, will illustrate this phenomenon.

Roundoff errors can cause unstable algorithms to give disastrous results. However, it would be virtually impossible to account for every roundoff error made at every arithmetic operation in a

complex series of calculations such as those involved in most linear algebra calculations. This would constitute a *forward* error analysis. The concept of *backward* error analysis based on the definition of numerical stability given above provides a more practical alternative. To illustrate this, let us consider the singular value decomposition of a $n \times n$ matrix A with coefficients in $\mathbb{R}$ or $\mathbb{C}$ [46]

$$A = U\Sigma V^H \ . \tag{1.8}$$

Here U and V are $n \times n$ unitary matrices, respectively, and Σ is an $n \times n$ matrix of the form

$$\Sigma = \mathrm{diag}\{\sigma_1, \cdots, \sigma_n\} \tag{1.9}$$

with the *singular value* σ_i being positive and satisfying $\sigma_1 \geqslant \sigma_2 \cdots \geqslant \sigma_n > 0$. The computation of this decomposition is, of course, subject to rounding errors. Denoting computed quantities by an overbar, we generally have for some *error matrix* E_A:

$$\overline{A} = A + E_A = \overline{U}\,\overline{\Sigma}\,\overline{V}^H \tag{1.10}$$

The computed decomposition thus corresponds exactly to a *perturbed* matrix $\overline{A}$. When using the SVD algorithm available in the literature[46], this perturbation can be bounded by:

$$\| E_A \| \leqslant \pi\epsilon \| A \|, \tag{1.11a}$$

where ϵ is the machine precision and π some quantity depending on the dimensions m and n, but reasonably close to 1 (see also [71]). Thus, the *backward error* E_A induced by this algorithm, has roughly the same norm as the *input error* E_i that results, for example, when reading the data A into the computer. Then, according to the definition of numerical stability given above, when a bound such as that in (1.11a) exists for the error induced by a numerical algorithm, the algorithm is said to be *backward stable* [148], [23]. Notice that backward stability does not guarantee any bounds on the errors in the result $\overline{U}, \overline{\Sigma}$, and $\overline{V}$. In fact this depends on how perturbations in the data (namely $E_A = \overline{A} - A$) affect the resulting decomposition (namely $E_U = \overline{U} - U, E_\Sigma = \overline{\Sigma} - \Sigma$, and $E_V = \overline{V} - V$). As pointed out earlier, this is commonly measured by the condition $\kappa(f, A)$ [110]. For the singular values Σ of a matrix A one shows [46] that

$$\kappa(\Sigma, A) = 1 \tag{1.11b}$$

independently of the matrix A ! Together with (1.11a) and (1.7b,1.7c) we then obtain a forward error bound on the computed singular values:

$$\|\Sigma - \overline{\Sigma}\|_2 \leqslant \|A - \overline{A}\|_2 = \|E_A\|_2 \leqslant \pi\epsilon\|A\|_2. \tag{1.11c}$$

In a sense this can be considered as an indirect forward error analysis by merely bounding the conditioning and the backward error analysis independently and using (1.7b,1.7c).

It is important to note that backward stability is a property of an algorithm while conditioning is associated with a problem and the specific data for that problem. The errors in the result depend on both the stability of the algorithm used and the conditioning of the problem solved. A *good* algorithm should therefore be backward stable since the size of the errors in the result is then mainly due to the condition of the problem, not to the algorithm. An unstable algorithm, on the other hand, may yield a large error even when the problem is well-conditioned.

Bounds of the type (1.11a) are obtained by an error analysis of the algorithm used; see, e.g., [148], [149]. The condition of the problem is obtained by a sensitivity analysis; see, e.g., [148], [142], [122], [127] for some examples. Two simple examples will illustrate further some of the concepts introduced above.

Example 1.3.

Let x and y be two floating-point computer numbers and let $fl(x*y)$ denote the result of multiplying them in floating-point computer arithmetic. In general, the product $x*y$ will require more precision to be represented exactly than was used to represent x or y. But what can be shown for most computers is that

$$fl(x * y) = x * y(1 + \delta) \tag{1.12}$$

where $|\delta| < \epsilon$ (= relative machine precision). In other words, $fl(x * y)$ is $x * y$ correct to within a unit in the last place. Now, another way to write (1.12) is as

$$fl(x * y) = x(1 + \delta)^{1/2} * y(1 + \delta)^{1/2} \tag{1.13}$$

where $|\delta| < \epsilon$. This can be interpreted as follows: the computed result $fl(x*y)$ is the exact product of the two slightly perturbed numbers $x(1+\delta)^{1/2}$ and $y(1+\delta)^{1/2}$. Note that the slightly perturbed data (not unique) may not even be representable floating-point numbers. The representation (1.13) is simply a way of accounting for the roundoff incurred in the algorithm by an initial (small) perturbation in the data.

The above example is actually the error analysis of the basic operation of multiplying two scalar numbers. One shows [148], [46] that present day machines satisfy the following basic error bounds for the elementary arithmetic operations:

$$fl(x * y) = (x * y)(1 + \delta_1) \tag{1.14a}$$

$$fl(x \pm y) = (x \pm y)(1 + \delta_2) \tag{1.14b}$$

$$fl(x/y) = (x/y)(1 + \delta_3) \tag{1.14c}$$

$$fl(\sqrt{x}) = (\sqrt{x})(1 + \delta_4) \tag{1.14d}$$

where $|\delta_i| < \epsilon$ (= relative machine precision). The reader ought to check that this implies in fact that these elementary operations are both backward and forward stable.

Example 1.4.

Gaussian elimination with no pivoting for solving the linear system of equations

$$Ax = b \tag{1.15}$$

is known to be numerically unstable. The following data will illustrate this phenomenon. Let

$$A = \begin{bmatrix} 0.0001 & 1.000 \\ 1.000 & -1.000 \end{bmatrix}, b = \begin{bmatrix} 1.000 \\ 0.000 \end{bmatrix}.$$

All computations will be carried out in four-significant-figure decimal arithmetic. The "true answer" $x = A^{-1}b$ is easily seen to be

$$\begin{bmatrix} 0.9999 \\ 0.9999 \end{bmatrix}.$$

Using row 1 as the "pivot row" (i.e., subtracting 10,000 $\times$ row 1 from row 2) we arrive at the equivalent triangular system

$$\begin{bmatrix} 0.0001 & 1.000 \\ 0 & -1.000 \times 10^4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.000 \\ -1.000 \times 10^4 \end{bmatrix}.$$

Note that the coefficient multiplying x_2 in the second equation should be -10,001, but because of roundoff, becomes -10,000. Thus, we compute $x_2 = 1.000$ (a good approximation), but back-substitution in the equation

$$0.0001x_1 = 1.000 - fl(1.000 * 1.000)$$

yields $x_1 = 0.000$. This extremely bad approximation to x_1 is the result of numerical instability. The problem itself can be shown to be quite well-conditioned.

This discussion only gives some indications about stability and conditioning but does not indicate how to prove these properties, or better how to find tight bounds for them. Below, we now give some indications about this, without going into too much details.

Sensitivity analysis

A sensitivity analysis is typically done from bounding first order perturbations. We give an example for the symmetric eigenvalue problem. Let A have the decomposition $A = U\Lambda U^H$ where the columns u_i of U are the eigenvectors of A and the diagonal elements λ_i of the diagonal matrix Λ are the eigenvalues of A. Now perturb A by δA and consider the perturbed equation

$$A + \delta A = (U + \delta U)(\Lambda + \delta\Lambda)(U^H + \delta U^H).$$

Equating the first order terms on both sides of this equation yields

$$\delta A = U\delta\Lambda U^H + \delta U\Lambda U^H + U\Lambda\delta U^H$$

or also

$$U^H AU = \delta\Lambda + U^H\delta U\Lambda + +\Lambda\delta U^H U.$$

One then shows [148] that the two last terms in this equation are zero on the diagonal and hence that the perturbation of the diagonals of Λ are just the diagonal elements of $U^H AU$, i.e.,

$$\delta\lambda_i = u_i^H Au_i$$

From this identity one now easily derives the sensitivity of the symmetric eigenvalue problem (see [148] for more details). All this of course implied that the above first order perturbation decomposition exists. It is well known [148] that for the case of multiple eigenvalues with Jordan blocks such a first order expansion does *not* exist. The analysis can thus in general be much more involved than this simple general principle described above.
Another way to proceed experimentally is to run a *stable* algorithm on the given problem. Errors on the result will then essentially be due to the sensitivity of the problem, which can then be measured by artificially perturbing the data and comparing differences in the result.

Stability analysis

One uses bounds of the type (1.14a, 1.14b, 1.14c, 1.14d) for the elementary operations to start with. As shown above these bounds are good. Then one checks the propagation and accumulation of these bounds during the evolution of the algorithm. In a sense this is comparable to the stability analysis of a dynamical system. When is the propagation of certain quantities stable ? One easily checks that error propagation is a nonlinear phenomenon, but luckily one is only interested in bounds for the errors, which makes the analysis more tractable. It is difficult to give a general methodology for performing an error analysis: therefore it is often referred to as an art in which J. H. Wilkinson certainly excelled [148]. Let us just mention the importance of unitary matrices in the stability proof of algorithms using them. Elementary orthogonal transformations such as Givens rotations and Householder transformations have been shown to be backward stable (see [148] and also next section). Since these transformations (as well as their inverse) have 2-norm equal to 1, one typically manages to prove stability of algorithms that *only* use orthogonal transformations. This relies on the fact that errors performed in previous steps of such algorithms propagate in a bounded fashion and can then be mapped back to the original data without increase in norm. How this is precisely done differs of course for each algorithm.
Also for stability analysis there is an experimental way of checking it. When running an algorithm on a well conditioned problem, it should give errors on the result which are proportional to the backward error of the algorithm. This does not guarantee stability of the algorithm in general, since then this ought to hold for *all data*, but usually it is a pretty good test.

1.4 Basic Problems in Numerical Linear Algebra

In this section we give a brief overview of some of the fundamental problems in numerical linear algebra which serve as building blocks or "tools" for the solution of problems in systems, control, and estimation. For an elementary but very pleasant introduction to this material we refer to [122]. For a more elaborate discussion on this material we refer to [46]. The latter is also an invaluable source of further references to the rather extended literature in this field.

A. Elementary Linear Algebraic Operations

All of the decompositions in this section are based on three types of matrix transformations only: elementary transformations (Gaussian type), Givens rotations and Householder transformations. We quickly review here their properties and complexity.

Elementary transformations are matrices that differ from the identity only in one element (typically below the diagonal). We depict one here with an element α in the bottom left corner:

$$E = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & & \ddots & \ddots & 0 \\ \alpha & 0 & \cdots & \cdots & 1 \end{bmatrix}.$$

Applying such a matrix transformation E to an $m \times n$ matrix A e.g., adds row 1 times α to the last row. If α is chosen equal to $-a_{n,1}/a_{1,1}$ then this "eliminates" the corresponding $(n,1)$ element in the product EA. The *complexity* of this operation measured in flops (1 flop = 1 addition + 1 multiplication) is n. The elimination procedure is *backward stable* provided $|\alpha| \leqslant 1$, or equivalently, if $|a_{n,1}| \leqslant |a_{1,1}|$, i.e., if the eliminated element is smaller than the element used to annihilate it (also called the "pivot"). When this is not satisfied, one typically interchanges the two concerned rows first, an operation which is called "pivoting". See [148], [122] for more details.

Givens rotations are matrices that differ from the identity only in four elements at positions (i,i), (i,j), (j,i) and (j,j):

$$G = \begin{bmatrix} 1 & .. & 0 & .. & 0 & .. & 0 \\ : & & : & & : & & : \\ 0 & .. & \cos\theta & .. & \sin\theta & .. & 0 \\ : & & : & & : & & : \\ 0 & .. & -\sin\theta & .. & \cos\theta & .. & 0 \\ : & & : & & : & & : \\ 0 & .. & 0 & .. & 0 & .. & 1 \end{bmatrix}.$$

Applying such a matrix G to an $m \times n$ matrix A "rotates" rows i and j of A. Again, if θ is chosen such that $\tan\theta = a_{j,1}/a_{i,1}$, then this annihilates the element $a_{j,1}$ in the product GA. The *complexity* of this operation measured in flops is now $4n$. but the elimination procedure is always *backward stable*, i.e., no pivoting is required. See [148], [122] for more details.

Householder transformations are matrices that differ from the identity by a rank one matrix:

$$H = I_m - 2uu^T, \;\; u^T u = 1.$$

Notice that for this reason the matrix has typically no nonzero elements, but multiplying a given $m \times n$ matrix A with H is still relatively cheap, since $HA = A + u(2u^T A)$.This is implemented by multiplying out first $u^T A$, then multiplying this vector by 2 and then adding $u(2u^T A)$ to A. The total complexity of this is $2mn$ if u has m nonzero elements, and it allows to eliminate $m-1$ elements in A. E.g., all but 1 element in the first column $a_{.,1}$ of A are annihilated if $u = v/\|v\|$ where v equals this first column $a_{.,1}$ except for the first element $v_1 = a_{1,1} - sign(a_{1,1})\|a_{.,1}\|$. The total complexity is therefore $2n$ per zero element created in HA. The algorithm is also proven to be backward stable irrespective of A. See [148, 122] for more details.

For the above three basic matrix operations E, G and H we note that in terms of complexity for creating a single zero element in a transformed matrix A, E is the cheapest, then H is twice as expensive, and G is four times as expensive.

B. Linear Algebraic Equations and Linear Least Squares Problems

Probably the most fundamental problem in numerical computing is the calculation of a vector x which satisfies the linear system

$$Ax = b \tag{1.16}$$

where $A \in \mathbb{R}^{n\times n}$(or $\mathbb{C}^{n\times n}$) and has rank n. A great deal is now known about solving (1.16) in finite arithmetic both for the general case and for a large number of special situations. Some of the standard references include [28, 34, 59, 122], and [46].

The most commonly used algorithm for solving (1.15) with general A and small n (say $n \leqslant 200$) is Gaussian elimination with some sort of pivoting strategy, usually "partial pivoting." This essentially amounts to factoring some permutation P of the rows of A into the product of a unit lower triangular matrix L and an upper triangular matrix U:

$$PA = LU. \tag{1.17}$$

The algorithm uses a sequence of elementary transformations to achieve this and it is effectively stable, i.e., it can be proved that the computed solution is near the exact solution of the system

$$(A + E)x = b \tag{1.18}$$

with $\|E\| \leqslant \pi(n)\,\gamma\,\|A\|\epsilon$ where $\pi(n)$ is a modest function of n depending on details of the arithmetic used, γ is a "growth factor" (which is a function of the pivoting strategy and is usually—but not always—small), and ϵ is the machine precision. In other words, except for moderately pathological situations, E is "small"—on the order of $\epsilon\,\|A\|$. The total complexity of the decomposition (1.14a) is $2/3n^3$ and the backsubstitutions for finding x have a lower order complexity than that of the decomposition. See [122], [46] for further details.

The following question now arises. If, because of roundoff errors, we are effectively solving (1.16) rather than (1.15), what is the relationship between $(A + E)^{-1}b$ and $A^{-1}b$? A condition number for the problem (1.15) is given by

$$\kappa(A) := \|A\|\ \|A^{-1}\|. \tag{1.19a}$$

Simple perturbation results can be used to show that perturbation in A and/or b can be magnified by as much as $\kappa(A)$ in the computed solution. Estimation of $\kappa(A)$ (since, of course, A^{-1} is unknown) is thus a crucial aspect of assessing solutions of (1.15) and the particular estimation procedure used is usually the principal difference between competing linear equation software packages. One of the more sophisticated and reliable condition estimators presently available is based on [20] and is implemented in LINPACK [28] and its successor LAPACK [2].

A special case of the above decomposition is the Cholesky decomposition for symmetric definite matrices A.

$$A = LL^T \tag{1.19b}$$

Because of properties of positive definite matrices, one shows that this decomposition can be computed in a backward stable manner *without pivoting*. Because of symmetry the complexity of this decomposition is now $1/3n^3$. The sensitivity of the solution is still the same as that for the general case. In the case of the positive definite matrices, one is also interested in the sensitivity of the decomposition itself, i.e., in $\kappa(L, A)$. The sensitivity of this factorization has been studied by Stewart in [125] and boils down essentially to that of a Sylvester like equation (we refer to [125] for more details).

Another important class of linear algebra problems and one for which codes are available in LINPACK and LAPACK is the linear least squares problem

$$\min \|Ax - b\|_2 \tag{1.20}$$

where $A \in \mathbb{R}^{m\times n}$ and has rank k, with (in the simplest case) $k = n \leqslant m$. The solution of (1.18) can be written formally as $x = A^+b$. Here, standard references include [28], [78], [46], and [122]. The method of choice is generally based upon the QR factorization of A (for simplicity, let $\operatorname{rank}(A) = n$)

$$A = QR \tag{1.21}$$

where $R \in \mathbb{R}^{n\times n}$ is upper triangular and $Q \in \mathbb{R}^{m\times n}$ has orthonormal columns, i.e., $Q^TQ = I$. With special care and analysis the case $k < n$ can also be handled similarly. The factorization is effected through a sequence of Householder transformations H_i applied to A. Each H_i is symmetric and orthogonal and of the form $I - 2uu^T/u^Tu$ where $u \in \mathbb{R}^m$ is specially chosen so that zeros are introduced at appropriate places in A when it is premultiplied by H_i. After n such transformations we have

$$H_nH_{n-1}\cdots H_1A = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

from which the factorization (1.19a,1.19b) follows. Defining c and d by

$$\begin{bmatrix} c \\ d \end{bmatrix} = H_nH_{n-1}\cdots H_1b$$

where $c \in \mathbb{R}^n$, it is easily shown that the least squares solution x of (1.18) is given by the solution of the linear system of equations

$$Rx = c\ . \tag{1.22}$$

The above algorithm can be shown to be numerically stable and has complexity $2n^2(m-n/3)$ when using Householder transformations and $4n^2(m-n/3)$ when using Givens transformations. Again, a well-developed perturbation theory exists from which condition numbers for the solution can be obtained, this time in terms of

$$\kappa(A) := \|A\|\ \|A^+\|.$$

We refer to [130] [47] for more details. Least squares perturbation theory is fairly straightforward when $\mathrm{rank}(A) = n$, but is considerably more complicated when A is rank-deficient. The reason for this is that while the inverse is a continuous function of the data (i.e., the inverse is a continuous function in a neighborhood of a nonsingular matrix), the pseudoinverse is discontinuous. For example, consider

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = A^+$$

and perturbations

$$E_1 = \begin{bmatrix} 0 & 0 \\ \delta & 0 \end{bmatrix}$$

and

$$E_2 = \begin{bmatrix} 0 & 0 \\ 0 & \delta \end{bmatrix}$$

with δ small. Then

$$(A+E_1)^+ = \begin{bmatrix} \frac{1}{1+\delta^2} & \frac{\delta}{1+\delta^2} \\ 0 & 0 \end{bmatrix}$$

which is close to A^+ but

$$(A + E_2)^+ = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\delta} \end{bmatrix}$$

which gets arbitrarily far from A^+ as δ is decreased towards 0. For a complete survey of perturbation theory for the least squares problem and related questions, see [125], [127].

Instead of Householder transformations, Givens transformations (elementary rotations or reflections) may also be used to solve the linear least squares problem. Details can be found in [28, 78, 107, 124], and [147]. Recently, Givens transformations have received considerable attention for the solution of both linear least squares problems as well as systems of linear equations in a parallel computing environment. The capability of introducing zero elements selectively and the need for only local interprocessor communication make the technique ideal for "parallelization." Indeed, there have been literally dozens of "parallel Givens" algorithms proposed and we include [41, 53, 79, 117, 81], and [94] as representative references.

C. Eigenvalue and Generalized Eigenvalue Problems

In the algebraic eigenvalue/eigenvector problem for $A \in \mathbb{R}^{n\times n}$, one seeks nonzero solutions $x \in \mathbb{C}^n$ and $\lambda \in \mathbb{C}$ which satisfy

$$Ax = \lambda x. \tag{1.23}$$

The classic reference on the numerical aspects of this problem is Wilkinson [148] with Parlett [107] providing an equally thorough and up-to-date treatment of the case of symmetric A (in which $x \in \mathbb{R}^n, \lambda \in \mathbb{R}$). A more brief textbook introduction is given in [122] [46].

It is really only rather recently that some of the computational issues associated with solving (1.23) — in the presence of rounding error — have been resolved or even understood. Even now some problems such as the invariant subspace problem continue to be active research areas. For an introduction to some of the difficulties which may be encountered in trying to make numerical sense out of mathematical constructions such as the Jordan canonical form, the reader is urged to consult [48].

The most common algorithm now used to solve (1.23) for general A is the QR algorithm of Francis [35, 36]. A shifting procedure (see [122, 46] for a brief explanation) is used to enhance convergence and the usual implementation is called the double-Francis-QR algorithm. Before the QR process is applied, A is initially reduced to upper Hessenberg form $A_H (a_{ij} = 0 \text{ if } i-j \geqslant 2)$ [84]. This is accomplished by a finite sequence of similarities which can be chosen to be of the Householder form discussed above. The QR process then yields a sequence of matrices which are orthogonally similar to A and which converge (in some sense) to a so-called quasi-upper-triangular matrix S which is also called the real Schur form (RSF) of A. The matrix S is block-upper-triangular with 1×1 diagonal blocks corresponding to real eigenvalues of A and 2×2 diagonal blocks corresponding to complex-conjugate pairs of eigenvalues. The quasi-upper-triangular form permits all arithmetic done to be real rather than complex as would be necessary for convergence to an upper triangular matrix. The orthogonal transformations from both the Hessenberg reduction and the QR process may be accumulated into a single orthogonal transformation U so that

$$U^T A U = R \tag{1.24}$$

compactly represents the entire algorithm. It is known that the algorithm for this decomposition is backward stable and has a complexity of kn^3, where k accounts for the iteration in the algorithm

and may vary between $10 and 25$. For the sensitivity of eigenvalues, eigenvectors and eigenspaces we refer to [148, 123] and [46]. For the computation of the Jordan normal form there are no results of guaranteed backward stability around and the complexity of the decomposition is much higher than that of the Schur decomposition. For this reason, we strongly recommend not to use this decomposition whenever one can use instead the more reliable Schur form.

An analogous process can be applied in the case of symmetric A and considerable simplifications and specializations result. Moreover, [107, 148, 46], and [127] may be consulted regarding an immense literature concerning stability of the QR and related algorithms, and conditioning of eigenvalues and eigenvectors. Both subjects are vastly more complex for the eigenvalue/eigenvector problem than for the linear equation problem.

Quality mathematical software for eigenvalues and eigenvectors is available; the EISPACK [39], [120] collection of subroutines represents a pivotal point in the history of mathematical software. This collection is primarily based on the algorithms collected in [149]. The successor to EISPACK (and LINPACK) is the recently released LAPACK [2] in which the algorithms and software have been restructured to provide high efficiency on vector processors, high performance workstations, and shared memory multiprocessors.

Closely related to the QR algorithm is the QZ algorithm [96] for the generalized eigenvalue problem

$$Ax = \lambda Mx \tag{1.25}$$

where $A, M \in \mathbb{R}^{n\times n}$. Again, a Hessenberg-like reduction, followed by an iterative process are implemented with orthogonal transformations to reduce (1.25) to the form

$$QAZy = \lambda QMZy \tag{1.26}$$

where QAZ is quasi-upper-triangular and QMZ is upper triangular. For a review and references to results on stability, conditioning, and software related to (1.25) and the QZ algorithm see [46]. The generalized eigenvalue problem is both theoretically and numerically more difficult to handle than the ordinary eigenvalue problem, but it finds numerous applications in control and system theory [131, 132], [141]. The algorithm is again backward stable and its complexity is kn^3 where k varies between 30 and 70.

D. The Singular Value Decomposition and Some Applications

One of the basic and most important tools of modern numerical analysis, particularly numerical linear algebra, is the singular value decomposition (SVD). We make a few comments about its properties and computation as well as its significance in various numerical problems.

Singular values and the singular value decomposition have a long history particularly in statistics and more recently in numerical linear algebra. Even more recently the ideas are finding applications in the control and signal processing literature, although their use there has been overstated somewhat in certain applications. For a survey of the singular value decomposition, its history, numerical details, and some applications in systems and control theory, see [71].

Theorem 1.1. Let $A \in \mathbb{C}^{m\times n}$ with $\operatorname{rank}(A) = r$. Then there exist unitary matrices $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ such that

$$A = U\Sigma V^H \tag{1.27}$$

where

$$\Sigma = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}$$

is real and $\Sigma_r = \text{diag}\,\{\sigma_1, \cdots, \sigma_r\}$ with $\sigma_1 \geqslant \cdots \geqslant \sigma_r > 0$. If A is real instead of complex, U and V will be real orthogonal.

The proof of Theorem 1.1 is straightforward and can be found in, for example, [43, 46], and [122]. Geometrically, the theorem says that bases can be found (separately) in the domain and codomain spaces of a linear map with respect to which the matrix representation of the linear map is diagonal. The numbers $\sigma_1, \cdots, \sigma_r$ together with $\sigma_{r+1} = 0, \cdots, \sigma_n = 0$ are called the singular values of A and they are the positive square roots of the eigenvalues of $A^H A$. The columns $\{u_k, k = 1, \ldots, m\}$ of U are called the left singular vectors of A (the orthonormal eigenvectors of AA^H), while the columns $\{v_k, k = 1, \cdots, n\}$ of V are called the right singular vectors of A (the orthonormal eigenvectors of $A^H A$). The matrix A can then also be written (as a dyadic expansion) in terms of the singular vectors as follows:

$$A = \sum_{k=1}^{r} \sigma_k u_k v_k^H.$$

The matrix A^H has m singular values, the positive square roots of the eigenvalues of AA^H. The r [= rank (A)] nonzero singular values of A and A^H are, of course, the same. The choice of $A^H A$ rather than AA^H in the definition of singular values is arbitrary. Only the nonzero singular values are usually of any real interest and their number, given the SVD, is the rank of the matrix. Naturally, the question of how to distinguish nonzero from zero singular values in the presence of rounding error is a nontrivial task. The standard algorithm for computing the SVD is based on a preliminary bidiagonalization obtained by Householder transformations followed by a further iteration to diagonalize the matrix. The overall process requires kn^3 operations where k lies between 4 and 26. Another algorithm due to Kogbetliantz [72] uses only Givens transformations and is usually slower but is better suited for matrices with small off-diagonal elements to start with. Both algorithms are backward stable.

It is not generally advisable to compute the singular values of A by first finding the eigenvalues of $A^H A$ (remember the folk theorem!), tempting as that is. Consider the following real example with μ a real number with $|\mu| < \sqrt{\epsilon}$ (so that $fl(1 + \mu^2) = 1$ where $fl(\cdot)$ denotes floating-point computation). Let

$$A = \begin{bmatrix} 1 & 1 \\ \mu & 0 \\ 0 & \mu \end{bmatrix}.$$

Then

$$fl(A^T A) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

so we compute $\hat{\sigma}_1 = \sqrt{2},\ \hat{\sigma}_2 = 0$ leading to the (erroneous) conclusion that the rank of A is 1. Of course, if we could compute in infinite precision we would find

$$A^T A = \begin{bmatrix} 1 + \mu^2 & 1 \\ 1 & 1 + \mu^2 \end{bmatrix}$$

with $\sigma_1 = \sqrt{2 + \mu^2}, \sigma_2 = |\mu|$ and thus rank $(A) = 2$. The point is that by working with $A^T A$ we have unnecessarily introduced μ^2 into the computations. The above example illustrates a potential

pitfall in attempting to form and solve the normal equations in a linear least squares problem, and is at the heart of what makes square root filtering so attractive numerically. Very simplistically speaking, square root filtering involves working directly on an "A-matrix," for example updating it, as opposed to working on (updating, say) an "A^TA-matrix." See [10] for further details and references.

Square root filtering is usually implemented using the QR factorization (or some closely related algorithm) as described previously rather than SVD. The key thing to remember is that in most current computing environments, the condition of the least-squares problem is squared unnecessarily in solving the normal equations. Moreover, critical information may be lost irrecoverable by simply forming A^TA.

Returning now to the SVD there are two features of this matrix factorization that make it so attractive in finite arithmetic: First, it can be computed in a numerically stable way, and second, singular values are well-conditioned. Specifically, there is an efficient and numerically stable algorithm due to Golub and Reinsch [45] (based on [43]) which works directly on A to give the SVD. This algorithm has two phases: In the first phase, it computes unitary matrices U_1 and V_1 such that $B = {U_1}^H A V_1$ is in bidiagonal form, i.e., only the elements on its diagonal and first super-diagonal are non-zero. In the second phase, the algorithm uses an iterative procedure to compute unitary matrices U_2 and V_2 such that ${U_2}^H B V_2$ is diagonal and non-negative. The SVD defined in (1.27) is then given by $\Sigma = U^H B V$, where $U = U_1 U_2$ and $V = V_1 V_2$. The computed U and V are unitary to approximately the working precision, and the computed singular values can be shown to be the exact σ_i's for $A+E$ where $\|E\|/\|A\|$ is a modest multiple of ϵ. Fairly sophisticated implementations of this algorithm can be found in [28] and [39]. The well-conditioned nature of the singular values follows from the fact that if A is perturbed to $A+E$, then it can be proved that

$$|\sigma_i(A+E) - \sigma_i(A)| \leqslant \|E\|.$$

Thus, the singular values are computed with small absolute error although the relative error of sufficiently small singular values is not guaranteed to be small. A new algorithm for computing the singular values of a bidiagonal matrix [26] overcomes this deficiency to the extent that it computes the singular values of a bidiagonal matrix to the same relative precision as that of the individual matrix entries. In other words, the algorithm will obtain accurate singular values from accurate bidiagonal matrices. However, one cannot in general guarantee high accuracy in the reduction to bidiagonal form.

It is now acknowledged that the singular value decomposition is the most generally reliable method of determining rank numerically (see [48] for a more elaborate discussion). However, it is considerably more expensive to compute than, for example, the QR factorization which, with column pivoting [28], can usually give equivalent information with less computation. Thus, while the SVD is a useful theoretical tool, its use for actual computations should be weighed carefully against other approaches.

Only rather recently has the problem of numerical determination of rank become well-understood. A recent treatment of the subject can be found in the paper by Chan [18]; see also [126]. The essential idea is to try to determine a "gap" between "zero" and the "smallest nonzero singular value" of a matrix A. Since the computed values are exact for a matrix near A, it makes sense to consider the rank of all matrices in some δ-ball (with respect to the spectral norm $\|\cdot\|$, say) around A. The choice of δ may also be based on measurement errors incurred in estimating the coefficients of A, or the coefficients may be uncertain because of roundoff errors incurred in a previous computation

to get them. We refer to [126] for further details. We must emphasize, however, that even with SVD, numerical determination of rank in finite arithmetic is a highly nontrivial problem.

That other methods of rank determination are potentially unreliable is demonstrated by the following example which is a special case of a general class of matrices studied by Ostrowski [101]. Consider the matrix $A \in \mathbb{R}^{n\times n}$ whose diagonal elements are all -1, whose upper triangle elements are all $+1$, and whose lower triangle elements are all 0. This matrix is clearly of rank n, i.e., is invertible. It has a good "solid" upper triangular shape. All of its eigenvalues $(= -1)$ are well away from zero. Its determinant is $(-1)^n$—definitely not close to zero. But this matrix is, in fact, very nearly singular and gets more nearly so as n increases. Note, for example, that

$$\begin{bmatrix} -1 & +1 & \cdots & & \cdots & +1 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & +1 \\ 0 & \cdots & & \cdots & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2^{-1} \\ \vdots \\ 2^{-n+1} \end{bmatrix}$$

$$= \begin{bmatrix} -2^{-n+1} \\ -2^{-n+1} \\ \vdots \\ -2^{-n+1} \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} (n \rightarrow +\infty).$$

Moreover, adding 2^{-n+1} to every element in the first column of A gives an exactly singular matrix. Arriving at such a matrix by, say Gaussian elimination, would give no hint as to the near-singularity. However, it is easy to check that $\sigma_n(A)$ behaves as 2^{-n+1}. A corollary for control theory: eigenvalues do not necessarily give a reliable measure of "stability margin." As an aside it is useful to note here that in this example of an invertible matrix, the crucial quantity, $\sigma_n(A)$, which measures nearness to singularity, is simply $1/\|A^{-1}\|$, and the result is familiar from standard operator theory. There is nothing intrinsic about singular values in this example and, in fact, $\|A^{-1}\|$ might be more cheaply computed or estimated in other matrix norms. This is precisely what is done in estimating the condition of linear systems in LINPACK where $\|\cdot\|_1$ is used [20].

Since rank determination, in the presence of roundoff error, is a nontrivial problem, all the same difficulties will naturally arise in any problem equivalent to or involving rank determination, such as determining the independence of vectors, finding the dimensions of certain subspaces, etc. Such problems arise as basic calculations throughout systems, control, and estimation theory. Selected applications are discussed in more detail in [71].

Finally, let us close this section with a brief example illustrating a totally inappropriate use of SVD. The rank condition

$$\text{rank } [B, AB, \cdots, A^{n-1}B] = n \tag{1.28}$$

for the controllability of (1.1) is too well-known. Suppose

$$A = \begin{bmatrix} 1 & \mu \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 \\ \mu \end{bmatrix}$$

with $|\mu| < \sqrt{\epsilon}$. Then

$$fl[B, AB] = \begin{bmatrix} 1 & 1 \\ \mu & \mu \end{bmatrix}$$

and now even applying SVD, the erroneous conclusion of uncontrollability is reached. Again the problem is in just forming AB; not even SVD can come to the rescue after that numerical *faux pas*.

Chapter 2

IDENTIFICATION

In this chapter we present a number of identification techniques for linear time invariant systems. Such techniques exist for both continuous-time and discrete-time systems, but since we start from input/output signals, it is much more convenient to treat the discrete-time case. In practice the continuous-time case if often solved via the (discrete) sampled versions of the signals anyway. So here we treat the discrete-time case only and we will make a few remarks about the continuous-time case when appropriate. We will consider the cases when the data collected from the system are

- the impulse response of the system
- an input output pair of the system
- covariance data from the system excited by white noise.

In each of these cases we will consider both the single-input single-output (SISO) case and the multi-input multi-output (MIMO) case. We do not stress the theory of the identification problem but rather the various aspects of the underlying matrix problems, such as numerical accuracy, sensitivity and complexity of the algorithms.

2.1 SISO identification from the impulse response

Suppose a system can be modeled by the transfer function

$$h(z) = \frac{n(z)}{d(z)}, \tag{2.1}$$

where $n(z)$ and $d(z)$ are scalar polynomials satisfying the condition

$$n \doteq deg.d(z) \geqslant deg.n(z) \doteq m.$$

This amounts to say that the system is assumed to be causal. Moreover we assume the impulse response of the system is given, i.e., the response of

$$(d_n z^n + \ldots + d_1 z + d_0)\, y_i = (n_m z^m + \ldots + n_1 z + n_0)\, u_i, \tag{2.2}$$

where $\{u_i\} = \{1, 0, 0, 0, \ldots\}$. If we denote by 1 the z-transform of this input sequence, the the output sequence $\{y_i\}$ has a z-transform equal to

$$y(z) = \frac{n(z)}{d(z)} \cdot 1. \tag{2.3}$$

This is nothing but the transfer function (2.1) and thus the outputs y_i are the coefficients h_i of the expansion of (2.1) in z^{-1}:

$$h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + \ldots = h(z) = \frac{n(z)}{d(z)}. \tag{2.4}$$

Rewriting this as

$$h(z).d(z) = n(z), \tag{2.5}$$

and equating terms of equal powers of z yields the semi-infinite system:

$$\left[\begin{array}{ccccc} 0 & \ldots & \ldots & 0 & h_0 \\ \vdots & \iddots & \iddots & h_0 & h_1 \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ 0 & h_0 & h_1 & \iddots & h_{n-1} \\ h_0 & h_1 & \iddots & & h_n \\ \hline h_1 & h_2 & \iddots & & h_{n+1} \\ h_2 & \iddots & & \iddots & \vdots \\ \vdots & & \iddots & \iddots & \vdots \end{array}\right] \cdot \left[\begin{array}{c} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{array}\right] = \left[\begin{array}{c} n_n \\ n_{n-1} \\ \vdots \\ n_1 \\ n_0 \\ \hline 0 \\ \vdots \\ \vdots \end{array}\right] \tag{2.6}$$

If we normalize $n(z)/d(z)$ such that $d_n = 1$, then the bottom part of the above system can be rewritten as:

$$\left[\begin{array}{ccccc} h_1 & h_2 & \iddots & & h_n \\ h_2 & \iddots & & \iddots & h_{n+1} \\ \vdots & & \iddots & \iddots & \vdots \\ h_n & h_{n+1} & \iddots & & h_{2n-1} \\ \hline h_{n+1} & \ldots & \ldots & h_{2n-1} & h_{2n} \\ \vdots & \iddots & \iddots & h_{2n} & h_{2n+1} \\ \vdots & \iddots & \iddots & \iddots & \vdots \end{array}\right] \cdot \left[\begin{array}{c} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \end{array}\right] + \left[\begin{array}{c} h_{n+1} \\ h_{n+2} \\ \vdots \\ h_{2n} \\ \hline h_{2n+1} \\ \vdots \\ \vdots \end{array}\right] = 0 \tag{2.7}$$

This system is solvable if the left matrix has full column rank n. Once the coefficients of $d(z)$ are known it is clear that those of $n(z$ can be found from the top $n+1$ equations of (2.6). Also if some of the leading parameters $h_0, h_1, \ldots$ are zero (i.e., if there is a delay in the response) then clearly the degree m of $n(z)$ will be lower than the degree n of $d(z)$. One proves the following result:

Theorem 2.1. If n is the order of the system with impulse response $\{h_i,\ i = 0, 1, 2, \ldots\}$ then the semi-infinite matrix

$$\left[\begin{array}{ccccc} h_1 & h_2 & .^{\cdot} & & h_i \\ h_2 & .^{\cdot} & & .^{\cdot} & h_{i+1} \\ \vdots & & .^{\cdot} & .^{\cdot} & \vdots \\ h_i & h_{i+1} & .^{\cdot} & & h_{2i-1} \\ \hline h_{i+1} & \ldots & \ldots & h_{2i-1} & h_{2i} \\ \vdots & .^{\cdot} & .^{\cdot} & h_{2i} & h_{2i+1} \\ \vdots & .^{\cdot} & .^{\cdot} & .^{\cdot} & \vdots \end{array}\right] \tag{2.8}$$

has rank n for all $i \geqslant n$.

Proof. This property follows from the Hankel structure of that matrix. Since there is an n-th order system describing the input output behavior, (2.7) holds and the rank of the matrix (2.8) can then be at most n since each column $i > n$ of (2.8) is also a subcolumn of the right hand side of (2.7), and equation (2.7) says that it is a linear combination of the n previous columns in the matrix (2.8). For $i = n$ the rank must be equal to n otherwise we would find a model of order lower than n using (2.7). □

We will also prove later on that if n is the correct order of the system, it actually follows that the leading $n \times n$ matrix in (2.7) (above the horizontal line) is in fact invertible. This thus suggests the following identification algorithm.

Algorithm 2.1.

Construct

$$A = \left[\begin{array}{cccc} h_1 & h_2 & .^{\cdot} & h_n \\ h_2 & .^{\cdot} & .^{\cdot} & h_{n+1} \\ \vdots & .^{\cdot} & .^{\cdot} & \vdots \\ h_n & h_{n+1} & .^{\cdot} & h_{2n-1} \end{array}\right],\ B = \left[\begin{array}{cccc} 0 & \ldots & 0 & h_0 \\ \vdots & .^{\cdot} & h_0 & h_1 \\ 0 & .^{\cdot} & .^{\cdot} & .^{\cdot} \\ h_0 & h_1 & .^{\cdot} & h_n \end{array}\right],\ b = -\left[\begin{array}{c} h_{n+1} \\ h_{n+2} \\ \vdots \\ h_{2n} \end{array}\right]$$

then one solves the unknown vectors

$$d = \left[\begin{array}{c} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \end{array}\right],\ n = \left[\begin{array}{c} n_n \\ \vdots \\ n_1 \\ n_0 \end{array}\right]$$

using

$$d = A^{-1}b,\ \ n = B\left[\begin{array}{c} d \\ 1 \end{array}\right].$$

One checks that the following MATLAB commands actually implement this, where the column vector `h` is assumed to contain the impulse response $\{h_1,\ i = 0, 1, 2, \ldots\}$ of the system.

```
A=hankel(h(2:n+1),h(n+1,2*n)); b=-h(n+2:2*n+1);
B=hankel([zeros(1,n),h(1)],h(1:n+1)); d=A\b; d=[d;1]; n=B*d;
```

□This algorithm only uses the first $2n+1$ samples of the impulse response to determine the $2n+1$ unknown parameters of $d(z)$ and $n(z)$. But what should we do if we have to our disposal the impulse response $\{h_0, h_1, \ldots, h_N\}$ up to sample N where $N \gg 2n+1$? It makes sense to try to use as much information as possible for determining the polynomials $d(z)$ and $n(z)$. Since the system (2.7) is compatible, we can as well define

```
A=hankel(h(2:n+1),h(n+1,N-1)); b=-h(n+2:N);
```

and compute then

```
B=hankel([zeros(1,n),h(1)],h(1:n+1)); d=A\b; d=[d;1]; n=B*d;
```

This now solves $Ad = b$ in least squares sense. Can we say anything about the numerical reliability of one approach versus the other ? The sensitivity bounds we find e.g., in [46] say the following. The sensitivity of the square system

$$A_{n,n}d = b_n$$

has sensitivity

$$\|\overline{d} - d\| \leqslant \kappa(A_{n,n})\|d\| \left(\frac{\|\delta A_{n,n}\|}{\|A_{n,n}\|} + \frac{\|\delta b_n\|}{\|b_n\|} \right)$$

where all norms are 2-norms. For the least squares problem

$$A_{N,n}d = b_N$$

the comparable formula is

$$\|\overline{d} - d\| \leqslant \kappa(A_{N,n})\|d\| \left(\frac{\|\delta A_{N,n}\|}{\|A_{N,n}\|} + \frac{\|\delta b_N\|}{\|b_N\|} \right) \left(c_1 + c_2\kappa(A_{N,n}) \frac{\|r\|}{\|b\|} \right)$$

where c_1 and c_2 are scalars close to 1 and r is the residual vector of the least squares system. Although it appears from this that the least squares problem may have a condition number which is essentially the square of the linear system, it happens here that $\|r\| = 0$ and then both formulas look essentially the same. In practice, it just happens that the linear system is often much more sensitive than the least squares problem. The reason for this is twofold. First, since $A_{n,n}$ is a submatrix of $A_{N,n}$ its singular values can only be smaller ([46]) and hence typically its condition number is larger. Secondly, if the data in the interval $0 \leqslant i \leqslant 2n$ happen to be much smaller than in the complete interval, the relative errors on these samples are typically larger as well. An example of this poorer behaviour of the linear system is given below, where indeed it appears that the least squares problem behaves much better.

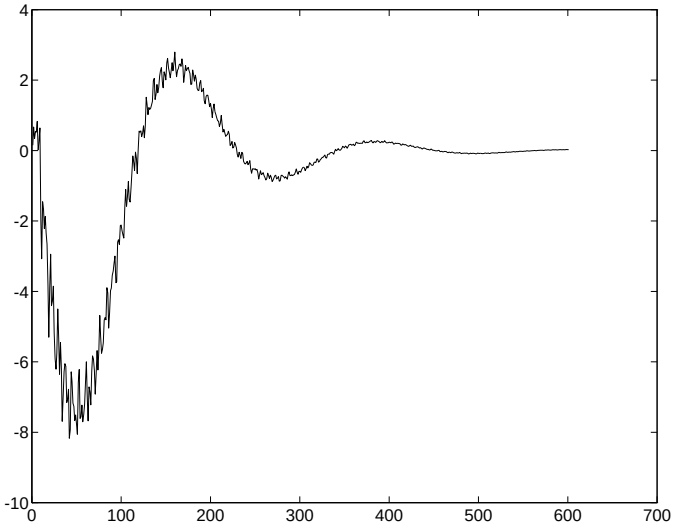


Figure 2.1: Impulse response of a lightly damped system.

Example 2.1.

The following example is a 8-th order system with impulse response as given by the plot in Figure 2.1.

Clearly the system is lightly damped and it makes sense to reconstruct the transfer function from all available data since the first few samples are much smaller than the rest of the data. The two graphs in Figure 2.2 give the errors obtained from simulating the original impulse response from the two reconstructed systems. The first one is from the linear system, the second one is from the least squares system.

For each system we first slightly perturbed the given impulse response with noise of size 10^{-9} before identifying the system, in order to estimate the sensitivity of the reconstruction. For the linear system that seems to be around 10^4 whereas for the least squares problem it is about 10^1. □

Example 2.2.

Take $\{u_i\} = \{1, 0, 0, 0, \ldots\}$ and $\{y_i\} = \{0, 0, 1, 0, 0, \ldots\}$, i.e., the system acts as a double delay and we should find $n = 2$. Then the square system is just as good as the semi-infinite one since all data beyond $2n = 4$ is zero. So we solve:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \end{bmatrix} = 0$$

which gives indeed the polynomial $d(z) = z^2$. The numerator then is found from

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ 1 \end{bmatrix} = \begin{bmatrix} n_2 \\ n_1 \\ n_0 \end{bmatrix}$$

and we find $n(z) = 1$. The transfer function $h(z) = n(z)/d(z)$ is thus a double delay $h(z) = 1/z^2$ as expected. □

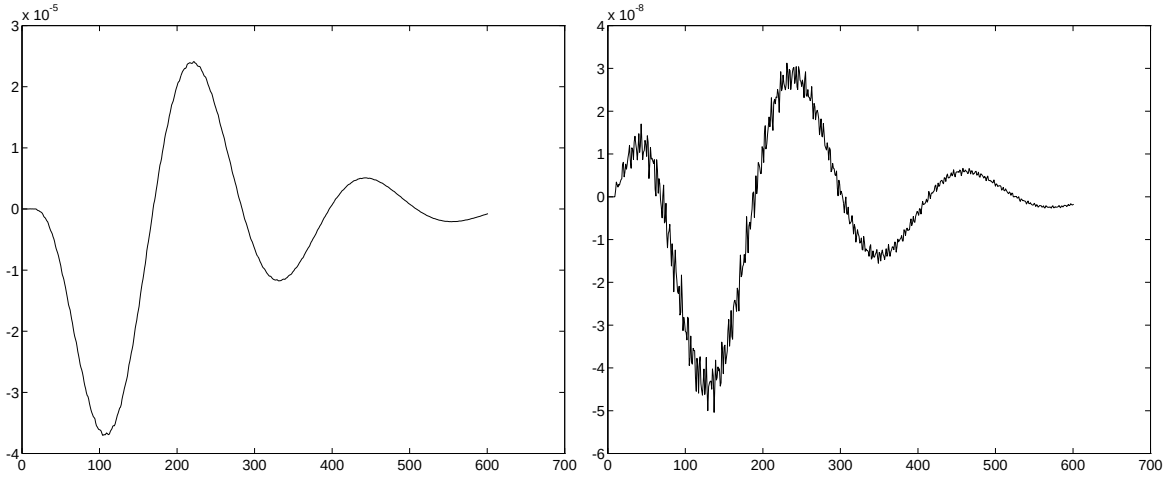


Figure 2.2: Errors for the reconstructed signals.

Remark

Notice that the equation $A_{N,n}d = b_N$ may also be weighted by a diagonal matrix $diag(w_1, \ldots, w_N)$ which gives different weight to each individual equation. This is typically done in recursive least squares where N is steadily increasing to infinity. The weighting matrix cares for the triangular factor to remain bounded, but can also be used to obtain better "tracking" capabilities (see later). □

So far we have assumed that the order of the system was given. In practice, this is another important parameter to be determined from the data. Theorem 2.1 in fact gives an important indication about this: the rank of the Hankel matrices of growing dimensions ought to be bounded by the order of the underlying system. But rank determination is something that can be done reliably using the singular value decomposition [47] (or a rank revealing QR decomposition [18]). The idea of this decomposition is the following. The complexity of the SVD is about $10Nn^2$, whereas that of the QR decomposition is only $2n^2(N - n/3)$, for a $N \times n$ matrix. One can actually preprocess any $N \times n$ matrix $A_{N,n}$ where $N \gg n$ by a QR decomposition:

$$A_{N,n} = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$$

and then apply a SVD to R:

$$R = U\Sigma V^H$$

to obtain the SVD of A:

$$A_{N,n} = Q \begin{bmatrix} U & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^H.$$

This "trick" already reduces the amount of work to $2n^2(N - n/3) + 10n^3 = 2n^2N + 9\frac{1}{3}n^3$, which is a lot of saving when $N \gg n$ (these operation counts are with the construction of Q included). In combination with this the QR decomposition has the possibility to *estimate* the rank of the matrix

via several techniques [18], without actually computing the singular values. This shows that when e.g., $N = 1000$ samples of the impulse response are given then the order of the system should rather be checked on a tall thin matrix rather than on a square one. If e.g., the suspected order is, say 8, then the rank property of Theorem 2.1 could as well be checked on a 999×10 matrix as on a 500×500 one. The complexity of the first approach is roughly $10.(500)^3$ whereas for the second it is roughly $2(10)^2(990) + 9\frac{1}{3}(10)^3$. In this particular case the latter is about 6,000 times faster !

