Time-Variant and Quasi-separable Systems

Matrix theory is the lingua franca of everyone who deals with dynamically evolving systems, and familiarity with efficient matrix computations is an essential part of the modern curriculum in dynamical systems and associated computation.

This is a master's-level textbook on dynamical systems and computational matrix algebra. It is based on the remarkable identity of these two disciplines in the context of linear, time-variant, discrete-time systems and their algebraic equivalent, quasi-separable systems. The authors' approach provides a single transparent framework that yields simple derivations of basic notions, as well as new and fundamental results such as constrained model reduction, matrix interpolation theory and scattering theory. This book outlines all the fundamental concepts that allow readers to develop the resulting recursive computational schemes needed to solve practical problems.

An ideal treatment for graduate students and academics in electrical and computer engineering, computer science and applied mathematics.

Patrick Dewilde was a professor at the Delft University of Technology for 31 years and previously Director of the Delft Institute for Microelectronics, Chairman of a major Dutch research funding agency and Director of the Institute of Advanced Study of the Technical University of Munich. He is an Institute of Electrical and Electronics Engineers (IEEE) Fellow, a winner of the IEEE Belevitch Award and an elected member of the Dutch Royal Academy of Arts and Science.

Klaus Diepold is a professor at the Technical University of Munich. During his time in industry, he was the chief architect of award-winning software tools MotionPerfect and SteadyHand and is a coauthor of *Understanding MPEG-4: Technology and Business Insights* (2004). He is a board member of the Center for Digital Technology and Management and Fellow for Innovation in University Education. He was awarded the Start-Up Mentor of Excellence Award in 2021 by the Technical University of Munich. In 2023 he received the Unipreneurs Award by the Federal Ministry for Education and Research for his entrepreneurial activities.

Alle-Jan Van der Veen is Professor and Chair of the Signals and Systems group at Delft University of Technology. He is an IEEE Fellow and IEEE SPS Vice President – Technical Directions. Previous IEEE positions include: Editor-in-Chief of *Transactions on Signal Processing*, Chairman of the Signal Processing Society, elected member of the SPS Board of Governors and Chair of the SPS Signal Processing Theory and Methods Technical Committee and the Kilby Medal selection committee.

Time-Variant and Quasi-separable Systems

Matrix Theory, Recursions and Computations

PATRICK DEWILDE
Technical University of Munich

KLAUS DIEPOLD
Technical University of Munich

ALLE-JAN VAN DER VEEN
Delft University of Technology

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

Shaftesbury Road, Cambridge CB2 8EA, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre,
New Delhi – 110025, India

103 Penang Road, #05–06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of Cambridge University Press & Assessment,
a department of the University of Cambridge.

We share the University's mission to contribute to society through the pursuit of
education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781009455626

DOI: 10.1017/9781009455640

When citing this work, please include a reference to the DOI 10.1017/9781009455640

First published 2025

A catalogue record for this publication is available from the British Library

Library of Congress Cataloging-in-Publication Data
Names: Dewilde, Patrick, author. | Diepold, Klaus, author. |
Veen, Alle-Jan van der, author.
Title: Time-variant and quasi-separable systems :
matrix theory, recursions and computations / Patrick Dewilde,
Klaus Diepold, Alle-Jan Van der Veen.
Description: Cambridge ; New York, NY : Cambridge University Press, 2025. |
Includes bibliographical references and index.
Identifiers: LCCN 2023051167 | ISBN 9781009455626 (hardback) |
ISBN 9781009455640 (ebook)
Subjects: LCSH: Matrices. | Linear time invariant systems. | Separable
algebras. | Mathematical optimization. | Computer algorithms.
Classification: LCC QA188 .D49 2025 | DDC 512.9/434–dc23/eng/20240404
LC record available at https://lccn.loc.gov/2023051167

ISBN 978-1-009-45562-6 Hardback

Cambridge University Press & Assessment
978-1-009-45562-6 — Time-Variant and Quasi-separable Systems
Patrick Dewilde , Klaus Diepold , Alle-Jan Van der Veen
Frontmatter
More Information

Contents

Cambridge University Press & Assessment
978-1-009-45562-6 — Time-Variant and Quasi-separable Systems
Patrick Dewilde , Klaus Diepold , Alle-Jan Van der Veen
Frontmatter
More Information

Preface

This book develops the intimate connection between linear time-variant dynamical systems and matrix calculus.

