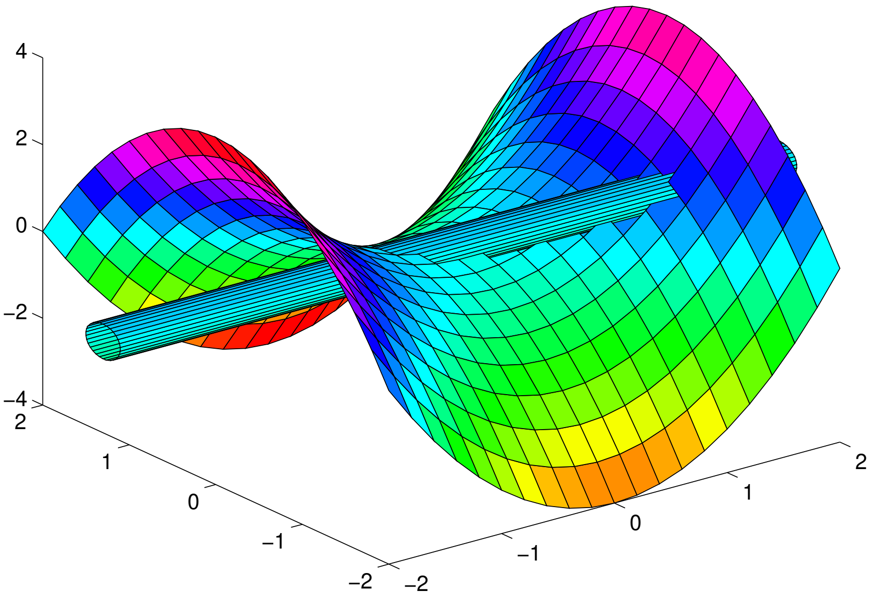
Matrix Computations: Factorizing in Parallel and Surfing the Kronecker Structure Hierarchies

Erik Elmroth



Ph.D. Thesis, 1995
Department of Computing Science
Umeå University
Sweden

Matrix Computations: Factorizing in Parallel and Surfing the Kronecker Structure Hierarchies

Erik Elmroth

Ph.D. Thesis, 1995
UMINF-95.15

Department of Computing Science
Umeå University
S-901 87 Umeå, Sweden

Akademisk avhandling som med tillstånd av rektorsämbetet vid Umeå Universitet framläggs till offentlig granskning fredagen den 9 juni klockan 13.15 i hörsal C, Samhällsvetarhuset, för avläggande av filosofie doktorsexamen.

UmU Tryckeri
UMINF-95.15 ISSN-0348-0542 ISBN 91-7191-065-4

Abstract

This dissertation concerns design, modeling, and evaluation of portable parallel block matrix factorizations and certain numerical computations related to the Kronecker canonical form (KCF) of matrix pencils.

Efficient parallel block algorithms for the *LU*, *Cholesky*, and *QR* factorizations, transportable over a range of shared memory multiprocessor architectures are presented. The algorithms are evaluated from the point of view of explicit versus implicit parallelization and dynamic versus static load balancing. Limitations from software overhead and sequential bottlenecks are identified and quantified for several different implementations on the IBM 3090VF/600J.

We present a block column wrap-mapping approach for design of parallel block matrix factorization algorithms that are portable over and between shared memory multiprocessors and distributed memory multicomputers. The optimal block size and the parallel execution time are predicted by a performance model.

By viewing an $m \times n$ matrix pencil $A - \lambda B$ as a point in $2mn$ space, certain numerical computations relating to the KCF can be viewed as moving matrix pencils from manifold to manifold in the $2mn$ space. We make use of the geometry of this space, to improve our knowledge of numerical algorithms and their failures. With this knowledge our *goal* is to develop more robust and accurate algorithms for computing the generalized Schur form of a general matrix pencil and increase their functionality for solving important "nearness" problems.

A comprehensive study of the set of 2-by-3 pencils improves our understanding of the "nearness" problems. The closure hierarchy of the orbits (manifolds of strictly equivalent pencils) of all different KCFs is derived and presented in a closure graph that show how the structures relate to each other in the 12-dimensional space of 2-by-3 pencils. Necessary conditions on perturbations for transiting from the orbit of one KCF to another in the closure hierarchy are derived. We show normwise bounds for the smallest perturbations of a generic 2-by-3 pencil for finding a specific non-generic KCF. We study the behaviour of non-generic structures under random perturbations in finite precision arithmetic.

We derive versal deformations of the KCF by deriving the tangent space and orthogonal bases for the normal space to the orbits of strictly equivalent matrix pencils. The deformations reveal the local perturbation theory of matrix pencils related to the KCF. We also obtain a new singular value bound for the distance to the orbits of less generic pencils. Experiments illustrate how versal deformations can be used to increase the understanding our "nearness" problems.

We give new interpretations of important results by Pokrzywa, for determining closure relations among orbits of KCFs, partly by generalizing classical theorems by Gantmacher. The results are used to derive an algorithm for computation of the stratification of the KCFs (i.e., the closure hierarchy).

Preface

The thesis consists of the following five papers and an introduction including a summary of the papers.

I. Krister Dackland, Erik Elmroth, Bo Kågström, and Charles Van Loan. Parallel Block Matrix Factorizations on the Shared Memory Multiprocessor IBM 3090 VF/600J*. *International Journal of Supercomputer Applications*, Vol. 6:1, 1992.

II. Krister Dackland, Erik Elmroth, and Bo Kågström. A Ring-Oriented Approach for Block Matrix Factorizations on Shared and Distributed Memory Architectures†. In R. F. Sinovec et. al. *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, Vol 1, SIAM, Philadelphia, 1993.

III. Erik Elmroth and Bo Kågström. The Set of 2-by-3 Matrix Pencils — Kronecker Structures and their Transitions under Perturbations‡. To appear in *SIAM Journal on Matrix Analysis and its Applications.*

IV. Alan Edelman, Erik Elmroth, and Bo Kågström. A Geometric Approach to Perturbation Theory of Matrices and Matrix Pencils. Part I: Versal Deformations. Report UMINF-95.09, Department of Computing Science, Umeå University, 1995. Submitted to *SIAM Journal on Matrix Analysis and its Applications.*

V. Erik Elmroth. On the Stratification of the Kronecker Canonical Form. Report UMINF-95.14, Department of Computing Science, Umeå University, 1995.

Design, modeling, and evaluation of parallel matrix factorization for shared and distributed memory architectures are the subjects of papers I and II. Chapter 1 gives an introduction to these problems together with a summary of the two papers.

Papers III–V focus on numerical computations related to the Kronecker canonical form. Chapters 2 and 3 give an introduction to the standard and generalized eigenvalue problems, respectively, and Chapter 3 ends the introduction by a summary of papers III–V.

Acknowledgements

First of all, I would like to thank my thesis advisor, professor Bo Kågström, for sharing his great expertise and always making so much of his time available, for his continuing inspiration and encouragement, and for always providing the best of conditions for writing this thesis. Bo is also coauthor of four of the papers.

I would also like to express my gratitude to my other three coauthors. To Krister Dackland for being a good friend while we were doing a lot of joint work resulting in papers I and II. To Charles Van Loan for joint work on Paper I. To Alan Edelman for bringing up many bright ideas that finally lead to our joint Paper IV and for providing the nice picture of the manifold of singular matrices used as cover page of this thesis, and originally presented in Paper IV.

Thanks also to the colleagues at the Department of Computing Science at Umeå University for providing a nice atmosphere and good working conditions.

Special thanks to the Swedish cross country skiing community, for being too good competitors, thereby making it an easy choice for me to start working on this thesis.

I would like to express additional gratitudes to Boel for always supporting me, and for showing great interest, even when discussing matrix pencils.

Finally, I would like to thank my parents and my sister for giving me the best of backgrounds and thereby providing me with the attitude required for writing this thesis.

Financial support has been provided by the Swedish National Board of Industrial and Technical Development and the Faculty of Mathematics and Natural Sciences, Umeå University.

Umeå, May 3, 1995

Erik Elmroth

Contents

Chapter 1

Parallel Matrix Factorizations

With the introduction of advanced parallel computer architectures a demand for efficient parallel algorithms has emerged. During the last years we have seen many research activities concerning algorithm design for different vector and parallel architectures. Today it is well-known that block algorithms are required to exploit the full potential of hierarchical memory computers, multiprocessors, and multicomputers in matrix computations.

Different matrix factorizations, such as LU, *Choelsky*, and QR factorization, are basic and important tools in most scientific, economic, and engineering computational problems. Our goal is to design parallel block algorithms with high parallel efficiency for these factorizations. The implementations should be transportable over a wide range of parallel MIMD architectures, including both shared and distributed memory systems.

To attain this goal we have to consider as many as possible of the parallel processing key factors: balancing the load over the processors, maintaining the locality of data by suitable choice of granularity, minimizing memory contention between processors, and minimizing the number of synchronization points in a parallel algorithm.

Block matrix factorizations provide a good platform for the development of general design principles for parallel algorithms in the field of matrix computations.

1.1 Block Algorithms

By a block algorithm we mean one that is rich in basic matrix-matrix operations such as matrix-matrix multiply, rank-k updates, and solving triangular systems with multiple right-hand sides. In the blocked algorithms the entities on which we perform the operations are submatrices (or blocks). Let an $m \times n$ matrix A be partitioned in p row blocks and q column blocks such that

$$\begin{bmatrix} A_{11} & \cdots & A_{1q} \\ \vdots & & \vdots \\ A_{p1} & \cdots & A_{pq} \end{bmatrix}$$

where block A_{ij} has dimension $m_i \times n_j$ and we say that $A = (A_{ij})$ is a $p \times q$ block matrix. Typically the block size is fixed so that $m_i = n_j = nb$. If A is defined as above, B is similarly defined as a q-by-t block matrix and C as a p-by-t block matrix, then the matrix multiplication $C = AB$ can be expressed as

$$C_{ij} = \sum_{k=1}^{q} A_{ik} B_{kj} \quad i = 1, \ldots, p \text{ and } \quad j = 1, \ldots, t.$$

The methodology used to develop block algorithms for more sophisticated matrix computational problems is to restructure well-known and stable elementwise algorithms to perform block matrix-matrix operations in their inner loops.

It is well-known that block algorithms are required to exploit the full potential of hierarchical memory systems. Typically, the memory hierarchies consists of registers (which may include vector registers) and cache memory at the top level. On lower levels there are a main memory, an optional expanded storage (slower but not as costly as main memory), disc and tape storage. In parallel shared and distributed memory systems, some of these memories are either local for each processor, shared among the processors or distributed between the processors.

The purpose of a memory hierarchy is to have a memory that appears to be very large and fast, but is reasonably cheap to manufacture. If the memory reference pattern in a program is kept local in the memory, then the memory hierarchy can be used efficiently. A drawback is that if distant data are referenced often, access times to memory may be very long.

The crucial fact is that excessive movement of data to and from main memory decreases the performance. Since matrix-matrix operations typically involves $\mathcal{O}(n^3)$ operations on $\mathcal{O}(n^2)$ data, which essentially gives the fraction of arithmetic operations to data movements as $\mathcal{O}(n)$, block algorithms make it possible to reuse data in cache memory and in vector registers.

High performance in the matrix-matrix operations is achieved by structuring the code to reduce memory traffic in the hierarchy. In order to minimize the memory traffic the data reuse is maximized at different levels in the hierarchy. This is typically done by blocking the matrix-matrix algorithms, where the size of each block is chosen in order to fit into the memory at a certain level in the hierarchy. The order of the block operations is determined so as to keep the most heavily referenced blocks at a high level in the hierarchy as long as possible.

1.2 High Performance Software Libraries

Linear algebra software libraries such as LINPACK [4] and EISPACK [37, 21] have been widely used for many years, but on high performance computers they often achieve only a small fraction of the peak performance of the machines. This mainly because of lack of ability to efficiently use the memory hierarchies and because of very limited potential parallelism. Therefore, LAPACK [1] has been designed to supersede LINPACK and EISPACK, principally by achieving much greater efficiency on high performance computers. LAPACK was originally targeted to achieve good performance on single processor vector machines and shared memory multiprocessors. This is partly carried out by performing most operations in Level 3 Basic Linear Algebra Subprograms (BLAS) [17].

The Level 3 BLAS [17] is a small set of computational kernels that was proposed as a standard for basic matrix-matrix operations, with the purpose to efficiently exploit memory hierarchy computers. Computer manufacturers are expected to provide highly optimized Level 3 kernels that can be called from user programs to efficiently utilize the architecture and hardware organization of the machines.

However, the performance of optimized Level 3 kernels depends on the size and the shape of the matrix blocks participating in the operations [27, 35]. Typically they do not reach their optimal performance if the total size of the blocks is too small since this lead to a small number of floating point operations, or if one or more of the dimensions are very small since this may lead to poor data reuse.

1.3 Parallel Block Algorithms

To achieve high performance on shared memory multiprocessors (SMM) and distributed memory multicomputers (DMM) one of the major key-factors is to reduce the communication-to-computation ratio, i.e., to keep high data locality. This means for a DMM that we strive for algorithms that let each processor perform as much computations as possible on data located in its own local memory, to reduce the need for message passing between processors. On a SMM we want different processors to work as much as possible on different data, which, e.g. reduces the amount of memory traffic for keeping cache memories coherent.

For matrix computations, block algorithms have shown to be the correct level of abstraction for algorithm design, also for parallel computers. Just as block algorithms enable data reuse in complex memory hierarches, they provide good opportunities to achieve high data locality on parallel computers. When we, on a single processor, have memory traffic up and down the local memory hierarchy, we also have interprocessor communication in a parallel system. Therefore, in order to minimize communication overhead due to data transfers, the ratio of

$\mathcal{O}(n^3)$ operations to $\mathcal{O}(n^2)$ data provided by matrix-matrix operations, is not only desirable for systems with complex memory hierarchies but also for parallel systems.

1.4 Contributions in This Thesis

Our *objective* is to develop general principles for design, implementation, modeling, and evaluation of parallel algorithms for matrix computations in general, and matrix factorizations in particular. We aim at developing general techniques for construction of highly efficient portable block algorithms and to develop performance models as well as predicting optimal block sizes and performance.

Most of these results are also presented as parts of [7].

Paper I

Efficient parallel block algorithms for the LU, $Cholesky$, and QR factorizations, transportable over a range of Shared Memory Multiprocessor (SMM) architectures are presented. Explicitly parallel implementations of different block algorithms that utilize optimized uniprocessor level-3 BLAS are compared with corresponding implicitly parallelized routines of LAPACK.

Parallelism is mainly invoked implicitly in LAPACK by replacing calls to uniprocessor level-3 kernels by calls to parallel level-3 kernels. The main advantage with this approach is that the portability of LAPACK is maintained. However, the implicit parallelism imposes a synchronization after each level-3 operation, which means that idle processors must wait until all processors have completed their share of work. This can be overcome by invoking the parallelism explicitly at the block level of the algorithm. Then processors can work on independent block operations which makes it possible to overlap and pipeline different matrix-matrix operations. Further, only efficient uniprocessor level-3 kernels are required. Here, the parallelism is invoked by using parallel language constructs.

Load balancing of the explicitly parallel block algorithms is by static or dynamic scheduling of the work. The static scheduling utilizes cost functions based on the number of floating point operations (flops) expressed as GEMM equivalents. Since the different BLAS operations can be performed in different speed, it is not enough to partition the work so that each processor is given the same amount of work in terms of flops. Therefore we multiply the number of flops in each BLAS operation by a factor in order to give an approximation of the number of flops that can be computed by GEMM (GEneral Matrix Multiply and add) in the same time, i.e., the GEMM equivalent flops. In one iteration of a block algorithm, a cost funtion computes the total number of GEMM equivalent flops, which is to be distributed evenly on the processors. In the dynamic scheduling approach, any processor that is out of work is assigned new work from a queue of tasks. As soon as all data that is required for the computation of a new task is completed, that task is put in the queue. In the implicitly parallel approach, the load balancing is inherited from the parallel level-3 kernels.

The implementations are done in IBM Parallel FORTRAN and performance results for an IBM 3090VF/600J are presented. Theoretical models give upper bounds on the best possible speedup. For the implicitly parallel algorithms these bounds are based on Amdahl's law, i.e., they are derived from the performance limitations given by the sequential bottlenecks of the algorithms. For the explicitly parallel algorithms, the bounds are derived from a model of the parallelization software overhead, including the time to create new jobs and to terminate jobs.

Parts of this work is also presented in [5, 9].

Paper II

A block (column) wrap-mapping approach for design of parallel block matrix factorization algorithms that are (trans)portable over and between shared memory multiprocessors (SMM) and distributed memory multicomputers (DMM) is presented.

By reorganizing the matrix on the SMM architecture, the same ring-oriented algorithms can be used on both SMM and DMM systems with all machine dependencies comprised to a small set of communication routines. On a SMM system, the (simulated) communication is implemented as a change of references for pointers to blocks in the reorganized matrix. On a DMM system, the communication is implemented by using traditional message passing routines. The reorganization of the matrix ensures that each processor's references to the matrix are made similar on both SMM and DMM, which is a requirement when we want to use the same generic algorithm on both SMM and DMM systems.

The algorithms are described on high level with focus on performance and portability aspects. Implementation aspects of the *LU*, *Cholesky*, and *QR* factorizations and machine specific communication routines for some SMM and DMM systems are discussed. Timing results show that our portable algorithms have similar performance as machine specific implementations.

For DMM platforms this similarity is due to the DMM programming model used. For the SMM environments the good performance of the ring-oriented algorithms are explained by the reorganization of the matrix, giving each processor block columns that are stored consecutively, and the proportionally low synchronization costs. The consecutive storage enables updating of several block columns in one operation, instead of one operation per block column. This implies less amount of software overhead and larger matrices in the level-3 BLAS operations.

A performance model that identifies different components of the total execution time is presented in terms of *arithmetic time*, *communication time*, and *waiting time*. The modeled execution time does not only predict the execution time, but more important for our purposes, it also predicts the best block size to be the same as the best one in real execution, except occasionally, when it predicts a nearby block size which in real execution shows a performance close to the optimal one.

More detailed presentations of the results in Paper II may be found in [6, 8].

Chapter 2

Eigenvalue Problems

From the standpoint of classical algebra, the algebraic eigenvalue problem has been completely solved. The problem is subject of classical similarity theory, and fundamental results are embodied in the Jordan canonical form. However, from the standpoint of numerical analysis, the computation of the Jordan canonical form is associated with ill-posed computational problems [10, 19, 23, 24, 29, 28, 32]. Before we turn into the numerical issues, we review some basic results of the standard eigenvalue problem [22, 44].

2.1 Eigenvalues and Eigenvectors

The *eigenvalues* of an n-by-n matrix A are the n roots of the *characteristic equation* $p(z) \equiv \det(A - zI) = 0$. The set of roots is called the *spectrum* and is denoted $\lambda(A)$.

If $\lambda \in \lambda(A)$ then a non-zero n-by-1 vector x that satisfies

$$Ax = \lambda x$$

is called a right *eigenvector* (or simply an eigenvector) corresponding to the eigenvalue λ. A vector y that satisfies $y^H A = \lambda y^H$ is referred to as a left eigenvector.

Since the multiplication Ax of an eigenvector x does not alter the direction of x, the vector x is said to be invariant with respect to multiplication by A. The subspace $\mathcal{S}$ spanned by all invariant vectors is called the *invariant subspace* of A.

If A has distinct eigenvalues λ_i with associated eigenvectors x_i, then we write in matrix notation,

$$AX = JX,$$

where $J = \mathrm{diag}(\lambda_1, \lambda_2, \ldots \lambda_n)$ and $X = [x_1, x_2, \ldots, x_n]$. For distinct eigenvalues, the set of eigenvectors is always linearly independent, that is, the set of eigenvalues is *non-defective.* If λ_i is a multiple eigenvalue, i.e., $\lambda_i = \lambda_j$, where $i \neq j$, then the number of identical eigenvalues is said to be the *algebraic multiplicity* of λ_i. The number of linearly independent eigenvectors corresponding to λ_i is said to be the *geometric multiplicity* of λ_i. An eigenvalue λ_i is said to be *defective* if its algebraic multiplicity exceeds its geometric multiplicity, i.e., if there does not exist one linearly independent eigenvector for each eigenvalue $\lambda_j = \lambda_i$. A matrix with a defective eigenvalue is said to be defective.

If A is non-defective, then the eigenvector matrix X is invertible and we write

$$X^{-1}AX = J,$$

where the eigenvalue preserving transformation $X^{-1}AX$ is said to be a *similarity transformation* and A is said to be *diagonalizable.*

For the diagonalization we require X to be the set of eigenvectors, but for a similarity transformation in general we do only require a non-singular transformation matrix. The matrices A and $B = P^{-1}AP$ (for any non-singular P) are said to be *similar* and the set of all matrices similar to A defines an *orbit*

$$\mathrm{orbit}(A) = \{P^{-1}AP : det(P) \neq 0\}.$$

2.2 The Jordan Normal Form

Let, for example,

$$A = \begin{bmatrix} a & b \\ 0 & a \end{bmatrix},$$

then $\lambda = a$ is a multiple eigenvalue and $x = [1,\ 0]^T$ is the only (linearly independent) eigenvector. Since there is only one eigenvector there exists no non-singular matrix X that diagonalizes A. However, computation of a vector x_2 such that $(A - aI)x_2 = x_1$, gives the vector $x_2 = [0,\ 1/b]^T$ and a full rank matrix X such that

$$X^{-1}AX = \begin{bmatrix} 1 & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} a & b \\ 0 & a \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1/b \end{bmatrix} = \begin{bmatrix} a & 1 \\ 0 & a \end{bmatrix}.$$

The vector x_2 is said to be a *principal vector* (of grade 2) and satisfies $(A - aI)^2 x_2 = 0$. Eigenvectors are principal vectors of grade 1.

In the general case, a matrix A with p distinct eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$, of algebraic multiplicities $t_1, t_2, \ldots t_p$, has at least one eigenvector corresponding to each distinct eigenvalue. Let s_k be the geometric multiplicity of the eigenvalue λ_k, i.e., there exists s_k linearly independent eigenvectors corresponding to λ_k. If $s_k < t_k$, then there is not enough eigenvectors to span the complete space, and the eigenvalue is defective. Then we have to add $t_k - s_k$ principal vectors forming s_k chains

$$(A - \lambda_k I)x_k^{(l)} = x_k^{(l-1)}, \quad \text{for} \quad l = 1, 2, \ldots, h_k, \tag{2.2.1}$$

where h_k is the height of the chain and l is the grade of the principal vector. In general, a principal vector of grade l satisfies $(A-\lambda_k I)^l x_k^{(l)} = 0$. For each chain, the vector $x_k^{(0)}$ is the zero vector and $x_k^{(1)}$ as an eigenvector, giving one chain per linearly independent eigenvector. Of course, non-defective eigenvalues will only give chains of height 1, i.e., the eigenvector is the only vector in the chain (except for the zero vector $x_k^{(0)}$).

In matrix notation the chain relation (2.2.1) is written

$$AX_k = X_k J_{h_k}(\lambda_k),$$

where $X_k = [x_k^{(1)}, x_k^{(2)}, \ldots, x_k^{(h_k)}]$ and $J_{h_k}(\lambda_k)$ is a h_k-by-h_k Jordan block

$$J_{h_k}(\lambda_k) = \begin{bmatrix} \lambda_k & 1 & & \cdots & 0 \\ 0 & \lambda_k & 1 & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \cdots & & 0 & \lambda_k \end{bmatrix}.$$

By forming $X = [X_1, X_2, \ldots, X_q]$, where the columns of each X_i are given by the chain (2.2.1) corresponding to one of the q Jordan blocks, we have the *Jordan normal form* (JNF),

$$X^{-1}AX = \operatorname{diag}(J(\lambda_1), J(\lambda_2), \ldots, J(\lambda_p)),$$

where

$$J(\lambda_k) = \operatorname{diag}(J_{h_1}(\lambda_k), J_{h_2}(\lambda_k), \ldots, J_{h_{s_k}}(\lambda_k)).$$

In the Jordan normal form, the number and the sizes of the Jordan blocks associated with each distinct eigenvalue are unique, although their ordering is not. Since similarity transformations do not change the number or sizes of the blocks, orbit(A) consists of all matrices with the same Jordan normal form (or Jordan structure) as A.

A matrix A is said to be *derogatory* if there in its Jordan structure are more than one Jordan block associated with one eigenvalue, i.e., A is derogatory if the geometric multiplicity $s_k > 1$ for any eigenvalue λ_k. This imply that any diagonalizable matrix with a multiple eigenvalue is derogatory, which for example is the case for the identity matrix.

The *Weyr* characteristics corresponding to an eigenvalue λ_k is the sequence $\alpha_1, \alpha_2, \ldots$, where α_i is the number of principal vectors of grade i corresponding to λ_k [18].

The sequence of numbers given by listing the sizes of the Jordan blocks corresponding to an eigenvalue λ_k in decreasing order is known as the *Segre* characteristics corresponding to the eigenvalue λ_k.

2.3 Ill-conditioning and Rank Decisions

Almost all n-by-n matrices are diagonalizable. We say that a *generic* matrix have a full set of linearly independent eigenvectors and only 1-by-1-blocks in its Jordan canonical form. Only if the matrix lies in a manifold that exactly satisfies some non-linear restrictions, can it have a non-trivial Jordan structure. Mathematically these manifolds correspond to singularities, and numerically the diagonalization process is ill-conditioned near these manifolds. The manifolds of non-diagonal Jordan forms are mathematically a set of measure zero.

When determining the rank of a matrix using finite precision arithmetic, we want to make "safe" rank-decisions and therefore compensate for rounding errors by using a tolerance below which singular values are considered being zero. Actually, by considering all such small singular values as zero, we have thickened the manifolds so that they are no longer a set of measure zero.

In practice, the rank decisions are made even more robust. When one or more of the singular values are interpreted as zeros, we insist on a certain gap between the smallest non-zero singular value σ_k and the singular value σ_{k+1} which is the largest one that is interpreted as zero. If the gap is too small, also σ_k is interpreted as zero and the process is repeated by looking for an appreciable gap between σ_{k-1} and σ_k, and so on.

2.4 Staircase Algorithms for Matrices

The computation of the Jordan structure is typically done by so called staircase-type algorithms, using unitary similarity transformations [36, 29, 28]. In the following we outline the main ideas of this type of algorithms without getting into too much technicalities. The main step of the procedure is to compute the nullspace of $(A - \lambda I)^j$ for $j = 1, 2, \ldots$, for each $\lambda \in \lambda(A)$. This can be accomplished by a staircase algorithm, using the singular value decomposition for successive and robust rank decisions. Small singular values are interpreted as zero if they are smaller than a given tolerance, provided that the gap condition is satisfied, as explained in Section 2.3.

We start by discussing the case when A only has one eigenvalue μ of multiplicity n, giving that the matrix $A - \mu I$ has the only eigenvalue 0. Since we now want to work on the shifted pencil, we assume that $B = A - \mu I$ is a matrix with 0 as an eigenvalue of multiplicity n.

First, we compute the singular value decomposition of $B = B^{(1)}$ such that

$$B^{(1)} = U_1 \Sigma_1 V_1^H,$$

and the singular values in Σ_1 are ordered in increasing order of magnitude. Then, the matrix

$$B^{(2)} = V_1^H U_1 \Sigma_1 = V_1^H B^{(1)} V_1 = \begin{bmatrix} B_{11}^{(2)} & B_{12}^{(2)} \\ B_{21}^{(2)} & B_{22}^{(2)} \end{bmatrix},$$

has the vector norms of the columns equal to the singular values of $B^{(1)}$. If the first m_1 singular values in Σ_1 are smaller than the required tolerance and the gap condition is satisfied, the corresponding first columns in $B^{(2)}$ are replaced by zeros. This means that we have decided that the first m_1 columns $V_1^{(1)}$ of

$$V_1 = \left[\begin{array}{cc} V_1^{(1)} & V_1^{(2)} \end{array}\right],$$

are the eigenvectors of B. It follows that there are m_1 Jordan blocks in the computed Jordan structure of B.

The procedure is repeated on the remaining lower right $(n-m_1)\times(n-m_1)$ block of $B^{(2)}$, that is $B_{22}^{(2)}$, in order to determine the number of principal vectors of grade 2 (m_2). This tells us that the number of $J_k(0)$ blocks for $k \geq 2$ in B is m_2.

By repeatedly performing the same operations on the remaining lower right-most block of the transformed matrix and computing the principal vectors of grade $3, 4, \ldots$, we finally arrive at

$$\tilde{B} = W^H(B - E^{(h)})W,$$

where $\tilde{B}$ is nilpotent and in staircase form. For $m_1 = 3$, $m_2 = 2$, and $m_3 = 2$ $\tilde{B}$ has the form

$$\tilde{B} = \begin{array}{c} \begin{array}{ccc} \overbrace{\qquad\quad}^{m_1} & \overbrace{\qquad}^{m_2} & \overbrace{\qquad}^{m_3} \end{array} \\ \left[\begin{array}{ccc|cc|cc} 0 & 0 & 0 & x & x & x & x \\ & 0 & 0 & x & x & x & x \\ & & 0 & x & x & x & x \\ \hline & & & 0 & 0 & x & x \\ & & & & 0 & x & x \\ \hline & & & & & 0 & 0 \\ & & & & & & 0 \end{array}\right] \end{array}.$$

This decomposition tells us that there are m_1 eigenvectors, m_2 principal vectors of grade 2, and m_3 principal vectors of grade 3, which for $m_1 = 3$, $m_2 = 2$, and $m_3 = 2$ corresponds to the Jordan structure $J_1(0) \oplus J_3(0) \oplus J_3(0)$. It follows that the computed the Jordan structure for A is $J_1(\mu) \oplus J_3(\mu) \oplus J_3(\mu)$.

Chapter 3

Generalized Eigenvalue Problems

The *generalized eigenvalue problem* $Ax = \lambda Bx$, where A and B are m-by-n matrices, is a straightforward generalization of the standard eigenvalue problem, where $B = I$ and the matrices are square. The generalized eigenvalue problem is often written $(A - \lambda B)x = 0$, where $A - \lambda B$ is said to be a *matrix pencil* [20]. Although the generalized eigenvalue problem looks like a simple generalization of the standard eigenvalue problem, it exhibits some important differences. First, it is possible for $\det(A - \lambda B)$ to be identically zero, independent of λ. Second, if $A - \lambda B$ is regular and B is singular the generalized eigenvalue problem has infinite eigenvalues. It follows that the Kronecker canonical form, the generalization of the Jordan canonical form, is much more complex and that it also include some type of blocks other than Jordan blocks.

3.1 Matrix Pencils

For m-by-n matrices A and B, the pencil $A - \lambda B$ is said to be *regular* if and only if $m = n$ and $\det(A - \lambda B)$ is not identically zero. If $\det(A - \lambda B) \equiv 0$ for any λ or $m \neq n$, $A - \lambda B$ is said to be *singular*.

The *equivalence transformation*

$$P^{-1}(A - \lambda B)Q = P^{-1}AQ - \lambda P^{-1}BQ = \tilde{A} - \lambda\tilde{B},$$

does not affect the eigenvalues of the pencil. If $A - \lambda B$ is regular, the eigenvalues are found by transforming the pencil by equivalence transformations into an upper triangular pencil $S - \lambda T$. This transformation can be done with unitary matrices and results in the generalized Schur form of a regular $A - \lambda B$. Then the eigenvalues of $S - \lambda T$ are equal to the eigenvalues of $A - \lambda B$, and they are given by the pairs $(s_{ii}, t_{ii}) \neq (0,0)$. The finite eigenvalues are s_{ii}/t_{ii}, where $t_{ii} \neq 0$. The eigenvalues are infinite if $s_{ii} \neq 0$ and $t_{ii} = 0$. If $(s_{ii}, t_{ii}) = (0,0)$ for some i, then $A - \lambda B$ is singular, and the pair (s_{ii}, t_{ii}) does not correspond to an eigenvalue.

Since the eigenvalues of $A - \lambda B$ are the reciprocals of the eigenvalues of $B - \lambda A$, an infinite eigenvalue of $A - \lambda B$ corresponds to a zero eigenvalue of $B - \lambda A$. Therefore, the computation of an infinite eigenvalue is not more ill-conditioned than of a zero eigenvalue. Just as in the matrix case a regular pencil can be transformed into Weierstrass canonical form [43] to completely reveal the Jordan structure of finite as well as infinite eigenvalues.

However, the generalized eigenvalue problem becomes more complex when considering singular pencils. If $A - \lambda B$ is m-by-n and $m \neq n$ then almost all pencils have no eigenvalues at all. To be able to deal with singular pencils we have to introduce the *Kronecker canonical form* [31], which is a generalization of the Weierstrass canonical form [43] to singular pencils.

3.2 The Kronecker Canonical Form

Any m-by-n pencil $A - \lambda B$ can by equivalence transformations be transformed into the Kronecker canonical form

$$P^{-1}(A-\lambda B)Q = \operatorname{diag}(L_{\epsilon_1}, \ldots, L_{\epsilon_p}, J_{j_1}(\mu_1), \ldots, J_{j_k}(\mu_k), N_{i_1}, \ldots, N_{i_k}, L^T_{\eta_1}, \ldots, L^T_{\eta_q}),$$

where $J_j(\mu)$ corresponds to a j-by-j Jordan block for a zero or non-zero finite eigenvalue μ and N_j corresponds to a j-by-j Jordan block for an infinite eigenvalue:

$$J_j(\mu) \equiv \begin{bmatrix} \mu-\lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \mu-\lambda \end{bmatrix} \quad \text{and} \quad N_j \equiv \begin{bmatrix} 1 & -\lambda & & \\ & \ddots & \ddots & \\ & & \ddots & -\lambda \\ & & & 1 \end{bmatrix}.$$

The L_j and L_j^T blocks are *singular blocks of right* (column) and *left* (row) *indices of grade* j. These blocks are of size j-by-$(j+1)$ and $(j+1)$-by-j, respectively, and have the form

$$L_j \equiv \begin{bmatrix} -\lambda & 1 & & \\ & \ddots & \ddots & \\ & & -\lambda & 1 \end{bmatrix} \quad \text{and} \quad L_j^T \equiv \begin{bmatrix} -\lambda & & \\ 1 & \ddots & \\ & \ddots & -\lambda \\ & & 1 \end{bmatrix}.$$

The singular blocks have no eigenvalues and there exists a right singular (column) vector that for each λ zeroes out the L_j block identically:

$$\begin{bmatrix} -\lambda & 1 & & \\ & \ddots & \ddots & \\ & & -\lambda & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \lambda \\ \lambda^2 \\ \vdots \\ \lambda^j \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Similarly, there exists a left singular (row) vector that zeroes out the L_j^T block identically:

$$\begin{bmatrix} 1 & \lambda & \lambda^2 & \cdots & \lambda^j \end{bmatrix} \begin{bmatrix} -\lambda & & \\ 1 & \ddots & \\ & \ddots & -\lambda \\ & & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}.$$

If $A - \lambda B$ is regular, the L_j and L_j^T blocks are not present in the Kronecker canonical from.