We terminate this section with a statistical interpretation of the problem or order determination using singular values. When equating the inputs and outputs of a system as

$$(d_n z^n + \ldots + d_1 z + d_0)\, y_i = (n_m z^m + \ldots + n_1 z + n_0)\, u_i$$

one normally has no exact match (either due to noise on the date, or just due to rounding errors in these equations). Let us now rewrite this as

$$(d_n z^n + \ldots + d_1 z + d_0)\, y_i = (n_m z^m + \ldots + n_1 z + n_0)\, u_i + \ e_i \tag{2.9}$$

where e_i is some kind of "noise process". The bottom part of equation (2.6) which was used to determine $d(z)$ now becomes

$$\begin{bmatrix} h_1 & h_2 & \iddots & h_{n+1} \\ h_2 & \iddots & \iddots & \vdots \\ \vdots & h_{n+1} & \iddots & \vdots \\ h_{n+1} & \iddots & \iddots & h_{2n+1} \\ \vdots & \iddots & \iddots & \vdots \\ h_{N-n} & \iddots & \iddots & h_N \end{bmatrix} \cdot \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} e_{n+1} \\ e_{n+2} \\ \vdots \\ \vdots \\ e_N \end{bmatrix}.$$

Now this equation says that there is a linear combination of the columns of the Hankel matrix $H_{N-n,n+1}$ on the left that is "nearly" zero, or just noise e_i. What the singular value decomposition does is give the smallest possible perturbation $e \doteq [e_{n+1}, \ \ldots \ , e_N]^T$ which can be obtained with a vector $d = [d_0, \ \ldots \ , d_n]^T$ normalized to have unit length. In terms of the singular value decomposition of the matrix $H_{N-n,n+1}$, the vector d is then in fact the last right singular vector v_{n+1} of the Hankel matrix, and the vector e then equals $\sigma_{n+1} u_{n+1}$, the last left singular vector scaled by the smallest singular value. In other words, the smallest singular value is the norm (or variance) you need to allow in (2.9) to make the equation hold, with a polynomial $d(z)$ normalized to have a coefficient vector d of unit norm. When normalizing the polynomial with $d_n = 1$ as suggested earlier, this has to be scaled with the factor $d_n / \|d\|$.

2.2 State-space realizations

Single input single output systems can also be represented by state-space realizations

$$\begin{aligned} x_{k+1} &= A.x_k + b.u_k & (2.10) \\ y_k &= c.x_k + d.u_k, & (2.11) \end{aligned}$$

where A is $n \times n$, b and c are n-vectors and d is a scalar. In z-transforms language, this becomes:

$$\begin{aligned} (zI - A)x_k &= b.u_k \\ y_k &= c.x_k + d.u_k. \end{aligned}$$

When eliminating the variable x_k from this, we find the transfer function $h(z)$ of the system as

$$h(z) = c(zI - A)^{-1}b + d.$$

For the pair of polynomials $d(z)$ and $n(z)$ with transfer function and with impulse response as given in (2.4), we actually can derive immediately a state-space realization as follows:

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 1 \\ -d_0 & -d_1 & \dots & \dots & -d_{n-1} \end{bmatrix}, \; b = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ \vdots \\ h_n \end{bmatrix}, \; c = \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \end{bmatrix}, \; d = h_0. \quad (2.12)$$

A proof of this is found by equating the two expressions for the transfer function [65], but a simple intuitive way of checking this is to compute the impulse response of the above state space model, say for the first n samples. With zero initial state $x_0 = 0$ and input $u_i = 1, 0, 0, \dots$ it is obvious that $x_1 = A.x_0 + b.u_0 = b = \begin{bmatrix} h_1 & h_2 & \dots & h_n \end{bmatrix}^T$ and $y_0 = d.u_0 = h_0$. For the subsequent steps $1 < k \leqslant n$ one easily checks that $x_k = A.x_{k-1} = \begin{bmatrix} h_k & \dots & h_n & * \dots * \end{bmatrix}^T$. Since the output y_k for $0 < k$ is just the first entry of x_k one indeed checks that the outputs are matched for the first n samples. A more rigorous proof is given later on.

Notice that if a realization $\{A, b, c, d\}$ of a transfer function is given, then for any invertible tranformation T the quadruple $\{\hat{A}, \hat{b}, \hat{c}, \hat{d}\} \doteq \{T^{-1}AT, T^{-1}b, cT, d\}$ also realizes the same transfer function. One easily checks that indeed their transfer functions are identical:

$$\hat{c}(zI - \hat{A})^{-1}\hat{b} + \hat{d} = cT(zI - T^{-1}AT)^{-1}T^{-1}b + d = c(zI - A)^{-1}b + d. \quad (2.13)$$

Another way to check the relation between the system $\{A, b, c, d\}$ and the transfer function $h(z)$ is by expanding the expression $c(zI - A)^{-1}b + d$ in powers of z^{-1} as in (2.4) for the transfer function. Using the Neumann expansion of $(zI - A)^{-1}$:

$$(zI - A)^{-1} = z^{-1}(I - z^{-1}A)^{-1} = z^{-1}I + z^{-2}A + z^{-3}A^2 + z^{-4}A^3 + \dots$$

we easily find

$$\begin{aligned} h(z) &= d + cbz^{-1} + cAbz^{-2} + cA^2bz^{-3} + cA^3bz^{-4} + \dots \\ &= h_0 + h_1z^{-1} + h_2z^{-2} + h_3z^{-3} + h_4z^{-4} + \dots \end{aligned}$$

So we have the identities:

$$h_0 = d; \;\; h_i = cA^{i-1}b, \; i \geqslant 1. \quad (2.14)$$

These can now be arranged in an infinite Hankel matrix:

$$\mathcal{H} \doteq \begin{bmatrix} h_1 & h_1 & h_2 & h_3 & \dots \\ h_2 & h_2 & h_3 & \iddots & \dots \\ h_2 & h_3 & \iddots & \iddots & \dots \\ h_3 & \iddots & \iddots & \iddots & \\ \vdots & \vdots & \vdots & & \end{bmatrix} = \begin{bmatrix} c \\ cA \\ cA^2 \\ cA^3 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} b & Ab & A^2b & A^3b & \dots \end{bmatrix} \quad (2.15)$$

Notice that this again says that the infinite Hankel matrix can indeed be factorized into a product of two rank n matrices, where n is the dimension of the smallest possible realization $\{A, b, c, d\}$. The left and right matrices in this product are also known as the observability matrix and the controllability matrices of the system $\{A, b, c, d\}$ and we will denote them by $\mathcal{O}$ and $\mathcal{C}$, respectively. It is well known from the control literature that if a system $\{A, b, c, d\}$ is *minimal* (i.e., its transfer function can not be represented by a smaller dimensional system), then $\mathcal{O}$ and $\mathcal{C}$ have full rank n [65]. Moreover, the *finite dimensional matrices* $\mathcal{O}_n$ and $\mathcal{C}_n$:

$$\mathcal{O}_n \doteq \begin{bmatrix} c \\ cA \\ \vdots \\ cA^{n-1} \end{bmatrix}, \quad \mathcal{C}_n \doteq \begin{bmatrix} b & Ab & \dots & A^{n-1}b \end{bmatrix} \tag{2.16}$$

have rank n as well. Notice that this implies that the leading $n \times n$ hankel matrix in (2.17) is invertible, as mentioned earlier after Theorem 2.1. Also notice that any system $\{\hat{A}, \hat{b}, \hat{c}, \hat{d}\} \doteq \{T^{-1}AT, T^{-1}b, cT, d\}$ has also the same impulse response

$$h_i = cA^{i-1}b = (cT)(T^{-1}AT)^{i-1}(T^{-1}b) = \hat{c}\hat{A}^{i-1}\hat{b}.$$

The matrices $\mathcal{O}$ and $\mathcal{C}$, on the other hand, are transformed as

$$\hat{\mathcal{O}} \doteq \begin{bmatrix} \hat{c} \\ \hat{c}\hat{A} \\ \hat{c}\hat{A}^2 \\ \vdots \end{bmatrix} = \begin{bmatrix} c \\ cA \\ cA^2 \\ \vdots \end{bmatrix} T, \quad \hat{\mathcal{C}} \doteq \begin{bmatrix} \hat{b} & \hat{A}\hat{b} & \hat{A}^2\hat{b} & \dots \end{bmatrix} = T^{-1} \begin{bmatrix} b & Ab & A^2b & \dots \end{bmatrix} \tag{2.17}$$

which is easily checked by inspection. So we have:

$$\mathcal{H} = \hat{\mathcal{O}}\hat{\mathcal{C}} = \mathcal{O}T \cdot T^{-1}\mathcal{C}$$

Both the expressions on the right hand side are *rank factorizations* of the hankel matrix $\mathcal{H}$, and the above discussion also indicates that to each factorization into some observability matrix $\mathcal{O}$ and controllability matrix $\mathcal{C}$ there actually corresponds a particular realization $\{A, b, c, d\}$. We now make this explicit in the following theorem.

Theorem 2.2. Let

$$\mathcal{H} = \hat{\mathcal{O}}\hat{\mathcal{C}} = \mathcal{O}\mathcal{C}$$

be rank factorizations of $\mathcal{H}$, i.e., the matrices $\hat{\mathcal{O}}$, $\hat{\mathcal{C}}$, $\mathcal{O}$ and $\mathcal{C}$ have full rank n, then these matrix factorization are related by

$$\hat{\mathcal{O}} = \mathcal{O}T, \;\; \hat{\mathcal{C}} = T^{-1}\mathcal{C}$$

Proof. From the rank condition it follows that $\hat{\mathcal{O}}$ and $\mathcal{O}$ have a left inverse and that the matrices $\hat{\mathcal{C}}$ and $\mathcal{C}$ have a right inverse. Denoting generalized inverses with a $.^+$, we have:

$$\mathcal{O}^+\hat{\mathcal{O}} = \mathcal{C}\hat{\mathcal{C}}^+$$

Defining the resulting matrix to be equal to T, then

$$\hat{\mathcal{O}}^+\mathcal{O} = \hat{\mathcal{C}}\mathcal{C}^+$$

equals T^{-1} and the result readily follows. □

From this theorem it appears that to each rank factorization one can associate a particular realization $\{A, b, c, d\}$. The following construction shows that this is indeed the case.

Algorithm 2.2

Construct a rank factorization $\mathcal{O}_i\mathcal{C}_j$ of the $i \times j$ hankel matrix $\mathcal{H}_{i,j}$, where both i and j are assumed larger than n. Then

$$\mathcal{H}_{i,j} = \mathcal{O}_i\mathcal{C}_j = \begin{bmatrix} c \\ cA \\ \vdots \\ cA^{i-1} \end{bmatrix} . \begin{bmatrix} b & Ab & \dots & A^{j-1}b \end{bmatrix}$$

Then define the submatrices

$$\mathcal{O}_- \doteq \begin{bmatrix} c \\ cA \\ \vdots \\ cA^{i-2} \end{bmatrix}, \ \mathcal{O}_+ \doteq \begin{bmatrix} cA \\ cA^2 \\ \vdots \\ cA^{i-1} \end{bmatrix},$$

$$\mathcal{C}_- \doteq \begin{bmatrix} b & Ab & \dots & A^{j-2}b \end{bmatrix}, \ \mathcal{C}_+ \doteq \begin{bmatrix} Ab & A^2b & \dots & A^{j-1}b \end{bmatrix},$$

Since $i, j > n$ these matrices all have full rank n and we can find A from either of the following equations:

$$\mathcal{O}_-A = \mathcal{O}_+, \ \ A\mathcal{C}_- = \mathcal{C}_+$$

Both these systems are compatible and hence solvable, yielding

$$A \ = \ \mathcal{O}_-^+\mathcal{O}_+ = \mathcal{C}_+\mathcal{C}_-^+.$$

The vectors b and c are just found as the first column and row of the matrices $\mathcal{C}$, respectively $\mathcal{O}$.□

Exercise 2.1.

Prove that the realization obtained under (2.14) in fact immediately follows when using the factorization

$$\mathcal{H} = \left[\begin{array}{cccc} 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \\ \hline l_{n+1,1} & \dots & \dots & l_{n+1,n} \\ \vdots & \vdots & \vdots & \vdots \end{array}\right] . \left[\begin{array}{cccc|cc} h_1 & h_2 & \iddots & h_n & h_{n+1} & \dots \\ h_2 & \iddots & \iddots & h_{n+1} & & \dots \\ \vdots & \iddots & \iddots & \vdots & & \dots \\ h_n & h_{n+1} & \iddots & h_{2n-1} & h_{2n} & \dots \end{array}\right]$$

Hint: using the construction of Algorithm 2.2, you need only to prove that the vector $[l_{n+1,1} \ \dots \ \dots \ l_{n+1,n}]$ is in fact the vector $[-d_0 \ \dots \ \dots \ -d_{n-1}]$. □

We know of several factorizations in linear algebra and the one of the above exercise is in fact not a recommended one. It essentially amounts to "block LU" without pivoting and requires the leading principal $n \times n$ minor $\mathcal{H}_{n,n}$ of $\mathcal{H}$ to be easy to invert (i.e., a good condition number). Notice that we know this matrix *has* to be invertible from the discussion before Theorem 2.2, but this may still be delicate from a numerical point of view. Other factorizations that can be considered are LU with pivoting, QR factorizations and the Singular Value Decomposition.

Exercise 2.2.

Prove that if we choose a QR factorization of the matrix $\mathcal{H}$ into a form

$$\mathcal{H} = \left[\begin{array}{cccc} q_{1,1} & \dots & \dots & q_{1,n} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ q_{n,1} & \dots & \dots & q_{n,n} \\ \hline q_{n+1,1} & \dots & \dots & q_{n+1,n} \\ \vdots & \vdots & \vdots & \vdots \end{array}\right] \cdot \left[\begin{array}{cccc|cc} r_{1,1} & r_{1,2} & \dots & r_{1,n} & r_{1,n+1} & \dots \\ 0 & r_{2,2} & \ddots & \vdots & \vdots & \dots \\ \vdots & \ddots & \ddots & \vdots & \vdots & \dots \\ 0 & \dots & 0 & r_{n,n} & r_{n,n+1} & \dots \end{array}\right]$$

where the leading $n \times n$ matrix in R is upper triangular and invertible, then the realization $\{A, b, c, d\}$ will have A in upper Hessenberg form and b will have only the first element non-zero (this is called a controller-Hessenberg form). □

2.3 Balanced realizations

We now show that using the SVD actually leads to realizations with very particular properties for the quadruple $\{A, b, c, d\}$. The particular factorization we consider is described by the following algorithm (see [152] for a preliminary version of this).

Algorithm 2.3

Decompose the $n_1 \times n_2$ matrix $\mathcal{H}_{n_1,n_2}$ of rank n into the SVD:

$$\mathcal{H}_{n_1,n_2} = \left[\begin{array}{cccc} u_{1,1} & \dots & \dots & u_{1,n} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ u_{n,1} & \dots & \dots & u_{n,n} \\ \hline u_{n+1,1} & \dots & \dots & u_{n+1,n} \\ \vdots & & & \vdots \\ u_{n_1,1} & \dots & \dots & u_{n_1,n} \end{array}\right] \cdot \left[\begin{array}{cccc} \sigma_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_n \end{array}\right] \cdot \left[\begin{array}{cccc|c} v_{1,1} & \dots & \dots & v_{n,1} & \dots & v_{n_2,1} \\ \vdots & & & \vdots & & \vdots \\ \vdots & & & \vdots & & \vdots \\ v_{1,n} & \dots & \dots & v_{n,n} & \dots & v_{n_2,n} \end{array}\right]$$

Now define the factors $\mathcal{O}$ and $\mathcal{C}$ as follows:

$$\mathcal{O}_{n_1,n} \doteq \left[\begin{array}{cccc} u_{1,1} & \dots & \dots & u_{1,n} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ u_{n,1} & \dots & \dots & u_{n,n} \\ \hline u_{n+1,1} & \dots & \dots & u_{n+1,n} \\ \vdots & & & \vdots \\ u_{n_1,1} & \dots & \dots & u_{n_1,n} \end{array}\right] \cdot \left[\begin{array}{cccc} \sigma_1^{\frac{1}{2}} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_n^{\frac{1}{2}} \end{array}\right] \doteq U_{n_1,n}\Sigma_n^{\frac{1}{2}}$$

$$\mathcal{C}_{n_2,n} \doteq \left[\begin{array}{cccc} \sigma_1^{\frac{1}{2}} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_n^{\frac{1}{2}} \end{array}\right] \cdot \left[\begin{array}{ccc|ccc} v_{1,1} & \dots & v_{n,1} & v_{n+1,1} & \dots & v_{n_2,1} \\ \vdots & & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & & \vdots \\ v_{1,n} & \dots & v_{n,n} & v_{n+1,n} & \dots & v_{n_2,n} \end{array}\right] \doteq \Sigma_n^{\frac{1}{2}} V_{n_2,n}^H$$

Now derive the system $\{A, b, c, d\}$ from this factorization using the standard approach of Algorithm 2.2. □

The following MATLAB code actually implements this method:

```
function[a,b,c,d]=balss(h,n,n1,n2)
%
% function[a,b,c,d]=balss(h,n,n1,n2)
%
% this realizes a quadruple {a,b,c,d} of order n for the siso system
% defined by the impulse response h{0}, h{1}, ...h{N} contained
% in the column vector h=[h{0}, ..., h{N}]' where N>n1+n2-1
% The method makes a n1 x n2 Hankel matrix from h{1}
% to h{n1+n2-1} and then gets the SVD from that.  We assume
% n1 geq n2 and n leq n1,n2.
H=hankel(h(2:n1+1),h(n1+1:n1+n2));
[q,r]=qr(H);[u,s,v]=svd(r);sq=sqrt(s(1:n,1:n));
c=q(1,:)*u(:,1:n)*sq;b=sq*v(1,1:n)';d=h(1);
h1=q(1:n1-1,:)*u(:,1:n)*sq;h2=q(2:n1,:)*u(:,1:n)*sq;a=h1\h2;
```

Notice that we have assumed here that $n < n_1, n_2$ and the the order n is correct. In normal circumstances the order has to be determined and the SVD is one of the more reliable ways to determine this. Such things depend though on a user specified threshold of what ought to be considered as noise when deciding about the order of the system.

The realizations obtained by the above scheme are called *balanced realizations*. We now derive some properties of these realizations.

Diagonal Gramians. First of all it is easy to see that

$$\mathcal{O}_{n_1,n}^H \mathcal{O}_{n_1,n} = \Sigma_n = \mathcal{C}_{n_2,n}\mathcal{C}_{n_2,n}^H \tag{2.18}$$

This follows automatically from the expressions for these matrices and the fact that

$$U_{n_1,n}^H U_{n_1,n} = I_n = V_{n_2,n}^H V_{n_2,n}.$$

Moreover from the relation with the observability and controllability matrices we also have that

$$\mathcal{O}_{n_1,n}^H \mathcal{O}_{n_1,n} = \sum_{i=0}^{n_1-1} {A^i}^H c^H c\, A^i \doteq G_o(0, n_1 - 1), \tag{2.19}$$

$$\mathcal{C}_{n_2,n} \mathcal{C}_{n_2,n}^H = \sum_{i=0}^{n_2-1} A^i b\, b^H {A^i}^H \doteq G_c(0, n_2 - 1), \tag{2.20}$$

where $G_o(0, n_1 - 1)$ and $G_c(0, n_2 - 1)$ are the observability and controllability Gramians over the finite intervals $(0, n_1 - 1)$ and $(0, n_2 - 1)$, respectively. So, balanced realizations are realizations for which the (finite horizon) Gramians are *equal and diagonal*. Such realizations are well known for the case of *infinite horizon* Gramians, in which case some additional properties are obtained. The Gramians G_o and G_c are then also the solutions to the following Lyapunov equations:

$$A^H G_o A + c^H c = G_o, \quad A G_c A^H + b\, b^H = G_c$$

Sign symmetry. From these relations one proves [108] that the matrices of the quadruple $\{A, b, c, d\}$ actually have some nice symmetry properties. Arranging the quadruple into the matrix

$$\mathcal{E} \doteq \left[\begin{array}{c|c} A & b \\ \hline c & d \end{array}\right]$$

we have that this matrix is symmetric up to some sign changes, i.e., there exist a diagonal matrix S of ± 1 such that

$$\mathcal{E} S = S \mathcal{E}^H$$

Such property is not met when considering the finite horizon Gramians instead, but the following exercise indicates that we should not be very far from it.

Exercise 2.3.

Prove that the finite horizon Gramians are the solution to the following equations:

$$A^H G_o(0, n_1 - 1) A + c^H c - {A^{n_1}}^H c^H c A^{n_1} = G_o(0, n_1 - 1)$$

$$A G_c(0, n_2 - 1) A^H + b\, b^H - A^{n_2} b\, b^H {A^{n_2}}^H = G_c(0, n_2 - 1)$$

□

Clearly if A is stable and n_1 and n_2 are sufficiently large the finite horizon Gramians will be close to the infinite horizon ones, and the same sign symmetry ought to be observed. In practice, this occurs if one uses a Hankel matrix $\mathcal{H}_{n_1,n_2}$ of which the elements have started to "die out", i.e., which is a reasonable approximation of the infinite horizon Hankel matrix.

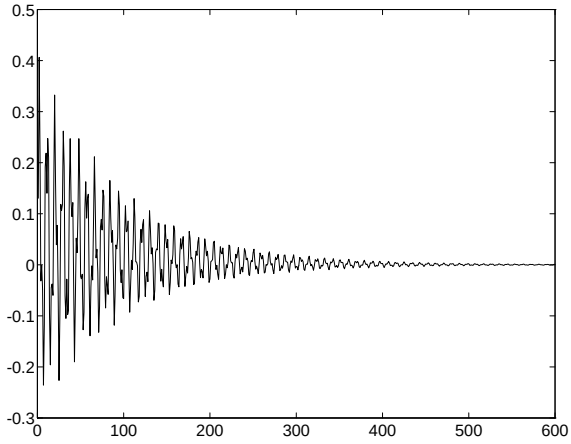


Figure 2.3: Impulse response of a 5-th order lightly damped system.

Below we show the elements of the realization of a 5-th order system identified from the first 100 samples of the impulse response given in Figure 2.3.

The matrix resulting from this is:

$$\mathcal{E} \doteq \left[\begin{array}{c|c} A & b \\ \hline c & d \end{array}\right] = \left[\begin{array}{rrrrr|r} 0.7787 & 0.6258 & -0.0148 & -0.0207 & 0.0157 & 0.1553 \\ -0.6258 & 0.7569 & -0.0206 & -0.0273 & 0.0219 & 0.4308 \\ 0.0148 & -0.0206 & -0.2016 & 0.9682 & -0.0695 & 0.2607 \\ -0.0207 & 0.0273 & -0.9682 & -0.1834 & -0.0740 & -0.2420 \\ -0.0157 & 0.0219 & -0.0695 & 0.0740 & 0.9871 & -0.2081 \\ \hline -0.1553 & 0.4308 & 0.2607 & 0.2420 & -0.2081 & 0.1304 \end{array}\right]$$

which clearly has sign symmetry. When executing in MATLAB the command

```
s=diag([-1 1 1 -1 1 1]); norm(s*e-e'*s)/norm(e)
```

one gets the response $1.7212e-14$ which shows how close we are to a sign symmetric matrix, although we have hardly waited for the response to die out (we used only 100 samples of the impulse response).

Computation of poles and zeros. One possible preliminary step in many eigenvalues solvers for an arbitrary matrix M is to perform first a diagonal scaling to insure that rows and columns have equal norm. This avoids a possible "imbalance" in the elements of the matrix M and improves the accuracy of the computed eigenvalues. For the above matrix A this step is superfluous since sign symmetry implies that row i and column i have equal norms already. So the computation of the poles of the transfer function – i.e., the zeros of $d(z)$ – from the eigenvalues of A ought to have small sensitivity. The zeros of the transfer function – i.e., the zeros of the $n(z)$ – are in fact the eigenvalues of $A_z \doteq A - bd^{-1}c$ which for the above model is again sign symmetric. So the balanced realization has the nice property that the computation of poles and zeros of the transfer function

should have good sensitivity properties. For the above model the zero matrix is:

$$A_z = \begin{bmatrix} 0.9637 & 0.1127 & -0.3253 & -0.3089 & 0.2635 \\ -0.1127 & -0.6663 & -0.8819 & -0.8268 & 0.7094 \\ 0.3253 & -0.8819 & -0.7228 & 0.4844 & 0.3465 \\ -0.3089 & 0.8268 & -0.4844 & 0.2657 & -0.4602 \\ -0.2635 & 0.7094 & 0.3465 & 0.4602 & 0.6550 \end{bmatrix}$$

and the poles and zeros are: $-0.1940 \pm 0.9708i, 0.7679 \pm 0.6249i, 0.9900$ and $-1.7323, 1.0015 \pm 0.3022i, 0.1123 \pm 1.0242i$, which all turn out to have reasonably low sensitivity.

Low sensitivity to noise propagation. The paper that actually introduced balanced realizations [99] did so for minimizing the sensitivity of the models to various types of noise propagation. Without going into details, we give an indication here of this property. Let us write again the evolution of the state space equation as

$$\begin{bmatrix} x_{k+1} \\ y_k \end{bmatrix} = \left[\begin{array}{c|c} A & b \\ \hline c & d \end{array}\right] \cdot \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

then errors in the inputs, state and output propagate in the same manner:

$$\begin{bmatrix} \delta x_{k+1} \\ \delta y_k \end{bmatrix} = \left[\begin{array}{c|c} A & b \\ \hline c & d \end{array}\right] \cdot \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}$$

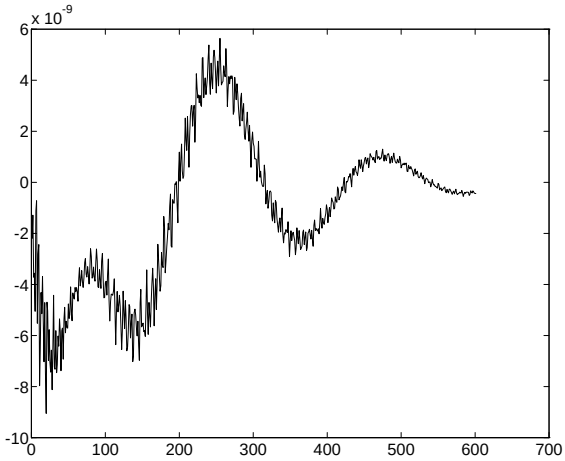


Figure 2.4: Impulse response of a 5-th order lightly damped system.

and the size of this propagation will be diminished if the norm of the evolution operator $\mathcal{E}$ can be kept as small as possible. It turns out that for balanced realizations, this norm is nearly minimized over all possible realization. E.g., taking again the perturbed data of Example 2.1 yields a balanced realization with an evolution matrix of norm close to 1 whereas the companion form (2.14) for this model has norm close to 10. Partly as a result of this the reconstructed impulse response from the balanced realization gives errors that are about a factor 10 smaller as well (see Figure 2.4).

2.4 Padé algorithm

If we do not know from beforehand what the order of the system will be one can take the viewpoint to try to identify a family of realizations of growing orders 1, 2, 3, ... This would amount to finding the following polynomials from the following Hankel matrices:

$$\left[\ h_1\ \right], \left[\ h_2\ \right] \quad \Rightarrow \quad \frac{n_{(1)}(z)}{d_{(1)}(z)} = \frac{n_0}{z + d_0}$$

$$\begin{bmatrix} h_1 & h_2 \\ h_2 & h_3 \end{bmatrix}, \begin{bmatrix} h_3 \\ h_4 \end{bmatrix} \quad \Rightarrow \quad \frac{n_{(2)}(z)}{d_{(2)}(z)} = \frac{n_1 z + n_0}{z^2 + d_1 z + d_0}$$

$$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_2 & h_3 & h_4 \\ h_3 & h_4 & h_5 \end{bmatrix}, \begin{bmatrix} h_4 \\ h_5 \\ h_6 \end{bmatrix} \quad \Rightarrow \quad \frac{n_{(3)}(z)}{d_{(3)}(z)} = \frac{n_2 z^2 + n_1 z + n_0}{z^3 + d_2 z^2 + d_1 z + d_0}$$

and so on, where degree.$n_{(i)}(z) <$ degree.$d_{(i)}(z)$ because we do not use the parameter h_0 in this process. Notice that there are as many parameters on each side of the above arrows, namely $2i$ at stage i of the above process. So we can match the functions $\frac{n_{(i)}(z)}{d_{(i)}(z)}$ to the given parameters $h_1, \ldots, h_{2i}$ in each stage . In fact, given the original transfer function (without h_0 again):

$$\frac{n(z)}{d(z)} = h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + \ldots \tag{2.21}$$

the above rational functions $\frac{n_{(i)}(z)}{d_{(i)}(z)}$ match the Taylor series (2.23) in z^{-1} in its first $2i$ terms. It will be shown below that in fact what is now known as the Padé algorithm solves for all of these *partial realizations* from degree 1 to n in only $\mathrm{O}(n^2)$ operations !

Some history.

This problem is actually known as the Padé approximation problem [15]. The relations of this problem to that of Hankel matrices was already observed in the late 1800's by people like H. Hankel (1862), G. F. Frobenius (1886) and T. J. Stieltjes (1894) (see [14] for more history on this). In the system theory literature the relation of partial realizations to Hankel matrices were rediscovered – and extended to the multi-input multi-output case – by Ho-Kalman [56], Youla-Tissi [151] and Silverman[118]. The $\mathrm{O}(n^2)$ algorithms for these partial realizations were rediscovered by Massey and Berlekamp [85] in the context of convolutional codes, and later on by Rissanen [113] in the context of identification. Several years later, de Jong [24] showed that these algorithms all suffer from numerical instability.

Example 2.3.

Consider the Taylor series

$$h(z) = 1z^{-1} + 2z^{-2} + 3z^{-3} + 3z^{-4} + 1z^{-5} - 4z^{-6} - 8z^{-7} + \ldots$$

then we find the partial realizations:

$$\begin{bmatrix} 1 \end{bmatrix}, \begin{bmatrix} 2 \end{bmatrix} \quad \Rightarrow \quad \frac{n_{(1)}(z)}{d_{(1)}(z)} = \frac{1}{z-2}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \begin{bmatrix} 3 \\ 3 \end{bmatrix} \quad \Rightarrow \quad \frac{n_{(2)}(z)}{d_{(2)}(z)} = \frac{z-1}{z^2-3z+3}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 3 \\ 3 & 3 & 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \\ -4 \end{bmatrix} \quad \Rightarrow \quad \frac{n_{(3)}(z)}{d_{(3)}(z)} = \frac{z^2}{z^3-2z^2+z+1}$$

Exercise 2.4.

Check with MATLAB that the above i-th order partial realizations fit the Taylor series in the first $2i$ coefficients. Also check the poles and zeros of the partial realizations. Repeat the same for the series

$$h(z) = 1/2z^{-1} + 1/2z^{-2} + 3/8z^{-3} + 3/16z^{-4} + 1/32z^{-5} - 1/16z^{-6} - 1/16z^{-7} + \dots$$

obtained from replacing z by $2z$. Results ought to be related to the previous ones. Use the identification algorithm 2.1 with the minimum amount of data for finding the partial realizations. □

We now derive a simplified version of the Padé algorithm to show how the Hankel structure of the underlying matrices leads to an $O(n^2)$ algorithm. We first need the following theorem about the existence of LU decompositions.

Theorem 2.3. A $n \times n$ invertible matrix M has a LU decomposition (without pivoting on rows or columns)

$$M = L \cdot U$$

iff all its leading principal minors are invertible.

Proof. The *only if* part is simple. Since the LU decomposition exists, L and U must be invertible, otherwise M would be singular. But then the leading principal submatrices $L(1:i, 1:i)$ and $U(1:i, 1:i)$ of L and U are also invertible, and since

$$M(1:i, 1:i) = L(1:i, 1:i)U(1:i, 1:i)$$

the result follows.
For the *if* part we construct L and U as follows. Since $M(1:i, 1:i)$ is non-singular for $i = 1, \dots, n-1$, we can solve for a vector $x_{(i+1)}$ in

$$M(1:i, 1:i)x_{(i+1)} = -M(1:i, i+1), \text{ or also } M(1:i, 1:i+1) \begin{bmatrix} x_{1,i+1} \\ \vdots \\ x_{i,i+1} \\ 1 \end{bmatrix} = 0. \qquad (2.22)$$

Arranging this into an array we find

$$M \cdot \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,n} \\ 0 & 1 & \dots & x_{2,n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 \end{bmatrix} = \begin{bmatrix} \times & 0 & \dots & 0 \\ \times & \times & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \times & \dots & \times & \times \end{bmatrix}$$

Since the two matrices on the left are invertible, so is the lower triangular matrix on the right. From the above equation we clearly see that we have in fact constructed the factors L (on the right) and U as the inverse of the matrix with the $x_{i,j}$ elements. □

Corollary 2.1. A $n \times n$ matrix M has a LU decomposition (without pivoting on rows or columns)

$$M = L \cdot U$$

iff its leading principal minors of order 1 to $n-1$ are invertible. In this decomposition one can normalize either L or U to have 1's on the diagonal. Moreover, if M is symmetric, L and U are equal up to a diagonal scaling S:

$$M = M^T = LSL^T.$$

In this latter case one can always normalize L to be unit upper triangular.

Proof. We refer to [46] for this extension of the above theorem. □

Using these results we have the following theorem.

Theorem 2.4. Let the Hankel matrix $\mathcal{H}_{n+1,n+1}$ have non-singular leading principal minors of order 1 to n, then:

$$\mathcal{H}_{n+1,n+1} = LSL^T$$

with L chosen to be unit lower triangular. Defining the unit lower triangular matrix $X^T = L^{-1}$, we also have:

$$X^T \mathcal{H}_{n+1,n+1} X = S \tag{2.23}$$

and column $i+1$ of X contains then the coefficients of the polynomials $d_{(i)}(z)$ of the partial realizations of the corresponding impulse response.

Proof. Using a similar argument as in (2.24) we find that column $i+1$ of X has its $i+1$ leading coefficients satisfying:

$$\mathcal{H}(1:i, 1:i+1) \begin{bmatrix} x_{1,i+1} \\ \vdots \\ x_{i,i+1} \\ 1 \end{bmatrix} = 0. \tag{2.24}$$

This is precisely the equation for the denominator $d_{(i)}(z)$ of the partial realization built on the $i \times i+1$ Hankel matrix $\mathcal{H}(1:i, 1:i+1)$ (see Algorithm 2.1). □

Example 2.4.

One easily checks the following identity in MATLAB:

$$\begin{bmatrix} 1 & 2 & 3 & 3 \\ 2 & 3 & 3 & 1 \\ 3 & 3 & 1 & -4 \\ 3 & 1 & -4 & -8 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 3 & 3 & 1 & \\ 3 & 5 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & -1 & & \\ & & 1 & \\ & & & 4 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 3 \\ & 1 & 3 & 5 \\ & & 1 & 2 \\ & & & 1 \end{bmatrix} = L^T S L$$

from which the polynomials in Example 2.4 are recovered via

$$L^{-T} = \begin{bmatrix} 1 & 2 & 3 & 3 \\ & 1 & 3 & 5 \\ & & 1 & 2 \\ & & & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -2 & 3 & 1 \\ & 1 & -3 & 1 \\ & & 1 & -2 \\ & & & 1 \end{bmatrix}^{-1} = X$$

□

So far we showed that the LSL^T decomposition of $\mathcal{H}$, or rather the inverse X of L^T, gives all the partial realizations but such a decomposition typically requires $\mathrm{O}(n^3)$ flops. We now derive a *fast algorithm* that only requires $\mathrm{O}(n^2)$ flops and is in fact a rewrite of the Padé algorithm. The original Padé algorithm involves polynomials but we prefer to use the decomposition (2.25) instead. Suppose we have this factorization up to step i, i.e.,

$$X(1:i,1:i)\mathcal{H}(1:i,1:i)X(1:i,1:i)^T = S(1:i,1:i)$$

then we have to find the next column of X in $\mathrm{O}(i)$ flops in order to wind up with an total of $\mathrm{O}(n^2)$ flops after n steps ! So the next columns must be very simple to compute indeed. Let us try for column $I = 1$ of X the *shifted* version of column i, i.e., the polynomial $\hat{d}_{(i)}(z)$ is approximated by the shifted polynomial $zd_{(i-1)}(z)$. In other words:

$$\hat{X}(1:i+1,1:i+1) = \left[\begin{array}{c|c} & 0 \\ X(1:i,1:i) & x_{1,i} \\ & \vdots \\ \hline & x_{i-1,i} \\ & 1 \end{array}\right]$$

Then it follows that

$$\hat{X}(1:i+1,1:i+1)\mathcal{H}(1:i+1,1:i+1)\hat{x}(1:i+1,1:i+1) = \left[\begin{array}{cccc|c} s_{1,1} & 0 & \dots & 0 & 0 \\ 0 & s_{2,2} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & a_i \\ 0 & \dots & 0 & s_{i,i} & b_i \\ \hline 0 & .. & 0 & a_i & b_i \end{array}\right]$$

or a matrix that is almost diagonal, except for the elements a_i and b_i. In order to prove this we look at the last columns of $\mathcal{H}(1:i+1,1:i+1)\hat{X}(1:i+1,1:i+1)$ and $\mathcal{H}(1:i,1:i)X(1:i,1:i)$. Because of the last columns of $\hat{X}(1:i+1,1:i+1)$ and $X(1:i,1:i)$ are shifted with respect to each other and since the Hankel matrix has shifted rows as well, it follows that only the last

3 elements of that column are nonzero. Multiplying this now with $\hat{X}^T(1:i+1,1:i+1)$ on the left hand side does not change the zero pattern of that column, which proves a_i and b_i are the only nonzero off-diagonal elements. In fact one checks that $a_i = s_{i,i}$. So starting from the right hand side of this result, one needs only two additional elementary eliminations to reduce this to a diagonal matrix:

$$\left[\begin{array}{cccc|c} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & 0 \\ \hline 0 & .. & 0 & \alpha_i & \beta_i & 1 \end{array}\right] \left[\begin{array}{cccc|c} s_{1,1} & 0 & \dots & 0 & 0 \\ 0 & s_{2,2} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & a_i \\ 0 & \dots & 0 & s_{i,i} & b_i \\ \hline 0 & .. & 0 & a_i & b_i & c_i \end{array}\right] \left[\begin{array}{cccc|c} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \alpha_i \\ 0 & \dots & 0 & 1 & \beta_i \\ \hline 0 & \dots & \dots & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{cccc|c} s_{1,1} & 0 & \dots & 0 & 0 \\ 0 & s_{2,2} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & s_{i,i} & 0 \\ \hline 0 & \dots & \dots & 0 & s_{i+1,i+1} \end{array}\right]$$

where $\alpha_i = -a_i/s_{i-1,i-1}$ and $\beta_i = -b_i/s_{i,i}$. From this it finally follows that the last column of the correct $X(1:i+1,1:i+1)$ is a linear combination of the last three columns of $\hat{X}$:

$$\hat{X}(1:i+1, i-1:i+1) \begin{bmatrix} \alpha_i \\ \beta_i \\ 1 \end{bmatrix}.$$

In terms of polynomials this can also be written as:

$$d_{(i)}(z) = (z+\beta_i) d_{(i-1)}(z) + \alpha_i d_{(i-2)}(z),$$

where we have used $\hat{d}_{(i+1)}(z) = z d_{(i)}(z)$. This is the well known three term recurrence of the polynomials in the Padé table, which is also connected with orthogonal polynomials over the real line [50].

Example 2.5.

Consider again Example 2.4 and the partial decomposition

$$\begin{bmatrix} 1 & & \\ -2 & 1 & \\ 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 3 \\ 3 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 3 \\ & 1 & -3 \\ & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & -1 & \\ & & 1 \end{bmatrix}$$

then the larger system with shifted column/row yields:

$$\begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ 3 & -3 & 1 & \\ 0 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 3 \\ 2 & 3 & 3 & 1 \\ 3 & 3 & 1 & -4 \\ 3 & 1 & -4 & -8 \end{bmatrix} \begin{bmatrix} 1 & -2 & 3 & 0 \\ & 1 & -3 & 3 \\ & & 1 & -3 \\ & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & -1 & & 1 \\ & & 1 & -1 \\ & 1 & -1 & 4 \end{bmatrix}$$

From this we find $\alpha_3 = \beta_3 = 1$ and

$$d_{(3)}(z) = (z+1)d_{(2)}(z) + d_{(1)}(z) = z^3 - 2z^2 + z + 1$$

□

The complexity of the i-th update of the algorithm is $4i$: two inner products for computing b_i and c_i ($a_i = s_{i-1,i-1}$ is already known), and two scalar-vector multiplies in (2.27). Summing this for $i = 1, \ldots, n$ then yields $2n^2$ for the total complexity of the algorithm. We have only described here the "regular form" of the Padé algorithm since we assumed non-singularity of all minors as needed by the algorithm. There exist many modifications of the above algorithm which are still $O(n^2)$ and hence exploit the Hankel structure in the case that singular minors are encountered. Since it is not our purpose here to enter in such details, we refer the reader to [14, 113, 24] for this.

Numerical stability. This very performant algorithm has a major drawback: it has been proven to be numerically unstable [24]. This is in a sense not surprising since the algorithm amounts to Gaussian elimination without pivoting. Example 1.3 we gave in Chapter1 to illustrate the numerical instability of Gaussian elimination without pivoting, is in fact a 2×2 Hankel matrix with the Padé algorithm applied to it. Applying pivoting to the Hankel matrix would destroy the shift structure of the matrix which then results again in a $O(n^3)$ algorithm. We have thus sacrificed stability for a lower complexity. Yet, recent papers [17], [37] have shown that one can cleverly mix both approaches to yield an algorithm that is $O(n^2)$ "in most cases" and has yet good numerical properties. Numerical stability in a strict sense has not been proved yet, but the modified algorithm tries to avoid small pivots, which is the main reason of instability.

Connection to state-space realizations. Since the Padé algorithm produces a LU factorization of the Hankel matrix $\mathcal{H}$ it will eventually detect its rank. If the order of the underlying system is n then the $(n+1) \times (n+1)$ Hankel matrix must be singular and this will result in a zero diagonal element $s_{n+1,n+1}$. We thus have a rank factorization of $\mathcal{H}_{n+1,n+1}$ and we may wonder what realization corresponds to this. This is explained in the following theorem.

Theorem 2.5. Let the Hankel matrix $\mathcal{H}_{n+1,n+1}$ corresponding to an n-th order system have non-singular leading principal minors of order 1 to n, then $\mathcal{H}_{n+1,n+1}$ has a rank factorization

$$\mathcal{H}_{n+1,n+1} = L_{n+1,n} \cdot U_{n,n+1} = \begin{bmatrix} l_{1,1} & 0 & \ldots & 0 \\ l_{2,1} & l_{2,2} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ l_{n,1} & \ldots & \ldots & l_{n,n} \\ l_{n+1,1} & \ldots & \ldots & l_{n+1,n} \end{bmatrix} \cdot \begin{bmatrix} u_{1,1} & u_{1,2} & \ldots & u_{1,n} & u_{1,n+1} \\ 0 & u_{2,2} & & \vdots & \vdots \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \ldots & 0 & u_{n,n} & u_{n,n+1} \end{bmatrix}$$

and the corresponding realization $\{A, b, c, d\}$ is such that

$$\left[\begin{array}{c|c} d & c \\ \hline b & A \end{array}\right] = \left[\begin{array}{c|cccc} h_0 & x_1 & 0 & \ldots & 0 \\ \hline y_1 & z_1 & x_2 & \ddots & \vdots \\ 0 & y_2 & z_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & x_n \\ 0 & \ldots & 0 & y_n & z_n \end{array}\right]$$

is tridiagonal.

Proof. The result is trivial for $d = h_0$ as well as for b and c, since these are the first column of U and first row of L, respectively (i.e., $x_1 = l_{1,1}$ and $z_1 = u_{1,1}$). For the matrix A, it follows from Algorithm 2.2 that A satisfies both

$$L(1:n,1:n)^{-1}L(2:n+1,1:n) = A = U(1:n,2:n+1)U(1:n,1:n)^{-1}.$$

From the left equality it follows that A must be lower Hessenberg and from the right equality it follows that A must be upper Hessenberg. Together this shows that A must be tridiagonal. □

Corollary 2.2. If in the above LU factorization L is chosen to have 1's on diagonal then all $x_i = 1$; if U is chosen to have 1's on diagonal then all $y_i = 1$. □

This corollary shows that there are only $2n + 1$ "parameters" in $\{A, b, c, d\}$, just as in a polynomial pair $n(z)/d(z)$. In fact one shows that once e.g., $x_i = 1$, the other parameters z_i and y_i are closely related to the recurrence parameters α_i and β_i for the polynomials $d_{(i)}(z)$ of (27). The dual result holds of course when normalizing $y_i = 1$.

Comparison of SISO identification methods via the impulse response. In the previous 4 sections we saw several techniques for identifying a SISO system from its impulse response (also called the Markov parameters of the system) $\{h_i, i = 0,\ 1,\ 2, \ldots\}$. Ordered by increasing complexity they are:

1. the LU decomposition via the Padé algorithm. This requires only $2n^2$ flops but is an unstable algorithm. Stabilized algorithms with a similar complexity are being proposed these days.

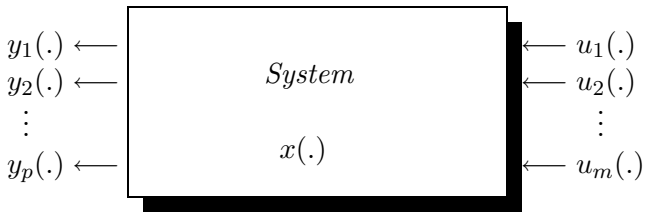
2. the PLU decomposition with pivoting P. A numerically stable and a reasonable complexity: $2/3n^3$.

3. the QR decomposition on a $n_1 \times n + 1$ matrix. Numerically stable and a complexity of $2n_1^2(n_1 - n/3)$. This algorithm usually has better sensitivity properties than LU and PLU because more data are involved in the identification.

4. the SVD on a $n_1 \times n_2$ matrix ($n_1 \leqslant n_2$). The algorithm is numerically stable and has complexity $2n_1n_2^2 + 10n_1^3$ when using a preliminary QR decomposition for the SVD. This has the most reliable order estimation of the system and has the additional advantage to give balanced realizations.

2.5 Multi-input multi-output impulse response

Here we look at how to extend the SISO algorithms to systems with several inputs and outputs, i.e., we assume a system of the form



where the input vector $u(.)$ is thus m-dimensional and the output vector is p-dimensional. For this system one could of course always identify the SISO systems $h_{i,j}(z) \doteq n_{i,j}(z)/d_{i,j}(z)$ corresponding to the impulse response of input $u_j(.)$ to output $y_i(.)$. The global transfer function between the input vector $u(.)$ and output vector $y(.)$ would then be the transfer matrix

$$H_{m,p}(z) \doteq \begin{bmatrix} \frac{n_{1,1}(z)}{d_{1,1}(z)} & \cdots & \frac{n_{1,m}(z)}{d_{1,m}(z)} \\ \vdots & & \vdots \\ \frac{n_{p,1}(z)}{d_{p,1}(z)} & \cdots & \frac{n_{p,m}(z)}{d_{p,m}(z)} \end{bmatrix} \tag{2.25}$$

But this approach has a major disadvantage. If the multivariable system has a transfer function $H_{p,m}(z)$ of degree n, then it is very likely that each scalar transfer function $h_{i,j}(z)$ has degree n as well. This is e.g., the case when all the dynamics of the system are present in each input/output pair. As a consequence of this, all polynomials $d_{i,j}(z)$, if normalized to be monic, ought to be equal. But under rounding errors or any other noise source, we can hardly expect this to happen. The identified transfer function $H_{p,m}(z)$ would then have a resulting degree that is likely to be mnp, i.e., mp times too large. Of course part of this would be detected by model reduction techniques applied to this composite model, but this would be an unnecessary detour.

Since the correct approach is to identify directly the multivariable system $H(z)$, we first have to define what is meant by the impulse response of such a system. Denote the ith unit vector by e_i and consider the response of the input sequence $u_{(j)}(.) = \{e_j,\ 0,\ 0,\ 0, \ldots\}$, i.e., all components of $u_{(j)}(.)$ are zero except the j-th one which is an impulse. Denote the corresponding output vector sequence by $y_{(j)}(.) = \{h_0^{(j)},\ h_1^{(j)},\ h_2^{(j)}, \ldots\}$. Now assemble the p-vectors $h_0^{(j)}$ into the matrix H_0, the p-vectors $h_1^{(j)}$ into the matrix H_1, etc. Then we call the matrix sequence $\{H_0,\ H_1,\ H_2, \ldots\}$ the impulse response of the system $H(z)$. One checks that the (i,j) elements of this matrix sequence is in fact the impulse response of the j-th component of the vector $u(.)$ to the i-th component of the vector $y(.)$, provided all other components of the input vector $u(.)$ are kept zero. As a result of this we have again the relation that the matrices H_i are the coefficients of the Taylor expansion of $H(z)$ around z^{-1} :

$$H(z) = H_0 + H_1 z^{-1} + H_2 z^{-2} + H_3 z^{-3} + \ldots$$

If one wants now to identify the system $H(z)$ from this expansion then one can try to extend the SISO techniques to this case. Early attempts [27] to extend the polynomial identification techniques lead to techniques where several nested rank checks have to be performed in order to reconstruct particular columns of polynomial matrix pairs $D(z)$ and $N(z)$. These methods will not be described here because of the simple reason that their numerical reliability is very poor. Not only is this approach unstable, but the algorithm is also very complex.