Both dynamical systems and computers implementing matrix calculus are ubiquitous in our present technical world. Dynamical systems are often steered or controlled by an embedded computer that executes an algorithm. Conversely, a computer executing an algorithm may be viewed as a dynamical system in its own right. This correspondence can be exploited to great benefit for the understanding of the behavior of a dynamical system and, by the same token, the construction of accurate and efficient computations. The present book develops this connection for the most simple and important class, linear systems, and, equivalently, matrix computations.

Traditionally, system theory and computational algebra have developed and evolved in very different ways. We show in this book that bringing the fields together in one body of knowledge has great advantages both from a theoretical and a practical point of view. In doing so, we show that only a few amazingly simple basic concepts and numerical methods are needed to cover the whole field, provided one is willing to look at the key issues in an unadorned and fresh way.

The insight that such an approach was possible arose in the seminal work of Bellman and Kalman, who ventured into the new environment of time-variant systems in order to solve estimation and control problems related to the Apollo spaceflight mission. Classically, such problems are treated with Laplace and z-transform theory and have an algebraic basis in complex function analysis and module theory. It soon appeared that an effective time-variant algebra was lacking. The latter was developed over time, on the mathematical side as nest algebras, on the system theory side as time-variant systems, and on the numerical side, in a limited way, as semi-separable systems. It took time to harmonize the various approaches, but the net result is that discrete-time, time-variant linear systems are now fully equivalent to a generalization of semi-separable matrix systems, which has been given the name *quasi-separable systems*, to our knowledge by Israel Gohberg. The present book offers a systematic but also fully didactical introduction to the unified theory.

The basis for the transformation or rejuvenation of system theory presented in this book is both traditional and remarkably simple. The central idea may be informally formulated as “the state of a system is what the system remembers of its past at any moment,” or, equivalently, “the minimal information needed at any moment to produce its future development, given future inputs.” This information presents itself as Nerode equivalent classes and, in the traditional thinking, as an *ideal* or a *submodule* in an overall algebraic setting – that is, strong structures that incorporate the time invariance. It was thought for a long time that these notions could not extend beyond a time-invariant framework. We show that, on the contrary, a systematic exploiting of the notion of state in a time-variant setting is not only possible but also produces most if not all essential system operations and properties. Time invariance is not essential, but a new, alternative, and as it turns out, more elementary but also more powerful algebraic framework is needed.

At the numerical algebra side, the power of orthogonal transformations gradually became evident, exemplified by the present day resurgence of QR factorization and the singular value decomposition (SVD) as the most ubiquitous numerical methods, insights going way back to Jacobi and Gauss, but then resurrected by numerical analysts such as Givens and Householder. It turns out that the only central mathematical concept needed in dynamical system theory is that of "range of an operator," often represented by a suitable orthonormal basis, and computed recursively using QR- or LQ-type transformations (its dual), or perhaps SVD for even more accuracy.

In the present book, we treat operations on matrices as a dynamical system in its own right, that is, as respecting the recursive order viewed as evolution in time. General matrix computations can just be viewed as evolution of a numerical system and, conversely, the evolution of time-variant dynamical systems consists of matrix operations. The notions "matrix" plus "linear progression in time" or "indexing order" coincide with the notion "discrete-time dynamical system" in this approach.

Basic matrix operations are then operations on systems and vice versa. For the development of the whole theory, only the most elementary matrix operations are needed, namely additions and multiplications as they occur in QR and LQ factorizations. The majority of results ranging from system identification to system optimization, estimation and model reduction can remarkably be obtained without much more than elementary matrix operations. Conversely, this approach leads to novel matrix methods and new results in matrix theory, especially in the areas of matrix inversion and matrix approximation, interpolation and factorization.

To realize this plan, an extension of the matrix theory framework is needed in order to accommodate dynamically changing dimensions. A typical time-variant system starts at some time index, its evolving state, inputs and outputs have variable dimensions during its existence, and it dies out or stops operating at some time. Variable dimensions force an adaptation of the classical treatment of matrices, including the introduction of zero dimensions (i.e., empty matrices), matrices whose elements are matrices themselves and the systemic use of block diagonal matrices as building blocks for system representations. These extensions of classical matrix theory turn out to be pretty natural and to conform without difficulty to computer science practices.