Gantmacher [20] has shown how to identify the existence of L_k and L_k^T blocks in the KCF of a given pencil $A - \lambda B$. Let the Gantmacher matrix $R[A, B, i]$ of size $(i + 2)m \times (i + 1)n$ be defined by

$$R[A,B,i] = \begin{bmatrix} A & 0 & \cdots & 0 \\ B & A & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & B & A \\ 0 & \cdots & 0 & B \end{bmatrix}, \tag{3.2.1}$$

where A and B are $m \times n$ matrices. With this notation we can state the following theorem.

Theorem 3.1 *[20] The following statements are equivalent.*

- *$A - \lambda B$ is singular with a right (column) minimal index of lowest degree $k \geq 0$, i.e., $A - \lambda B$ has no right minimal indices of degree $< k$.*
- *$A - \lambda B$ is equivalent to the pencil*

$$\begin{bmatrix} L_k & 0 \\ 0 & A' - \lambda B' \end{bmatrix}, \tag{3.2.2}$$

where L_k is a $k \times (k+1)$ Kronecker block. $A' - \lambda B'$ may have right minimal indices of higher degree.

- $R[A, B, i]$ *has full column rank* $\mathrm{r}(R[A, B, i]) = (i+1)n$ *for* $i = 0, 1, \ldots, k-1$*, while* $\mathrm{r}(R[A, B, k]) < (k+1)n$*, or equivalently, the column nullity* $\mathrm{n}(R[A, B, i]) = 0$ *for* $i = 0, 1, \ldots, k-1$ *and* $\mathrm{n}(R[A, B, k]) > 0$.

A dual form of Theorem 3.1 can be stated for a left (row) minimal index of lowest degree $k \geq 0$. Then L_k^T takes the place of L_k and $L[A, B, i]$ of size $(i+1)m \times (i+2)n$ replaces $R[A, B, i]$, where

$$L[A, B, i] = \begin{bmatrix} A & B & 0 & \cdots & 0 \\ 0 & A & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A & B \end{bmatrix}, \tag{3.2.3}$$

and we are considering row ranks (or row nullities) of $L[A, B, i]$.

However, most applications do not require $A - \lambda B$ to be transformed into Kronecker canonical form. Most often it is enough to transfer $A - \lambda B$ to a *generalized Schur form* or similar [3, 15, 16, 26, 30, 39, 45], which reveals the complete Kronecker structure, e.g. to GUPTRI form [15, 16]

$$P^H(A - \lambda B)Q = \begin{bmatrix} A_r - \lambda B_r & * & * \\ 0 & A_{reg} - \lambda B_{reg} & * \\ 0 & 0 & A_l - \lambda B_l \end{bmatrix}, \tag{3.2.4}$$

where P (m-by-m) and Q (n-by-n) are unitary and $*$ denotes arbitrary conforming submatrices. Here the square upper triangular block $A_{reg} - \lambda B_{reg}$ is regular and has the same regular structure as $A - \lambda B$ (i.e., contains all generalized eigenvalues (finite and infinite) of $A - \lambda B$). The rectangular blocks $A_r - \lambda B_r$ and $A_l - \lambda B_l$ contain the singular structure (right and left minimal indices) of the pencil and are block upper triangular.

Given $A - \lambda B$ in GUPTRI form we also know different pairs of *reducing subspaces* [41, 15]. Suppose the eigenvalues on the diagonal of $A_{reg} - \lambda B_{reg}$ are ordered so that the first k, say, are in Λ_1 (a subset of the spectrum of $A_{reg} - \lambda B_{reg}$) and the remainder are outside Λ_1. Let $A_r - \lambda B_r$ be m_r-by-n_r. Then the left and right reducing subspaces associated with Λ_1 are spanned by the leading $m_r + k$ columns of P and the leading $n_r + k$ columns of Q, respectively. When Λ_1 is empty, the corresponding reducing subspaces are called *minimal*, and when Λ_1 contains the whole spectrum the reducing subspaces are called *maximal*.

3.3 Generic and Non-generic Structures

If $A - \lambda B$ is m-by-n, where $m \neq n$, then for almost all A and B it will have the same KCF, depending only on m and n (the *generic case*). The generic

Kronecker structure for $A - \lambda B$ with $d = n - m > 0$ is

$$\mathrm{diag}(L_\alpha, \ldots, L_\alpha, L_{\alpha+1}, \ldots, L_{\alpha+1}), \tag{3.3.1}$$

where $\alpha = \lfloor m/d \rfloor$, the total number of blocks is d, and the number of $L_{\alpha+1}$ blocks is m mod d (which is 0 when d divides m) [39, 11]. The same statement holds for $d = m - n > 0$ if we replace $L_\alpha, L_{\alpha+1}$ in (3.3.1) by $L_\alpha^T, L_{\alpha+1}^T$. Square pencils are generically regular, i.e., $\det(A - \lambda B) = 0$ if and only if λ is an eigenvalue. The generic singular pencils of size n-by-n have the Kronecker structures [42]:

$$\mathrm{diag}(L_j, L_{n-j-1}^T), \quad j = 0, \ldots, n-1. \tag{3.3.2}$$

In summary, generic rectangular pencils have only trivial reducing subspaces and no generalized eigenvalues at all. Generic square singular pencils have the same minimal and maximal reducing subspaces. Only if $A - \lambda B$ satisfies a special condition (lies in a particular manifold) does it have nontrivial reducing subspaces and generalized eigenvalues (the *non-generic case*). Moreover, only if it is perturbed so as to move continuously within that manifold do its reducing subspaces and generalized eigenvalues also move continuously and satisfy interesting error bounds [13, 15]. These requirements are natural in many control and systems theoretic problems such as computing controllable subspaces and uncontrollable modes [14, 40].

3.4 Staircase Algorithms for Matrix Pencils

Several authors have proposed (staircase-type) algorithms for computing a generalized Schur form (e.g. see [3, 12, 34, 30, 25, 26, 33, 39, 45]). They are numerically stable in the sense that they compute the exact Kronecker structure (generalized Schur form or something similar) of a nearby pencil $A' - \lambda B'$. Let $\|\cdot\|_E$ denote the Euclidean (Frobenius) matrix norm. Then $\delta \equiv \|(A - A', B - B')\|_E$ is an upper bound on the distance to the closest $(A + \delta A, B + \delta B)$ with the KCF of (A', B'). Recently, robust software with error bounds for computing the GUPTRI form of a singular $A - \lambda B$ has been published [15, 16]. Computational experiments that use this software will be presented in papers III – V.

The main steps of the `GUPTRI` algorithm [15, 16] are as follows. One phase of the algorithm extracts, for example, the Jordan structure of the zero eigenvalue and the right singular structure of $A - \lambda B$ by a finite sequence of range and nullspace separations, using unitary equivalence transformations. In step k $(= 0, 1, \ldots)$ of the first phase, the `GUPTRI` algorithm first determines $m_k =$ dimension of the column nullspace of $A^{(k)}$. From the singular value decomposition of $A^{(k)}$, the `GUPTRI` algorithm obtains Σ_A, i.e., a diagonal matrix with the singular values in increasing order on the diagonal, and V_A, the corresponding right singular vectors. The dimension of the column nullspace of $A^{(k)} = m_k$ is equal to the number of singular values interpreted as zero, where this decision is made by a similar process as described in Section 2.3. By computing

$[0\ A_2] = A^{(k)}V_A$ and $[B_1\ B_2] = B^{(k)}V_A$, where B_1 and the zero block to the left of A_2 both have m_k columns, the `GUPTRI` algorithm extracts the part B_1 of $B^{(k)}$ that may contain a common column nullspace with $A^{(k)}$. Indeed, the dimension of the common column nullspace of $A^{(k)}$ and $B^{(k)}$ is the same as the dimension of the column nullspace of B_1. By computing the singular value decomposition of B_1, the number of singular values interpreted as non-zero is found and denoted s_k. Then, the common column nullspace of $A^{(k)}$ and $B^{(k)}$ is $m_k - s_k$. This completes the process for the determination of the structure indices in step k.

Here, $A^{(0)} = A$ and $B^{(0)} = B$ and $(A^{(k)}, B^{(k)})$ for $k \geq 1$ corresponds to the deflated matrix pair obtained after the equivalence transformation in step $k-1$. After the structure indices are determined for step k, some consideration have to be made for the accumulation of the unitary transformations that transform $A - \lambda B$ into GUPTRI form, before the computation of the indices of step $k+1$ starts on the lower rightmost block of the transformed pencil.

The structure indices display the Kronecker structure as follows:

- $m_k - s_k =$ number of L_k blocks.
- $s_k - m_{k+1} =$ number of $J_{k+1}(0)$ blocks.

Applying the same algorithm to $B - \mu A$ results in the Jordan structure of the infinite eigenvalue and the right singular structure. The Jordan structure (and structure indices) associated with a finite but non-zero eigenvalue is obtained by applying the algorithm to a shifted pencil. One way to find the left singular structure is to apply the same algorithm to the transposed pencil. Another way is to directly determine the sizes of the corresponding row nullspaces as done in the `GUPTRI` algorithm, resulting in the Jordan structure of the infinite eigenvalue and the left singular structure. Then $m_k - s_k$ is the number of L_k^T blocks and $s_k - m_{k+1}$ equals the number of $N_{k+1} \equiv J_{k+1}(\infty)$ blocks.

However, the existing algorithms do not guarantee that the computed generalized Schur form is the "most" non-generic Kronecker structure within distance δ. However, if δ is of the size $O(\|(A, B)\|_E \epsilon)$, where ϵ is the relative machine precision, we know that (A, B) is close to a matrix with the Kronecker structure that the algorithm reports. It would of course be desirable to have algorithms that could solve the following "nearness" problems:

- Compute the closest non-generic pencil of a generic $A - \lambda B$.
- Compute the closest matrix pencil with a specified Kronecker structure.
- Compute the most non-generic pencil within a given distance δ.

3.5 The Geometry of Matrix Pencil Space

Analogously to the matrix case, we define an *orbit* to be the set of *strictly equivalent* pencils in $2mn$ dimensional space:

$$\mathrm{orbit}(A - \lambda B) = \{P^{-1}(A - \lambda B)Q : \det(P)\det(Q) \neq 0\}.$$

The dimension of $\mathrm{orbit}(A - \lambda B)$ is equal to the dimension of the tangent space, $\tan(A - \lambda B)$, to the orbit of $A - \lambda B$. The tangent space is defined as

$$f(X, Y) = X(A - \lambda B) - (A - \lambda B)Y, \tag{3.5.1}$$

where X is an $m \times m$ matrix and Y is an $n \times n$ matrix [11]. Since (3.5.1) maps a space of dimension $m^2 + n^2$ linearly to a space of dimension $2mn$, the dimension of the tangent space is $m^2 + n^2 - d$, where d is the number of (linearly) independent solutions of $f(X, Y) = 0$.

The codimension is the dimension of the space complementary to the tangent space, i.e.,

$$\mathrm{cod}(A - \lambda B) = 2mn - \dim(\tan(A - \lambda B)) = d - (m - n)^2.$$

The codimensions of the orbits depend only on their Kronecker structures, and it can be computed by summing the contributions to the codimension from different blocks in the KCF [11].

3.6 Contributions in This Thesis

By viewing an $m \times n$ matrix pencil $A - \lambda B$ as a point in $2mn$ space, certain numerical computations relating to the Kronecker canonical form can be viewed as moving matrix pencils from point to point or manifold to manifold in the $2mn$ space. Our *objective* is to make use of the geometry of the matrix and matrix pencil spaces, to improve our knowledge of numerical algorithms and their failures. Moreover, with this new knowledge our *goal* is to develop more robust and accurate algorithms (and software) for computing the generalized Schur form of a general matrix pencil and increase their functionality in order to be able to solve some of the "nearness" problems mentioned above.

Any existing algorithm for the computation of the Jordan or Kronecker canonical form of a matrix or pencil proceeds in stages. The most generic matrices are nonsingular and diagonalizable, so at the start of the computation the best possible guess is that the matrix will be diagonalizable. This notion is subject to modification as more information from the computation is obtained. As new information is obtained, our best guess narrows, so we may say that the matrix (or pencil) becomes increasingly less generic. (Of course, the matrix is not really changing, merely our knowledge about the matrix increases.)

It is clear that the orbit of the generic m-by-n pencils spans the complete $2mn$-dimensional space. It follows that all matrix pencils with other Kronecker structures "live" in the space spanned by the orbit of the generic pencils. We say that they are in the *closure* of the orbit of the generic pencils. It is just as clear that the m-by-n zero pencil is in the closure of the orbit of any other m-by-n pencil. However, all other closure relations between the orbits of different m-by-n matrix pencils are not that obvious.

The problem of *stratification* is to understand how the orbits relate to each other, i.e., to understand the closure hierarchy of Jordan and Kronecker structures. The algorithms move matrices (or pencils) from one orbit (or stratum) to another. Our *goal* is to understand how the geometry of the orbits influence the algorithms. To understand all possible Jordan forms, it is sufficient to consider the case of nilpotent matrices, i.e., matrices with all eigenvalues equal to 0. We aim at a similar understanding for the matrix pencil case.

Our geometric approach to the understanding of computations related to the Kronecker canonical form is a complement to existing perturbation theory for eigenvalue problems (e.g. see [38, 13]).

In the following we present short summaries of our contributions to these problems.

Paper III

We present a comprehensive study of the set of 2-by-3 pencils in order to get a greater understanding of the mentioned "nearness" problems. The set (or family) of 2-by-3 matrix pencils $A - \lambda B$ comprises 18 structurally different Kronecker structures (canonical forms). We show how all the non-generic structures can be generated by a staircase-type algorithm, starting from the generic canonical form (one L_2 block). Moreover, the algebraic and geometric characteristics of the generic and the 17 non-generic cases are examined in full detail.

All 2-by-3 pencils "live" in a 12-dimensional space spanned by the set of all generic pencils (i.e., orbit(L_2)). The dimension of orbit$(A - \lambda B)$ is equal to the dimension of the tangent space to the orbit and the codimension $(\mathrm{cod}(A - \lambda B))$ is the dimension of the space complementary to the tangent space (e.g. the normal space which is the space perpendicular to $\tan(A - \lambda B)$). Since orbit(L_2) with codimension zero spans the complete 12-dimensional space, it is obvious that all other structures are in the closure of the orbit of L_2, and it is just as obvious that $3L_0 \oplus 2L_0^T$ (the zero pencil) with codimension 12 is in the closure of the orbit of any other KCF. All other closure relations are not that obvious. The complete closure hierarchy, or the stratification, of the orbits of all different Kronecker structures is derived and presented in a closure graph that show how the structures relate to each other in the 12-dimensional space spanned by the set of 2-by-3 pencils.

Necessary conditions on perturbations for transiting from the orbit of one Kronecker structure to another in the closure hierarchy are derived and presented in a labeled closure graph. The node and arc labels show geometric characteristics of an orbit's Kronecker structure and the change of geometric characteristics when transiting to an adjacent node, respectively.

Computable normwise bounds for the smallest perturbations $(\delta A, \delta B)$ of a generic 2-by-3 pencil $A - \lambda B$ such that $(A + \delta A) - \lambda(B + \delta B)$ has a specific non-generic Kronecker structure are presented. First explicit expressions for the perturbations that transfer $A - \lambda B$ to a specified non-generic form are derived. In this context tractable and intractable perturbations are defined. Secondly, a modified `GUPTRI` that computes a specified Kronecker structure of a generic pencil is used. Perturbations devised to impose a certain non-generic structure is computed in a way that guarantees to find a KCF on the closure of the orbit of the intended KCF. Both approaches for computing perturbations to the generic structure in order to find the non-generic structures are illustrated by computational experiments.

Moreover, a study of the behaviour of the non-generic structures under random perturbations in finite precision arithmetic (using the `GUPTRI` software [15, 16]) show for which sizes of perturbations the structures are invariant and also that structure transitions occur in accordance with the closure hierarchy, i.e., all structure transitions correspond to going upwards, along the arcs in the closure hierarchy graph.

Finally, some of the results are extended to the general m-by-$(m+1)$ case.

Paper IV

The versal deformation of the Jordan normal form, derived by V. I. Arnold, 1971 [2], is an important tool for the understanding of how perturbations in different direction in the matrix space affect the Jordan structure.

By computing the tangent space to the orbit in the point defined by a matrix A in Jordan normal form he derived a parametrized deformation, transversal to the orbit (and the tangent space). A versal deformation that is obtained by adding the transversal deformation to A is then a normal form that for all matrices close to A depends continuously on the elements in the matrix. Of course, numerical analysts prefer a deformation in the normal space, i.e., orthogonal to the tangent space, instead of the transversal space.

By making use of the geometry of the matrix and matrix pencil spaces, we believe that our knowledge of numerical algorithms and their failures can be improved. Therefore it is not the versal deformation itself that is most important for our purposes. We are more interested in the metrical information that it provides for the perturbation theory of matrices and matrix pencils relevant to the Jordan and Kronecker canonical forms.

We generalize and extend the concept of versal deformations to matrix pencils (matrix pairs) and we derive a normal form to which not only one specific matrix pencil, but an arbitrary family of matrix pencils close to it can be reduced to by means of a mapping smoothly depending on the elements of the matrix pairs. The versal deformation for matrix pencils is more complicated than for the matrix case, since the Kronecker canonical form may consist of Jordan blocks for both finite and infinite eigenvalues as well as Kronecker blocks for right and left minimal indices.

For $A - \lambda B$ in Kronecker canonical form our approach is to show how to compute the tangent space $\tan(A - \lambda B)$ of the orbit and a parameterized deformation $Z_A - \lambda Z_B$ that defines an orthogonal basis for the normal space of $\mathrm{orbit}(A - \lambda B)$. The versal deformation

$$(A + \lambda B) + (Z_A - \lambda Z_B)$$

is a normal form that depends continuously on the elements of A and B for all pencils close to $A - \lambda B$ and it spans the complete $2mn$ space. Such a versal deformation with minimum number of parameters is a called a miniversal deformation.

We conclude by numerical experiments where we use the `GUPTRI` software to compute the Kronecker structure of a given 2-by-3 matrix pencil $A - \lambda B$ in KCF after adding random perturbations to it. We decompose the random perturbations in two parts, one in $\tan(A - \lambda B)$ and one in $\mathrm{nor}(A - \lambda B)$. We study how far we can move in the tangent and normal directions before the pencil turn generic.

We also illustrate by a more detailed example how the versal deformation can be used to find the structures above a given KCF in the Kronecker structure hierarchy.

Paper V

Any existing algorithm for computation of the Jordan or Kronecker canonical form proceeds in stages where each stage reduces the number of possible structures, or geometrically, reduces the matrix space or matrix pencil space to a space that contains a smaller number of orbits. The problem of *stratification* is to understand how the orbits relate to each other. More specific, since one step of the algorithm reduces the matrix or matrix pencil space to a space that contains a smaller number of orbits, it is of great interest to know which orbits that are in the closure of some other orbits. For matrices, the stratification can simply be revealed by a study of nilpotent matrices. For matrix pencils the problem becomes more complex.

For a given matrix pencil $A - \lambda B$, the Kronecker structure hierarchy shows all structures that are within the closure of $\mathrm{orbit}(A - \lambda B)$, and each structure, whose orbit's closure contains $A - \lambda B$. In order to gain new insight in the problem of stratification, we give new interpretations of important results by Pokrzywa, for determining closure relations among orbits of Kronecker structures. This is partly done by a generalization of classical theorems (Theorem 3.1 and its dual) by Gantmacher. Using these extensions, we rewrite some of Pokrzywas results in terms of Weyr characteristics and the dimension of the column and row nullspaces of the Gantmacher matrices $R[A, B, i]$ and $L[A, B, i]$, $i = 0, 1, 2, \ldots$. The results are used to derive an algorithm for computation of the complete Kronecker structure hierarchy, or the Kronecker structure hierarchy above or below a given structure. The algorithm is presented in terms of the rank-decisions required in a staircase algorithm, in order to compute the Kronecker structure hierarchy.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[2] V. I. Arnold. On Matrices Depending on Parameters. *Russian Math. Surveys*, pages 29–43, 1971.

[3] T. Beelen and P. Van Dooren. An improved algorithm for the computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 105:9–65, 1988.

[4] J. Bunch, J. Dongarra, C. Moler, and G. W. Stewart. *LINPACK User's Guide*. SIAM, Philadelphia, PA, 1979.

[5] K. Dackland and E. Elmroth. Parallel Computations on the IBM 3090/600E VF: Block Algorithms for Matrix Multiplication and LU Factorization. *Master's thesis* UMNAD-66.90, Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden, 1990. (In Swedish).

[6] K. Dackland and E. Elmroth. Design and Performance Modeling of Parallel Block Matrix Factorizations for Distributed Memory Multicomputers. In *Proceedings of the Industrial Mathematics Week*, pages 102–116, 1992.

[7] K. Dackland and E. Elmroth. Design, Modeling, and Evaluation of Parallel Block Matrix Factorization Algorithms for Shared and Distributed Memory Architectures. *Licentiate thesis* UMINF-92.07, Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden, June 1992.

[8] K. Dackland and E. Elmroth. Parallel Block Matrix Factorizations for Distributed Memory Multicomputers. Report UMINF-92.03, Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden, May 1992.

[9] K. Dackland, E. Elmroth, B. Kågström, and C. Van Loan. Design and Evaluation of Parallel Block Algorithms: LU Factorization on an IBM

3090 VF/600J. In J. J. Dongarra et al, editor, *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, pages 3–10, Houston, 1992. SIAM Publications.

[10] J. Demmel. The condition number of equivalence transformations that block diagonalize matrix pencils. *SIAM J. Num. Anal.*, 20(3):599–610, June 1983.

[11] J. Demmel and A. Edelman. The Dimension of Matrices (Matrix Pencils) with Given Jordan (Kronecker) Canonical Forms. Report LBL-31839, Mathematics Department, Lawrence Berkeley Laboratories, University of California, Berkeley, CA 94720, 1992. To appear in *Lin. Alg. Appl.*

[12] J. Demmel and B. Kågström. Stably computing the Kronecker structure and reducing subspaces of singular pencils $A - \lambda B$ for uncertain data. In Jane Cullum and Ralph A. Willoughby, editors, *Large Scale Eigenvalue Problems*, pages 283–323. North-Holland, Amsterdam, 1986. Mathematics Studies Series Vol. 127, Proceedings of the IBM Institute Workshop on Large Scale Eigenvalue Problems, July 8-12, 1985, Oberlech, Austria.

[13] J. Demmel and B. Kågström. Computing stable eigendecompositions of matrix pencils. *Lin. Alg. Appl.*, 88/89:139–186, April 1987.

[14] J. Demmel and B. Kågström. Accurate solutions of ill-posed problems in control theory. *SIAM J. Mat. Anal. Appl.*, 9(1):126–145, January 1988.

[15] J. Demmel and B. Kågström. The Generalized Schur Decomposition of an Arbitrary Pencil $A - \lambda B$: Robust Software with Error Bounds and Applications. Part I: Theory and Algorithms. *ACM Trans. Math. Software*, Vol.19(No. 2):160–174, June 1993.

[16] J. Demmel and B. Kågström. The Generalized Schur Decomposition of an Arbitrary Pencil $A - \lambda B$: Robust Software with Error Bounds and Applications. Part II: Software and Applications. *ACM Trans. Math. Software*, Vol.19(No. 2):175–201, June 1993.

[17] J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Software*, 18(1):1–17, 1990.

[18] C.C. Mac Duffee. *The Theory of Matrices.* Chelsea Publishing Company, New York, 1956.

[19] T. F. Fairgrieve. The Application of Singularity Theory to the Computation of Jordan Canonical Form. *Master's thesis*, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, September 1986.

[20] F. Gantmacher. *The Theory of Matrices, Vol. I and II (transl.).* Chelsea, New York, 1959.

[21] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler. *Matrix Eigensystem Routines – EISPACK Guide Extension*, volume 51 of *Lecture Notes in Computer Sciences*. Springer-Verlag, Berlin, 1977.

[22] G. Golub and C. Van Loan. *Matrix Computations.* Second Edition. Johns Hopkins University Press, Baltimore, MD, 1989.

[23] G. Golub and J. H. Wilkinson. Ill-conditioned eigensystems and the computation of the Jordan canonical form. *SIAM Review*, 18(4):578–619, 1976.

[24] B. Kågström. How to compute the Jordan normal form: the choice between similarity transformations and methods using the chain relations. Report UMINF-91.81, Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden, 1981.

[25] B. Kågström. The generalized singular value decomposition and the general $A - \lambda B$ problem. *BIT*, 24:568–583, 1984.

[26] B. Kågström. RGSVD - an algorithm for computing the Kronecker canonical form and reducing subspaces of singular matrix pencils $A - \lambda B$. *SIAM J. Sci. Stat. Comp.*, 7(1):185–211, 1986.

[27] B. Kågström, P. Ling, and C. Van Loan. High-Performance GEMM-Based Level-3 BLAS: Sample Routines for Double Precision Real Data. In M. Durand and F. El Dabaghi, editors, *High Performance Computing II*, pages 269–281. Elsevier Science Publisher B.V., 1991.

[28] B. Kågström and A. Ruhe. ALGORITHM 560: An algorithm for the numerical computation of the Jordan normal form of a complex matrix [F2]. *ACM Trans. Math. Software*, 6(3):437–443, 1980.

[29] B. Kågström and A. Ruhe. An algorithm for the numerical computation of the Jordan normal form of a complex matrix. *ACM Trans. Math. Software*, 6(3):389–419, 1980.

[30] V. B. Khazanov and V. Kublanovskaya. Spectral problems for matrix pencils. Methods and algorithms. I. *Sov. J. Numer. Anal. Math. Modelling*, 3:337–371, 1988.

[31] L. Kronecker. Algebraische Reduction der Schaaren Bilinearer Formen. S. B. Akad., 1890. pp. 763–776.

[32] V. Kublanovskaya. On a method for solving the complex eigenvalue problem for a degenerate matrix. *USSR Comput. Math. Phys.*, 6(4):1–14, 1968.

[33] V. Kublanovskaya. An approach to solving the spectral problem of $A - \lambda B$. In B. Kågström and A. Ruhe, editors, *Matrix Pencils*, pages 17–29. Springer-Verlag, Berlin, 1983. Lecture Notes in Mathematics, vol. 973, Proceedings, Pite Havsbad, 1982.

[34] V. Kublanovskaya. AB-algorithm and its modifications for the spectral problem of linear pencils of matrices. *Num. Math.*, 43:329–342, 1984.

[35] P. Ling. A Set of High Performance Level 3 BLAS Structured and Tuned for the IBM 3090 VF and Implemented in Fortran 77. *The Journal of Supercomputing*, 7(3):323–355, September 1993.

[36] A. Ruhe. An algorithm for numerical determination of the structure of a general matrix. *BIT*, 10:196–216, 1970.

[37] 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*, volume 6 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1976.

[38] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory*. Academic Press, New York, 1990.

[39] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 27:103–141, 1979.

[40] P. Van Dooren. The generalized eigenstructure problem in linear system theory. *IEEE Trans. Autom. Contr.*, AC-26(1):111–129, 1981.

[41] P. Van Dooren. Reducing subspaces: Definitions, properties and algorithms. In B. Kågström and A. Ruhe, editors, *Matrix Pencils*, pages 58–73. Springer-Verlag, Berlin, 1983. Lecture Notes in Mathematics, vol. 973, Proceedings, Pite Havsbad, 1982.

[42] W. Waterhouse. The codimension of singular matrix pairs. *Lin. Alg. Appl.*, 57:227–245, 1984.

[43] K. Weierstrass. Zur theorie der bilinearen und quadratischen Formen. *Monatsh. Akad. Wiss. Berlin*, 1:310–338, 1867.

[44] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford Science Publications, 1965.

[45] J. H. Wilkinson. Linear differential equations and Kronecker's canonical form. In C. de Boor and G. Golub, editors, *Recent Advances in Numerical Analysis*, pages 231–265. Academic Press, 1978.

Paper I

Parallel Block Matrix Factorizations on the Shared Memory Multiprocessor IBM 3090 VF/600J*

Krister Dackland†, Erik Elmroth†,
Bo Kågström, and Charles Van Loan‡

Department of Computing Science, Umeå University
S-901 87 Umeå, Sweden.
E-mail: dacke@cs.umu.se, elmroth@cs.ume.se, bokg@cs.ume.se, cv@cs.cornell.edu

Abstract

Efficient parallel block algorithms for the LU factorization with partial pivoting, the Cholesky factorization, and QR factorization transportable over a range of parallel MIMD architectures are presented. Parallel implementations of different block algorithms which utilize optimized uniprocessor level-3 BLAS are compared with corresponding routines of LAPACK (under development). Parallelism is mainly invoked implicitly in LAPACK by replacing calls to uniprocessor level-3 kernels by calls to parallel level-3 kernels and thereby maintaining portability. However, by parallelizing at the block level (explicitly) it is possible to overlap and pipeline different matrix-matrix operations and thereby gain some performance. Load balancing of the explicitly parallel block algorithms is by static or dynamic scheduling of the work. The static scheduling utilizes cost functions based on the number of *flops* expressed as GEMM equivalents. The load balancing of the implicit approach is inherited from the parallel level-3 kernels. The implementations are done in IBM Parallel Fortran and performance results for an IBM 3090 VF/600J are presented. Theoretical models give upper bounds on the best possible speedup of the explicitly and implicitly parallel block algorithms for the target machine.

*From *THE INTERNATIONAL JOURNAL OF SUPERCOMPUTER APPLICATIONS*, Vol. 6:1, by permission of the MIT Press, Cambridge, Massachusetts.

†Financial support has been received by the Swedish National Board of Industrial and Technical Development under grant NUTEK 89-02578P.

‡Department of Computer Science, Cornell University, Ithaca, New York 14853-7501.

1 Introduction

With the introduction of advanced parallel computer architectures a demand for efficient parallel algorithms has emerged. During the last five years we have seen many research activities concerning algorithm design for different vector and parallel architectures (e.g. see [12, 15, 1]). Today it is well-known that block algorithms are required to exploit the full potential of hierarchical memory computers and multiprocessors [14]. Typically, the memory hierarchy of parallel shared memory systems consists of registers (including vector registers) and cache memory at the top level and a global shared memory. At the lower level we may also see an optional expanded storage (slower but not as costly as main memory [7]), disc and tape storage. The crucial fact is that excessive movement of data to and from main memory decreases the performance. Block algorithms make it possible to reuse data in cache memory and in vector registers. By suitable choices of block sizes that are both architecture and problem dependent it is possible to obtain high performance on many of the commercially available vector multiprocessors produced, for example, by Alliant, Cray, and IBM.

Different matrix factorizations are basic and important tools in most scientific, economic, and engineering computational problems. In this paper we focus on the LU factorization with partial pivoting of a general matrix, the LL^T (Cholesky) factorization of a positive definite matrix and the QR factorization of a general matrix. Our goal is to design parallel block algorithms with high parallel efficiency for these factorizations and that still are transportable over a range of parallel MIMD architectures. To attain this goal we have to consider as many as possible of the following parallel processing "key factors": balancing the load over the processors, maintain the locality of data by suitable choice of granularity, minimizing memory contention between processors, and minimizing the number of synchronization points in a parallel algorithm. Notice that several of these factors can be contradictory.

Our contribution builds on the work of many other research groups (e.g. [12, 14, 8, 26, 5]). The single most important source of inspiration is the LAPACK project [5, 2, 1] which goal is to design and implement in Fortran 77 a portable linear algebra library for efficient use on a variety of high-performance computers. LAPACK is based on block algorithms and therefore especially designed to utilize the level-3 BLAS [10] as the major computational kernels. Parallelism is mainly invoked implicitly in LAPACK by replacing calls to uniprocessor level-3 kernels by calls to parallel level-3 kernels. The main advantage with this approach is that the portability of LAPACK is maintained. However, the implicit parallelism imposes a synchronization after each level-3 operation, which means that idle processors must wait until all processors have completed their share of work. This can be overcome by invoking the parallelism explicitly at the block level of the algorithm. Then processors can work on independent block operations which makes it possible to overlap and pipeline different matrix-matrix operations. Further, only efficient uniprocessor level-3 kernels are required. Here the parallelism is invoked by using parallel language constructs. The proposal for standardization of such constructs in Fortran by the *Parallel Computing Forum (PCF)* [25] will make it possible to write portable code too.

Although we are using the vector multiprocessor IBM 3090 VF/600J [30] as our target architecture and IBM Parallel Fortran [29] as our implementation language, it is possible to implement the parallel algorithms described here on similar architectures like Alliant and Cray. The parallel language extensions of IBM Parallel Fortran [29] that we are using (see Section 7) are functionally equivalent to the PCF proposal [25]. Further, the data partitioning, inherent overlapping, and pipelining at the block level, and the description of the parallel block algorithms as node programs also make them suitable to implement as distributed ring-oriented block algorithms with column block-wrap mapping of the matrix to factorize.

The outline of the rest of the paper is as follows. In Section 2 block algorithms and basic linear algebra subprograms are introduced. Optimized uniprocessor level-3 BLAS and parallel level-3 kernels are described. Section 3 describes different variants of block algorithms for the LU, LL^T, and QR factorizations considered. Uniprocessor performances of these variants for different matrix sizes and block sizes are also presented. In Section 4 implicit and explicit parallelism is introduced and the implemented parallel block algorithms for the LU, LL^T, and QR factorizations are described. Explicit parallelization with static load balancing is the topic of Section 5. Cost functions based on the number of *flops* expressed as GEMM equivalents and used to balance the load on available processors are described. In a multi-user system dynamic scheduling of the work to processors is likely to be a better approach. Section 6 presents explicitly parallel block algorithms with dynamic load balancing. Performance results of the parallel block algorithms discussed in sections 4–6 are presented in Section 7. Finally, in Section 8 we draw some conclusions.