A much more reliable approach is that of identifying directly a state space model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k, \end{aligned} \tag{2.26}$$

where the constant matrices A, B, C and D are of dimensions $n \times n$, $n \times p$, $m \times n$, and $m \times p$, respectively. Using again the Neumann expansion of $(zI - A)^{-1}$ we obtain very similar identities to the SISO case:

$$\begin{aligned} H(z) &= D + CBz^{-1} + CABz^{-2} + CA^2Bz^{-3} + CA^3Bz^{-4} + \ldots \\ &= H_0 + H_1 z^{-1} + H_2 z^{-2} + H_3 z^{-3} + H_4 z^{-4} + \ldots \end{aligned}$$

So we have the identities:

$$H_0 = D; \;\; H_i = CA^{i-1}B, \; i \geqslant 1, \tag{2.27}$$

which can be arranged in an infinite Hankel matrix:

$$\mathcal{H} \doteq \begin{bmatrix} H_1 & H_1 & H_2 & H_3 & \dots \\ H_2 & H_2 & H_3 & \iddots & \dots \\ H_2 & H_3 & \iddots & \iddots & \dots \\ H_3 & \iddots & \iddots & \iddots & \\ \vdots & \vdots & \vdots & & \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} B & AB & A^2B & A^3B & \dots \end{bmatrix}. \tag{2.28}$$

As before (see Algorithm 2.2) a rank factorization $\mathcal{O}_i\mathcal{C}_j$ of a finite *block Hankel* matrix $\mathcal{H}_{i,j}$, where both i and j are assumed large enough, will yield an observability and controllability matrix from which the matrices A, B and C can be recovered. The number of blocks i and j ought to be chosen such that in

$$\mathcal{H}_{i,j} = \mathcal{O}_i\mathcal{C}_j = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{i-1} \end{bmatrix} . \begin{bmatrix} B & AB & \dots & A^{j-1}B \end{bmatrix}$$

both $\mathcal{O}_i$ and $\mathcal{C}_j$ have full rank n, assuming of course the underlying model is minimal. It is well known from state space theory that the minimum for i is the largest observability index and the minimum for j is the largest controllability index, and both of these are bounded by n [65]. This of course does not prevent to take larger values of i or j, since then the rank of these matrices does not change anymore. From this rank factorization one then defines the submatrices

$$\mathcal{O}_- \doteq \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{i-2} \end{bmatrix}, \; \mathcal{O}_+ \doteq \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{i-1} \end{bmatrix},$$

$$\mathcal{C}_- \doteq \begin{bmatrix} B & AB & \dots & A^{j-2}B \end{bmatrix}, \mathcal{C}_+ \doteq \begin{bmatrix} AB & A^2B & \dots & A^{j-1}B \end{bmatrix}.$$

Since i, j are large enough, these matrices all have full rank n and we can find A from either of the following equations:

$$\mathcal{O}_- A = \mathcal{O}_+, \;\; A\mathcal{C}_- = \mathcal{C}_+$$

Both these systems are compatible and hence solvable, yielding

$$A = \mathcal{O}_-^+\mathcal{O}_+ = \mathcal{C}_+\mathcal{C}_-^+.$$

The matrices B and C are found again as the first block column and block row of the matrices $\mathcal{C}_j$, respectively $\mathcal{O}_i$.

One checks readily that when using the singular value decomposition for this rank factorization, one obtains again Gramians $G_o(0, i-1)$ and $G_c(0, j-1)$ over the finite intervals $(0, i-1)$ and $(0, j-1)$:

$$\mathcal{O}_{i,n}^H \mathcal{O}_{i,n} = \sum_{i=0}^{i-1} {A^i}^H C^H C\, A^i \doteq G_o(0, i-1), \tag{2.29}$$

$$\mathcal{C}_{j,n} \mathcal{C}_{j,n}^H = \sum_{i=0}^{j-1} A^i B\, B^H {A^i}^H \doteq G_c(0, j-1), \tag{2.30}$$

that are equal and diagonal, since

$$\mathcal{O}_{j,n}^H \mathcal{O}_{i,n} = \Sigma_n = \mathcal{C}_{j,n} \mathcal{C}_{j,n}^H. \tag{2.31}$$

So, balanced realizations in the MIMO case are realizations for which the (finite horizon) Gramians are *equal and diagonal*.

In the SISO case we had nice additional properties which only hold to a certain extent in the MIMO case. We still have that the *infinite horizon* Gramians are also the solutions to Lyapunov equations:

$$A^H G_o A + C^H C = G_o, \;\; A G_c A^H + B\, B^H = G_c$$

but this does not imply any sign symmetry anymore for the realization. Yet, it is still true that these realizations have good numerical properties in terms of eigenvalue sensitivity (for computing poles and zeros) and in terms of round-off noise propagation (for simulation purposes). Roughly speaking, these realization have very similar numerical advantages in the MIMO case as what we showed for the SISO case.

The sensitivity of the balanced realization algorithm depends directly on the sensitivity of the singular value decomposition. The matrices B and C are just submatrices of this rank factorization and the dependency on the factorization is thus obvious. For the matrix A one essentially solves

$$\mathcal{O}_- A = \mathcal{O}_+$$

in each identification method, but the coordinate system used for balanced realizations typically gives a low norm solution $\|A\|$, which then also results in a lower perturbation $\|\delta A\|$ for the same sensitivity. One notices this typically in the computation of the eigenvalues of A, which are the poles of the system and are thus invariant for each identification. The best precision of these poles are usually obtained from the balanced realization.

2.6 Input-Output Pairs

In many circumstances we do not have access to the impulse response of a system. A typical examples is a system which is operational and for which it is too costly to affect the input (let alone apply an impulse as input) in order to measure the corresponding output. In some other examples it is physically impossible to affect the input and one can only observe (or measure) inputs and outputs.

So the problem of determining a linear time invariant finite dimensional system

$$\begin{aligned} H(z) &= D^{-1}(z)N(z) \qquad (2.32) \\ &\text{or} \\ H(z) &= C(zI - A)^{-1}B + D \qquad (2.33) \end{aligned}$$

from input output measurements is certainly an important one. A first problem that immediately occurs here is that some input output pairs generated by an irreducible (or minimal) system (2.32) or (2.33) of a certain order, could as well have been generated by a lower order one. The input-output pair then essentially behaves as that of a lower order system. So we need to assume that this does not happen. The standard assumption made in this context is that of persistence of excitation, which we define loosely as follows.

Definition 2.1. An input-output pair $\{u_i\}$, $\{y_i\}$ is said to be persistently exciting with respect to the underlying model (2.32)-(2.33) if the same pair can not be generated by a lower order model. □

Depending on the context in which persistence of excitation is being used, different definitions occur, but they amount to the same idea. The dynamical interpretation of this is that when a pair $\{u_i\}$ $\{y_i\}$ is not persistently exciting, then not all the characteristic modes of the system are being "excited" or triggered. We will see below that impulse responses are always persistently exciting, which is why this concept was not introduced earlier.

We start by considering the single input single output case:

$$d(z)y(z) = n(z)u(z) \qquad (2.34)$$

or:

$$(d_n z^n + \cdots + d_1 z + d_0)y_i = (n_m z^m + \cdots + n_1 z + n_0)u_i \qquad (2.35)$$

where we assume the input to start at some time $t = 0$. We can rewrite this into a semi-infinite matrix vector equation

$$\begin{bmatrix} 0 & \cdots & 0 & y_0 \\ \vdots & \iddots & \iddots & y_1 \\ 0 & \iddots & \iddots & y_2 \\ y_0 & \iddots & \iddots & \vdots \\ y_1 & \iddots & \iddots & \\ y_2 & \iddots & & \\ \vdots & & & \end{bmatrix} \begin{bmatrix} d_0 \\ \vdots \\ \vdots \\ d_n \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 & u_0 \\ \vdots & \iddots & \iddots & u_1 \\ 0 & \iddots & \iddots & u_2 \\ u_0 & \iddots & \iddots & \vdots \\ u_1 & \iddots & \iddots & \\ u_2 & \iddots & & \\ \vdots & & & \end{bmatrix} \begin{bmatrix} n_0 \\ \vdots \\ \vdots \\ n_n \end{bmatrix} \qquad (2.36)$$

where we assumed $\deg n(z) = m \leqslant \deg d(z) = n$ and hence $d_n \neq 0$. Normalizing again to $d_n = 1$,

we have a system

$$\underbrace{\left[\begin{array}{cccc} 0 & \cdots & 0 & u_0 \\ \vdots & \iddots & \iddots & u_1 \\ 0 & \iddots & \iddots & u_2 \\ u_0 & \iddots & \iddots & \vdots \\ u_1 & \iddots & \iddots & \\ u_2 & \iddots & & \\ \vdots & & & \end{array}\right.}_{n+1} \underbrace{\left.\begin{array}{cccc} 0 & \cdots & \cdots & 0 \\ \vdots & \iddots & \iddots & y_0 \\ 0 & \iddots & \iddots & y_1 \\ y_0 & \iddots & \iddots & \vdots \\ y_1 & \iddots & \iddots & \\ y_2 & \iddots & & \\ \vdots & & & \end{array}\right]}_{n} \left[\begin{array}{c} n_0 \\ \vdots \\ n_n \\ \hline -d_0 \\ \vdots \\ -d_{n-1} \end{array}\right] = \left[\begin{array}{c} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \\ \vdots \end{array}\right] \tag{2.37}$$

which we can solve for the other coefficients of $n(z)$ and $d(z)$ provided the matrix on the left hand side has full column rank. We prove the following theorem in this context.

Theorem 2.6. Let n be the order of the system generating the causal input output pair $\{u_i\}$ $\{y_i\}$. If the pair is persistently exciting then

$$[U_{:,n+1} \mid Y_{:,n+1}] = \underbrace{\left[\begin{array}{cccc} 0 & \cdots & 0 & u_0 \\ \vdots & \iddots & \iddots & u_1 \\ 0 & \iddots & \iddots & u_2 \\ u_0 & \iddots & \iddots & \vdots \\ u_1 & \iddots & \iddots & \\ u_2 & \iddots & & \\ \vdots & & & \end{array}\right.}_{n+1} \underbrace{\left.\begin{array}{cccc} 0 & \cdots & 0 & y_0 \\ \vdots & \iddots & \iddots & y_1 \\ 0 & \iddots & \iddots & y_2 \\ y_0 & \iddots & \iddots & \vdots \\ y_1 & \iddots & \iddots & \\ y_2 & \iddots & & \\ \vdots & & & \end{array}\right]}_{n+1} \tag{2.38}$$

is singular and

$$[U_{:,n+1} \mid Y_{:,n}] = \underbrace{\left[\begin{array}{cccc} 0 & \cdots & 0 & u_0 \\ \vdots & \iddots & \iddots & u_1 \\ 0 & \iddots & \iddots & u_2 \\ u_0 & \iddots & \iddots & \vdots \\ u_1 & \iddots & \iddots & \\ u_2 & \iddots & & \\ \vdots & & & \end{array}\right.}_{n+1} \underbrace{\left.\begin{array}{cccc} 0 & \cdots & \cdots & 0 \\ \vdots & \iddots & \iddots & y_0 \\ 0 & \iddots & \iddots & y_1 \\ y_0 & \iddots & \iddots & \vdots \\ y_1 & \iddots & \iddots & \\ y_2 & \iddots & & \\ \vdots & & & \end{array}\right]}_{n} \tag{2.39}$$

has full column rank.

Proof. Because of the assumption made, (2.36) holds and hence (2.38) is singular since the vector $[n_0, \ldots, n_n, -d_0, \ldots, -d_n]^T$ is in its kernel. The second part is proved by contradiction. Assume

$[U_{:n+1} \mid Y_{:,n}]$ would *not* have full column rank. Then there exists a vector $[n_0, \ldots, n_n, -d_0, \ldots, -d_{n-1}]^T$ in its kernel. But the first row of this equation then reduces to

$$u_0 n_n \quad = \quad 0. \tag{2.40}$$

Causality implies that the sequence y_i is zero as long as u_i is zero, so we can assume without loss of generality that $u_0 \neq 0$. But then (2.40) implies $n_n = 0$. This now implies we also have a solution $[n_0, \ldots, n_{n-1}, -d_0, \ldots, -d_{n-1}]^T$ for a system (2.35) of order $n-1$ instead of n, which contradicts the assumption of persistence of excitation. □

Remark

1. An impulse is always persistently exciting. This follows from (2.36) which becomes

$$\left[\begin{array}{cccc|cccc} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & h_0 \\ \vdots & \iddots & \iddots & 0 & \vdots & \iddots & \iddots & \vdots \\ 0 & \iddots & \iddots & \vdots & 0 & \iddots & \iddots & \vdots \\ 1 & 0 & \ldots & 0 & h_0 & \ldots & \ldots & h_n \\ \hline 0 & \ldots & \ldots & 0 & h_1 & h_2 & \ldots & h_{n+1} \\ \vdots & & & \vdots & h_2 & & & \vdots \\ & & & & \vdots & & & \end{array}\right] \left[\begin{array}{c} u_0 \\ \vdots \\ n_n \\ \hline -d_0 \\ \vdots \\ -d_n \end{array}\right] \quad = \quad 0 \tag{2.41}$$

which is the standard equation we had in (2.6) and for which no assumption of persistence of excitation was needed.

2. The assumption of minimality of a system is different from persistence of excitation since that is a property of a system, not of a signal pair. These should not be confused with each other.

Exercise 2.5.

Consider the pair $\{u_i\} = \{1, 0, 0, 0, \ldots\}$ and $\{y_i\} = \{1, 0, -1, 0, 0, \ldots\}$, which is assumed to be a second order signal pair. Then

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}\right] \left[\begin{array}{c} -1 \\ 0 \\ 1 \\ \hline 0 \\ 0 \\ -1 \end{array}\right] \quad = \quad 0$$

and from this we find $n(z)/d(z) = (z^2 - 1)/z^2 = 1 - z^{-2}$. Clearly the matrix containing the first five columns has full column rank. □

The columns of the above matrix pair can be rearranged into a block Hankel matrix, i.e. a Hankel matrix with blocks $h_{n+i+1} = [u_i \;\; y_i]$ as elements :

$$\mathcal{H} = \begin{bmatrix} [0,0] & \cdots & [0,0] & [u_0y_0] \\ \vdots & \iddots & \iddots & [u_1y_1] \\ [0,0] & \iddots & \iddots & [u_2y_2] \\ [0,0] & \iddots & \iddots & [u_2y_2] \\ [u_0y_0] & \iddots & \iddots & \vdots \\ [u_1y_1] & \iddots & \iddots & \\ [u_2y_2] & \iddots & & \\ \vdots & & & \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & \cdots & h_{n+1} \\ h_2 & & \iddots & h_{n+2} \\ \vdots & \iddots & \iddots & h_{n+3} \\ h_{n+1} & \iddots & \iddots & \vdots \\ h_{n+2} & \iddots & \iddots & \\ h_{n+3} & \iddots & & \\ \vdots & & & \end{bmatrix} \tag{2.42}$$

If the maximum rank in this matrix is $2n+1$, then a $2(n+1)\times 2(n+2)$ matrix ought to be sufficient to detect the order of the system. Using MIMO Hankel matrix ideas, one shows that the system can be identified from a minimum number of $2n+1$ rows of the above Hankel matrix. This proves again that the minimum number of equations needed to construct $n(z)$ and $d(z)$ is $2n+1$:

$$[U_{2n+1,n+1} \mid Y_{2n+1,n}] \begin{bmatrix} n_1 \\ \vdots \\ n_n \\ \hline -d_0 \\ \vdots \\ -d_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ \vdots \\ \vdots \\ y_{2n} \end{bmatrix}. \tag{2.43}$$

This always yields a solvable system, if n is the correct order of the underlying system.

Exercise 2.6.

Check that $2n+1$ equations are sufficient for the case of the impulse response, by using results of earlier sections. □

Remarks

1. The LU decomposition of (2.43) satisfies for identifying the system $n(z)$, $d(z)$.

2. If more data are available, the larger least squares problem involving the matrix $[U_{N,n+1} \mid Y_{N,n}]$ ought to be preferred for reasons of lower sensitivity.

3. For a more reliable order estimation one should use the singular value decomposition rather than the QR or LU decomposition. On the system

$$M_{N,2n+2} \doteq [U_{N,n+1} \mid Y_{N,n+1}] \tag{2.44}$$

the following interpretation can be given to this decomposition. Although the order indicates that this system only has rank $2n+1$ and hence ought to have one singular value $\sigma_{2n+2} = 0$, this cannot be expected in practice due to noise on the data as well as roundoff noise in

the computations. The SVD of this matrix can now be interpreted in terms of perturbed equations

$$d(z)y(z) \quad = \quad n(z)u(z) + e(z) \tag{2.45}$$

or

$$[U_{N,n+1} \mid Y_{N,n+1}] \left[\begin{array}{c} n_0 \\ \vdots \\ n_n \\ \hline -d_0 \\ \vdots \\ -d_n \end{array}\right] \quad = \quad \left[\begin{array}{c} -e_0 \\ \vdots \\ \vdots \\ \vdots \\ -e_N \end{array}\right]. \tag{2.46}$$

Indeed, let

$$M_{N,2n+2} \quad = \quad U\Sigma V^T \tag{2.47}$$

be the SVD of M, where U is $N \times N$, V is $(2n+2) \times (2n+2)$ and Σ is $N \times (2n+2)$ with the singular values σ_i on diagonal. Let σ_{2n+2} be the smallest of these singular values, then it follows from the properties of this decomposition that

$$\min_{\|v\|=1} \|Mv\| \quad = \quad \sigma_{2n+2} \tag{2.48}$$

and equality is met for

$$Mv_{2n+2} \quad = \quad \sigma_{2n+2} \cdot u_{2n+2} \tag{2.49}$$

where u_{2n+2} and v_{2n+2} are the singular vectors corresponding to σ_{2n+2} in this decomposition. This implies that if one chooses

$$[n_0, \ldots, n_n, -d_0, \ldots, -d_n]^T \quad = \quad v_{2n+2} \tag{2.50}$$

then this is the choice of polynomials $n(z)$ and $d(z)$ with norm $\|n\|^2 + \|d\|^2 = 1$, which minimizes the error

$$\|e\|^2 \quad = \quad \sum_{i=0}^{N} e_i^2 \tag{2.51}$$

among all such polynomials. This can be interpreted as minimizing the variance of the residual error (or noise process) over all possible models of a given order. The least squares solution has the interpretation that it orthogonalizes the error e to previous "data vectors".

The SVD can not be used here anymore to construct a balanced realization from I/O pairs. A construction using more involved concepts is given later.

4. The equations to be solved still display some structure (block Hankel) which could be exploited to yield a fast algorithm. Extensions of the Padé algorithm have been proposed to such matrices but they are very complex and unstable like the basic form of the Padé algorithm. Stabilized version have not been proposed yet.

5. The complexities for identifying an n-th order system (i.e., the polynomials $n(z)$ and $d(z)$ of order n) are thus

   - $16/3n^3$ for the LU decomposition on a minimal matrix $M_{2n+1,2n+1}$
   - $8Nn^2$ for the QR decomposition on $M_{N,2n+1}$ where $N \ll 2n+1$
   - $8Nn^2 + 9\frac{1}{3}(2n)^3$ on the same matrix using the economic SVD approach (first a QR decomposition followed by an SVD on R).

2.7 Recursive least squares

When input output data $\{u_i\}$, $\{y_i\}$ are being collected in real time, (i.e., newer data are available at later time instants) we would like to have the best possible estimate of the system at each time instant. If we use least squares approximations then we want to solve for a family of QR decompositions. If we use minimum variance approximations we what to solve for a family of SVD's. In both cases, Givens transformation play a crucial role in the updating of such decompositions.

We explain first the QR updating problem. Consider the (weighted) least squares problem

$$A_N = \begin{bmatrix} w_0 a_0^T \\ w_1 a_1^T \\ \vdots \\ w_n a_N^T \end{bmatrix} \tag{2.52}$$

where $w_i = \lambda^{N-i}$ and $|\lambda| < 1$ is an exponential weighting factor (also called forgetting factor), and where a_i^T is a column vector of the matrix $[U_{N,n+1} \mid Y_{N,n+1}]$.

We assume we have a QR decomposition of A_N:

$$A_N = Q_N R_N = Q_N \begin{bmatrix} R_- \\ 0^T \\ \vdots \\ 0^T \end{bmatrix} \tag{2.53}$$

where R_- is an upper triangular matrix, and we want to find the corresponding decomposition of the updated matrix

$$A_{N+1} = Q_{N+1} R_{N+1} = Q_{N+1} \begin{bmatrix} R_+ \\ 0^T \\ \vdots \\ 0^T \end{bmatrix}. \tag{2.54}$$

We have the following relation between A_{N+1} and A_N:

$$A_{N+1} \doteq \left[\begin{array}{c} \lambda A_N \\ \hline a_{N+1}^T \end{array}\right] = \left[\begin{array}{c|c} Q_N & \\ \hline & 1 \end{array}\right] \left[\begin{array}{c} \lambda R_- \\ 0 \\ \hline a_{N+1}^T \end{array}\right] \tag{2.55}$$

and it follows from this that solving for the QR decomposition of A_{N+1} involves only the updating decomposition :

$$\left[\begin{array}{c} \lambda R_- \\ \hline a_{N+1}^T \end{array}\right] \;=\; Q_{up} R_{up}. \tag{2.56}$$

From (2.53)-(2.54) one derives indeed that

$$R_{up} \;=\; \left[\begin{array}{c} R_+ \\ \hline 0^T \end{array}\right] \tag{2.57}$$

and Q_{N+1} is a mere block product of Q_N and $Q_{up} \doteq \left[\begin{array}{c|c} Q_{11} & q_{12} \\ \hline q_{21}^T & q_{22} \end{array}\right]$, padded with identities :

$$Q_{N+1} \;=\; \left[\begin{array}{c|c} Q_N & \\ \hline & 1 \end{array}\right] \cdot \left[\begin{array}{c|c|c} Q_{11} & & q_{12} \\ \hline & I & \\ \hline q_{21}^T & & q_{22} \end{array}\right]. \tag{2.58}$$

Since orthogonal transformations can be discarded in least squares problems – once the matrix and the right hand side have been updated with it – we can leave details about storing Q behind here, and concentrating on updating R_-. Solving (2.54) can be done via a sequence of Givens relations and this involves n Givens rotations if the vector a_{N+1}^T has n elements in it. The order in which the elements are eliminated is indicated in the matrix below by the index i of each $\otimes_i$ (illustrated for $n = 5$):

$$\left[\begin{array}{ccccc} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \\ \otimes_1 & \otimes_2 & \otimes_3 & \otimes_4 & \otimes_5 \end{array}\right]. \tag{2.59}$$

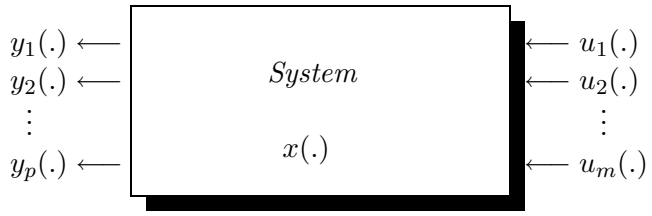
Each element $\otimes_i$ is eliminated by a Givens rotation between rows i and $n+1$ of the matrix. The flop count for these operations is

$$4n + 4(n-1) + \cdots 4(1) \;=\; 4\sum_{i=1}^{n} i \approx 2n^2$$

When applying this to a $N \times n$ matrix we have a total of $2Nn^2$. For the $N \times 2n$ matrix considered in the previous section this would be $2N(2n)^2 = 8Nn^2$. The Householder method for the same matrix is $8Nn^2$ but it constructs both Q and R. If only R is requested, the Householder method reduces to $4Nn^2$, i.e., half of the work of the recursive least squares decomposition only. And the recursive approach meanwhile also constructed the QR decompositions of all intermediate sizes !

2.8 MIMO identification via I/O pairs

We now assume the system to be identified has m inputs and p outputs, so $u_k \in \mathbb{R}^m$ and $y_k \in \mathbb{R}^p$.



We want to identify a system of order n given in state space coordinates:

$$\left\{ \begin{array}{rcl} x_{k+1} & = & Ax_k + Bu_k \\ y_k & = & Cx_k + Du_k \end{array} \right. \tag{2.60}$$

explaining the I/O measurements $\{u_i\}$ $\{y_i\}$ for $i = 1, \ldots, N$, where N of course has to be large enough to be able to reconstruct the model $\{A_{nn}, B_{nm}, C_{pn} D_{pm}\}$. As before we have to assume persistence of excitation of the I/O signals. The componentwise identification of the polynomials $n_{ij}(z)/d_{ij}(z)$ of the transfer function between input j and output i leads to an overestimate of the order when assembling these scalar systems into the $p \times m$ transfer function. So identifying scalar models $n_{ij}(z)/d_{ij}(z)$ or $\{A, b_j, c_i, d_{ij}\}$ has to be abandoned in favor of a direct identification of $\{A, B, C, D\}$.

We start by noting that the problem would be much simpler if the sequence of states x_k would be known as well. From

$$\left[\begin{array}{c} x_{k+1} \\ \hline y_k \end{array} \right] = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right] \left[\begin{array}{c} x_k \\ \hline u_k \end{array} \right] \tag{2.61}$$

we can indeed write the concatenated matrix

$$\left[\begin{array}{cccc} x_2 & x_3 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_{N-1} \end{array} \right] = \left[\begin{array}{cc} A & B \\ C & D \end{array} \right] \left[\begin{array}{cccc} x_1 & x_2 & \cdots & x_{N-1} \\ u_1 & u_2 & \cdots & u_{N-1} \end{array} \right]. \tag{2.62}$$

Under the assumption of persistence of excitation one shows that the right "data matrix" in (2.62) has full column rank $n + m$ and has thus a right inverse. Equivalently, (2.62) can be solved in a least squares sense for the evolution matrix

$$\mathcal{E} = \left[\begin{array}{cc} A & B \\ C & D \end{array} \right]. \tag{2.63}$$

So the problem is solved as soon as the states x_i are determined. But those depend on the choice of coordinates chosen for the state space model. Replace indeed x_i by $\hat{x}_i = T^{-1} x_i$, then (2.62) becomes the related problems

$$\left[\begin{array}{cccc} \hat{x}_2 & \hat{x}_3 & \cdots & \hat{x}_N \\ y_1 & y_2 & \cdots & y_{N-1} \end{array} \right] = \left[\begin{array}{cc} \hat{A} & \hat{B} \\ \hat{C} & D \end{array} \right] \left[\begin{array}{cccc} \hat{x}_1 & \hat{x}_2 & \cdots & \hat{x}_{N-1} \\ u_1 & u_2 & \cdots & u_{N-1} \end{array} \right]. \tag{2.64}$$

or

$$\left[\begin{array}{cccc} T^{-1}x_2 & T^{-1}x_3 & \cdots & \hat{x}_N \\ y_1 & y_2 & \cdots & y_{N-1} \end{array} \right] = \left[\begin{array}{cc} T^{-1}AT & T^{-1}B \\ CT & D \end{array} \right] \left[\begin{array}{cccc} T^{-1}x_1 & T^{-1}x_2 & \cdots & T^{-1}x_{N-1} \\ u_1 & u_2 & \cdots & u_{N-1} \end{array} \right]. \tag{2.65}$$

So each sequence of states

$$X_{1,N} \;=\; [\, x_1 \;\; x_2 \;\; \ldots \;\; x_N \,] \tag{2.66}$$

can only be expected to be known up to an invertible row transformation corresponding to the particular coordinate system of the reconstructed model $\{A, B, C, D\}$. Also the rank condition for (2.62) to be solvable implies that (2.63) must be full rank n since this is a submatrix of the righthand side matrix in (2.62). We now try to derive equations with the vectors $\{u_i\}$ $\{y_i\}$ and $\{x_i\}$ from which we might reconstruct (2.63). By combining

$$\left\{ \begin{array}{rcl} x_{k+2} &=& Ax_{k+1} + Bu_{k+1} \\ y_{k+2} &=& Ax_{k+1} + Bu_{k+1} \end{array} \right.$$

with (2.60) at an earlier instant and eliminating x_{k+1} from this, we find:

$$\begin{array}{rcl} x_{k+2} &=& A^2 x_k + [AB \;\; B] \left[\begin{array}{c} u_k \\ u_{k+1} \end{array} \right] \\ \left[\begin{array}{c} y_k \\ y_{k+1} \end{array} \right] &=& \left[\begin{array}{c} C \\ CA \end{array} \right] x_r + \left[\begin{array}{cc} D & 0 \\ CD & D \end{array} \right] \left[\begin{array}{c} u_k \\ u_{k+1} \end{array} \right]. \end{array}$$

Repeating the same process i times we find the compound equations

$$\begin{array}{rcl} x_{k+i} &=& \mathbf{A}_i x_k + \mathbf{B}_i U_{k,i} \\ Y_{k,i} &=& \mathbf{C}_i x_k + \mathbf{D}_i U_{k,i} \end{array} \tag{2.67}$$

where

$$\left[\begin{array}{c|c} \mathbf{A}_i & \mathbf{B}_i \\ \hline \mathbf{C}_i & \mathbf{D}_i \end{array} \right] \;=\; \left[\begin{array}{c|cccc} A^i & A^{i-1}B & \ldots & AB & B \\ \hline C & D & & & \\ CA & CD & D & & \\ \vdots & \vdots & & \ddots & \\ CA^{i-1} & CA^{i-2}B & \ldots & \ldots & D \end{array} \right] \tag{2.68}$$

and

$$U_{k,i} \doteq \left[\begin{array}{c} u_k \\ \vdots \\ u_{k+i-1} \end{array} \right]; \qquad Y_{k,i} \doteq \left[\begin{array}{c} y_k \\ \vdots \\ y_{k+i-1} \end{array} \right]. \tag{2.69}$$

Now stacking several of these vectors $U_{k,i}$ and $Y_{k,i}$ next to each other yields

$$Y_{k,i,j} \;\doteq\; \left[\begin{array}{cccc} y_k & y_{k+1} & \cdots & y_{k+j-1} \\ \vdots & \vdots & & \vdots \\ y_{k+i-1} & y_{k+i} & \cdots & y_{k+i+j-2} \end{array} \right] \tag{2.70}$$

$$U_{k,i,j} \;\doteq\; \left[\begin{array}{cccc} u_k & u_{k+1} & \cdots & y_{k+j-1} \\ \vdots & \vdots & & \vdots \\ u_{k+i-1} & u_{k+i} & \cdots & y_{k+i+j-2} \end{array} \right] \tag{2.71}$$

which are both block Hankel matrices. Together with

$$X_{k,j} \doteq [\, x_k \; x_{k+1} \quad \dots \quad x_{k+j-1} \,] \tag{2.72}$$

we have from (2.64), (2.65), and (2.66) that

$$Y_{k,i,j} = \mathbf{C}_i X_{k,j} + \mathbf{D}_i U_{k,i,j} \tag{2.73}$$

which says that the rows of $\mathbf{Y}_{k,i,j}$ are linear combinations of the row of $X_{k,j}$ and $U_{k,i,j}$. This then leads to the following theorem.

Theorem 2.7. Let

$$H_{k,i,j} = \left[\frac{Y_{k,i,j}}{U_{k,i,j}}\right] \tag{2.74}$$

be a block Hankel matrix of input output pairs. Provided $i \geqslant n$, $j \geqslant (m+p)i$ and $\{u_i\}$ $\{y_i\}$ is a persistently exciting pair, then

$$\text{rank}(H_{k,i,j}) = \text{rank}(U_{k,i,j}) + \text{rank}(X_{k,j}) = mi + n \tag{2.75}$$

Proof. If $i \geqslant n$ then $\mathbf{C}_i$ is full column rank n. If $j \geqslant (m+p)i$ then $Y_{k,i,j}$, $X_{k,j}$ and $U_{k,i,j}$ are all matrices with more columns than rows. From (2.73)-(2.74) we find then that

$$\text{rank}(H_{k,i,j}) \leqslant \text{rank}(U_{k,i,j}) + \text{rank}(X_{k,j})$$

since the rows of $Y_{k,i,j}$ are linear combinations of those of $U_{k,i,j}$ and $X_{k,j}$. If persistence of excitation is present then the rows of $U_{k,i,j}$ and $X_{k,j}$ are linearly independent, as well as the way they enter via the matrix $Y_{k,i,j}$ in (2.73) and then we have

$$\text{rank}(H_{k,i,j}) = \text{rank}(U_{k,i,j}) + \text{rank}(X_{k,j})$$

which completes the proof. □

We now use this result to determine a basis for the rows of $X_{k+i,j}$. Consider two equations as in (2.73) shifted over i time steps. So in addition to (2.73) we have

$$Y_{k+i,i,j} = \mathbf{C}_i X_{k+i,j} + \mathbf{D}_i U_{k+i,i,j}. \tag{2.76}$$

Hence

$$\text{rank}(H_{k+i,i,j}) = \text{rank}(U_{k+i,i,j}) + \text{rank}(X_{k+i,j}) = mi + n. \tag{2.77}$$

This leads to the following theorem.

Theorem 2.8. Define

$$X_{k+i,j} \doteq [\, x_{k+i} \; x_{k+i+1} \quad \dots \quad x_{k+i+j-1} \,], \tag{2.78}$$

then

$$Im\left[X_{k+i,j}^T\right] = Im\left[H_{k,i,j}^T\right] \cap Im\left[H_{k+i,i,j}^T\right] \tag{2.79}$$

provided the underlying I/O pair is persistently exciting and $i \geqslant n$, $j \geqslant (m+p)i$.

Proof. The notation $Im\left[M^T\right]$ stands for the row space of a matrix M. From (2.74)-(2.77) we find

$$\text{rank}(H_{k,i,j}) = \text{rank}(H_{k+i,i,j}) = mi + n.$$

But theorem 2.7 applies also to the larger Hankel matrix $H_{k,2i,j}$:

$$\text{rank}(H_{k,2i,j}) = 2mi + n.$$

And since $H_{k,2i,j}$ is just a row permutation P of the rows of $H_{k,i,j}$ and $H_{k+i,i,j}$:

$$H_{k,2i,j} = P \begin{bmatrix} H_{k,i,j} \\ H_{k+i,i,j} \end{bmatrix},$$

it follows that

$$\dim(Im\left[H_{k,i,j}^T\right] \cap Im\left[H_{k+i,i,j}^T\right]) = n. \tag{2.80}$$

This follows from the general theorem that

$$\dim(Im\left[M_1 \mid M_2\right]) = \dim(Im\left[M_1\right]) + \dim(Im\left[M_2\right]) - \dim(Im\left[M_1\right] \cap Im\left[M_2\right]).$$

So the dimension of the intersection (2.80) has the assumed dimension of (2.78). In order to prove the theorem, we thus only have to prove that

$$Im\left[X_{k+i,j}^T\right] \subset Im\left[H_{k,i,j}^T\right] \cap Im\left[H_{k+i,i,j}^T\right]$$

or in other words, that the row space of $X_{k+i,j}$ lies in both row spaces of $H_{k,i,j}$ and $H_{k+i,i,j}$. Now (2.76) and the fact that $\mathbf{C}_i$ has full column rank, indicates that

$$Im\left[X_{k+i,j}^T\right] \subset Im\left[H_{k,i,j}^T\right].$$

Also from an appropriate concatenation of (2.67) we find

$$Im\left[X_{k+i,j}^T\right] \subset Im\left[H_{k+i,i,j}^T\right]$$

because of (2.72). This completes the proof of the theorem. □

In practice, due to perturbations on the data, the row spaces typically do not intersect. An approximate intersection, using the singular value decomposition or some rank revealing QR decomposition has thus to be constructed. A possible implementation of this idea is the following decomposition :

$$\left[\begin{array}{c} H_{k,i,j} \\ \hline H_{k+i,i,j} \end{array}\right] = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & Q \end{array}\right] \left[\begin{array}{cc|c} A_{11} & A_{12} & 0_1 \\ \hline A_{21} & 0_3 & 0_2 \\ \times & \times & A_{33} \end{array}\right] V^T \tag{2.81}$$

where :

- $[A_{11}A_{12}]$ has full column rank equal to the rank of $H_{k,i,j}$ (or $mi+n$ under the assumption of persistence of excitation)

- A_{33} has full row rank which must be smaller than $mi + n$ if an intersection is to be detected
- A_{21} has full row rank equal to the dimension of the intersection, hence n.

The order in which this decomposition is constructed is as follows. First the transformation V^T is constructed to compress the columns of $H_{k,i,j}$, yielding the trailing zero matrix 0_1. Then the rows of the trailing bottom matrix are compressed with the transformation Q, yielding 0_2 and a full row rank A_{33}. Then V^T is updated to yield the full column rank matrix A_{21} and the trailing zero matrix 0_3. Notice that all three steps involve arank factorization which essentially can be done with QR decompositions. The center matrix in this decomposition has a form which trivially displays the intersection of row spaces of the top and bottom parts, namely :

$$Im \begin{bmatrix} A_{11}^T \\ A_{12}^T \\ 0 \end{bmatrix} \cap Im \begin{bmatrix} A_{21}^T & \times \\ 0 & \times \\ 0 & A_{33}^T \end{bmatrix} = Im \begin{bmatrix} I_n \\ 0 \\ 0 \end{bmatrix}.$$

Because of the transformation V^T in (2.81) one derives that

$$Im(H_{k.i.j}^T) \cap Im(H_{k+i.i.j}^T) = V \cdot Im \begin{bmatrix} I_n \\ 0 \\ 0 \end{bmatrix},$$

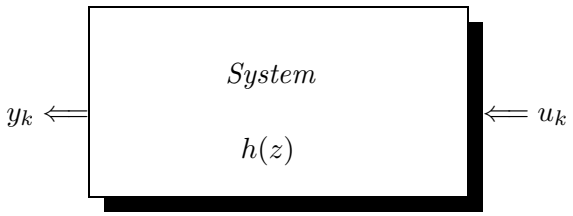
i.e., the first n rows of V^T are a representation of $X_{k+i,j}$. From this we can now construct $\{A, B, C, D\}$ as explained in (2.62)-(2.65). An alternative and slightly more expensive method based on SVD's is proposed in [97] and matlab codes are appended below.

```
function [H]=bhankel(M,Nr,Nc,nr,nc)
%
% The function [H]=bhankel(M,Nr,Nc,nr,nc) constructs
% a block hankel matrix H from a given matrix M.
% The matrix H will have Nr block rows and Nc block
% columns, each block being nr x nc. The blocks are
% given in M either columnwise (M has then nc columns)
% or rowwise (M has then nr rows). M must contain at
% least Nr+Nc-1 such blocks.
%
[mr,mc]=size(M);
if mr*mc < (Nc+Nr-1)*nr*nc, disp('Data matrix too small'); return, end
H=zeros(Nr*nr,Nc*nc);
if mr==nr,
   for i=1:Nr, H((i-1)*nr+1:i*nr,:)=M(:,(i-1)*nc+1:(i+Nc-1)*nc); end
  else
   if mc==nc,
      for i=1:Nc, H(:,(i-1)*nc+1:i*nc)=M((i-1)*nr+1:(i+Nr-1)*nr,:); end
     else
      disp('Wrong dimensions');
   end
end
```

```
function [a,b,c,d]=abcdio(H,m,p,tol)
%
% The function [a,b,c,d]=abcdio(H,m,n,tol) determines a system
% {a,b,c,d} which reproduces the io pairs u(k) y(k) corresponding
% to
%                   x(k+1) = a x(k) + b u(k)
%                     y(k) = c x(k) + d u(k)
%
% from a given hankel matrix H containing the io pairs
%
%         H(i,j)=z(i-j+1)   with z(k)'=[u(k)' y(k)']
%
% The input and output dimensions are m and p respectively.
% The tolerance tol is used to find an appropriate state
% dimension for x(.) in order to fit the data.
%
[mh,nh]=size(H);mh2=mh/2;i=mh2/(m+p);
if mh2 ~= i*(m+p), disp('Incorrect dimensions'); return, end
[u,s,v]=svd(H(1:mh2,:));pp=rank(s,tol);n=pp-m*i;
if n < 1 , disp('Zero or negative state dimension'); return, end
uu=H(mh2-m-p+1:mh2-p,2:nh);yy=H(mh2-p+1:mh2,2:nh);
H=H*v;h3=H(mh2+1:mh,pp+1:nh);[u3,s3,v3]=svd(h3);
H(mh2+1:mh,:)=u3'*H(mh2+1:mh,:);h2=H(mh2+m*i+1:mh,1:pp);
[u2,s2,v2]=svd(h2);H(:,1:pp)=H(:,1:pp)*v2;
x=v(:,1:pp)*v2(:,1:n);x=x';
B=[x(:,2:nh);yy];A=[x(:,1:nh-1);uu];M=B/A;
a=M(1:n,1:n);b=M(1:n,n+1:n+m);
c=M(n+1:n+p,1:n);d=M(n+1:n+p,n+1:n+m);
```

2.9 Linear prediction

In this section we try to identify a system with transfer function $H(z)$ from its response to an input signal which is a stationary white noise process:



So the input signal u_k is not known to us, but we know that

$$E\{u_k u_{k-i}\} \quad = \quad \delta_{ik} n$$

where we assume $n = 1$ for simplicity. Since we do not have access to $\{u_k\}$, we can only try to find a transfer function $h(z)$ such that the output of the above system has a cross correlation

$$E\{y_k y_{k-i}\} \quad = \quad r(k, k-i)$$

which is compatible with this picture. If one restricts the system $h(z)$ to be an "all pole" (also called autoregressive) system,

$$h(z) \quad = \quad \frac{z^n}{d(z)}$$

where $d(z)$ is a certain polynomial of degree n, then this problem turns out to have a simple recursive solution. Write $d(z)/z^n$ as

$$\frac{d(z)}{z^n} \quad = \quad \frac{1 + a_1 z^{-1} + \cdots + a_n z^{-n}}{c}$$

then one tries to find a system of the type

$$h(z) = \frac{c}{a(z^{-1})} = \frac{c}{1 + \sum_{i-1}^{n} a_i z^{-i}}$$

where $a(\cdot)$ is an nth order polynomial. The response to the system is thus:

$$y_k \quad = \quad c u_k - \sum_{i=1}^{n} a_i y_{k-i}. \tag{2.82}$$

If the a_i and c coefficients are fixed, then $\{y_k\}$ is again a stationary process. Therefore,

$$E\{y_k, y_{k-j}\} \quad = r_j \quad = r_{-j}$$

is only function of the distance i between the samples. Multiplying (2.82) by y_{k-j} and taking expected values yields

$$r_j \quad = \quad -\sum_{i=1}^{n} a_i r_{j-i}, \qquad \text{for } 1 \leqslant |j| \leqslant \infty$$

and

$$r_0 \quad = \quad c - \sum_{i=1}^{n} a_i r_i$$

because we know that u_r is uncorrelated with y_{k-j} for $j > 0$. One can put this together in the matrix equation

$$\begin{bmatrix} r_0 & \dots & r_{n-1} \\ \vdots & \ddots & \vdots \\ r_{n-1} & \dots & r_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad \equiv \quad \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}, \tag{2.83}$$

which is a system of equations that has a lot of similarity with the Hankel equation for identifying a system from its impulse response (see [82] for more details on this).

The above matrix T_n is called a Toeplitz matrix and is known to be positive definite because it is the autocorrelation matrix of $\{y_k\}$. From (2.83) one sees that the vector $[1, a_1, \ldots, a_n]^T/c$ is in fact the first column of the inverse of T_{n+1}, since

$$\begin{bmatrix} r_0 & \ldots & r_n \\ \vdots & \ddots & \vdots \\ r_n & \ldots & r_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} c \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.84}$$

In order to solve (2.83) or (2.84) we thus need a factorization of T_{n+1}. Since it is positive definite and symmetric there exist matrices L and X such that

$$T_{n+1} = LSL^T \tag{2.85}$$

and

$$X^T T_{n+1} X = S \tag{2.86}$$

where L is unit lower triangular, $X^T = L^{-1}$ is unit lower triangular and S is diagonal. This follows from Corollary 2.1 and the fact that any positive definite matrix has positive principal minors (see [46]). Notice that this implies also that the diagonal elements of Sare positive.

There are two fast algorithms for computing the above decomposition. The Levinson algorithm goes back to (1949) and computes X in $0(n^2)$ operations. The Schur algorithm goes back to (18..) and computes L in $0(n^2)$ operations.

We derive here the simplest one of both algorithms but MATLAB codes for both are given in Appendix. Just us for the Padé algorithm, the key idea in the Levinson algorithm is recursivity. We assume that we have a solution to (2.84) for T_{i+1}, i.e.,

$$\begin{bmatrix} r_0 & \ldots & r_i \\ \vdots & \ddots & \vdots \\ r_i & \ldots & r_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1^{(i)} \\ \vdots \\ a_i^{(i)} \end{bmatrix} = \begin{bmatrix} c_i \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.87}$$

Because T_i is symmetric with respect to both the diagonal and the anti-diagonal (this is called centre symmetry) we can flip around this equation and get the identity

$$\begin{bmatrix} r_0 & \ldots & r_i \\ \vdots & \ddots & \vdots \\ r_i & \ldots & r_0 \end{bmatrix} \begin{bmatrix} a_i^{(i)} \\ \vdots \\ a_1^{(i)} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ c_i \end{bmatrix}. \tag{2.88}$$

For the solution of the system (2.87), (2.88) incremented by 1 we just try out the shifted vectors

first, and get,

$$\begin{bmatrix} r_0 & \dots & r_{i+1} \\ \vdots & \ddots & \vdots \\ r_{i+1} & \dots & r_0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ a_1^{(i)} & a_i^{(i))} \\ \vdots & \vdots \\ a_i^{(i)} & a_1^{(i))} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} c_i & \gamma_i \\ 0 & 0 \\ \vdots & \vdots \\ \gamma_i & c_i \end{bmatrix}. \tag{2.89}$$

where $\gamma_i = r_{i+1} \sum_{j=1}^{i} a_j^{(i)} r_{i+1-j}$. Now define a little 2×2 transformation $\begin{bmatrix} 1 & \rho_i \\ \rho_i & 1 \end{bmatrix}$, $\rho_i = -\gamma_i / c_i$ and apply it to the right hand side of (2.89) to get

$$\begin{bmatrix} c_i & \gamma_i \\ 0 & 0 \\ \vdots & \vdots \\ \gamma_i & c_i \end{bmatrix} \begin{bmatrix} 1 & \rho_i \\ \rho_i & 1 \end{bmatrix} = \begin{bmatrix} c_{i+1} & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & c_{i+1} \end{bmatrix}$$

with $c_{i+1} = c_i(1 - \gamma_i^2/c_i^2) = c_i(1 - \rho_i^2)$. Since the right hand sides are now in the form required by (2.87)-(2.88), the same must hold for the left hand side, i.e.,

$$\begin{bmatrix} a_{i+1}^{(i+1)} \\ a_i^{(i+1)} \\ \vdots \\ a_1^{(i+1)} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ a_1^{(i)} & a_i^{(i)} \\ \vdots & \vdots \\ a_i^{(i)} & a_1^{(i)} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \rho_i \\ 1 \end{bmatrix} \tag{2.90}$$

which is the requested update formula. The link with the decomposition (2.86) is that the last column of X in (2.86) satisfies an equation like (2.88) as easily follows from taking the last column of $T_{n+1}X = X^{-T}S$. The recurrence (2.90) thus generates the column of X. The complexity of the i-th step is $2i$ flops, i for computing γ_i and i for (2.90). Summed over n steps this yields an operation count of $\sum_{i=1}^{n} 2i = n^2$ flops.

The coefficients ρ_i are called the reflection coefficients and have important physical meaning in seismic data analysis and in speech processing. They are bounded by 1 but can get close to 1 when T_n is nearly singular. This follows from the determinant of T_n which equals the product of the coefficients L_i.

The numerical stability of the Levinson algorithm (in a weak sense) has been proved by Cybenko [22] and that of Schur algorithm by Bultheel [16]. Both results prove that the errors in the computed decompositions are proportional to $\epsilon \cdot \kappa(T_n)$, i.e., can only be big when the problem is badly conditioned. This does *not* mean that the algorithms are backward stable, but it does say that the forward errors are comparable in size to what a backward algorithm would imply. Since bad conditioning is indicated by reflection coefficients being close to 1, it is easy to detect when the problem is badly conditioned.

Instead of deriving the Schur algorithm we just give the algorithm without proof and refer to [25] for an elegant matrix proof. We first assume T_n to be scaled such that $r_0 = 1$, which is

obtained by dividing T_n by r_0. This scaling is easily absorbed in the diagonal matrix S. Then we start with the $2 \times (n+1)$ array:

$$G = \begin{bmatrix} 1 & r_1 & \cdots & r_n \\ 0 & r_1 & \cdots & r_n \end{bmatrix} \dot{=} \begin{bmatrix} 1 & l_2^{(1)} & \cdots & l_{n+1}^{(1)} \\ 0 & g_2^{(1)} & \cdots & g_{n+1}^{(1)} \end{bmatrix}.$$

Each step of the Schur algorithm consists of the following operations. First shift the bottom row to the left to fill the zero element and also drop the last element of the top row:

$$G := \begin{bmatrix} 1 & l_2^{(1)} & \cdots & l_n^{(1)} \\ g_2^{(1)} & g_3^{(1)} & \cdots & g_{n+1}^{(1)} \end{bmatrix}.$$

Then perform a little 2×2 transformation $1/(1-\rho_1^2) \begin{bmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{bmatrix}$

$$\frac{1}{(1-\rho_1^2)} \begin{bmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{bmatrix} G = \begin{bmatrix} 1 & l_2^{(2)} & \cdots & l_n^{(2)} \\ 0 & g_2^{(2)} & \cdots & g_n^{(2)} \end{bmatrix}$$

and then repeat the same procedure until the array G is empty (i.e., after $n+1$ steps). One shows that the rows $\left[1, l_2^{(i)}, \ldots l_{n-i+2}^{(i)} \right]$ are in fact the rows of L in (2.90). The complexity can again be shown to be n^2 flops when approximately implementing the 2×2 transformation. It can be shown as well that the coefficients ρ_i are again those of the Levinson algorithm, which indicates the relation between both algorithms.