The result is that time-variant system theory and elementary matrix calculus largely coincide. This is a major didactical advantage of the approach. No complex function calculus, transform theory, module theory or whatever complicated algebraic structures are needed for most, if not all of the results, provided one stays within the context of systems evolving in discrete time. With some extra effort, many time-invariant results can be derived from the time-variant basics, but they often require the solution of an additional fixed-point problem. Historically, the time-invariant way was thought to be simpler and more insightful. The opposite is true: invariance complicates matters considerably and may even be seen as "unnatural." By going time variant, one can sidestep most of the classical, transform-based literature in favor of straight and simple matrix theory, or, as a modern saying goes, *array processing* – a great logical and didactical simplification.

Motivation for the Method

System theory has mostly been restricted traditionally to linear, time-invariant systems. The consequence is that matrices that describe the global behavior of the system have a special structure, namely (block) Toeplitz or (block) Hankel, and the theory is geared toward representing such structures algebraically, mainly using transform theory, which tends to become a goal in itself. However, many instances of (linear) recursive computations deal with unstructured or weakly structured global system matrices. We show that the more general time-variant system theory developed in this book succeeds surprisingly easily in remedying the lack of correspondence between system theory and general matrix algebra of the traditional approach. This move, which at first seems to make things more complex, turns out to produce a great theoretical simplification, thereby using and even highlighting all the main principles and leading to all the main basic results.

With reference to the immense literature on system theory, control theory, optimization and filtering, it is perhaps hard to believe that there exists a simple and easily accessible computational approach to all these topics, given the variety of mathematical methods people have developed to attack the various problems they encountered, often using ad hoc or brute-force methods. But *there is* a unifying approach, and the present book shows how even *profound* system-theoretic problems can be approached with elementary matrix algebra.

The ability to compute effectively *and* to connect computations to fundamental issues in systems engineering is a major challenge limiting the ability of our future engineers, irrespective of whether they are more active in engineering or in data numerics; see Figure 1. One of the most appealing properties of the proposed approach is the fact that matrix computations and system theory are two sides of the same coin: methods of one field are directly relevant to the other.

The challenge to students and researchers in systems engineering appears to be the mastery of elementary matrix algebra, especially of the *geometry* connected to it. A matrix defines a linear operator, and various related subspaces (range, co-range, kernel and co-kernel) play the central geometric role. Most algebraic operations aim at characterizing these spaces, mostly by deriving orthonormal bases for them, an operation that in signal processing terms is called *orthogonal filtering*. Depending on the application at hand, whether realization theory, optimal control or estimation theory,

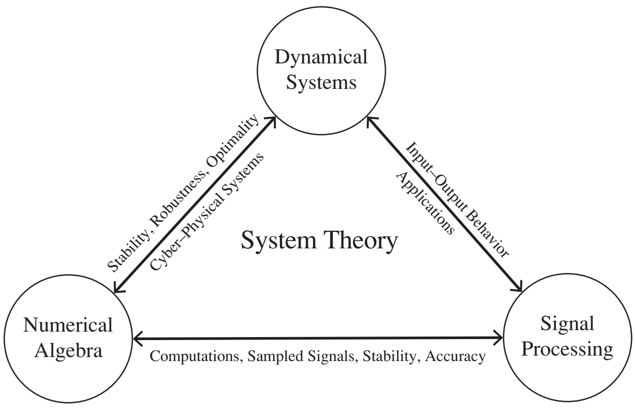

Figure 1 Our main disciplines are closely related.

the filtering procedure has an idiosyncratic name and is called, for example, dynamic programming, a Wiener filter, a Kalman filter or a selective filter. They all have similar operations that combine system-theoretic recursions with recursive numerical methods in common.

Conversely, the study we propose equips the students with a wealth of examples through which they can refine their intimate knowledge of matrix algebra as the main mathematical vehicle to achieve key results in numerical analysis and signal processing. Familiarity with matrix algebra is a *to be or not be* question for a modern engineer who has to handle data in a variety of situations, and the detailed study of system theory offers an ideal environment to develop that familiarity. Such an achievement can be seen as the main end term of the course we propose.

Adjoining Website

There is an adjoining website to this book containing:

- worked examples in concrete applications;
- computer programs of algorithms;
- PowerPoint presentations;
- advanced topics; and
- related papers from the open literature.

Prerequisites

We assume knowledge of elementary (undergraduate-level) matrix algebra (real and complex numbers, vectors, matrices, matrix products, echelon forms, matrix inversion, eigenvectors and eigenvalues, QR decomposition and SVD) and basic analysis (functions, limits, ordinary and partial differentials, integration, series). Matrices are our lingua franca, and our students and readers will develop familiarity with them by working through the book. More advanced mathematical properties (in particular, some elementary properties of Hilbert spaces) are introduced where they are used, and this does not happen in core theory, only in places where connections are made with other approaches. All these topics are covered by many textbooks on matrix or linear algebra, for example [40, 63], and textbooks in analysis and Hilbert space theory, for example [59].