2 Block Algorithms and BLAS

By a *block algorithm* we mean one that is rich in basic matrix-matrix operations such as matrix-matrix multiply, rank-k matrix updates, and solving triangular systems with multiple right-hand sides. In a block algorithm the entities on which we perform operations on are submatrices (or blocks). Let an $m \times n$ matrix A be partitioned in p row blocks and q column blocks such that

$$\begin{bmatrix} A_{11} & \cdots & A_{1q} \\ \vdots & \ddots & \vdots \\ A_{p1} & \cdots & A_{pq} \end{bmatrix}$$

where block A_{ij} has dimension $m_i \times n_j$ and we say that $A = (A_{ij})$ is a $p \times q$ block matrix (notation from [16]). Typically we will work with a fixed block size so that $m_i = n_j = nb$. In these cases the pth block row and the qth block column may have non-square blocks. If nothing else is explicitly stated we assume that m and n are even multiples of nb. This simplifies the notation when describing block algorithms in the coming sections. (Otherwise we need an extra if-statement that takes care of the last block row or column.) In the following, we use the colon notation to specify columns, rows, or submatrices of A. For example, $A_{:,i}$ denotes the ith column of A and $A_{i:j,k:l}$ denotes the entries of A in rows i through j and columns k through l.

The methodology used to develop a block algorithm for a matrix computation is to restructure well-known and stable elementwise algorithms to perform block matrix-matrix operations in their inner loops. The most common matrix factorizations like LU and LL^T involve three loops controlled by indices i, j, and k, respectively. Different orderings of the three loops imply different data movements and amounts of data transferred [12]. Since the implementations are done in Fortran [29], [23], we mainly consider column-oriented algorithms (i.e., kji, jki or jik orderings of the loops). Further, different loop orderings may also require different block matrix operations. Level-3 BLAS [10] and some of the operations from level-1 BLAS [21] and level-2 BLAS [9] are enough to implement most matrix factorizations. In the coming two subsections we will introduce the level-3 kernels used in our work.

2.1 Optimized Uniprocessor Level-3 BLAS

The level-3 BLAS [10] is a small set of computational kernels proposed to be a standard for basic matrix-matrix operations such as matrix-matrix multiply, rank-k matrix updates, and solving triangular systems with multiple right-hand sides. The large granularity of the level-3 BLAS makes it possible to reuse data at the top of the memory hierarchy (vector registers and cache memory) and to minimize data movements. One way of measuring this is the ratio of arithmetic operations to memory references (see e.g. [1]). In Table 1 these ratios for a level-1 (`DAXPY`), level-2 (`DGEMV`), and level-3 (`DGEMM`) operation are summarized (vectors and matrices are of size $n \times 1$ and $n \times n$, respectively). We strive for as large a ratio as possible since this implies that a piece of the data can be kept longer in the cache memory and thereby reused more efficiently.

The performance numbers of these operations for an IBM 3090 VF (J model) are displayed in the right-most column of Table 1 which shows the great advantage of using matrix-matrix operations. The theoretical peak performance of the IBM 3090 VF (J model, one processor) is 138 Mflops (two instructions executed during one clock cycle, 14.5 $nsec$). We define the *practical peak performance* of a machine as the performance of the level-3 BLAS `DGEMM` which for the J-model is around 80% of the theoretical peak performance. This degradation in performance is due mainly to the characteristics of the vector instructions and the memory hierarchy of the machine [30, 7].

Computer manufactures are expected to provide highly optimized implemen-

Table 1: Performance of level-1, level-2 and level-3 BLAS ($\alpha, \beta \neq 0, \pm 1$).

BLAS Op	Mem Refs	Arith Ops	Ratio Ops:Refs	Performance
Level-1 BLAS				
$y = \alpha x + y$	$3n$	$2n$	$2:3$	11.9 Mflops
Level-2 BLAS				
$y = \alpha A x + \beta y$	$n^2 + 3n$	$2n^2 + 2n$	$2:1$	78.3 Mflops
Level-3 BLAS				
$C = \alpha AB + \beta C$	$4n^2$	$2n^3 + 2n^2$	$n:2$	107.2 Mflops

tations of the level-3 BLAS. At present this is not always the case; for example IBM provides only two of the six level-3 BLAS in double precision (`DGEMM` and `DTRSM`) [17]. (The new Release 1.4 of ESSL provides also `DSYRK`.) Our research group in Umeå has developed a set of high performance level-3 BLAS for the IBM 3090 VF implemented in Fortran 77 [19, 22]. High performance is achieved by structuring and tuning the Fortran code to maximize the reuse of data in vector registers and in cache memory and to direct the compiler to use compound vector instructions. For details of implementation and performance results see [22]. On the basis of performance results we have chosen the best available routines which are `DGEMM` from the IBM ESSL library [17] and all other routines from [22]. These uniprocessor routines are used in the explicitly parallel block algorithms (see Section 4) and on individual processors in the parallel level-3 kernels.

2.2 Parallel Level-3 Kernels

At present IBM is not providing any parallel kernels with the same functionality as level-3 BLAS. (The parallel matrix multiply in Release 1.4 of ESSL does not have the functionality of `DGEMM`.) Therefore we have developed some level-3 kernels in Parallel Fortran [29] which call uniprocessor level-3 BLAS. This work is in progress and so far we have only developed parallel level-3 kernels with the functionality required by the block algorithms discussed in Section 3. The work of the parallel level-3 kernels is load balanced either by *static scheduling* in which the distribution of the computational work onto available processors is determined in advance (`PDGEMM`, `PDTRSM`, and `PDTRMM`), or *dynamic scheduling* in which the work is distributed during the execution so that an idle processor is scheduled the next available task (`PDSYRK`).

`PDGEMM` is the parallel version of the general matrix-matrix multiply-and-add routine `DGEMM`. The parallelization is based on a row block partition of A and C while B is left unblocked. A and C are partitioned into row blocks of size *VSS* (= vector section size, which is the length of the vector registers in double precision words, 256 for the J model). Typically, this will leave a row stripe with fewer than *VSS* rows. This stripe is partitioned columnwise in a similar way. Now the row blocks and column blocks are distributed evenly over the number of available processors P. For each matrix-matrix multiply-and-add operation the optimized uniprocessor version of `DGEMM` is called. The performance gain obtained by blocking in this way is overall between 5% and 10% compared to when A and C are partitioned into P row blocks. `PDGEMM` is used by LU ($C = \beta C + \alpha AB$), LL^T ($C = \beta C + \alpha AB^T$), and QR ($C = \beta C + \alpha A^T B$ and $C = \beta C + \alpha AB^T$).

`PDTRSM` is the parallel version of the level-3 kernel for solving a triangular system with multiple right-hand sides $B = \alpha T^{-1} B$, where B is $m \times n$, T is $m \times m$ and lower triangular, $n > m$. The matrix B of right-hand sides is partitioned into P column blocks and each processor solves a triangular system with n/P right hand sides by calling the optimized uniprocessor version of `DTRSM`. The last processor $P - 1$ may solve a system with fewer right hand sides. In the case $B = \alpha BT^{-T}$, where B is $m \times n$, T is $n \times n$ and lower triangular, $n < m$, B is partitioned into P row blocks and processor $i = 0 : P - 1$ performs the operation $B_{s:e,:} = \alpha B_{s:e,:} \cdot T^{-T}$ ($s = i \cdot m/P + 1$, $e = (i+1) \cdot m/P$) by calling the optimized uniprocessor version of `DTRSM`. The last row block of B may have fewer rows than m/P. `PDTRSM` is used by LU (first case) and by LL^T (second case).

`PDTRMM` is the parallel version of the level-3 kernel triangular matrix-matrix multiply $B = \alpha BT$, where B is $m \times n$, T $n \times n$ and upper triangular, $m > n$. Also here B is partitioned into P row blocks and processor $i = 0 : P-1$ performs the operation $B_{s:e,:} = \alpha B_{s:e,:} \cdot T$ ($s = i \cdot m/P + 1$, $e = (i+1) \cdot m/P$) by calling the optimized uniprocessor version of `DTRMM`. The last row block of B may have fewer rows than m/P. `PDTRMM` is only used by QR.

`PDSYRK` is the parallel version of the symmetric rank-k update operation $C = \alpha AA^T + \beta C$, where $C = C^T$ is $n \times n$, A is $n \times k$. For the case $k < n$, A is partitioned into $nbb \times k$ block rows which means that the symmetric rank-k update of C can be performed by symmetric rank-k updates of $nbb \times nbb$ diagonal blocks (using `DSYRK`) and rectangular off-diagonal blocks are updated by `DGEMM` operations which is the approach of the GEMM-based level-3 BLAS [20]. Since each single call to these level-3 BLAS writes to different parts of C all level-3 operations can be executed in parallel. By considering the level-3 operations as a queue of tasks the parallel work is dynamically scheduled among the processors. The blocking and parallelization of the rank-k update are discussed in more detail in sections 4.2 and 6.2. `PDSYRK` is only used by LL^T.

Depending on the problem size there is a break-even point for each parallel level-3 kernel when it is more profitable to solve the problem on only one processor. Only `PDGEMM` and `PDSYRK` handles this tradeoff.

In Table 2 some sample results of the parallel level-3 kernels are presented (performance in Mflops and speedup in relation to the uniprocessor performance). The results presented are obtained using up to five physical processors of a quasi-dedicated IBM 3090 VF/600J. The smallest dimension (m, n, or k)

is chosen as the block size nb typically used in the block algorithms described later. The two other dimensions are set to 1000 and the leading dimension of A, B, and C to 1200. The results of PDSYRK for 1 to 5 processors are obtained with $nbb = 256, 256, 192, 128$, and 96, respectively.

Table 2: Performance of parallel level-3 kernels.

Routine Used in	PDGEMM LU	PDGEMM LL^T	PDTRSM LU	PDTRSM LL^T	PDTRMM QR	PDSYRK LL^T
Block size	$k = 48$	$n = 256$	$m = 48$	$n = 64$	$n = 32$	$k = 64$
1 proc	100.9	102.2	69.9	82.6	85.1	97.5
2 proc	182.6	196.1	137.3	162.7	165.2	194.3
Speedup	*1.81*	*1.92*	*1.96*	*1.97*	*1.94*	*1.99*
3 proc	252.0	290.2	192.2	224.4	225.3	285.1
Speedup	*2.50*	*2.84*	*2.75*	*2.72*	*2.65*	*2.92*
4 proc	313.6	397.4	261.8	311.8	304.4	368.9
Speedup	*3.11*	*3.89*	*3.75*	*3.77*	*3.58*	*3.78*
5 proc	402.6	481.6	312.5	371.9	363.4	455.0
Speedup	*3.99*	*4.72*	*4.47*	*4.50*	*4.27*	*4.67*

3 Block Matrix Factorizations

This section is devoted to a unified description of different variants of block algorithms for the LU, LL^T, and QR factorizations considered. Block algorithms are described in a notation quite similar to the one used in MATLAB [24] and CONLAB [13, 18] with references to the level-3 BLAS operations to use. One exception here is that the end of loop and control constructs are determined by indentation of statements (operations). Uniprocessor performance results of these variants for different matrix sizes and block sizes are also presented.

3.1 Block LU Algorithms

The LU factorization of an $m \times n$ matrix A is given by

$$PA = LU,$$

where the $m \times n$ matrix L is unit lower trapezoidal, the $n \times n$ matrix U is upper triangular, and P is a permutation matrix corresponding to row interchanges of A. The basic algorithm to compute a reasonably stable LU factorization is Gaussian elimination with partial pivoting [16]; see also [8].

We have considered three column-oriented block algorithms that can be derived easily from the following block-matrix equality

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \cdot \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

where we have ignored the permutations required by partial pivoting.

They have in common that a block column of the matrix is LU factorized by a level-2 routine. Then a part of U is computed by solving a triangular system with multiple right-hand sides, an operation that can be performed by the level-3 BLAS routine `DTRSM`. Finally, a part of the matrix is updated by a rank-k update performed by the level-3 BLAS routine `DGEMM` (size of k is different for the algorithms). They differ in the order in which they perform the above operations, which also affects the way matrix elements are referenced. A short description of the major steps of the block algorithms that perform exactly the same number of flops follows. The level-2 LU factorization used in all three block algorithms is a left-looking algorithm (see below) that calls the level-2 BLAS routines `DTRSV` and `DGEMV` for solving triangular systems and multiplying a matrix by a vector, respectively. As for the elementwise algorithms L and U overwrite A, i.e., no extra storage is required by the block algorithms.

3.1.1 LU: Block Right - Looking

The block right-looking algorithm, which also is called the block KJI or block SAXPY algorithm, refers data mainly on the right-hand side of the current block column. We illustrate the main operations of the algorithm by referring to the 3×3 block matrix above. First, the LU factorization of the first block column of A is computed giving L_{11}, L_{21}, L_{31}, and U_{11}. Then U_{12} and U_{13} are obtained by solving $L_{11}[U_{12}, U_{13}] = [A_{12}, A_{13}]$. The remaining part of A is updated with respect to the first block column of L and block row of U:

$$\begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix} - \begin{bmatrix} L_{21} \\ L_{31} \end{bmatrix} \cdot \begin{bmatrix} U_{12} & U_{13} \end{bmatrix}.$$

This completes one step of the block algorithm and the same operations are now applied to the recently updated submatrix of A until the factorization is completed.

For $i = 1 : min(n/nb, m/nb)$
  $s = (i-1) \cdot nb + 1$ — $\{$Start of block to factorize$\}$
  $e = i \cdot nb$ — $\{$End of block to factorize$\}$
  $u = e + 1$ — $\{$Start position for update$\}$
  1. $L_{s:m,s:e} U_{s:e,s:e} = A_{s:m,s:e}$ — $\{$Use level-2 routine$\}$
  If $i > 1$

2. Pivotings are applied to $A_{s:e,1:s-1}$ {From last factorization}

If $e < n$

3. Pivotings are applied to $A_{s:e,u:n}$ {From all previous factorizations}

4. $U_{s:e,u:n} \leftarrow L^{-1}_{s:e,s:e} A_{s:e,u:n}$ {Use `DTRSM` to solve for part of U}

If $e < m$

5. $A_{u:m,u:n} \leftarrow A_{u:m,u:n} - L_{u:m,s:e} U_{s:e,u:n}$ {Use `DGEMM`}

Notice that the parts of L and U that have been computed are never referenced again except in the application of pivotings from the last factorization (operation 2). The if-statement before operation 5 handles the case when $m < n$.

3.1.2 LU: Block Left-Looking

The block left-looking algorithm, which is also called the block JKI or block GAXPY algorithm, refers data mainly on the left-hand side of the current block column. It also starts by computing the LU factorization of the first block column of A giving L_{11}, L_{21}, L_{31}, and U_{11}. Then only U_{12} is computed by solving $L_{11}[U_{12}] = [A_{12}]$. Finally, the next block column of A is updated with all previous parts of L and U which in the first step is

$$\begin{bmatrix} A_{22} \\ A_{32} \end{bmatrix} = \begin{bmatrix} A_{22} \\ A_{32} \end{bmatrix} - \begin{bmatrix} L_{21} \\ L_{31} \end{bmatrix} \cdot \begin{bmatrix} U_{12} \end{bmatrix}$$

This completes one step of the block algorithm, and the same operations are now applied to the recently updated submatrix of A until the factorization is completed.

For $i = 1 : min(n/nb, m/nb)$

$s = (i-1) \cdot nb + 1$ {Start of block to factorize}

$e = i \cdot nb$ {End of block to factorize}

$u = e + 1$ {Start position for update}

$v = u + nb - 1$ {End position for update}

1. $L_{s:m,s:e} U_{s:e,s:e} = A_{s:m,s:e}$ {Use level-2 routine}

If $i > 1$

2. Pivotings are applied to $A_{s:e,1:s-1}$ {From last factorization}

If $e < n$

3. Pivotings are applied to $A_{s:e,u:v}$ {From all previous factorizations}

4. $U_{1:e,u:v} \leftarrow L^{-1}_{1:e,1:e} A_{1:e,u:v}$ {Use `DTRSM`}

5. $A_{u:m,u:v} \leftarrow A_{u:m,u:v} - L_{u:m,1:e} U_{1:e,u:v}$ {Use `DGEMM`}

If $n > m + nb$

6. Pivotings are applied to $A_{1:m,m+nb+1:n}$ {From all previous factorizations}

7. $U_{1:m,m+nb+1:n} \leftarrow L^{-1}_{1:m,1:m} A_{1:m,m+nb+1:n}$ {Use `DTRSM`}

The if-statement before operations 6 and 7 handles the case when $m < n$ and there is still parts of U to compute. In this algorithm where the triangular solve and the update are performed on nb columns at a time, all columns on the left-hand side of the current block column have to be referenced in the triangular solve and in the update. This also results in a larger number of operations performed by `DTRSM` and a smaller number performed by `DGEMM` compared to the block right-looking algorithm.

3.1.3 LU: Block Crout

The third algorithm, which is called the block Crout algorithm, the block JIK, or the block SDOT algorithm, is in some sense a compromise between the two described algorithms.

For $i = 1 : min(n/nb, m/nb)$
- $s = (i-1) \cdot nb + 1$ {Start of block to factorize}
- $e = i \cdot nb$ {End of block to factorize}
- $u = e + 1$ {Start position for update}
- $v = u + nb - 1$ {End position for update}
- 1. $L_{s:m,s:e}U_{s:e,s:e} = A_{s:m,s:e}$ {Use level-2 routine}
- If $i > 1$
  - 2. Pivotings are applied to $A_{s:e,1:s-1}$ {From last factorization}
- If $e < n$
  - 3. Pivotings are applied to $A_{s:e,u:n}$ {From all factorizations}
  - 4. $A_{s:e,u:n} \leftarrow A_{s:e,u:n} - L_{s:e,1:e}U_{1:e,u:n}$ {Use `DGEMM`}
  - 5. $U_{s:e,u:n} \leftarrow L^{-1}_{s:e,s:e}A_{s:e,u:n}$ {Use `DTRSM`}
  - 6. $A_{u:m,u:v} \leftarrow A_{u:m,u:v} - L_{u:m,1:e}U_{1:e,u:v}$ {Use `DGEMM`}

The block Crout algorithm performs the same number of operations in `DTRSM` and `DGEMM` as the block right-looking algorithm. The matrix multiplication, though, is split into two separate operations, of which the second is similar to the one performed in the block left-looking algorithm.

3.1.4 Pivotings

It is a well-known fact that, on the IBM 3090 VF, the row swaps on a block are done most efficiently if all permutations are applied to one column at a time (e.g. see [2]). This means that the level-1 BLAS routine `DSWAP` should *not* be used. We have also found it advantageous in the block right-looking algorithm to delay the application of pivotings on the left-hand side of the factorized block until the end of the computations. This gives a possibility to do all swaps to each completed column in one iteration (implemented in the explicitly parallel algorithms, discussed in sections 4–6).

3.2 Block Cholesky Algorithms

The Cholesky factorization of an $n \times n$ positive definite, symmetric matrix A is given by

$$A = LL^T,$$

where L is the unique lower triangular matrix. Since A is symmetric positive definite no pivoting is needed to guarantee numerical stability. For a discussion of algorithms for computing the Cholesky factorization see [16]. We consider two column-oriented block algorithms (right- and left-looking, respectively) and one row-oriented block algorithm (top-looking).

The three block algorithms have in common that an $nb \times nb$ diagonal block is factorized by a level-2 routine. Then the remaining part of a block column (or row) of L is computed by solving a triangular system with multiple right-hand sides (`DTRSM`). Finally, the Cholesky factor of a block is applied to a part of A by a symmetric rank-k update. The update work is performed by `DGEMM` or `DSYRK`, or both of them.

The block Cholesky algorithms described below can be derived from the block LU algorithms by replacing U with L^T and removing unnecessary operations. Notice that in the block LU algorithms the level-2 routine factorizes a complete block column, which is imposed by the partial pivoting, while in the Cholesky algorithms only a diagonal block is factorized by a level-2 algorithm and then `DTRSM` is used to compute the remaining part of a block column (or row). The block algorithms perform exactly the same number of flops. Since A is symmetric only its lower (or upper) triangular part is accessed and overwritten by L. In all three algorithms only the *lower triangular* portion of A is referenced.

3.2.1 Cholesky: Block Right-Looking

The Cholesky block right-looking algorithm can be described as follows:

For $i = 1 : n/nb$
  $s = (i-1) \cdot nb + 1$ {Start of block to factorize}
  $e = i \cdot nb$ {End of block to factorize}
  $u = e + 1$ {Start position for update}
  1. $L_{s:e,s:e} L^T_{s:e,s:e} = A_{s:e,s:e}$ {Use level-2 routine}
  If $e < n$
    2. $L_{u:n,s:e} \leftarrow A_{u:n,s:e} L^{-T}_{s:e,s:e}$ {Use `DTRSM`}
    3. $A_{u:n,u:n} \leftarrow A_{u:n,u:n} - L_{u:n,s:e} L^T_{u:n,s:e}$ {Use `DSYRK`}

A is factorized one block column at a time, and the symmetric rank-nb update is applied to all remaining block columns of A in each iteration.

3.2.2 Cholesky: Block Left-Looking

The Cholesky block left-looking algorithm can be described as follows:

For $i = 1 : n/nb$
  $s = (i-1) \cdot nb + 1$ {Start of block to factorize}
  $e = i \cdot nb$ {End of block to factorize}
  $u = s + nb$ {Start position for update}
  If $i > 1$
    1. $A_{s:e,s:e} \leftarrow A_{s:e,s:e} - L_{s:e,1:s-1} L^T_{s:e,1:s-1}$ {Use DSYRK}
  2. $L_{s:e,s:e} L^T_{s:e,s:e} = A_{s:e,s:e}$ {Use level-2 routine}
  If $i > 1 \wedge i < n/nb$
    3. $A_{u:n,s:e} \leftarrow A_{u:n,s:e} - L_{u:n,1:s-1} L^T_{s:e,1:s-1}$ {Use DGEMM}
  If $i < n/nb$
    4. $L_{u:n,s:e} \leftarrow A_{u:n,s:e} L^{-T}_{s:e,s:e}$ {Use DTRSM}

A is factorized one block column at a time. In each iteration the previous factorized block columns are accessed in the symmetric rank-$(s-1)$ update of the next block column.

3.2.3 Cholesky: Block Top-Looking

The Cholesky block top-looking algorithm can be described as follows:

For $i = 1 : n/nb$
  $s = (i-1) \cdot nb + 1$ {Start of block to factorize}
  $e = i \cdot nb$ {End of block to factorize}
  If $i > 1$
    1. $L_{s:e,1:s-1} \leftarrow A_{s:e,1:s-1} L^{-T}_{1:s-1,1:s-1}$ {Use DTRSM}
    2. $A_{s:e,s:e} \leftarrow A_{s:e,s:e} - L_{s:e,1:s-1} L^T_{s:e,1:s-1}$ {Use DSYRK}
  3. $L_{s:e,s:e} L^T_{s:e,s:e} = A_{s:e,s:e}$ {Use level-2 routine}

A is factorized one block row at a time. In each iteration all previous factorized block rows are accessed in the triangular solve. Rows below the current block row remain untouched until they become the current block row to factorize.

3.3 Block QR Algorithms

The QR factorization of an $m \times n$ matrix A is given by

$$A = QR,$$

where the $m \times m$ matrix Q is orthogonal and the $m \times n$ matrix R is upper triangular (trapezoidal). The QR factorization can be computed in several ways, e.g. by utilizing Householder transformations or Givens transformations [16]. Here we only consider methods based on Householder transformations

$$H = H(v) = I - 2vv^T, \|v\|_2 = 1.$$

The orthogonal Q is computed as a product of Householder transformations

$$Q = H_1 H_2 \cdots H_n$$

such that $Q^T A = R$ and H_i annihilates the elements below the main diagonal of the ith column of the transformed matrix A. To simplify the notation we assume that $m \geq n$, and we can describe the Householder QR algorithm as follows:

For $i = 1 : n$
1. Generate Householder vector v_i from $A_{i:m,i}$
2. $w \leftarrow A^T_{i:m,i:n} v_i$
3. $A_{i:m,i:n} \leftarrow A_{i:m,i:n} - 2v_i w^T$

The application of a Householder transformation involves a matrix-vector multiply (operation 2) and a rank-1 update (operation 3) which both are level-2 BLAS operations. It is possible to combine several Householder transformations into a block Householder transformation and thereby obtain a level-3 formulation of the Householder QR algorithm [6]. Let Q_k be the product of k Householder transformations

$$Q_k = H_1 H_2 \cdots H_k.$$

From the structure of the Householder vectors v_i it follows that Q_k is a rank-k modification of the identity matrix and can be expressed as

$$Q_k = I - YSY^T$$

where Y is an $m \times k$ matrix and S is a $k \times k$ upper triangular matrix [28]. This block form is called the *compact WY representation*. The generation of Y and S require $\mathcal{O}(mk^2)$ extra flops and $k(k+1)/2$ extra storage. For a detailed description see [28, 4].

Let nb be the block column width and assume that n is an integer multiple of nb. Then a level-3 algorithm utilizing the compact WY representation can be described as follows:

For $i = 1 : n/nb$
  $s = (i-1) \cdot nb + 1$ {Start of block to factorize}

$e = min(s + nb - 1, n)$	{End of block to factorize}
1. $[S, Y] \leftarrow \mathrm{hshbg}(A_{s:m,s:e})$	{Generate S and Y}
2a. $W \leftarrow A^T_{s:m,e+1:n}Y$	{Use `DGEMM` }
2b. $W \leftarrow WS$	{Use `DTRMM` }
3. $A_{s:m,e+1:n} \leftarrow A_{s:m,e+1:n} - YW^T$	{Use `DGEMM` }

The level-3 formulation performs two major steps: level-2 QR factorization of a block column (operation 1), and updating of the remaining block columns (operations 2 and 3) which can be performed by level-3 BLAS operations. The generation of S and Y in operation 1 also involves the application of the block Householder transformation to $A_{s:m,s:e}$. It is the presumptive high performance of the level-3 operations that pays off the extra work ($\mathcal{O}(m \cdot n \cdot nb)$ flops) performed by the block algorithm compared to the level-2 algorithm. A is overwritten by R and the Householder vectors Y.

In the terminology left-looking and right-looking the described block algorithm is right-looking. A block left-looking algorithm would delay the application of computed block Householder transformations until a block column becomes the next to factorize.

3.4 Uniprocessor Performance and Choice of Algorithms

Here we summarize some performance results of the block algorithms for different block sizes and problem sizes. All results presented are obtained from IBM 3090 VF (J model, one processor). The codes used here are from the second prerelease of LAPACK [2]. We also discuss the choice of block algorithms for further parallelization. See also [3] for an evaluation of block algorithms in LAPACK.

3.4.1 Block LU Algorithms

Tables 3–5 show the performance in Mflops for the block right-looking, block left-looking, and block Crout variants of the LU factorization with partial pivoting. Benchmarks of the three different level-2 algorithms show that the left-looking and Crout variants have the best and similar performance (right-looking only 60% of their performance). We chose the left-looking level-2 algorithm in all block variants. The numbers in boldface show the best performance for a given problem size. The overall best block size is $nb = 48$. As we can see from the results the block right-looking algorithm gives the best results, and it is the routine that we will work on further and explicitly parallelize (see Section 4).

The block right-looking algorithm is more `DGEMM`-intensive than the block left looking algorithm and it performs the `DGEMM` operations on larger submatrices than the block Crout algorithm. The last column of Table 3 shows the performance of the block LU algorithm in the ESSL library [17]. As we can see the

Table 3: Performance of block right-looking LU.

Right	nb					ESSL
$m = n$	16	32	48	64	96	$m = n$
100	**32.2**	30.2	24.9	22.8	17.9	49.0
200	**57.9**	55.4	52.0	45.9	41.7	71.2
300	63.5	**70.6**	65.0	61.6	59.9	81.2
400	73.9	77.4	**77.9**	70.3	69.8	85.3
500	77.4	82.2	**83.7**	77.1	75.8	87.3
1000	86.2	93.7	**95.1**	89.6	91.1	97.6

Table 4: Performance of block left-looking LU.

Left	nb				
$m = n$	16	32	48	64	96
100	**25.4**	24.4	23.7	17.7	17.7
200	43.0	**46.7**	46.5	40.4	40.4
300	53.5	59.2	**61.4**	57.4	57.4
400	60.5	66.4	**69.2**	68.6	67.4
500	63.9	73.1	74.4	**75.5**	73.3
1000	77.0	84.6	87.4	87.4	**87.5**

Table 5: Performance of block Crout LU.

Crout	nb				
$m = n$	16	32	48	64	96
100	**29.2**	26.8	24.5	22.2	17.7
200	**52.1**	51.9	49.0	44.2	41.5
300	60.3	**65.0**	63.8	58.7	58.0
400	67.7	**71.2**	70.3	65.4	66.8
500	71.8	**77.0**	76.6	70.5	72.9
1000	81.2	86.9	**88.6**	79.4	84.9

performance of the right-looking algorithm implemented in Fortran reaches the performance of the hand-tuned ESSL routine for large matrices. Notice that the ESSL routine does not perform all permutations explicitly.

3.4.2 Block Cholesky Algorithms

Tables 6–8 show the performance in Mflops for the block right looking, block left-looking and block top-looking variants of the *Cholesky* factorization. We have chosen the left-looking variant as the level-2 routine in all block variants. The numbers in boldface show the best performance for a given problem size. There is no single best block size nb. If $nb > n$ the block size n is used.

Here all three algorithms give very similar results for appropriate block sizes. We chose to explicitly parallelize the right-looking algorithm (see Section 4). Notice that for $n = 100, 200$ the level-2 algorithm gives the best performance for all three variants. The last column of Table 6 shows the performance of the block LL^T algorithm in the ESSL library [17] which works on A in packed data format (the triangular part is packed sequentially column by column in a one-dimensional array that offers better data reuse).

Table 6: Performance of block right-looking Cholesky.

Right	nb									ESSL
n	32	64	96	128	160	192	224	256	288	n
100	30.7	32.3	29.1	**44.9**						39.0
200	55.4	55.0	53.7	55.8	52.4	50.1	**70.3**			74.0
300	65.7	68.8	67.5	67.8	**69.6**	68.4	66.1	64.9	62.6	84.1
400	70.6	70.0	71.2	68.7	76.0	73.8	**77.7**	77.1	74.9	88.2
500	76.3	76.9	76.3	78.3	75.5	74.2	72.0	**84.1**	82.8	90.8
1000	85.1	85.6	85.2	86.8	84.3	84.6	82.4	**89.0**	87.6	96.5

3.4.3 Block QR Algorithms

In Table 9 we display some performance results of the block right looking QR algorithm for different block sizes nb. Notice that the Mflops numbers are based on the number of flops of the level-2 routine as suggested in [2]. Presently, IBM does not provide any block QR routine in ESSL.

In summary, for large matrices the right-looking variants give up to 90% of the best `DGEMM` performance (the practical peak performance on one processor).

Table 7: Performance of block left-looking Cholesky.

Left	nb								
n	32	64	96	128	160	192	224	256	288
100	26.9	31.7	28.3	**44.7**					
200	46.3	49.5	51.3	55.7	52.1	50.0	**71.0**		
300	57.5	60.8	63.0	64.8	**69.5**	67.8	66.0	64.5	63.1
400	64.4	67.6	69.6	70.7	70.9	72.8	**77.3**	75.9	75.1
500	69.7	73.4	74.8	78.0	76.2	76.2	75.7	**84.0**	81.9
1000	81.5	84.7	86.3	87.4	87.0	87.7	85.7	**90.4**	84.4

Table 8: Performance of block top-looking Cholesky.

Top	nb								
n	32	64	96	128	160	192	224	256	288
100	21.6	31.9	23.1	**44.1**					
200	32.4	40.3	44.4	55.9	53.2	49.8	**71.6**		
300	37.4	52.1	52.3	59.7	**70.2**	68.2	66.4	64.1	62.9
400	40.6	52.3	57.8	61.6	68.9	64.7	**77.9**	76.3	74.9
500	42.7	59.3	61.8	74.0	67.0	76.1	73.1	**84.0**	82.8
1000	44.5	62.5	70.4	78.6	77.7	79.6	83.8	**89.1**	80.0

Table 9: Performance of block right-looking QR.

Right	nb				
$m = n$	16	32	48	64	96
100	**35.8**	30.5	24.9	23.7	18.1
200	**61.2**	56.8	51.3	45.0	39.0
300	**72.6**	71.4	66.2	60.9	53.4
400	**77.8**	76.6	73.2	69.3	61.7
500	80.6	**82.7**	79.0	74.8	66.9
1000	86.4	**91.4**	88.4	86.9	81.7

4 Parallel Block Matrix Algorithms

Parallelism in block algorithms can be invoked in several ways, and we distinguish between *implicit* and *explicit* parallelism. The easiest way to introduce parallelism is to replace calls to uniprocessor level-3 kernels by calls to parallel level-3 kernels. Since the parallelism is invoked inside (within) the level-3 kernels we define this type of block algorithms as *implicitly parallel*. Provided that efficient parallel level-3 kernels exist this approach makes it easy to write portable programs for parallel architectures and this approach is highlighted in the LAPACK project [1]. However, the implicit parallelism imposes a synchronization after each level-3 operation, which means that idle processors must wait until all processors have completed their share of work. Of course, this may lead to an unnecessary degradation of performance. The amount of degradation is determined by the efficiency of the parallel level-3 kernels and the data dependency of the block algorithm.

The synchronization discussed above is not algorithm inherent and can therefore be overcome by invoking the parallelism at the block level in the algorithm. Then processors can work on independent block operations, which e.g. makes it possible to overlap and pipeline different matrix-matrix operations. Further, only efficient uniprocessor level-3 kernels are required. We define this type of block algorithm as *explicitly parallel*. Here the parallelism must be invoked by using parallel language constructs. The proposal for standardization of such constructs in Fortran by the *Parallel Computing Forum (PCF)* [25] will make it possible to write portable code too. The constructs of IBM Parallel Fortran [29] that we are using are functionally equivalent to the PCF proposal (see Section 7). It is, of course, also possible to consider a hybrid of implicit and explicit parallelism in block algorithms.

The number of synchronization points of an explicit parallel block algorithm is in theory only determined by the data dependency at the block level and can therefore be chosen minimal. Further, the explicit approach makes it easy to balance the work over available processors and it is possible to use either *static* or *dynamic* scheduling of the work.

In the following subsections we describe the parallel block algorithms that are implemented. Similar variants of some of these algorithms are discussed in [8] (LU factorization) and [14, 26]. For each matrix factorization we have chosen to implicitly parallelize the block algorithms that show the "best" uniprocessor performance (see Section 3.4). This choice is based on the assumption that the LAPACK project will make the choice of block algorithms from this criterion. The parallel level-3 kernels used in these algorithms have been described in Section 2.2. Therefore, we concentrate our description on the explicitly parallel block algorithms for the matrix factorizations discussed in Section 3. Each algorithm is presented as a *node program* that each processor $me = 0 : P - 1$ will execute. Individual processors will work on different parts of A and perform sometimes different operations. For example, *processor* 0 will always compute the level-2 factorizations when the parallelizable work is statically scheduled (see also sections 5 and 6). The statement $[u,v]$ = My_block(me, P, $size$, nb) determines the start position u and the end position v of the columns that processor me will work on at this step. (Here $size$ denotes the current problem size.)