Vector processes can also be dealt with in the above problem. We then start from an input sequence which is stationary white noise, meaning

$$E\{u_k \; u_{k-i}^T\} = \delta_{ik} I_m.$$

The cross correlation of the output process is measured

$$E\{y_k \; y_{k-i}^T\} = R_i$$

and a predictor polynomial matrix

$$A(z^{-1}) = I + \sum_{i=1}^{n} A_i z^{-i}$$

is found from the decomposition of the block Toeplitz matrix

$$T_{n+1} = \begin{bmatrix} R_0 & R_1 & \ldots & R_n \\ R_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & R_1 \\ R_n & \ldots & R_1 & R_0 \end{bmatrix}.$$

Both the Levinson and the Schur algorithm have block versions which have complexity $n^2 m^3$ instead of $n^3 m^3$ for a general method for decomposing T_{n+1}.

Some Remarks

The covariance r_i or R_i is usually estimated from the data sequence $\{y_k\}$. This implies switching from expectation to time averages, which only holds if the process is ergodic. Moreover, time averages are typically taken over a finite or windowed interval :

$$R_i = \sum_k w_k y_k y_{k-i}^T, \quad \sum_k w_k = 1.$$

The above algorithms all extend to non definite Toeplitz matrices but breakdowns may then occur and the algorithms are then unstable in general. The stability for the definite case is linked to the fact that for positive definite matrices no pivoting is required in the LU decomposition.

The Levinson recurrence has also polynomial versions, which are:

$$a_{(i+1)}(z) \quad = \quad z a_{(i1)}(z) + \rho_i \hat{a}_{(i)}(z)$$

where $\hat{a}_{(i)}(z)$ is the "reversed" polynomial of $a_{(i)}(z)$, i.e., the polynomial obtained by flipping the order of the coefficients. One checks that this is nothing but equation (2.90) in polynomial language.

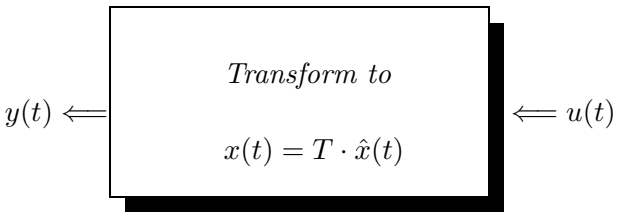
Chapter 3

STATE SPACE ANALYSIS

3.1 Orthogonal State-Space Transformations

In this chapter we present a number of more reliable methods for analysis and design of a control systems using state space representations. A common factor is all these methods is the use of orthogonal state-space transformations rather than just invertible ones. We illustrate the use of orthogonal (or unitary state-space transformation in the complex case) via the concept of condensed forms introduced in [140].

If one applies a general state space transformation T to a system



with $y \in \mathbb{R}^m$, $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^p$, then the new coordinates give the system

$$\begin{cases} \lambda\hat{x}(t) &= \hat{A}\hat{x}(t) + \hat{B}u(t) \\ y(t) &= \hat{C}\hat{x}(t) + Du(t) \end{cases} \tag{3.1}$$

with $\{\hat{A}, \hat{B}, \hat{C}, D\} = \{T^{-1}AT, T^{-1}B, CT, D\}$.

Since an invertible matrix T has essentially n^2 degrees of freedom (only one inequality constraint $\det.\ T \neq 0$ is imposed) we can use these degrees of freedom to assign a number of zeros in the transformed system $\{\hat{A}, \hat{B}, \hat{C}, D\}$. We illustrate this for $m = p = 1$ and $n = 5$ and for a system with unrepeated poles. The transformation T can be chosen to obtain one of the following forms:

a) the Jordan canonical form

$$\left[\begin{array}{c|c} \hat{b}_j & \hat{A}_j \\ \hline d & \hat{c}_j \end{array}\right] = \left[\begin{array}{c|ccccc} \times & \times & 0 & 0 & 0 & 0 \\ \times & 0 & \times & 0 & 0 & 0 \\ \times & 0 & 0 & \times & 0 & 0 \\ \times & 0 & 0 & 0 & \times & 0 \\ \times & 0 & 0 & 0 & 0 & \times \\ \hline \times & 1 & 1 & 1 & 1 & 1 \end{array}\right], \tag{3.2}$$

b) the controller canonical form

$$\left[\begin{array}{c|c} \hat{b}_c & \hat{A}_c \\ \hline d & \hat{c}_c \end{array}\right] = \left[\begin{array}{c|ccccc} 1 & \times & \times & \times & \times & \times \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \hline \times & \times & \times & \times & \times & \times \end{array}\right], \tag{3.3}$$

c) the observer canonical form

$$\left[\begin{array}{c|c} \hat{b}_o & \hat{A}_o \\ \hline d & \hat{c}_o \end{array}\right] = \left[\begin{array}{c|ccccc} \times & 0 & 0 & 0 & 0 & \times \\ \times & 1 & 0 & 0 & 0 & \times \\ \times & 0 & 1 & 0 & 0 & \times \\ \times & 0 & 0 & 1 & 0 & \times \\ \times & 0 & 0 & 0 & 1 & \times \\ \hline \times & 0 & 0 & 0 & 0 & 1 \end{array}\right]. \tag{3.4}$$

Notice that each of these forms has n^2 elements assigned either to 0 or to 1. If one now uses orthogonal transforms instead, then the best one can hope for is to create $n(n-1)/2$ zeros since these are the degree of freedom in an orthogonal transformation U. The (symmetric) equation $U^T U = I_n$ indeed imposes $n(n+1)/2$ constraints on the elements of U. The equivalent orthogonal forms to the three above ones would be:

a) the Schur form

$$\left[\begin{array}{c|c} \hat{b}_s & \hat{A}_s \\ \hline d & \hat{c}_s \end{array}\right] = \left[\begin{array}{c|ccccc} \times & \times & \times & \times & \times & \times \\ \times & 0 & \times & \times & \times & \times \\ \times & 0 & 0 & \times & \times & \times \\ \times & 0 & 0 & 0 & \times & \times \\ \times & 0 & 0 & 0 & 0 & \times \\ \hline \times & \times & \times & \times & \times & \times \end{array}\right], \tag{3.5}$$

b) the controller Hessenberg form

$$\left[\begin{array}{c|c} \hat{b}_c & \hat{A}_c \\ \hline d & \hat{c}_c \end{array}\right] = \left[\begin{array}{c|ccccc} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \\ \hline \times & \times & \times & \times & \times & \times \end{array}\right], \tag{3.6}$$

c) the observer Hessenberg form

$$\left[\begin{array}{c|c} \hat{b}_o & \hat{A}_o \\ \hline d & \hat{c}_o \end{array}\right] = \left[\begin{array}{c|ccccc} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & 0 & \times & \times & \times & \times \\ \times & 0 & 0 & \times & \times & \times \\ \times & 0 & 0 & 0 & \times & \times \\ \hline \times & 0 & 0 & 0 & 0 & \times \end{array}\right]. \tag{3.7}$$

Clearly each of these forms have a strong similarity to their non-orthogonal counterpart. The "canonical forms" have been used a lot in textbooks because of their simplicity and their use in several analysis and synthesis problems. The orthogonal forms, which we will call "condensed forms", will be shown to be just as useful for analysis and synthesis problems, and moreover they have other major advantages from a numerical point of view.

Remark. Canonical forms arise whenever one defines a transformation group, i.e., a set of transformation which is closed under inversion and multiplication. Clearly invertible transformations T form a group, but so do the unitary transformations. One shows that in fact the condensed forms are canonical forms under the transformation group of orthogonal transforms.

3.2 Condensed Forms

The choice of unitary or orthogonal state-space transformations is governed by the following items

i) the cost of these transformations in reasonable, namely $O(n^3)$ for each of them,

ii) they do not affect the sensitivity of most underlying problems, because condition numbers are usually invariant under unitary transformations,

iii) they go along with bounded error propagation, or can be implemented in a numerically stable manner,

iv) they put the state-space model in a condensed form which simplifies the subsequent control problem.

In practice, n is considerably larger than m and p and most of the zeros will therefore be obtained in $\hat{A}$. The four main condensed forms encountered in the literature are:

i) the *state-Hessenberg* form, where $\hat{A}$ is upper Hessenberg,

ii) the *state-Schur* form, where $\hat{A}$ is upper Schur,

iii) the *controller-Hessenberg* form, where the compound matrix $[\hat{B} \mid \hat{A}]$ is upper trapezoidal,

iv) the *observer-Hessenberg* form, where the compound matrix $\left[\frac{\hat{A}}{\hat{C}}\right]$ is upper trapezoidal.

These forms are illustrated below for $m = 2$, $n = 7$, $p = 3$.

i) state Hessenberg form (complex arithmetic)

$$\left[\begin{array}{c|c} \hat{B}_h & \hat{A}_h \\ \hline D & \hat{C}_h \end{array}\right] = \left[\begin{array}{cc|ccccccc} \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & 0 & \times & \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & \times & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & 0 & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & 0 & 0 & \times & \times \\ \hline \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \end{array}\right], \tag{3.8}$$

ii) state Schur form (complex arithmetic)

$$\left[\begin{array}{c|c} \hat{B}_s & \hat{A}_s \\ \hline D & \hat{C}_s \end{array}\right] = \left[\begin{array}{cc|ccccccc} \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & 0 & \times & \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & \times & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & 0 & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & 0 & 0 & \times & \times \\ \times & \times & 0 & 0 & 0 & 0 & 0 & 0 & \times \\ \hline \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \end{array}\right], \tag{3.9}$$

iii) controller Hessenberg form

$$\left[\begin{array}{c|c} \hat{B}_c & \hat{A}_c \\ \hline D & \hat{C}_c \end{array}\right] = \left[\begin{array}{cc|ccccccc} \times & \times & \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & 0 & \times & \times & \times \\ \hline \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \end{array}\right], \tag{3.10}$$

iv) observer Hessenberg form

$$\left[\begin{array}{c|c} \hat{B}_o & \hat{A}_o \\ \hline D & \hat{C}_o \end{array}\right] = \left[\begin{array}{cc|ccccccc} \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & 0 & \times & \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & \times & \times & \times & \times \\ \hline \times & \times & 0 & 0 & 0 & 0 & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & 0 & 0 & \times & \times \\ \times & \times & 0 & 0 & 0 & 0 & 0 & 0 & \times \end{array}\right]. \tag{3.11}$$

The complexity of constructing the forms (i), (iii) and (iv) is $n^2(5/3n + m + p)$ for computing the new state space system $\{\hat{A}, \hat{B}, \hat{C}, D\}$ and an additional n^3 for computing the transformation U itself. The form (ii) also involves an iterative process (the eigenvalues of A are computed) and this requires $n^2\left(\frac{5}{2}kn + m + p\right)$ for computing $\{\hat{A}, \hat{B}, \hat{C}, D\}$ and an additional $\frac{5}{2}kn^3$ for computing U as well. Here k is the average number of iteration steps for finding each eigenvalue up to machine accuracy, and typically lies between 1 and 2.

We note here that there are many variants of the above forms. First of all, to every "upper" form there is a corresponding "lower" form, which we did not show here. Then, the Schur form shown above implies the use of complex arithmetic. Under real (orthogonal) transformations one can *not* triangularize a matrix as shown under ii) if it has complex conjugate eigenvalues. Instead one can always obtain a quasi-triangular form with 1×1 or 2×2 diagonal block corresponding to the real and complex conjugate eigenvalues, respectively. Finally forms (i), (iii) and (iv) have variants in which some of the *pivot elements*, (i.e., the elements directly above and to the right of the triangle of zeros) are zero as well (see later).

The construction of these condensed forms is now illustrated for the controller Hessenberg form and for $m = 1$. Essentially, we have to find here an orthogonal transform such that

$$U^H b = \begin{bmatrix} x \\ 0 \\ \vdots \\ 0 \end{bmatrix}; \qquad U^H A U = \begin{bmatrix} x & x & \dots & x \\ x & x & \dots & x \\ \vdots & \vdots & \ddots & \vdots \\ x & x & \dots & x \end{bmatrix}.$$

It is easy to see that one can find a first orthogonal transformation U_1 such that

$$U_1^H b = \left[\begin{array}{c} x_1 \\ \hline 0 \\ \vdots \\ 0 \end{array}\right]; \qquad U_1^H A U_1 = \left[\begin{array}{c|ccc} x & x & \dots & x \\ \hline x & x & \dots & x \\ \vdots & \vdots & \ddots & \vdots \\ x & x & \dots & x \end{array}\right].$$

In fact U_1 is just a Householder transformation H_1 putting b in the required form (see Chapter 1). If one now redefines the matrices b and A as b_1 and A_1 and the bottom two blocks of $U_1^H A U_1$ as b_2 and A_2, then one sees that the additional transformation

$$U_2^H = \left[\begin{array}{c|c} 1 & \\ \hline & H_2^H \end{array}\right]$$

applied to

$$U_2^H U_1^H b_1 = \left[\begin{array}{c} x_1 \\ \hline 0 \\ \vdots \\ 0 \end{array}\right] \qquad U_2^H U_1^H A U_1 U_2 = \left[\begin{array}{c|ccc} x & x & \dots & x \\ \hline H_2^H b_2 & & H_2^H A_2 H_2 & \end{array}\right]$$

in fact applies H_2 to b_2 and A_2 in a similar manner as H_1 was applied to b_1 and A_1 whereby previously created zeros in $U_1^H b_1$ remain unaffected. So we can now choose H_2 such that

$$U_2^H U_1^H b_1 = \left[\begin{array}{c} x_1 \\ \hline 0 \\ \vdots \\ 0 \end{array}\right] \qquad U_2^H U_1^H A_1 U_1 U_2 = \left[\begin{array}{c|ccc} x & x & \dots & x \\ \hline x_2 & x & x & \dots & x \\ 0 & & & & \\ \vdots & b_3 & & A_3 & \\ 0 & & & & \end{array}\right].$$

The same process now continues on the subsystem (b_3, A_3) using

$$U_3 \left[\begin{array}{c|c} I_2 & \\ \hline & H_3 \end{array}\right]$$

and so on, until the required controller Hessenberg form is obtained. All transformations are also applied to the vector c which does not play a role in the above construction.

For the derivation of the general forms (i), (iii) and (iv) essentially the same ideas apply. For the derivation of form (ii) one uses the construction of the Schur form explained in Chapter 2. In the following sections, we show how these condensed forms are now exploited for the derivation of efficient algorithms.

3.3 Controllability, Observability and Minimality

It is well known that a state space realization

$$\left\{\begin{array}{rcl} \lambda x & = & Ax + Bu \\ y & = & Cx + Du \end{array}\right. \tag{3.12}$$

with state dimension n, input dimension m and output dimension p, is minimal if and only if it is controllable and observable. The most used algebraic definition for these two conditions are:

$$\text{C1:} \quad \text{rank} \left[\begin{array}{cccc} B & AB & \cdots & A^{n-1}B \end{array}\right] = n \tag{3.13}$$

and

$$\text{O1:} \quad \text{rank} \left[\begin{array}{c} C \\ CA \\ \ddots \\ CA^{n-1} \end{array}\right] = n \tag{3.14}$$

but one encounters also conditions

$$\text{C2:} \quad \text{rank} \left[\begin{array}{cc} B & A - \lambda I \end{array}\right] = n \quad \forall \lambda \tag{3.15}$$

$$\text{O2:} \quad \text{rank} \left[\begin{array}{c} A - \lambda I \\ \hline C \end{array}\right] = n \quad \forall \lambda. \tag{3.16}$$

Exercise 3.1

Prove that conditions C1 and C2 are equivalent (or equivalently conditions O1 and O2). □

A third condition of more practical relevance is:

$$\text{C3: } \exists F \Rightarrow \lambda(A+BF) \cap \lambda(A) \quad = \quad \phi \tag{3.17}$$

$$\text{O3: } \exists K \Rightarrow \lambda(A+KC) \cap \lambda(A) \quad = \quad \phi \tag{3.18}$$

in other words, eigenvalues can all be moved by feedback or feedforward operations. This last condition is the relevant one for design purposes. We now indicate that each of these conditions is rather questionable from a numerical point of view. Since observability and controllability are dual to each other we focus on controllability.

Example 3.1

We take the following matrix pair:

$$A = \begin{bmatrix} 1 & & & & & \\ 20 & 2 & & & & \\ & \ddots & \ddots & & & \\ & & 20 & 19 & & \\ & & & 20 & 20 \end{bmatrix} \qquad b = \begin{bmatrix} 20 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}. \tag{3.19}$$

When constructing the controllability matrix of this example we have to take up to power 19 of the matrix A. Let us look at the singular values of the matrix in C1, using MATLAB:

```
A = diag([1:20]) + diag(ones (1,19),-1) * 20;
b = [zeros(19,1);20];
C1 = ctrb(A,b);
svd(C1)
```

which yields

```
ans =
2.6110e+28
2.1677e+28
    .
    .
    .
4.0545e+04
7.8271e+01
2.0000e+01
```

which indicates a singular matrix for all practical purposes. Yet test C2 gives as rank text for eigenvalue 20 (the worst one):

```
svd([a-20*eye(size(a)),b])
ans =
3.6313e+01
     .
     .
     .
5.9214e+00
```

which is far from being singular. □

This shows that test C1 is not that reliable. But test C2 also suffers from difficulties for the above matrix A. Let us perturb A in position $(1, 20)$ with a small value $\delta = 10^{-12}$ then it is well known for this matrix [147, 148, 149] that the eigenvalues $1, 2, \ldots, 20$ change in their first or second digit. As a consequence of this, test C2 will not be computed in the *true* eigenvalues of $\hat{A}$ but in strongly perturbed ones. This is independent of the vector b and will be incurred by stable eigenvalue algorithms as well. If we now take a $\hat{b}$ vector which is such that a particular eigenvalue, say 1, is not controllable, i.e.,

$$rank \left[\hat{A} - I \mid \hat{b} \right] < 20$$

then rank test C2 applied to the *computed* eigenvalue of $\hat{A}$ will give completely erroneous results.

The same can be said about test C3. If $\hat{A}, \hat{b}$ is uncontrollable in eigenvalue $\lambda = 1$, then a feedback f will not be able to move the eigenvalue $\lambda = 1$ in $\hat{A} + \hat{b}f$. Yet since the eigenvalues of A (and $\hat{A}$) are so sensitive it is very unlikely that $\hat{A}$ and $\hat{A} + \hat{b}f$ will both have a very close eigenvalue to $\lambda = 1$.

Table 3.1: .

Eigenvalues $\lambda_i(A)$	Eigenvalues $\mu_1(A + \hat{b}f)$	$K(\hat{b}; A - \lambda_i I)$
-.32985±j1.06242	.99999	.002
.92191±j3.13716	-8.95872±j3.73260	.004
3.00339±j4.80414	-5.11682±j9.54329	.007
5.40114±j 6.17864	-.75203±j14.148167	.012
8.43769±j 7.24713	5.77659±j 15.58436	.018
11.82747±j 7.47463	11.42828±j14.28694	.026
15.10917±j 6.90721	13.30227±j 12.90197	.032
18.06886±j 5.66313	18.59961±j 14.34739	.040
20.49720±j 3.81950	23.94877±j 11.80677	.052
22.06287±j 1.38948	28.45618±j 8.45907	.064
	32.68478	

The following table was obtained for tests C2 and C3 using $\hat{b} = [0, 1, 1, \ldots, 1]^T$ and $\hat{A}$ as in (3.19). Instead of perturbing an element of $\hat{A}$ we perform a random orthogonal state-space transformation Q on the system $(\hat{A}, b)$. The resulting eigenvalues of $\hat{A} + \hat{b}f$ for a random f and the condition numbers of $\left[\hat{b} \mid \hat{A} - \overline{\lambda}_i I \right]$ in the computed eigenvalues $\overline{\lambda}_i$ of $\hat{A}$ are given in Table 3.1. The machine precision for the test was $\epsilon \sim 10^{-8}$ [102].

When using instead of $\overline{\lambda}_i(\hat{A})$, the correct values $\lambda_i(A)$, test C2 gives $\kappa \left[\hat{b} \mid \hat{A} - \lambda_1 I \right] \sim 10^{-8}$ which indicates that the trouble here lies indeed in the eigenvalue sensitivity.

3.4 Staircase form

We now propose a method that avoids several of the above drawbacks and is based on an orthogonal state-space transformation.

Theorem 3.1. There always exist an orthogonal state-space transformation U such that

$$[U^H B \| U^H A U] = \left[\begin{array}{c|c|ccc|c} X_1 & A_{1,1} & A_{1,2} & \cdots & A_{1,k} & A_{1,k+1} \\ 0 & X_2 & A_{2,2} & & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & & \ddots & X_k & A_{k,k} & A_{k,k+1} \\ \hline 0 & \cdots & \cdots & 0 & 0 & A_{k+1,k+1} \end{array}\right] \tag{3.20}$$

where
– $A_{i,i}, i = 1, \ldots, k$, are $\rho_i \times \rho_i$ matrices
– x_i, $i = 1, \ldots, k$, are $\rho_i \times \rho_{i-1}$ matrices of full row rank ρ_i (with $\rho_0 \doteq m$).

As a consequence

$$m = \rho_0 \geqslant \rho_i \geqslant \cdots \geqslant \rho_k > 0$$

and $A_{k+1,k+1}$ is a square matrix of dimension $(n - \sigma_k) \times (n - \sigma_k)$ with $\sigma_k = \sum_{i=1}^{k} \rho_i$.

Proof: This form is a block version of the controller Hessenberg form, stopping as soon as a "zero" pivot is encountered, at step $k + 1$ in this case. The proof is thus constructive. At each step $i = 1, \ldots, k$ a QR factorization is performed of the current B_i matrix yielding X_i of full row rank. If a zero rank matrix B_i is encountered (at step $k+1$ here) then we obtain the form (3.20). If not, then the method terminates with $n = \sigma_k$ and the bottom matrix $A_{k+1,k+1}$ is void. □

A MATLAB code for the above decomposition is given below. It also contains a variant in which the "rank carrying stairs" X_i are themselves in the special form

$$X_i = \left[\begin{array}{cccccc} 0 & \cdots & 0 & \times & \cdots & \times \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & \ldots & \ldots & \ldots & 0 & \times \end{array}\right] \Bigg\} \rho_i. \tag{3.21}$$

One shows that in general this also requires an input transformation V that can be chosen orthogonal as well.

```
function [a1,b1,k,u,v] = stair(a,b,u,v)
%    [A1,B1,K,U,V]=STAIR(A,B,U,V) performs the staircase reduction of
%    the pair (A,B). It updates the state-space transformation U and
%    input transformation V such that the transformed pair (A1,B1) has
%    the typical staircase form (here with 4 "stairs") :
%
```

```
%                                  | X : *  *  *  *  * | } K(1)
%                                  |   : X  *  *  *  * | } K(2)
%       (B1:A1):=(U'*B*V:U'*A*U) = |   :    X  *  *  * | } K(3)
%                                  |   :       X  *  * | } K(4)
%                                  |   :             Z |
%
%    where each "stair" X has full row rank K(i) and has a triangular
%    form (right adjusted). Matrices U and V must be given as input
%    parameters and may be initialized with the identity matrix.
%    In case V is not given, this transformation is not performed
%    and the "stairs" X have just full row rank. The square matrix
%    Z (if present) contains the uncontrollable modes of (A,B).
%
size(b);n=ans(1);m=ans(2);
[q,r,e]=qr(b);b1=r*e';a1=q'*a*q;u=u*q;
k(1)=rank(r);k1=1;k2=k(1)+1;
if (k2 <= n) , b1(k2:n,1:m)=zeros(n-k2+1,m); end
for i=2:n,
  bh=a1(k2:n,k1:k2-1);ah=a1(k2:n,k2:n);[q,r,e]=qr(bh);
  q=[eye(k2-1),zeros(k2-1,n-k2+1);zeros(n-k2+1,k2-1),q];
  a1=q'*a1*q;a1(k2:n,k1:k2-1)=r*e';u=u*q;r=rank(r,eps);
  if (k2+r <= n) , a1(k2+r:n,k1:k2-1)=zeros(n-k2-r+1,k2-k1); end
  if r == 0 , break , end
  k(i)=r;k1=k2;k2=k2+r;
  if k2 > n , break, end
end
kmax=prod(size(k));nmax=sum(k);
if nargin == 4 ,
  k3=nmax;k2=k3-k(kmax)+1;k1=k2;
  if kmax > 1 ,
    for i=kmax:-1:2,
      k1=k1-k(i-1);
      if k2 > k1+1,
         [q,r]=qr(pert(a1(k2:k3,k1:k2-1)));q=pert(q)';
         a1(k1:k2-1,:)=q'*a1(k1:k2-1,:);a1(:,k1:k2-1)=a1(:,k1:k2-1)*q;
         u(:,k1:k2-1)=u(:,k1:k2-1)*q;a1(k2:k3,k1:k2-1)=pert(r);
      end
      k3=k2-1;k2=k1;
    end
    if k3 > k2, b1(k2:k3,:)=q'*b1(k2:k3,:); end
  end
  [q,r]=qr(pert(b1(k2:k3,:)));q=pert(q)';
  v=v*q;b1=b1*q;b1(k2:k3,:)=pert(r);
end
```

History

A nonorthogonal version of this result was first obtained by Tse et al. (1978) [129]. The orthogonal version is due to Van Dooren (1979) [131, 132] and was redrived by several authors in subsequent years (Boley, Eldin, Eising, Konstantinov et.al., Paige, Patel, Skelton et.al., Varga), each with a particular application in mind. □

The staircase form is now shown to be relevant for the determination of the controllability of a pair (A, B).

Theorem 3.2. The rank of the controllability submatrix

$$C^{(j)}(A,B) \doteq [B, AB, \cdots, A^{j-1}B] \tag{3.22}$$

is $\sigma_j = \sum_{i=1}^{j} \rho_i$ and that of the full matrix is σ_k.

Proof. A state space transformation U applied to the system (3.12) results in a left transformation of the matrix $C^{(j)}(A,B)$, since

$$\begin{aligned} C_U^{(j)}(U^HAU, U^HB) &= [U^HB, U^HAUU^HB, \ldots, (U^HAU)^{j-1}U^HB] \\ &= U^H[B, AB, \ldots, A^{j-1}B] \\ &= U^HC^{(j)}(A,B). \end{aligned} \tag{3.23}$$

Since an invertible transformation does not change the rank of a matrix one can as well check the rank of $C_U^{(j)}(U^HAU, U^HB)$ which is of the form

$$C_U = \left[\begin{array}{c|c|c|c|c|c|c} X_{1,1} & \times & \ddots & \times & \times & \ldots & \times \\ 0 & X_{2,1} & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & X_{k,1} & \times & \times & \ldots & \times \\ 0 & \ldots & 0 & 0 & \ldots & \ldots & 0 \end{array}\right] \begin{array}{c} \}\rho_1 \\ \vdots \\ \}\rho_k \end{array} \tag{3.24}$$

where the matrices

$$X_{i,1} \doteq X_i \cdot X_{i-1} \cdots X_1$$

are full row rank since they are the product of full row rank matrices. But the very form of (3.24) indicates that C_u has row rank $\sigma_k = \rho_1 + \cdots + \rho_k$ and similarly the requested result also follows for $C_U^{(j)}$. □

Remark.

a) In the case of $m = 1$, the staircase form is nothing but the controller Hessenberg form where one checks for nonzero diagonal elements along the pivot elements. All ρ_i are thus equal to 1 and k is obtained from the first zero pivot element $x_{k+1} = 0$. The corresponding controllability matrix is then in trapezium form and would have been the result of the QR factorization of

the matrix $C(A,B)$:

$$[\, b_k \| A_k \,] \;=\; \left[\begin{array}{c|ccccc|c} \times_1 & \times & \dots & \dots & \times & & \times \\ 0 & \times_2 & \ddots & \ddots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & & \vdots \\ 0 & \dots & 0 & \times_k & \times & & \times \\ \hline 0 & & & & 0 & & \\ \vdots & \dots & \dots & \dots & \vdots & & A_{k+1,k+1} \\ 0 & & & & 0 & & \end{array}\right]$$

$$\begin{aligned} C(A,b) &= UC_U(A_k,b_k) \\ &= U \left[\begin{array}{cccc|ccc} \times & \times & \dots & \times & \dots & \dots & \times \\ 0 & \times & \ddots & \times & \times & \dots & \times \\ \vdots & \ddots & \ddots & \vdots & & & \\ 0 & \dots & 0 & \times & \times & \dots & \times \\ \hline 0 & \dots & \dots & 0 & 0 & \dots & 0 \end{array}\right] \end{aligned}$$

b) In the multi-put case the form (3.24) is a block version of a QR factorization. It is important to notice here that the rank is found form submatrices of the matrices A and B and that no powers of A have to be computed explicitly. Also no eigenvalues of A are involved here. So this approach avoids the pitfalls of all three methods explained in Section 3.3.

c) The controllability indices $\{c_i, i = 1, \dots, m\}$ defined for multi-input systems can be retrieved from the rank increases $\{\rho_j, j = 1, \dots, k\}$ (see [131, 132, 141]) one proves that they in fact form a dual partition of each other (see [65]).

Example 3.2

Take again the matrices (3.19) of Example 3.4. We see that this pair is already in staircase form with $\rho_1 = \rho_2 = \dots \rho_{20} = 1$ and $\sigma_{20} = 20$. All "stairs" are 20 and clearly nonsingular. So the system is controllable. □

By duality all the above comments about controllability in fact also apply to that of observability by simply interchanging A with A^T and B with C^T. The staircase form for testing observability though, is slightly different since again one prefers the form to be upper triangular. This is given in the following theorem.

Theorem 3.3. There always exists an orthogonal state-space transformation U such that

$$\left[\begin{array}{c} U^H A U \\ \hline\hline CU \end{array}\right] \;=\; \left[\begin{array}{c|ccc} A_{k+1,k+1} & A_{k,k+1} & \cdots & A_{1,k+1} \\ \hline 0 & A_{k,k} & \cdots & A_{1,k} \\ 0 & Y_k & \cdots & A_{1,2} \\ \vdots & \ddots & Y_2 & A_{1,1} \\ \hline\hline 0 & \dots & 0 & Y_1 \end{array}\right]$$

where
– $A_{i.i}$ $i = 1, \ldots, k$ are $\rho_i \times \rho_i$ matrix
– Y_i $i = 1, \ldots, k$ are $\rho_{i-1} \times \rho_i$ matrices of full column ρ_i (with $\rho_0 = p$).

As a consequence

$$p = \rho_0 \geqslant \rho_1 \geqslant \cdots \geqslant \rho_k > 0$$

and $A_{k+1,k+1}$ is a square matrix of dimension $(n - \sigma_k) \times (n - \sigma_k)$ with $\sigma_k = \sum_{i=1}^{k} \rho_i$.

Proof. Dual to that of Theorem 3.1. □

Distance to controllability

So it appears that the staircase form is a reliable *direct method* to test controllability of the multi input system. But the form can still be sensitive to perturbations and may therefore fail in certain pathological cases [131, 132, 141]. The most reliable method for checking controllability is to convert the rank test into a distance problem. Let us define $\mathcal{S}_{\overline{c}}$ to be the set of *uncontrollable* (A, B) *pairs*:

$$\mathcal{S}_{\overline{c}} \quad = \quad \{(A, B) \mid \text{rank} C(A, B) < n\}\,.$$

Then the following distance function

$$\mu \quad = \quad \min\{\|[\Delta A \mid \Delta B]\| \mid (A + \Delta A, B + \Delta B) \in \mathcal{S}_{\overline{c}}\}$$

gives the distance of the pair (A, B) to the set of uncontrollable pairs. By definition this is a *robust measure* of controllability since a small perturbation in (A, B) give a small perturbation in μ (prove this!). This definition was introduced by Eising [29] was discerns between the *real* distance μ_r and *complex* distance μ_c by allowing only real or complex perturbations [29].

Exercise 3.2

Prove that both μ_r and μ_c are robust measures and that $\mu_c \leqslant \mu_r$.
Use the staircase form to derive simple upper bounds for μ_r and μ_c. □

To compute μ_r and μ_c one has to solve a minimization problem for which various algorithms exist [12, 30]. Below we give a MATLAB code of the method of Boley which is based on eigenvalue computations of an embedded square system. We refer to [12] for more details on the method.

```
function [upper2,upper1,lower0,location,radius]=dist(F,G);
%
% function [upper2,upper1,lower0,location,radius]=dist(F,G);
% find distance from (F,G) to nearest uncontrollable pair
% according to method of Boley SIMAX Oct 1990.
%

%form matrices used in paper.
A=[F';G'];
[nrows,ncols]=size(A);
E=eye(nrows);
B=E(:,1:ncols);
D=E(:,ncols+1:nrows);
```

```
% get random orthonormal columns to append
C=rand(nrows,nrows-ncols)-.5*ones(nrows,nrows-ncols);
[Q,R]=qr(C);
C=Q(:,1:nrows-ncols);

%get eigenvalues/vectors;
[V,Lambda]=eig([A,C]);

%scale eigenvectors to "optimize" condition number
for i=1:nrows,
    V(:,i)=V(:,i)/norm(V(:,i));
    end;
K=cond(V);

%separate head and tail from each vector
X=V(1:ncols,:);
Y=V(ncols+1:nrows,:);

%compute easy upper bound beta1 for every case and extract smallest one.
for i=1:nrows,
    upp0(i)=norm(C*Y(:,i)-Lambda(i,i)*D*Y(:,i))/norm(X(:,i));
    end;
[SortedUpp0,indices]=sort(upp0);
first=indices(1);

%compute all the other bounds based on the one picked above.
location=Lambda(first,first);
upper1=upp0(first);
ans=svd(A-location*B);
upper2=ans(ncols);
lower0=upper2/(K+1);
radius=upper2*K;
```

3.5 Subspaces and Minimal Realizations

For a discrete time system

$$\left\{ \begin{array}{rcl} x_{k+1} & = & Ax_k + Bu_k \\ y_k & = & Cx_k + Du_k \end{array} \right. \tag{3.25}$$

one can also associate certain spaces with the input to state map and with the state to output maps. One defines the following geometrical concepts to these respective mappings:

$$\begin{array}{rcl} \{u_k\} & \Longrightarrow & \{x_k\} : \text{controllability, reachability} \\ \{x_k\} & \Longrightarrow & \{y_k\} : \text{observability, reconstructability.} \end{array}$$

Precise definition of these concepts are:

Controllable subspace

The set $\mathcal{C}_k(A,B)$ is the set of states $x_0 \in \mathcal{X}$ that can be *driven to zero* in k steps (i.e. $x_k = 0$) by an appropriate choice of inputs $\{u_i,\, i = 0,\ldots,k-1\}$. □

Reachable subspace

The set $\mathcal{R}_k(A,B)$ is the set of states $x_k \in \mathcal{X}$ that can be *reached from the zero initial state* (i.e. $x_0 = 0$) in k steps by an appropriate choice of inputs $\{u_i,\, i = 0,\ldots,k-1\}$. □

Unobservable subspace

The set $\mathcal{O}_k(A,C)$ is the set of states $x_0 \in \mathcal{X}$ that with absence of inputs (i.e. $u_i = 0,\, i = 0,\ldots,k-1$) *will produce zero output* during k steps $\{y_i = 0,\, i = 0,\ldots,k-1\}$. □

Unreconstructable subspace

The set $\mathcal{S}_k(A,C)$ is the set of states $x_k \in \mathcal{X}$ that with absence of inputs (i.e. $u_i = 0,\, i = 0,\ldots,k-1$) *has produced zero output* during k steps $\{y_i = 0,\, i = 0,\ldots,k-1\}$. □

The best way to analyze these concepts is to use again the stacked up notation for inputs and outputs over an interval of time $0,\ldots,k-1$:

$$U_k \doteq \begin{bmatrix} u_0 \\ \vdots \\ u_{k-1} \end{bmatrix}; \qquad Y_k \doteq \begin{bmatrix} y_0 \\ \vdots \\ y_{k-1} \end{bmatrix}. \tag{3.26}$$

As shown earlier, recursive substitution of (3.25) yields

$$\begin{aligned} x_k &= \mathbf{A}_k x_0 + \mathbf{B}_k U_k \\ Y_k &= \mathbf{C}_k x_0 + \mathbf{D}_k U_k \end{aligned} \tag{3.27}$$

where

$$\left[\begin{array}{c|c} \mathbf{A}_k & \mathbf{B}_k \\ \hline \mathbf{C}_k & \mathbf{D}_k \end{array}\right] = \left[\begin{array}{c|cccc} A^k & A^{k-1}B & \cdots & AB & B \\ \hline C & D & & & \\ CA & CB & D & & \\ \vdots & \vdots & & \ddots & \\ CA^{k-1} & CA^{k-2}B & \cdots & \cdots & D \end{array}\right] \tag{3.28}$$

The following theorems now express directly the above spaces in forms of the matrices in (3.27).

Theorem 3.4. The subspaces $\mathcal{R}_k(A,B)$ and $\mathcal{O}_k(A,C)$ are given by

$$\mathcal{R}_k(A,B) = Im\mathbf{B}_k \tag{3.29}$$

$$\mathcal{O}_k(A,C) = Ker\mathbf{C}_k \tag{3.30}$$

Proof. For zero initial state $x_0 = 0$ we find from (3.27)-(3.28)

$$x_k = [A^{k-1}B \ldots AB\ B]U_k.$$

Clearly the states x_k that can be reached from arbitrary U_k is $\mathcal{R}_k(A,B) = Im\mathbf{B}_k$. For $\mathcal{O}_k(A,C)$ there are no inputs (i.e., $U_k = 0$) and hence

$$Y_k = \begin{bmatrix} C \\ CA \\ \ddots \\ CA^{k-1} \end{bmatrix} x_0.$$

Thus the set $\mathcal{O}_k(A,C)$ of initial states x_0 that produce a zero output Y_k is clearly $Ker\mathbf{C}_k$. □

Similarly for the controllable and unreconstructable subspaces we have:

Theorem 3.5. The subspaces $\mathcal{C}_k(A,B)$ and $\mathcal{S}_k(A,C)$ are given by

$$\mathcal{C}_k(A,B) = A^{-k}\mathcal{R}_k(A,B) \tag{3.31}$$
$$\mathcal{S}_k(A,C) = A^k\mathcal{O}_k(A,C). \tag{3.32}$$

Proof. In the equation (3.27) we now want $x_k = 0$ hence

$$0 = A^k x_0 + [A^{k-1}B \ldots AB\ B]U_k$$

with U_k arbitrary. Hence $A^k x_0$ must lie in the image of $[A^{k-1}B \ldots AB, B]$ or $\mathcal{R}_k(A,B)$ which proves (3.31). For (3.32) we have zero input U_k and hence

$$x_k = A^k x_0; \quad Y_k = \begin{bmatrix} C \\ CA \\ \ddots \\ CA^{k-1} \end{bmatrix} x_0.$$

For zero output Y_k we need $x_0 \in \mathcal{O}_k(A,C)$ and the states x_k corresponding to this are clearly given by (3.32). □

From the above theorem it follow that the spaces $\mathcal{R}_k(A,B)$ and $\mathcal{O}_k(A,C)$ are in fact constructed by the staircase algorithm applied to (A,B) and (A^T,C^T), respectively. From the definition it follows easily that the spaces $\mathcal{R}_k(A,B)$ and $\mathcal{C}_k(A,B)$ grow with k, whereas the dual spaces $\mathcal{O}_k(A,C)$ and $\mathcal{S}_k(A,C)$ decrease with k. From the staircase algorithm it also follows that these spaces "converge" to a fixed set at an index k smaller or equal to n. Orthogonal bases for these spaces are in fact constructed by the staircase algorithm as well.

Theorem 3.6. Let ρ_i be the rank increases of the staircase algorithm applied to the pair (A,B) and define $\rho_i = 0$ for $i > k$, the index of the last "stair". Then

$$\mathcal{R}_j(A,B) = U \cdot I_m \begin{bmatrix} I_{\sigma_j} \\ 0 \end{bmatrix} \tag{3.33}$$

with $\sigma_j = \sum_{i=1}^{j} \rho_i$ and U the $n \times n$ transformation constructed by the staircase algorithm of Theorem 3.1.

Proof. This follows from (3.23) indicating that

$$U^H R_j(A,B) = Im \begin{bmatrix} I_{\sigma_j} \\ 0 \end{bmatrix}.$$

and the invertibility of the transformation U. □

The dual result of this is given without proof.

Theorem 3.7. Let ρ_i be the rank increases of the staircase algorithm applied to the pair (A^T, C^T) and define $\rho_i = 0$ for $i > k$, the index of the last "stair". Then

$$\mathcal{O}_j(A, C) \quad = \quad U \cdot Im \begin{bmatrix} I_{\tau_j} \\ 0 \end{bmatrix} \tag{3.34}$$

with $\tau_j = n - \sum_{i=1}^{j} \rho_i$ and U the $n \times n$ transformation constructed by the staircase algorithm of Theorem 3.3.

Proof. Dual to that of the previous theorem. □

Since these spaces converge in less than n steps to the "steady state" results $\mathcal{R}(A, B)$ and $\mathcal{O}(A, C)$ we thus computed orthogonal basis for these respective spaces in $0(n^2(5/3n + m + p))$ operations at most. This follows from the fact that the staircase forms never requires more operations than the corresponding condensed forms.

Remark.

a) The concept of reachability has the same meaning for a continuous time system as that of controllability. For a discrete time system, both concepts are different, though. It would be more appropriate to always use reachability instead of controllability since usually, this is what one refers to.

b) It follows from theorems 3.4 and 3.5 that if A is invertible, then,

$$\begin{aligned} dim\mathcal{C}_k(A, B) &= dim\mathcal{R}_k(A, B) \\ dim\mathcal{S}_k(A, C) &= dim\mathcal{O}_k(A, C). \end{aligned}$$

□

When A is not invertible, the concepts of reachability and constructability can indeed be very different as shown by the following simple example.

Example 3.3

Let

$$A \quad = \quad \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

The matrix $[B \quad AB] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ has image $Im \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and this is the only vector that can be "reached". On the other hand, any vector $x_0 = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ can be driven to zero by an input $u_0 = -(\alpha + \beta)$ as is seen from

$$x_1 \quad = \quad \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} (-\alpha - \beta) = 0.$$

So $\mathcal{C}_1(A_1B) = C^2$. Although the dimensions of $\mathcal{C}_1(A,B)$ and $\mathcal{R}_1(A,B)$ differ, one easily checks also that

$$A\mathcal{C}_1(A,B) \;\; = \;\; \mathcal{R}_1(A,B).$$

□

The above results now lead to the construction of minimal realizations. We first note that the staircase form of applied to (A,B) yields a new system $(\hat{A},\hat{B})$ of the form

$$[U^HB \mid U^HAU] \;\; = [\hat{B} \mid \hat{A}] = \;\; \left[\begin{array}{c|cc} B_r & A_r & \times \\ 0 & 0 & A_{\overline{r}} \end{array}\right] \tag{3.35}$$

where the subsystem (A_r, B_r) is reachable. The number of rows/columns of A_r is σ_k and the first σ_k columns of U span $R(A,B)$. Similarly, the staircase algorithm applied to (A,C) yields a new system $(\hat{A},\hat{C})$ of the form

$$\left[\frac{U^HAU}{CU}\right] \;\; = \;\; \left[\frac{\hat{A}}{\hat{C}}\right] = \left[\begin{array}{cc} A_{\overline{o}} & \times \\ 0 & A_0 \\ \hline 0 & C_0 \end{array}\right] \tag{3.36}$$

where the subsystem (A_0, C_0) is observable. The number of rows/columns of $A_{\overline{o}}$ is τ_k and the first τ_k columns of U span $\mathcal{O}(A,C)$.

Combining intersections and completions of the above decompositions yields a coordinate system $(\hat{A},\hat{B},\hat{C})$ where

$$\left[\begin{array}{c|c} \hat{A} & \hat{B} \\ \hline \hat{C} & D \end{array}\right] \;\; = \;\; \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & B_1 \\ 0 & A_{22} & A_{23} & B_2 \\ 0 & 0 & A_{33} & 0 \\ \hline 0 & C_2 & C_3 & D \end{array}\right]$$

and where the subsystem $\{A_{22}, B_2, C_2, D\}$ is both observable and reachable (i.e., minimal). Moreover the transfer functions of $\{\hat{A},\hat{B},\hat{C},D\}$ and $\{A_{22},B_2,C_2,D\}$ are equal. This is obtained by the following construction. Start with a decomposition (3.35) displaying the reachable part of the system (apply the transformation also to C):

$$\left[\begin{array}{c|c} U_r^HAU_r & U_r^HB \\ \hline CU_r & D \end{array}\right] \;\; = \;\; \left[\begin{array}{cc|c} A_r & \times & B_r \\ 0 & A_{\overline{r}} & 0 \\ \hline C_r & C_{\overline{r}} & D \end{array}\right].$$

Then take the subsystem (A_r, C_r) and perform the decomposition (3.36) (apply again the transformation to B as well):

$$\left[\begin{array}{c|c} U_o^HA_rU_o & U_oB_r \\ \hline C_rU_o & D \end{array}\right] \;\; = \;\; \left[\begin{array}{cc|c} A_{r\overline{o}} & \times & B_{r\overline{o}} \\ 0 & A_{ro} & B_{ro} \\ \hline 0 & C_{ro} & D \end{array}\right]$$

then define

$$U \;\; \doteq \;\; U_r \left[\begin{array}{c|c} U_o & \\ \hline & I_{\overline{r}} \end{array}\right]$$

$$\hat{A} \doteq U^H A U; \quad \hat{B} \doteq U^H B; \quad \hat{C} \doteq CU$$

to obtain

$$\left[\begin{array}{c|c} \hat{A} & \hat{B} \\ \hline \hat{C} & D \end{array}\right] = \left[\begin{array}{ccc|c} A_{r\bar{o}} & \times & \times & B_{r\bar{o}} \\ 0 & A_{ro} & \times & B_{ro} \\ 0 & 0 & A_{\bar{r}} & 0 \\ \hline 0 & C_{ro} & C_{\bar{r}} & D \end{array}\right].$$

The transfer function of such a state space system is:

$$\hat{C}(zI - \hat{A})^{-1}\hat{B} + D = [\, 0 \quad C_{ro} \quad C_{\bar{r}} \,] \begin{bmatrix} zI - A_{r\bar{o}} & \times & \times \\ 0 & zI - A_{ro} & \times \\ 0 & 0 & zI - A_{\bar{r}} \end{bmatrix}^{-1} \begin{bmatrix} B_{r\bar{o}} \\ B_{ro} \\ 0 \end{bmatrix} + D =$$

$$[\, 0 \quad C_{ro} \quad C_{\bar{r}} \,] \begin{bmatrix} (zI - A_{r\bar{o}})^{-1} & \times & \times \\ 0 & zI - {A_{ro}}^{-1} & \times \\ 0 & 0 & (zI - A_{\bar{r}})^{-1} \end{bmatrix}^{-1} \begin{bmatrix} B_{r\bar{o}} \\ B_{ro} \\ 0 \end{bmatrix} + D =$$

$$C_{ro}(zI - A_{ro})^{-1}B_{ro} + D,$$

which finally leads to the following theorem.

Theorem 3.8. (Kalman Decomposition) There always exists an orthogonal state space transformation U such that

$$\left[\begin{array}{c|c} U^H A U & U^H B \\ \hline CU & D \end{array}\right] = \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & B_1 \\ 0 & A_{22} & A_{23} & B_2 \\ 0 & 0 & A_{33} & 0 \\ \hline 0 & C_2 & C_3 & D \end{array}\right].$$

where A_{11} contains unobservable modes, A_{33} contains unreachable modes and the subsystem (A_{22}, B_2, C_2) is both reachable and observable. Moreover the latter subsystem has the same transfer function as the original system (A, B, C). □

3.6 Poles and Zeros

Poles and zeros are rather classical concepts when single input single output systems are concerned. They are then typically defined in terms of the transfer function of the system:

$$y(\cdot) = h(\lambda) \cdot u(\cdot). \tag{3.37}$$

For the class of systems we consider $h(\lambda)$ is a rational function

$$h(\lambda) = n(\lambda)/d(\lambda) \tag{3.38}$$

and the poles are thus the zeros of the polynomial $d(\lambda)$, while the zeros are the zeros of the polynomial $u(\lambda)$. A natural way to extend this to multi-input multi-output systems with transfer function

$$y(\cdot) = H(\lambda)u(\cdot) \tag{3.39}$$

is to say that the poles of $H(\lambda)$ are those values of λ in the complex phase where (some entry of) $H(\lambda)$ becomes infinite, whereas the zeros of $H(\lambda)$ are those values of λ in the complex plane where the rank of $H(\lambda)$ drops below its normal value. It is easy to see that this reduces to the definition for the scalar case.

But this definition fails in identifying coalescent poles and zeros and also fails in giving more detailed information about multiplicities of poles and zeros. A precise definition is based on a decomposition that essentially reduces the matrix case to the scalar case.

Theorem 3.9. (McMillan Form) Every rational transfer matrix $H(\lambda)$ has a decomposition into the product of matrices:

$$H_{pm}(\lambda) \quad = \quad M_{pp}(\lambda) D_{pm}(\lambda) N_{mm}(\lambda) \tag{3.40}$$

where $M(\lambda)$ and $N(\lambda)$ are polynomial matrices with constant determinant (i.e., unimodular) and

$$D(\lambda) \quad = \quad \left[\begin{array}{ccc|c} \frac{e_1(\lambda)}{f_1(\lambda)} & & & \\ & \ddots & & 0 \\ & & \frac{l_r(\lambda)}{f_r(\lambda)} & \\ \hline & & & 0_{p-r,m-r} \end{array}\right] \tag{3.41}$$

with $e_i(\lambda)$ and $f_i(\lambda)$ polynomials that divide each other as follows

$$\left\{\begin{array}{l} e_{i+1}(\lambda) \text{ divides } e_i(\lambda) \\ f_{i+1}(\lambda) \text{ divides } f_i(\lambda) \end{array}\right. \quad \text{for } i = 1, \ldots, r-1. \tag{3.42}$$

Proof. See [65] for a constructive proof. □

Notice that in this decomposition $M(\lambda)$ and $N(\lambda)$ are invertible and finite for any λ in the complex plane. So if $H(\lambda)$ has poles and zeros they must be poles and zero of the quasidiagonal matrix $D(\lambda)$. This thus leads to the following definitions.