A Summary of the Chapters

The book starts out with a motivating chapter to answer the question *why is it worthwhile to develop system theory?* To do so, we jump fearlessly into the very center of our methods, using a simple and straight example of *optimal control*. Although optimization is not our main subject – that is system theory – it provides for one of the main application areas, namely the optimization of the performance of a dynamical system in a time-variant environment (e.g., driving a car or sending a rocket to the moon). The chapter starts out with a review of the *Moore–Penrose pseudo-inverse*, which is a central concept of matrix algebra, used throughout the book. Next it describes a simple case of optimal control, which is first solved in a global way and then in a much more attractive recursive way called *dynamic programming*. The chapter then ends by showing how the method generalizes to linear, discrete-time, time-variant models.

Chapter 2 moves into a more philosophical mood to introduce *the basic notions on which system theory is based*. In this endeavor, it follows the insights first provided by Rudy Kalman and his coauthors in [49], and it does so as a narrative, laying the foundations for what will become the mathematical framework in the remainder of the book. A few fundamental and very appealing notions such as “state,” “behavior,” “reachability” and “observability” provide a sufficient basis for the whole theory (sufficient in the sense that no further fundamental notions are needed). In further chapters, we gradually develop the remarkable mathematical consequences of the concepts introduced here, exclusively using elementary matrix algebra and a necessary organizational extension of it. The chapter ends by introducing the reader to the main types of system they may encounter in practice, thereby illustrating how the basic concepts provide unity in diversity.

Chapter 3 starts developing our central *Linear Time-Variant (LTV) prototype environment*, a class that coincides perfectly with the linear algebra or matrix algebra context, making the correspondence systems-matrix computations a mutually productive reality. People familiar with the classical approach, in which the z-transform or other types of transform are used, will easily recognize the notational or graphic resemblance, but there is a major difference: everything remains elementary, no complex function calculus is involved and only the simplest matrix operations addition and multiplication are needed. Appealing expressions for the state-space realization of a system appear, as well as the global representation of the input–output operator in terms of four block-diagonal matrices $\{A, B, C, D\}$ and the (now non-commutative but elementary) *causal shift* Z. The consequences for and relation to linear time-invariant (LTI) systems and infinitely indexed systems are fully documented in the sections marked with $*$, which can be skipped by students or readers more interested in numerical linear algebra than in LTI system control or estimation.

From this point on, the main issues in system theory are tackled. The very first, considered in Chapter 4, is the all-important question of *system identification*. This is perhaps the most basic question in system theory and related linear algebra, with a strong pedigree starting from Kronecker's characterization of rational functions to its elegant solution for time-variant systems. Identification, often also called "realization," is the problem of deriving the internal system's equations (called state-space equations) from input–output data. In this chapter, we consider only the causal, or lower block triangular case, although the theory applies just as well to an anti-causal system, for which one lets the time run backwards, applying the same theory in a dual form.

In Chapter 5, we consider the central issue of *minimality* of the state-space system representation, as well as *equivalences of such representations*. The question introduces important new basic operators and spaces related to the state-space description. In our time-variant context, what we call the *Hankel operator* plays the central role, via a minimal composition (i.e., product) of a *reachability operator* and an *observability operator*. Corresponding results for LTI systems (a special case) follow readily from the LTV case as well as the theory for infinitely indexed systems, but they entail some extra complications, which are not essential for the main treatment offered and can be skipped on first reading.

Chapter 6 is a straightforward but essential chapter. It shows how the recursive structure of the state-space representations is exploited to make *elementary matrix operations* such as matrix addition and multiplication efficient. Also, elementary matrix inversion is treated for cases where the elementary inverse exists. The notions of *outer operator* and *inner operator* are introduced as basic types of matrices playing a central role in various specific matrix decompositions and factorizations to be treated in further chapters.

Several types of factorization solve the main problems of system theory (identification, estimation, system inversion, system approximation and optimal control). The factorization type depends on what kind of operator is factorized and what form the factors should have. Chapters 7 and 8 are therefore devoted to the two main types of factorization: Chapter 7 treats what is traditionally called *coprime factorization*, while Chapter 8 is devoted to *inner–outer factorization*. Coprime factorization, here called *external factorization* for more generality, characterizes the system's dynamics and plays a central role in system characterization and control issues. A remarkable result of our approach is the derivation of Bezout equations for time-variant and quasi-separable systems, made possible without the use of Euclidean divisibility theory. From a numerical point of view, all these factorizations reduce to recursively applied QR or LQ factorizations, applied on appropriately chosen operators.