4.1 Parallel Block LU Algorithms

All three block LU algorithms have been implicitly parallelized by using the parallel level-3 kernels `PDTRSM` and `PDGEMM`. The *block right-looking* algorithm has been explicitly parallelized on the basis of data dependency at the block level. It is obvious that the level-2 factorization of a block column $A_{s:m,s:e}$ must be completed before the update with respect to this block column can take place. In the following node program there is only one synchronization needed in each iteration.

If $me = 0$
  1. $L_{1:m,1:nb}U_{1:nb,1:nb} = A_{1:m,1:nb}$ {Use level-2 routine}
For $i = 2 : min(n/nb, m/nb)$
  2. Wait for all processors {Synchronization}
  $s = (i-2) \cdot nb + 1$ {Start of last factorization}
  $e = (i-1) \cdot nb$ {End of last factorization}
  3. $[u, v]$ = My_block(me, P, $n-e$, nb) {Get block indices for me}
  If $u \leq n$
    4. Apply pivotings to $A_{s:e,u:v}$ {From all factorizations}
    5. $U_{s:e,u:v} \leftarrow L^{-1}_{s:e,s:e}A_{s:e,u:v}$ {Use `DTRSM`}
    If $e < m$
      6. $A_{e+1:m,u:v} \leftarrow A_{e+1:m,u:v} - L_{e+1:m,s:e}U_{s:e,u:v}$ {Use `DGEMM`}
  If $me = 0$
    7. $L_{u:m,u:e+nb}U_{u:e+nb,u:e+nb} = A_{u:m,u:e+nb}$ {Use level-2 routine}
8. $[s, e]$ = Next_columns(nb) {Get indices for next block}
While $s < n - nb$
  9. Apply pivotings to $A_{e+1:m,s:e}$ {From all factorizations}
  10. $[s, e]$ = Next_columns(nb) {Get indices for next block}

In each step i the number of columns of the remaining matrix ($n - e$ = current problem size) is partitioned evenly over the processors with the exception that processor 0 is always given the leftmost block in the remaining matrix, a block with at least nb columns (the next block to factorize). The work is shared so that each processor computes the part of U (operation 5) that is required for updating the remaining rows of its part of A (operation 6). This excludes any synchronization between the triangular solve and the matrix update. Furthermore, processor 0 can start the factorization of the next nb columns as soon as it has completed its part of the update of A.

The pivotings that, so far, have not been applied to the left-hand side of the current block columns are performed after all block columns are factorized. This is done in parallel so that each processor applies all row interchanges to a block consisting of nb columns. The routine $Next\_columns$ returns the start position s and end position e for the next column block where processor me will do the row interchanges.

It is clear that processor 0 will do more work than the other processors. To obtain a better load balancing, the processor that performs the level-2 factorization must have less work in the triangular solve and the matrix update (operations 5 and 6) than the other processors. We will discuss this trade-off in Section 5.

4.2 Parallel Block Cholesky Algorithms

The *block right-looking* algorithm has been implicitly and explicitly parallelized. The implicit parallel version makes use of the parallel level-3 kernels `PDSYRK`, `PDGEMM` (only the left-looking variant), and `PDTRSM`. The explicit parallel version is outlined below. To invoke parallelism and possibly also achieve better performance on one processor we have made some changes to the right-looking algorithm given in Section 3.2.1. Rather than using only `DSYRK` to do the symmetric rank-nb update of the $(n-e)\times(n-e)$ submatrix, we partition it in several smaller blocks, and use a number of calls to `DSYRK` for updating $nbb \times nbb$ diagonal blocks and the off-diagonal blocks are updated using `DGEMM`. To simplify the notation we assume that nbb is an integer multiple of $n-e$ which gives us $(n-e)/nbb$ diagonal blocks. The rank-nb update

3. $A_{u:n,u:n} \leftarrow A_{u:n,u:n} - L_{u:n,s:e}L^T_{u:n,s:e}$ {Use `DSYRK` to update}

can now be described as follows:

For $ii = 1 : (n-e)/nbb$
  $ss = (ii-1)\cdot nbb + u$ {Start of iith diagonal block}
  $ee = ss + nbb - 1$ {End of iith diagonal block}
  $uu = ee + 1$ {Start of `DGEMM`-update}
  3.1 $A_{ss:ee,ss:ee} \leftarrow A_{ss:ee,ss:ee} - L_{ss:ee,s:e}L^T_{ss:ee,s:e}$ {Use `DSYRK`}
  If $ee < n$
    3.2 $A_{uu:n,ss:ee} \leftarrow A_{uu:n,ss:ee} - L_{uu:n,s:e}L^T_{ss:ee,s:e}$ {Use `DGEMM`}

The block size nbb in the update loop can be chosen different from the block size nb at the outer level. For one processor we may choose nbb to get as much work as possible in DGEMM and still have proper speed in DGEMM and DSYRK. Since each single call to a level-3 BLAS in the update loop writes to different parts of the matrix A, all level-3 operations can be executed in parallel. As soon as the first nb columns of the $(n-e) \times (n-e)$ submatrix is updated the next block column can be factorized by the level-2 routine. As before, processor 0 performs the level-2 factorizations. The explicit parallel version of the block right-looking algorithm requires only one synchronization per $(n-e) \times nb$ block column and is described in the following node program.

If $me = 0$
  1. $L_{1:nb,1:nb} L^T_{1:nb,1:nb} = A_{1:nb,1:nb}$ {Use level-2 routine}
  If $nb < n$
    2. $L_{nb+1:n,1:nb} \leftarrow A_{nb+1:n,1:nb} L^{-T}_{1:nb,1:nb}$ {Use DTRSM}
For $i = 2 : n/nb$
  3. Wait for all processors {Synchronization}
  $s = (i-2) \cdot nb + 1$ {Start of last factorization}
  $e = (i-1) \cdot nb$ {End of last factorization}
  $u = e + 1$ {Start of next factorization}
  $v = i \cdot nb$ {End of next factorization}
  If $me = 0$
    4. $A_{u:v,u:v} \leftarrow A_{u:v,u:v} - L_{u:v,s:e} L^T_{u:v,s:e}$ {Use DSYRK}
    5. $L_{u:v,u:v} L^T_{u:v,u:v} = A_{u:v,u:v}$ {Use level-2 routine}
    If $v < n$
      6. $A_{v+1:n,u:v} \leftarrow A_{v+1:n,u:v} - A_{v+1:n,s:e} A^T_{u:v,s:e}$ {Use DGEMM}
      7. $L_{v+1:n,u:v} \leftarrow A_{v+1:n,u:v} L^{-T}_{u:v,u:v}$ {Use DTRSM}
  8. $[Syrk,\ Gemm,\ NoSyrk,\ NoGemm] =$
      $\mathrm{My\_Block}(me,\ P,\ n-e,\ nbb)$
  For $k = 1 : NoSyrk$ {Update diagonal blocks}
    $l = 2k - 1,\ ss = Syrk(l),\ ee = Syrk(l+1)$
    9. $A_{ss:ee,ss:ee} \leftarrow A_{ss:ee,ss:ee} - L_{ss:ee,s:e} L^T_{ss:ee,s:e}$ {Use DSYRK}
  For $k = 1 : NoGemm$ {Update off-diagonal blocks}
    $l = 3k - 2,\ ss = Gemm(l)$
    $ee = Gemm(l+1),\ uu = Gemm(l+2)$
    10. $A_{uu:n,ss:ee} \leftarrow A_{uu:n,ss:ee} - L_{uu:n,s:e} L^T_{ss:ee,s:e}$ {Use DGEMM}

In each step i the rank-nb update of the remaining $(n-e) \times (n-e)$ matrix is partitioned over the P processors. The number of diagonal and off-diagonal blocks processor me will update in the inner blocking of step i is computed in the routine *My_Block* (operation 8). *NoSyrk* returns the number of $nbb \times nbb$ diagonal blocks and the index vector *Syrk* holds the start and end positions for these (ss and ee). Similarly, *NoGemm* returns the number of off-diagonal blocks and the index vector *Gemm* holds the indices for each DGEMM-update (ss, ee, and uu). Notice that the off-diagonal updates are of different size, and we can accommodate this imbalance in the index vector *Gemm*. We will discuss this trade-off in more detail in Section 5. It is also possible to implicitly parallelize operation 2 by calling `PDTRSM` giving a hybrid version.

4.3 Parallel Block QR Algorithms

The *block right-looking* algorithm has been implicitly and explicitly parallelized. The implicit parallel version makes use of the parallel level-3 kernels `PDGEMM` and `PDTRMM`. The explicit parallel version is outlined in the following node program.

For $i = 1 : n/nb$	
$s = (i-1)nb + 1$	{Start of current block}
$e = min(s + nb - 1, n)$	{End of current block}
If $me = 0$	
1. $[S, Y] \leftarrow \text{hshbg}(A_{s:m,s:e})$	{Generate S and Y}
2. $[u, v] = \text{My\_block}(me, P, n-e, nb)$	
3. Wait for all processors	{Synchronization}
4.a $W \leftarrow A^T_{s:m,u:v} Y$	{Use `DGEMM`}
4.b $W \leftarrow WS$	{Use `DTRMM`}
5. $A_{s:m,u:v} \leftarrow A_{s:m,u:v} - YW^T$	{Use `DGEMM`}

When a block Householder transformation is generated (operation 1) and Y and S are to be applied to the rest of the matrix, different block columns of the matrix can be updated in parallel without any synchronization between the two `DGEMM` operations and the `DTRMM`. This is possible since each processor can execute operations 4.a, 4.b, and 5 without any interaction with the other processors. If in iteration i, u and v for processor 0 are chosen so that u equals s for iteration $i+1$ and v of iteration $i+1$ is at least equal to e, processor 0 can also generate the Householder transformation for iteration $i+1$ before any synchronization is required. In the algorithm described s and e are the first and last column of the current block to be factorized; u and v (operation 2) hold the first and last column, respectively, of the block column to be updated by processor me. For simplicity we assume that all processors are given equal-sized blocks, which implies that processor 0 will do more work than the other processors. To obtain a better load balancing, the processor that performs the level-2 factorization must have less work in the matrix update (operations 4.a, 4.b, and 5) than the other processors. This trade-off is discussed below.

5 Explicit Parallelization and Static Load Balancing

Since processor 0 performs all level-2 factorizations in the explicitly parallel block algorithms described in sections 4.1 through 4.3 it will in general do more work than processors $1 : P-1$. As mentioned earlier, to obtain a balanced load processor 0 must have less work in the level-3 operations that perform the update of the remaining matrix in each step i. This imbalance can be cured by having a cost function incorporated in the routine *My_Block* (defined in Section 4) that determines the computational work that is performed in each step at the block level. This information is then utilized to determine the start position u and the end position v of the columns that each processor me $(= 0 : P-1)$ will work on at the next step. The cost function that is based on *the number of flops measured (expressed) in GEMM equivalents* is described below.

Table 10 shows the percentage cost in flops and real execution time of different suboperations in the block right-looking algorithms ($m = n$, $nb = 48$ for LU, $nb = 32$ and $nbb = 128$ for LL^T, $nb = 32$ for QR).

Table 10: Percentage cost of suboperations in block algorithms.

Matrix size	$m = n = 800$		$m = n = 1200$	
Suboperation	*Flops*	*Time*	*Flops*	*Time*
Level-2 LU	4.6	8.6	3.0	5.8
Pivotings_R	0.1	9.0	0.1	6.4
`DTRSM`	4.2	5.4	2.9	3.9
`DGEMM`	91.0	74.7	93.9	82.3
Pivotings_L	0.1	2.3	0.1	1.6
Level-2 LL^T	0.2	1.3	0.1	0.6
`DTRSM`	5.8	7.9	3.8	4.9
`DSYRK`	20.6	27.6	14.5	20.6
`DGEMM`	73.4	63.2	81.6	73.9
Level-2 QR	2.9	6.0	2.0	4.1
Generate S	1.5	2.2	0.9	1.4
`DGEMM` no. 1	47.2	45.6	48.1	47.9
`DTRMM`	1.4	1.8	0.9	1.2
`DGEMM` no. 2	47.2	44.4	48.1	45.4

In the subtable for the block LU algorithm Pivotings_R corresponds to operation 4 in Section 4.1. Further, Pivotings_L corresponds to operations 8–10 in Section 4.1. Different suboperations perform with different speed (performance), which means, for example, that the level-2 factorizations and permutations consume more time than their percentage share of the total number of flops. The level-2 operations cost relatively more when the matrix dimension decreases. This is illustrated in Table 11, which displays the real execution time (in seconds) of different suboperations in each iteration of the block right-looking *LU* algorithm ($m = n = 800$, $nb = 48$). The percentage cost for the level-2 factorization increases from about 5% in the second iteration to 43% in the last iteration. Similarly, the percentage cost for the `DGEMM` operation decreases from about 84% in the second iteration to about 20% in the last iteration. However, the first iterations contribute most to the total execution time.

Table 11: Execution time of suboperations in block *LU*.

Itera -tion	Matrix size	Level-2 *LU* *Secs*	Pivotings_R *Secs*	`DGEMM` *Secs*	`DTRSM` *Secs*
1	800	0.032	0.	0.	0.
2	752	0.030	0.025	0.505	0.043
3	704	0.029	0.023	0.447	0.040
4	656	0.027	0.022	0.397	0.039
5	608	0.027	0.021	0.343	0.036
6	560	0.025	0.019	0.290	0.034
7	512	0.023	0.017	0.236	0.032
8	464	0.022	0.016	0.199	0.026
9	416	0.020	0.014	0.161	0.023
10	368	0.020	0.012	0.130	0.020
12	320	0.018	0.012	0.101	0.018
13	272	0.016	0.010	0.069	0.015
14	224	0.015	0.007	0.047	0.012
15	176	0.013	0.006	0.030	0.009
16	128	0.012	0.005	0.017	0.007
17	80	0.011	0.003	0.008	0.004
18	32	0.004	0.002	0.002	0.002

In Table 12 the operation counts (multiplications and adds) for the different suboperations in each block iteration of the block right-looking *LU* algorithm are shown together with weights $l_i \geq 1.0$ computed as

$$l_1 = \frac{Mflops\_GEMM}{Mflops\_LU\_2}; \qquad l_2 = \frac{Mflops\_GEMM}{Mflops\_SWAP};$$

$$l_3 = \frac{Mflops\_GEMM}{Mflops\_TRSM}; \qquad l_4 = 1.0;$$

Here, *SWAP* corresponds to the permutations performed in each block iteration (i.e., Pivotings_R in Table 10). The performance in *Mflops* of individual suboperations is measured on one processor. *Mflops_GEMM* expresses the performance of `DGEMM` for the problem size of the corresponding suboperation. The weights are experimentally chosen and represent a weighted average of the performance of the different block iterations. Now, the cost function for the block LU algorithm is expressed as

$$GEMM_{flops}(LU) = l_1 \cdot LU\_2 + l_2 \cdot SWAP + l_3 \cdot TRSM + l_4 \cdot GEMM$$

Similar cost functions for the block right-looking LL^T and QR algorithms are given below:

$$GEMM_{flops}(Chol) = c_1 \cdot Chol\_2 + c_2 \cdot TRSM + c_3 \cdot SYRK + c_4 \cdot GEMM$$

and

$$GEMM_{flops}(QR) = q_1 \cdot QR\_2 + q_2 \cdot Gen\_S + q_3 \cdot TRMM + q_4 \cdot GEMM$$

The corresponding operation counts and weights are displayed in tables 13 and 14. In Table 13 $k = div(n - e - nb, nbb)$ and $r = mod(n - e - nb, nbb)$.

6 Explicit Parallelization and Dynamic Load Balancing

The static mapping of work on processors as described in Section 5 will work well on a dedicated system in which each scheduled task is guaranteed a physical processor. However, if the program is executing in a multi-user environment, the parallel performance may degrade since other users will compete for the resources of the multiprocessor system (e.g. processors and shared memory). When a system cannot be dedicated *dynamic scheduling* of the work to processors is likely to be a better approach. The idea is to split the parallelizable work into a *queue of tasks*, and an idle processor is scheduled the next available task in the queue. In the following subsections we describe two block matrix factorization algorithms (block right-looking LU and LL^T) with dynamic scheduling of tasks.

6.1 Parallel Block LU Algorithm with Dynamic Scheduling

The parallelizable work in the block right-looking LU algorithm (i.e., columns to be updated in each step) is partitioned into $2 \times P$ tasks (P = number of processors to be used). A larger number of tasks (with fewer columns per block) would more likely prevent idle processors. However, it would also imply a possible degradation of the performance of `DTRSM` and `DGEMM`. Our choice of $2 \times P$ tasks in the queue is based on benchmark results with different sizes of task queue. In the following node program processor me is scheduled the next task by a call to *Next_block*.

Table 12: Operation counts and weights of LU-cost function.

Suboperation	Operation counts	i	l_i
LU_2	$(m-e)nb^2 - \frac{1}{3}nb^3 - \frac{1}{2}nb^2 + \frac{5}{6}nb$	1	1.92
$SWAP$	$(n-e)nb$	2	120.
$TRSM$	$(n-e)nb^2$	3	1.62
$GEMM$	$2(m-e)(n-e)nb$	4	1.0

Table 13: Operation counts and weights of LL^T-cost function.

Suboperation	Operation counts	i	c_i
$Chol\_2$	$\frac{1}{3}nb^3 + \frac{1}{2}nb^2 + \frac{1}{6}nb$	1	9.6
$TRSM$	$(n-e-nb)nb^2$	2	1.45
$SYRK$	$nb^2(nb+1) + k \cdot nb \cdot nbb(nbb+1) + nb \cdot r(r+1)$	3	1.45
$GEMM$	$2(n-e-nb)nb^2 + k(k \cdot nbb - nbb + 2r)nbb \cdot nb$	4	1.0

Table 14: Operation counts and weights of QR-cost function.

Suboperation	Operation counts	i	q_i
QR_2	$2(m-e)nb^2 - \frac{2}{3}nb^3 + (m-e)nb + nb^2 + \frac{14}{3}nb$	1	1.85
Gen_S	$(m-e)nb^2 - \frac{1}{3}nb^3 + \frac{1}{3}nb$	2	1.85
$TRMM$	$(n-e)nb^2$	3	1.3
$GEMM$	$4m(n-e)nb$	4	1.0

If $me = 0$
  1. $L_{1:m,1:nb}U_{1:nb,1:nb} = A_{1:m,1:nb}$ {Use level-2 routine}
$[s, e, u, v] = \text{Next\_block}(me, P, n-e, nb)$ {Next task from queue}
While $e < n$
  If $u \leq n$
    2. Wait until my block is free {Synchronization}
    3. Pivotings are applied to $A_{s:e,u:v}$ {From all factorizations}
    4. $U_{s:e,u:v} \leftarrow L^{-1}_{s:e,s:e}A_{s:e,u:v}$ {Use `DTRSM`}
    If $e < m$
      5. $A_{e+1:m,u:v} \leftarrow A_{e+1:m,u:v} - L_{e+1:m,s:e}U_{s:e,u:v}$ {Use `DGEMM`}
  If *Next factorization in my block*
    6. $L_{u:m,u:e+nb}U_{u:e+nb,u:e+nb} = A_{u:m,u:e+nb}$ {Use level-2 routine}
  $[s, e, u, v] = \text{Next\_block}(me, P, n-e, nb)$ {Next task from queue}
$[s, e] = \text{Next\_columns}(nb)$
While $s < n - nb$
  7. Pivotings are applied to $A_{s+nb:m,s:e}$ {From all factorizations}
  $[s, e] = \text{Next\_columns}(nb)$

The monitor routine *Next_block* returns necessary information for the next task; u and v hold the start position and the end position, respectively, of the columns for the next task; s and e hold the first and last column, respectively, of the last factorized block column. The queue of tasks is not empty as long as $e < n$. When the factorization is complete the remaining pivoting work is set up as a new queue of tasks. The routine *Next_columns* works as described in Section 4.1 and is implemented as a `PARALLEL LOOP` (i.e., with implicit dynamic scheduling of the work [29]).

The algorithm described above does not require the same kind of synchronization as the statically scheduled algorithm. Here different processors can be working with operations belonging to two different block iterations i at the same time. The synchronization (operation 2) is required to ensure that the submatrices to be used in the triangular solve and in the matrix update are computed and that no other processor is working on my column block.

6.2 Parallel Block Cholesky Algorithm with Dynamic Scheduling

The columns of A to be updated in each step (i.e., the parallelizable work of the block right-looking LL^T algorithm) are partitioned into column blocks consisting of nbb columns (inner level blocking). To start we do not put any restriction on nbb (can be equal to, smaller than or larger than nb, the outer level block size). Each processor takes a block column (with nbb columns) and updates it using `DSYRK` for the triangular diagonal block and `DGEMM` for the rest of the block column. When a processor is finished with that task it continues to update a new block column (the next task in the queue). The next level-2 factorization is performed by the processor that updated (the last referenced columns of) the first nb columns of the $(n-e)\times(n-e)$ submatrix. If a processor reaches the last column n of the matrix it starts to update the first column block of the new $(n-e)\times(n-e)$ submatrix. This goes on until the last nb columns of A have been factorized.

In the following node program the dynamic scheduling of tasks is controlled by the global variables *s_global* (start position of the last level-2 factorization) and *u_global* (start position for the next update of nbb columns). It contains a critical section (Lock ... Unlock) in which these variables are updated.

If $me = 0$	
1. $L_{1:nb,1:nb}L^T_{1:nb,1:nb} = A_{1:nb,1:nb}$	{Use level-2 routine}
If $nb < n$	
2. $L_{nb+1:n,1:nb} \leftarrow A_{nb+1:n,1:nb}L^{-T}_{1:nb,1:nb}$	{Use DTRSM}
3. $s\_global = 1$	{Start of last factorization}
4. $u\_global = nb + 1$	{Start of next update}
$e = nb$	{End of last factorization}
5. Wait for all processors	{Synchronization}
While $e < n$	
Lock	{Start of critical section}
If $u\_global > n$	
6. $s\_global = s\_global + nb$	
7. $u\_global = s\_global + nb$	
$s = s\_global$	
$u = u\_global$	{Start of my update}
$my\_nbb = \min(nbb, n - u + 1)$	{Size of last block to update}
8. $u\_global = u\_global + my\_nbb$	
Unlock	{End of critical section}
$v = u + my\_nbb - 1$	{End of my update}
$e = s + nb - 1$	
If $e < n$	
9. While Not($Free$)	
Wait until $Free$ = True	{Synchronization}
10. $A_{u:v,u:v} \leftarrow A_{u:v,u:v} - L_{u:v,s:e}L^T_{u:v,s:e}$	{Use DSYRK}
If $v < n$	
11. $A_{v+1:n,u:v} \leftarrow A_{v+1:n,u:v} - L_{v+1:n,s:e}L^T_{u:v,s:e}$	{Use DGEMM}
If $Factor$	
$ss = s + nb$	{Start of next factor}
$ee = e + nb$	{End of next factor}
12. $L_{ss:ee,ss:ee}L^T_{ss:ee,ss:ee} = A_{ss:ee,ss:ee}$	{Use level-2 routine}
If $ee < n$	
13. $L_{ee+1:n,ss:ee} \leftarrow A_{ee+1:n,ss:ee}L^{-T}_{ss:ee,ss:ee}$	{Use DTRSM}

The logical variable $Free$ is true if no other processor is working in the same area as processor me wants to write into. If two processors compete for writing in the same memory area the one that reads from the leftmost area starts (operations 10 and 11). This guarantees that the updates will be performed in correct order. The logical variable $Factor$ is true if the first nb columns of the submatrix have been updated, i.e., it is ready to be factorized. Notice that the two level-2 factorizations (last and next) can be completed before the updates with respect to the first (i.e., last) factorization is completed. In this way we obtain a pipeline between two consecutive iterations at the block level.

The best parallel efficiency is obtained by choosing nbb such that $\frac{n-e}{nbb} \gg P$. However, the choice of nbb must also be balanced so that we obtain good performance of the level-3 operations 10 and 11 on individual processors. As for the static scheduled algorithm operation 2 can be implicitly parallelized by calling `PDTRSM`.

7 Performance Results of Parallel Block Algorithms

In this section we present performance results of the parallel block algorithms described in sections 4 through 6 and implemented in IBM Parallel Fortran [29]. The parallel language extensions used are summarized below: `ORIGINATE` to create tasks, `TERMINATE` to terminate tasks, `SCHEDULE` or `DISPATCH` to assign work to a task (makes it possible to execute independent subroutines in parallel), `WAIT FOR` to detect when a task has completed the assigned work, `PARALLEL LOOP` to execute iterations of a loop concurrently, and routines from the parallel library that handles locks and events are used to implement different synchronization techniques. Notice that subroutine calls are not permitted within a `PARALLEL LOOP` in IBM Parallel Fortran. For more details of the parallel language extensions see [29] (similar parallel constructs are included in the new release 2.5 of VS FORTRAN). Code that does not make use of parallel language constructs is compiled with the IBM compiler VS FORTRAN 2.4, which at present has better optimization for vectorization [23]. All results presented are obtained using up to five physical processors of a quasi-dedicated IBM 3090 VF/600J. The SDCN (Supercomputer Centre North) machine is located at Norrdata, Skellefteå, in Sweden. The uniprocessor performance results presented in Section 3.4 are based on measured CPU times (IBM Fortran routine `CPUTIME` is used). All results for parallel routines presented in this section are based on wall-clock timing (IBM Fortran routine `CLOCKX` is used) at night. The results for a specific parallel block algorithm are the best performance obtained from a series of tests. Performance results, measured in Mflops, and speedup factors are presented for implicitly parallel and explicitly parallel block algorithms. The theoretical peak performance of IBM 3090 VF/600J for 64 bits (= 1 word) arithmetic is 828

$(6 \cdot 138)$ Mflops while its practical peak performance is around 600 Mflops (500 Mflops on 5 processors). The best result we have obtained on five processors is 482 Mflops (PDGEMM, see Table 2).

7.1 Parallel Block LU Algorithms

Table 15 shows performance results of the block right-looking and Crout variants of the LU factorization with partial pivoting. These two block algorithms showed the best uniprocessor performance of the block LU variants (see Section 3.4). The main change we have made to the prerelease LAPACK [2] routines, is to replace calls to level-3 BLAS by calls to the parallel level-3 kernels described in Section 2.2. We have also modified the routine DLASWP that performs permutations to a block, so that they are performed columnwise as suggested in [2].

Table 15: Performance of implicitly parallel block LU algorithms.

Block algorithm	*Right*		*Crout*	
Matrix size	500	1200	500	1200
1 proc	83.7	95.8	72.7	90.0
2 proc	104.4	167.4	90.2	138.7
Speedup	*1.25*	*1.75*	*1.24*	*1.54*
3 proc	112.0	217.4	94.4	169.5
Speedup	*1.34*	*2.27*	*1.30*	*1.88*
4 proc	110.7	231.2	92.4	185.3
Speedup	*1.32*	*2.41*	*1.27*	*2.06*
5 proc	104.5	253.1	89.2	191.5
Speedup	*1.25*	*2.64*	*1.23*	*2.13*

Table 16 shows performance results of the explicitly parallel block algorithms with static and dynamic scheduling of the work, respectively, as described in sections 5 and 6 (block size $nb = 48$). The corresponding results for the ESSL block LU routine are also shown.

Table 17 displays the performance ratio of the explicit algorithm with static scheduling versus the implicit algorithms of Table 15. Similar performance ratios of the explicitly parallel algorithms in Table 16 are displayed in Table 18.

7.2 Parallel Block Cholesky Algorithms

Table 19 shows performance results of the implicitly parallel block algorithm and the static and dynamic versions of the explicitly parallelized block right-looking LL^T algorithm. Similar results for the ESSL block Cholesky routine are also shown. As before, the implicitly parallel algorithm is obtained by

Table 16: Performance of explicitly parallel block LU algorithms.

Block algorithm	*Static*		*Dynamic*		*ESSL*	
Matrix size	500	1200	500	1200	500	1200
1 proc	83.5	98.4	84.9	98.1	88.7	99.0
2 proc	147.5	192.4	148.3	192.9	162.7	195.4
Speedup	*1.77*	*1.96*	*1.75*	*1.97*	*1.83*	*1.97*
3 proc	193.7	276.6	196.4	277.6	212.2	292.8
Speedup	*2.32*	*2.81*	*2.31*	*2.83*	*2.39*	*2.96*
4 proc	219.8	355.1	223.4	355.2	226.9	377.9
Speedup	*2.63*	*3.61*	*2.63*	*3.62*	*2.56*	*3.82*
5 proc	234.8	430.2	239.9	431.0	257.8	457.2
Speedup	*2.81*	*4.37*	*2.83*	*4.39*	*2.91*	*4.62*

Table 17: Performance ratio of explicitly versus implicitly parallel block LU algorithms.

m=n=1200	*Number of processors*				
Routine	1	2	3	4	5
Explicit (Static)	98.4	192.4	276.6	355.1	430.2
Implicit (Right)	95.8	167.4	217.4	231.3	253.1
Implicit (Crout)	90.0	138.7	169.5	185.3	191.5
$\frac{Explicit}{Right}$	*1.03*	*1.15*	*1.27*	*1.54*	*1.70*
$\frac{Explicit}{Crout}$	*1.09*	*1.39*	*1.63*	*1.92*	*2.25*

Table 18: Performance ratio of explicitly parallel block LU algorithms.

m=n=1200	*Number of processors*				
Routine	1	2	3	4	5
Dynamic (Day)	88.2	177.8	230.8	254.3	243.8
Static (Day)	90.0	163.4	216.7	251.0	236.1
$\frac{Dynamic}{Static}$	*0.980*	*1.088*	*1.065*	*1.013*	*1.033*
Dynamic (Night)	98.1	192.9	277.6	355.2	431.0
Static (Night)	98.4	192.4	276.6	355.1	430.2
ESSL (Night)	98.9	195.4	292.8	377.9	457.2
$\frac{Dynamic}{Static}$	*0.997*	*1.003*	*1.004*	*1.000*	*1.002*
$\frac{ESSL}{Dynamic}$	*1.008*	*1.012*	*1.054*	*1.064*	*1.061*

replacing calls to level-3 BLAS by calls to the parallel level-3 kernels described in Section 2.2. The block size nb is kept fixed (32 for the static version, 64 for the implicit and dynamic versions). The block size nbb is chosen according to the discussion in Section 6.2. For one processor $nbb = 256$ and decreases with increasing number of processors. For example, in the dynamic variant when $n = 1200$, nbb is chosen to be $256, 128, 64, 32$, and 32 corresponding to $1, 2, 3, 4$, and 5 processors, respectively.

Table 20 displays performance ratios of the explicit algorithms with static scheduling and dynamic scheduling and the ESSL routine.

Table 19: Performance results of parallel block Cholesky algorithms.

Routine	*Implicit*		*Static*		*Dynamic*		ESSL	
Matrix size	500	1200	500	1200	500	1200	500	1200
1 proc	79.4	92.2	76.0	90.9	78.5	91.9	91.0	98.0
2 proc	115.4	170.9	120.6	173.3	121.0	173.1	166.9	191.4
speed up	*1.45*	*1.85*	*1.59*	*1.91*	*1.54*	*1.88*	*1.83*	*1.95*
3 proc	128.7	240.1	134.3	246.1	138.0	247.2	233.6	278.2
speed up	*1.62*	*2.60*	*1.77*	*2.71*	*1.76*	*2.69*	*2.57*	*2.84*
4 proc	126.0	294.9	131.2	302.3	124.7	307.8	278.5	360.4
speed up	*1.58*	*3.20*	*1.73*	*3.33*	*1.59*	*3.35*	*3.06*	*3.68*
5 proc	115.5	344.0	116.6	350.9	108.2	354.4	314.3	435.2
speed up	*1.45*	*3.73*	*1.53*	*3.86*	*1.39*	*3.86*	*3.45*	*4.44*

Table 20: Performance ratios of parallel block Cholesky algorithms.

n = *1200*	*Number of processors*				
Routine	1	2	3	4	5
Explicit (Static)	90.9	173.3	246.1	302.3	350.9
Implicit (Right)	92.2	170.9	240.1	294.9	344.0
Explicit (Dynamic)	91.9	173.1	247.2	307.8	354.4
ESSL	98.0	191.4	278.2	360.4	435.2
$\frac{Static}{Implicit}$	*0.99*	*1.01*	*1.02*	*1.03*	*1.02*
$\frac{Dynamic}{Static}$	*1.01*	*1.00*	*1.00*	*1.02*	*1.01*
$\frac{ESSL}{Dynamic}$	*1.07*	*1.11*	*1.13*	*1.17*	*1.23*

We have also implicitly parallelized the block left-looking variant. However, the `DSYRK` operations of the left-looking variant (case $k > n$) did not parallelize as well as the corresponding `DSYRK` operations of the right-looking variant (case $k < n$, see Section 2.2). For the case $k > n$, A is partitioned into P column blocks and each processor computes an $n \times k/P$ contribution of the rank-k update. The global summation of these contributions to C is a sequential bottleneck.

7.3 Parallel Block QR Algorithms

Table 21 shows performance results of the implicitly and explicitly parallel block right-looking QR algorithm (block size $nb = 32$). Here the performance ratio is also displayed. In agreement with the results for the LU and LL^T factorizations we expect somewhat better performance for an explicitly parallel block algorithm with dynamic scheduling.

Table 21: Performance results of parallel block QR algorithms.

Routine	*Implicit*		*Explicit*		$\frac{Explicit}{Implicit}$
Matrix size	500	1200	500	1200	1200
1 proc	72.1	94.2	71.3	94.2	*1.00*
2 proc	94.3	160.5	132.4	182.9	*1.14*
Speedup	*1.31*	*1.70*	*1.86*	*1.94*	
3 proc	101.5	208.2	176.9	266.0	*1.28*
Speedup	*1.41*	*2.21*	*2.48*	*2.82*	
4 proc	103.1	245.5	203.5	345.2	*1.41*
Speedup	*1.43*	*2.61*	*2.85*	*3.66*	
5 proc	103.7	268.5	213.8	413.2	*1.54*
Speedup	*1.44*	*2.85*	*3.00*	*4.39*	

7.4 Discussion

The results in tables 15–21 show in summary that the explicitly parallel block algorithms give the best performance on more than one processor. The difference in performance between the two approaches increases with the number of processors. During daytime, when the machine normally has the heaviest load, the dynamic scheduling of the work gave up to 9% better performance than static scheduling of the work (LU, see Table 18). During night-time, the two scheduling approaches gave very similar results. Notice that the load of the machine varies over time and the difference between dynamic and static scheduling may therefore be much larger than shown here.