Definition 3.1. The zeros of $H(\lambda)$ are the zeros of the polynomials $e_i(\lambda)$, $i = 1, \ldots, r$. The multiplicities of each zero λ_0 are those of λ_0 in each polynomial $e_i(\lambda)$. □

Definition 3.2. The poles of $H(\lambda)$ are the zeros of the polynomials $f_i(\lambda)$, $i = 1, \ldots, r$. The multiplicities of each pole λ_0 are those of λ_0 in each polynomial $f_i(\lambda)$. □

This definition is very elegant and close to that of the scalar case, but unfortunately it relies on the McMillan form which requires unstable transformations on the coefficients of rational matrices (see [136, 137]). The following theorem now reduces everything to that of computing eigenvalues and their multiplicities. The new definition are based on minimal realizations and on corresponding eigenvalue and generalized eigenvalue problems.

Theorem 3.10. Let $\{A, B, C, D\}$ be a *minimal* realization of the transfer function $H(\lambda)$, i.e.,

$$H(\lambda) \quad = \quad C(\lambda I - A)^{-1} B + D \tag{3.43}$$

where (A, B) is controllable (reachable) and (A, C) observable. Then the poles of $H(\lambda)$ are the eigenvalues of A and their mulplicities are the lengths of the corresponding Jordan chains; and the zeros of $H(\lambda)$ are the generalized eigenvalues of

$$S(\lambda) \quad = \quad \left[\begin{array}{c|c} \lambda I - A & B \\ \hline -C & D \end{array}\right] \tag{3.44}$$

and their multiplicities are the lengths of the corresponding Jordan chains.

Proof. See [114] and [143] for more details. □

It is interesting to notice that this definition has a nice interpretation in terms of eigenfrequencies "undriven" by the input and eigenfrequencies "blocked" at the output. The most natural way to describe this is in the continuous time case. So consider the system (3.43) where $\lambda = \frac{d}{dt}$.

Then the *poles of the system* are the frequencies of the eigensolutions undriven by $u(t)$, but appearing in $y(t)$. So since $u(t) = 0$ we have

$$\dot{x}(t) = Ax(t); \qquad x(0) = x_0.$$

If we choose x_0 and eigenvector of A with eigenvalue λ_0 then $Ax_0 = \lambda_0 x_0$ and

$$x(t) = x_0 e^{\lambda_0 t}; \qquad y(t) = Cx_0 e^{\lambda_0 t}.$$

For the zeros we consider

$$\left[\frac{0}{y(t)}\right] = \left[\begin{array}{c|c} \frac{d}{dt}I - A & B \\ \hline -C & D \end{array}\right] \left[\frac{-x(t)}{u(t)}\right]$$

and require $y(t) = 0$. So choosing $\left[\frac{-x_0}{u_0}\right]$ as a generalized eigenvector of $S(\lambda)$ with generalized eigenvalue λ_0, then

$$\left[\frac{-x(t)}{u(t)}\right] = \left[\frac{-x_0}{u_0}\right] e^{\lambda_0 t}$$

and

$$\left[\begin{array}{cc} \lambda_0 I - A & B \\ -C & D \end{array}\right] \left[\frac{-x_0}{u_0}\right] = 0.$$

This interpretation only agrees with the definitions given earlier when the transfer function $H(\lambda)$ is square and invertible. In that case we also have that $S(\lambda)$ is square and invertible since

$$\left[\begin{array}{c|c} I & 0 \\ \hline -C(\lambda I - A)^{-1} & I \end{array}\right] S(\lambda) = \left[\begin{array}{c|c} \lambda I_n - A & B \\ \hline 0 & H(\lambda) \end{array}\right]. \tag{3.45}$$

So rank drops in $S(\lambda)$ and $H(\lambda)$ must occur at the same points λ_0, as long as they are not eigenvalues of A (or poles of $H(\lambda)$). In this case we have that

$$\det . S(\lambda) \not\equiv 0$$

and since the coefficient of λ is $\left[\begin{array}{cc} I_n & 0 \\ 0 & 0 \end{array}\right]$, this is a polynomial of degree at most n. If the constant matrix D is invertible, then one sees that

$$S(\lambda) \left[\begin{array}{c|c} I & 0 \\ \hline D^{-1}C & I \end{array}\right] = \left[\begin{array}{c|c} \lambda I_n - A + BD^{-1}C & B \\ \hline 0 & D \end{array}\right] \tag{3.46}$$

and $\det .S(\lambda) = \det .D$, $\det .(\lambda I - (A - BD^{-1}C))$ which implies that the zeros of $\det .S(\lambda)$ are the eigenvalues of the matrix

$$\tilde{A} \quad = \quad A - BD^{-1}C. \tag{3.47}$$

A better way to compute the zeros of $S(\lambda)$, through, is to use a unitary transformation on $S(\lambda)$, rather than the invertible transformation (3.46).

One can always construct a transformation W which is orthogonal and reduces the columns of the matrix $[\ -C \mid D\ ]$ to $\left[\ 0 \mid \hat{D}\ \right]$ where $\hat{D}$ is of full column rank. Applying this transformation to $S(\lambda)$ yields:

$$S(\lambda)W = \left[\begin{array}{c|c} \lambda I - A & B \\ \hline -C & D \end{array}\right] \qquad W = \left[\begin{array}{c|c} \lambda\hat{E} - \hat{A} & x \\ \hline 0 & \hat{D} \end{array}\right] \tag{3.48}$$

where $\hat{D}$ is square invertible since $S(\lambda)$ is. One shows (see [141, 133]) that in fact

$$(\lambda I - \tilde{A}) \quad = \quad (\lambda\hat{E} - \hat{A})\hat{E}^{-1} \tag{3.49}$$

and that the eigenvalues of $\tilde{A}$ are in fact the generalized eigenvalues of the pencil $(\lambda\hat{E} - \hat{A})$. These are then found via the QZ algorithm:

$$Q^H(\lambda\hat{E} - \hat{A})Z \quad = \quad \lambda \begin{bmatrix} e_{11} & \times & \dots & \times \\ & \ddots & \ddots & \vdots \\ & & \ddots & \times \\ & & & e_{nn} \end{bmatrix} - \begin{bmatrix} a_{11} & \times & \dots & \times \\ & \ddots & \ddots & \vdots \\ & & \ddots & \times \\ & & & a_{nn} \end{bmatrix} \tag{3.50}$$

and the solutions of $\det .(\lambda\hat{E} - \hat{A})$ are obviously the ratios $\lambda_i = a_{ii}/e_{ii}$. Notice that if $\det .S(\lambda) \not\equiv 0$ then it can not occur that both a_{ii} and e_{ii} are simultaneously zero, so the ratio λ_i is well defined. But λ_i can be infinite if $e_{ii} = 0$, and it is known [32] that this occurs when the matrix D is singular. In such case there will be less than n finite zeros.

If we want to extend the above results to arbitrary systems $\{A, B, C, D\}$ then the above approach does not apply anymore. Yet Theorem 3.10 indicates that the generalized eigenvalues of $S(\lambda)$ are the finite zeros of the system. The following staircase like decomposition now reduces the general problem to one we can solve again via QZ techniques.

Theorem 3.11. Any system matrix

$$S(\lambda) = \left[\begin{array}{c|c} \lambda I - A & B \\ \hline -C & D \end{array}\right]$$

can be transformed via orthogonal row and column transformations to

$$VS(\lambda)W = \begin{bmatrix} \boxed{X_1} & \times & \dots & & & \dots & \times \\ 0 & \ddots & & & & & \vdots \\ \vdots & & \boxed{X_k} & & & & \\ & & & \left[\begin{array}{c|c} \lambda I - A_f & B_f \\ \hline -C_f & D_f \end{array}\right] & & & \\ & & & & \boxed{Y_\ell} & & \vdots \\ \vdots & & & & & \ddots & \times \\ 0 & \dots & & & \dots & 0 & \boxed{Y_1} \end{bmatrix}. \tag{3.51}$$

where the X_i are full row rank matrices, the Y_i are full column rank matrices, and D_f is invertible.

Moreover the finite zeros of $S(\lambda)$ are those of

$$S_f(\lambda) \;=\; \left[\begin{array}{c|c} \lambda I - A_f & B_f \\ \hline -C_f & D_f \end{array}\right]$$

and are the eigenvalues of $A_f - B_f D_f^{-1} C_f$.

Proof. See [32]. □

This algorithm is also implemented in MATLAB in the function `tzero.m((A, B, C, D)` which performs the preliminary reduction (3.51) and then uses (3.48) and (3.50) to find the finite generalized eigenvalues of $\{A, B, C, D\}$. Below is a practical example which was solved via this technique.

Example 3.3

This is a 9th order model of a boiler [3].

$$A = \begin{bmatrix} \text{-3.93} & \text{-3.15E-3} & 0 & 0 & 0 & \text{4.03E-5} & 0 & 0 & 0 \\ \text{3.68E2} & \text{-3.05} & \text{3.03} & 0 & 0 & \text{-3.77E-3} & 0 & 0 & 0 \\ \text{2.74E1} & \text{7.87E-2} & \text{-5.96E-2} & 0 & 0 & \text{-2.81E-4} & 0 & 0 & 0 \\ \text{-6.47E-2} & \text{-5.20E-5} & 0 & \text{-2.55E-1} & \text{-3.35E-6} & \text{3.60E-7} & \text{6.33E-5} & \text{1.94E-4} & 0 \\ \text{3.85E3} & \text{1.73E1} & \text{-1.28E1} & \text{-1.26E4} & \text{-2.91} & \text{-1.05E-1} & \text{1.27E1} & \text{4.31E1} & 0 \\ \text{2.24E4} & \text{1.80E1} & 0 & \text{- 3.56E1} & \text{-1.04E-4} & \text{-4.14E-1} & \text{9.00E1} & \text{5.69E1} & 0 \\ 0 & 0 & \text{2.34E-3} & 0 & 0 & \text{2.22E-4} & \text{-2.03E-1 0} & 0 & \\ 0 & 0 & 0 & \text{-1.27} & \text{-1.00E-3} & \text{7.86E-5} & 0 & \text{-7.17E-2} & 0 \\ \text{-2.20} & \text{-1.77E-3} & 0 & \text{-8.44} & \text{-1.11E-4} & \text{1.38 E-5} & \text{1.49E-3} & \text{6.02E-3} & \text{-1.00E-10} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1.56 & 0 \\ 0 & -5.13E-6 \\ 8.28 & -1.55 \\ 0 & 1.78 \\ 2.33 & 0 \\ 0 & -2.45E-2 \\ 0 & 2.94E-5 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad D = 0.$$

The computed zeros and the relative backward residuals $\sigma_{11}(s(\lambda_i))/\sigma_1(S(\lambda_i))$ were

Zeros	$\frac{\sigma_{11}}{\sigma_1}$
-26.39513728882773	$< \epsilon$
-2.957771985292096 $\pm$ .3352657201870191	$< \epsilon$
.74860641907556	$< \epsilon$
.09403261020202233	$< \epsilon$
- .009546070612163396	$< \epsilon$

Notice that there are only 6 finite zeros. Also, the normal rank of the transfer function is 2 and that of $S(\lambda)$ is 11. So, in order to check the reliability of the computed eigenvalues λ_i we check if the 11-th singular value $\sigma_{11}(S(\lambda_i)$ is indeed ϵ small with respect to the 1st singular value $\sigma_1(S(\lambda_i)$. For each of the computed eigenvalues, this was the case □

Remarks.

1. Because of the use of orthogonal transformations only, we can prove that the above algorithm for computing zeros is backward stable. Moreover, its complexity is $O(n(n+m)(n+p))$ flops (see [32]).

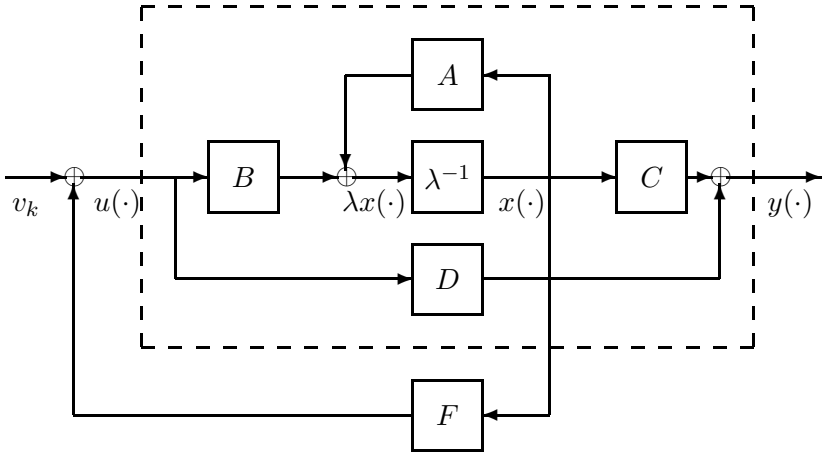
2. The method can be applied to systems $\{A, B, C, D\}$ where $m \neq p$, i.e., with different number of inputs and outputs. In fact, when $m = 0$ it actually performs the staircase decomposition (3.27) and finds the eigenvalues of $A_{\overline{o}}$ in (3.40), i.e., the unobservable modes of the system. When $p = 0$ it finds the staircase decomposition (3.20) and finds the eigenvalues of $A_{\overline{r}}$ in (3.33), i.e., the unreachable modes of the system. Notice that in both cases the system does not need to be minimal.

Chapter 4

STATE SPACE DESIGN

4.1 State feedback and pole placement

A typical "control action" used to influence the behaviour of the system, is *linear state feedback*. It consists of "feeding back" a linear function $Fx(\cdot)$ of the state to the input $u(\cdot)$:



If we redefine the input $u(\cdot)$ as $Fx(\cdot) + u(\cdot)$, then the system becomes:

$$\begin{cases} \lambda x(\cdot) &= (A+BF)x(\cdot) + Bu(\cdot) \\ y(\cdot) &= (C+DF)x(\cdot) + Du(\cdot) \end{cases} . \tag{4.1}$$

This control action is typically used to modify the dynamics of the system, and more particularly, the eigenvalues of the matrix $A + BF$. The problem to be solved in this context is :

Given (A, B), what eigenvalues of $A + BF$ can be obtained by an appropriate choice of F? Moreover, what Jordan structure of the multiple eigenvalues can be obtained?

First we notice that a similarity transformation T does not affect the solution of this problem. Indeed, the feedback $F_t \doteq FT$ applied to the transformed pair

$$(A_t, B_t) \doteq (T^{-1}AT, T^{-1}B) \tag{4.2}$$

yields the closed loop matrix

$$A_t + B_t F_t \;=\; T^{-1}(A+BF)T \tag{4.3}$$

which has the same eigenvalues and Jordan structure as $A + BF$ since there is only a similarity transformation between both. We can therefore choose T so that the new coordinate system simplifies the construction of F_t (and F) and also makes explicit its possible solution(s). Let us start to put (A, B) in the form

$$[A_t \mid B_t] \quad = \quad \left[\begin{array}{cc|c} A_r & \times & B_r \\ 0 & A_{\overline{r}} & 0 \end{array}\right] \tag{4.4}$$

where $A_{\overline{r}}$ contains the unreachable modes and (A_r, B_r) is reachable (T can be chosen unitary). Then one easily sees that $F_t = [F_r \mid F_{\overline{r}}]$ can only affect the spectrum of A_r:

$$[A_t + B_t F_t] = \left[\begin{array}{cc} A_r + B_r F_r & \times \\ 0 & A_{\overline{r}} \end{array}\right]. \tag{4.5}$$

While the eigenvalues of $A_{\overline{r}}$ are unchanged, the corresponding eigenvectors can be modified [70]. We show below that by construction that the spectrum of $A_r + B_r F_r$ can always be chosen arbitrarily, but we will indicate constraints on the choice of Jordan structure. We first consider the single input case, where (A_t, b_t) is in controller canonical form:

$$A_t = \left[\begin{array}{ccccc} -a_{n-1} & -a_{n-2} & \cdots & -a_1 & -a_0 \\ 1 & 0 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & 1 & 0 \end{array}\right], b_t = \left[\begin{array}{c} b_0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{array}\right] \tag{4.6}$$

with characteristic polynomial

$$\det(zI_n - A_t) \quad = \quad z^n + a_{n-1}z^{n-1} + \cdots + a_1 z + a_0 \doteq a(z). \tag{4.7}$$

It is clear that when choosing

$$f \quad = \quad \frac{1}{b_0}\left[a_{n-1} - f_{n-1}, \ldots, a_0 - f_0\right] \tag{4.8}$$

then $A_t - b_t f_t$ has the same form as A_t, but with a_i replaced by f_i. Hence

$$\det(zI_n - (A_t + b_t f_t)) \quad = \quad z^n + f_{n-1}z^{n-1} + \cdots + f_0 \doteq f(z). \tag{4.9}$$

This discussion leads to the following theorem.

Theorem 4.1. Let (A, b) be a reachable single input pair, then a feedback vector f can be found to assign any desired spectrum to $A + bf$. Multiple eigenvalues of $A + bf$ correspond to a single Jordan block and if A, b, f are real then only a spectrum that is symmetric to the real axis $\Lambda = \overline{\Lambda}$ can be assigned.

Proof. The part of the theorem related to Λ is obvious from (4.9) : any desired characteristic polynomial can be assigned and if A, b, f are real then the polynomial is real as well which implies $\Lambda = \overline{\Lambda}$. For multiple eigenvalues in $A_f = A + bf$ we point out that in an appropriate coordinate system A_f is a companion matrix and that therefore $A_f - \lambda_0 I$ has rank $n - 1$ for any eigenvalue λ_0 of A_f. Hence there is only one eigenvector corresponding to λ_0 and one Jordan block as well. □

The above discussion also suggests a simple method for constructing f. The method is known as Ackerman's formula and is also implemented in MATLAB in the function `acker(A,b,l)`. The algorithm goes as follows. First construct the controllability matrix

$$T_1 \quad = \quad [b \; Ab \; \cdots \; A^{n-1}b] \tag{4.10}$$

which transforms (A, b) to

$$(A_1, b_1) \quad \doteq \quad \left[T_1^{-1}AT_1 \mid T_1^{-1}b\right] = \left[\begin{array}{cccc|c} & & & -a_0 & 1 \\ 1 & & & \vdots & 0 \\ & \ddots & & \vdots & \vdots \\ & & 1 & -a_{n-1} & 0 \end{array}\right] \tag{4.11}$$

giving the characteristic polynomial of A. Then transform this with

$$T_2 \quad = \quad \left[\begin{array}{cccc} 1 & a_{n-1} & \dots & a_1 \\ & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1} \\ & & & 1 \end{array}\right] \tag{4.12}$$

to

$$(A_2, b_2) \quad = \left[T_2^{-1}A_1T_2 \mid T_2^{-1}b_1\right] \quad = \left[\begin{array}{cccc|c} a_{n-1} & \dots & \dots & -a_0 & 1 \\ 1 & & & & 0 \\ & \ddots & & & \vdots \\ & & \ddots & & \vdots \\ & & & 1 & 0 \end{array}\right]. \tag{4.13}$$

Now compute f_2 in this coordinate system and transform back to $f = f_2T_2^{-1}T_1^{-1}$.

This is a typical example of the use of canonical forms and badly conditioned transformations in order to construct the solution of a problem in a trivial coordinate system. It is well known that the transformation T_1 can be very badly conditioned for random system of moderate size. Although the problem is trivial in this coordinate system, the transformation has turned a possibly insensitive problem into a very sensitive one. So we recommend instead to use the staircase form

$$[A_t, b_t] \quad = \quad \left[\begin{array}{ccccc|c} \times & \dots & \dots & \dots & \times & \times_1 \\ \times_2 & \ddots & & & \vdots & 0 \\ 0 & \ddots & \ddots & & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & \times_n & \times & 0 \end{array}\right]. \tag{4.14}$$

A very simple method to solve the pole placement problem is to compute the eigenvectors x_i of

$$A_t + b_tf_t \quad = \quad X\Lambda X^{-1} \tag{4.15}$$

first. The equation

$$(A_t + b_t f_t)x_i \quad = \quad x_i \lambda_i \tag{4.16}$$

indeed defines x_i from its last $(n-1)$ rows, which do *not* contain $b_t f_t$. So we have:

$$\underbrace{\begin{bmatrix} \times_2 & \times - \lambda_i & \dots & \times \\ & \ddots & \ddots & \vdots \\ & & \times_n & \times - \lambda_i \end{bmatrix}}_{n} x_i \quad = \quad \tilde{A}_i x_i = 0. \tag{4.17}$$

Since $\tilde{A}_i$ is a rank $n-1$ matrix, this defines x_i up to a scale factor, which we choose such that $\|x_i\| = 1$. This normalized vector is easily computed from the RQ decomposition of $\tilde{A}_i$:

$$\tilde{A}_i \quad = \quad \begin{bmatrix} 0 & \times & \cdots & \times \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \times \end{bmatrix} Q_i^H = \begin{bmatrix} 0 & & \\ \vdots & & R_i \\ 0 & & \end{bmatrix} Q_i^H \tag{4.18}$$

The first column of Q_i is indeed in the null space of A_i and has norm 1. Decomposition (4.18) only requires $n-1$ Givens rotations and $2n^2$ flops, so computing all eigenvectors requires $2n^3$ flops. The feedback vector f is then solved from the top row of

$$A_t - X \Lambda X^{-1} \quad = \quad b_t f_t \tag{4.19}$$

yielding

$$f_t \quad = \quad \times_1^{-1} e_1^T (A_t - X \Lambda X^{-1}). \tag{4.20}$$

We illustrate the relative performance of these two simple algorithms in an example.

Example 4.1.

Take a random system (A, B) (n=10) and assign random eigenvalues L symmetric with the real axis

$$L \quad = \quad \begin{bmatrix} -0.5013 \pm 0.6173i \\ -0.3504 \pm 0.3844i \\ -1.1269 \pm 0.1590i \\ -0.0905 \pm 0.0763i \\ -0.1749 \pm 0.0403i \end{bmatrix}.$$

First use Ackerman's formula (Control Toolbox):

$$F1 \quad = \quad -acker(A, B, L).$$

The gap between prescribed and computed eigenvalues is:

$$L1 = eig(A + B * F1); norm(sort(L1) - sort(L), 'inf')$$
$$ans = 0.0045$$

which shows computed eigenvalues of poor quality! Then use the method computing the eigenvector matrix X of the matrix $(A + BF)$ to be constructed. This is implemented in the MATLAB file rplace.m :

$$[F2, X] = rplace(A, B, L); F2$$

The ordered difference is this time:

$$L2 = eig(A + B * F2); norm(sort(L2) - sort(L), 'inf')$$
$$ans = 2.510210^{-10}$$

which shows much better results! This example is rather difficult for two reasons. The eigenvalues of A are "stiff" as seen from the following ratio :

$$abs(eig(A)); stiff = max(ans)/min(ans)$$
$$stiff = 17.5797$$
$$ctr = cond(ctrb(A, B)), cx = cond(X)$$
$$ctr = 7.174610^{9}$$
$$cx = 1.897810^{6}.$$

The controllability matrix is then poorly conditioned and Ackerman's method usually fails for $n > 10$. The sensitivity of the eigenvalues after placement is also large:

$$sens = 1.0\ 10^{6} \begin{bmatrix} 0.0011 \\ 0.0011 \\ 0.0026 \\ 0.0026 \\ 0.0196 \\ 0.0196 \\ 0.3441 \\ 0.3441 \\ 1.1930 \\ 1.1930 \end{bmatrix}$$

which explains the loss of at least 6 digits using any method as is also observed in the eigenvector method. □

The above eigenvector method uses the orthogonal decomposition (4.14) but then computes the resulting feedback using the Jordan form (4.15). This of course is from a numerical point of view not that appealing when the decomposition is sensitive. Moreover, it certainly fails when requiring multiple eigenvalues since these automatically belong to a single Jordan block as pointed out by Theorem 4.1. The following modification, due to [91, 92] avoids these drawbacks.

Algorithm 4.1. Put (A, b) in the form (4.14). Construct $\tilde{A}_1$ as in (4.17) and then decompose $\tilde{A}_1$ as in (4.18). Since $A - bf - \lambda_1 I$ is singular, so must be

$$(A - bf - \lambda_1 I)Q_1 = \begin{bmatrix} a_1 & \times \cdots \times \\ 0 & \\ \vdots & R_i \\ 0 & \end{bmatrix} \tag{4.21}$$

where a_1 depends linearly on f. Choose f_1 such that $a_1 = 0$. Using the first column x_1 of Q, this is:

$$e_1^T(A - \lambda_1 I)x_1 - e_1^T b f x_1 \quad = \quad 0 \tag{4.22}$$

or taking $f = \beta_1 x_1$ we have $\beta_1 = e_1^T(A - \lambda_1 I)x_1 / e_1^T b$. One then shows that

$$Q_1^H(A - bf)Q_1 \quad = \quad \left[\begin{array}{c|cccc} \lambda_1 & \times & \cdots & \cdots & \times \\ \hline 0 & \times & \cdots & \cdots & \times \\ 0 & \times_3 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \times_n & \times \end{array}\right] \tag{4.23}$$

$$Q_1^H b \quad = \quad \left[\begin{array}{c} \times \\ \hline \times_1 \\ 0 \\ \vdots \\ 0 \end{array}\right] \tag{4.24}$$

with $\times_i \neq 0$ for $i = 2, \ldots, n$. So the bottom $(n-1) \times (n-1)$ subsystem is again in staircase form and the next eigenvalue can be assigned using the same procedure. □

Remarks.

1. The same algorithm was essentially derived independently by several authors in different contexts [91, 92], [93], [73], [131, 132]. Each of these algorithms has a $O(n^3)$ complexity but stability results are not as strong as one would like except when the eigenvalues λ_i are bounded by the norm of the original matrix A [73] [131, 132].

2. The first proposed method [91, 92] does not require the explicit subtraction of λ_i in (4.21) but finds Q_i from implicit shift techniques. It also has an elegant implementation of these ideas in case of complex conjugate roots, by avoiding complex arithmetic altogether.

4.2 Multi-input feedback

Very similar ideas to the above ones apply for the multi input case. The following theorem holds.

Theorem 4.2. Let (A, B) be a reachable multi input pair, then a feedback matrix F exists to assign any desired spectrum Λ to $A_F \doteq A + BF$. If all matrices are real then Λ must be symmetric with respect to the real axis $\Lambda = \hat{\Lambda}$. A multiple eigenvalue λ_0 of A_F corresponds to at most ρ_1 Jordan blocks and

$$\operatorname{rank}(A_F - \lambda_0 I)^i \quad \geqslant \quad n - \sum_{j=1}^{i} \rho_j = n - \sigma_i$$

where ρ_i are the rank increases of the staircase form.

Proof. Follows from Rosenbrock's structure theorem. [114] □

We will see that this theorem has important implications in deadbeat control. Yet in general one does not want multiple eigenvalues since they have infinite sensitivity if associated with a same Jordan block. A method that tries to use the degrees of freedom in this spirit, is due to Kautsky, Nichols and Van Dooren [68]. In multi-input feedback we can minimize the sensitivity $\kappa(\lambda_i)$ of $\lambda_i(A_F)$ versus perturbations in A, B, F. Let x_i and y_i^T be right and left eigenvectors corresponding to the eigenvalue λ_i then the sensitivity of λ_i equals:

$$\kappa(\lambda_i) = \frac{\| x_i \|_2 \| y_i \|_2}{|y_i^T x_i|}. \tag{4.25}$$

Only when $A + BF$ is diagonalizable all $\kappa(\lambda_i)$ are finite:

$$A + BF = X \Lambda X^{-1}$$

where $\Lambda = \text{diag}\{\lambda_1, \ldots, \lambda_n\}$, $X = [x_1 \mid x_2 \mid \cdots \mid x_n]$ and $X^{-1} = Y^T = [y_1 \mid y_2 \mid \cdots | y_n]^T$.

If in the above formula we choose to normalize $\| x_i \|_2 = 1$ then $\kappa(\lambda_i) = \| y_i \|_2$ and a "global" robustness measure is $\| [\kappa(\lambda_1), \ldots, \kappa(\lambda_n)] \|_2 = \| Y \|_F$. Minimizing $\|Y\|_F$ is then obtained by simple updating techniques [68]. The possible choices for x_i are found from the staircase form:

$$\left[\begin{array}{c|c} B_c & A_c \end{array}\right] = \left[\begin{array}{ccc|cccccc} 0 & \text{X} & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \text{X} & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \text{X} & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \text{X} & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & 0 & \text{X} & \times & \times \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{X} & \times \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{X} \end{array}\right] = \left[\begin{array}{c|c} B_1 & A_1 \\ 0 & A_2 \end{array}\right], \tag{4.26}$$

where B_1 is of full row rank ρ_1 and A_2 has $n - \rho_1$ rows. Then for each eigenvector x_i of $A + BF$ we have

$$(A + BF)x_i = \lambda_i x_i$$

and from the bottom $n - \rho_1$ equations, again :

$$x_i \in \mathcal{S}_i \doteq Ker(A_2 - \lambda_i [\, 0 \;\; I \,]). \tag{4.27}$$

Computing an orthogonal basis for each $\mathcal{S}_i$ is cheap when the pair is in staircase form and is done via an RQ decomposition as in (4.18). This requires $O(mn^2)$ for each $\mathcal{S}_i$ and $O(mn^3)$ for all of them. The subsequent optimization for selecting the best x_i in these spaces is of the same order of complexity, and the feedback matrix F is then found in a similar manner to (4.20)

$$F = B^+(A - X\Lambda X^{-1}). \tag{4.28}$$

The following example illustrates the different methods in [68] with earlier optimization methods. Method1 is also implemented in MATLAB in the routine `place.m` of the Control Toolbox.

Example 4.2.

Distillation column (Klein and Moore 1982)

$$n = 5, \qquad m = 2$$

$$A = \begin{bmatrix} -0.1094 & 0.0628 & 0 & 0 & 0 \\ 1.306 & -2.132 & 0.9807 & 0 & 0 \\ 0 & 1.595 & -3.149 & 1.547 & 0 \\ 0 & 0.0355 & 2.632 & -4.257 & 1.855 \\ 0 & 0.00227 & 0 & 0.1636 & -0.1625 \end{bmatrix}$$

$$B^T = \begin{bmatrix} 0 & 0.0638 & 0.0838 & 0.1004 & 0.0063 \\ 0 & 0 & -0.1396 & -0.206 & -0.0128 \end{bmatrix}$$

$$EIG(A) = -0.07732, -0.01423, -0.8953, -2.841, -5.982$$

λ_j	Method 1	Method 2/3	Method KM†	Method GR‡
-0.2	1.8%	1.5%	2.5%	73%
-0.5	0.1%	0.2%	1.2%	85%
-1.0	0.2%	5.0%	0.3%	40%
$-1\pm 1i$	2.4%	1.9%	3.0%	130%

†Klein and Moore (1982)

‡Gourishanker and Ramar (1976)

Percentage errors for .01% perturbation in F.

Methods GR and KM are complex optimization techniques. Methods $1, 2, 3$ are simple updating techniques based on unitary transformations. □

Another way to exploit the degrees of freedom in multi input control is to place all eigenvalues on the same location and choose the Jordan structure. This is done in deadbeat control, which only applies to discrete time systems. There the closed loop equation is

$$x_{k+1} = (A + BF)x_k + Bu_k \tag{4.29}$$

where we require $\lambda(A + BF) = \{0, 0, \ldots, 0\}$. We have the following theorem in this context.

Theorem 4.3. Let ℓ be the size of the largest Jordan block of $A + BF$, then $(A + BF)^\ell = 0$ and $(A + BF)^{\ell+1} \neq 0$. This is also called the index of nilpotency [57] of $A + BF$.

Proof. Write down the Jordan canonical form of $A + BF$:

$$A + BF = T \begin{bmatrix} J_1 & & & \\ & J_\ell & & \\ & & \ddots & \\ & & & J_k \end{bmatrix} T^{-1}$$

where each J_i is a $\ell_i \times \ell_i$ matrix of the form

$$J_i = \left.\begin{bmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix}\right\} \ell_i$$

then

$$\begin{aligned} (A+BF)^\ell &= T \begin{bmatrix} J_1 & & \\ & \ddots & \\ & & J_k \end{bmatrix}^\ell T^{-1} \\ &= T \begin{bmatrix} J_1^\ell & & \\ & \ddots & \\ & & J_k^\ell \end{bmatrix} T^{-1}. \end{aligned}$$

But now $J_i^\ell = 0$ iff $\ell \geqslant \ell_i$ which completes the proof. □

Achieving a small index of nilpotency has the following advantage. Define $A_F = A + BF$ then the response of the closed loop system is

$$x_k = A_F^k x_0 + [A_F^{k-1}B \ \ldots \ A_F B \ \ B] \begin{bmatrix} u_0 \\ \vdots \\ u_{k-1} \end{bmatrix}. \tag{4.30}$$

If now $A_F^\ell = 0$ then only the last ℓ inputs u_{k-i} for $i = 1, \ldots \ell$ affect the state the x_k. All inputs previous to that do not enter into the equation (4.30) because they are "beaten to death" by the transition matrix A_F raised to a power higher than ℓ. This property is applied in observer design as will be shown later on. Theorem 4.2 now says that

$$\operatorname{rank}(A_F - \lambda_0 I)^i \geqslant n - \sum_{j=1}^{i} \rho_j = n - \sigma_i \tag{4.31}$$

for any eigenvalue λ_0 of A_F. So if we want to have $A_F^i = 0$ at the eigenvalue $\lambda_0 = 0$ then clearly we must have

$$n - \sum_{j=1}^{i} \rho_j = 0 \tag{4.32}$$

which is only possible for $i = k$, the last index for which $\rho_k \neq 0$. This value k happens to be the *largest controllability index* of the pair (A, B) (see [65]). The following example illustrates this.

Example 4.3

Consider the pair

$$[A \| B] = \left[\begin{array}{c|cc||cc} 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 2 & 0 & 1 \end{array}\right]$$

which is in staircase from with indices $\rho_1 = 2$, $\rho_2 = 1$. A has eigenvalues $\{0, 1, 2\}$ and we perscribe all $\lambda_i = 0$. Two feedbacks performing this are:

$$F_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \qquad A + BF_1 = \begin{bmatrix} 0 & 1 & \\ & 0 & 1 \\ & & 0 \end{bmatrix}$$

$$F_2 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}, \qquad A + BF_2 = \left[\begin{array}{cc|c} 0 & 1 & \\ & 0 & 0 \\ \hline & & 0 \end{array}\right].$$

Clearly $(A + BF_1)$ has index of nilpotency 3 while $(A + BF_2)$ has index of nilpotency 2 only. An index of nilpotency of 1 is impossible according to (4.31). This would in fact imply $(A + BF) = 0$ which is easily seen to be impossible for this example. □

We thus have proven meanwhile the following Corollary to Theorem 4.2.

Corollary 4.1. The smallest index of nilpotency ℓ of $A + BF$ for a reachable (A, B) pair with staircase sizes ρ_i, $i = 1, \ldots, k$ is k. □

An algorithm for constructing a feedback F that achieves this is described in [134] and is based on a preliminary reduction to staircase form. The algorithm is analogous in spirit to that of Miminis and Paige [91, 92]. In fact, it constructs the minimum norm feedback F that achieves this deadbeat control. The numerical stability (weak form) of this algorithm is also proven in [134].

What can be said about the sensitivity of the deadbeat control problem? According to the results of eigenvalue sensitivity of Jordan block it should be infinite. Indeed, example 1.3 showed that the sensitivity of an eigenvalue 0 of a Jordan block of size ℓ is

$$\kappa(\lambda, J_\ell) \quad = \quad \lim_{\delta \to 0} \frac{\delta}{\sqrt{\ell}\delta^{1/\ell}} = \infty.$$

For the smallest perturbation δ, the multiple eigenvalue "explodes" in a cluster of ℓ eigenvalues at the complex ℓ-th roots $\delta^{1/\ell}$ of δ.

But for the deadbeat control, the key issue is not the location of the eigenvalues, but the fact that $(A + BF)^\ell = 0$. An indication that this property is not as sensitive is that the eigenvalues of $(A + BF)^\ell$ are the ℓ-th powers of those of $A + BF$:

$$\lambda_i[(A + BF)^\ell] = \lambda_i^\ell[A + BF] \approx \delta$$

which thus are again of the order of δ. From this, one proves then that

$$\lim_{\delta \to 0} \frac{\|(A_\delta + B_\delta F_\delta)^\ell\|}{\delta} \quad = \quad c$$

is actually bounded. As a result of this we still have that

$$x_k \quad = \quad [A_F^{\ell-1} B \cdots A_F B \quad B] \begin{bmatrix} u_{k-\ell} \\ \vdots \\ \vdots \\ u_{k-1} \end{bmatrix} + r_k, \quad \|r_k\| \approx \delta$$

i.e., x_k "essentially" depends on the last ℓ inputs only.

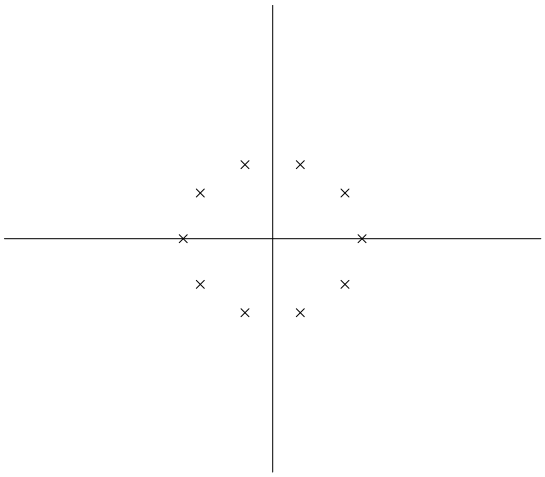


Figure 4.1: Perturbed eigenvalue cluster of a Jordan block.

Example 4.4.

This example takes a random 5×5 matrix A and 5×1 vector b and places a fifth order Jordan block at 0.

```
format long
a=rand(5,5);b=rand(5,1);u=eye(5);v=1;
[a1,b1,k,u0,v0]=stair(a,b,u,v);
[f,u1]=deadbeat(a1,b1,k,u);
e=eig(af1)
naf5=norm(af1^5,'fro')

e =

   1.0e-03 *

 -0.35497696756525 + 0.25791119622289i
 -0.35497696756525 - 0.25791119622289i
  0.13559429295592 + 0.41729341088988i
  0.13559429295592 - 0.41729341088988i
  0.43876534921829


naf5 =

     3.808289238413386e-15
```

Although the absolute values of the eigenvalues are of the order of $\epsilon^{\frac{1}{5}}$ it is clear that $\|A+BF\|$ is of the order of ϵ. □

A MATLAB code for the method constructing this minumum norm deadbeat control F from the staircase form is given below.

```
function [f,u] = deadbeat(a1,b1,kr,u)
%
% DEADBEAT(A,B,K,U)
% Computes the feedback matrix F and the state-space transformation
% U such that A+B*(F*U') is nilpotent with minimal index size(K).
%
% PS : Perform previous to this one [U,V,K,AB]=stair(A0,B0,U0,V0)
%
k=prod(size(kr));size(b1);f=zeros(ans(2),ans(1));dim=ans(1);k1=1;
for i=1:k,
   dimk=dim-k1+1;k2=k1+kr(i)-1;
   [ui,ar]=qr(a1(dim:-1:k1,dim:-1:k1)');
   ui=ui(dimk:-1:1,dimk:-1:1);a1(:,k1:dim)=a1(:,k1:dim)*ui;
   ai=a1(k1:k2,k1:k2);bi=b1(k1:k2,:);g=-pinv(bi)*ai;
   a1(:,k1:k2)=a1(:,k1:k2)+b1*g;a1(k1:dim,:)=ui'*a1(k1:dim,:);
   b1(k1:dim,:)=ui'*b1(k1:dim,:);u(:,k1:dim)=u(:,k1:dim)*ui;
   f(:,k1:k2)=g;k1=k2+1;
end
```

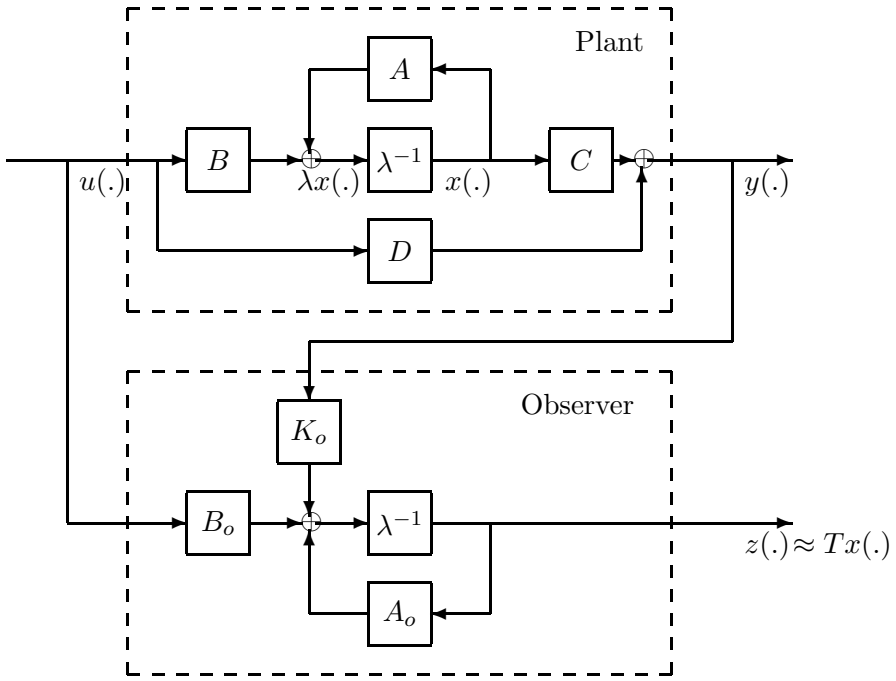
Remarks.

1. The sensitivity of the pole placement problem depends on what is called the solution of the problem. Is it F the feedback matrix, or $A+BF$, the closed loop matrix, or $\Lambda(A+BF)$ the closed loop spectrum. In either of the three cases one can analyze the sensitivity of the function that calculates the solution by looking at the Fréchet derivative. Bounds for this have been derived in [87], [88], [74], [109].

2. Since the pole placement problem is so sensitive, it makes more sense to solve a robust version of it, which is to construct F such that the eigenvalues of $A+BF$ lie in a prescribed set (say eigenvalues with a given decay rate). Techniques that fall in this class are H_∞ control and stability radius minimization [19], [100]. These methods typically guarantee a certain robustness margin of the problem.

4.3 Observers

In many instances the state is not available for feedback and one has to estimate it first. This is done via an observer, which uses the input and output of the plant to try to reconstruct its state. The observer itself is also a dynamical system. The corresponding equations are



$$\begin{cases} \lambda x(\cdot) &= Ax(\cdot) + Bu(\cdot) \\ y(\cdot) &= Cx(\cdot) + Du(\cdot) \\ \lambda z(\cdot) &= A_o z(\cdot) + B_o u(\cdot) + K_o y(\cdot) \end{cases} \tag{4.33}$$

One would like asymptotic convergence of $z(\cdot)$ to $Tx(\cdot)$ so that $\hat{x}(\cdot) \doteq T^{-1}z(\cdot)$ can be used as estimate of $x(\cdot)$ in some feedback design. The following theorem is due to Luenberger [80] and gives an algebraic solution to the problem.

Theorem 4.4. Let T satisfy the equation

$$TA - A_o T - K_o C = 0. \tag{4.34}$$

Then putting

$$B_o = TB - K_o D \tag{4.35}$$

we have

$$\lambda(z(\cdot) - Tx(\cdot)) = A_o(z(\cdot) - Tx(\cdot)). \tag{4.36}$$

If (A, C) is observable an invertible T always exists for any choice of spectrum for A_o and convergence is obtained when choosing A_o stable.

Proof. Multiply the top equation in (4.33) with T and subtract it from the bottom equation, then eliminate $y(\cdot)$ using the middle equation to get

$$\lambda[z(\cdot) - Tx(\cdot)] = A_o z(\cdot) - [TA - K_o C]x(\cdot) - [TB - K_o D - B_o]u(\cdot). \tag{4.37}$$

Now use (4.34), (4.35) to simplify this and obtain (4.36). If (A, C) is observable then one can always choose $T = I$ and solve

$$A_o = A - K_o C \tag{4.38}$$

via pole placement. Since this is the transpose of a standard feedback problem $A_o^T = A^T - C^T K_o^T$ we can always stabilize A_o via appropriate choice of K_o. □

The above proof already suggests how to solve these equations. Notice that (4.34) is a nonlinear equation in the unknowns T, A_o and K_o. Also there are less equations than unknowns so we can choose some variables first. Two standard approaches are

1. choose $T = I$ and solve $A_o = A - K_o C$ via pole placement
2. choose A_o stable and K_o arbitrary. Then solve for T from (4.34). Solvability of this equation is discussed in the next section.

Once (4.34) is solved we just assign B_o as in (4.35).

The observer problem depends on the exact solution of the algebraic equations (4.34)-(4.35). But since there are always residual errors (due to round off, or due to inaccuracies in the data A, B, C, D), the observer equation (4.36) will not be satisfied and $z(\cdot)$ does not converge asymptotically to $Tx(\cdot)$.

Instead, we have

$$\lambda[z(\cdot) - Tx(\cdot)] = A_o[z(\cdot) - Tx(\cdot)] + Ex(\cdot) + Fu(\cdot) \tag{4.39}$$

where $\|E\|$ and $\|F\|$ are small. This is essentially a stable difference/differential equation driven with a small input and its response will not go asymptotically to zero, but rather quickly converge to a residual signal of norm comparable to the residual input $Ex(\cdot) + Fu(\cdot)$.

In the discrete time case, one is tempted to choose A_o nilpotent, since then exact convergence is obtained after a finite number of steps. In analogy to the results for pole placement, we have the following result.

Corollary 4.2. Let $\{A_o, B_o, K_o\}$ be an observer of the observable system $\{A, B, C, D\}$. Then there exists a solution A_o with degree of nilpotency ℓ, where ℓ is the largest observability index of the pair (A, C). No observer can reconstruct x_k exactly after less time instants than ℓ. □

The following example illustrates this.

Example 4.5.

Let the system $\{A, B, C, D\}$ be given by:

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{array}\right].$$

The observer problem has a solution

$$T = I_3, A_o = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, K_o = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 2 \end{bmatrix}, B_o = B$$

A_o is nilpotent with degree of nilpotency 2, i.e. $A_o^2 = 0$. We should have convergence after 2 steps, independently of the starting vector z_0. Start the observer with zero initial state z_0, and let the plant be governed by:

$$x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}; u_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; u_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Then at step $k = 0$ we have:

$$x_1 = Ax_0 + Bu_0 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix},\ y_0 = Cx_0 + Du_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ z_1 = A_o z_0 + B_o u_0 + K_o y_0 = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$$

Clearly $x_1 - z_1 = A_o(x_0 - z_0) \neq 0$. But at time $k = 1$, we have

$$\begin{cases} x_2 &= Ax_1 + Bu_1 = \begin{bmatrix} 0 \\ 4 \\ 8 \end{bmatrix} \\ y_1 &= Cx_1 + Du_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \\ z_2 &= A_o z_1 + B_o u_1 + K_o y_1 = \begin{bmatrix} 0 \\ 4 \\ 8 \end{bmatrix} \end{cases}$$

and $z_2 = x_2$, which is maintained for all $k > 2$. □

The fact that the observer equation is not exact and that the reconstructed state does not converge asymptotically to the true state is not really harmful when the reconstructed state $T^{-1}z(\cdot)$ is used to perform feedback on the original system. To see this we embed both systems into one.

$$\begin{cases} \begin{bmatrix} \lambda x(\cdot) \\ \lambda z(\cdot) \end{bmatrix} &= \begin{bmatrix} A & 0 \\ K_o C & A_o \end{bmatrix} \begin{bmatrix} x(\cdot) \\ z(\cdot) \end{bmatrix} + \begin{bmatrix} B \\ B_o + K_o D \end{bmatrix} u(\cdot), \\ y(\cdot) &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(\cdot) \\ z(\cdot) \end{bmatrix} + Du(\cdot). \end{cases} \tag{4.40}$$

Apply $u(\cdot) = F\hat{x}(\cdot) + v(\cdot) = FT^{-1}z(\cdot) + v(\cdot)$:

$$\begin{aligned} \begin{bmatrix} \lambda x(\cdot) \\ \lambda z(\cdot) \end{bmatrix} &= \begin{bmatrix} A & BFT^{-1} \\ K_o C & A_o + (B_o + K_o D)FT^{-1} \end{bmatrix} \begin{bmatrix} x(\cdot) \\ z(\cdot) \end{bmatrix} + \begin{bmatrix} B \\ B_o + K_o D \end{bmatrix} v(\cdot) \\ y(\cdot) &= \begin{bmatrix} C & DFT^{-1} \end{bmatrix} \begin{bmatrix} x(\cdot) \\ z(\cdot) \end{bmatrix} + Dv(\cdot). \end{aligned} \tag{4.41}$$

Use now the observer identities to get an extended state transition matrix :

$$\hat{A} = \begin{bmatrix} A & BFT^{-1} \\ TA - A_o T & A_o + TBFT^{-1} \end{bmatrix}.$$

which, up to a similarity transformation, yields:

$$\begin{bmatrix} I & \\ -T & I \end{bmatrix} \hat{A} \begin{bmatrix} I & \\ T & I \end{bmatrix} = \begin{bmatrix} A + BF & BFT^{-1} \\ 0 & A_o \end{bmatrix}.$$

This equation is the extended state transition matrix of

$$\begin{array}{rcl} \begin{bmatrix} \lambda x(\cdot) \\ \lambda e(\cdot) \end{bmatrix} & = & \begin{bmatrix} A+BF & BFT^{-1} \\ 0 & A_o \end{bmatrix} \begin{bmatrix} x(\cdot) \\ e(\cdot) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} v(\cdot) \\ y(\cdot) & = & [C+DF \quad DFT^{-1}] \begin{bmatrix} x(\cdot) \\ e(\cdot) \end{bmatrix} + Dv(\cdot) \end{array} \tag{4.42}$$

where $e(\cdot) \doteq z(\cdot) - Tx(\cdot)$ is supposed to go to zero at $t \to \infty$. The left over system essentially undergoes a standard feedback $A + BF$. If now the observer equations (4.34)(4.35) have residuals E and F, then all matrices in (4.42) get perturbed by matrices of roughly the same norm. But since both $A + BF$ and A_o are stable, $\hat{A}$ is stable as well and the perturbed $\hat{A} + \Delta\hat{A}$ will remain stable provided $\|E\|$ and $\|F\|$ are not too large. This indicates the importance to use robust pole placement techniques here for the construction of both A_o and $A + BF$.