Chapter 8 then considers the likely most important operation in system theory: *inner–outer (and its dual, outer–inner) factorization*. This factorization plays a different role than the previously treated external or coprime factorization, in that it characterizes properties of the *inverse or pseudo-inverse* of the system under consideration, rather than the system itself. An important point is that such factorizations are computed on the state-space representation of the original data. Inner–outer consists in nothing else but recursive QR factorization, as was already observed in our motivational Chapter 1, and outer–inner in recursive LQ factorization, in the somewhat unorthodox terminology used in this book for consistency reasons: QR for "orthogonal Q with right factor R" and LQ for "left factor L with orthogonal Q." These types of factorization play the central role in a variety of applications (such as optimal control and tracking, state estimation, system pseudo-inversion and spectral factorization). We conclude the chapter showing briefly how the time-variant linear results generalize to the nonlinear case.

The set of basic topics then continues with a major application domain of our theory: linear least-squares estimation (LLSE) of the state of an evolving system (also known as Kalman filtering – Chapter 9), which turns out to be an immediate application of the outer–inner factorization theory developed in Chapter 8. To complete this discussion, we also show how the theory extends naturally to cover the smoothing case (which is often considered "difficult").

Two chapters conclude the basic course.

Chapter 10 presents an alternative theory of external and coprime factorization, using polynomial denominators in the time-variant shift Z rather than inner denominators, as is done in Chapter 8. "Polynomials in the shift Z" are equivalent to block lower matrices with a support defined by a (block) staircase and are essentially different from the classical matrix polynomials of module theory, although the net effect on system analysis is similar. The polynomial method differs substantially and in a complementary way from the inner method. It is computationally much simpler but does not use orthogonal transformations. It offers the possibility of treating highly unstable systems using unilateral series. Also, this approach leads to famous Bezout equations, which, again, can be derived without the benefit of Euclidean divisibility methods.

Chapter 11 considers the *Moore–Penrose inversion of full matrices with quasi-separable specifications*, that is, matrices that decompose into the sum of a lower block triangular and a block upper triangular matrix, whereby each have a state-space realization given. We show that the Moore–Penrose inverse of such a system has, again, a quasi-separable specification globally of the same complexity as the original, and show how this representation can be recursively computed with three intertwined recursions. The procedure is illustrated on a 4×4 (block) example.

Chapters 12–17 exhibit further contributions of the theory of time-variant and quasi-separable systems to matrix algebra.

Chapter 12 considers another type of factorization of a quasi-separable system, namely *LU, or, equivalently, spectral factorization*. This type of factorization does not necessarily exist, and, when it exists, does not traditionally enjoy stable computation methods. Here we present necessary and sufficient existence conditions for the general quasi-separable case and a stable numerical algorithm to compute the factorization based on the quasi-separable representation. The algorithm uses orthogonal transformations and scalar normalizations exclusively, in contrast to classical Gaussian elimination.

Chapter 13 introduces a different kind of problem, namely *direct constrained matrix approximation via interpolation*, the constraint being positive definiteness. It is the problem of completing a positive definite matrix for which only a well-ordered partial set of entries is given (and also giving necessary and sufficient conditions for existence of the completion), or, alternatively, the problem of parametrizing positive definite matrices. This problem can be solved elegantly when the specified entries contain the main diagonal and further entries crowded along the main diagonal with a staircase boundary. This problem turns out to be equivalent to a constrained interpolation problem defined for a causal contractive matrix, with staircase entries again specified as before. The recursive solution calls for the development of a machinery known as *scattering theory*, which involves the introduction of non-positive metrics and the use of J-unitary transformations where J is a sign matrix.

Chapter 14 then develops the *scattering formalism*, whose usefulness for interpolation has been demonstrated in Chapter 13, for the case of systems described by state-space realizations, in preparation for the next three chapters, which use it to solve various further interpolation and embedding problems.

Chapter 15 shows how *classical interpolation problems of various types (Schur, Nevanlinna–Pick, Hermite–Fejer)* carry over to the time-variant and/or matrix situation. We show that they all reduce to a single generalized constrained interpolation problem, elegantly solved by time-variant scattering theory. An essential ingredient is the definition of the notion of *valuation* for time-variant systems, in generalization of the valuation in the complex plane provided by the classical z-transform.