Two reasons why the static scheduling (contrary to intuition) is not better than the dynamic scheduling during night-time are that the system is not completely dedicated and that the dynamic scheduling does not impose an extra synchronization between each iteration.

The main reason for the differences in performance between the explicit parallel implementations of LU, LL^T and the corresponding routines in the

ESSL library is the different amount of parallelization overhead. Further, in the Cholesky case the ESSL routine works on a matrix in lower packed storage mode and uses a double precision workspace of size $96 \times n$, offering better data reuse on individual processors. The parallelization overhead of our parallel block algorithms is discussed later in this section.

In this context it is interesting to study the best possible speedup of the implicitly parallel block algorithms. Table 22 shows the real execution time level-3 fraction (f) and real execution time level-2 fraction ($1 - f$) of the block right-looking algorithms that are implicitly parallelized (numbers from Table 10). The second column of Table 22 shows the fractions for the LU factorization if we assume that all permutations also can be performed in parallel. In the second column all permutations are included in the level-2 fraction.

Table 22: Real execution time level-3 and level-2 fractions.

m=n=1200	LU	LU(pivot //)	LL^T	QR
f (level-3)	0.862	0.942	0.994	0.945
$1 - f$ (level-2)	0.138	0.058	0.006	0.055

Table 23 shows the speedup factors S_p of the implicitly parallel block algorithms if we assume that all level-3 operations are perfectly parallelized (with no overhead, see below) and all level-2 operations are performed on one processor. They are computed from Amdahl's law for parallel processing (see e.g. [11]):

$$S_P = \frac{P}{f + (1 - f)P}.$$

If $f < 1$, S_P is bounded by $(1 - f)^{-1}$ and this factor is displayed in the last column of Table 23.

Table 23: Perfect (theoretical) speedup factors of the implicitly parallel block algorithms.

m=n=1200	*Number of processors*					
Routine	1	2	3	4	5	$(1 - f)^{-1}$
LU	*1.00*	*1.76*	*2.35*	*2.83*	*3.22*	*7.25*
LU (pivot //)	*1.00*	*1.89*	*2.69*	*3.41*	*4.06*	*17.24*
LL^T	*1.00*	*1.99*	*2.96*	*3.93*	*4.88*	*166.7*
QR	*1.00*	*1.90*	*2.70*	*3.43*	*4.10*	*18.18*

One conclusion from these numbers and tables 15–21 is that it requires a small level-2 fraction, as in the block right-looking Cholesky algorithm, in order to be able to match the performance of the best explicitly parallel block algorithms. Possibly, the performance of the implicitly parallel block algorithms can be improved by also parallelizing the level-2 BLAS. Our experiences on the IBM 3090 VF/600J are that we do not gain anything by doing so. The granularity of the level-2 operations of the block algorithms is too small and the overhead for parallelization is too costly. Similar results are also shown in [8]. However, some level-2 operations have successfully been parallelized on other parallel shared memory systems [27].

7.5 Parallelization overhead

The results in tables 15, 16, 19, and 21 show very small speedup factors for matrices of size 500×500. The reason is that the overhead due to synchronization primitives is more or less constant while the work between synchronizations decreases with the size of the problem. Most overhead is associated with `ORIGINATE` (and `TERMINATE`, but this is less costly), and therefore the creation of tasks is done once in the beginning of the program. The overhead caused by `ORIGINATE` and `TERMINATE` is a function of the number of tasks created and the size of each task. The following theoretical model estimates the overhead cost associated with originating and terminating p tasks:

$$T_{ORIG}(p) = (o1 + o2 \cdot S) \cdot p$$

where S corresponds to the storage space in $Mbytes$ that is allocated for each task (including code and memory for local variables and arrays, etc). The parameters $o1$ and $o2$ in the linear model are determined from a least squares fit of overhead (caused by `ORIGINATE` and `TERMINATE`) timings for different matrix sizes. The values used are $o1 = 0.0075$ sec and $o2 = 0.0236$ sec.

The overhead costs associated with `SCHEDULE` (`DISPATCH`) and `WAIT FOR` vary between algorithms and factorizations but are normally much lower. Further, the overhead costs associated with `PARALLEL LOOP` and locks and events are negligible in this context. By considering the overhead costs associated with `ORIGINATE` and `TERMINATE` we can predict an upper bound on the theoretical speedup of our explicitly parallel block algorithms (see Table 24). The time on p processors is predicted as

$$T_p = \frac{T_1}{p} + T_{ORIG}(p-1).$$

$T_{ORIG}(p)$ is of the same magnitude for the explicitly and implicitly parallel block algorithms. However, the overhead due to `SCHEDULE` differs between the two approaches. The implicitly parallel block algorithms `SCHEDULE` $p-1$ tasks at each call to a parallel level-3 kernel (i.e., $2(p-1)$ or $3(p-1)$ tasks per block iteration). The explicitly parallel algorithms with static load balancing `SCHEDULE` $p-1$ tasks per block iteration, and the explicitly parallel algorithms with dynamic load balancing `SCHEDULE` only $p-1$ tasks totally.

Table 24: Upper bounds on performance and speedup factors of the explicitly parallel block algorithms.

m=n=1200	*Number of processors*				
Routine	1	2	3	4	5
LU	98.1	195.0	289.2	378.9	463.0
	1.00	*1.99*	*2.95*	*3.86*	*4.72*
LL^T	91.9	181.2	264.5	338.9	402.6
	1.00	*1.97*	*2.88*	*3.69*	*4.38*
QR	94.2	188.1	281.2	373.2	463.6
	1.00	*1.99*	*2.99*	*3.96*	*4.92*

8 Some Conclusions

Parallel block algorithms for the LU factorization with partial pivoting of a general matrix, the LL^T (Cholesky) factorization of a positive definite matrix, and the QR factorization of a general matrix have been presented. By replacing calls to level-3 BLAS in routines of the second prerelease of LAPACK [2] implicitly parallel block algorithms are obtained. This approach is compared to explicitly parallel block algorithms where the parallelism is invoked explicitly at the block level, offering the possibility to overlap and pipeline different matrix-matrix operations.

In order to design efficient and still transportable parallel block algorithms we have tried to fulfill the parallel key factors mentioned in the Introduction in the following way. The load balancing of the implicit approach is inherited from the parallel level-3 kernels. The load balancing of the explicit approach is performed by static or dynamic scheduling of the work. The static scheduling utilizes cost functions based on the number of flops expressed as GEMM equivalents. Benchmark tests give us (architecture- and problem-dependent) block sizes that define a suitable granularity of the block algorithms. Memory contention between processors is minimized by partitioning the data so that different processors work as much as possible on different parts of the matrix. The number of synchronization points is minimized by introducing the explicitly parallel block algorithms.

Performance results show that explicitly parallel block algorithms can reach close to practical peak performance on one to five processors by using optimized uniprocessor level-3 BLAS. Large problems ($n = 1200$ here) are required for high performance. Although the parallel efficiency decreases as a function of the number of processors, we have seen up to 90% parallel efficiency (defined as speedup factor divided by number of processors) on five processors of an IBM 3090 VF/600J system for the LU and QR factorizations and around 80% for the LL^T factorization. These results are justified by a theoretical model that predicts an upper bound on the best possible speedup with respect to the most significant parallelization overhead of language constructs and their implementations.

The corresponding best parallel efficiency of the implicitly parallel block algorithms is 75% for the LL^T factorization (just over 50% for the LU and QR factorizations). Notice that the best implicitly parallel block algorithm will depend on which parallel level-3 BLAS is best optimized, and the results presented here are all based on the level-3 kernels described in Section 2.2. By assuming perfect parallel speedup of the parallel level-3 BLAS we have shown that it is at least in theory possible to obtain around 80% parallel efficiency for the implicitly parallel block LU and QR algorithms, and over 95% for the corresponding implicitly parallel LL^T algorithm (LU and QR have larger real execution time level-2 fractions). The proportionately low efficiency of the implicitly parallel block algorithms can also be explained in terms of the parallelization overhead, which is architecture-dependent. To conclude, in order to exploit the full potential of hierarchical memory multiprocessor systems implicit parallelism (as in LAPACK) will require highly efficient parallel level-3 BLAS or explicit parallelism. The benefit from explicitly parallel block algorithms (compared to the implicitly parallel ones) increases with the number of physical processors. In general, the benefit from explicit parallelization is a function of both the granularity (block size) and the problem size.

Acknowledgements

We are grateful to the referees for some comments that have made our manuscript more clear.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen. LAPACK: A Portable Linear Algebra Library for High-Performance Computers. Tech. Report CS-90-105, Univ. of Tennessee, Knoxville, May 1990. (LAPACK Working Note #20).

[2] E. Andersson and J. Dongarra. Implementation Guide for LAPACK. Tech. Report CS-90-101, Univ. of Tennessee, Knoxville, April 1990. (LAPACK Working Note #18).

[3] E. Andersson and J. Dongarra. Evaluating Block Algorithm Variants in LAPACK. In J. Dongarra, P. Messina, D. Sorensen, and R. Voigt (eds), *Parallel Processing for Scientific Computing*, SIAM Publications, 1990, pp 3–8.

[4] C. Bischof. Adaptive Blocking in the QR Factorization. *The Journal of Supercomputing*, Vol. 3 (1989), pp 193–208.

[5] C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen. LAPACK Provisional Contents. Math. Comp. Sci. Report ANL-88-38, Argonne National Lab., Argonne, September 1988.(LAPACK Working Note #5).

[6] C. Bischof and C. Van Loan. The WY Representation for Products of Householder Matrices. *SIAM J. Scientific and Statistical Computing*, Vol. 8, (1987), pp s2–s13.

[7] E. Cohen, G. King, and J. Brady. Storage Hierarchies. *IBM Systems Journal*, Vol. 28(1) (1988) pp 62–76.

[8] M. Daydé and I. Duff. Use of Level 3 BLAS in LU Factorization in a Multiprocessing Environment on Three Vector Multiprocessors: the ALLIANT FX/80, the CRAY-2, and the IBM 3090 VF. Technical Report CERFACS, August 1990.

[9] J. Dongarra, J. Du Croz, S. Hammarling, and R. Hanson. An Extended Set of Fortran Basic Linear Algebra Subprograms. *ACM Trans. on Mathematical Software*, Vol. 14 (1988) pp 1–17, 18–32.

[10] J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff. A Set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. on Mathematical Software*, Vol. 16 (1990) pp 1–17, 18–28.

[11] J. Dongarra, I. Duff, D. Sorensen, and H. Van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers.* SIAM Publications, 1991.

[12] J. Dongarra and D. Sorensen. Linear Algebra on High-Performance Computers. In U. Schendel (ed), *High-Performance Computers 85*, North Holland, 1986, pp 3–32.

[13] J. Eriksson, P. Jacobson, B. Kågström, and E. Lindström. The CONLAB Environment:Algorithm Design for and Simulation of MIMD Architectures. In J. Dongarra, P. Messina, D. Sorensen, and R. Voigt (eds), *Parallel Processing for Scientific Computing*, SIAM Publications, 1990, pp 406–412.

[14] K. Gallivan, W. Jalby, U. Meier, and A. Sameh. Impact of Hierarchical Memory Systems on Linear Algebra Algorithm Design. *Int. J. Supercomputer Applications*, Vol 2 (1988), pp 12–48.

[15] K. Gallivan, R. Plemmons, and A. Sameh. Parallel Algorithms for Dense Linear Algebra Computations. *SIAM Review*, Vol. 32 (1990), pp 54–135.

[16] G. Golub and C. Van Loan. *Matrix Computations*. John Hopkins Press, 2nd edition, 1989.

[17] IBM. *Engineering and Scientific Subroutine Library Guide and Reference*. SC23-0184-3, November 1988.

[18] P. Jacobson. The CONLAB Environment. Report UMINF-173.90, Inst. of Information Processing, Univ. of Umeå, S-901 87 Umeå, 1990.

[19] B. Kågström and P. Ling. Level 2 and 3 BLAS Routines for IBM 3090 VF: Implementation and Experiences. In J. Dongarra, I. Duff, P. Gaffney, and S. McKee(eds), *Vector and Parallel Computing*, Ellis Horwood, 1989, pp 229–240.

[20] B. Kågström and C. Van Loan. GEMM-Based Level-3 BLAS. Technical Report, Dept. of Computer Science, Cornell University, December 1989.

[21] C. Lawson, R. Hanson, R. Kincaid, and F. Krogh. Basic Linear Algebra Subprograms for Fortran Usage. *ACM Trans. on Mathematical Software*, Vol. 5 (1979), pp 308–323.

[22] P. Ling. A Set of High Performance Level-3 BLAS Structured and Tuned for the IBM 3090 VF and Implemented in Fortran 77. Report UMINF-179.90, Inst. of Information Processing, Univ. of Umeå, S-901 87 Umeå, May 1990.

[23] B. Liu and N. Strother. Programming in VS Fortran on the IBM 3090 for Maximum Vector Performance. *IEEE Computer*, June (1988), pp 65–76.

[24] C. Moler, J. Little, and S. Bangert. *PRO-MATLAB User's Guide*. The MathWorks Inc., 1987.

[25] Parallel Computing Forum. *Proposal*, 1990.

[26] G. Radicati, Y. Robert, and P. Sguazzero. Block Processing in Linear Algebra on the IBM 3090 Vector Multiprocessor. *SUPERCOMPUTER*, Vol. 5, No. 1 (1988) pp 15–25.

[27] Q. Sheikh and J. Liu. Performance of Block Matrix Factorization Algorithms and LAPACK on CRAY Y-MP and CRAY-2. Cray Research Inc., 1990.

[28] R. Schreiber and C. Van Loan. A Storage Efficient WY Representation for Products of Householder Transformations. *SIAM J. Scientific and Statistical Computing*, Vol. 10 (1989), pp 53–57.

[29] L. Toomey, E. Plachy, R. Scarborough, R. Sahulka, J. Shaw, and A. Shannon. IBM Parallel Fortran. *IBM Systems Journal*, Vol. 27 (1988) pp 416–435.

[30] S. Tucker. The IBM 3090 System: An Overview. *IBM Systems Journal*, Vol. 25(1), (1986) pp 4–20.

Paper II

A Ring-Oriented Approach for Block Matrix Factorizations on Shared and Distributed Memory Architectures*

Krister Dackland[†], Erik Elmroth[†], and Bo Kågström

Department of Computing Science, Umeå University
S-901 87 Umeå, Sweden.
E-mail: dacke@cs.umu.se, elmroth@cs.ume.se, bokg@cs.ume.se

Abstract

A block (column) wrap-mapping approach for design of parallel block matrix factorization algorithms that are (trans)portable over and between shared memory multiprocessors (SMM) and distributed memory multicomputers (DMM) is presented. By reorganizing the matrix on the SMM architecture, the same ring-oriented algorithms can be used on both SMM and DMM systems with all machine dependencies comprised to a small set of communication routines. The algorithms are described on high level with focus on portability and scalability aspects. Implementation aspects of the *LU*, *Cholesky*, and *QR* factorizations and machine specific communication routines for some SMM and DMM systems are discussed. Timing results show that our portable algorithms have similar performance as machine specific implementations.

Keywords: Block matrix factorizations, parallel algorithms, portability, scalability, shared and distributed memory architectures.

---

*From R. F. Sinovec et. al. *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, Vol 1, by permission of SIAM, Philadelphia.

†Financial support has been received by the Swedish National Board of Industrial and Technical Development under grant NUTEK 89-02578P.

1 Introduction

With the introduction of advanced parallel computer architectures a demand for efficient and portable algorithms has emerged. Several attempts to design algorithms and implementations that are portable between *shared memory multiprocessors* (SMM) and *distributed memory multicomputers* (DMM) have been made. For example, implementations of virtual shared memory on DMM architectures [15, 21, 23] and the simulation of message passing with portable communication libraries on SMM architectures [3, 4, 13]. A major drawback with these attempts is that the demands for generality often deteriorates the performance for many problems that can be efficiently solved if the portability aspects are neglected. However, it is possible to obtain portability without loss of performance for a restricted class of problems. In this paper we restrict our study to parallel block matrix factorizations, such as the LU, *Cholesky*, and QR factorizations [7, 11, 12]. This class of problems can most likely be enlarged to several block algorithms, such as the ones included in LAPACK [2].

Our goal is to construct algorithms that, without loss of performance compared to implementations tuned for specific machines, are portable over and between different SMM and DMM architectures. This is done by further development of the ring-oriented block algorithms for DMM and SMM architectures, respectively, presented in [5, 7]. The algorithms are described on high level with focus on the portability aspects. Results are presented for implementations on Alliant FX2816, Intel iPSC/2 hypercube and IBM 3090 VF/600J and compared to the machine specific results presented in [5, 7]. We also discuss the scalability of these parallel algorithms, which is a measure of their capabilities to effectively use an increasing number of processors [18].

The outline is as follows. A generic ring-oriented block algorithm is presented in the perspective of a DMM architecture in Section 2. The algorithm is adapted to an SMM architecture in Section 3. The implementation of the machine specific communication routines are discussed in Section 4. Performance results are presented in Section 5, followed by a short discussion of performance modeling in Section 6. Finally, Section 7 discusses the results and presents some conclusions.

2 A Generic Ring-Oriented Block Algorithm

We assume that the target DMM architecture can embed a unidirected ring topology. The $m \times n$ matrix A to be factorized is block column wrap mapped onto the nodes to achieve good load balancing. For a given processor k $(= 0 : p-1)$, these blocks form a local matrix A_{local}, comprising the block columns $k+1$, $k+1+p$, $k+1+2p$, etc of A, where p is the number of processors.

Let nbl denote the number of block columns in A (for clarity we assume $m \geq n$), nbl_{local} the number of block columns in A_{local}, and *cb_index* the (block) index for the current block column. If the (local) block index i is initialized to 1, then a high level generic node algorithm for computing a matrix factorization can be described in the following way:

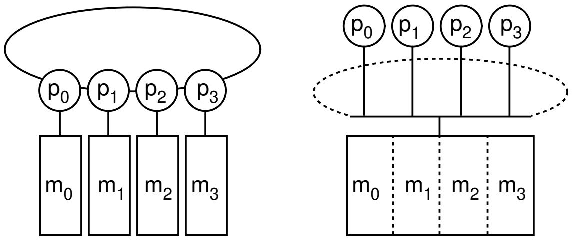
For $i\_global = 1 : nbl$
  If (I hold block i_global)
    If ($i\_global > 1$)
      UPDATE $A_{\text{local}}(i)$ w.r.t $CURRENT\_BLOCK(cb\_index)$ (1)
    End If
    FACTORIZE $A_{\text{local}}(i)$ & generate $NEXT\_CURRENT\_BLOCK$ (2)
    BROADCAST($NEXT\_CURRENT\_BLOCK, cb\_index$) (3)
    $i = i + 1$
  End If
  If ($i\_global > 1$)
    UPDATE $A_{\text{local}}(i : nbl_{\text{local}})$ w.r.t. $CURRENT\_BLOCK(cb\_index)$ (4)
  End If
  RECEIVE($CURRENT\_BLOCK, cb\_index$) (5)
End For

Processor 0 starts to factorize (2) the first block column and broadcasts (3) the factors, denoted *NEXT_CURRENT_BLOCK* to all other processors in the ring. Then, all processors update their remaining blocks (4), i.e., the blocks to the right of the last factorized block column in A, with respect to the received *CURRENT_BLOCK*. Notice that the next current processor, i.e., the right neighbour in the ring, updates (1) and factorizes (2) the next block column and broadcasts it (*NEXT_CURRENT_BLOCK*) before it completes the update corresponding to the previous factorization (*CURRENT_BLOCK*). This enables an overlapping called *pipelining between iterations* [5, 7].

In the implementations of the algorithms, the factorization of a block column is performed by a call to the corresponding level-2 routine, and the update is performed by one or more calls to level-3 BLAS [8, 19, 20]: `DGEMM` and `DTRSM` in *LU*, `DGEMM`, `DSYRK`, and `DTRSM` in *Cholesky*, `DGEMM` and `DTRMM` in *QR*.

The only extra storage needed is a matrix to hold the current (factorized) block. All synchronizations are implicit through the message passing of the current blocks. Except for the current blocks, a processor only refers to A_{local}. This gives an inherited data locality from the initial matrix distribution in the DMM environment, such that each processor performs the work on separate block columns. We would like to have the same data locality on an SMM environment.

Figure 1: DMM and SMM models viewed as a ring of processors.



3 Shared Memory and Ring-Oriented Block Algorithms

Our approach is to use the DMM paradigm as a model and run the same factorization routines on an SMM architecture viewed as a ring of processors, see Figure 1.

To accomplish this we reorganize A into A_{reorg} such that consecutive block columns of A_{reorg} are the block columns of A interchanged in a block column wrap mapped manner. By partitioning the work evenly over the p processors, each processor only makes access to a logical local matrix A_{local} that refers to a submatrix of the restructured global matrix A_{reorg}. In Figure 1 the memory units m_i of the DMM and SMM models store the same block columns of A. Moreover, it follows that the references to A_{local} can be made identical to the ones in the generic DMM algorithm. The main difference is how we deal with the current (factorized) block of A. In the DMM algorithm this block is broadcasted to all processors that keep it as a local copy. In the SMM algorithm the current block is a block column of A_{reorg} in the global shared memory, which can be accessed by all processors. The message passing of the current block is implemented as a change of reference for the current block to different locations in A_{reorg}.

4 Implementation of Communication Routines

To make the factorization routines (trans)portable over and between different SMM and DMM environments, all machine and compiler dependencies have been captured in communication routines with the same interfaces for both SMM and DMM systems. These implement the broadcasting and the reception of double precision and integer matrices on three MIMD machines using their extensions to FORTRAN 77 for communication and synchronization. The machines and compilers used are: Alliant FX2816 and FX/FORTRAN-2800 [1], IBM 3090 VF/600J and VS FORTRAN 2.5 [16], the Intel iPSC/2 hypercube and f77 [17].

The implementations on the iPSC/2 consist of interfaces to iPSC/2 communication library routines. For the SMM systems Alliant and IBM the `BROADCAST` and `RECEIVE` routines are implemented as an update of a current block pointer. To ensure that only one processor can get access to the pointer at a time it is synchronized by semaphores. In Figure 2 the body of the Alliant `BROADCAST` and `RECEIVE` subroutines are displayed. The routines look the same for the IBM system except for the syntax of the semaphores.

Figure 2: Alliant double precision `BROADCAST` and `RECEIVE` subroutine bodies.

```
C     BROADCAST
C
      IF (MSGID .EQ. CB) THEN
99       CALL ENTER()
         IF (SHDATA(1) .NE. 0) THEN
            CALL LEAVE()
            GOTO 99
         ELSE
            SHDATA(0) = LOCAL_CBI
            LOCAL_CBI = LOCAL_CBI + N
            SHDATA(1) = P
         ENDIF
         CALL LEAVE()
      ENDIF
```

```
C     RECEIVE
C
      IF (MSGID .EQ. CB) THEN
99       CALL ENTER()
         IF (SHDATA(0) .EQ. CBI) THEN
            CALL LEAVE()
            GOTO 99
         ELSE
            CBI = SHDATA(0)
            SHDATA(1) = SHDATA(1) - 1
         ENDIF
         CALL LEAVE()
       ENDIF
```

The `BROADCAST` and `RECEIVE` implementations are not general, but sufficient for the *LU*, *Cholesky*, and *QR* factorizations. For example, there is no need for explicit synchronization of the pivot vector in the *LU* factorization, since it is implicitly synchronized by the communication for the current block. Therefore, the communication routines for integer matrices are implemented as quick returns. Before calling a factorization routine, the "message passing" on the SMM system has to be initialized by a call to an initialization routine. Further details can be found in [6].

5 Performance results

We present performance results for the *LU*, *Cholesky*, and *QR* factorizations implemented as node subprograms in FORTRAN 77 and compiled for Alliant FX2816, Intel iPSC/2, and IBM 3090 VF/600J.

When we use these node subprograms the processors are assumed to be allocated and the matrix A is properly distributed or reordered. The timing of the routines measures the execution time for the factorizations, which all involve $O(n^3)$ arithmetic operations (for $m = n$) and communications (excluding the initial matrix distribution or reordering). The initial reordering of A on the SMM platforms is an $O(n^2)$ operation which is perfectly parallelizable.

The results in Tables 1 – 3 show the performance measured in *Mflops*, computed as the maximum time over p nodes divided by the number of floating point operations counted as in [2]. The optimal block size nb and the *parallel efficiency*, $E(p)$, are displayed for the three algorithms. The parallel efficiency is computed as

$$E(p) = \frac{Mflops(p)/Mflops(p_{\min})}{p/p_{\min}}.$$

where *Mflops*(x) is the *Mflops* number for x processors, and $p_{\min}$ is the minimum number of nodes that could solve the problem. The value of $p_{\min}$ is one for the Alliant and IBM systems while it is restricted by the available storage on each node for the iPSC/2 (at most 4 *Mbytes* per node including the code).

5.1 Alliant FX2816

The SMM system consists of 16 Intel i860 processors each with a theoretical peak performance of 40 *Mflops* in double precision real arithmetic. We use the DGEMM routine from the Alliant library libalgebra 2.0 [1]. Its best uniprocessor performance is about 35 *Mflops*. All other level-3 BLAS used are from the GEMM-based library [19, 20]. In Table 1, performance results on one to eight processors for the three routines are shown. By restricting the size of the cluster to eight, only the two processors (of four) that have distinct cache controllers on each processor module are used. This minimizes the memory conflicts in the global cache.

5.2 Intel iPSC/2 Hypercube

This DMM system has 64 scalar SX nodes. Each node is equipped with an Intel 80386, 4 *Mbyte* memory, and a Weitec 1167, which has a theoretical peak performance just under 0.6 *Mflops* in double precision real arithmetic. The BLAS used are the standard FORTRAN 77 implementations accompanying [2]. The best performance obtained from the level-3 BLA routine DGEMM is around 0.5 *Mflops* [5]. Results for iPSC/2 are presented in Table 2 together with previously presented results, *prev*, for machine specific implementations [5].

5.3 IBM 3090 VF/600J

The SMM system has 6 processors, each equipped with a vector facility (VF). The theoretical peak performance is 138 *Mflops* per processor in double precision real arithmetic. The practical peak performance obtained from the level-3 BLA

Table 1: Alliant FX2816: LU, $Cholesky$, and QR factorizations of a 1024×1024 matrix.

	LU			**Cholesky**			**QR**		
p	nb	*Mflops*	$E(p)$	nb	*Mflops*	$E(p)$	nb	*Mflops*	$E(p)$
1	64	25.1	1.00	64	25.2	1.00	32	25.8	1.00
2	48	47.8	0.95	64	47.7	0.95	32	49.4	0.96
3	48	67.2	0.89	48	66.2	0.88	32	70.9	0.92
4	48	86.4	0.86	32	79.8	0.79	32	90.8	0.88
5	48	102.0	0.81	32	90.0	0.71	16	109.0	0.85
6	32	113.3	0.75	32	96.0	0.63	16	125.5	0.81
7	32	125.7	0.72	16	103.6	0.59	16	141.7	0.79
8	32	137.0	0.68	16	109.5	0.54	16	153.2	0.74

Table 2: Intel iPSC/2: LU, $Cholesky$, and QR factorizations of a 1000×1000 matrix.

	LU				**Cholesky**				**QR**			
p	nb	*Mflops*	$E(p)$	$Prev$	nb	*Mflops*	$E(p)$	$Prev$	nb	*Mflops*	$E(p)$	$Prev$
4	4	1.29	1.00	1.29	8	1.29	1.00	1.29	4	1.55	1.00	1.55
8	2	2.53	0.98	2.53	8	2.54	0.99	2.53	2	3.06	0.99	3.06
16	2	4.93	0.96	4.93	4	4.88	0.95	4.83	2	5.99	0.97	6.00
32	2	9.40	0.91	9.39	2	8.73	0.85	8.39	1	11.58	0.93	11.65
64	1	17.10	0.83	17.02	2	13.40	0.65	12.79	2	21.54	0.87	21.71

Table 3: IBM 3090 VF/600J: LU, $Cholesky$, and QR factorizations of a 1200×1200 matrix.

	LU				**Cholesky**				**QR**			
p	nb	*Mflops*	$E(p)$	$Prev$	nb	*Mflops*	$E(p)$	$Prev$	nb	*Mflops*	$E(p)$	$Prev$
1	48	97.3	1.00	98.1	256	99.6	1.00	91.9	32	92.8	1.00	94.2
2	32	188.6	0.97	192.9	96	178.5	0.90	173.1	32	183.2	0.99	182.9
3	32	275.7	0.94	277.6	64	255.0	0.85	247.2	32	267.2	0.96	266.0
4	32	356.6	0.92	355.2	32	315.8	0.79	307.8	32	345.8	0.93	345.2
5	32	427.9	0.88	431.0	48	353.8	0.71	354.4	32	402.7	0.87	413.2

routine DGEMM is around 108 *Mflops* [7]. The level-3 BLAS used are from ESSL [16], except for a tuned FORTRAN implementation of DTRMM [22]. Results for IBM 3090 are presented in Table 3 together with previously presented results, *prev*, for machine specific implementations [7].

6 Performance Modeling

A model that identifies different components of the total execution time $T_{\text{pred}}(p)$ using p processors can be expressed as

$$T_{\text{pred}}(p) = T_a(p) + T_c(p) + T_w(p), \tag{6.1}$$

where $T_a(p)$ is the arithmetic time, $T_c(p)$ is the communication time, and $T_w(p)$ is the waiting time, including both idle and busy waiting. From $T_{\text{pred}}(p)$ it is, for example, possible to estimate optimal block sizes for given matrix sizes (m, n) and p. A reasonable simplification is to consider average times for the different time components. The motivation is that for sufficiently large matrices and almost optimal block sizes, the initial "matrix distribution" gives the processors approximately the same amount of work.

6.1 Arithmetic Time

We express the average arithmetic time $T_a(p)$ of an algorithm, implemented on a DMM or an SMM system, as

$$T_a(p) = \frac{\#\mathit{flops} \cdot t_a}{S_a(p, nb)} \mathcal{V}(m, n, nb), \tag{6.2}$$

where $\#\mathit{flops}$ is the total number of floating point operations of the matrix factorization, t_a is the time required for one floating point operation and $S_a(p, nb)$ is the arithmetic speedup on p processors with block size nb (excluding the costs for communication and waiting). Due to the memory hierarchy of the processor node and software overhead, the arithmetic time on one processor varies for different matrix and block sizes. The function $\mathcal{V}(m, n, nb)$ models this variation in performance. It can, for example, be determined from a least squares fit of uniprocessor timing results for different m, n and block sizes nb. As a consequence also the arithmetic speedup varies and we express $S_a(p, nb)$ as the linear speedup times a factor that models the variation of the arithmetic speedup $(p(1 + v)$, where $|v(m, n, nb)| < 1)$. Accurate iPSC/2 models for $\mathcal{V}(m, n, nb)$ and $S_a(p, nb)$ are presented in [5].

6.2 Communication Time

Since all communications are expressed by message passing primitives $T_c(p)$ can be modeled in terms of the number of messages each processor communicates (broadcasts or receives) and the cost for sending and receiving a message of a given size. To make the discussion clearer, we assume that $m = n$, all block columns have the same number of columns, i.e., n is a multiple of nb, and that the number of block columns $nbl = n/nb$ is a multiple of p. In Table 4 the total message traffic of the three ring-oriented block algorithms is displayed.

Table 4: Message traffic of the ring-oriented block algorithms for $m = n$ and $nbl = n/nb$.

	BROADCAST		RECEIVE	
Factorization	# Blocks	Size (in bytes)	# Blocks	Size (in bytes)
LU	nbl	$4(n^2+n \cdot nb)$	$(p-1)nbl$	$4(p-1)(n^2+n \cdot nb)$
	nbl	$8n$	$(p-1)nbl$	$8(p-1) \cdot n$
Cholesky	$nbl-1$	$4(n^2-n \cdot nb)$	$(p-1)(nbl-1)$	$4(p-1)(n^2-n \cdot nb)$
QR	nbl	$4(n^2+n \cdot nb)$	$(p-1)nbl$	$4(p-1)(n^2+n \cdot nb)$
	nbl	$8n \cdot nb$	$(p-1)nbl$	$8(p-1) \cdot n \cdot nb$

In each iteration of the generic algorithm one message is broadcasted and $p-1$ messages are received. For the *LU* factorization, the factorized block and a part of the pivot vector are broadcasted in each iteration. In the *Cholesky* factorization, only the factorized block is sent in each iteration (except for the last one). The size of the current block in iteration i is $(n-i \cdot nb)nb$ for the Cholesky factorization while it is $(n-(i-1)nb)nb$ for the LU and QR factorizations. In the QR factorization we use a compact WY representation of the block transformation in each step (see, e.g. [12]), so besides the factorized block we have to broadcast a triangular matrix S in each block iteration. The costs for communicating these messages are entirely determined by our target architecture. For a DMM environment the usual linear model $\alpha + M\beta$ can be used, where α is the startup cost and β is the cost per unit for communicating a message of M bytes. Average time models of $T_c(p)$ for iPSC/2 are presented in [5]. In an SMM environment a single address space is accessible to every processor in the system. The "message passing" is accomplished through synchronized access of shared variables in a shared memory system. The communication costs are determined by the memory hierarchy of the system and the costs for synchronization (e.g. cache and latency effects). A modeling of the Alliant FX2800 is under investigation.

6.3 Waiting Time

The average waiting time $T_w(p)$ can be divided into two parts; *busy waiting* time, $T_{bw}(p)$, and *idle waiting* time, $T_{iw}(p)$. $T_{bw}(p)$ comprises the time needed for the first processor to factorize and "distribute" the first current block, and the time when different processors are "waiting for" messages to arrive. The last part exists if the current processor uses more time to produce the next factorized block (including its broadcast) than the other processors need for the update corresponding to the previous factorization. Essentially, $T_{iw}(p)$ comprises the time some processors are idle at the end of the computations (mainly due to imbalance of the load). In [5] an iPSC/2 model for $T_w(p)$ is discussed.