4.4 Lyapunov and Sylvester Equations

Certain matrix equations arise naturally in linear control and system theory. Among those frequently encountered in the analysis and design of continuous-time systems are the Lyapunov equation

$$AX + XA^T + Q = 0, \tag{4.43}$$

and the Sylvester equation

$$AX + XF + Q = 0. \tag{4.44}$$

The appropriate discrete-time analogs are

$$AXA^T - X + Q = 0 \tag{4.45}$$

and

$$AXF - X + Q = 0. \tag{4.46}$$

It is important to notice that all these equations are *linear* in the elements x_{ij} of the unknown matrix X. So, using the vector notation

$$\text{vec}(X) = [X(:,1); X(:,2); \ldots; X(:,m)] \tag{4.47}$$

for the $n \times n$ matrix X we should be able to write any of the above equation in the form

$$M\text{vec}(X) = \text{vec}(Q). \tag{4.48}$$

We show how to do this for the Sylvester equation (4.44) from which all others can be derived. For this we introduce the Kroneder product of two matrices X and Y, denoted as $X \bigotimes Y$.

In MATLAB notation Kron $(X, Y) = X \bigotimes Y$ represents the matrix

$$\begin{array}{llll} [x_{11} * Y, & x_{12} * Y, & \ldots, & x_{1m} * Y; \\ x_{21} * Y, & x_{22} * Y, & \ldots, & x_{2m} * Y; \\ \vdots & & & \\ x_{n1} * Y, & x_{n2} * Y, & \ldots, & x_{nm} * Y]. \end{array} \tag{4.49}$$

Using this notation we have an equivalent reformulation of the Sylvester equation (4.44) where we assume A and F to be $n \times n$ and $m \times m$, respectively.

Theorem 4.5. The Sylvester equation $AX + XF + Q = 0$ is equivalent to

$$[(I_m \otimes A) + (F^T \otimes I_n)]\text{vec}(X) + \text{vec}(Q) = 0. \tag{4.50}$$

The eigenvalues of $[(I_m \bigotimes A) + (F^T \bigotimes I_n)]$ are the differences

$$(\alpha_i - \gamma_i) \quad i = 1, \ldots, n; j = 1, \ldots, m. \tag{4.51}$$

Proof. We only give a sketch of a proof here and refer to [46] for details. Clearly we have

$$\text{vec}(AX) + \text{vec}(XF) + \text{vec}(Q) = 0.$$

Now

$$(I \otimes A)\text{vec}(X) = \begin{bmatrix} A & & & \\ & A & & \\ & & \ddots & \\ & & & A \end{bmatrix} \begin{bmatrix} X(:,1) \\ X(:,2) \\ \vdots \\ X(:,m) \end{bmatrix} = \begin{bmatrix} AX(:,1) \\ AX(:,2) \\ \vdots \\ AX(:,m) \end{bmatrix} = \text{vec}(AX)$$

and

$$(F^T \otimes I)\text{vec}(X) = \begin{bmatrix} f_{11}I & \ldots & f_{m1}I \\ \vdots & \ddots & \vdots \\ f_{1m}I & \ldots & f_{mm}I \end{bmatrix} \begin{bmatrix} X(:,1) \\ \vdots \\ X(:,m) \end{bmatrix} = \begin{bmatrix} XF(:,1) \\ \vdots \\ XF(:,m) \end{bmatrix} = \text{vec}(XF)$$

which yields the required result. For the eigenvalue identity, one proves essentially that in an appropriate coordinate system the matrix in (4.50) is upper triangular with the differences (4.51) on diagonal. □

It is important to notice that such a rewrite can be done for each of the equations, by just replacing F by A^T in (4.43) and by inverting A^T and F in (4.45) and (4.46). We thus have

Corollary 4.3. The equations (4.43)-(4.46) are all linear equations in the elements of X and they have a unique solution if and only if

$$\begin{aligned} \Lambda(A) \cap \Lambda(-A) &= \emptyset \quad \text{for} \quad (4.43) \\ \Lambda(A) \cap \Lambda(-F) &= \emptyset \quad \text{for} \quad (4.44) \\ \Lambda(A) \cap \Lambda(A^{-1}) &= \emptyset \quad \text{for} \quad (4.45) \\ \Lambda(A) \cap \Lambda(F^{-1}) &= \emptyset \quad \text{for} \quad (4.46). \end{aligned}$$

□

For the Lyapunov equation, when A is stable, the solutions of the above equations are also equal to the reachability and observability Grammians $G_r(T)$ and $G_o(T)$, respectively, for $T = +\infty$ for the system $\{A, B, C\}$:

$$G_r = \int_0^\infty e^{tA}BB^Te^{tA^T}dt; \qquad G_o = \int_0^\infty e^{tA^T}C^TCe^{tA}dt \tag{4.52}$$

$$G_r = \sum_{k=0}^{\infty} A^kBB^T(A^T)^k; \qquad G_o = \sum_{k=0}^{\infty}(A^T)^kC^TCA^k. \tag{4.53}$$

Since A is stable in these cases one clearly satisfies the conditions of Corollary 4.2 and the solutions

$$AG_r + G_r A^T + BB^T = 0; \qquad G_o A + A^T G_o + C^T C = 0 \tag{4.54}$$
$$AG_r A^T - G_r + BB^T = 0; \qquad A^T G - 0A - G_o + C^T C = 0. \tag{4.55}$$

Notice that for general A, one can determine the number of stable and unstable eigenvalues of A. We define first the inertia of a matrix as a triple of integers (s, m, u) indicating the numbers of stable, marginal and unstable eigenvalues of that matrix. Notice that for continuous time systems and discrete-time systems, this has a different meaning in terms of the eigenvalue locations.

Theorem 4.6. Let (A, B) be reachable and (A, C) be observable then

$$I_n(A) \quad = I_n(G_r) = \quad I_n(G_o). \tag{4.56}$$

Proof. See [58] or [38]. □

This shows the relevance of the Lyapunov equation in various types of stability tests. Another important application is that of balancing. Suppose we have a state space model $\{A, B, C\}$ with Grammians G_r and G_o, then a state space transformation $\{T^{-1}AT, T^{-1}B, CT\}$ affects the Grammians as $T^{-1}G_r T^{-T}$ and $T^T G_o T$, which is sometimes called a contradgradient transformation. It is well known that such a transformation T can be chosen to make both matrices diagonal and equal

$$T^{-1}G_r T^{-T} \quad = \Sigma = \quad T^T G_o T \tag{4.57}$$

which is exactly the balanced coordinate system. From (4.43) it follows that

$$T^{-1}G_r G_o T \quad = \quad \Sigma^2 \tag{4.58}$$

and hence the eigenvector matrix T that diagonalizes the product $G_r G_o$ is also the transformation that balances the system $\{A, B, C\}$. This in fact only holds up to a diagonal scaling (scaling of the eigenvectors) and is accurate only when there are no repeated eigenvalues (in this case T is defined up to a diagonal scaling). A better way to choose T is to start from the Cholesky factorization of G_r and G_o;

$$G_r = L_1 L_1^T; \qquad G_o = L_2^T L_2 \tag{4.59}$$

where L_1 and L_2 are both lower triangular. One then constructs the singular value decomposition of the upper triangular matrix $L_1^T L_2^T$:

$$L_1^T L_2^T \quad = \quad U\Sigma V^T. \tag{4.60}$$

Then, defining $T = L_1 U \Sigma^{-1/2}$; $T^{-1} = \Sigma^{-1/2} V^T L_2$ one checks that

$$TT^{-1} = L_1 U \Sigma^{-1} V^T L_2 = L_1 (L_1^T L_2^T)^{-T} L_2 = I$$

and

$$\begin{aligned} T^{-1}G_r T^{-T} \quad &= \Sigma^{1/2} U^T L_1^{-1} (L_1 L_1^T) L_1^{-T} U \Sigma^{1/2} = \quad \Sigma \\ T^{-1}G_o T^{-T} \quad &= \Sigma^{1/2} V^T L_2^{-T} (L_2^T L_2) L_2^{-1} V \Sigma^{1/2} = \quad \Sigma. \end{aligned}$$

An efficient algorithm to find the singular value decomposition of a product of two upper triangular matrices is described in [52].

For the Sylvester equations we have already seen an application in observer design, but there it was possible to circumvent the equation by solving a pole placement problem instead (this is in fact also a set of linear equations in an appropriate coordinate system). The following extension to reduced order observers does not allow to reduce the problem directly to a pole placement problem and here we have indeed to solve a Sylvester equation where T is nonsquare.

Theorem 4.7. Let

$$\begin{cases} \lambda x = Ax + Bu \\ y = Cx + Du \end{cases} \tag{4.61}$$

be a plant with unknown state. Then the reduced order observer of state dimension $n - m$

$$\lambda z = A_o z + B_o u + K_o y \tag{4.62}$$

satisfies the error equation

$$\lambda(z - Tx) = A_o(z - Tx) \tag{4.63}$$

provided

$$\begin{cases} TA - A_o T - K_o C = 0 \\ TB - B_o - K_o D = 0 \end{cases}. \tag{4.64}$$

If (A, C) is observable then (4.64) always has a solution where A_o is stable and $\left[\frac{T}{C}\right]$ invertible. □

In order to solve the above equations one actually does not use the system of linear equations because that involves $m \times n$ equations in $m \times n$ unknowns and would require typically $O(m^3 \cdot n^3)$ operations. Instead one performs unitary transformations U and V on the matrices A and F:

$$A_u = U^H A U; \qquad F_v = V^H F V \tag{4.65}$$

resulting in

$$A_u X_{uv} + X_{uv} F_v + Q_{uv} \;=\; 0 \tag{4.66}$$

where

$$X_{uv} = U^H X V; \qquad Q_{uv} = U^H Q V. \tag{4.67}$$

Notice that because unitary transformations were chosen, the sensitivity of the problem has not changed. We now choose U and V appropriately such that the equation (4.66) is easy to solve. In [7] it is recommended to take A_u in upper Schur form and F_v in lower Schur form. Once this is chosen one can partition the system as follows:

$$\left[\begin{array}{c|c} A_{11} & a_{12} \\ \hline 0 & \alpha_{22} \end{array}\right] \left[\begin{array}{c|c} X_{11} & x_{12} \\ \hline x_{21} & x_{22} \end{array}\right] + \left[\begin{array}{c|c} X_{11} & x_{12} \\ \hline x_{21} & x_{22} \end{array}\right] \left[\begin{array}{c|c} F_{11} & 0 \\ \hline f_{21} & \gamma_{22} \end{array}\right] + \left[\begin{array}{c|c} Q_{11} & q_{12} \\ \hline q_{21} & q_{22} \end{array}\right] \;=\; 0.$$

From this we find

$$(\alpha_{22} + \gamma_{22}) x_{22} + q_{22} \;=\; 0 \tag{4.68}$$

which has a solution if $a_{22} + \gamma_{22} \neq 0$ (notice that α_{22} and γ_{22} are eigenvalues of A and F, respectively). Then we find

$$x_{21}(\alpha_{22}I + F_{11}) = q_{21} - x_{22}f_{21} \tag{4.69}$$

$$(A_{11} + \gamma_{22}I)x_{12} = q_{12} - x_{22}a_{21} \tag{4.70}$$

which again is solvable if $-\alpha_{22}$ is not an eigenvalue of F and $-\gamma_{22}$ is not and eigenvalue of A. From here on we can then perform the same procedure recursively until all of X is determined.

The complexity of this algorithm is $O(n^3) + O(m^3)$ for the Schur decomposition and then of the same order of magnitude for the subsequent backsubstitution step. The algorithm is shown to be "weakly stable" in [44].

Remarks

1. In the case of real arithmetic, the Schur forms may have 2×2 diagonal blocks, in which case one may have to solve a 2×2 subsystem in (4.68) rather than a scalar one. Also the systems (4.69) and (4.70) will be more involved but may be solved using the Korenecker product approach. This will not increase the order of magnitude of the complexity but will increase its coefficient.

2. For the Lyapunov equation (4.69) and (4.70) are identical and the complexity is essentially halved.

3. In the case of a Sylvester equation it is more economical to choose the Hessenberg form for one of the two matrices (the largest one). The recurrence is then essentially the same for the rest except that one has to solve Hessenberg systems at each step rather than triangular ones. This method is explained in [44].

4. In the reduced observer problem one has in fact a quadratic equation in the unknown T, A_o and K_o. Using the Schur form for A_o and the staircase form for (A, C) it is shown in [135] that one of these equations is linear and can always be solved in least squares sense. Once this solution is filled in, there is another equation that becomes linear, and this process can be continued to solve the complete system of equations.

5. Algorithms and software for the more general Sylvester equation

   $$A_1 X F_1^T + A_2 X F_2^T + Q = 0$$

   and it symmetric Lyapunov counterparts

   $$AXF^T + FXA^T + Q = 0$$

   and

   $$AXA^T + FXF^T + Q = 0$$

   occur in implicit systems of difference and differential equations. Algorithms are given in [44] and [40].

6. Since the Sylvester equation can be viewed as a linear equation one can compute its sensitivity. But this implies writing a large structured matrix M and computing the condition number of M. Simplified bounds are obtained for various special cases in the literature. For the stable Lyapunov equation see e.g. [55]. Relatively cheap "black box" condition estimators can be obtained using a few multiplications with the structured matrix M.

4.5 Algebraic Riccati Equation

We consider the ARE as it occurs in the optimal control problem. The equation for the Kalman filter are *dual* and essentially the same techniques hold there as well. The equations considered now are thus in continuous time:

$$PA + A^T P - PBR^{-1}B^T P + Q = 0 \tag{4.71}$$

and in discrete time:

$$\begin{aligned} P &= A^T[P - PB(R + B^T PB)^{-1}B^T P]A + Q \\ &= A^T[P^{-1} + BR^{-1}B^T]^{-1}A + Q. \end{aligned} \tag{4.72}$$

We first consider the continuous time problem and leave the discrete time case for later since it merely consists of transforming all techniques for the continuous time case to the analogous discrete time problem.

The origin of the ARE for the continuous time problem is the solution of the optimal feedback $u = Fx$ minimizing the functional (where we assume $Q \geqslant 0$, $R > 0$):

$$\begin{aligned} J &= \int_0^\infty \{x^T(t)Qx(t) + u^T(t)Ru(t)\}dt \\ \text{subject to} \quad \dot{x}(t) &= Ax(t) + Bu(t); x(0) = x_0. \end{aligned}$$

Theorem 4.8. If A, B is reachable then the optimal cost will be bounded. Otherwise there always exist an initial state x_0 which can make the cost unbounded if $Q > 0$. □

Proof. If A, B is reachable then there exists a feedback F_0 such that

$$u(t) = F_0 x(t);\ \dot{x}_u = (A + BF_0)x_u;\ x_u(0) = x_0$$

and then

$$\begin{aligned} J_0 &= \int_0^\infty x_u^T(t)[Q + F_0^T RF_0]x_u(t)dt \\ &= \int_0^\infty x_0^T e^{(A+BF_0)^T t}(Q + F_0^T RF_0)e^{(A+BF_0)t}x_0 dt \\ &\leqslant \|x_0\|_2^2 \left\| \int_0^\infty e^{(A+BF_0)^T t}(Q + F_0^T RF_0)e^{(A+BF_0)t}dt \right\|_2 = c\|x_0\|_2^2 \end{aligned}$$

where c is bounded. The "optimal" solution J^* is thus bounded from above by $c\|x_0\|_2^2$ and since $R > 0$, $Q \geqslant 0$, J is always positive. So J^* is clearly bounded. If (A, B) is not reachable, there exist an initial state x_0 (with a nonzero component along unreachable eigenvalue/eigenvector pairs) which will make the cost factor $x^T(t)Qx(t)$ grow unbounded if $Q > 0$. □

It is thus natural to assume (A, B) reachable. In a dual manner (A, C) observable is typically assumed for the Kalman filter in order to ensure bounded variance for the difference $x(t) - \hat{x}(t)$. In the sequel we will typically make these assumptions. We will see that this also guarantees existence of particular solutions of the ARE.

Remark.

The conditions of reachability and observability can be weakened to those of stabilizability and detectability [150]. □

Most of the methods that are numerically reliable and of reasonable complexity are based on the Hamiltonian matrix

$$H = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \tag{4.73}$$

which shows up in the state/costate equations

$$\begin{bmatrix} \dot{x} \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = H \begin{bmatrix} x \\ \lambda \end{bmatrix}. \tag{4.74}$$

This equation is derived from variational principles and its solution with suitable boundary conditions also solves the optimal control problems (see [116] for details). The term Hamiltonian comes from the following property.

Lemma 4.1. Let $J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}$ then $JH = -H^T J$.

Proof. Premultiplying J with H we obtain $JH = \begin{bmatrix} -Q & -A^T \\ -A & BR^{-1}B^T \end{bmatrix}$ which is symmetric, hence $JH = (JH)^T = H^T J^T$. Since $J = -J^T$ we have the required result. □

The above property is often referred to as H being "*J symmetric*" or "*Hamiltonian*".

Lemma 4.2. If (λ, x) is an eigenvalue/vector pair of H then, $(-\lambda, Jx)$ is an eigenvalue/vector pair of H^T.

Proof. Let $Hx = \lambda x$ then apply x to $JH = -H^T J$. This yields $JHx = \lambda(Jx) = -H^T(Jx)$ which is the required result. □

Corollary 4.4. If λ is an eigenvalue of H, then so is $-\lambda$, since H and H^T have the same eigenvalues.

If follows from the assumption made earlier that H has no eigenvalues on the $j\omega$ axis.

Lemma 4.3. If (A, B) is reachable or (A, Q) is observable then H has no eigenvalues on the $j\omega$-axis.

Proof. See [77]. □

The ARE is then derived from imposing $\lambda = Px$ in this homogeneous differential equation. Inserting this in (4.74) yields

$$\begin{bmatrix} I \\ P \end{bmatrix} \dot{x} = H \begin{bmatrix} I \\ P \end{bmatrix} x. \tag{4.75}$$

Premultiplying this with $[-P, I]$ yields the condition

$$[\,-P\;\;I\,]\,H\left[\begin{array}{c} I \\ P \end{array}\right] x(t) \;\;=\;\; 0, \forall x(t)$$

from which we obtain

$$[\,-P\;\;I\,]\,H\left[\begin{array}{c} I \\ P \end{array}\right] \;\;=\;\; 0 \tag{4.76}$$

which is precisely the ARE (4.71). Writing this into the matrix product

$$\left[\begin{array}{cc} I & 0 \\ -P & I \end{array}\right] H \left[\begin{array}{cc} I & \\ P & I \end{array}\right] \;\;\doteq\;\; H_t$$

we find that H_t has the block form (if $P = P^T$)

$$T^{-1}HT \;\;=\;\; H_t = \left[\begin{array}{cc} A_F & -BR^{-1}B^T \\ 0 & -A_F^T \end{array}\right] \tag{4.77}$$

with $A_F = A - BR^{-1}B^TP = A + BF$ (when defining $F = -R^{-1}B^TP$). This last equation amounts to splitting the eigenvalues of H into two groups (those of A_F and those of $-A_F^T$) which appear to be symmetric with respect to the $j\omega$-axis.

In order to get into more details of the solution method we introduce the concept of invariant subspace at a matrix.

Definition 4.1. An invariant subspace $\mathcal{X}$ of a matrix A is any space $\mathcal{X}$ such that $Ax \in \mathcal{X}$ for any $x \in \mathcal{X}$. This is typically denoted by

$$A\mathcal{X} \subset \mathcal{X}. \tag{4.78}$$

Theorem 4.8. If $\mathcal{X}$ is a k-dimensional invariant subspace of the $n \times n$ matrix A, then for any basis X_{nk} of $\mathcal{X}$, there exists a $k \times k$ matrix $\hat{A}$ such that

$$AX \;\;=\;\; X\hat{A}. \tag{4.79}$$

Proof. Denote by x_i the i-th column of X. Since Ax_i lies in the space spanned by X there exist a column $\hat{a}_i$ such that $Ax_i = X\hat{a}_i$. Putting these columns together yields what we needed. □

This now leads to the following basic theorem.

Theorem 4.9. Let $\mathcal{X}$ be an invariant subspace of A and X_1 be a basis for it. Then there exists a "completion" of X_1 to an invertible matrix $X = [X_1 \mid X_2]$. For each such X we have

$$X^{-1}AX \;\;=\;\; \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline 0 & A_{22} \end{array}\right]. \tag{4.80}$$

If we choose X to be a unitary basis U_1 then the completion can always be chosen unitary: $U = [U_1 \mid U_2]$, $U^HU = I$.

Proof. Let X_1 be a basis for the invariant subspace then $AX_1 = X_1\hat{A}$ for some $\hat{A}$. Completing $X = [X_1 \mid X_2]$ then yields:

$$X^{-1}X \quad = \quad \left[\frac{Y_1}{Y_2}\right][X_1 \mid X_2] = \left[\begin{array}{c|c} I_{n_1} & 0 \\ \hline 0 & I_{n_2} \end{array}\right]$$

if we partition X^{-1} and X conformably. Apply now X^{-1}, X to A to get

$$X^{-1}AX = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \text{ where } A_{ij} \doteq Y_i A X_j.$$

But $AX_1 = X_1\hat{A}$ implies $A_{21} = Y_2AX_1 = Y_2X_1\hat{A} = 0$ which yields (4.80). The rest of the theorem is trivial. □

Remark

1. The concept of invariant subspace is a direct extension of that of eigenvector. An eigenvector x also spans a 1 dimensional space $\mathcal{X}$. It satisfies indeed

$$A\mathcal{X} \subset \mathcal{X} \tag{4.81}$$

and there exists a scalar λ for each basis vector x of $\mathcal{X}$ such that

$$Ax \quad = \quad x\lambda. \tag{4.82}$$

These are the 1 dimensional equivalents of (4.78)(4.79).

2. One shows that any space spanned by eigenvectors is an invariant subspace. Invariant subspaces can be slightly more general than just that when A has Jordan blocks.

Theorem 4.10. To every solution P of the ARE there corresponds a n-dimensional invariant subspace $Im\begin{bmatrix} I \\ P \end{bmatrix}$ of H. To every n-dimensional invariant subspace $Im\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ of H, there corresponds a solution $P = X_2X_1^{-1}$ of the ARE, provided X_1 is invertible.

Proof. The first part follows from (4.77) since it implies

$$H\begin{bmatrix} I \\ P \end{bmatrix} \quad = \quad \begin{bmatrix} I \\ P \end{bmatrix} A_F.$$

The second part follows from

$$H\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad = \quad \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \hat{H}.$$

If we assume X_1 invertible, then we have

$$H\begin{bmatrix} I \\ P \end{bmatrix} \quad = \quad \begin{bmatrix} I \\ P \end{bmatrix} X_1\hat{H}X_1^{-1} \doteq \begin{bmatrix} I \\ P \end{bmatrix} A_F$$

which is a solution to the ARE. □

One shows easily that the eigenvalues of H are symmetric with respect to the $j\omega$ axis and that every decomposition of the type (4.77) corresponds to a real symmetric solution P, while Theorem 4.5 in fact holds for general P.

Finding the real symmetric solution to the ARE amounts to finding groupings of eigenvalues in two sets that are symmetric to each other (with respect to the $j\omega$ axis). The solution P_0 of interest in optimal control is that which selects all the stable eigenvalues.

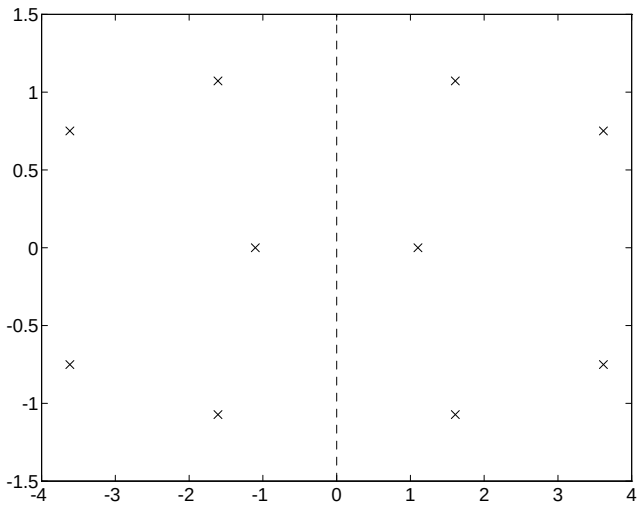


Figure 4.2: Eigenvalue pattern of a Hamiltonian matrix

A simple explanation for this is that A_F in

$$H \begin{bmatrix} I \\ P \end{bmatrix} = \begin{bmatrix} I \\ P \end{bmatrix} A_F$$

is then stable and this is also the closed loop matrix in the optimal regulator. All this leads to the following construction of the stablizing solution of the ARE.

Theorem 4.11. Let

$$U^H H U = \begin{bmatrix} h_{11} & \times & \dots & \times \\ & \ddots & \ddots & \vdots \\ & & \ddots & \times \\ & & & h_{2n,2n} \end{bmatrix} = \left[\begin{array}{c|c} H_{11} & H_{12} \\ \hline 0 & H_{22} \end{array}\right] \tag{4.83}$$

be the Schur decomposition of the Hamiltonian matrix H, where we ordered the eigenvalues h_{ii} of H such that $h_{11}, \dots, h_{n,n}$ are stable and the others unstable. Then partitioning U as

$$U = \left[\begin{array}{c|c} U_{11} & U_{12} \\ \hline U_{21} & U_{22} \end{array}\right] \tag{4.84}$$

we have that

$$P_0 = U_{21}U_{11}^{-1}$$

is the stablizing solution of the ARE. □

Remarks

1. We have assumed that H has no eigenvalues on the $j\omega$-axis. One shows that they can in fact only occur when the conditions of reachability and observability are violated. Under the same conditions one shows that P_0 is the only positive definite solution of the ARE.

2. The QR algorithm for finding the Schur decomposition with reordered eigenvalues is backward stable. Its complexity is $O(n^3)$ even with reordering (which implies updating U as well). [46] [124].

3. The inversion of U_{11} is $O(n^3)$ and can be done in a stable manner as well but the concatenation of both steps has not been proved to be stable, although experience shows this is a very reliable method.

4. The sensitivity of the decomposition (4.83) is well known. It is shown in [122] that the invariant subspace (and any orthogonal basis for it) has sensitivity

$$\kappa\left(\begin{bmatrix} U_{11} \\ U_{21} \end{bmatrix}\right) = 1/\text{ sep }(H_{11}, H_{22})$$

   where sep (H_{11}, H_{22}) is "essentially" the minimum distance between the eigenvalues of H_{11} and H_{22}. In other words, when eigenvalues occur very close to the $j\omega$ axis, the solution to the Riccati equation will be very sensitive.

5. The inversion of U_{11} depends more on the controllability of the system. When the system is poorly controllable, U_{11} is badly conditioned and P will have a large norm. This is to be expected since in order to stabilize $A + BF = A - BR^{-1}B^T$ one requires a large feedback gain. The bad conditioning of U_{11} will also affect the sensitivity of P.

6. When all matrices are real, P is also real and it should be computed using the real Schur form which is a variant of (4.83) with possibly 2×2 diagonal blocks in H_{11} and H_{22}.

We now give similar results for the discrete time Riccati equation, without giving new proofs since they are all similar. The problem is to minimize the functional

$$\begin{cases} J = \sum_0^\infty x_k^T Q x_k + u_k^T R u_k \\ \text{subject to } x_{k+1} = Ax_k + Bu_k, \quad x_0 \text{ given} \end{cases}$$

Stabilizability is again needed to have a bounded solution for all x. From variational calculus [116], one obtains the equations

$$\begin{bmatrix} x_{k+1} \\ \lambda_r \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ Q & A^T \end{bmatrix} \begin{bmatrix} x_k \\ \lambda_{k+1} \end{bmatrix}$$

or also

$$\begin{bmatrix} I & BR^{-1}B^T \\ 0 & A^T \end{bmatrix} \begin{bmatrix} x_{r+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} A & 0 \\ -Q & I \end{bmatrix} \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} \tag{4.85}$$

with boundary conditions specified for x_0 and λ_∞. This can be rewritten as

$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} I & BR^{-1}B^T \\ 0 & A^T \end{bmatrix} \begin{bmatrix} A & 0 \\ -Q & I \end{bmatrix}}_{S} \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} = S \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} \tag{4.86}$$

Theorem 4.12. The matrix S in (4.86) satisfies $SJS^T = J$ and is called sympletic.

Proof. One needs to prove $\begin{bmatrix} A & 0 \\ -Q & I \end{bmatrix} J \begin{bmatrix} A^T & -Q \\ 0 & I \end{bmatrix} = \begin{bmatrix} I & BR^{-1}B^T \\ 0 & A^T \end{bmatrix} J \begin{bmatrix} I & 0 \\ BR^{-1}B^T & A \end{bmatrix}$ but both sides equal $\begin{bmatrix} 0 & A \\ -A^T & 0 \end{bmatrix}$ which proves the result. □

Notice that we then also have $S^T JS = J$ since

$$(S^T J)SJS^T(JS^T)^{-1} = (S^T J)J(JS^T)^{-1} = J.$$

We then have

Lemma 4.4. If λ is an eigenvalue of S then λ^{-1} is an eigenvalue of S^T.

Proof. Similar to Hamiltonian. □So the eigenvalues lie in a symmetric fashion with respect to the unit circle. The figure 4.3 was in fact generated via random matrices as follows :

```
a=randn(5);q=randn(5);q=q*q';z=zeros(5,5);e=eye(5);b=rand(5,2);
l=eig([a z ;-q e],[e b*b';z a'])
plot(real(l),imag(l),'x',cos([0:30]*pi/15),sin([0:30]*pi/15),'--')
```

As in the continuous time case we have a lemma ruling out eigenvalues of S on the unit circle.

Lemma 4.5. If (A, B) is reachable or (A, Q) is observable, then S has no eigenvalues on the unit circle.

Proof. See [105]. □

One way to see the relation of this eigenvalue problem with the original ARE (4.72) is to rewrite it as a quadratic equation in P by getting rid of the inversion in (4.72). Multiply the bottom equation by $[P^{-1} + BR^{-1}B^T]A^{-T}$ on the left to obtain

$$(P^{-1} + BR^{-1}B^T)A^{-T}(P - Q) - A = 0$$

and after another multiplication with P and some regrouping:

$$-A^{-T}Q - P(BR^{-1}B^T A^{-T}Q + A) + A^{-T}P + PBR^{-1}B^T A^{-T}P = 0$$

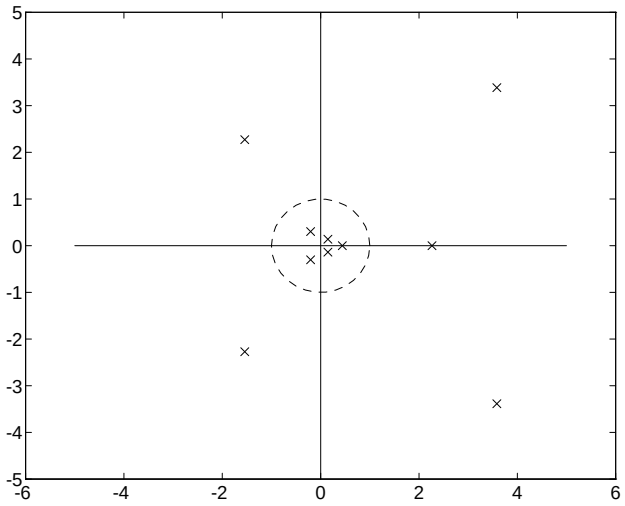


Figure 4.3: Eigenvalue pattern of a symplectic matrix

which is

$$[\,-P\;\;I\,]\begin{bmatrix} A+BR^{-1}B^TA^{-T}Q & -BR^{-1}B^TA^{-T} \\ -A^TQ & A^{-T} \end{bmatrix}\begin{bmatrix} I \\ P \end{bmatrix} = 0. \tag{4.87}$$

Since the matrix in the middle is in fact S:

$$S \;=\; \begin{bmatrix} I & BR^{-1}B^T \\ 0 & A^T \end{bmatrix}^{-1}\begin{bmatrix} A & 0 \\ -Q & I \end{bmatrix} = \begin{bmatrix} A+BR^{-1}B^TA^{-T}Q & -BR^{-1}B^TA^{-T} \\ -A^{-T}Q & A^{-T} \end{bmatrix} \tag{4.88}$$

we see that the discrete time ARE is again equivalent to finding an invariant subspace of a matrix S, which this time is symplectic instead of Hamiltonian. We now show that underlying this eigenvalue problem, there is a generalized eigenvalue problem which ought to be solved instead. Notice that when A is singular, the matrix S does not exist, whereas the matrices

$$E = \begin{bmatrix} I & BR^{-1}B^T \\ 0 & A^T \end{bmatrix}; \qquad F = \begin{bmatrix} A & 0 \\ -Q & I \end{bmatrix} \tag{4.89}$$

do exist. So the question arises if we can formulate definitions of eigenvalues, eigenvectors and invariant subspaces of $S = E^{-1}F$ directly in terms of E and F, without having to invert E.

It is easy to see that the definition of eigenvalue and eigenvector of $E^{-1}F$:

$$\det(\lambda_i I - E^{-1}F) = 0; \qquad (\lambda_i I - E^{-1}F)x_i = 0 \tag{4.90}$$

is equivalent to

$$\det(\lambda_i E - F) = 0; \qquad (\lambda_i E - F)x_i = 0 \tag{4.91}$$

when E is invertible. The important point here is that definition (4.91) even exists when E is singular and is well defined when $\det(\lambda E - F) \neq 0$. When this last condition is met the pencil

$\lambda E - F$ is called regular. The concept of an invariant subspace $\mathcal{X}$ of $E^{-1}F$ now generalizes to that of a deflating subspace [122] of $\lambda E - F$ as follows.

Definition 4.2. A deflating subspace $\mathcal{X}$ of a regular matrix pencil $\lambda E - F$ is any subspace $\mathcal{X}$ such that

$$\dim(E\mathcal{X} + F\mathcal{X}) \quad = \quad \dim \mathcal{X}. \tag{4.92}$$

□

Notice that if E is invertible then $\dim(E\mathcal{X}+F\mathcal{X}) = \dim(\mathcal{X}+E^{-1}F\mathcal{X})$ and hence (4.92) implies then

$$E^{-1}F\mathcal{X} \subset \mathcal{X}.$$

The matrix decomposition of Theorem 4.5 also carries over to generalized eigenvalue problems.

Theorem 4.13. If $\mathcal{X}$ is a k-dimensional deflating subspace of the $n \times n$ regular pencil $\lambda E - F$, then for any basis X_{nk} of $\mathcal{X}$, there exist $k \times k$ matrices $\hat{E}$ and $\hat{F}$ and a matrix Y_{nk} such that

$$EX = Y\hat{E}; \qquad FX = Y\hat{F}. \tag{4.93}$$

Proof. Let Y be a basis for $\mathcal{Y} = E\mathcal{X} + F\mathcal{X}$ then the above equations just express that $E\mathcal{X} \subset \mathcal{Y}$ and $F\mathcal{X} \subset \mathcal{Y}$. But this is precisely expressed by the matrix equations (4.93), which proves the result. □

This now again leads to a block triangular decomposition.

Theorem 4.14. Let $\mathcal{X}$ be a deflating subspace of a regular pencil $\lambda E - F$ and let X_1 be a basis for $\mathcal{X}$ and Y_1 a basis for $\mathcal{Y} = E\mathcal{X} + F\mathcal{X}$. Then there exist completions of X_1 and Y_1 to invertible matrices $X = [X_1 \mid X_2]$ and $Y = [Y_1 \mid Y_2]$. For each such matrices X and Y we have

$$Y^{-1}EX = \left[\begin{array}{c|c} E_{11} & E_{12} \\ \hline 0 & E_{22} \end{array}\right]; \qquad Y^{-1}FX = \left[\begin{array}{cc} F_{11} & F_{12} \\ \hline 0 & F_{22} \end{array}\right]. \tag{4.94}$$

If the bases X_1 and Y_1 are chosen unitary then the completions can also be chosen unitary.

Proof. Define Z as the inverse of Y and partition it as follows

$$ZY = \left[\frac{Z_1}{Z_2}\right][Y_1 \mid Y_2] = \left[\begin{array}{c|c} I_{n_1} & \\ \hline & I_{n_2} \end{array}\right]. \tag{4.95}$$

What ought to be the E_{21} and F_{21} blocks in (4.94) equal then

$$E_{21} = Z_2EX_1, \qquad F_{21} = Z_2FX_1.$$

But Theorem 4.5 implies that $EX_1 = Y_1\hat{E}$, $FX_1 = Y_1\hat{F}$. Filling this in we then have

$$E_{21} = Z_2Y_1\hat{E} = 0; \qquad F_{21} = Z_2Y_1\hat{F} = 0$$

because of (4.95). The orthogonal completions are trivial. □

Because of the similarities with the continuous time case we state the following theorem without proof.

Theorem 4.15. To every solution P of the discrete time ARE there corresponds an n-dimensional deflating subspace of $\lambda E - F$. To every n-dimensional deflating subspace $Im \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ of $\lambda E - F$, there corresponds a solution $P = X_2 X_1^{-1}$ of the discrete time ARE, provided X_1 is invertible.

Proof. See [105]. □

Just as in the standard eigenvalue problem, one constructs deflating subspaces from the generalized Schur form of a regular pencil $\lambda E - F$.

Theorem 4.16. Let

$$Q^H E U = \begin{bmatrix} \ell_{11} & \times & \dots & \times \\ & \ddots & \ddots & \vdots \\ & & \ddots & \times \\ & & & e_{2n,2n} \end{bmatrix} = \left[\begin{array}{c|c} E_{11} & E_{12} \\ \hline 0 & E_{22} \end{array}\right],$$

$$Q^H F U = \begin{bmatrix} f_{11} & \times & \dots & \times \\ & \ddots & \ddots & \vdots \\ & & \ddots & \times \\ & & & f_{2n,2n} \end{bmatrix} = \left[\begin{array}{c|c} F_{11} & F_{12} \\ \hline 0 & F_{22} \end{array}\right], \tag{4.96}$$

be the generalized Schur decomposition of the simpletic pencil $\lambda E - F$, where we ordered the generalized eigenvalues f_{ii}/e_{ii} such that the first n ones are stable and the others unstable. Then partitioning U as

$$U = \left[\begin{array}{c|c} U_{11} & U_{12} \\ \hline U_{21} & U_{22} \end{array}\right]$$

we have that

$$P_0 = U_{21} U_{11}^{-1}$$

is the stabilizing solution of the discrete time ARE. □

Remarks

1. The QZ algorithm with reordering [96] [133] is backward stable and has complexity $O(n^3)$. For the inversion of U_{11} the same comments apply as for the ARE. The same holds also for the sensitivity issues of the discrete time ARE. For real pencils, there is again a real generalized Schur from which solves the ARE in real arithmetic.

2. Just as the inversion of A and E was avoided in the discrete-time ARE, we can avoid the inversion of R in both schemes. The generalized eigenvalue problems that occur in this context are

$$\lambda \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} A & 0 & B \\ -Q & -A^T & 0 \\ 0 & -B^T & -R \end{bmatrix}$$

and

$$\lambda \begin{bmatrix} I & 0 & 0 \\ 0 & A^T & 0 \\ 0 & B^T & 0 \end{bmatrix} - \begin{bmatrix} A & 0 & B \\ -Q & I & 0 \\ 0 & 0 & -R \end{bmatrix}$$

for the continuous-time and discrete-time case, respectively. In both cases one shows that these generalized eigenvalue problems can be used for the construction of the solution of the ARE. The major advantage of this approach is that these eigenvalue problems are formulated directly in terms of the data of the problem, namely A, B, Q and R. This approach e.g., allows to solve the problem when R is singular whereas the ARE do not exit in this case [133] [31]. Similar enlarged pencils for solving spectral factorization problems are also described in [133]. The spirit of this approach is to solve problems directly in terms of its data whenever possible, without reducing it to a derived problem.

3. The sensitivity of the algebraic Riccati equation has been studied in [13], [69], [75]. It is important to see that this is different from just the perturbation analysis of eigenspaces of the underlying Hamiltonian or symplectic pencils.

4. There is a lot of structure in the eigenvalue problems that one considers to solve the Riccati equations. New methods were developed that exploit this structure to yield faster algorithms. Remarkably, one can ensure at the same time that the backward error of these methods are structured as well [9]. Moreover these methods have forward errors that are then compatible with the sensitivity analysis of the previous point.

5. Several references address particular numerical issues of optimal control [119] and related problems such as the singular case [86].

Chapter 5

KALMAN FILTERING

5.1 Kalman filter implementations

Since the appearance of Kalman's 1960 paper [66], the so-called Kalman filter (KF) has been applied successfully to many practical problems, especially in aeronautical and aerospace applications. As applications became more numerous, some pitfalls of the KF were discovered such as the problem of *divergence* due to the *lack of reliability of the numerical algorithm* or to *inaccurate modeling* of the system under consideration [61].

In this section we reconsider the numerical robustness of existing KF's and derive some results giving new and/or better insights into their numerical performance. Here we investigate four "basic" KF implementations: the Conventional Kalman Filter (CKF), the Square Root Covariance Filter (SRCF), the Chandrasekhar Square Root Filter (CSRF) and the Square Root Information Filter (SRIF).

We first introduce some notation. We consider the discrete time-varying linear system,

$$x_{k+1} = A_k x_k + B_k w_k + D_k u_k \tag{5.1}$$

and the linear observation process,

$$y_k = C_k x_k + v_k \tag{5.2}$$

where x_k, u_k and y_k are, respectively, the state vector to be estimated ($\in R^n$), the deterministic input vector ($\in R^r$) and the the measurement vector ($\in R^p$), where w_k and v_k are the process noise ($\in R^m$) and the measurement noise ($\in R^p$) of the system, and, finally, where A_k, B_k, C_k and D_k are *known* matrices of appropriate dimensions. The process noise and measurement noise sequences are assumed zero mean and uncorrelated:

$$E\{w_k\} = 0\ ,\ E\{v_k\} = 0\ ,\ E\{w_k v_j^T\} = 0 \tag{5.3}$$

with *known* covariances:

$$E\{w_j w_k^T\} = Q_k \delta_{jk}\ ,\ E\{v_j v_k^T\} = R_k \delta_{jk} \tag{5.4}$$

where $E\{.\}$ denotes the mathematical expectation and Q_k and R_k are *positive definite* matrices.

The SRF algorithms use the Cholesky factors of the covariance matrices or their inverse in order to solve the optimal filtering problem. Since the process noise covariance matrix Q_k and the

measurement noise covariance matrix R_k are assumed to be positive definite, the following Cholesky factorizations exist :

$$Q_k = Q_k^{1/2}[Q_k^{1/2}]^T, R_k = R_k^{1/2}[R_k^{1/2}]^T \tag{5.5}$$

where the factors $Q_k^{1/2}$ and $R_k^{1/2}$ may be chosen *upper* or *lower* triangular. This freedom of choice is exploited in the development of the fast KF implementations presented later. The problem is now to compute the *minimum variance estimate* of the stochastic variable x_k, provided y_1 up to y_j have been measured:

$$\hat{x}_{k|j} = \hat{x}_{k|y_1,...,y_j}. \tag{5.6}$$

When $j = k$ this estimate is called the *filtered estimate* and for $j = k - 1$ it is referred to as the one-step predicted or, shortly, the *predicted estimate*. The above problem is restricted here to these two types of estimates except for a few comments in the concluding remarks. Kalman filtering is a recursive method to solve this problem. This is done by computing the variances $P_{k|k}$ and/or $P_{k|k-1}$ and the estimates $\hat{x}_{k|k}$ and/or $\hat{x}_{k|k-1}$ from their previous values, this for $k = 1, 2, ...$ Thereby one assumes $P_{0|-1}$ (i.e. the covariance matrix of the initial state x_0) and $\hat{x}_{0|-1}$ (i.e. the mean of the initial state x_0) to be *given*.

The Conventional Kalman Filter (CKF)

The above recursive solution can be computed by the CKF equations, summarized in the following "covariance form" [1]:

$$R_k^e = R_k + C_k P_{k|k-1} C_k^T \tag{5.7}$$

$$K_k = A_k P_{k|k-1} C_k^T [R_k^e]^{-1} \tag{5.8}$$

$$P_{k+1|k} = A_k[I - P_{k|k-1}C_k^T[R_k^e]^{-1}C_k]P_{k|k-1}A_k^T + B_k Q_k B_k^T \tag{5.9}$$

$$\hat{x}_{k+1|k} = A_k\hat{x}_{k|k-1} - K_k[C_k\hat{x}_{k|k-1} - y_k] + D_k u_k \tag{5.10}$$

This set of equations has been implemented in various forms, see [1]. An efficient implementation that exploits the symmetry of the different matrices in (7-10) requires per step $3n^3/2 + n^2(3p + m/2) + n(3p^2/2 + m^2) + p^3/6$ "flops" (where 1 flop = 1 multiplication + 1 addition). By not exploiting the symmetry of the matrices in equations (7-10) one requires $(n^3/2 + n^2m/2 + np^2/2)$ more flops. In the error analysis, it is this "costly" implementation that is initially denoted as the CKF for reasons that are explained there. We also give some other variants that lead to further improvements in the number of operations.

The Square Root Covariance Filter (SRCF)

Square root covariance filters propagate the Cholesky factors of the error covariance matrix $P_{k|k-1}$:

$$P_{k|k-1} = S_k.S_k^T \tag{5.11}$$

where S_k is chosen to be lower triangular. The computational method is summarized by the following scheme [1]:

$$\underbrace{\begin{bmatrix} R_k^{1/2} & C_kS_k & 0 \\ 0 & A_kS_k & B_kQ_k^{1/2} \end{bmatrix}}_{\text{(prearray)}} .U_1 = \underbrace{\begin{bmatrix} R_k^{e1/2} & 0 & 0 \\ G_k & S_{k+1} & 0 \end{bmatrix}}_{\text{(postarray)}}, \tag{5.12}$$

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k-1} - G_k R_{e,k}^{-1/2}(C_k \hat{x}_{k|k-1} - y_k) + D_k u_k \tag{5.13}$$

where U_1 is an orthogonal transformation that triangularizes the prearray. Such a triangularization can e.g. be obtained using Householder transformations [46]. This recursion is now initiated with $\hat{x}_{0|-1}$ and the Cholesky factor S_0 of $P_{0|-1}$ as defined in (11). The number of flops needed for (12) and (13) is $7n^3/6+n^2(5p/2+m)+n(p^2+m^2/2)$. In order to reduce the amount of work, we only compute here the diagonal elements of the covariance matrix $P_{k+1|k}$, since usually $diag\{P_{k+1|k}\}$ carries enough information about the estimate $\hat{x}_{k+1|k}$ (namely the variance of the individual components). For this reason our operation counts differ e.g. from those of [67]. Below, we shortly discuss some other variants that lead to further improvements in the number of operations.

The Chandrasekhar Square Root Filter (CSRF)

If the system model (1-2) is *time-invariant*, the SRCF described above may be simplified to the Chandrasekhar square root filter, described in [98], [64]. Here one formulates recursions for the *increment* of the covariance matrix, defined as:

$$inc.P_k = P_{k+1|k} - P_{k|k-1} \tag{5.14}$$

In general this matrix can be factored as:

$$inc.P_k = L_k . \underbrace{\begin{bmatrix} I_{n_1} & 0 \\ 0 & -I_{n_2} \end{bmatrix}}_{\Sigma} . L_k^T \tag{5.15}$$

where the rank of $inc.P_k$ is $n_1 + n_2$ and Σ is called its signature matrix. The CSRF propagates recursions for L_k and $\hat{x}_{k+1|k}$ using [98]:

$$\underbrace{\begin{bmatrix} R_{k-1}^{e1/2} & CL_{k-1} \\ G_{k-1} & AL_{k-1} \end{bmatrix}}_{\text{(prearray)}} . U_2 = \underbrace{\begin{bmatrix} R_k^{e1/2} & 0 \\ G_k & L_k \end{bmatrix}}_{\text{(postarray)}}, \tag{5.16}$$

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} - G_k R_{e,k}^{-1/2}(C\hat{x}_{k|k-1} - y_k) + Du_k \tag{5.17}$$

with $L_0 \Sigma L_0^T = P_{1|0} - P_{0|-1}$. Here U_2 is a Σ_p - unitary transformation, i.e. $U_2 \Sigma_p U_2^T = \Sigma_p$ with

$$\Sigma_p = \begin{bmatrix} I_p & 0 \\ 0 & \Sigma \end{bmatrix} \tag{5.18}$$

Such transformations are easily constructed using "skew Householder" transformations (using an indefinite Σ_p-norm) and require as many operations as the classical Householder transformations [98]. For this implementation the operation count is $(n_1 + n_2)(n^2 + 3np + p^2)$ flops.