Chapter 16 then provides for a further extension of constrained interpolation that is capable of solving the *constrained model reduction problem*, namely the generalization of Schur–Takagi-type interpolation to the time-variant setting. This remarkable result demonstrates the full power of time-variant system theory as developed in this book.

Chapter 17 completes the scattering theory with an elementary approach to inner embedding of a contractive, quasi-separable causal system (in engineering terms: the embedding of a *lossy* system in a *lossless* system, often called *Darlington synthesis*). Such an embedding is always possible in the finitely indexed case, but does not generalize to infinitely indexed matrices (this issue requires more advanced mathematical methods and lies beyond the subject matter of the book).

The appendix on the data model used throughout the book describes what can best be called an *algorithmic design specification*, that is, the functional and graphical characterization of an algorithm, chosen so that it can be translated to a computer architecture (be it in software or hardware). We follow hereby a powerful "data flow model" that generalizes the classical signal flow graphs and which can be further formalized to generate the information necessary for the subsequent computer system design at the architectural level (i.e., the assignment of operations, data transfer and memory usage.) The model provides for a natural link between mathematical operations and architectural representations. It is, by the same token, well adapted to the generation of parallel processing architectures.

Notation

We mostly use standard usage in linear algebra, but with some systematic differences, induced by keeping the notation close to the practice exercised by MATLAB. We do, however, use special notations for mathematical objects that occur often in our developments and try to avoid annoying overloads of symbols (which is sometimes impossible). Notation is an important issue, both in algebra and in computer science, and we try to be as straight and consistent as possible. Notations that deviate from common practice in either linear algebra or MATLAB are defined wherever they are introduced (this is especially true for "empty" objects, which play an important role in making the theory fully consistent). Also, we introduce the most important numerical methods in the chapters where they are intensely used. Here is a short summary of nonstandard notations used in this book:

- Many of our objects are matrices, and we often have to consider either special indexing conventions or take out submatrices from a given matrix. As such operations can become unwieldy, we adopt systematically a MATLAB-like annotation of index ranges. Suppose A is a matrix, then $A_{k:\ell,m:n}$ is a submatrix of A consisting of a selection of rows from index k to ℓ (inclusive) and columns from index m to n inclusive (if no confusion is expected, we may also write $A_{k:\ell}^{m:n}$: row indices at the bottom and column indices at the top). If rows or columns are not there originally, they are just considered empty (see the next item). We also systematically economize the notation (in contrast to many textbooks): A, a and $\mathbf{A}$ are different matrices, with $A_{i,j}$, $a_{i,j}$ and $\mathbf{A}_{i,j}$ being the different elements of each respectively.
- An index is actually also a function, namely from a subset of the natural numbers to the set of objects under consideration. So a_k can also be written as $a(k)$. We do this only when we want to emphasize or discuss this functional character. Often, either the functional arguments or the indices are omitted in a discussion or proof: these are then inferred from the context. This common practice enhances readability at the cost of precision (so we typically write A instead of $A_{1:n}^{1:m}$ for a simple $n \times m$ matrix.)
- Very often block matrices appear, that is, matrices whose entries are themselves matrices (called "blocks"). Blocks may consist of blocks themselves, but they are indexed in the usual fashion and have to be commensurate (dimensions in rows or columns must match throughout). For example: $A_{k,\ell}$ may be a block in a block matrix A, and $[A_{k,\ell}]_{m,n}$ is then a block entry at the position (m,n) in that original block. We do allow blocks with zero dimensions: they are just empty, but do have index numbers and dimensions. We introduce special notation for empty objects: an entry of dimensions $(0,1)$ is denoted by "–," an entry of dimensions $(1,0)$ by

"|" and one of dimensions $(0,0)$ by "·" (these are actually place holders: they have indices but no entries). Special (logical) computational rules apply for such entries and are introduced in the text where this type of notation is first used. This extension of matrix algebra appears to be very useful: in many matrix operations (in particular, reductions and approximations), one cannot say beforehand whether certain entries survive the operation (e.g., deleting rows or columns of zeros in a matrix). Just as the introduction of the empty set $\emptyset$ and the number 0 prove extremely useful in set theory and algebra, so do indexed empty entries. They also correspond to the empty symbol ($\perp$) often used in computer science.