7 Discussion

The performance results in Section 5 show that the generic ring-oriented algorithms are (trans)portable and give good parallel performance on both DMM and SMM environments. The results compete favourably with previously published results for machine specific implementations of similar algorithms on IBM 3090 VF/600J [7] and iPSC/2 [5]. To our knowledge the Alliant FX2816 results are the best available for these three matrix factorizations programmed entirely in FX/FORTRAN-2800. The similarity in performance on the iPSC/2 is explained by the small differences between the generic implementations and the machine specific versions. It follows that the modeling and performance evaluation in [5] also are valid for the ring-oriented algorithms discussed here. The modeled execution time, T_{pred}, predicts the best block size to be the same as the best one in real execution, except occasionally, when it predicts a nearby block size which in real execution shows a performance close to the optimal one. For the SMM environments the good performance of the ring-oriented algorithms are explained by the reorganization of the matrix, giving each processor block columns that are stored consecutively, and the proportionately low synchronization costs. The consecutive storage enables updating of several block columns in one operation, instead of one operation per block column. This implies less amount of software overhead and larger matrices in the level-3 BLAS operations. In [7] software overhead and level-3 fractions for parallel block matrix factorizations and their impact on the performance of an SMM system are discussed.

Due to space limits, we have only presented results for a fixed problem size and a varying number of processors. In this case the parallel speedup is an appropriate measure of the scalability of the algorithm. Both the parallel speedup and the parallel efficiency are determined by the serial fraction of the algorithm, as well as, communication costs and possible overhead due to redundant work. For the number of processors we have used, the algorithm-architecture combinations show speedups with a linear behaviour and we can argue that the algorithms have good scaling properties. In general, the solution time may finish to decrease (or even increase) due to the parallel overheads if we continue to increase the number of processors while maintaining a fixed problem size. It would then be more appropriate to consider the *scaled-speedup* which is defined as the speedup obtained when the problem size is increased linearly with p [14]. For example, results for iPSC/2 in [5] show that the maximal performance per processor is almost constant if the matrix is large enough (see also [9]).

The GEMM-based level-3 BLAS [19, 20] and a highly optimized uni-processor DGEMM routine make the ring-oriented approach efficient on both DMM and SMM platforms. Here demonstrated on the Alliant FX2816 system. Now, all machine dependencies are restricted to the communication primitives, the DGEMM routine and lower level BLAS. The implementation of the communication primitives on an SMM environment is cost effective and easy, and will be further simplified in FORTRAN 90, which offers dynamic data structures.

We believe our approach is applicable to block algorithms in general, making it possible to construct a complete library portable over and between DMM and SMM platforms. This enables use of routines in sequence, hopefully without redistribution or reorganization of the matrix between each operation. If a reorganization is needed on a SMM system, it is an easily parallelized $O(m \cdot n)$ operation. The cost for redistributing a matrix on a DMM environment is determined by the connectivity properties of the physical interconnection network, and is normally more expensive than on an SMM model. Moreover, it is possible to extend our approach to a 2-dimensional mesh topology and use a square block scattered decomposition of the matrices (e.g. see [10, 9]) for both DMM and SMM environments.

References

[1] Alliant Computer Systems Corporation. FX/FORTRAN-2800, Programmer's Handbook. FX/SERIES Linear Algebra Library. 1990.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[3] L. Bomans, D. Roose, and R. Hempel. The Argonne/GMD macros in Fortran for portable programming and their implementation on the Intel iPSC/2. *Parallel Computing*, 15, (1990), pp 119–132.

[4] J. Boyle, R. Butler, T. Disz, B. Glickfeld, E. Lusk, R. Overbeek, J. Patterson, and R. Stevens, *Portable Programs for Parallel Processing*, Holt, Rinehart, and Winston, Inc, 1987.

[5] K. Dackland and E. Elmroth. Parallel Block Matrix Factorizations for Distributed Memory Multicomputers. Report UMINF-92.03, Inst. of Information Processing, Univ. of Umeå, S-901 87 Umeå, May 1992.

[6] K. Dackland and E. Elmroth. Ring-oriented Block Matrix Factorization Algorithms for Shared and Distributed Memory Architectures. Report

UMINF-92.04, Inst. of Information Processing, Univ. of Umeå, S-901 87 Umeå, June 1992.

[7] K. Dackland, E. Elmroth, B. Kågström, and C. Van Loan. Parallel Block Matrix Factorizations on the Shared Memory Multiprocessor IBM 3090 VF/600J. *International Journal of Supercomputer Applications*, Vol 6.1, MIT Press (1992), pp 69–97

[8] J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff. A Set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. on Mathematical Software*, Vol. 16 (1990) pp 1–17, 18–28.

[9] J. Dongarra, R. van de Geijn, and D. Walker. A Look at Scalable Dense Linear Algebra Libraries. ORNL/TM-12126, Oak Ridge National Lab., Oak Ridge, Tennessee 37831, July 1992.

[10] G. Fox, G. Lyzenga, S. Otto, J. Salmon, and D. Walker, *Solving Problems on Concurrent Processors*, Vol. 1, Prentice Hall, Englewood Cliffs, N.J., 1988.

[11] K. Gallivan, R. Plemmons, and A. Sameh. Parallel Algorithms for Dense Linear Algebra Computations. *SIAM Review*, Vol. 32 (1990), pp 54–135.

[12] G. Golub and C. Van Loan, *Matrix Computations*, 2nd edition, John Hopkins Press, 1989.

[13] A. Geist, M. Heath, B. Peyton, and P. Worley. PICL a Portable Instrumented Communication Library. Tech. Memorandum ORNL/TM-6150, Oak Ridge National Laboratory, 1990, pp 1213–1222.

[14] J. L. Gustafson, G. R. Montry, and R. E. Benner. Development of Parallel Methods for a 1024-Processor Hypercube. *SIAM J. Sci. Statist. Comput.*, Vol. 9(4) (1988), pp 609–638.

[15] H. Hellwagner. A Survey of Virtually Shared Memory Schemes. TUM-I9056, Institut für Informatik, Technische Universität München, 1990.

[16] IBM. Engineering and Scientific Subroutine Library Guide and Reference. SC23-0184-3, Nov.1988. VS FORTRAN Version 2 Release 5 Language and Library Reference. SC26-4221-6, Dec.1990.

[17] Intel Corporation. iPSC/2 and iPSC/860, Programmer's Reference Manual. June 1990.

[18] V. Kumar and A. Gupta. Analyzing Scalability of Parallel Algorithms and Architectures. TR 91-18, Dept. of Computer Science, Univ. of Minnesota, Minneapolis, MN - 55455, Revised Nov. 1992.

[19] B. Kågström P. Ling, and C. Van Loan. High Performance GEMM-Based Level-3 BLAS: Sample Routines for Double Precision Real Data. In M. Durand and F. El Dabaghi (eds.), *High Performance Computing II*, Elsevier Science Publishers B.V. (North-Holland), (1991), pp 269–281.

[20] B. Kågström and C. Van Loan. GEMM-Based Level-3 BLAS. Tech. Report CTC91TR47, Cornell University, December 1989.

[21] K. Li. IVY: A Shared Virtual Memory on Loosely Coupled Multiprocessors. Ph.D. Thesis, Yale University, 1986.

[22] P. Ling. A Set of High Performance Level-3 BLAS Structured and Tuned for the IBM 3090 VF and Implemented in Fortran 77. Report UMINF-179.90, Inst. of Information Processing, Univ. of Umeå, S-901 87 Umeå, May 1990.

[23] M. Stumm and S. Zhou. Algorithms Implementing Distributed Shared Memory. *IEEE Computer*, Vol. 23, No. 5, (1989), pp 54–64.

Paper III

The Set of 2-by-3 Matrix Pencils — Kronecker Structures and their Transitions under Perturbations*

Erik Elmroth and Bo Kågström

Department of Computing Science, Umeå University,
S-901 87 Umeå, Sweden.
E-mail: elmroth@cs.ume.se, bokg@cs.ume.se

Abstract

The set (or family) of 2-by-3 matrix pencils $A - \lambda B$ comprises 18 structurally different Kronecker structures (canonical forms). The algebraic and geometric characteristics of the generic and the 17 non-generic cases are examined in full detail. The complete closure hierarchy of the orbits of all different Kronecker structures is derived and presented in a closure graph that show how the structures relate to each other in the 12-dimensional space spanned by the set of 2-by-3 pencils. Necessary conditions on perturbations for transiting from the orbit of one Kronecker structure to another in the closure hierarchy are presented in a labeled closure graph. The node and arc labels show geometric characteristics of an orbit's Kronecker structure and the change of geometric characteristics when transiting to an adjacent node, respectively. Computable normwise bounds for the smallest perturbations $(\delta A, \delta B)$ of a generic 2-by-3 pencil $A - \lambda B$ such that $(A + \delta A) - \lambda(B + \delta B)$ has a specific non-generic Kronecker structure are presented. First explicit expressions for the perturbations that transfer $A - \lambda B$ to a specified non-generic form are derived. In this context tractable and intractable perturbations are defined. Secondly, a modified `GUPTRI` that computes a specified Kronecker structure of a

*To appear in *SIAM Journal on Matrix Analysis and its Applications.* Preprinted by permission of SIAM, Philadelphia.

generic pencil is used. Perturbations devised to impose a certain non-generic structure is computed in a way that guarantees to find a KCF on the closure of the orbit of the intended KCF. Both approaches are illustrated by computational experiments. Moreover, a study of the behaviour of the non-generic structures under random perturbations in finite precision arithmetic (using the `GUPTRI` software) show for which sizes of perturbations the structures are invariant and also that structure transitions occur in accordance with the closure hierarchy. Finally, some of the results are extended to the general m-by-$(m+1)$ case.

Keywords: Matrix pencils (2-by-3), Kronecker canonical form, generalized Schur decomposition, orbit, codimension, Kronecker structure hierarchy, closest non-generic structure, controllability.

1 Introduction

Singular matrix pencils $A-\lambda B$, where A and B are m-by-n matrices with real or complex entries, appear in several applications. Examples include problems in control theory relating to a linear system $E\dot{x}(t) = Fx(t)+Gu(t)$, where E and F are p-by-p matrices, and G is p-by-k. Solvability issues of a singular system (i.e., $\det(E) = 0$), such as the existence of a solution, consistent initial values, and its explicit solution can be revealed from the Kronecker structure of $A - \lambda B \equiv F - \lambda E$ (e.g. see [9, 20]). The problems to find the controllable subspace, uncontrollable modes or an upper bound on the distance to uncontrollability for a controllable system $E\dot{x}(t) = Fx(t) + Gu(t)$ can all be formulated and solved in terms of certain reducing subspaces of the matrix pencil $A - \lambda B \equiv [G \quad F] - \lambda[0 \quad E]$ (e.g. see [15, 17, 18, 6]).

In most applications it is enough to transfer $A - \lambda B$ to a *generalized Schur form* (e.g. to GUPTRI form [7, 8])

$$P^H(A-\lambda B)Q = \begin{bmatrix} A_r - \lambda B_r & * & * \\ 0 & A_{reg} - \lambda B_{reg} & * \\ 0 & 0 & A_l - \lambda B_l \end{bmatrix}, \tag{1.1}$$

where P (m-by-m) and Q (n-by-n) are unitary and $*$ denotes arbitrary conforming submatrices. Here the square upper triangular block $A_{reg} - \lambda B_{reg}$ is regular and has the same regular structure as $A - \lambda B$ (i.e., contains all generalized eigenvalues (finite and infinite) of $A - \lambda B$). The rectangular blocks $A_r - \lambda B_r$ and $A_l - \lambda B_l$ contain the singular structure (right and left minimal indices) of the pencil and are block upper triangular. The *singular blocks of right* (column) and *left* (row) *indices of grade* j are

$$L_j \equiv \begin{bmatrix} -\lambda & 1 & & \\ & \cdot & \cdot & \\ & & -\lambda & 1 \end{bmatrix} \quad \text{and} \quad L_j^T \equiv \begin{bmatrix} -\lambda & & \\ 1 & \cdot & \\ & \cdot & -\lambda \\ & & 1 \end{bmatrix}, \tag{1.2}$$

of size j-by-$(j+1)$ and $(j+1)$-by-j, respectively. $A_r - \lambda B_r$ has only right minimal indices in its Kronecker canonical form (KCF), indeed the same L_j blocks as $A - \lambda B$. Similarly, $A_l - \lambda B_l$ has only left minimal indices in its KCF, the same L_j^T blocks as $A - \lambda B$. If $A - \lambda B$ is singular at least one of $A_r - \lambda B_r$ and $A_l - \lambda B_l$ will be present in (1.1). The explicit structure of the diagonal blocks in staircase form can be found in [8]. If $A - \lambda B$ is regular $A_r - \lambda B_r$ and $A_l - \lambda B_l$ are not present in (1.1) and the GUPTRI form reduces to the upper triangular block $A_{reg} - \lambda B_{reg}$. Staircase forms that reveal the Jordan structure of the zero and infinite eigenvalues are contained in $A_{reg} - \lambda B_{reg}$.

Given $A - \lambda B$ in GUPTRI form we also know different pairs of reducing subspaces [18, 7]. Suppose the eigenvalues on the diagonal of $A_{reg} - \lambda B_{reg}$ are ordered so that the first k, say, are in Λ_1 (a subset of the spectrum of $A_{reg} - \lambda B_{reg}$) and the remainder are outside Λ_1. Let $A_r - \lambda B_r$ be m_r-by-n_r. Then the left and right reducing subspaces associated with Λ_1 are spanned by the leading $m_r + k$ columns of P and the leading $n_r + k$ columns of Q, respectively. When Λ_1 is empty, the corresponding reducing subspaces are called *minimal*, and when Λ_1 contains the whole spectrum the reducing subspaces are called *maximal*.

If $A - \lambda B$ is m-by-n, where $m \neq n$, then for almost all A and B it will have the same KCF, depending only on m and n (the *generic case*). The generic Kronecker structure for $A - \lambda B$ with $d = n - m > 0$ is

$$\operatorname{diag}(L_\alpha, \ldots, L_\alpha, L_{\alpha+1}, \ldots, L_{\alpha+1}), \tag{1.3}$$

where $\alpha = \lfloor m/d \rfloor$, the total number of blocks is d, and the number of $L_{\alpha+1}$ blocks is $m \bmod d$ (which is 0 when d divides m) [16, 3]. The same statement holds for $d = m - n > 0$ if we replace $L_\alpha, L_{\alpha+1}$ in (1.3) by $L_\alpha^T, L_{\alpha+1}^T$. Square pencils are generically regular, i.e., $\det(A - \lambda B) = 0$ if and only if λ is an eigenvalue. The generic singular pencils of size n-by-n have the Kronecker structures [19]:

$$\operatorname{diag}(L_j, L_{n-j-1}^T), \quad j = 0, \ldots, n-1. \tag{1.4}$$

In summary, generic rectangular pencils have only trivial reducing subspaces and no generalized eigenvalues at all. Generic square singular pencils have the same minimal and maximal reducing subspaces. Only if $A - \lambda B$ satisfies a special condition (lies in a particular manifold) does it have nontrivial reducing subspaces and generalized eigenvalues (the *non-generic case*). Moreover, only if it is perturbed so as to move continuously within that manifold do its reducing subspaces and generalized eigenvalues also move continuously and satisfy interesting error bounds [5, 7]. These requirements are natural in many control and systems theoretic problems such as computing controllable subspaces and uncontrollable modes.

Several authors have proposed (staircase-type) algorithms for computing a generalized Schur form (e.g. see [1, 4, 13, 14, 11, 12, 16, 20]). They are numerically stable in the sense that they compute the exact Kronecker structure (generalized Schur form or something similar) of a nearby pencil $A' - \lambda B'$. Let $\|\cdot\|_E$ denote the Euclidean (Frobenius) matrix norm. Then $\delta \equiv \|(A - A', B - B')\|_E$ is an upper bound on the distance to the closest $(A + \delta A, B + \delta B)$ with the KCF of (A', B'). Recently, robust software with error bounds for computing the GUPTRI form of a singular $A - \lambda B$ has been published [7, 8]. Some computational experiments that use this software will be discussed later.

The existing algorithms do not guarantee that the computed generalized Schur form is the "most" non-generic Kronecker structure within distance δ. However, if δ is of the size $O(\|(A, B)\|_E \epsilon)$, where ϵ is the relative machine precision, we know that (A, B) is close to a matrix with the Kronecker structure that the algorithm reports. It would of course be desirable to have algorithms that could solve the following "nearness" problems:

- Compute the closest non-generic pencil of a generic $A - \lambda B$.
- Compute the closest matrix pencil with a specified Kronecker structure.
- Compute the most non-generic pencil within a given distance δ.

If the closest structure is not unique we are mainly interested in the most non-generic KCF. From the perturbation theory for singular pencils [5] we know that all these problems are ill-posed in the sense that the generalized eigenvalues and reducing subspaces for a non-generic $A - \lambda B$ can change discontinuously as a function of A and B. Therefore, to be able to solve these problems we need to regularize them by restricting the allowable perturbations as mentioned above. In this contribution we make a comprehensive study of the set of 2-by-3 pencils in order to get a greater understanding of (i) these "nearness" problems and how to solve them, and (ii) existing algorithms/software for computing the Kronecker structure of a singular pencil. The full implications of this "case study" to general m-by-n pencils are topics for further research.

In the following we give a summary of our contribution and the organization of the rest of the paper. Section 2 is devoted to algebraic and geometric characteristics of the set of 2-by-3 pencils. In Section 2.1 we disclose the structurally different Kronecker structures and show how all the non-generic structures can be generated by a staircase-type algorithm, starting from the generic canonical form. Some algebraic and geometric characteristics of the 18 different Kronecker structures are summarized in three tables. Section 2.2 introduces the concepts of orbits of matrix pencils and their (co)dimensions. The codimensions of the orbits of the 2-by-3 matrix pencils, which depend only on their

Kronecker structures [3], are displayed in Table 3. They vary between zero (the generic case) and 12 (= $2mn$) for the zero pencil (the most non-generic case). Indeed, all 2-by-3 pencils "live" in a 12-dimensional space spanned by the set of all generic pencils. In Section 2.3 we derive a graph describing the closure hierarchy of the orbits of all 18 different Kronecker structures for the set of 2-by-3 pencils. The closure graph is presented in Figure 1. By labeling the nodes in the closure graph with their geometric characteristics and the arcs with the change in geometric characteristics for transiting to an adjacent node, we get a labeled graph showing necessary conditions on perturbations for transiting from one Kronecker structure to another. The labeled closure graph is presented in Figure 2 in Section 2.4.

Section 3 is devoted to an experimental study of how the non-generic Kronecker structures behave under random perturbations in finite precision arithmetic, using the `GUPTRI` software [7, 8]. Assuming a fixed relative accuracy of the input data, structure invariances and transitions of each non-generic case are studied as a function of the size of the perturbations added. The results summarized in Table 4 are discussed in terms of tolerance parameters used in `GUPTRI` for determining the Kronecker structure. For large enough perturbations all non-generic pencils turn generic (as expected). Some non-generic cases transit between several non-generic structures before turning generic. These transitions always go from higher to lower codimensions, along the arcs in the closure graph.

In Section 4 we present computable normwise bounds for the smallest perturbations $(\delta A, \delta B)$ of a generic 2-by-3 pencil $A - \lambda B$ such that $(A+\delta A) - \lambda(B+\delta B)$ has a specific non-generic Kronecker structure. Two approaches to impose a non-generic structure are considered. First, explicit expressions for the perturbations that transfer $A - \lambda B$ to a specified non-generic form are derived in Section 4.1. In this context tractable and intractable perturbations are defined. We compute a perturbation $(\delta A, \delta B)$ such that $(A + \delta A) - \lambda(B + \delta B)$ is guaranteed to be in the closure of the manifold (orbit) of a certain KCF. If the KCF found is the intended KCF, then the perturbation is said to be tractable. If the KCF found is even more non-generic then the perturbation is intractable. An intractable perturbation finds any other structure within the closure of the manifold, i.e., a structure that can be found by traveling along the arcs from the intended KCF in the closure graph. A summary of these perturbations is presented in a perturbation graph (Figure 3), where the path to each KCF's node shows the tractable perturbation required to find that KCF starting from the generic KCF (an L_2 block). After illustrating intractable perturbations we derive some results regarding the closest non-generic Kronecker structure of a generic 2-by-3 (and 1-by-2) pencil. In the second approach, we use a modified `GUPTRI` for computing a specified Kronecker structure of a generic pencil (Section 4.2). Computational experiments on random 2-by-3 pencils for the two approaches are presented in Section 4.3. It is the intractable perturbations, which impose the most non-generic structure (with highest codimension) for a given size of the perturbations (e.g. the relative accuracy of the data), that are requested in applications (e.g. computing the uncontrollable subspace). Finally, in Section 5 we comment on the general case and extend our results for the closest non-generic pencil to a generic m-by-$(m+1)$ pencil.

2 Algebraic and Geometric Characteristics of the Set of 2-by-3 Matrix Pencils

In this section we disclose the structurally different Kronecker structures and show how all the non-generic structures can be generated by a staircase-type algorithm, starting from the generic canonical form. Moreover, we discuss the codimensions of associated orbits and derive a closure graph, showing the Kronecker structure hierarchy of the set of 2-by-3 pencils.

2.1 Structurally Different Kronecker Structures

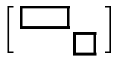
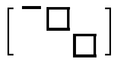
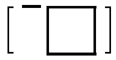
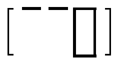
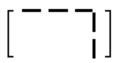
The *generic* case corresponds to A and B of size 2-by-3 both having full row rank and non-intersecting column nullspaces. This implies that $A - \lambda B$ is strictly equivalent to an L_2 block:

$$P^{-1}(A - \lambda B)Q = L_2 \equiv \begin{bmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.1}$$

By inspection, we see that the A- and B-parts of L_2 have row rank 2 and non-intersecting 1-dimensional column nullspaces. The generic canonical form L_2 can be obtained by deleting the last row of $J_3(0) - \lambda I_3$, a 3-by-3 Jordan block corresponding to the zero eigenvalue. $J_3(0)$ is the generic canonical form of a 3-by-3 matrix with zero as a triple eigenvalue and the associated non-generic Jordan structures are $J_2(0) \oplus J_1(0)$ and $J_1(0) \oplus J_1(0) \oplus J_1(0)$ (i.e., a 3-by-3 zero matrix). Notice that a generic 3-by-3 matrix is diagonalizable with unspecified non-zero eigenvalues (i.e., all Jordan blocks of size 1-by-1).

In the following we disclose the structurally different non-generic singular cases of size 2×3. By structurally different we mean that all cases have different Kronecker structures (canonical forms). There exists 17 different non-generic singular cases. The simplest way to construct all non-generic canonical forms of size 2×3 is to generate all possible combinations of L_1, L_0, J_2, J_1, R_1, N_1, N_2, L_0^T, and L_1^T blocks as in Table 1. Algorithms for computing the Kronecker structure of a singular pencil reveal the right (or left) singular structure and the Jordan structure of the zero (or infinite) eigenvalue simultaneously. Therefore, we only distinguish the zero and infinite Jordan structures and put a non-zero and finite eigenvalue in R_1, a regular 1-by-1 block with an unspecified eigenvalue. We will use R_2 to denote a 2-by-2 block with non-zero finite eigenvalues, i.e., R_2 is used to denote any of the three structures $J_1(\alpha) \oplus J_1(\beta)$, $J_1(\alpha) \oplus J_1(\alpha)$,

Table 1: 2×3 pencils built from different Kronecker and Jordan blocks.

Number of cases	Block structure	KCF
1		L_2
3		$L_1 \oplus \{J_1, R_1, N_1\}$
5		$L_0 \oplus \{J_1, R_1, N_1\} \oplus \{J_1, N_1\}$
3		$L_0 \oplus \{J_2, R_2, N_2\}$
1		$L_0 \oplus L_1 \oplus L_0^T$
1		$2L_0 \oplus L_1^T$
3		$2L_0 \oplus \{J_1, R_1, N_1\} \oplus L_0^T$
1		$3L_0 \oplus 2L_0^T$

and $J_2(\alpha)$, where $\alpha,\ \beta \neq \{0, \infty\}$. Notice that if $R_2 = J_2(\alpha)$ then $A - \alpha B$ and B has $J_2(0)$ in its KCF. It is only for the case $L_0 \oplus R_2$ that we can have an $J_2(\alpha)$ block. If we treat these three cases separately we get 19 non-generic cases, but for our purposes it is sufficient to define R_2 as above.

In order to get more insight into the non-generic structures we would like to show how all the non-generic structures can be generated by a staircase-type algorithm. By dropping the row rank of the A-part and/or B-part of L_2 (2.1) and imposing different sizes of their "common column or row nullspace(s)" (see Table 3) we are able to generate all 17 non-generic cases starting from the generic canonical form (in the following denoted $A - \lambda B$). Algorithmically, we keep the rank of, for example, B constant and vary the row rank of A while imposing possible sizes of their "common nullspace(s)". A decrease of the row rank is done by deleting a non-zero element ($= 1$) in the first or second row of A and/or B and the dimension of the common column nullspace is imposed by permutations of the non-zero elements. After decreasing the row rank of B by one we repeat the procedure until the row rank of B equals zero. By doing so we can generate 12 structurally different non-generic pencils of size 2×3. These correspond to cases 2–13 in Table 2, where we display a case number i, the matrix pair (A_i, B_i), $\mathrm{r}(A_i), \mathrm{r}(B_i)$, the row ranks of A_i and B_i, respectively, $\mathrm{n}(A_i, B_i)$, the dimension of the common column nullspace of A_i and B_i. Finally, in the last column we display the generalized Schur forms (GUPTRI forms) which correspond to the Kronecker block structures displayed in Table 1.

Case 1 in Table 2 corresponds to the generic structure. Cases 2–5 are obtained by keeping $\mathrm{r}(B_i) = 2$ and varying $\mathrm{r}(A_i)(2,1,0)$ and $\mathrm{n}(A_i,B_i)(0,1)$. In cases 6–10 we keep $\mathrm{r}(B_i) = 1$ and vary $\mathrm{r}(A_i)$ (as before) and $\mathrm{n}(A_i,B_i)(0,1,2)$. Finally, in cases 11–13 $\mathrm{r}(B_i) = 0$, $\mathrm{r}(A_i)$ and $\mathrm{n}(A_i,B_i)$ are varied ((0, 1, 2) and (1, 2, 3), respectively). In cases 8, 9, 12 and 13, the matrix pairs have a common row nullspace as well, corresponding to L_0^T blocks in their KCF. The number of L_0^T blocks equals the dimension of the common row nullspace (1 for cases 8, 9 and 12 and 2 for case 13). Notice that $\mathrm{n}(A_i,B_i) = 2$ for three of these four cases and $\mathrm{n}(A_i,B_i) = 3$ for case 13. However, $\mathrm{n}(A_i,B_i) = 2$ is neither a necessary or sufficient condition for a 2-by-3 matrix pair to have a common row nullspace (see cases 7' and 9' below). If we exchange the roles of A and B in the derivation of the non-generic forms 2–13 they will appear in a different order with the N_k blocks and $J_k(0)$ blocks exchanged.
We have five more cases to retrieve, denoted 1', 10', 4', 7' and 9' in Table 2. Case x' denotes a case that has the same row-ranks and column-nullities as case x, and is obtained from case x by permuting rows or columns.

Case 1': By swapping columns 2 and 3 in B_1 we still have a matrix pair with $\mathrm{r}(A_i) = \mathrm{r}(B_i) = 2$ and $\mathrm{n}(A_i,B_i) = 0$. We denote this pencil case 1'. As can be seen in Table 2, `GUPTRI` delivers the KCF $L_1 \oplus R_1$ for $A_{1'} - \lambda B_{1'}$. After the first step of deflation in `GUPTRI` (which identifies that $A_i, i = 1, 1'$ has a 1-dimensional column nullspace ($n(A_i) = 1$) and that $\mathrm{n}(A_i,B_i) = 0, i = 1, 1'$) we are left with the pencils:

$$A_1^{(1)} - \lambda B_1^{(1)} = \left[\begin{array}{cc} 0 & 1 \end{array}\right] - \lambda \left[\begin{array}{cc} 1 & 0 \end{array}\right], \quad A_{1'}^{(1)} - \lambda B_{1'}^{(1)} = \left[\begin{array}{cc} 0 & 1 \end{array}\right] - \lambda \left[\begin{array}{cc} 0 & 1 \end{array}\right]. \tag{2.2}$$

The difference is that $\mathrm{n}(A_1^{(1)}, B_1^{(1)}) = 0$ while $\mathrm{n}(A_{1'}^{(1)}, B_{1'}^{(1)}) = 1$. Is there any algebraic explanation? We find the answer in the classical characterization of a singular pencil with a right (column) index [9].

Let the matrix $R[A,B,i]$ of size $(i+2)m \times (i+1)n$ be defined by

$$R[A,B,i] = \begin{bmatrix} A & 0 & \cdots & 0 \\ B & A & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & B & A \\ 0 & \cdots & 0 & B \end{bmatrix}, \tag{2.3}$$

where A and B are $m \times n$ matrices. When it is clear from context we use the abbreviated notation $R[i]$ for $R[A,B,i]$. With the notation above we can state the following theorem.

Table 2: Summary of the 18 structurally different 2×3 pencils, numbered and presented in the order they are derived in Section 2.

i	A_i	B_i	$\mathrm{r}(A_i)$	$\mathrm{r}(B_i)$	$\mathrm{n}(A_i, B_i)$	GUPTRI form
1	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	2	2	0	$\begin{bmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 1 \end{bmatrix}$
2	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	1	2	0	$\begin{bmatrix} 1 & -\lambda & 0 \\ 0 & 0 & -\lambda \end{bmatrix}$
3	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	0	2	1	$\begin{bmatrix} 0 & -\lambda & 0 \\ 0 & 0 & -\lambda \end{bmatrix}$
4	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	1	2	1	$\begin{bmatrix} 0 & -\lambda & 1 \\ 0 & 0 & -\lambda \end{bmatrix}$
5	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	2	2	1	$\begin{bmatrix} 0 & 1-\lambda & 0 \\ 0 & 0 & 1-\lambda \end{bmatrix}$
6	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	2	1	0	$\begin{bmatrix} -\lambda & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
7	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	1	1	1	$\begin{bmatrix} 0 & -\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}$
8	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	0	1	2	$\begin{bmatrix} 0 & 0 & -\lambda \\ 0 & 0 & 0 \end{bmatrix}$
9	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	1	1	2	$\begin{bmatrix} 0 & 0 & 1-\lambda \\ 0 & 0 & 0 \end{bmatrix}$
10	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	2	1	1	$\begin{bmatrix} 0 & 1 & -\lambda \\ 0 & 0 & 1 \end{bmatrix}$
11	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	2	0	1	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
12	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	1	0	2	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$
13	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	0	0	3	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
$1'$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	2	2	0	$\begin{bmatrix} -\lambda & 1 & 0 \\ 0 & 0 & 1-\lambda \end{bmatrix}$
$10'$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	2	1	1	$\begin{bmatrix} 0 & 1-\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}$
$4'$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	1	2	1	$\begin{bmatrix} 0 & -\lambda & 0 \\ 0 & 0 & 1-\lambda \end{bmatrix}$
$7'$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	1	1	1	$\begin{bmatrix} 0 & -\lambda & 1 \\ 0 & 0 & 0 \end{bmatrix}$
$9'$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	1	1	2	$\begin{bmatrix} 0 & 0 & -\lambda \\ 0 & 0 & 1 \end{bmatrix}$

Theorem 2.1 *[9] The following statements are equivalent.*

- $A - \lambda B$ *is singular with a right (column) minimal index of lowest degree* $k \geq 0$, *i.e.,* $A - \lambda B$ *has no right minimal indices of degree* $< k$.
- $A - \lambda B$ *is equivalent to the pencil*

$$\begin{bmatrix} L_k & 0 \\ 0 & A' - \lambda B' \end{bmatrix}, \tag{2.4}$$

where L_k *is a* $k \times (k+1)$ *Kronecker block.* $A' - \lambda B'$ *may have indices of higher degree.*

- $R[i]$ *has full column rank* $\mathrm{r}(R[i]) = (i+1)n$ *for* $i = 0, 1, \ldots, k-1$, *while* $\mathrm{r}(R[k]) < (k+1)n$, *or equivalently, the column nullity* $\mathrm{n}(R[i]) = 0$ *for* $i = 0, 1, \ldots, k-1$ *and* $\mathrm{n}(R[k]) > 0$.

By applying Theorem 2.1 to cases 1 and 1' we see that $\mathrm{n}(R[1]) = 0, \mathrm{n}(R[2]) = 1$ for case 1 while $\mathrm{n}(R[1]) = 1, \mathrm{n}(R[2]) = 2$ for case 1', which justify that case 1 has an L_2 block as its KCF and case 1' has an L_1 block in its KCF. After the second deflation of case 1', `GUPTRI` is left with the pencil $[1] - \lambda[1]$ which corresponds to R_1, a regular block of size 1×1.

Case 10': By swapping columns 2 and 3 of B_{10} we still get a matrix pair with $\mathrm{r}(A_i) = 2, \mathrm{r}(B_i) = 1$ and $\mathrm{n}(A_i, B_i) \equiv \mathrm{n}(R[0]) = 1$. We denote this pencil case 10'. This swapping does not change the singular structure. However, the N_2 block in case 10 is now split into two regular 1×1 blocks N_1 and R_1, i.e., one infinite eigenvalue is turned non-zero.
To get the remaining three cases we will swap rows 1 and 2 in A_i for $i = 4, 7$ and 9.

Case 4': If we swap rows 1 and 2 in A_4 we still get a matrix pair with $\mathrm{r}(A_i) = 1, \mathrm{r}(B_i) = 2$ and $\mathrm{n}(A_i, B_i) = 1$. We denote this pencil case 4'. The only difference is that the $J_2(0)$ block in case 4 is now split into two regular 1×1 blocks $J_1(0)$ and R_1, i.e., one zero eigenvalue is turned non-zero.

A dual form of Theorem 2.1 can be stated for a left (row) minimal index of lowest degree $k \geq 0$. Then L_k^T takes the place of L_k and $L[A, B, i]$ of size $(i+1)m \times (i+2)n$ replaces $R[A, B, i]$, where

$$L[A, B, i] = \begin{bmatrix} A & B & 0 & \cdots & 0 \\ 0 & A & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A & B \end{bmatrix}, \tag{2.5}$$

and we are considering row ranks (or row nullities) of $L[A, B, i]$. (When it is clear from context we also here use the abbreviated notation $L[i]$ for $L[A, B, i]$.) We use this dual form to characterize the last two cases. Notice that $\mathrm{n}(R[A, B, 0])$

is equivalent to the dimension of the common column nullspace for A and B and that $\mathrm{n}(L[A,B,0])$ is equivalent to the dimension of the common row nullspace for the two matrices.