The Square Root Information Filter (SRIF)

The information filter accentuates the recursive least squares nature of filtering [10][1]. The SRIF propagates the Cholesky factor of $P_{k|k}^{-1}$ using the Cholesky factor of the inverses of the process- and measurement noise covariance matrices:

$$P_{k|k}^{-1} = T_k^T . T_k \tag{5.19}$$

$$Q_k^{-1} = [Q_k^{-1/2}]^T . Q_k^{-1/2} \tag{5.20}$$

$$R_k^{-1} = [R_k^{-1/2}]^T . R_k^{-1/2} \tag{5.21}$$

where the right factors are all chosen upper triangular. We now present the Dyer & McReynolds formulation of the SRIF (except for the fact that the time and measurement updates are combined here as in [1]) which differs from the one presented by Bierman (see [10] for details). One recursion of the SRIF algorithm is given by [1]:

$$U_3 . \underbrace{\begin{bmatrix} Q_k^{-1/2} & 0 & 0 \\ T_k A_k^{-1} B_k & T_k A_k^{-1} & T_k \hat{x}_{k|k} \\ 0 & R_{k+1}^{-1/2} C_{k+1} & R_{k+1}^{-1/2} y_{k+1} \end{bmatrix}}_{\text{(prearray)}} =$$

$$\underbrace{\begin{bmatrix} Q_{k+1}^{e-1/2} & \star & \star \\ 0 & T_{k+1} & \hat{\xi}_{k+1|k+1} \\ 0 & 0 & r_{k+1} \end{bmatrix}}_{\text{(postarray)}} \tag{5.22}$$

and the filtered state estimate is computed by,

$$\hat{x}_{k+1|k+1} = T_{k+1}^{-1} \hat{\xi}_{k+1|k+1} + D_k u_k \tag{5.23}$$

An operation count of this filter is $7n^3/6 + n^2(p + 7m/2) + n(p^2/2 + m^2)$ flops. Here we did not count the operations needed for the inversion and/or factorization of Q_k, R_k and A_k (for the time-invariant case e.g. these are computed only once) and again (as for the SRCF) only the diagonal elements of the information matrix $P_{k|k}^{-1}$ are computed at each step.

Variants of the above basic KF implementations have been developed which mainly exploit some particular structure of the given problem in order to reduce the amount of computations. E.g. when the measurement noise covariance matrix R_k is *diagonal*, it is possible to perform the *measurement update* in p scalar updates. This is the so-called *sequential processing* technique, a feature that is exploited by the UDU^T-algorithm to operate for the multivariable output case. A similar processing technique for the *time update* can be formulated when the process noise covariance matrix Q_k is *diagonal*, which is then exploited in the SRIF algorithm. Notice that no such technique can be used for the CSRF. The UDU^T-algorithm also saves operations by using unit triangular factors U and a diagonal matrix D in the updating formulas for which then special versions can be obtained [10]. By using modified Givens rotations [42] one could also obtain similar savings for the updating of the usual Cholesky factors, but these variants are not reported in the sequel.

For the *time-invariant* case, the matrix multiplications and transformations that characterize the described KF implementations can be made more efficient when the system matrices $\{A, B, C\}$ are first transformed by *unitary similarity transformations* to so-called *condensed form*, whereby these system matrices $\{A_t, B_t, C_t\}$ contain a lot of zeros. From the point of view of *reliability*, these forms are particularly interesting here, because no loss of accuracy is incurred by these unitary similarity transformation [140]. The following types of condensed forms can be used to obtain considerable savings in computation time in the subsequent filter recursions [140]: the *Schur form*, where A_t is in upper or lower Schur form, the *observer-Hessenberg form*, where the compound matrix $\left(A_t^T, C_t^T\right)$

is upper trapezoidal and the the *controller-Hessenberg form*, where the compound matrix (A_t, B_t) is upper trapezoidal. In [140], an application is considered were these efficient implementations are also valid for the *time varying* case. Note that the use of condensed forms and "sequential processing" could very well be combined to yield even faster implementations.

The operation counts for particular mechanizations of these variants are all given in table 5.1 and indicated by respectively the "seq.", "Schur", "o-Hess." and "c-Hess." abreviations, while "full" refers to the implementations described in previous sections where the full implementation of the CKF exploits symmetry.

filter	type	complexity
CKF	full	$(3/2)n^3 + n^2(3p + m/2) + n(3p^2/2 + m^2) + p^3/6$
	seq.	$(3/2)n^3 + n^2(3p + m/2) + n(p^2 + m^2)$
	Schur	$(3/4)n^3 + n^2(5p/2 + m/2) + n(3p^2/2 + m^2) + p^3/6$
	o-Hess.	$(3/4)n^3 + n^2(7p/2 + m/2) + n(2p^2 + m^2) + p^3/6$
SRCF	full	$(7/6)n^3 + n^2(5p/2 + m) + n(p^2 + m^2/2)$
	seq.	$(7/6)n^3 + n^2(5p/2 + m) + n(m^2/2)$
	Schur	$(1/6)n^3 + n^2(5p/2 + m) + n(2p^2)$
	o-Hess.	$(1/6)n^3 + n^2(3p/2 + m) + n(2p^2) + 2p^3/3$
SRIF	full	$(7/6)n^3 + n^2(p + 7m/2) + n(p^2/2 + m^2)$
	seq.	$(7/6)n^3 + n^2(p + 7m/2) + n(p^2/2)$
	Schur	$(1/6)n^3 + n^2(p + 5m/2) + n(2m^2)$
	c-Hess.	$(1/6)n^3 + n^2(3m/2 + p) + n(m^2 + p^2/2)$
CSRF	full	$(n_1 + n_2)(n^2 + 3np + p^2)$
	Schur	$(n_1 + n_2)(n^2/2 + 3np + p^2)$
	o-Hess.	$(n_1 + n_2)(n^2/2 + 3np + p^2)$

Table 5.1: Operation counts for the different KF's

5.2 Error analysis

In this section we analyze the effect of rounding errors on Kalman filtering in the four different implementations described above. The analysis is split in three parts: (1) what bounds can be obtained for the errors performed in step k, (2) how do errors performed in step k propagate in subsequent steps and (3) how do errors performed in different steps interact and accumulate. Although this appears to be the logical order in which one should treat the problem of error build-up in KF, we first look at the second aspect, which is also the only one that has been studied in the literature so far. Therefore, we first need the following lemma which is easily proved by inspection.

Lemma 1:

Let A be a square non-singular matrix with smallest singular value σ_{min} and let E be a perturbation of the order of $\delta = \|E\|_2 << \sigma_{min}(A)$ with $\|.\|_2$ denoting the 2-norm. Then

$$(A + E)^{-1} = A^{-1} + \Delta_1 = A^{-1} - A^{-1}EA^{-1} + \Delta_2 \tag{5.24}$$

where

$$\|\Delta_1\|_2 \leqslant \delta/\sigma_{min}(\sigma_{min} - \delta) = O(\delta) \tag{5.25}$$

$$\|\Delta_2\|_2 \leqslant \delta^2/\sigma_{min}^2(\sigma_{min} - \delta) = O(\delta^2) \tag{5.26}$$

Notice that when A and E are symmetric, these first and second order approximations (25) and (26) are also symmetric.

We now thus consider the propagation of errors from step k to step $k+1$ *when no additional errors are performed during that update.* We denote the quantities in computer with an upperbar, i.e. $\overline{P}_{k|k-1}$, $\overline{\hat{x}}_{k|k-1}$, $\overline{G}_k$, $\overline{S}_k$, $\overline{T}_k$, $\overline{R}_k^{e1/2}$, $\overline{F}_k$, $\overline{K}_k$, or $\overline{L}_k$, depending on the algorithm.

For the CKF, let $\delta P_{k|k-1}$ and $\delta x_{k|k-1}$ be the accumulated errors in step k, then:

$$\overline{P}_{k|k-1} = P_{k|k-1} + \delta P_{k|k-1} \ , \ \overline{\hat{x}}_{k|k-1} = \hat{x}_{k|k-1} + \delta \hat{x}_{k|k-1} \tag{5.27}$$

By using Lemma 1 for the inverse of $\overline{R}_k^e = R_k^e + C_k \delta P_{k|k-1} C_k^T$ we find

$$[\overline{R}_k^e]^{-1} = [R_k^e]^{-1} - [R_k^e]^{-1} C_k \delta P_{k|k-1} C_k^T [R_k^e]^{-1} + O(\delta^2) \tag{5.28}$$

From this one then derives:

$$\begin{array}{rcl} \overline{K}_k & = & A_k \overline{P}_{k|k-1} C_k^T \overline{R}_{e,k}^{-1} \\ \delta K_k & = & F_k \delta P_{k|k-1} C_k^T R_{e,k}^{-1} + O(\delta^2) \end{array} \tag{5.29}$$

where

$$F_k = A_k (I - P_{k|k-1} C_k^T R_{e,k}^{-1} C_k) = A_k - K_k C_k \tag{5.30}$$

and (assuming $\overline{P}_{k|k-1}$ is not necessarily symmetric, which would e.g. occur when applying (9) bluntly):

$$\begin{array}{l} \overline{P}_{k+1|k} = A_k (\overline{P}_{k|k-1} - \overline{P}_{k|k-1}^T C_k^T \overline{R}_{e,k}^{-1} C_k \overline{P}_{k|k-1}) A_k^T + B_k Q_k B_k^T \\ \delta P_{k+1|k} = F_k \delta P_{k|k-1} F_k^T + A_k (\delta P_{k|k-1} - \delta P_{k|k-1}^T) A_k^T - A_k (\delta P_{k|k-1} - \delta P_{k|k-1}^T) F_k^T + O(\delta^2) \end{array} \tag{5.31}$$

For the estimate $\hat{x}_{k+1|k}$ we have:

$$\begin{array}{l} \overline{\hat{x}}_{k+1|k} = \overline{F}_k \overline{\hat{x}}_{k|k-1} + \overline{K}_k y_k + D_k u_k \\ \delta \hat{x}_{k+1|k} = F_k [\delta \hat{x}_{k|k-1} + \delta P_{k|k-1} C_k^T R_{e,k}^{-1} (y_k - C_k \hat{x}_{k|k-1})] + O(\delta^2) \end{array} \tag{5.32}$$

When on the other hand $\delta P_{k|k}$ and $\hat{x}_{k|k}$ are given, one derives analogously,

$$\delta P_{k+1|k+1} = \tilde{F}_k \delta P_{k|k} \tilde{F}_k^T + A_k (\delta P_{k|k} - \delta P_{k|k}^T) A_k^T - A_k (\delta P_{k|k} - \delta P_{k|k}^T) \tilde{F}_k^T + O(\delta^2) \tag{5.33}$$

$$\delta \hat{x}_{k+1|k+1} = \tilde{F}_k [\delta \hat{x}_{k|k} + \delta P_{k|k} A_k^T C_{k+1}^T R_{k+1}^{e-1} (y_{k+1} - C_{k+1} A_k \hat{x}_{k|k})] + O(\delta^2) \tag{5.34}$$

where $\tilde{F}_k = (I - P_{k+1|k} C_{k+1}^T R_{k+1}^{e-1} C_{k+1}) A_k$ has the same spectrum as F_{k+1} in the time-invariant case, since $F_{k+1} A_k = A_{k+1} \tilde{F}_k$ [38].

We thus find that when $\delta P_{k|k-1}$ or $\delta P_{k|k}$ is symmetric, only the first term in (31) or (33) remains and the error propagation behaves roughly as:

$$||\delta P_{k+1|k}||_2 \approx ||F_k||_2^2 . ||\delta P_{k|k-1}||_2 = \gamma_k^2 . ||\delta P_{k|k-1}||_2 \tag{5.35}$$

$$||\delta P_{k+1|k+1}||_2 \approx ||\tilde{F}_k||_2^2 . ||\delta P_{k|k}||_2 = \tilde{\gamma}_k^2 . ||\delta P_{k|k}||_2 \tag{5.36}$$

which are decreasing in time when F_k and $\tilde{F}_k$ are contractions (i.e. when γ_k and $\tilde{\gamma}_k < 1$). The latter is usually the case when the matrices A_k, B_k, C_k, Q_k and R_k do not vary too wildly in time [1]. For the time-invariant case one can improve on this by saying that F_k and $\tilde{F}_k$ tend to the constant matrices F_∞ and $\tilde{F}_\infty$ respectively, with (equal) spectral radius $\rho_\infty < 1$ and one then has for some appropriate matrix norm [8]:

$$||\delta P_{k+1|k}|| \approx \rho_\infty^2 . ||\delta P_{k|k-1}|| \tag{5.37}$$

$$||\delta P_{k+1|k+1}|| \approx \rho_\infty^2 . ||\delta P_{k|k}|| \tag{5.38}$$

for sufficiently large k. Notice that ρ_∞ is smaller than γ_∞ or $\tilde{\gamma}_\infty$, whence (37-38) are better bounds than (35-36). Using this, it then also follows from (37-38) that all three errors $\delta P_{k|k-1}$, δK_k and $\delta \hat{x}_{k|k-1}$ are decreasing in time when no additional errors are performed. The fact that past errors are weighted in such a manner is the main reason why many Kalman filters do not diverge in presence of rounding errors.

The property (35-38) was already observed before [61], but for symmetric $\delta P_{k|k-1}$. However if symmetry is removed, divergence may occur when A_k (i.e. the original plant) is unstable. Indeed, from (31)(33) we see that when A_k is unstable the larger part of the error is skew symmetric:

$$\delta P_{k+1|k} \approx A_k . (\delta P_{k|k-1} - \delta P_{k|k-1}^T) . A_k^T \tag{5.39}$$

$$\delta P_{k+1|k+1} \approx A_k . (\delta P_{k|k} - \delta P_{k|k}^T) . A_k^T \tag{5.40}$$

and the lack of symmetry *diverges* as k increases. This phenomenon is well known in the extensive literature about Kalman filtering and experimental experience has lead to a number of different "remedies" to overcome it. The above first order perturbation analysis in fact explains why they work :

1. A first method to avoid divergence due to the loss of symmetry when A_k is unstable, is to *symmetrize* $\overline{P}_{k|k-1}$ or $\overline{P}_{k|k}$ at each recursion of the CKF by averaging it with its transpose. This makes the errors on P symmetric and hence the largest terms in (31)(33) disappear!

2. A second method to make the errors on P symmetric, simply computes only the *upper* (or lower) *triangular* part of these matrices, such as indicated by the implementation in table 5.1.

3. A third technique to avoid the loss of symmetry is the so-called (Joseph's) stabilized KF [11]. In this implementation, the set of equations for updating P are rearranged as follows:

   $$P_{k+1|k} = F_k P_{k|k-1} F_k^T + K_k R_k K_k^T + B_k Q_k B_k^T \tag{5.41}$$

   A similar first order perturbation study as for the CKF above, learns that *no symmetrization* is required in order to avoid divergence since here the error propagation model becomes :

   $$\delta P_{k+1|k} = F_k \delta P_{k|k-1} F_k^T + O(\delta^2) \tag{5.42}$$

   where there are no terms anymore related to the loss of symmetry.

Since for the moment we assume that *no additional errors* are performed in the recursions, one *inherently* computes the same equations for the SRCF as for the CKF. Therefore, starting with errors δS_k and $\delta \hat{x}_{k|k-1}$ equations (29),(31),(32),(35) and (37) still hold, whereby now

$$\delta P_{k|k-1} = S_k . \delta S_k^T + \delta S_k . S_k^T + \delta S_k . \delta S_k^T \tag{5.43}$$

is clearly symmetric by construction. According to (31) this now ensures the convergence to zero of $\delta P_{k|k-1}$ and hence of δS_k, δK_k and $\delta\hat{x}_{k|k-1}$ if γ_k is sufficiently bounded in the time-varying case.

For the SRIF we start with errors δT_k and $\delta\hat{x}_{k|k}$ and use the identity

$$\delta P_{k|k}^{-1} = T_k^T.\delta T_k + \delta T_k^T.T_k + \delta T_k^T.\delta T_k \tag{5.44}$$

$$\delta x_{k|k} = (T_k + \delta T_k)^{-1}\delta\hat{\xi}_{k|k} \tag{5.45}$$

to relate this problem to the CKF as well. Here one apparently does *not* compute $\hat{x}_{k+1|k+1}$ from $\hat{x}_{k|k}$ and therefore one would expect *no propagation* of errors between them. Yet, such a propagation is *present* via the relation (45) with the errors on $\delta\hat{\xi}_{k+1|k+1}$ and $\delta\hat{\xi}_{k|k}$, which *do* propagate from one step to another. This in fact is reflected in the recurrence (34) derived earlier. Since the SRIF update is *inherently equivalent* to an update of $P_{k|k}$ and $\hat{x}_{k|k}$ as in the CKF, the equations (33)(36) still hold where now the symmetry of $\delta P_{k|k}$ is ensured because of (44). From this it follows that $\delta P_{k|k}$ and $\delta\hat{x}_{k|k}$, and therefore also δT_k and $\delta\hat{\xi}_{k|k}$, converge to zero as k increases, provided $\tilde{\gamma}_k$ is sufficiently bounded in the time-varying case.

Finally, for the CSRF we start with errors δL_{k-1}, δG_{k-1}, $\delta R_{k-1}^{e1/2}$ and $\delta\hat{x}_{k|k-1}$. Because of these errors, (16) is perturbed *exactly* as follows:

$$\begin{bmatrix} R_{k-1}^{e1/2} + \delta R_{k-1}^{e1/2} & C(L_{k-1} + \delta L_{k-1}) \\ G_{k-1} + \delta G_{k-1} & A(L_{k-1} + \delta L_{k-1}) \end{bmatrix} .\overline{U}_2 = \begin{bmatrix} R_k^{e1/2} + \delta R_k^{e1/2} & 0 \\ G_k + \delta G_k & L_k + \delta L_k \end{bmatrix} \tag{5.46}$$

where $\overline{U}_2$ is also Σ_p-unitary. When $\lambda = \|C.L_{k-1}\| << \|R_{k-1}^{e1/2}\|$ (which is satisfied when k is sufficiently large), Lemma A.3 yields after some manipulations:

$$\begin{bmatrix} \delta R_{k-1}^{e1/2} & C\delta L_{k-1} \\ \delta G_{k-1} & A\delta L_{k-1} \end{bmatrix} .U_2 = \begin{bmatrix} \delta R_k^{e1/2} & 0 \\ \delta G_k & \delta L_k \end{bmatrix} + O(\delta.\lambda) \tag{5.47}$$

Now the $(1,1)$ and (1,2) blocks of U_2^T are given by $R_k^{e-1/2}.R_{k-1}^{e1/2}$ and $R_k^{e-1/2}.C.L_{k-1}.\Sigma$, respectively. From this, one then derives that for k sufficiently large

$$\begin{aligned} \delta R_k^{e1/2} &= \delta R_{k-1}^{e1/2}.[R_k^{e-1/2}.R_{k-1}^{e1/2}]^T + C.\delta L_{k-1}.[R_k^{e-1/2}.C.L_{k-1}.\Sigma]^T + O(\delta.\lambda) \\ &= \delta R_{k-1}^{e1/2}.[R_k^{e-1/2}.R_{k-1}^{e1/2}]^T + O(\delta.\lambda) \end{aligned} \tag{5.48}$$

$$\begin{aligned} \delta G_k &= \delta G_{k-1}.[R_k^{e-1/2}.R_{k-1}^{e1/2}]^T + A.\delta L_{k-1}.[R_k^{e-1/2}.C.L_{k-1}.\Sigma]^T + O(\delta.\lambda) \\ &= \delta G_{k-1}.[R_k^{e-1/2}.R_{k-1}^{e1/2}]^T + O(\delta.\lambda) \end{aligned} \tag{5.49}$$

Here again thus the errors $\delta R_{k-1}^{e1/2}$ and δG_{k-1} are multiplied by the matrix $[R_k^{e-1/2}.R_{k-1}^{e1/2}]^T$ at each step. When Σ is the identity matrix (i.e. when $inc.P_k$ is non-negative) this is a contraction since $R_k^e = R_{k-1}^e + C.L_{k-1}.L_{k-1}^T.C^T$. From this, we then derive similar formulas for the propagation of δK_k and $\delta\hat{x}_{k+1|k}$. Using Lemma 1 for the perturbation of the inverse in $K_k = G_k.R_k^{e-1/2}$, we find:

$$\begin{aligned} \delta K_k &= \delta G_k.R_k^{e-1/2} - G_k.R_k^{e-1/2}.\delta R_k^{e1/2}.R_k^{e-1/2} + O(\delta^2) \\ &= \delta G_k.R_k^{e-1/2} - K_k.\delta R_k^{e1/2}.R_k^{e-1/2} + O(\delta^2) \end{aligned} \tag{5.50}$$

Using (49)(50) and the fact that for large k, $K_k = K_{k-1} + O(\lambda)$, we then obtain

$$\begin{aligned} \delta K_k &= \delta G_{k-1}.[R_{e,k}^{-1/2}.R_{k-1}^{e1/2}]^T.R_{e,k}^{-1/2} \\ &\quad -K_{k-1}.\delta R_{k-1}^{e1/2}.[R_{e,k}^{-1/2}.R_{k-1}^{e1/2}]^T.R_{e,k}^{-1/2} + O(\delta.\lambda) \\ &= \delta G_{k-1}.R_{k-1}^{e-1/2}.[R_{k-1}^{e}.R_{e,k}^{-1}] \\ &\quad -K_{k-1}.\delta R_{k-1}^{e1/2}.R_{k-1}^{e-1/2}.[R_{k-1}^{e}.R_{e,k}^{-1}] + O(\delta.\lambda) \end{aligned} \tag{5.51}$$

which because of (50) decremented by 1 becomes:

$$\delta K_k = \delta K_{k-1}.[R_{k-1}^{e}.R_{e,k}^{-1}] + O(\delta.\lambda) \tag{5.52}$$

Using (17) we then also obtain from this:

$$\delta\hat{x}_{k+1|k} = F_k.\delta\hat{x}_{k|k-1} + \delta K_k.(y_k - C.\hat{x}_{k|k-1}) + O(\delta^2) \tag{5.53}$$

For the same reason as above, the matrix $[R_{k-1}^{e}.R_{e,k}^{-1}]$ is a contraction when $\Sigma = I$, which guarantees the convergence to zero of δK_k and $\delta\hat{x}_{k+1|k}$. Notice however, that here the contraction becomes closer to the identity matrix as k increases, which suggests that the inherent decaying of errors performed in previous steps will be less apparent for this filter. Besides that, nothing is claimed about δL_k or $\delta P_{k+1|k}$, but apparently these are less important for this implementation of the KF since they do not directly affect the precision of the estimate $\hat{x}_{k+1|k}$. Moreover, when Σ is not the identity matrix, the above matrix has norm larger than 1 and divergence may be expected. This has also been observed experimentally.

We now turn our attention to the numerical errors performed in one single step k. Bounds for these errors are derived in the following theorem.

Theorem 5.1.
Denoting the *norms* of the absolute errors due to round-off during the construction of $P_{k+1|k}$, K_k, $\hat{x}_{k+1|k}$, S_k, T_k, $P_{k+1|k+1}^{-1}$ and $\hat{x}_{k|k}$ by Δ_p, Δ_k, Δ_x, Δ_s, Δ_t, Δ_{pinv} and Δ_x, respectively, we obtain the following upper bounds (where all norms are 2-norms):

1. CKF

$$\begin{aligned} \Delta_p &\leqslant \epsilon_1.\sigma_1^2/\sigma_p^2.\|P_{k+1|k}\| \\ \Delta_k &\leqslant \epsilon_2.\sigma_1^2/\sigma_p^2.\|K_k\| \\ \Delta_x &\leqslant \epsilon_3.(\|F_k\|.\|\hat{x}_{k|k-1}\| + \|K_k\|.\|y_k\| + \|D_k\|.\|u_k\|) \\ &\quad +\Delta_k.(\|C_k\|.\|\hat{x}_{k|k-1}\| + \|y_k\|) \end{aligned}$$

2. SRCF

$$\begin{aligned} \Delta_s &\leqslant \epsilon_4.(1+\sigma_1/\sigma_p).\|S_{k+1}\|/cos\phi_1 \\ \Delta_p &\leqslant \epsilon_5.(1+\sigma_1/\sigma_p).\|P_{k+1|k}\|/cos\phi_1 \\ \Delta_k &\leqslant \epsilon_6/\sigma_p.(\sigma_1/\sigma_p.\|S_{k+1}\| + \sigma_1.\|G_k\| + \|S_{k+1}\|/cos\phi_1) \\ \Delta_x &\leqslant \epsilon_7.(\|F_k\|.\|\hat{x}_{k|k-1}\| + \|K_k\|.\|y_k\| + \|D_k\|.\|u_k\|) \\ &\quad +\Delta_k.(\|C_k\|.\|\hat{x}_{k|k-1}\| + \|y_k\|) \end{aligned}$$

3. CSRF

$$\begin{aligned} \Delta_k &\leqslant \epsilon_8.\kappa(U_2)/\sigma_p.(\sigma_1/\sigma_p.\|L_k\| + \sigma_1.\|G_k\| + \|L_k\|/cos\phi_2) \\ \Delta_x &\leqslant \epsilon_9.(\|F_k\|.\|\hat{x}_{k|k-1}\| + \|K_k\|.\|y_k\| + \|D\|.\|u_k\|) \\ &\quad +\Delta_k.(\|C\|.\|\hat{x}_{k|k-1}\| + \|y_k\|) \end{aligned}$$

4. SRIF

$$
\begin{aligned}
\Delta_t &\leqslant \epsilon_{10}.\{\kappa(A_k) + \kappa(R_k^{1/2}) + \tau_1/\tau_m.[\kappa(Q_k^{1/2}) + \kappa(A_k)]\}.\|T_{k+1}\|/cos\phi_3 \\
\Delta_{pinv} &\leqslant \epsilon_{11}.\{\kappa(A_k) + \kappa(R_k^{1/2}) + \tau_1/\tau_m.[\kappa(Q_k^{1/2}) + \kappa(A_k)]\}.\|P_{k+1|k+1}^{-1}\|/cos\phi_3 \\
\Delta_p &\leqslant \Delta_{pinv}.\|P_{k+1|k+1}\|^2 \\
\Delta_x &\leqslant \epsilon_{12}.\|D_k\|.\|u_k\| \\
&\quad +\Delta_t.[\kappa^2(T_{k+1}).\|r_{k+1}\| + \kappa(T_{k+1}).\|\hat{x}_{k+1|k+1}\| + \|r_{k+1}\|/cos\phi_4]
\end{aligned}
$$

where σ_i and τ_i are the i-th singular value of $R_{e,k}^{1/2}$ and $Q_{e,k+1}^{-1/2}$ respectively, ϵ_i are constants close to the machine precision ϵ and $cos\phi_i$ are defined as follows

$$
\begin{aligned}
cos\phi_1 &= \|S_{k+1}\|/\| [G_k|S_{k+1}] \| \\
cos\phi_2 &= \|L_k\|/\| [G_k|L_k] \| \\
cos\phi_3 &= \|T_{k+1}\|/\| \begin{bmatrix} T_k A_k^{-1} \\ R_{k+1}^{-1/2} C_{k+1} \end{bmatrix} \| \\
cos\phi_4 &= \|r_{k+1}\|/\| \begin{bmatrix} \hat{\xi}_{k+1|k+1} \\ r_{k+1} \end{bmatrix} \|
\end{aligned}
$$

and are usually close to 1.

Proof: see [144] □

These bounds are crude simplifications of the complicated process of rounding errors in linear algebra, but are often a good indication of what can go wrong in these problems (see e.g. [125] and [130]). This will be investigated more precisely in the experimental analysis of the next section. It is interesting to note here that the bounds derived in Theorem 5.1 disprove in a certain sense a result that was often used to claim the numerical supremacy of the SRF's, namely that the sensitivity of $P_{k+1|k}$, K_k and $\hat{x}_{k+1|k}$ (which according to Theorem 5.1 depends mainly on the singular values of R_k^e) as computed by the SRF's is the square root of that of the same quantities computed via the CKF. As far as the error analysis is concerned, this can only be claimed for $P_{k+1|k}$ and *not* for K_k or $\hat{x}_{k+1|k}$, as follows from a quick comparison of the CKF and the SRF's in Theorem 5.1. Therefore, we conclude that for situations that allow the application of the CKF, the SRF's *do not necessarily improve* the calculations of the Kalman gain or filtered estimates, although such a behavior is often observed.

Note also that when $\kappa(R_k^e) = 1$ all quantities are computed with roughly the same accuracy in the CKF and the SRCF. This particular situation arises e.g. when appropriately scaling the output measurements (this is also a known technique [11] to improve the performance of the CKF) or when using the "sequential processing" technique [89], described in the introduction.

Corollary 1: The above theorem also gives bounds on the errors due to model deviations δA_k, δB_k, δC_k, δD_k, δQ_k and δR_k, assuming that the latter are sufficiently small, as follows. Let η be the relative size of these errors, i.e. $\|\delta M\| \leqslant \eta\|M\|$ for M equal to each of the above model matrices, then the above bounds hold when replacing the ϵ_i by numbers η_i which are now all of the order of η.

Proof: The model errors can indeed be interpreted as backward errors on the matrices A_k, etc., but then on a machine of precision η. The same analysis then holds, but with ϵ replaced by η. □

Note that other modelling errors, such as bias errors on the input signals, discretization errors, etc. do not fall under this category and a separate analysis or treatment is required for each of them see e.g. [10].

The above theorem is now used together with the analysis of the *propagation* of errors through the recursion of the KF to yield bounds on the *total* error of the different filters at a given step k, which we denote by the prefix δ_{tot} instead of δ.

For this we first turn to the (symmetrized) CKF. For the total error $\delta_{tot}P_{k+1|k}$ we then have according to (29)(31)(33)(35) and Theorem 5.1 (for any *consistent* norm [123]):

$$\|\delta_{tot}P_{k+1|k}\| \leqslant \gamma_k^2.\|\delta_{tot}P_{k|k-1}\| + \overline{\Delta}_p \tag{5.54}$$

$$\|\delta_{tot}K_k\| \leqslant c_1.\gamma_k.\|\delta_{tot}P_{k|k-1}\| + \overline{\Delta}_k \tag{5.55}$$

$$\|\delta_{tot}\hat{x}_{k+1|k}\| \leqslant \gamma_k.\{\|\delta_{tot}\hat{x}_{k|k-1}\| + c_2.\|\delta_{tot}P_{k|k-1}\|\} + \overline{\Delta}_x \tag{5.56}$$

Here the upperbar on the Δ's indicate that these are not the exact bounds of Theorem 5.1 (which are derived under the assumption that the computations up to step k are exact), but analogous bounds derived for the perturbed results stored in computer at step k. Under the assumption that at step k the accumulated errors are still of the order of the local errors performed in one step (i.e. those estimated in Theorem 5.1), one easily finds that the Δ- and $\overline{\Delta}$-quantities are $O(\delta^2)$-close to each other. It is thus reasonable to assume that they are equal to each other. Denoting by $\Delta_{tot}.$ the norm of the corresponding matrix $\delta_{tot}.$, then finally yields:

$$\begin{bmatrix} \Delta_{tot}P_{k+1|k} \\ \Delta_{tot}K_k \\ \Delta_{tot}\hat{x}_{k+1|k} \end{bmatrix} \leqslant \gamma_k. \begin{bmatrix} \gamma_k & 0 & 0 \\ c_1 & 0 & 0 \\ c_2 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \Delta_{tot}P_{k|k-1} \\ \Delta_{tot}K_{k-1} \\ \Delta_{tot}\hat{x}_{k|k-1} \end{bmatrix} + \begin{bmatrix} \Delta_p \\ \Delta_k \\ \Delta_x \end{bmatrix} \tag{5.57}$$

where the inequality is meant elementwise. From this one then easily sees that the total errors will remain of the order of the local errors as long as the norms γ_k do not remain too large for a long period of time. This is also confirmed by the experimental results of the next section. For a time-invariant system, γ_k can be replaced by ρ_k — if the norm is chosen appropriately as discussed in (37) —, which then becomes eventually smaller than 1. Comparable results can also be stated about the γ_k if the time variations in the model are sufficiently smooth.

Using the above inequality recursively from 0 to ∞ one finally obtains

$$\begin{bmatrix} \Delta_{tot}P_{\infty} \\ \Delta_{tot}K_{\infty} \\ \Delta_{tot}\hat{x}_{\infty} \end{bmatrix} \leqslant \begin{bmatrix} 1/(1-\hat{\gamma}^2) & 0 & 0 \\ c_1\hat{\gamma}/(1-\hat{\gamma}^2) & 1 & 0 \\ c_2\hat{\gamma}/((1-\hat{\gamma}^2)(1-\hat{\gamma})) & 0 & 1/(1-\hat{\gamma}) \end{bmatrix} . \begin{bmatrix} \Delta_p \\ \Delta_k \\ \Delta_x \end{bmatrix} \tag{5.58}$$

if $\hat{\gamma} < 1$, where $\hat{\gamma}$ is the largest of the γ_k's. When γ_k tends to a fixed value γ_∞ it is easily shown that $\hat{\gamma}$ can be replaced by γ_∞ in (58), since the contributing terms to the summation are those with growing index k. For a *time-invariant* system, finally, this can then be replaced by ρ_∞ as was remarked earlier, and the condition $\hat{\gamma} = \rho_\infty < 1$ is then always satisfied.

For the SRCF, one uses the relation to the CKF (as far as the propagation of errors from one step to another is concerned) to derive (58) in an analogous fashion, but now with Δ_p, Δ_k and Δ_x appropriately adapted for the SRCF as in Theorem 5.1. For the SRIF one also obtains analogously the top and bottom inequalities of (57) for Δ_p and Δ_x adapted for the SRIF as in Theorem 5.1 and where now $\hat{\gamma}$ is the largest of the $\tilde{\gamma}_k$'s. Upon convergence, the same remarks hold as above for replacing $\hat{\gamma}$ by $\tilde{\gamma}_\infty$ and ρ_∞. Finally for the CSRF, we can only derive from (52)(53) a recursion of the type:

$$\begin{bmatrix} \Delta_{tot}K_k \\ \Delta_{tot}\hat{x}_{k+1|k} \end{bmatrix} \leqslant \begin{bmatrix} \beta_k & 0 \\ c_2 & \gamma_k \end{bmatrix} . \begin{bmatrix} \Delta_{tot}K_{k-1} \\ \Delta_{tot}\hat{x}_{k|k-1} \end{bmatrix} + \begin{bmatrix} \Delta_k \\ \Delta_x \end{bmatrix} \tag{5.59}$$

where $\beta_k = \|R^e_{k-1}.R^{-1}_{e,k}\|_2$. Recursive summation of these inequalities as was done to obtain (58), only converges here — for both $\Delta_{tot}K_\infty$ and $\Delta_{tot}\hat{x}_\infty$ — when the β_k increase sufficiently slow to 1 as k grows. We remark here that these are only upper bounds (just as the bounds for the other filters), but the fact that they may diverge does indeed indicate that for the CSRF numerical problems are more likely to occur.

Notice that the first order analysis of this section collapses when $O(\delta^2)$ and $O(\delta)$ errors become comparable. According to lemma 1, this happens when $\kappa(R^e_k) \approx 1/\delta$, but in such cases it is highly probable that divergence will occur for all filter implementations.

5.3 Experimental evaluation of the different KF's

We show a series of experiments reflecting the results of our error analysis. For these examples the upper bounds for numerical round-off developed in the previous section are reasonably close to the true error build up. The simulations are performed for a realistic flight-path reconstruction problem, described in [145]. This analysis indicated the key relevant parameters $\kappa(R^e_k)$, the spectral norm γ_k and spectral radius $\rho_k = \rho(F_k)$, which in turn can be affected by $\rho(A)$. For details on the experiments, we refer to [145]. Because of the inclusion of the CSRF, only the *time-invariant* case is considered here. The SRCF and the SRIF algorithms are closely related from numerical point of view. They are therefore first compared to the CKF and secondly to the CSRF.

Comparing the SRCF/SRIF with the CKF

Two test were performed to analyze the effect of $\kappa(R^e_k)$ and $\rho(F_k)$, which turn out to be very close to $\kappa(R)$ and $\rho(A)$.

Test 1 - Fig.5.1a: ($\rho(A) = 1.0$ and $\kappa(R) = 10^2$)

Since symmetry of the error state covariance matrix P is not preserved by the CKF, the round-off error propagation model for the local error $\delta P_{k|k-1}$ says that divergence will occur if the original system is unstable. This experiment confirms this also when $\rho(A) = 1.0$, as is the case for the considered flight-path reconstruction problem [145]. Furthermore, it is observed from Fig.5.1a that the error on P with the CKF is almost completely determined by the *loss of symmetry* $||\overline{P}_{k|k-1} - \overline{P}^T_{k|k-1}|| = \Delta_{sym}P_{k|k-1}$. Different methods have been proposed to solve this problem. One particular method consists in forcing a symmetric error by averaging the off-diagonal elements of P after each recursion. The behavior of $\Delta_{tot}P_{k|k-1}$ for this implementation, denoted by CKF(S) in Fig.5.1a, indicates that this implementation becomes again competitive, even when the original system is unstable. On the other hand, the behavior of $\Delta_{tot}P_{k|k-1}$ for Joseph's stabilized CKF, denoted by (J)CKF in Fig.5.1a, confirms that the round-off errors do not diverge even when the symmetry of P is not retained. We also observe from Fig.5.1a that the round-off error on P with these modified CKF remains 10 times higher than the SRCF/SRIF combination.

Test 2 - Fig.5.1b: ($\rho(A) = 0.9$ and $\kappa(R) = 10^2$)

If we make the original system stable, the CKF is numerically stable. Moreover, the accuracy with which the Kalman gain is computed is of the same order as that of the SRCF. This is in contrast with a general opinion that SRF's improve the calculations of the Kalman gain or filtered estimates. We can state that they don't make accuracy poorer. From Fig.5.1b it is observed that *only* the error covariance matrix P is computed more accurately, which confirms the upperbounds for the round-off errors obtained earlier.

A comparison of the (a) and (b) bounds indicates that when the accuracy of the Kalman gain is considered no preference should exist for the SRF's to the CKF when A_k is *stable and time-invariant*. However, the experimental results demonstrate that for the latter conditions the loss of accuracy with the CKF(S) is still higher than the SRF's. Here we only want to draw the attention to the clear difference to be expected (and also reflected by the experiments) between the accuracy of $P_{k|k-1}$ and K_k in the CKF(S) implementation with respect to those of SRF filters.

Fig. 5.1 (a and b) Comparison SRCF/SRIF and CKF

Comparison SRCF/SRIF with CSRF

The upperbounds for the local errors Δ_k and Δ_x given in Theorem 5.1 indicate that the error propagation is convergent when $\beta_k = ||R^e_{k-1}.(R^e_k)^{-1}|| < 1$, which is the case only if the signature matrix Σ is the identity matrix I. Note that the error variation $\Delta_{tot}K_k$ is now weighted by β_k (instead of γ_k for the other filters), which even for $\Sigma = I$ becomes very close to 1 for large k. This is also the main reason of the poor numerical behavior of this filter. When $\Sigma \neq I$ (which depends on the choice of $P_{0|-1} \neq 0$) β_k is larger than 1 and $\kappa(U_2)$ *may* also become large. The influence of the choice of $P_{0|-1}$ is analyzed by the following two tests.

Test 3 - Fig.5.2a: ($P_{0|-1} \neq 0$, $\rho(A) = 1.0$ and $\kappa(R) = 1.0$)

The choice of $P_{0|-1} \neq 0$ influences the CSRF implementation *negatively*. First, in this experiment the computational efficiency decreases in comparison to the case $P_{0|-1} = 0$, discussed in the following test. This is because $(n_1 + n_2)$ in table 5.1 becomes greater than p or m. This was the case for all the tests performed with $P_{0|-1} \neq 0$. Secondly, the transformations used in each recursion to triangularize the prearray become Σ-unitary, i.e. having a condition number > 1. This is due to the fact that $inc.P_0$ is not definite. From Fig.5.2a, this negative effect is clearly observed. Both the error levels on P and K are a factor 10^2 larger than for the SRCF or SRIF. For the covariance

type algorithms considered here, it is observed that the error on the Kalman gain is always higher than the error on the state error covariance matrix. This is partly due to the extra calculation $G_k(R_k^e)^{-1/2}$ needed for the Kalman gain, where the condition number of $(R_k^e)^{1/2}$ determines the loss of accuracy.

Test 4 - Fig.5.2b: ($P_{0|-1} = 0$, $\rho(A) = 1.0$ and $\kappa(R) = 1.0$)

For this case $inc.P_0 = B.Q.B^T$ is positive definite, causing the transformations used in each recursion to be *unitary*. From the experimental results in Fig.5.2b we observe that the error on P is very small, while the error on K is much higher than for the SRCF calculations. Furthermore, the errors on K with the CSRF *increase very slowly* because the coefficient β_k becomes very close to 1. Generally, the CSRF is *less reliable* than the SRCF/SRIF combination. For zero initial conditions of the state error covariance matrix maximal reliability can be achieved with the CSRF. Therefore, for situations where $n >> m$, the CSRF may be preferred because of its increased computational efficiency despite its loss of accuracy. This property is obviously only valid for the time-invariant case.

Fig. 5.2 (a and b) Comparison SRCF/SRIF and CSRF

Comparison of the SRCF and the SRIF

In the previous experiments the SRCF/SRIF combination performed equally well. In this section a further analysis is made to compare both implementations. Theorem 5.1 says that besides $\kappa(R_k)$ and $\rho(F_k)$, other system parameters influence the round-off error accumulation in the SRIF. The effect of these parameters is analyzed in the following tests.

Test 5 - Fig.5.3a:

In this test very large condition numbers for A, Q and R are considered. As expected this indeed

causes the error on P to be much higher (a factor 10^3) for the SRIF than for the SRCF. As in test 2, the large value of $\kappa(R)$ again causes a great loss in the accuracy of the Kalman gain calculation in the SRCF. In this test we analyzed the deterioration of the error covariance matrix by the SRIF implementation by (fairly unrealistic) large condition numbers. The effect of a high $\kappa(A_k)$ may influence the accuracy of the SRIF considerably.

Test 6 - Fig.5.3b:

For this test, the measurement error statistics were taken from real flight-test measurement calibrations resulting in $Q = \text{diag}\{8.10^{-6}, 5.10^{-5}, 5.10^{-8}\}$ and $R = \text{diag}\{5.10^{-2}, 2.10^{-1}\}$. In Fig.5.3b the simulated error $\Delta_{tot}x$ on the state calculations is plotted for both filter implementations. Here, the error level with the SRIF is significantly higher than that for the SRCF, while P is computed with roughly equal accuracy. This is due to the high condition number of T_k in the calculation of the filtered state with the SRIF.

Fig. 5.3 (a and b) Comparison SRCF and SRIF

5.4 Comparison of the different filters

In this section we compare the different filter implementations based on the error analysis and the simulation examples.

We first look at the time-varying case (whence excluding the CSRF). According to the error bounds of Theorem 5.1, it appears that the SRCF has the lowest estimate for the *local* errors generated in a single step k. The *accumulated* errors during subsequent steps is governed by the norms γ_k for all three filters in a similar fashion (at least for the error on the estimate) — this of course under the assumption that a "symmetrized" version of the CKF or the stabilized CKF is considered. From these modifications, the implementation computing only the upper (or lower)

triangular part of the state error covariance matrix is the most efficient. The experiments with the realistic flight path reconstruction problem indeed demonstrate that the CKF, the SRCF and the SRIF seem to yield a comparable accuracy for the estimates $\hat{x}_{k+1|k}$ or $\hat{x}_{k+1|k+1}$, unless some of the "influential" parameters in the error bounds of Theorem 5.1 become critical. This is e.g. true for the SRIF which is likely to give worse results when choosing matrices A_k, R_k or Q_k that are hard to invert. As far as R_k or Q_k are concerned, this is in a sense an artificial disadvantage since in some situations the inverses R_k^{-1} and Q_k^{-1} are the given data and the matrices R_k and Q_k have then to be computed. This then would of course disadvantage the SRCF. In [104] it is shown that the problems of inverting covariances can always be by-passed as well for the SRIF as for the SRCF. The problem of inverting A_k, on the other hand, is always present in the SRIF.

For the computational cost, the SRCF/SRIF have a marginal advantage over the CKF when n is significantly larger then m and p (which is a reasonable assumption in general), even when computing the upper (or lower) triangular part of P with the CKF. Moreover preference should go to the SRCF (resp. SRIF) when $p < m$ (resp. $p > m$), with slightly preference for the SRCF when $p = m$. As is shown in [98], condensed forms or even the CSRF can sometimes be used in the time-varying case as well, when e.g. only some of the matrices are time-varying or when the variations are structured. In that case the latter two may yield significant savings in computing time. Similarly, considerable savings can be obtained by using sequential processing [10] when diagonal covariances are being treated (which is often the case in practice).

For the time-invariant case, the same comments as above hold for the accuracy of the CKF, SRCF and SRIF. The fourth candidate, the CSRF, has in general a much poorer accuracy than the other three. This is now not due to pathologically chosen parameters, but to the simple fact that the accumulation of rounding errors from one step to another is usually much more significant than for the three other filters. This is particularly the case when the signature matrix Σ is not the identity matrix, which may then lead to divergence as shown experimentally.

As for the complexity, the Hessenberg forms of the SRCF and the SRIF seem to be the most appealing candidate, except when the coefficient $n_1 + n_2$ in table 5.1 for the CSRF is much smaller than n. This is e.g. the case when the initial covariance $P_{0|-1}$ is zero, in which case the CSRF becomes the fastest of all four filters. Although the Schur implementations of the SRCF and SRIF are almost as fast as the Hessenberg implementations, they also have the small additional disadvantage that the original state-space transformation U for condensing the model to Schur form is more expensive than that for the other condensed forms and that the (real) Schur form is not always exactly triangular but may contain some 2×2 "bumps" on the diagonal (corresponding to complex eigenvalues of the real matrix A). Finally, the c-Hess. form of the SRIF (given in table 5.1) requires a more complex initial transformation U, since it is constructed from the pair $(A^{-1}, A^{-1}B)$ which also may be numerically more delicate due to the inversion of A.

As a general conclusion, we recommend the SRCF, and its observer-Hessenberg implementation in the time-invariant case, as the optimal choice of KF implementation because of its good balance of reliability and efficiency. Other choices may of course be preferable in some specific cases because of special conditions that would then be satisfied.

Chapter 6

POLYNOMIAL VERSUS STATE-SPACE MODELS

We have seen that there are two major classes of models for which a number of numerical algorithms have been developed. These models are state-space models

$$\left\{ \begin{array}{rcl} \lambda E x & = & Ax + Bu \\ y & = & Cx + Du \end{array} \right. \tag{6.1}$$

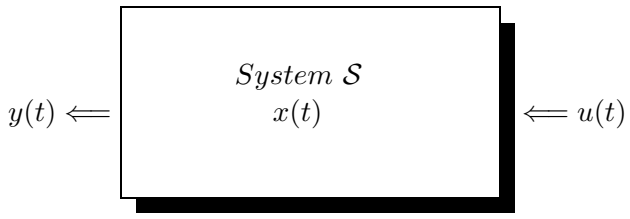
and polynomial models

$$D(\lambda) y \;=\; N(\lambda) u. \tag{6.2}$$

Several algorithms were already discussed in previous chapters and it appeared clearly that polynomial models have the advantage of speed over the state-space models. The main reason for this is the number of parameters in the model. For a same transfer function of a given degree n, a polynomial model typically has much less free parameters than a corresponding state-space model. E.g., for a SISO systems a polynomial model will have $2n$ parameters versus $(n+1)^2$ for the state-space model. This is a factor of roughly $\frac{n}{2}$ less parameter for the polynomial model. For a $m \times m$ transfer function of degree n the corresponding numbers are roughly $2mn$ parameters versus $(n+m)^2$, which already gives a smaller advantage for polynomial models. Also when using e.g., condensed forms these ratios have to be divided by 2. The same ratio was also observed in the complexity of SISO polynomial algorithms versus SISO state-space algorithms: the complexity is typically an order n smaller for polynomial algorithms. On the other hand, it was also observed that algorithms for polynomial models may suffer from numerical instabilities whereas there exist numerically stable algorithms for most problems formulated in state-space. This chapter analyzes these differences in more detail and tries to answer the question when to favor a particular type of model.

6.1 System Models And Transformation Groups

A system



with m inputs and p outputs can be represented in essentially four different ways:

1) Rational transfer function

$$y(t) \quad = \quad R(\lambda)u(t) \tag{6.3}$$

2) Polynomial fraction

$$D(\lambda)y(t) \quad = \quad N(\lambda)u(t) \tag{6.4}$$

3) Generalized state-space

$$\left\{ \begin{array}{rcl} \lambda E x(t) & = & Fx(t) + Gu(t) \\ y(t) & = & Hx(t) + Ju(t) \end{array} \right. \tag{6.5}$$

4) state-space

$$\left\{ \begin{array}{rcl} \lambda x(t) & = & Ax(t) + Bu(t) \\ y(t) & = & Cx(t) + Du(t). \end{array} \right. \tag{6.6}$$

The corresponds between these models are found from the transfer function of each model:

$$R(\lambda) = D^{-1}(\lambda)N(\lambda) = \quad H(\lambda E - F)^{-1}G + J \quad = C(\lambda I - A)^{-1}B + D. \tag{6.7}$$

All these models are mathematically equivalent in the sense that they describe the same object, namely the system $\mathcal{S}$. There also exist transformation methods to derive any of the above four models from the others ([65]). But from a numerical point of view the models are clearly not equivalent. This follows quite easily from the types of transformations typically used in algorithms dealing with these different models.