- In the literature, many notations are used to indicate the transpose of a real matrix or the Hermitian transpose (conjugate transpose) of a complex matrix. In this book and as in MATLAB, we use only real or complex arithmetic and indicate these transpositions by a single symbol, namely a single accent (i.e., A' is in all cases the Hermitian transpose of A; in the real case, it is then also just the transpose). A motivation for this choice is (i) the overload of the symbols T or H, as we have special use for those (we do not like the notation T^T for the transpose of T nor H^H for the Hermitian transpose of the Hankel matrix H!) and (ii) consistency with MATLAB (with the understanding that the accent is "transpose conjugate" in the complex case). We reserve the star (e.g., with a an operator, a^*) for the dual of an object in a context where duality is defined (a dual is not necessarily a transpose, although it often will be). We use tildes and hats as normal typographical symbols – they do not have any other meaning than to characterize the object under discussion. However, there is one exception. In estimation theory, we often use the upper bar as shorthand for expectation: $\overline{X} = \mathbf{E}X$, and the hat as indicating an estimate: $\widehat{X}$ is an estimate of X.
- We use *constructors* systematically, following a good habit of computer science. Constructors are written in normal font. For example, "col" is the column constructor. It takes a sequence of elements (e.g., numbers or blocks with appropriate dimensions) and makes a column out of them. For example, $\mathrm{col}(u_1,u_2) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ (compare this with the awkward $\begin{bmatrix} u_1^T & u_2^T \end{bmatrix}^T$). Further constructors are "row," "Toeplitz," "Hankel" and so on (often abbreviated).
- Names and formulas: The general convention is that names are written in a regular font. For example, "ker" or "ran" are the names of the functions that return the kernel or the range of an operator, respectively, while *ker* is the product of k, e and r. This convention allows us to use indexed names as well: so K7 is just a name, while the notation K_7 assumes K to be a vector whose element with index 7 is K_7. A new name occurring in the text, a theorem or a proof is introduced with the symbol ":="; for example, $T := D + C(I - ZA)^{-1}ZB$ defines T in terms of the right-hand side symbols (assumed to be already defined).
- Another convenient notation that we shall use extensively is a shorthand for system realizations. Z is systematically used as the "forward" or "causal" shift, with its conjugate Z' the "backward" or "anti-causal" shift. A causal, LTV system T may

have a "realization" $T = D + C(I - ZA)^{-1}ZB$ for which we use the shorthand

$$T \sim_c \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{1}$$

to mean "is represented by the causal realization." Similarly, an anti-causal LTV system may have a realization $T = D + C(I - Z'A)^{-1}Z'B$, with the shorthand

$$T \sim_a \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{2}$$

to mean "is represented by the anti-causal realization." Z is a generic operator or constructor. It just shifts the indexing scheme but stands by the same token for a generic collection of shift matrices, which have a different effect whether applied to the left or the right of an indexed object. For example, suppose u is an indexed column vector, then Zu is again an indexed column vector whose elements are $(Zu)_k = u_{k-1}$. In our formalism, the dimensions of the u_k may vary (and even become empty), and the dimensions of the entries in Z, interpreted as a matrix, have to adapt. This question is addressed at length in the chapter on LTV systems. Some constructors can be represented as matrices. What characterizes a constructor is that it entails an organizational operation rather than a numerical operation.

- We also need shifts along diagonals. We denote $A^{\langle +1 \rangle}$ as a "forward" diagonal shift on a matrix, that is, a shift in the southeast direction, with as its conjugate the "backward" diagonal shift denoted as $A^{\langle -1 \rangle}$, that is, a shift in the northwest direction.
- We shall also occasionally use "continuous products," defined for integers $i > k$ as $A^{>}_{i,k} = A_{i-1}A_{i-2}\cdots A_{k+1}$ with $A^{>}_{i,i} := I$ (NB: this convention may differ from what is done in the literature). Also, $A^{\geq}_{i,k} := A_i A_{i-1} \cdots A_k$ will be used when convenient. To avoid confusion with the use of upper indices for columns and lower ones for rows, we often use the abbreviated notation $A^{i>k} := A^{>}_{i:k}$ and $A^{i \geq k} := A^{\geq}_{i:k}$ as well, in which the upper index now refers to a product instead of a column.

A Note on the Cover Picture

In 1903, the mathematician Max Dehn investigated for the first time "a simple and yet quite general problem of geometry" [78]: a square is to be divided exactly into partial squares of different sizes. This decomposition appeared to be highly nontrivial. A square together with such a division is called a *perfect* square.