Case 7': By swapping rows 1 and 2 in A_7 we still get a matrix pair with $\mathrm{r}(A_i)=1, \mathrm{r}(B_i)=1$ and $\mathrm{n}(A_i,B_i)=1$. We denote this pencil case 7'. However, this swap imposes a common row nullspace of $A_{7'}$ and $B_{7'}$ as well, and will therefore change the singular structure completely. The regular part $(J_1(0)\oplus N_1)$ disappears and is replaced by $L_1 \oplus L_0^T$, i.e., the generic singular structure of a 2-by-2 pencil [19]. $\mathrm{n}(A_i,B_i) \equiv \mathrm{n}(R[0]) = 1$ for $i = 7$ and 7'. For case 7, $\mathrm{n}(R[1]) = 2, \mathrm{n}(L[0]) = 0$ while $\mathrm{n}(R[1]) = 3, \mathrm{n}(L[0]) = 1$ for case 7'.

Case 9': By swapping rows 1 and 2 in A_9 we still get a matrix pair with $\mathrm{r}(A_i)=1, \mathrm{r}(B_i)=1$ and $\mathrm{n}(A_i,B_i)=2$. We denote this pencil case 9'. However, $A_{9'}$ and $B_{9'}$ do not have a common row nullspace. Also here the regular part disappears and $R_1 \oplus L_0^T$ turns into L_1^T, i.e., a generic 2-by-1 pencil. $\mathrm{n}(L[0]) = 1$ for case 9, while $\mathrm{n}(L[0]) = 0, \mathrm{n}(L[0]) = 1$ for case 9'.

In Table 3 we display ranks of A_i, B_i and nullities of $R[k]$ and $L[k]$ for some values of k together with our structurally different singular structures of the set of 2-by-3 pencils. The ordering of the cases is explained in Section 2.2.

Table 3: Geometric characteristics of the 18 structurally different 2×3 pencils.

Case	$\mathrm{r}(A_i)$	$\mathrm{r}(B_i)$	$\mathrm{n}(A_i,B_i)$	$\mathrm{n}(R[1])$	$\mathrm{n}(R[2])$	$\mathrm{n}(L[0])$	$\mathrm{n}(L[1])$	KCF	$\mathrm{Cod}(A_i-\lambda B_i)$
1	2	2	0	0	1	0	0	L_2	0
1'	2	2	0	1	2	0	0	$L_1\oplus R_1$	1
2	1	2	0	1	2	0	0	$L_1\oplus J_1$	2
6	2	1	0	1	2	0	0	$L_1\oplus N_1$	2
5	2	2	1	2	3	0	0	$L_0\oplus R_2$	2
4'	1	2	1	2	3	0	0	$L_0\oplus J_1\oplus R_1$	3
10'	2	1	1	2	3	0	0	$L_0\oplus N_1\oplus R_1$	3
4	1	2	1	2	3	0	0	$L_0\oplus J_2$	4
10	2	1	1	2	3	0	0	$L_0\oplus N_2$	4
7	1	1	1	2	3	0	0	$L_0\oplus J_1\oplus N_1$	4
7'	1	1	1	3	5	1	2	$L_0\oplus L_1\oplus L_0^T$	5
3	0	2	1	2	3	0	0	$L_0\oplus 2J_1$	6
11	2	0	1	2	3	0	0	$L_0\oplus 2N_1$	6
9'	1	1	2	4	6	0	1	$2L_0\oplus L_1^T$	6
9	1	1	2	4	6	1	2	$2L_0\oplus R_1\oplus L_0^T$	7
8	0	1	2	4	6	1	2	$2L_0\oplus J_1\oplus L_0^T$	8
12	1	0	2	4	6	1	2	$2L_0\oplus N_1\oplus L_0^T$	8
13	0	0	3	6	9	2	4	$3L_0\oplus 2L_0^T$	12

2.2 Orbits and Their Codimensions

Each of the 18 singular canonical forms (A_i, B_i) in Table 3 defines a manifold of *strictly equivalent* pencils in $2mn(=12)$ dimensional space:

$$\text{orbit}(A_i - \lambda B_i) = \{P_i^{-1}(A_i - \lambda B_i)Q_i : \det(P_i)\det(Q_i) \neq 0\}.$$

The dimension of $\text{orbit}(A - \lambda B)$ is equal to the dimension of the tangent space, $\tan(A - \lambda B)$, to the orbit of $A - \lambda B$. The tangent space is defined as

$$f(X, Y) = X(A - \lambda B) - (A - \lambda B)Y, \tag{2.6}$$

where X is an $m \times m$ matrix and Y is an $n \times n$ matrix [3]. Since (2.6) maps a space of dimension $m^2 + n^2$ linearly to a space of dimension $2mn$, the dimension of the tangent space is $m^2 + n^2 - d$, where d is the number of (linearly) independent solutions of $f(X, Y) = 0$.

The codimension is the dimension of the space complementary to the tangent space, i.e.,

$$\text{cod}(A - \lambda B) = 2mn - \dim(\tan(A - \lambda B)) = d - (m - n)^2.$$

The codimensions of the orbits depend only on their Kronecker structures. Demmel and Edelman [3] show that the codimension of the orbit of an $m \times n$ pencil $A - \lambda B$ can be computed as the sum of separate codimensions:

$$\text{cod}(A - \lambda B) = c_{\text{Jor}} + c_{\text{Right}} + c_{\text{Left}} + c_{\text{Jor,Sing}} + c_{\text{Sing}},$$

where the different components are defined as follows.

The codimension of the Jordan structure is

$$c_{\text{Jor}} = \sum_{\lambda \neq 0,\infty} (q_1(\lambda) + 3q_2(\lambda) + 5q_3(\lambda) + \ldots - 1) + \sum_{\lambda = 0,\infty} (q_1(\lambda) + 3q_2(\lambda) + 5q_3(\lambda) + \ldots),$$

where the summation is over all eigenvalues and $q_1(\lambda) \geq q_2(\lambda) \geq q_3(\lambda) \ldots$, denote the sizes of the Jordan blocks corresponding to the eigenvalue λ. The first part of c_{Jor} corresponds to unspecified eigenvalues different from zero and infinity, which explains the term -1 in the codimension count.

The codimension of the right and left singular blocks are

$$c_{\text{Right}} = \sum_{j>k} (j - k - 1) \quad \text{and} \quad c_{\text{Left}} = \sum_{j>k} (j - k - 1),$$

respectively, where the summation for c_{Right} is over all pairs of blocks L_j and L_k, for which $j > k$, and the summation for c_{Left} is over all pairs of blocks L_j^T and L_k^T for which $j > k$.

The codimension due to interaction between the Jordan structure and the singular blocks is

$$c_{\text{Jor,Sing}} = (\text{size of complete regular part}) \cdot (\text{number of singular blocks}).$$

The codimension due to interaction between right and left singular blocks is

$$c_{\text{Sing}} = \sum_{j,k} (j + k + 2),$$

where the summation is over all pairs of blocks L_j and L_k^T.
The codimensions of our 18 different canonical forms are displayed in the last column of Table 3. We have ordered the cases by increasing codimension. In general, we see that by making A and B more rank deficient and increasing their "common nullspace(s)" ($\mathrm{n}(R[k])$ and $\mathrm{n}(L[k])$ for $k \geq 0$) we generate non-generic pencils with higher codimension. The generic pencil has codimension 0 while the matrix pair $(A, B) = (0_{2\times 3}, 0_{2\times 3})$ has codimension 12 ($= 2mn$), i.e., defines a "point" in 12-dimensional space.

2.3 The Closure Graph for Different Kronecker Structures

Since orbit(L_2) spans the complete 12-dimensional space, it is obvious that all other structures are in the closure of the orbit of L_2, and it is just as obvious that $3L_0 \oplus 2L_0^T$ (the zero pencil) is in the closure of the orbit of any other KCF. Since all other closure relations are not that obvious, we derive a complete closure graph for the set of 2-by-3 matrix pencils.

Throughout the paper we display graphs such that orbits (nodes) with the same codimension are displayed on the same horizontal level.

Theorem 2.2 *For the set of* 2-*by*-3 *pencils, the directed graph in Figure 1 shows all closure relations as follows. One KCF is in the closure of the orbit of another KCF if and only if it exists a path to its node from the node of the KCF defining the closure (downwards in the graph).*

Proof. First we prove that each arc in the graph correspond to a closure relation, and then we prove that these are all arcs that can exist. We prove that one KCF is in the closure of the orbit of another KCF by showing that the one in the closure is just a special case of the one defining the closure. We show proofs for each arc starting from the zero pencil. Since the proof is rather space demanding, we here limit ourselves to prove one of the arcs and refer to appendix A for the complete proof.

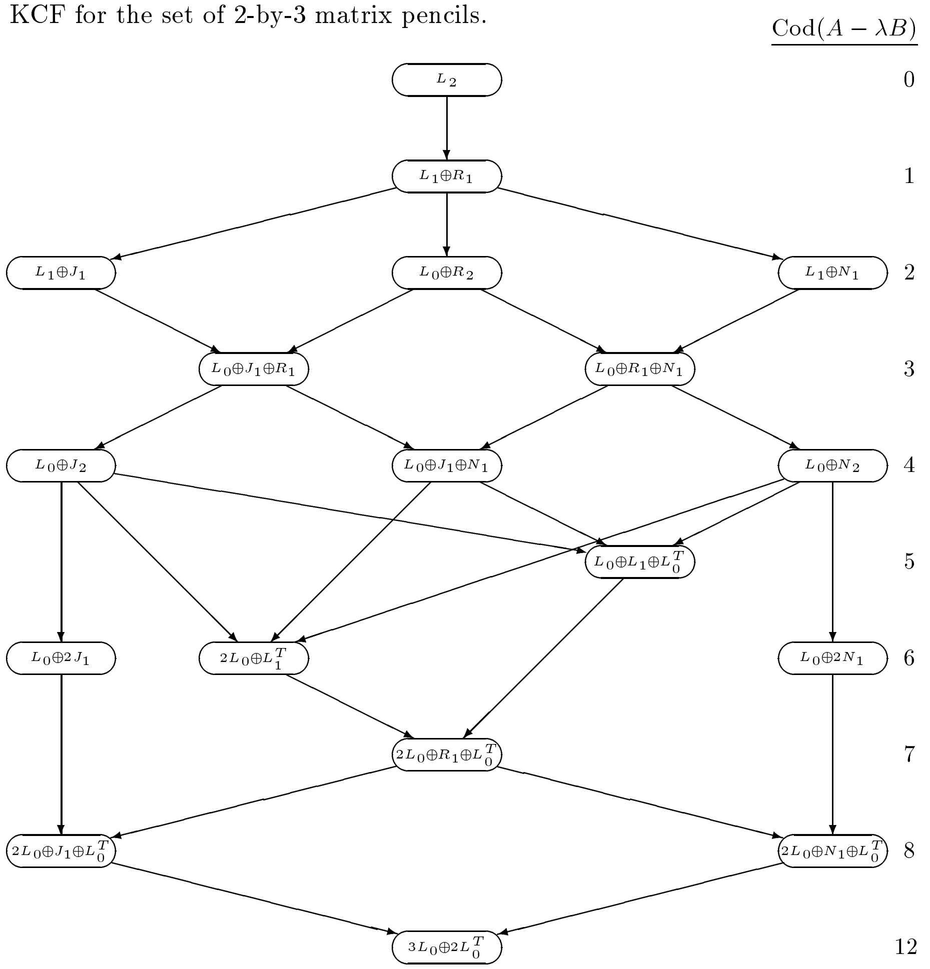
Starting at the zero pencil, the first arc with non-trivial proof corresponds to that $2L_0 \oplus J_1 \oplus L_0^T$ is in the closure of orbit($2L_0 \oplus R_1 \oplus L_0^T$). This follows from the fact that $2L_0 \oplus J_1 \oplus L_0^T$ is the special case $\alpha = 0$ of

$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & \beta \\ 0 & 0 & 0 \end{bmatrix},$$

which is equivalent to $2L_0 \oplus R_1 \oplus L_0^T$ for all other α (assuming that β is non-zero).

The proofs for all other arcs are done similarly. For some of them, an equivalence transformation is needed for transformation to KCF. $\square$

Figure 1: A graph displaying the closure hierarchy of the orbits of all 18 different KCF for the set of 2-by-3 matrix pencils.



2.4 Labeled Closure Graph Showing Necessary Conditions on Perturbations for Transiting from One Structure to Another

One way to interpret a relation in the closure hierarchy is that a KCF that is in the closure of the orbit of another KCF "lives" in the space defined by that orbit. That is, if we consider the closure of the orbit of a non-generic KCF with certain rank-defects in Table 3, then to be in that closure a KCF must preserve or increase these defects. For example, since $L_1 \oplus J_1$ has $\mathrm{rank}(A) = 1$, no KCF with $\mathrm{rank}(A) > 1$ can be in its closure. A necessary condition for a KCF to be in the closure of $\mathrm{orbit}(L_1 \oplus J_1)$ is that the geometric characteristics $\mathrm{r}(A) \leq 1, \mathrm{r}(B) \leq 2, \mathrm{n}(A,B) \geq 0, \mathrm{n}(R[1]) \geq 1, \mathrm{n}(R[2]) \geq 2, \mathrm{n}([L[0]) \geq 0$ and $\mathrm{n}([L[1]) \geq 0$ are satisfied (see Table 3). Moreover, the change in geometric characteristics from, for example, $L_1 \oplus J_1$ whose orbit spans a 10-dimensional space (codimension is 2), to $L_0 \oplus J_1 \oplus R_1$ whose orbit spans a 9-dimensional space (codimension is 3), is nothing but a 1-dimensional restriction of the 10-dimensional space. We also note that $L_0 \oplus J_1 \oplus R_1$ is in the closure of $\mathrm{orbit}(L_0 \oplus R_2)$, which also spans a 10-dimensional space. Indeed, $L_0 \oplus J_1 \oplus R_1$ spans a 9-dimensional space in the intersection of the two 10-dimensional spaces spanned by the closures of $\mathrm{orbit}(L_1 \oplus J_1)$ and $\mathrm{orbit}(L_0 \oplus R_2)$.

When looking for perturbations corresponding to the arcs in the graph, a necessary condition for these perturbations is to fulfill the change in geometric characteristics. Indeed, by combining the geometric characteristics in Table 3 and the closure graph we get necessary conditions on perturbations $(\delta A, \delta B)$ for transiting from one structure to another.

We introduce the following labels. Let

$$[\mathrm{n}_r(A), \mathrm{n}_r(B), \mathrm{n}(A,B), \mathrm{n}(R[1]), \mathrm{n}(R[2]), \mathrm{n}([L[0]), \mathrm{n}([L[1])]$$

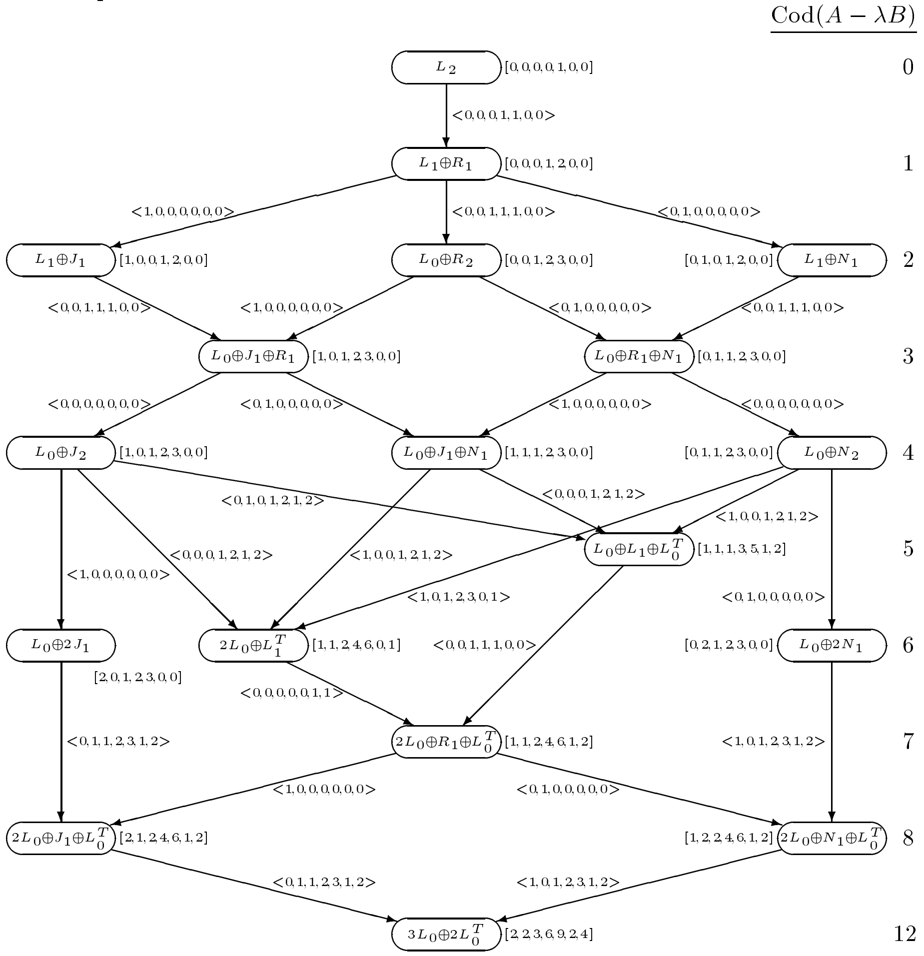
label the geometric characteristics for one node in the graph, where $n_r(A)$ and $n_r(B)$ denote the dimension of the row-nullspace in A and B, respectively, and all other characteristics are as in Table 3. Moreover, we label the change in geometric characteristics for transiting from one structure to an adjacent node by

$$< \mathrm{n}_r(A), \mathrm{n}_r(B), \mathrm{n}(A,B), \mathrm{n}(R[1]), \mathrm{n}(R[2]), \mathrm{n}([L[0]), \mathrm{n}([L[1]) > .$$

In Figure 2 a labeled closure graph is presented, with the geometric characteristics shown for each KCF and the change in geometric characteristics shown for each arc.

When transiting from one KCF to another, the geometric characteristics of the source node and the geometric characteristics on the arc are added to give the characteristics of the destination KCF. Since a KCF in the closure of another ones orbit cannot have a smaller dimensional nullspace for any of the matrices of the labels, the values on the arcs must all be non-negative.

Figure 2: The labeled closure graph for all 18 different KCF for the set of 2-by-3 matrix pencils.



Notice that the arc from $L_0 \oplus J_1 \oplus R_1$ to $L_0 \oplus J_2$ and the arc from $L_0 \oplus R_1 \oplus N_1$ to $L_0 \oplus N_2$ both have no change in the geometric characteristics. For these transitions the non-zero finite eigenvalue is turned to a zero eigenvalue and to an infinite eigenvalue, respectively. This does not affect any of the nullspaces displayed in the labels.

To transit several levels in the closure graph we just add the labels of changes in geometric characteristics for the arcs that are traveled during the transition. Each label of changes in geometric characteristics define necessary conditions on the perturbations $(\delta A, \delta B)$ to perform the transit. Later, we will derive perturbations required to transit from L_2 to any of the non-generic structures. In our derivation, however, we for most cases transit directly to the intended structure. There are only a few cases that require compound perturbations that transit via another KCF.

3 Structure Invariances and Transitions of Non-Generic Pencils under Perturbations

Since computing the Kronecker structure of a singular pencil is a potentially ill-posed problem [5], it is interesting to see how the non-generic cases behave under perturbations in finite precision arithmetic. We add (uniformly distributed) random perturbations of different sizes $\epsilon_n (= 10^{-10}, 10^{-9}, \ldots, 10^{-2})$ to all A_i and B_i, corresponding to the generic and 17 non-generic cases, and compute their generalized Schur forms using `GUPTRI` [7, 8] assuming a fixed relative accuracy $\epsilon_u (= 10^{-8})$ of the input data. We repeat this procedure 100 times and study the structure invariances and transitions of each non-generic case as a function of the size of the perturbations added.

`GUPTRI` has two input parameters `EPSU` (ϵ_u above) and `GAP` which are used to make rank decisions in order to determine the Kronecker structure of an input pencil $A - \lambda B$. Inside `GUPTRI` the absolute tolerances $\mathtt{EPSUA} = \|A\|_E \cdot \mathtt{EPSU}$ and $\mathtt{EPSUB} = \|B\|_E \cdot \mathtt{EPSU}$ are used in all rank decisions, where the matrices A and B, respectively, are involved. Suppose the singular values of A are computed in increasing order, i.e., $0 \leq \sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_k \leq \sigma_{k+1} \leq \ldots$; then all singular values $\sigma_k < \mathtt{EPSUA}$ are interpreted as zeros. The rank decision is made more robust in practice: if $\sigma_k < \mathtt{EPSUA}$ but $\sigma_{k+1} \geq \mathtt{EPSUA}$, `GUPTRI` insists on a gap between the two singular values such that $\sigma_{k+1}/\sigma_k \geq \mathtt{GAP}$. If $\sigma_{k+1}/\sigma_k < \mathtt{GAP}$, σ_{k+1} is also treated as zero. This process is repeated until an appreciable gap between the zero and non-zero singular values is obtained. In all of our tests we have used $\mathtt{EPSU} = 10^{-8}$ and $\mathtt{GAP} = 1000.0$. All computations (in sections 3 and 4) are performed on a SUN SPARC workstation in double precision complex arithmetic with unit roundoff $= O(10^{-17})$.

In Table 4 we display the computed Kronecker structures of the 17 perturbed non-generic pencils for 100 random perturbations for each ϵ_n. For each case all structure invariances and transitions are shown from left to right. The symbol $\xrightarrow{10^{-x}}$ indicates that the Kronecker structure is invariant under perturbations smaller than $\epsilon_n = 10^{-x}$, and that the structure changes (at least for some of the 100 tests) for perturbations of size 10^{-x}. For a size of the perturbations that has not given the same structure for all 100 tests, all KCF:s found are placed within curly brackets with a number within parentheses after each KCF showing the number of that particular KCF that has been found. As before, the cases are displayed in increasing codimension order and the transit KCF forms within curly brackets are ordered similarly.

From Table 4 we see that for large enough perturbations all non-generic structures turn generic (as expected). `GUPTRI` finds the same non-generic structure as long as $\epsilon_n < tol \equiv \min(\texttt{EPSUA}, \texttt{EPSUB}) \cdot \texttt{GAP}$. This behaviour is in agreement with the perturbation theory for singular pencils [5, 7]. Only if $A - \lambda B$ lies in a particular manifold does it have a non-generic Kronecker structure with non-trivial reducing subspaces and possibly eigenvalues. Moreover, only if it is perturbed so as to move continuously within that manifold does its original Kronecker structure remain. Actually, by choosing a $tol > 0$, we have thickened the manifolds so that they are no longer a set of measure zero.

All transitions from the initial case to the final generic case is clearly from cases with higher codimension to cases with lower. By a closer look we can also see that all the transitions are performed upwards (or backwards) along the arcs in the closure graph (Figure 1). This means that the perturbations cure the rank deficiencies in the non-generic pencil without contributing with any new singularities. `GUPTRI` increases the rank in A and B and decreases the size of their "common nullspace(s)", i.e., the "inverse" operations compared to what we did in Section 2.1. In other words, when a pencil $A - \lambda B$ with a given non-generic KCF is perturbed, by $\delta A - \lambda \delta B$ then $A - \lambda B$ is in the closure of $\text{orbit}((A + \delta A) - \lambda(B + \delta B))$.

Even if we see that all of the cases transit via some other non-generic structures before all 100 tests turn generic, we can also see that if we for each case and each size of the perturbation only consider the KCF that has been found in most tests, then it is only for cases 8 and 12 a transit KCF is found. Notice that all tests for cases 8 and 12 find the same other non-generic KCF for the smallest perturbation. In other words, when the perturbation is big enough to change the KCF for most tests of a case, then the generic KCF is the most likely to find, except for cases 8 and 12.

How can we explain the behaviour in cases 8 and 12? For these two cases one matrix is the zero matrix. This means that $tol \equiv \min(\texttt{EPSUA}, \texttt{EPSUB}) \cdot \texttt{GAP} = 0$ implying that $\epsilon_n > tol$ already for the smallest perturbation, which in turn explains why case transitions occur already for the smallest perturbation. Since either `EPSUA` or `EPSUB` is zero, all singular values in the perturbed zero matrix will be interpreted as non-zero, explaining why A or B are interpreted as a full rank matrix already for the smallest perturbations. Notice also the "jumps"

Table 4: Computed Kronecker structures and transitions of 100 perturbed non-generic 2×3 pencils. The size ϵ_n of each perturbation is shown above the corresponding arrow.

$$1':\ L_1 \oplus R_1 \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (81) \\ L_1 \oplus R_1 & (19) \end{array}\right\} \xrightarrow{10^{-3}} \left\{\begin{array}{ll} L_2 & (98) \\ L_1 \oplus R_1 & (2) \end{array}\right\} \xrightarrow{10^{-2}} L_2$$

$$2:\ L_1 \oplus J_1 \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_2 & (18) \\ L_1 \oplus J_1 & (82) \end{array}\right\} \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (98) \\ L_1 \oplus J_1 & (2) \end{array}\right\} \xrightarrow{10^{-3}} L_2$$

$$6:\ L_1 \oplus N_1 \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (92) \\ L_1 \oplus R_1 & (6) \\ L_1 \oplus N_1 & (2) \end{array}\right\} \xrightarrow{10^{-3}} \left\{\begin{array}{ll} L_2 & (99) \\ L_1 \oplus R_1 & (1) \end{array}\right\} \xrightarrow{10^{-2}} L_2$$

$$5:\ L_0 \oplus R_2 \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (63) \\ L_1 \oplus R_1 & (28) \\ L_0 \oplus R_2 & (9) \end{array}\right\} \xrightarrow{10^{-3}} \left\{\begin{array}{ll} L_2 & (95) \\ L_1 \oplus R_1 & (4) \\ L_0 \oplus R_2 & (1) \end{array}\right\} \xrightarrow{10^{-2}} \left\{\begin{array}{ll} L_2 & (99) \\ L_1 \oplus R_1 & (1) \end{array}\right\} \xrightarrow{10^{-1}} L_2$$

$$4':\ L_0 \oplus J_1 \oplus R_1 \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_2 & (1) \\ L_1 \oplus R_1 & (17) \\ L_0 \oplus J_1 \oplus R_1 & (82) \end{array}\right\} \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (82) \\ L_1 \oplus R_1 & (16) \\ L_1 \oplus J_1 & (1) \\ L_0 \oplus J_1 \oplus R_1 & (1) \end{array}\right\} \xrightarrow{10^{-3}} L_2$$

$$10':\ L_0 \oplus N_1 \oplus R_1 \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_2 & (90) \\ L_0 \oplus N_1 \oplus R_1 & (10) \end{array}\right\} \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (98) \\ L_0 \oplus R_2 & (2) \end{array}\right\} \xrightarrow{10^{-3}} L_2$$

$$4:\ L_0 \oplus J_2 \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_2 & (22) \\ L_0 \oplus J_1 \oplus R_1 & (17) \\ L_0 \oplus J_2 & (61) \end{array}\right\} \xrightarrow{10^{-4}} L_2$$

$$10:\ L_0 \oplus N_2 \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_2 & (10) \\ L_0 \oplus N_1 \oplus R_1 & (15) \\ L_0 \oplus N_2 & (75) \end{array}\right\} \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (99) \\ L_0 \oplus N_2 & (1) \end{array}\right\} \xrightarrow{10^{-3}} L_2$$

$$7:\ L_0 \oplus J_1 \oplus N_1 \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_2 & (5) \\ L_1 \oplus N_1 & (13) \\ L_0 \oplus J_1 \oplus N_1 & (82) \end{array}\right\} \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (92) \\ L_1 \oplus R_1 & (6) \\ L_1 \oplus J_1 & (2) \end{array}\right\} \xrightarrow{10^{-3}} \left\{\begin{array}{ll} L_2 & (97) \\ L_1 \oplus R_1 & (3) \end{array}\right\} \xrightarrow{10^{-2}} L_2$$

$$7':\ L_0 \oplus L_1 \oplus L_0^T \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_2 & (3) \\ L_1 \oplus R_1 & (7) \\ L_1 \oplus N_1 & (12) \\ L_0 \oplus L_1 \oplus L_0^T & (78) \end{array}\right\} \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (93) \\ L_1 \oplus R_1 & (7) \end{array}\right\} \xrightarrow{10^{-3}} \left\{\begin{array}{ll} L_2 & (98) \\ L_1 \oplus R_1 & (2) \end{array}\right\} \xrightarrow{10^{-2}} L_2$$

$$3:\ L_0 \oplus 2J_1 \xrightarrow{10^{-10}} L_2$$

$$11:\ L_0 \oplus 2N_1 \xrightarrow{10^{-10}} L_2$$

$$9':\ 2L_0 \oplus L_1^T \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_0 \oplus R_2 & (3) \\ L_0 \oplus J_1 \oplus R_1 & (34) \\ L_0 \oplus N_2 & (15) \\ 2L_0 \oplus L_1^T & (48) \end{array}\right\} \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (86) \\ L_1 \oplus R_1 & (8) \\ L_0 \oplus R_2 & (4) \\ L_1 \oplus J_1 & (2) \end{array}\right\} \xrightarrow{10^{-3}} L_2$$

$$9:\ 2L_0 \oplus R_1 \oplus L_0^T \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_0 \oplus R_2 & (2) \\ L_0 \oplus J_1 \oplus R_1 & (33) \\ L_0 \oplus N_1 \oplus R_1 & (16) \\ 2L_0 \oplus R_1 \oplus L_0^T & (49) \end{array}\right\} \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (80) \\ L_1 \oplus R_1 & (17) \\ L_0 \oplus R_2 & (1) \\ L_1 \oplus J_1 & (1) \\ L_0 \oplus J_1 \oplus R_1 & (1) \end{array}\right\} \xrightarrow{10^{-3}} \left\{\begin{array}{ll} L_2 & (96) \\ L_1 \oplus R_1 & (4) \end{array}\right\} \xrightarrow{10^{-2}} L_2$$

$$8:\ 2L_0 \oplus J_1 \oplus L_0^T \xrightarrow{10^{-10}} L_1 \oplus N_1 \xrightarrow{10^{-6}} \left\{\begin{array}{ll} L_2 & (2) \\ L_1 \oplus N_1 & (98) \end{array}\right\} \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_2 & (24) \\ L_1 \oplus R_1 & (2) \\ L_1 \oplus N_1 & (54) \end{array}\right\} \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (86) \\ L_1 \oplus R_1 & (14) \end{array}\right\} \xrightarrow{10^{-3}} L_2$$

$$12:\ 2L_0 \oplus N_1 \oplus L_0^T \xrightarrow{10^{-10}} L_1 \oplus J_1 \xrightarrow{10^{-5}} \left\{\begin{array}{ll} L_2 & (18) \\ L_1 \oplus J_1 & (82) \end{array}\right\} \xrightarrow{10^{-4}} \left\{\begin{array}{ll} L_2 & (98) \\ L_1 \oplus J_1 & (2) \end{array}\right\} \xrightarrow{10^{-3}} L_2$$

$$13:\ 3L_0 \oplus 2L_0^T \xrightarrow{10^{-10}} L_2$$

these transitions correspond to in the closure graph. The argumentation here also explains why the zero pencil turns generic for the smallest perturbation.

We end this section by briefly discussing how the case invariances and transitions are affected by the choice of the fixed relative accuracy of the input data (`EPSU`). If we choose `EPSU` $= \epsilon_n$ then `GUPTRI` will retrieve the non-generic structure we started from for each ϵ_n considered. Notice that the distance from the input pencil to the computed Kronecker structure will normally be of size $O(\mathtt{EPSU}\cdot\|(A,B)\|_E)$ [8]. Increasing `EPSU` means that the case invariances will remain longer before any case transition take place. Decreasing `EPSU` will impose the generic structure sooner. For example, with `EPSU` equal to the relative machine precision and $\epsilon_n > tol$, `GUPTRI` will always extract the generic structure. This corresponds to the fact that in infinite precision arithmetic any non-generic $A - \lambda B$ can be made generic with arbitrary small perturbations. Moreover, travelling upwards in the closure hierarchy can always be effected with arbitrary small perturbations, while travelling downwards may require much larger perturbations.

4 Imposing Non-Generic Structures by Perturbing a Generic Pencil

In this section we study computable normwise bounds for the smallest perturbations $(\delta A, \delta B)$ of a generic 2-by-3 pencil $A - \lambda B$ such that $(A+\delta A) - \lambda(B+\delta B)$ has a specific non-generic Kronecker structure chosen from the 17 non-generic cases discussed earlier. Our goal is to find the closest non-generic pencil and the closest pencil with a specified non-generic Kronecker structure of a 2-by-3 generic pencil. We consider two approaches to impose a non-generic structure. First we derive explicit expressions for the perturbations that transfer $A - \lambda B$ to a specified non-generic form. Secondly, we have modified `GUPTRI` to be able to compute a specified Kronecker structure.

4.1 Explicit Perturbations to Impose Non-Generic Structures

We have seen in Section 2 that by making A and B more rank deficient and increasing their "common nullspace(s)" we can generate non-generic pencils with higher codimension. Here we elaborate on this fact and derive explicit expressions for the perturbations required to turn an arbitrary generic pencil into each of the 17 non-generic cases. The norms of these explicit expressions (measured as $\|(\delta A, \delta B)\|_E$) are upper bounds for the smallest perturbations required. Indeed, for 11 of the structures, the norms are the exact sizes of the smallest perturbations required.

We need the following notation. The size of the smallest perturbations $(\delta A, \delta B)$ such that $R[A + \delta A, B + \delta B, i]$ (2.3) of size $(i + 2)m \times (i + 1)n$ has a

k-dimensional column nullspace is defined as

$$d_k(R[A,B,i]) = \min_{(\delta A,\delta B)} \{\|(\delta A,\delta B)\|_E : \mathrm{n}(R[A+\delta A,B+\delta B,i]) = k\}, \quad (4.1)$$

where δA and δB vary over all m-by-n matrices with complex (or real) entries. Similarly, we define $d_k(L[A,B,i])$ as the size of the smallest perturbations that impose a k-dimensional row nullspace on $L[i]$ (2.5). When it is clear from context we use the abbreviated notation $d_k(R[i])$ and $d_k(L[i])$. Also, let $d_k(A)$ denote the size of the smallest perturbations such that $\mathrm{rank}(A+\delta A) = \min(m,n) - k$.