There are essentially two such sets of transformations

1) Unimodular transformation

$$P(\lambda) = \sum_{i=0}^{k} P_i \lambda^i; \qquad detP(\lambda) = c \neq 0 \tag{6.8}$$

2) Invertible transformations

$$T; \qquad detT \neq 0. \tag{6.9}$$

The first group of transformations is typically used for rational and polynomial models, the second group is used for state-space and generalized state-space models. Both these transformation sets, in fact, have a multiplicative group structure, which means that if M is a member of this group then:

$$\begin{array}{rcl} M \in \mathcal{G} & \Rightarrow & M^{-1} \in \mathcal{G} \\ M_1, M_2 \in \mathcal{G} & \Rightarrow & M_1 \cdot M_2 \in \mathcal{G}. \end{array} \tag{6.10}$$

When one has a transformation group then one can always define equivalence classes and canonical forms. These canonical forms play a fundamental role in many analysis and design problems and are therefore worth recalling. As related to the four different models we had, we briefly describe four canonical forms for rational matrices $R(\lambda)$, polynomial matrices $P(\lambda)$, pencils $\lambda E - F$ and monic pencils $\lambda I - A$. Under unimodular transformations $M(\lambda)$ and $N(\lambda)$, every rational matrix $R(\lambda)$ can be transformed to the quasidiagonal form (Smith McMillan form):

$$M(\lambda)R(\lambda)N(\lambda) \;=\; \left[\begin{array}{ccc|c} \frac{e_1(\lambda)}{f_1(\lambda)} & & & \\ & \ddots & & 0 \\ & & \frac{e_r(\lambda)}{f_r(\lambda)} & \\ \hline & 0 & & 0_{m-r,n-r} \end{array}\right]. \tag{6.11}$$

Similarly, every polynomial matrix $D(\lambda)$ can be transformed to a similar form (Smith form):

$$M(\lambda)D(\lambda)N(\lambda) \;=\; \left[\begin{array}{ccc|c} e_1(\lambda) & & & \\ & \ddots & & 0 \\ & & e_r(\lambda) & \\ \hline & 0 & & 0_{m-r,n-r} \end{array}\right]. \tag{6.12}$$

Constant transformations S, T can be used to reduce every pencil $\lambda E - F$ to a quasidiagonal form (Kronecker form):

$$S(\lambda E - F)T \;=\; \mathrm{diag}\{L_{\epsilon_1}, \ldots, L_{\epsilon_s}, L_{\eta_1}^T, \ldots, L_{\eta_t}, \lambda I - A, \lambda N - I\} \tag{6.13}$$

where (i) A is in Jordan normal form, (ii) N is nilpotent and in Jordan normal form, (iii) L_k is the $k \times (k+1)$ bidiagonal pencil

$$L_k = \underbrace{\begin{bmatrix} \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \lambda & 1 \end{bmatrix}}_{k+1}.$$

Similarly, constant transformations T^{-1}, T can be used to reduce the pencil $\lambda I - A$ to its quasidiagonal form :

$$T^{-1}(\lambda I - A)T = diag\{\lambda I_{k_i} - \begin{bmatrix} \alpha_i & -1 & & \\ & \ddots & \ddots & \\ & & \ddots & -1 \\ & & & \alpha_i \end{bmatrix}\}. \tag{6.14}$$

It is interesting to notice that with respect to the models (6.3)-(6.6) these canonical forms all reveal the polar structure of the transfer function, via the zeros of $f_i(\lambda)$ in (6.11), the zeros of $e_i(\lambda)$ in (6.12), the generalized eigenvalues of $\lambda E - F$ in (6.13), and the eigenvalues of A in (6.14). It is easily seen indeed that these are the values for which the transfer function (6.7) is unbounded.

6.2 Stabilized Transformations

The above canonical forms seem equivalent to each other in the sense that they define the same invariants of the system (the poles in this case). When including numerical considerations, the story is quite different. The group of constant invertible transformations has a subgroup of unitary transformations:

$$U, \qquad U^H U = I \Rightarrow \det U \neq 0. \tag{6.15}$$

This group is known to possess good properties in terms of error propagation. As a consequence of this, one can define modified canonical forms which essentially contain the same information as the Kronecker form and the Jordan form but which can be obtained using unitary transformations only. These forms are, of course, the generalized Schur form of a pencil $\lambda E - F$ and the Schur form of a pencil $\lambda I - A$, as presented in Chapter 5. Notice that the generalized Schur form in its most general form was, in fact, presented in Chapter 3 in connection to the computation of the zeros of a system.

In this section we show via a simple example that the class of unimodular transformations is unstable with respect to error propagation. Moreover, from the constraints of the algorithm considered, it appears to be impossible to stabilize these transformations. But the same problem can be solved using constant transformations and there of course we can restrict ourselves to unitary ones, which will stabilize the computations.

The problem considered is that of finding the zeros of a polynomial matrix of first order. Let

$$P(\lambda) = P_0 + \lambda P_1 = \begin{bmatrix} 0 & 0 & \delta \\ \delta & 1 & 0 \\ 0 & \delta & -1 \end{bmatrix} + \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.16}$$

where we assume δ to be small.

Notice that this is in fact a monic pencil, so that the solutions of

$$\det P(\lambda) \quad = \quad 0 \tag{6.17}$$

are just the eigenvalues of $-P_0$. We know there are stable algorithms for computing these eigenvalues, and moreover these eigenvalues are well conditioned when δ is small (since then P_0 is close to a diagonal matrix). We now show that the algorithm to construct the Smith form is unstable for this algorithm. The first step of the Smith decomposition algorithm permutes a nonzero element of minimum degree to the $(1,1)$ position (here we perform a permutation of rows 1 and 2)

$$M_1(P_0 + \lambda P_1) \quad = \quad \begin{bmatrix} \delta & \lambda + 1 & 0 \\ \lambda & 0 & \delta \\ 0 & \delta & \lambda - 1 \end{bmatrix}.$$

Then we perform column and row operations to reduce the degree of the other elements in row 1 and column 1:

$$M_2(\lambda)M_1(P_0+\lambda P_1)N_2(\lambda) \;=\; \begin{bmatrix} \delta & 0 & 0 \\ 0 & -\lambda(\lambda+1)/\delta & \delta \\ 0 & \delta & \lambda-1 \end{bmatrix}$$

where

$$M_2(\lambda) = \begin{bmatrix} 1 & 0 & 0 \\ -\lambda/\delta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, N_2(\lambda) = \begin{bmatrix} 1 & -(\lambda+1)/\delta & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

This is repeated until the element in $(1,1)$ divides all others and the rest of row 1 and column 1 is zero. Then the same procedure is repeated on the smaller dimensional matrix obtained after deleting row 1 and column 1:

$$M_3M_2(\lambda)M_1(P_0+\lambda P_1)N_2(\lambda) \;=\; \begin{bmatrix} \delta & 0 & 0 \\ 0 & \delta & \lambda-1 \\ 0 & -\lambda(\lambda+1)/\delta & \delta \end{bmatrix}$$

and

$$M_4(\lambda)M_3M_2(\lambda)M_1(P_0+\lambda P_1)N_2(\lambda)N_4(\lambda) \;=\; \begin{bmatrix} \delta & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & \frac{\lambda(\lambda+1)(\lambda-1)+\delta}{\delta^2} \end{bmatrix}$$

where

$$M_4(\lambda) = \begin{bmatrix} 1 & & \\ & 1 & \\ & \lambda(\lambda+1)/\delta^2 & 1 \end{bmatrix}, \qquad N_4(\lambda) = \begin{bmatrix} 1 & & \\ & 1 & -(\lambda-1)/\delta \\ & & 1 \end{bmatrix}.$$

From the diagonal form one finds

$$\det M(\lambda)P(\lambda)N(\lambda) = \delta^2\left[\frac{\lambda(\lambda+1)(\lambda-1)}{\delta^2}+\delta\right] = \lambda(\lambda+1)(\lambda-1)+\delta^3$$

which is indeed the characteristic polynomial of the original pencil. But the algorithm to obtain this form involved divisions by very small elements and this algorithm apparently does not show how this could be avoided. When δ goes to zero, it is easy to see that it becomes unstable for this reason.

6.3 State-Space Realizations Of Polynomials

We saw in Section 2.2 that to any SISO system $d^{-1}(\lambda)n(\lambda)$ in polynomial form there exists a realization that is easily derived from the coefficients of the polynomial $d(\lambda)$ and the first n samples

of the impulse response. A much simpler form is in fact given directly in terms of the coefficients of $n(\lambda)$ and $d(\lambda)$ in case degree $n(\lambda) <$ degree $d(\lambda)$:

$$\begin{array}{rl} A = \left[\begin{array}{cccc} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -d_0 & -d_1 & \dots & -d_{n-1} \end{array}\right] & b = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ 1 \end{array}\right] . \\ c = \left[\begin{array}{cccc} n_0 & n_1 & \dots & n_{n-1} \end{array}\right] & d = \left[\begin{array}{c} 0 \end{array}\right] \end{array} \tag{6.18}$$

A proof of this follows from the following identity:

$$\left[\begin{array}{cccc} \lambda & -1 & & \\ & \ddots & \ddots & \\ & & \lambda & -1 \\ d_0 & & d_{n-2} & \lambda + d_{n-1} \end{array}\right] \left[\begin{array}{c} 1 \\ \lambda \\ \vdots \\ \lambda^{n-1} \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ d(\lambda) \end{array}\right] . \tag{6.19}$$

Rewriting this we find

$$\left[\begin{array}{c} 1 \\ \lambda \\ \vdots \\ \lambda^{n-1} \end{array}\right] d^{-1}(\lambda) = (\lambda I - A)^{-1} \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ 1 \end{array}\right] = (\lambda I - A)^{-1} b \tag{6.20}$$

and hence, by just left multiplying this by the vector c:

$$c(\lambda I - A)^{-1} b = n(\lambda) d^{-1}(\lambda). \tag{6.21}$$

It is easy to see that the block version of this yields:

$$\left\{\begin{array}{rcl} N(\lambda) &=& N_0 + N_1\lambda + \cdots + N_{n-1}\lambda^{n-1} \\ D(\lambda) &=& D_0 + D_1\lambda + \cdots + D_{n-1}\lambda^{n-1} + I_m\lambda^n \end{array}\right. \tag{6.22}$$

and

$$\begin{array}{rcl} A_r &=& \left[\begin{array}{cccc} 0 & I_m & & \\ & \ddots & \ddots & \\ & & 0 & I_m \\ -D_0 & \dots & -D_{n-2} & -D_{n-1} \end{array}\right], \quad B_r = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ I_m \end{array}\right], \\ C_r &=& \left[\begin{array}{cccc} N_0 & \dots & N_{n-2} & N_{n-1} \end{array}\right], \qquad D_r = \left[\begin{array}{c} 0_{pm} \end{array}\right]. \end{array} \tag{6.23}$$

A quick look at the proof (6.19)-(6.21) shows that

$$C_r(\lambda I - A_r)^{-1} B_r + D_r = N(\lambda) D^{-1}(\lambda) \tag{6.24}$$

where the inverse $D^{-1}(\lambda)$ occurs at the right of $N(\lambda)$. By duality, one shows that for realizing the left product $D^{-1}(\lambda)N(\lambda)$ instead, one needs the dual formulas

$$\begin{array}{rl} A_\ell = \left[\begin{array}{cccc} 0 & & -D_0 & \\ I_p & \ddots & & \vdots \\ & \ddots & 0 & -D_{n-2} \\ & & I_p & -D_{n-1} \end{array}\right], & B_\ell = \left[\begin{array}{c} N_0 \\ \vdots \\ N_{n-2} \\ N_{n-1} \end{array}\right], \\ C_\ell = \left[\begin{array}{cccc} 0 & \dots & 0 & I_p \end{array}\right], & D_\ell = \left[\begin{array}{c} 0_{pm} \end{array}\right]. \end{array} \tag{6.25}$$

When $D(\lambda)$ is not a monic polynomial then one can not find simple state-space formulas for $D^{-1}(\lambda)N(\lambda)$ but instead it is easy to find a generalized state-space realization $C_g(\lambda E_g - A_g)^{-1}B_g + D_g = D^{-1}(\lambda)N(\lambda)$:

$$A_g - \lambda E_g = \begin{bmatrix} -\lambda I_p & & & -D_0 \\ I_p & \ddots & & \vdots \\ & \ddots & -\lambda I_p & -D_{n-1} \\ & & I_p & -D_{n-1} \end{bmatrix}, \quad B_g = \begin{bmatrix} N_0 \\ \vdots \\ N_{n-1} \\ N_n \end{bmatrix}, \tag{6.26}$$
$$C_g = [\, 0 \quad \dots \quad 0 \quad I_p \,], \quad D_g = [\, 0_{pm} \,].$$

where now $N(\lambda)$ and $D(\lambda)$ are arbitrary polynomial matrices.

Exercise 6.1

Prove that the above equations (6.26) satisfy $C_g(\lambda E_g - A_g)^{-1}B_g + D_g = D^{-1}(\lambda)N(\lambda)$ by using similar arguments as in (6.19)-(6.21). □

Notice that the realizations (6.18), (6.22), (6.24) and not necessarily minimal. Minimality of these realizations can be shown to be equivalent to *coprimeness* of the polynomials or polynomial matrices. In the case of matrices, one defines right and left coprimeness, which are not necessarily equivalent (see [114], [65]). Extraction of minimal realizations from these non-minimal ones can be done in a stable manner by using staircase forms both for state-space and generalized state-space realizations [141, 133].

6.4 Fast Versus Slow Algorithms

In this section we show a few fast algorithms using polynomial models and analyze their numerical reliability. In the case that they are unstable, alternative slower algorithms are discussed.

A first very classical algorithm for scalar polynomials is the Euclidean algorithm for finding the greatest common division $g(\lambda)$ of two polynomials $a(\lambda)$ and $b(\lambda)$:

$$\begin{cases} a(\lambda) &= a_0 + a_1\lambda + \cdots + a_k\lambda^k \\ b(\lambda) &= b_0 + b_1\lambda + \cdots + b_k\lambda^k. \end{cases} \tag{6.27}$$

The algorithm is based on the remainder theorem

$$\begin{cases} a(\lambda) &= b(\lambda)q(\lambda) + r(\lambda) \\ \deg r(\lambda) &< \deg b(\lambda). \end{cases} \tag{6.28}$$

Any polynomial dividing $a(\lambda)$ and $b(\lambda)$ clearly divides as well $r(\lambda)$ and conversely, any polynomial dividing $b(\lambda)$ and $r(\lambda)$ divides as well $a(\lambda)$. Hence

$$\gcd(a, b) = \gcd(b, r) \tag{6.29}$$

but meanwhile we are dealing with polynomials $b(\lambda)$ and $r(\lambda)$ of smaller degree. This is applied recursively as follows

$$a_1(\lambda) := a(\lambda); b_1(\lambda) := b(\lambda); i = 1$$

while $r_i(\lambda) \equiv 0$
$\quad r_i(\lambda) := r_i(\lambda) - b_i(\lambda)q_i(\lambda);$
$\quad a_{i+1}(\lambda) := b_i(\lambda); b_{i+1}(\lambda) := r_i(\lambda); i = i + 1;$
end
$g(\lambda) = a_i(\lambda)$

From (6.29) we have

$$\gcd(a_{i-1}, b_{i-1}) \quad = \quad \gcd(a_i, b_i). \tag{6.30}$$

When $r(\lambda) = 0$ we clearly have $\gcd(a, b) = b(\lambda)$, which yields the stopping criterion. This algorithm has complexity $O(n^2)$. In order to show this we look at the "generic" case, i.e., for "random" polynomials $a(\lambda)$ and $b(\lambda)$. For such polynomials the degree of $r_i(\lambda)$ decreases by 1 at each step, and each polynomial division involves a quotient $q_i(\lambda)$ of degree 1:

$$a_i(\lambda) = b_i(\lambda)(q_0^{(i)} + \lambda q_1^{(i)}) + r_i(\lambda), \tag{6.31}$$

i.e., $a_i(\lambda)$ has degree $(k - i + 1)$ and $b_i(\lambda)$ has degree $(k - i)$. The number of flops involved in computing $r_i(\lambda)$ in (6.31) is easily checked to be $2(k - i)$, and since this is performed at most k steps, we have a total complexity of

$$\sum_{i=1}^{k} 2(k-1) \quad = \quad k^2 + o(k) \tag{6.32}$$

flops. When the quotients happen to be of higher degree one shows that the complexity does in fact not exceed this simple bound.

Remark

Since $g(\lambda)$ is the gcd of $a(\lambda)$ and $b(\lambda)$ we have

$$a(\lambda) = \hat{a}(\lambda)g(\lambda); \qquad b(\lambda) = \hat{b}(\lambda)g(\lambda) \tag{6.33}$$

where $\gcd(\hat{a}, \hat{b}) = 1$. Since $\hat{a}(\lambda)$ and $\hat{b}(\lambda)$ are now coprime, it is well known [5] that there exist polynomials $\hat{c}(\lambda)$ and $\hat{d}(\lambda)$ such that

$$\hat{a}(\lambda)\hat{d}(\lambda) - \hat{b}(\lambda)\hat{c}(\lambda) \quad = \quad 1. \tag{6.34}$$

This identities imply

$$\begin{bmatrix} \hat{a}(\lambda) & \hat{c}(\lambda) \\ \hat{b}(\lambda) & \hat{d}(\lambda) \end{bmatrix} \begin{bmatrix} g(\lambda) \\ 0 \end{bmatrix} [1] \quad = \quad \begin{bmatrix} a(\lambda) \\ b(\lambda) \end{bmatrix} \tag{6.35}$$

where the left polynomial matrix has determinant 1. This is nothing but the Smith form of the 2×1 polynomial matrix $[a(\lambda), b(\lambda)]^T$. □

The above connection seems to indicate that, just as the Smith decomposition, this algorithm must be unstable in general. This is indeed the case and polynomials of relatively low order

(namely 6th order) were constructed where all 16 digits get corrupted during the calculations of the Euclidean algorithm [4].

The fact that the gcd algorithm is $O(n^2)$ can also nicely be interpreted in terms of the $2k \times 2k$ matrix

$$S_{2k} = \begin{bmatrix} a_0 & a_1 & \dots & a_k & 0 & \dots & 0 \\ b_0 & b_1 & \dots & b_k & 0 & \dots & 0 \\ 0 & a_0 & a_1 & \dots & a_k & \ddots & \vdots \\ 0 & b_0 & b_1 & \dots & b_k & & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a_0 & a_1 & \dots & a_k \\ 0 & \dots & 0 & b_0 & b_1 & \dots & b_k \end{bmatrix} \tag{6.36}$$

which is also known as the Sylvester matrix (up to a row permutation). Notice that this is in fact a block Toeplitz matrix. If the gcd $g(\lambda)$ has degree $\ell > 0$ then this matrix S_{2k} has the factorization:

$$S_{2k} = \begin{bmatrix} \hat{a}_0 & \hat{a}_1 & \dots & \hat{a}_k & 0 & \dots & 0 \\ \hat{b}_0 & \hat{b}_1 & \dots & \hat{b}_k & 0 & \dots & 0 \\ 0 & \hat{a}_0 & \hat{a}_1 & \dots & \hat{a}_k & \ddots & \vdots \\ 0 & \hat{b}_0 & \hat{b}_1 & \dots & \hat{b}_k & & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \hat{a}_0 & \hat{a}_1 & \dots & \hat{a}_k \\ 0 & \dots & 0 & \hat{b}_0 & \hat{b}_1 & \dots & \hat{b}_k \end{bmatrix} \begin{bmatrix} g_0 & \dots & g_\ell & 0 & \dots & 0 \\ 0 & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & \ddots & 0 \\ 0 & \dots & 0 & g_0 & \dots & g_\ell \end{bmatrix} \tag{6.37}$$

where the inner dimension of this factorization is $2k - \ell$. This identity is easily checked from the polynomial equations (6.33), and it implies that the nullity (or rank defect) of S_{2k} is at least ℓ. In [38], [5] it is shown that is actually equal to ℓ. This result again links the gcd problem to that of the rank of a matrix and therefore also to the singular values of a matrix. It then follows that the gcd of "random" polynomials is generically 1 (or its degree 0) since S_{2k} has generically full rank $2k$. The gcd problem should thus be rephrased as follows: given two polynomials $a(\lambda)$ and $b(\lambda)$, how close are they to having a nontrivial gcd $g(\lambda)$ of degree larger than 0? This is answered by looking at how close S_{2k} is to the matrices of lower rank and hence in values singular values of S_{2k}.

The Euclidean algorithm in fact amounts to a fast Gaussian elimination on S_{2k} yielding the decomposition (6.36). It is unstable because no pivoting is allowed in the algorithm and it does not compute singular values as in fact desired. The singular value decomposition on the other hand is slow, namely of the order of 80 k^3 on the $2k \times 2k$ matrix S_{2k}. Rank revealing factorizations of lower complexity than the SVD approach were considered in [4] but the problem of finding a reliable and fast algorithm for estimating how close $a(\lambda)$ and $b(\lambda)$ are to having a nontrivial gcd is still open. Let us point out here that computing the roots $a(\lambda)$ and $b(\lambda)$ and comparing those, does not solve this problem since roots may be badly conditioned.

Remark

The gcd problem is equivalent to that of finding the uncontrollable part of a corresponding realization

$$\left[\begin{array}{ccc} 1 & & -a_0 \\ & \ddots & \vdots \\ & & 1-a_{k-1} \end{array}\right], \quad B = \left[\begin{array}{c} b_0 \\ \vdots \\ b_{k-1} \end{array}\right] \tag{6.38}$$

where we assumed $\deg b(\lambda) < \deg a(\lambda)$ and $a(\lambda)$ monic. The corresponding closeness problem here is to find how close this (A, b) is to an uncontrollable one. □

A second problem involving polynomials for which now there exist a fast and reliable algorithm, is that of spectral factorization. Here one starts from a polynomial (which is positive real):

$$z(\lambda) \quad = \quad z_0 + z_1\lambda + \cdots + z_k\lambda^k \tag{6.39}$$

and one wants to find a polynomial

$$r(\lambda) \quad = \quad r_0 + r_1\lambda + \cdots + r_k\lambda^k \tag{6.40}$$

with roots inside the unit circle, and such that

$$z_*(\lambda) + z(\lambda) \quad = \quad r_*(\lambda) \cdot r(\lambda) \tag{6.41}$$

where the notation $z_*(\lambda)$ denotes the paraconjugate:

$$x_*(\lambda) \quad = \quad \overline{z}_0 + \overline{z}_1\lambda^{-1} + \cdots + \overline{z}_k\lambda^{-k} \tag{6.42}$$

and the positive real condition implies that

$$\Phi(\lambda) = z_*(\lambda) + z(\lambda) \geqslant 0 \quad \text{for} \quad \lambda = e^{j\omega}. \tag{6.43}$$

This problem is known to be equivalent to that of Riccati equations. The function $\Phi(\lambda)$ has zeros that are symmetric with respect to the unit circle just as the eigenvalue problem of the discrete time Riccati equation. The positive realness condition (6.43) guarantees that thee always exist a solution $r(\lambda)$ to (6.41). In fact $r(\lambda)$ is the polynomial that groups all the stable zeros of $\Phi(\lambda) = \Phi_*(\lambda)$. In [146] a fast algorithm was developed to construct $r(\lambda)$. It starts from an initial guess $r_0(\lambda)$ that is an arbitrary stable polynomial. Then it iterates as follows:

$$r_{i+1}(\lambda) \quad = \quad \frac{r_i(\lambda) + s_i(\lambda)}{2} \tag{6.44}$$

where

$$s_{i*}(\lambda)r_i(\lambda) + r_{i*}(\lambda)s_i(\lambda) = 2\Phi(\lambda). \tag{6.45}$$

This is in fact a Newton iteration for $r_0(\lambda)$. Assuming

$$r(\lambda) \quad = \quad r_i(\lambda) + \delta(\lambda) \tag{6.46}$$

and equating

$$\Phi(\lambda) \quad = \quad (r_{i*}(\lambda) + \delta_*(\lambda))(r_i(\lambda) + \delta(\lambda)) \tag{6.47}$$

we find

$$\Phi(\lambda) = r_{i*}(\lambda)r_i(\lambda) + \delta_*(\lambda)r_i(\lambda) + r_{i*}(\lambda)\delta(\lambda) + \delta_*(\lambda)\delta(\lambda). \tag{6.48}$$

Now this is a quadratic equation in $\delta(\lambda)$ but when we assume $\delta(\lambda)$ to be small (i.e., having small coefficients), then we can approximate this well by deleting the quadratic term and solving for a correction $\delta_i(\lambda)$ at step i:

$$\delta_{i*}(\lambda)r_i(\lambda) + r_{i*}(\lambda)\delta_i(\lambda) = \Phi(\lambda) - r_{i*}(\lambda)r_i(\lambda) \doteq \Delta\Phi(\lambda). \tag{6.49}$$

This is obviously a Newton step for solving (6.48), and it is the same as (6.44), (6.45) by substituting

$$s_i(\lambda) = r_i(\lambda) + 2\delta_i(\lambda). \tag{6.50}$$

The advantage of (6.45) over (6.49) is that the right hand side in (6.45) does not have to be computed at each step. What can we say about the complexity of this scheme? First of all the complexity of one iteration step is $O(k^2)$ where k is the order of the polynomials $z(\lambda)$, $r_i(\lambda)$ and $S_i(\lambda)$. This follows from the fact that the linear equation (6.45) has a structure related to that of Toeplitz matrices (see [146], [76]). The number of steps involved in the iterative process is typically very low and independent of k (say 4 or 5 steps). The reason for this is that the Newton iteration is quadratically convergent (see [76]). So the overall algorithm is still $O(n^2)$.

The numerical accuracy of the algorithm this time is quite satisfactory. The reason for this is that the Newton correction $\delta_i(\lambda)$ – or $s_i(\lambda)$ – is computed at each step from the *residual* $\Delta\Phi(\lambda)$ for the current approximate solution solution $r_i(\lambda)$. In other words, even if $\delta_i(\lambda)$ is not computed very accurately, this is not crucial since the next Newton step will be to evaluate $\Delta\Phi(\lambda)$ first and compute an additional correction from there on. It turns out that the fast solver for (6.45) may indeed suffer from loss of accuracy, but the iterative algorithm in a sense absorbs these errors. This of course does not guarantee that accurate results are always obtained. Indeed, the spectral factorization problem may be poorly conditioned in which case any algorithm will yield poor results.

Because of the link of this problem to that of Riccati equations it is known that the spectral factorization problem has a bad condition number when some of the zeros of $\Phi(\lambda)$ are close to the unit circle. Because of symmetry, $\Phi(\lambda)$ then also has almost double zeros which have to be separated in $r(\lambda)$ and $r_*(\lambda)$. It is well known that the Newton Raphson scheme, which is in fact a special case of this algorithm also will have problems.

We note here that both the gcd problem and spectral factorization problem could have been solved as well by just using a root finder. For the gcd of $a(\lambda)$ and $b(\lambda)$ one would just compute the roots of these two polynomials and check if there are any common ones. Obviously, this will not work in practice since roots may be sensitive to perturbations. For the spectral factorization problem one could just find the roots of the polynomial $\lambda^k.\ \Phi(\lambda)$ and choose the stable ones to go in the factor $r(\lambda)$. This approach would destroy the symmetry in this problem and may given an incorrect degree of $r(\lambda)$ if some of the roots are close to the unit circle again. So the above algorithm in general is more reliable.

6.5 Conclusion

In this chapter we showed that for polynomial models you typically tend to have a compacter representation of your model and therefore also faster algorithms.

This is typically the case for SISO system where the model and the design/analysis algorithms involve only two polynomials. These algorithms are then usually linked to root finding or to the Padé, Schur or Euclidean algorithm. Each of these is $O(k^2)$ in complexity versus $O(k^3)$ for a comparable algorithm in state-space form. The numerical stability of these algorithms on the other hand is often questionable and has to be analyzed carefully. We showed here that the Euclidean algorithm is unstable and showed the same for the Padéalgorithm earlier. But ideas of iterative refinement or look ahead schemes may salvage these algorithms without sacrificing their speed. An example of this is the look ahead Padéscheme.

For polynomial matrices the advantages are less obvious as far as complexity is concerned and stability problems are worse. Therefore we recommend to use state-space realizations instead. E.g. for the simple problem of finding the zeros of a polynomial matrix $P(\lambda)$, the state-space approach is still the only reliable method around [139].

Bibliography

[1] B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, NJ, 1979.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorenson. *LAPACK User's Guide*. SIAM, Philadelphia, PA, 1992.

[3] G. S. Axelby, A. J. Laub, and E. J. Davison. Further discussion on the calculation of transmission zeros. *Automatica*, 14:403–405, 1978.

[4] P. Bailey. A theoretical analysis of greatest common divisor algorithms. Master's thesis, University of Illinois, Urbana, IL, 1993.

[5] S. Barnett. *Polynomials and Linear Control Systems*. Dekker Inc., New York, NY, 1983.

[6] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donatoand J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. Templates for the solution of linear systems: building blocks for iterative methods. *SIAM*, NEED VOLUME(NEED NUMBER):NEED PAGES, NEED MONTH 1993.

[7] R. H. Bartels and G. W. Stewart. Solution of the matrix equation $AX + XB = C$. *Communications of the ACM*, 15(9):820–826, September 1972.

[8] R. Bellman. Some inequalities for the square root of a positive definite matrix. *Linear Algebra and its Applications* **I**, pages 321–324, 1968.

[9] P. Benner, V. Mehrmann and H. Xu. A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils. *Numer. Math.* 78:329–358, 1998.

[10] G. J. Bierman. *Factorization Methods for Discrete Sequential Estimation*. Academic Press, New York, NY, 1977.

[11] G. J. Bierman and C. L. Thornton. Numerical comparison of Kalman filter algorithms—orbit determination case study. *Automatica*, 13(23–35), 1977.

[12] D. L. Boley. On Kalman's procedure for the computation of the controllable/observable canonical form. *SIAM J. Control & Optimization.*, 18(6):624–626, November 1980.

[13] R. Byers. Numerical condition of the algebraic Riccati equation. *Contemp. Math.* 47:35–49, 1985.

[14] C. Brezinski. *History of continued fractions and Pade approximants*. Springer-Verlag, Berlin, 1991.

[15] H. Brezinski and O. Padé, editors. *Librairie Scientifique et Technique A*. Blanchard, Paris, 1984.

[16] A. Bultheel. Error analysis of incoming and outgoing schemes for the trigonometric moment problem. In Van Rossum and de Bruin, editors, *Lecture notes in Mathematics*, pages 100–109. Springer-Verlag, Berlin, 1980.

[17] S. Cabay and R. Meleshko. A weakly stable algorithm for padé approximants and the inversion of hankel matrices. *SIAM Matr. Anal. & Appl.*, 14:735–765, 1993.

[18] R. M. Chamberlain. An alternative view of lu factorization with partial pivoting on a hypercube multiprocessor. In *Hypercube Multiprocessors 1987: Proceedings of the Second Annual Conference on Hypercube Multiprocessors*, pages 59–575, Knoxville, TN, September 29-October 1, 1987. SIAM.

[19] M. Chilali, P. Gahinet and P. Apkarian. Robust pole placement in LMI regions. *IEEE Trans. Aut. Contr.*, 44(12):2257–2270, 1999.

[20] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson. An estimate for the condition number of a matrix. *SIAM Journal on Numerical Analysis*, 16(2):368–375, April 1979.

[21] T. P. Coleman and C. F. Van Loan. *Handbook for Matrix Computations*. SIAM, Philadelphia, PA, 1988.

[22] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[23] L. S. de Jong. Towards a formal definition of numerical stability. *Numer. Math.*, 28:211–220, 1977.

[24] S. de Jong. Numerical aspects of the recursive realization algorithms. *SIAM J. Contr. Optimiz.*, 16:646–659, 1978.

[25] J.-M. Delosme and I. Ipsen. Parallel solution of symmetric positive definite systems with hyperbolic rotations. *Lin. Alg. & Appl.*, 77:75–111, 1986.

[26] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Stat. Comput.*, 11:893–912, 1990.

[27] B. Dickinson, T. Kailath, and M. Morf. Canonical matrix fraction and state-space descriptiosn for deterministic and stochastic linear systems. *IEEE Trans. Automat. Control*, AC-19:656–667, December 1974.

[28] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart. *LINPACK User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1979.

[29] R. Eising. Between controllable and uncontrollable. *Syst. Contr. Lett.*, 4:263–264, 1984.

[30] L. Elsner and C. He. An algorithm for computing the distance to uncontrollability. *Syst. Contr. Lett.*, 17:453–464, 1991.

[31] A. Emami-Naeini and G. F. Franklin. Deadbeat control and tracking of discrete-time systems. *IEEE Trans. Automat. Control*, AC-27(1):176–181, Feb 1982.

[32] A. Emami-Naeini and P. Van Dooren. Computation of zeros of linear multivariable systems. *Automatica*, 18(4):415–430, July 1982.

[33] G. E. Forsythe, M. A. Malcolm, and C. B. Moler. *Computer Methods for Mathematical Computations*. Prentice Hall, Englewood Cliffs, NJ, 1977.

[34] G. E. Forsythe and C. B. Moler. *Computer Solution of Linear Algebraic Systems*. Prentice Hall, Englewood Cliffs, NJ, 1967.

[35] J. G. F. Francis. The QR transformation I. *Comput. J.*, 4:265–271, 1961.

[36] J. G. F. Francis. The QR transformation II. *Comput. J.*, 4:332–345, 1962.

[37] R. Freund and H. Zha. Stable algorithms for fast triangular factorization of general Hankel and Toeplitz matrices. To appear.

[38] F. R. Gantmacher. *The Theory of Matrices*. Chelsea, New York, NY, 1960.

[39] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler. Matrix eigensystem routines–EISPACK guide extension. *Lecture Notes in Computer Science*, 51, 1977. Springer-Verlag, New York.

[40] J. D. Gardiner, A. J. Laub, J. J. Amato, and C. B. Moler. Solution of the Sylvester matrix equation $AXB^T + CXD^T = E$. *ACM Trans. Math. Software*, 18:223–231, 1992.

[41] W. M. Gentleman and H. T. Kung. Matrix triangularization by systolic arrays. In *Proceedings of the SPIE, Real Time Signal Processing IV*, volume 298, pages 19–26, 1981.

[42] W. M. Gentleman. Least squares computation by Givens transformations without square roots. *JIMA*, vol 12, pages 329–336, 1973.

[43] G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.*, 2:205–224, 1965.

[44] G. H. Golub, S. Nash, and C. Van Loan. A Hessenberg-Schur method for the problem $AX + XB = C$. *IEEE Trans. Automat. Control*, AC-24(6):909–913, December 1979.

[45] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numer. Math*, 14:403–420, 1970.

[46] G. H. Golub and C. Van Loan. *Matrix Computations, 2nd ed.* Johns Hopkins University Press, Baltimore, MD, 1989. First Edition.

[47] G. H. Golub and J. H. Wilkinson. Note on the iterative refinement of least squares solution. *Numer. Mathematik*, 9:139–148, 1966.

[48] G. H. Golub and J. H. Wilkinson. Ill-conditioned eigensystems and the computation of the Jordan canonical form. *SIAM Rev.*, 18:579–619, 1976.

[49] G. Golub, B. Kågström, and P. Van Dooren. Direct block tridiagonalization of single-input single-output systems. *Systems & Control Letters*, 18:109–120, 1992.

[50] W. Gragg and A. Lindquist. On the partial realization problem. *Linear Algebra & Applications*, 50:277–319, 1983.

[51] E. Grimme, D. Sorensen, and P. Van Dooren. Mode reduction of large, sparse systems via an implicitly restarted Lanczos method. Submitted to *the 1994 American Control Conf., Baltimore, MD*, 1993.

[52] M. T. Heath, A. J. Laub, C. C. Paige, and R. C. Ward. Computing the singular value decomposition of a product of two matrices. *SIAM Journal of Science Stat. Computer*, 7(4):1147–1159, October 1986.

[53] D. Heller. A survey of parallel algorithms in numerical linear algebra. *SIAM Review*, 20(4):740–777, October 1978.

[54] P. Henrici. *Essentials of Numerical Analysis.* Wiley, New York, NY, 1982.

[55] G. Hewer and C. Kenney. The sensitivity of the stable Lyapunov equation. *SIAM J. Contr. Optim.* 26:321–344, 1988.

[56] B. L. Ho and R. E. Kalman. Effective construction of linear state-variable models from input/output functions. *Regelungstechnik*, 14:545–548, 1966.

[57] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[58] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, 1991.

[59] A. S. Householder. *The Theory of Matrices in Numerical Analysis.* Dover, New York, NY, 1964.

[60] I. M. Jaimoukha and E. M. Kasenally. Oblique projection methods for large scale model reduction. To appear, 1993.

[61] A. H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic, New York, NY, 1970.

[62] R. L. Johnson. *Numerical Methods: A Software Approach.* Wiley, New York, NY, 1982.

[63] D. Kahaner, C. Moler, and S. Nash. *Numerical Methods and Software*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[64] T. Kailath. Some new algorithms for recursive estimation in constant linear systems. *IEEE Transaction on Information Theory*, IT-19(6):750–760, 1973.

[65] T. Kailath. *Linear Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1980.

[66] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME. (J. Basic Eng.)*, 92D:34–45, March 1960.

[67] P. G. Kaminski, A. Bryson, and S. Schmidt. Discrete square root filtering—a survey of curren techniques. *IEEE Trans. Automat. Control*, AC-16:727–736, December 1971.

[68] J. Kautsky, N. K. Nichols, and P. Van Dooren. Robust pole assignment in linear state feedback. *Int. J. Control*, 41(5):1129–1155, 1985.

[69] C. Kenney and G. Hewer. The sensitivity of the algebraic and differential Riccati equations. *SIAM J. Contr. Optim.* 28:50–69, 1990.

[70] G. Klein and B. Moore. Eigenvalue-generalized eigenvector assignment with state feedback. *IEEE Aut. Contr.*, AC-22:140–141, 1977.

[71] V. C. Klema and A. J. Laub. The singular value decomposition: Its computation and some applications. *IEEE Trans. Automat. Control*, AC-25(2):164–76, April 1980.

[72] E. G. Kogbetliantz. Solutions of linear equations by diagonalization of coefficient matrix. *Quart. Appl. Math.*, 13:123–132, 1956.

[73] M. M. Konstantinov, P. H. Petkov, and N. D. Christov. Synthesis of linear systems with desired equivalent form. *J. Computat. Appl. Math.*, 6:27–35, 1980.

[74] M. M. Konstantinov, P. H. Petkov, and N. D. Christov. Sensitivity analysis of of the feedback synthesis problem. *IEEE Trans. Aut. Contr.*, 42:568–573, 1997.

[75] M. Konstantinov, P. Petkov, and D. Gu. Improved perturbation bounds for general quadratic matrix equations. *Numer. Func. Anal. Optim.*, 20:717–736, 1999.

[76] V. Kucera. *Discrete Linear Control: The Polynomial Equation Approach*. John Wiley, New York, NY, 1979.

[77] I. D. Landau. *Adaptive Control: The Model Reference Approach*. Marcel Dekker, New York, NY, 1979.

[78] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice Hall, Englewood Cliffs, NJ, 1974.

[79] R. E. Lord, J. S. Kowalik, and S. P. Kumar. Solving linear algebraic equations on a MIMD computer. In *Proceedings of the 1980 Int. Conf. on Parallel Proc.*, pages 205–210, 1980.

[80] D. G. Luenberger. Observers for multivariable systems. *IEEE Trans. Automat. Control*, AC-11:190–197, 1966.

[81] F. T. Luk. A rotation method for computing the QR factorization. *SIAM J. Sci. Stat. Comput.*, 7:452–459, 1986.

[82] J. Makhoul. Linear prediction: a tutorial review. *Proc. IEEE*, 63:561–580, 1975.

[83] T. Manteuffel. Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration. *Numer. Math.*, 31:183–208, 1978.

[84] R. S. Martin and J. H. Wilkinson. Similarity reduction of a general matrix to Hessenburg form. *Numer. Math*, 12:349–368, 1968.

[85] J. L. Massey. Shift register synthesis of BCH decoding. *IEEE Trans. Inf. Th.*, IT-15:122–127, 1969.

[86] V. Mehrmann. *The Autonomous Linear Quadratic Control Problem.* Lecture Notes in Control and Information Sciences. Springer-Verlag, Heidelberg, 1991.

[87] V. Mehrmann and H. Xu. An analysis of the pole placement problem I. *Electr. Trans. Numer. Anal.*, 4:89–105, 1996.

[88] V. Mehrmann and H. Xu. An analysis of the pole placement problem II. *Electr. Trans. Numer. Anal.*, 5:77–97, 1997.

[89] J. Mendel. Computational requirements for a discret Kalman filter. *IEEE Trans. Automat. Control*, AC-16(6):748–758, 1971.

[90] W. Miller and C. Wrathall. *Software for Roundoff Analysis of Matrix Algorithms.* Academic Press, New York, NY, 1980.

[91] G. Miminis and C. C. Paige. An algorithm for pole assignment of time invariant linear systems. *Int. J. Contr.*, 35:341–345, 1982.

[92] G. S. Miminis and C. C. Paige. A direct algorithm for pole assignment of time-invariant multi-input linear systems using state feedback. *Automatica*, 24:343–356, 1988.

[93] P. Misra and R. V. Patel. Numerical algorithms for eigenvalue assignment by constant and dynamic output feedback. *IEEE Trans. Automat. Contr.*, 34:579–588, 1989.

[94] J. J. Modi. *Parallel Algorithms and Matrix Computations.* Oxford University Press, Oxford, England, 1988.

[95] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20(4):801–836, October 1978.

[96] C. B. Moler and G. W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal of Numerical Analysis*, 10(2):241–256, April 1973.

[97] M. Moonen, B. De Moor, L. Vandenberghe, and J. Vandewalle. On- and off-line identification of linear state-space models. *Int. J. Contr.*, 49:219–232, 1989.

[98] M. Morf and T. Kailath. Square-root algorithms for least squares estimation. *IEEE Trans. Automat. Control*, AC-20:487–497, August 1975.

[99] C. Mullis and R. Roberts. Synthesis of minimum roundoff noise fixed point digital filters. *IEEE Trans. Circuits and Systems*, CAS-23:551–562, 1976.

[100] C. Oara, R. Stefan and P. Van Dooren. Maximizing the stability radius. In *Proceed. Americ. Contr. Conf.*, TP06-5, pages 3035–3040, 2001.

[101] A. M. Ostrowski. On the spectrum of a one-parametric family of matrices. *Journal fur die Reine und Angewandte Mathematik*, 193:143–160, 1954.

[102] C. C. Paige. *Properties of numerical algorithms related to computing controllability. IEEE Trans. Automat. Control*, AC-26(1):130–138, February 1981.

[103] C. C. Paige. Practical use of the symmetric Lanczos process with reorthogonalization. *BIT*, 10:183–195, 1970.

[104] C. C. Paige. Covariance matrix representation in linear filtering. *Special Issue of Contemporary Mathematics on Linear Algebra and its Role in Systems Theory, AMS*, 1985.

[105] T. Pappas, A. J. Laub, and N. R. Sandell, Jr. On the numerical solution of the discrete-time algebraic Riccati equation. *IEEE Trans. Automat. Control*, AC-25(4):631–641, August 1980.

[106] B. Parlett. Reduction to tridiagonal form and minimal realizations. *SIAM J. Matrix Anal. Appl.*, 13:567–593, 1992.

[107] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ, 1980.

[108] L. Pernebo and L. M. Silverman. Model reduction via balanced state space representations. *IEEE Trans. Automat. Control*, AC-27(2):382–387, April 1982.

[109] P. Petkov, N. Christov and M. Konstantinov. *Computational Methods for Linear Control Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1991.

[110] J. R. Rice. A theory of condition. *SIAM J. Numer. Anal.*, 3:287–310, 1966.

[111] J. R. Rice. *Matrix Computations and Mathematical Software*. McGraw-Hill, New York, NY, 1981.

[112] J. R. Rice. *Numerical Methods, Software, and Analysis*. McGraw-Hill, New York, NY, 1983.

[113] J. Rissanen. Recursive identification of linear systems. *SIAM J. Control*, 9:420–430, 1971.

[114] H. Rosenbrock. *State-space and multivariable theory*. Wiley, New York, NY, 1970.

[115] Y. Saad. *Numerical methods for large scale problems*. Manchester Univ Press, Manchester UK, 1992.

[116] A. P. Sage and C. C. White. *Optimum systems control*. Prentice Hall, Englewood Cliffs, 1977.

[117] A. H. Sameh and D. J. Kuck. On stable parallel linear system solvers. *J. Assoc. Comput. Mach.*, 25:81–91, 1978.

[118] L. M. Silverman. *Representation and realization of time-variable linear systems*. PhD thesis, Columbia University, New York, NY, 1966. Dept. El. Eng.

[119] V. Sima. *Algorithms for Linear Quadratic Optimization*. Marcel Dekker Inc. New York, 1996.

[120] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystem Routines–EISPACK Guide*, chapter 6. Springer-Verlag, New York, 2nd edition, 1976. Lecture Notes in Control and Information Sci.

[121] D. C. Sorensen. Implicit application of polynomial filters in a K-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[122] G. W. Stewart. Error and perturbation bounds associated with certain eigenvalue problems. *SIAM Review*, 15:727–764, 1973.

[123] G. W. Stewart. *Introduction to Matrix Computations*. Academic Press, New York, NY, 1973.

[124] G. W. Stewart. Algorithm 506 - HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix. *ACM Trans. Math. Software*, 2:275–280, 1976.

[125] G. W. Stewart. On the perturbation of pseudo-inverses, projections, and linear least squares. *SIAM Rev.*, 19:634–662, 1977.

[126] G. W. Stewart. Rand degeneracy. *SIAM J. Numer. Anal.*, 5:403–413, 1984.

[127] G. W. Stewart and J. G. Sun. *Matrix Perturbation Theory*. Academic Press, New York, NY, 1990.

[128] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer-Verlag, New York, NY, 1980.

[129] E. C. Y. Tse, J. V. Medanić, and W. R. Perkins. Generalized Hessenberg transformations for reduced-order modeling of large-scale systems. *Int. J. Control*, 27(4):493–512, April 1978.

[130] A. van der Sluis. Stability of the solution of linear least squares problems. *Numerische Mathematik*, 23:241–254, 1975.

[131] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 27:103–141, 1979.

[132] P. Van Dooren. Computing the eigenvalues of a polynomial matrix. In *Proceedings of the IBM-NFWO Symposium*, pages 213–223, Brussels, Belgium, December 1979.

[133] P. Van Dooren. A generalized eigenvalue approach for solving Riccati equations. *SIAM J. Sci. Stat. Comput.*, 2(2):121–135, June 1981. Erratum in Vol.4, No. 4, Dec. 83.

[134] P. Van Dooren. Deadbeat control, a special inverse eigenvalue problem. *BIT*, 24:681–699, December 1984.

[135] P. Van Dooren. Reduced order observers: A new algorithm and proof. *Systems & Control Lett.*, 4:243–251, July 1984.

[136] P. Van Dooren. Comments on 'minimum-gain minimum-time deadbeat controllers'. *Systems & Control Letters*, 12:93–94, 1989.

[137] P. Van Dooren. Numerical aspects of system and control algorithms. *Journal A*, 30:25–32, 1989.

[138] P. Van Dooren. Upcoming numerical linear algebra issues in systems and control theory, September 1992.

[139] P. Van Dooren and P. Dewilde. The eigenstructure of an arbitrary polynomial matrix: Computational aspects. *Lin. Alg. & Appl.*, 50:545–580, 1983.

[140] P. Van Dooren and M. Verhaegen. On the use of unitary state-space transformations. In B. N. Datta, editor, *Linear Algebra and its Role in Linear Systems Theory*, chapter 47, pages 447–463. AMS Contemporary Mathematics Series, Providence, RI, 1985.

[141] P. M. Van Dooren. The generalized eigenstructure problem in linear system theory. *IEEE Trans. Automat. Control*, AC-26(1):111–129, February 1981.

[142] C. F. Van Loan. The sensitivity of the matrix exponential. *SIAM J. Numer. Anal.*, 14:971–981, 1977.

[143] G. Verghese, P. Van Dooren, and T. Kailath. Properties of the system matrix of a generalized state-space system. *Int. J. Control*, 30(2):235–243, October 1979.

[144] M. Verhaegen and P. Van Dooren. Numerical aspects of different Kalman filter implementations. *IEEE Trans. Automat. Control*, AC-31:907–917, October 1986.

[145] M. Verma. *Synthesis of infinity-norm optimal linear feedback systems.* PhD thesis, University of Southern California, December 1985.

[146] Z. Vostry. New algorithm for polynomial spectral factorization with quadratic convergence. *Kybernetika*, 11:415–422, 1975.

[147] J. H. Wilkinson. *Rounding Errors in Algebraic Processes.* Princeton Hall, Englewood Cliffs, NJ, 1963.

[148] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford, England, Clarendon Press, 1965.

[149] J. H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation* , volume II. Springer-Verlag, New York, NY, 1971.

[150] W. M. Wonham. On a matrix equation of stochastic control. *SIAM J. Control*, 6:681–697, 1968.

[151] D. Youla and N. Tissi. N-port synthesis via reactance extration, Pt. I. *IEEE Int. Conv. Rec.*, 14:183–208, 1966.

[152] H. Zeiger and A. McEwen. Approximate linear realizations of given dimension via Ho's algorithm. *IEEE Trans. Automat. Control*, AC-19:153–156, 1974.