The search for solutions went on in vain for more than three decades. The problem attracted an unusually high amount of interest – probably because of its simple and generally understandable nature as a puzzle and because it can be brought under the rubric of "mathematical recreations" for the edification of a broad public. But mathematicians especially were interested because of the "hard" topological and combinatorial aspects.

In 1940, four English students, Brooks, Smith, Stone and Tutte, proposed a completely new method of solving the problem. The student quartet approached the problem based on a physical analogy of the problem to Kirchhoff nets or electrical systems. Based on this system model, they found perfect squares of order $n = 26$ and $n = 28$.

In the following decades, computer programs were used to create catalogs of perfect decompositions, but all attempts to find perfect squares of order $n < 25$ failed. In the early 1960s [79], at the Philips Computing Center in Eindhoven, researchers proved, by full enumeration, that no perfect squares exist up to order $n = 20$. The picture on the book cover shows the perfect square of minimal order $n = 21$. It is uniquely determined and was found on March 22, 1978, by A. J. W. Duijvestijn at the University of Twente, with the help of a DEC-10 computer. The solution was published in the *Journal of Combinatorial Theory* [80] and has since been shown as a signet on the cover of both series of the journal.

This example shows that system models and the associated views and intuitions can enhance the understanding of the structure of difficult mathematical problems. The example shown should make it clear that system models can do much more than simply illustrate mathematical facts in a circuit diagram familiar to the engineer.

In 2008, Rainer Pauli picked up the story of the perfect square of order $n = 21$ and produced an artistically appealing rendition of the solution – the *Perfect Square*. This image is shown on the cover of the book and is based on an inner connection between the color shade of the squares and the geometry. The color representation is based on a mixing ratio of red and yellow corresponding to the size of the squares.

Acknowledgments

This book would not have been possible without the collaboration with and influence of many colleagues, students, assistants and relatives. A full list would be prohibitively long, and the short list that we are including here may (or even will) overlook important contributors, a fate that appears unavoidable. Nonetheless, here are persons to whom we are grateful for either their contributions to our knowledge and insights, for the support they provided or, in many cases, both.

In the first place, we would like to mention the massive seminal contributions of the giants of our field who have passed away in recent times: Rudy Kalman, Jan Willems, Alfred Fettweis and Uwe Helmke.

From a methodological point of view, we have been most strongly influenced by Tom Kailath and quite a few of his students. Besides Tom, who pioneered the algebraic and computational approach to system theory, we have been principally influenced by his students Martin Morf, Sun-Yuan Kung, George Verghese, Hanoch Lev-Ari, Ali Sayed and Augusto Vieira. Our contacts with the Information Systems Laboratory group in Stanford have always been exceedingly productive, and we cannot be sufficiently grateful to Tom for mostly making them happen. We were graciously invited to heaven so many times.

Many colleagues contributed ideas and specificities to our work, besides those already mentioned. In particular, we wish to mention Daniel Alpay, Brian Anderson, Thanos Antoulas, Jo Ball, John Baras, Vitold Belevitch, Bob Brayton, Adhemar Bultheel, Francky Catthoor, Samarjit Chakraborty, Shiv Chandrasekaran, Prabhakar Chitrapu, Leon Chua, Ed Deprettere, Harry Dym, Yuli Eidelman, Paul Fuhrmann, Tryphon Georgiou, Keith Glover, Israel Gohberg, Nithin Govindarajan, Bill Helton, Jochen Jess, Rien Kaashoek, Geert Leus, Wolfgang Mathis, Mohammed Najim, Bob Newcomb, Ralph Otten, Rainer Pauli, Ralph Phillips, Justin Rice, Jacqueline Scherpen, Wil Schilders, Malcolm Smith, Lang Tong, Eugene Tyrtyshnikov, Joos Vandewalle, Marc Van Barel, Paul Van Dooren, Michel Verhaegen, Eric Verriest, Jan Zarzycki and many of their students.

Klaus Diepold likes to acknowledge, in addition, the inspiring contributions from colleagues who have accompanied him through many years with collaboration, inspiration, discussion and friendship, notably Rainer Pauli, Wolfgang Mathis, Albrecht Reibiger, Harald Martens and Sunil Tatavarti.

Patrick Dewilde wishes to acknowledge the intense support and patience of his spouse, Anne Renaer, which made the many additional hours of research and writing possible, in addition to being a permanent and loving muse. He also wishes to extend his gratitude first to Sarah Kailath, who unfortunately passed away after an unforgiving illness, and later to Anu Luther, for so often and so graciously being our hosts and allowing many years of collaboration with Tom Kailath to take place in the ideal environment of their home.