In general, to find $d_k(R[i])$ (or $d_k(L[i])$) is a type of a structured singular value problem. For $i \geq 1$ it is an open problem to find explicit expressions for $d_k(R[i])$ and $d_k(L[i])$. The following theorem summarizes some of their properties for the case $m = 2, n = 3$ and $k = 1$:

Theorem 4.1 *For a generic 2-by-3 pencil (A, B) the following inequalities hold:*

$$0 \equiv d_1(R[2]) < d_1(R[1]) \leq d_1(R[0]), \quad (4.2)$$

$$d_1(R[1]) \leq d_1(A), \quad d_1(R[1]) \leq d_1(B), \quad (4.3)$$

$$d_1(R[1]) < d_1(L[1]) \leq d_1(L[0]), \quad (4.4)$$

$$d_1(A) < d_2(A), \quad d_1(B) < d_2(B), \quad d_1(R[0]) < d_2(R[0]). \quad (4.5)$$

Proof. From Theorem 2.1 it follows that $d(R[2]) = 0$ for all 2-by-3 pencils (generic or non-generic). Decreasing the rank of the 4-by-3 $R[A,B,0]$ by one gives that $R[A+\delta A, B+\delta B, 0]$ has only two linearly independent columns. The same perturbations make the 6-by-6 matrix $R[A+\delta A, B+\delta B, 1]$ rank deficient (a rank drop from six to four) showing that (4.2) holds. Similarly, decreasing the rank of A (or B) by one means that $A+\delta A$ (or $B+\delta B$) only has one linearly independent row. For the same perturbations $R[A+\delta A, B+\delta B, 1]$ is rank deficient with only one of the two first (or two last) rows linearly independent, resulting in the inequalities (4.3).

$L[1]$ is row rank deficient if and only if there exists at least one L_0^T or L_1^T block in the KCF. Since all KCFs with at least one L_0^T block or one L_1^T block have both A and B rank deficient (see Table 3), there will always exist a strictly smaller perturbation of size $d_1(A)$ that only lowers the rank in A. (The similar is of course true for B.) Now applying inequality (4.3) proves the first part of (4.4). The last part follows from similar arguments as proving $d_1(R[1]) \leq d_1(R[0])$ above. The inequalities (4.5) follow from the definition of $d_k(\cdot)$. $\Box$

Theorem 4.1 will be used to identify the closest non-generic Kronecker structure of a generic 2-by-3 pencil. Notice that in general we cannot say anything about the relationship between $d_1(R[0])$ and $d_1(A)$ or $d_1(B)$ (see explicit expressions below). By varying α and β in

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & \alpha \end{bmatrix}, \quad B = \begin{bmatrix} \beta & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix},$$

(i.e., a generic $A - \lambda B$ for non-zero α and β) we show that any of them can be the smallest quantity (see Table 5).

Table 5: The quantities $d_1(A)$, $d_1(B)$ and $d_1(R[0])$ for three examples.

Parameters	$d_1(A)$	$d_1(B)$	$d_1(R[0])$
$\alpha = \beta = 1$	1.000	1.000	**0.765**
$\alpha = 0.1,\ \beta = 1$	**0.100**	1.000	0.451
$\alpha = 1,\ \beta = 0.1$	1.000	**0.100**	0.451

The following explicit expressions, derived from the Eckart–Young and Mirsky theorem for finding the closest matrix of a given rank (e.g. see[10]), appear in our explicit bounds discussed next:

$$d_1(A) = \sigma_{\min}(A), \quad d_1(B) = \sigma_{\min}(B),$$

$$d_2(A) = \|A\|_E, \quad d_2(B) = \|B\|_E,$$

$$d_1(R[A,B,0]) = \sigma_{\min}(R[0]), \quad d_1(L[A,B,0]) = \sigma_{\min}(L[0]),$$

$$d_2(R[A,B,0]) = (\sigma^2_{\min-1}(R[0]) + \sigma^2_{\min}(R[0]))^{1/2}.$$

Here, $\sigma_{\min}(X)$ and $\sigma_{\min-1}(X)$ (with $\sigma_{\min}(X) \le \sigma_{\min-1}(X)$) denote the two smallest non-zero singular values of (a full rank) matrix X.

4.1.1 Tractable Perturbations

In order to make the problem more tractable we (first) put restrictions on allowable perturbations. We can compute a perturbation $\delta A - \lambda \delta B$ such that $(A+\delta A) - \lambda(B+\delta B)$ is guaranteed to fall on the closure of the manifold (orbit) of a certain KCF. (Necessary conditions on the required perturbations are given in the labeled closure graph in Figure 2.) If the KCF found is the intended KCF, then the perturbation is said to be tractable. If the KCF found is even more non-generic (i.e., its orbit has higher codimension but belongs to the closure of the intended manifold), then the perturbation is defined intractable. In other words, a tractable perturbation finds the generic KCF (i.e., the least non-generic KCF) in the closure of the manifold of the intended KCF. An intractable perturbation finds any other structure in the closure of the same manifold, i.e., any structure that can be found by traveling along the arcs (downwards) from the intended KCF in the closure graph in Figure 1.

When computing perturbations such that $(A + \delta A) - \lambda(B + \delta B)$ is given a non-generic KCF, we compute δA and δB such that one or more of the geometric characteristics presented in Table 3 for $(A + \delta A) - \lambda(B + \delta B)$ differ from the characteristics of the generic (A, B). In other words, we put restrictions on the size of the perturbed pencil's nullspaces so that at least one of them is larger than for the generic case. The space given by this restriction may contain several non-generic matrix pencils. For example, if we restrict the set of pencils to those who have a rank deficiency in the A-part, this space contains all pencils that fulfill the condition $\mathrm{rank}(A) < 2$. However, if we compute a perturbation such that $\mathrm{rank}(A+\delta A) < 2$, the perturbed pencil will most likely be the generic (least non-generic) KCF with a rank-deficient A-part, i.e., $L_1 \oplus J_1$. This corresponds to the KCF with rank-deficient A-part whose orbit has the smallest codimension and the corresponding perturbation $(\delta A, \delta B)$ is tractable. The perturbation is intractable if $(A + \delta A) - \lambda(B + \delta B)$ has any KCF (with $\mathrm{rank}(A) < 2$) that is more non-generic than $L_1 \oplus J_1$. The set of possible structures are the ones that are in the closure of $\mathrm{orbit}(L_1 \oplus J_1)$.

Eleven of the 17 non-generic structures (2, 6, 5, 7, 7', 3, 11, 9', 8, 12, and 13) are imposed by (minimal) tractable perturbations that effectuate one of the following rank-decreasing operations:

- Rank drop in A and/or B by one or two.
- Rank drop in $R[A, B, 0]$ by one or two, i.e., imposing a common one or two dimensional column nullspace.
- Rank drop in $L[A, B, 0]$ by one, i.e., imposing a common row nullspace.

In Table 6 the size of the perturbations required to impose each of the eleven singular structures are displayed. When both $d_i(A)$ and $d_j(B)$ are involved, the size of the total perturbations is $(d_i^2(A) + d_j^2(B))^{1/2}$. The singular cases are reported in increasing codimension order (see Table 3). Since all these perturbations are made as the smallest possible to impose the required ranks on $A, B, R[0]$ or $L[0]$, these bounds are attained for each non-generic form, i.e., the strongest possible, which is equivalent to that the bounds in Table 6 also are lower bounds. That these perturbations really give the forms shown in the table follows from the fact that we here only are considering tractable perturbations and these are the least non-generic forms that have the imposed rank-deficiencies (see Table 3). For example, by imposing a one dimensional rank drop in A we have restricted the 12-dimensional space to a space that contains a subset of all non-generic pencils. Since the perturbation is supposed to be tractable, the KCF found is the least non-generic in that space, i.e., $L_1 \oplus J_1$.

The rank decreasing operations performed in Table 6 "affect the codimension(s)" in the following way: a rank drop by one in A, B or $R[0]$ increases the

Table 6: Minimal perturbations of a generic pencil to impose 11 of the 17 non-generic structures.

Case	KCF	Cod(·)	$d_1(A)$	$d_1(B)$	$d_1(R[0])$	$d_1(L[0])$	$d_2(A)$	$d_2(B)$	$d_2(R[0])$
2	$L_1 \oplus J_1$	2	×						
6	$L_1 \oplus N_1$	2		×					
5	$L_0 \oplus R_2$	2			×				
7	$L_0 \oplus J_1 \oplus N_1$	4	×	×					
$7'$	$L_0 \oplus L_1 \oplus L_0^T$	5				×			
3	$L_0 \oplus 2J_1$	6					×		
11	$L_0 \oplus 2N_1$	6						×	
$9'$	$2L_0 \oplus L_1^T$	6							×
8	$2L_0 \oplus J_1 \oplus L_0^T$	8		×			×		
12	$2L_0 \oplus N_1 \oplus L_0^T$	8	×					×	
13	$3L_0 \oplus 2L_0^T$	12					×	×	

Table 7: Compound perturbations: Non-generic structures imposed by transiting via a non-generic form.

Case	KCF	Cod(·)	Transit KCF	$d_1(R[\tilde{A}, \tilde{B}, 0])$
$4'$	$L_0 \oplus J_1 \oplus R_1$	3	$L_1 \oplus J_1$	×
$10'$	$L_0 \oplus N_1 \oplus R_1$	3	$L_1 \oplus N_1$	×

codimension by two, a rank drop by one in $L[0]$ increases the codimension by five, and a rank drop by two in A, B or $R[0]$ increases the codimension by six.

Two of the remaining six non-generic forms (4' and 10') are imposed by transiting via a non-generic form as shown in Table 7. For example, to derive perturbations of the generic $A - \lambda B$ that turn $(A + \delta A, B + \delta B)$ non-generic with KCF $L_0 \oplus J_1 \oplus R_1$ we have $(\delta A, \delta B) = (\delta A_1, \delta B_1) + (\delta A_2, \delta B_2)$, where $(\delta A_1, \delta B_1)$ is the smallest perturbation that lowers the rank of A (i.e., $\|(\delta A_1, \delta B_1)\|_E = d_1(A)$, $\delta B_1 = 0_{2\times 3}$) and $(\delta A_2, \delta B_2)$ is the smallest perturbation that imposes a common column nullspace on $(A + \delta A_1, B + \delta B_1)$ (i.e., $\|(\delta A_2, \delta B_2)\|_E = d_1(R[A + \delta A_1, B + \delta B_1, 0])$). In Table 7 we show how these forms are constructed. The size of the compound (total) perturbations $(\delta A, \delta B)$ for the two cases are obtained by adding the perturbations in Table 6 and Table 7. $\tilde{A} = A + \delta A_1$ and $\tilde{B} = B + \delta B_1$ in Table 7 represent the "transit" non-generic pencil. A rank drop by one in $R[\tilde{A}, \tilde{B}, 0]$ in Table 7 increases the codimension by one.

The last four non-generic structures (1', 4, 10 and 9) require perturbations to parts of the GUPTRI form of a transiting pencil $\tilde{A} - \lambda \tilde{B}$:

$$P^H(\tilde{A} - \lambda \tilde{B})Q = \tilde{S} - \lambda \tilde{T} \equiv \begin{bmatrix} \tilde{s}_{11} & \tilde{s}_{12} & \tilde{s}_{13} \\ 0 & \tilde{s}_{22} & \tilde{s}_{23} \end{bmatrix} - \lambda \begin{bmatrix} \tilde{t}_{11} & \tilde{t}_{12} & \tilde{t}_{13} \\ 0 & \tilde{t}_{22} & \tilde{t}_{23} \end{bmatrix}, \quad (4.6)$$

where some $\tilde{s}_{ij}, \tilde{t}_{ij}$ may be zero. The size of the perturbations $(\delta\tilde{S}, \delta\tilde{T})$ imposed on $\tilde{S}$ and/or $\tilde{T}$ are displayed in Table 8. Case 1', which transits via the `GUPTRI` form of L_2, is retrieved by imposing a common column nullspace of the A- and B-parts of the deflated 1-by-2 pencil $[\tilde{s}_{22} \quad \tilde{s}_{23}] - \lambda[\tilde{t}_{22} \quad \tilde{t}_{23}]$. For cases 4 and 10 we retrieve the requested structures by setting elements $\tilde{s}_{12} = 0$ and $\tilde{t}_{12} = 0$, respectively, in the `GUPTRI` forms of $\tilde{A} - \lambda\tilde{B}$ (4.6). For case 4 we impose a zero multiple eigenvalue in $\tilde{A} - \lambda\tilde{B}$. Similarly, a multiple eigenvalue is imposed at infinity for case 10. In other words, $J_1 \oplus R_1$ and $N_1 \oplus R_1$ in $\tilde{A} - \lambda\tilde{B}$ are turned J_2 and N_2, respectively. Case 9 is obtained by giving the A- and B-parts of the L_1 block in $\tilde{A} - \lambda\tilde{B}$ a common column nullspace, which turns L_1 into $L_0 \oplus R_1$. Since P and Q in (4.6) are unitary the perturbations imposed on $\tilde{A}$ and $\tilde{B}$ are of the same size as $\delta\tilde{S}$ and $\delta\tilde{T}$. The size of the compound (total) perturbations $(\delta A, \delta B)$ for the four cases are obtained by adding the appropriate perturbations in tables 6, 7 and 8. The perturbations explicitly imposed for the four cases in Table 8 increase the codimensions by one, except for case 9 where the rank drop by one increases the codimension by two.

Table 8: Compound perturbations: Non-generic structures imposed by perturbing the `GUPTRI` form (denoted Transit form) of the generic or some non-generic pencils.

Case	KCF	Cod$(\cdot)$	Transit form	$d_1\left(\begin{bmatrix} \tilde{s}_{22} & \tilde{s}_{23} \\ \tilde{t}_{22} & \tilde{t}_{23} \end{bmatrix}\right)$	$d_1\left(\begin{bmatrix} \tilde{s}_{12} & \tilde{s}_{13} \\ \tilde{t}_{12} & \tilde{t}_{13} \end{bmatrix}\right)$	$\tilde{s}_{12}$	$\tilde{t}_{12}$
$1'$	$L_1 \oplus R_1$	1	L_2	$\times$			
4	$L_0 \oplus J_2$	4	$L_0 \oplus J_1 \oplus R_1$			$\times$	
10	$L_0 \oplus N_2$	4	$L_0 \oplus N_1 \oplus R_1$				$\times$
9	$2L_0 \oplus R_1 \oplus L_0^T$	7	$L_0 \oplus L_1 \oplus L_0^T$		$\times$		

The compound perturbations discussed above are all supposed to be tractable, but are not necessarily optimal. A summary of the explicit perturbations in tables 6 - 8 is displayed in a perturbation graph in Figure 3, where the nodes are placed at the same positions as in the closure graph (Figure 1). The paths to a node show different ways to generate the tractable perturbation required to find the KCF of the node, starting from a generic $A - \lambda B$. Notice that some arcs are marked with a bullet and the corresponding paths from a generic pencil to a destination KCF generate perturbations which are not necessarily optimal (compound perturbations from Table 7 and Table 8). All other paths correspond to optimal perturbations from Table 6. We clarify the notation in Figure 3 with two examples. Let $(\delta A_1, \delta B_1)$ denote the optimal perturbation of size $d_1(A)$ that for a generic $A - \lambda B$ gives $\tilde{A} - \lambda\tilde{B} = (A + \delta A_1) - \lambda(B + \delta B_1)$ the Kronecker structure $L_1 \oplus J_1$. Similarly, let $(\delta\tilde{A}_2, \delta\tilde{B}_2)$ denote the optimal perturbation of size $d_1(R[\tilde{A}, \tilde{B}, 0])$ that moves $\tilde{A} - \lambda\tilde{B}$ to a pencil with Kronecker structure

$L_0 \oplus J_1 \oplus R_1$. Then $(\delta A_1 + \delta \tilde{A}_2, \delta B_1 + \delta \tilde{B}_2)$ is not necessarily the optimal perturbation for moving a generic pencil to orbit$(L_0 \oplus J_1 \oplus R_1)$. Therefore the arc to orbit$(L_0 \oplus J_1 \oplus R_1)$ is marked with a bullet. On the other hand, adding the perturbations going from orbit(L_2) to orbit$(L_0 \oplus 2J_1)$ via orbit$(L_1 \oplus J_1)$ give us the optimal perturbation, which is already shown in Table 6.

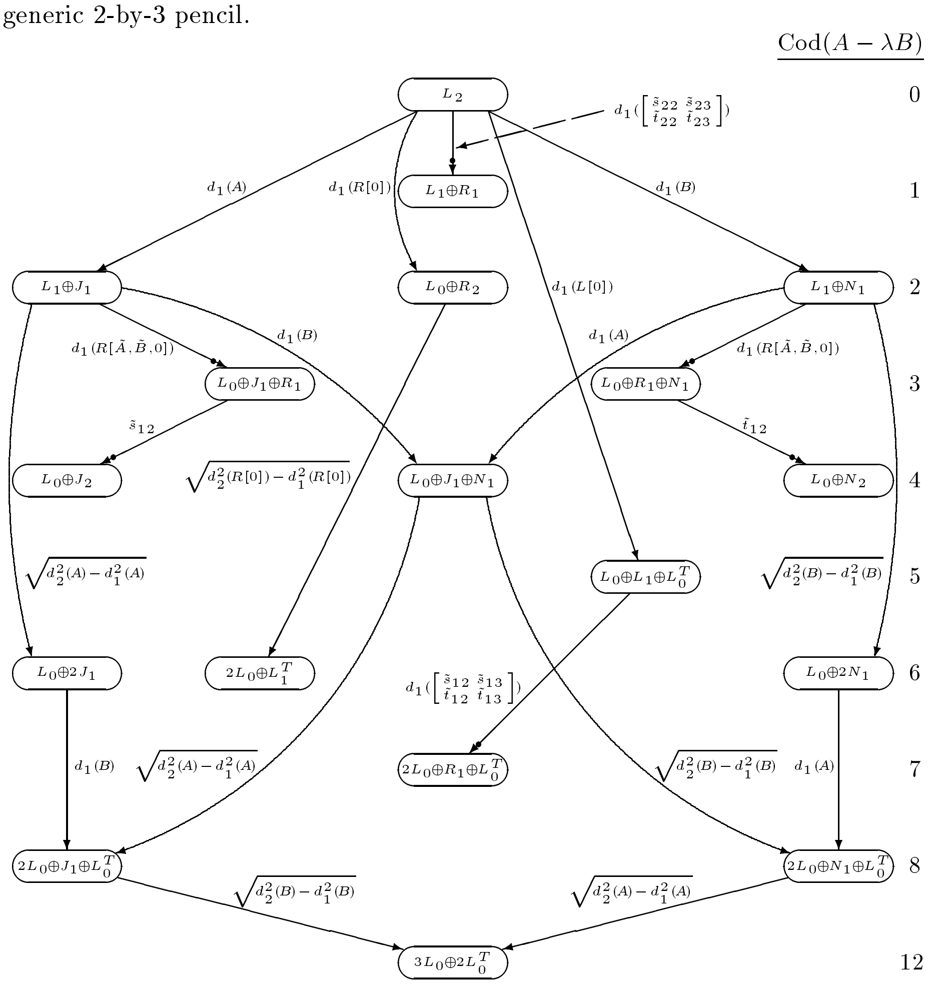
In order to relate our explicit perturbations to the (labeled) closure graph we consider 2-dimensional rank drops in Table 6 as results of two 1-dimensional rank drops. In practice, these 2-dimensional rank drops are computed directly. Some of the perturbations in Table 6 do not give a unique path in the graph, since the generic $A - \lambda B$ in some cases is perturbed in A and B simultaneously. For these cases all alternative paths are shown in the graph, e.g. there are three different paths to $2L_0 \oplus J_1 \oplus L_0^T$ and all of them correspond to the same perturbation (in infinite arithmetic) of size $(d_2^2(A) + d_1^2(B))^{1/2}$. From the construction of the explicit perturbations it follows that each arc in the perturbation graph connects a KCF with another KCF within its orbit's closure. Therefore, for each arc in the perturbation graph it exists a corresponding path in the closure graph. It is of course possible to find other paths in the (labeled) closure graph that give tractable perturbations.

The sizes of the perturbations are shown on the corresponding arcs in the graph, with notation as before. The reason for perturbation sizes such as $\sqrt{d_2^2(A) - d_1^2(A)}$ is that the total perturbation needed for this 2-dimensional rank-drop in A is $d_2(A)$ (as shown in Table 6), but it is here shown as a further perturbation of a case where a perturbation of size $d_1(A)$ already has imposed a 1-dimensional rank-drop in A.

For each case in Table 6 that in Figure 3 is shown as a compound perturbation, even though it can be computed directly, the size of the total perturbation is the square root of the sum of the squares of the sizes of the components of the perturbation. For example, the case $2L_0 \oplus J_1 \oplus L_0^T$ is found by a compound perturbation $(\delta A, \delta B) = (\delta A_1, \delta B_1) + (\delta A_2, \delta B_2) + (\delta A_3, \delta B_3)$, where $\|(\delta A_1, \delta B_1)\|_E = d_1(A)$, $\|(\delta A_2, \delta B_2)\|_E = \sqrt{d_2^2(A) - d_1^2(A)}$, and $\|(\delta A_3, \delta B_3)\|_E = d_1(B)$. The size of the total perturbation is $\|(\delta A, \delta B)\|_E = (d_1^2(A) + (d_2^2(A) - d_1^2(A)) + d_1^2(B))^{1/2} = (d_2^2(A) + d_1^2(B))^{1/2}$. Notably, since the perturbation $d_1(A) = \sigma_{\min}(A)$ and $d_2(A) = \|A\|_E = (\sigma_{\min-1}^2(A) + \sigma_{\min}^2(A))^{1/2}$ the size $\sqrt{d_2^2(A) - d_1^2(A)}$ is equal to $\sigma_{\min-1}(A)$.

For each compound perturbation in tables 7 and 8, the size of the total perturbation is found by adding the components of the perturbation and then computing the norm of the resulting perturbation. However, an upper bound on the size of the compound perturbation can be achieved by adding the sizes of the components of the perturbation. For example, $L_0 \oplus J_1 \oplus R_1$ is found by the compound perturbation $(\delta A, \delta B) = (\delta A_1, \delta B_1) + (\delta A_2, \delta B_2)$, where $\|(\delta A_1, \delta B_1)\|_E = d_1(A)$ and $\|(\delta A_2, \delta B_2)\|_E = d_1(R[\tilde{A}, \tilde{B}, 0])$, and an upper bound on $\|(\delta A, \delta B)\|_E$ is $d_1(A) + d_1(R[\tilde{A}, \tilde{B}, 0])$.

Figure 3: A graph displaying the tractable perturbations in tables 6 – 8 of a generic 2-by-3 pencil.



4.1.2 Intractable Perturbations and the Closest Non-Generic Structure

The following example shows a situation where the perturbations incidentally create extra non-generic characteristics that raise the codimension of the perturbed pencil further than devised.

$$A = \begin{bmatrix} 0 & \epsilon_1 & 0 \\ 0 & 0 & \epsilon_2 \end{bmatrix}, \quad B = \begin{bmatrix} \epsilon_3 & 0 & 0 \\ 0 & \epsilon_4 & 0 \end{bmatrix}, \quad \epsilon_2 = \min_i \epsilon_i > 0. \tag{4.7}$$

Suppose we are looking for the minimal perturbations that impose the structure $L_1 \oplus J_1$ (case 2). They are of size $d_1(A)$ with

$$\delta A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\epsilon_2 \end{bmatrix}, \quad \delta B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Incidentally, δA and δB also lower the rank of $R[0]$. (For this example, δA and δB are the minimal perturbations that cause the rank drop, i.e., $d_1(A) = d_1(R[0])$ and the minima are attained for the same perturbations.) This fact implies that the perturbations aimed to impose the non-generic structure $L_1 \oplus J_1$ (with codimension two) result in a perturbed pencil with two zero eigenvalues corresponding to the structure $L_0 \oplus J_2$ with codimension four (case 4). One possible remedy is to further perturb the undesired non-generic pencil. To obtain $L_1 \oplus J_1$ we add, for example, the perturbations

$$\delta A' = \begin{bmatrix} \delta & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \delta B' = \begin{bmatrix} 0 & 0 & \delta \\ 0 & 0 & 0 \end{bmatrix},$$

to $(A+\delta A, B+\delta B)$, where $\delta > 0$ is an arbitrary small number. These perturbations remove the common column nullspace ($\delta B'$) and the multiple eigenvalue at zero ($\delta A'$), making the compound perturbations tractable. If we start to look for the smallest perturbations of (A, B) that impose a common column nullspace that normally would generate the structure $L_0 \oplus R_2$ (case 5) we also get intractable perturbations and (in this case) the same structure $L_0 \oplus J_2$. We can also see from the closure graph in Figure 1 that $L_0 \oplus J_2$ is in the closure of each of the two orbits defined by $L_1 \oplus J_1$ and $L_0 \oplus R_2$.

Now, we turn to the problem of finding the closest non-generic Kronecker structure of a generic 2-by-3 pencil. Assume all inequalities relating to $d_1(R[1])$ in Theorem 4.1 are strict. Then the corresponding $R[A+\delta A, B+\delta B, 1]$ is rank deficient, and for all perturbations of size $\leq d_1(R[1])$, the (perturbed) matrices $A+\delta A, B+\delta B, R[A+\delta A, B+\delta B, 0]$ and $L[A+\delta A, B+\delta B, 0]$ must be of full rank, which correspond to the case $L_1 \oplus R_1$. Since all other non-generic cases require rank-deficiency in at least on of the matrices $A+\delta A, B+\delta B, R[A+\delta A, B+\delta B, 0]$ or $L[A+\delta A, B+\delta B, 0]$ (see necessary conditions in the labeled closure graph in Figure 2 or Table 3), we can formulate the following corollary.

Corollary 4.1 *If the inequalities (4.2) and (4.3) in Theorem 4.1 are strict,* $L_1 \oplus R_1$ *with codimension one (case 1') is the closest (unique) non-generic structure on distance* $d_1(R[1])$.

The presumptions of Corollary 4.1 are sufficient (but not necessary) to identify tractable perturbations that lowers the rank of $R[1]$. If equality holds in any of the inequalities of Theorem 4.1 (for the same perturbations $(\delta A, \delta B)$) we are faced with intractable perturbations which will result in non-generic structures with higher codimensions. We collect the different cases in the following corollary, where we list the closest Kronecker structure and the corresponding equality conditions. Notice that strict inequalities are assumed otherwise.

Corollary 4.2 *Assume strict inequalities hold in Theorem 4.1 when nothing else is stated. Then, if*

1. $d_1(R[1]) = d_1(R[0])$, $L_0 \oplus R_2$ *(case 5) is the closest non-generic form.*
2. $d_1(R[1]) = d_1(A)$, $L_1 \oplus J_1$ *(case 2) is the closest non-generic form.*
3. $d_1(R[1]) = d_1(B)$, $L_1 \oplus N_1$ *(case 6) is the closest non-generic form.*

All forms in Corollary 4.2 have codimension two. Notice that if it also exists some perturbations on distance $d_1(R[1])$ which do not lower the rank of $R[0], A$ and B, respectively, $L_1 \oplus R_1$ is also at the same distance as $L_0 \oplus R_2$, $L_1 \oplus J_1$ and $L_1 \oplus N_1$ for the three cases considered.

Assume that we can have equality in different combinations of the inequalities of Theorem 4.1. As before, we collect the possible cases in a corollary.

Corollary 4.3 *Assume two inequalities in Theorem 4.1 are satisfied with equality for the same perturbations* $(\delta A, \delta B)$. *Then, if*

1. $d_1(R[1]) = d_1(R[0]) = d_1(A)$, $L_0 \oplus J_1 \oplus R_1$ *(case 4' with codimension 3) or* $L_0 \oplus J_2$ *(case 4 with codimension 4) is the closest non-generic structure.*
2. $d_1(R[1]) = d_1(R[0]) = d_1(B)$, $L_0 \oplus R_1 \oplus N_1$ *(case 10' with codimension 3) or* $L_0 \oplus N_2$ *(case 10 with codimension 4) is the closest non-generic Kronecker structure.*

Notice that cases 4 and 10 have higher codimensions than cases 4' and 10', respectively, but the same algebraic characteristics in terms of the rank of $R[k]$ and $L[k]$ matrices as (see Table 3). The reason is that the 2-by-2 regular parts of cases 4 and 10 have one Jordan block with both eigenvalues specified, which increase the codimension by one compared to cases 4' and 10' (both with one eigenvalue unspecified).

The remark following Corollary 4.2, regarding a non-unique closest Kronecker structure can also be extended to apply to Corollary 4.3.

In applications (e.g. computing the uncontrollable subspace) we are interested to find the most non-generic structure (with highest codimension) for a given size of the perturbations. Is it possible to find intractable perturbations that result in a closest 2-by-3 non-generic structure with codimension > 4? The answer is no since all other cases require a rank drop of at least two in A, B or $R[0]$ or a simultaneous rank drop in A and B. It always exists strictly smaller perturbations that drop the rank by one (see (4.5)). Similar arguments also exclude $L_0 \oplus J_1 \oplus N_1$ with codimension 4 from being the closest non-generic pencil.

4.1.3 Closest Non-Generic Structures to a Generic 1-by-2 Pencil

Since we do not know any explicit expression for $d_1(R[1])$ it is hard to construct examples that illustrate different situations described in Section 4.1.2. By considering 1-by-2 pencils we overcome this problem. A generic 1-by-2 pencil has the Kronecker structure $L_1 = [-\lambda \quad 1] = [0 \quad 1] - \lambda[1 \quad 0] \equiv A - \lambda B$. The non-generic structures of size 1-by-2 are $L_0 \oplus R_1, L_0 \oplus J_1, L_0 \oplus N_1$ and $2L_0 \oplus L_0^T$ with codimensions 1, 2, 2 and 4, respectively.

Which form(s) can be the closest non-generic structure of a generic 1-by-2 pencil?

- $L_0 \oplus R_1$ if it exists perturbations of size $d_1(R[0])$ which do not simultaneously decrease the rank of A or B. This is, e.g. fulfilled if $d_1(R[0]) < \min(d_1(A), d_1(B))$.
- $L_0 \oplus J_1$ if $d_1(R[0]) = d_1(A)$.
- $L_0 \oplus N_1$ if $d_1(R[0]) = d_1(B)$.

Moreover, $2L_0 \oplus L_0^T$ can never be the closest non-generic structure. The size of the minimal perturbations that turn A and B to zero matrices is $(d_1^2(A) + d_1^2(B))^{1/2}$.

The following example illustrates a case where $d_1(R[0]) = d_1(A) = d_1(B)$ and there exist perturbations of size $d_1(R[0])$ that do not simultaneously decrease the rank of A or B. Consequently, $L_0 \oplus R_1, L_0 \oplus J_1$, and $L_0 \oplus N_1$ are all the closest non-generic Kronecker structure.

Let $A = [1 \quad 1]$ and $B = [-1 \quad 1]$. Then $R[0]$ has the singular value decomposition

$$R[0] \equiv \begin{bmatrix} A \\ B \end{bmatrix} = U\Sigma V^T \equiv \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}.$$

$d_1(R[0]) (= \sqrt{2})$ is attained for the (minimal) perturbations

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

while $A + \delta A$ and $B + \delta B$ remain full rank matrices, resulting in $L_0 \oplus R_1$ as the closest non-generic structure. The perturbations

$$\begin{bmatrix} \delta A_1 \\ \delta B_1 \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} \delta A_2 \\ \delta B_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix}$$

of the same minimal size make $R[0], A$ and $R[0], B$ drop rank, respectively. These perturbations generate the non-generic structures $L_0 \oplus J_1$ and $L_0 \oplus N_1$, respectively.

4.2 Using GUPTRI to Impose Non-Generic Structures

We have modified `GUPTRI` so that, for an $m \times n$ generic pencil $A - \lambda B$ as input, it is possible to impose a generalized Schur form with a specified Kronecker structure. (The modified `GUPTRI` also work for imposing a Kronecker structure of higher codimension on any non-generic pencil.) Given the block indices that define the specified Kronecker structure (n_i's and r_i's of the RZ-staircase and LI-staircase forms [8]), `GUPTRI` imposes the necessary rank deflations in order to compute the specified (non-generic) structure. The perturbations induced by these rank deflations are usually tractable. If the perturbations imposed by `GUPTRI` are intractable, `GUPTRI` computes the corresponding non-generic structure of higher codimension. The resulting generalized Schur decomposition can be expressed in finite arithmetic as

$$P^H((A + \delta A) - \lambda(B + \delta B))Q = \begin{bmatrix} A_r - \lambda B_r & * & * \\ 0 & A_{reg} - \lambda B_{reg} & * \\ 0 & 0 & A_l - \lambda B_l \end{bmatrix}, \tag{4.8}$$

where $*$ denotes arbitrary conforming submatrices. Let δ_σ^2 denote the sum of the squares of all deleted singular values (imposed as zeros) during the reduction to GUPTRI form. Then δ_σ is an accurate estimate of $\|(\delta A, \delta B)\|_E$ in (4.8). One interpretation is that `GUPTRI` computes an exact generalized Schur decomposition (with the specified Kronecker structure) for a pencil $A' - \lambda B'$ within distance δ_σ from the input pencil $A - \lambda B$. Moreover, δ_σ is an upper bound on the distance from $A - \lambda B$ to the nearest pencil with the Kronecker structure specified as input to `GUPTRI`.

Furthermore, this give us a method for computing an upper bound on the distance from a generic m-by-n pencil to the closest non-generic pencil:

- Compute the structure indices (n_i's and r_i's of the RZ-staircase and LI-staircase forms [8]) for all q structurally different non-generic GUPTRI forms of size $m \times n$. This is a finite integer matching problem.

- Use the modified version of `GUPTRI` to impose the q non-generic structures:

$$A_i - \lambda B_i = P_i^H((A + \delta A_i) - \lambda(B + \delta B_i))Q_i, \quad i = 1, \ldots, q. \tag{4.9}$$

- Compute the matrix pairs corresponding to the q non-generic structures:

$$\hat{A}_i = P_i A_i Q_i^H, \quad \hat{B}_i = P_i B_i Q_i^H, \qquad i = 1, \ldots, q. \tag{4.10}$$

- Compute

$$\delta = \min_{1 \le i \le q} \delta_i, \quad \delta_i = \|(A - \hat{A}_i, B - \hat{B}_i)\|_E. \tag{4.11}$$

Now, δ is an upper bound on the closest non-generic pencil to $A - \lambda B$ and the δ_i's are upper bounds on the closest non-generic pencils with the Kronecker structure of $A_i - \lambda B_i$ in (4.9).

The method described above is quite expensive already for moderate m and n (see Section 5) but is perfectly parallel. In a distributed memory environment it is possible to distribute the block indices for the different Kronecker structures evenly over the p ($\le q$) processors. Each processor also hold A and B and computes its local δ using the method above. Finally, a global minimum operation over all p processors gives us δ in (4.11).

4.3 Computational Experiments on Random 2-by-3 Pencils

We have performed computational experiments on 100 random 2-by-3 pencils $A - \lambda B$. The elements of A and B are chosen uniformly distributed in $(0, 1)$. For each random pencil we impose the 17 non-generic structures using the two approaches discussed in sections 4.1 and 4.2.

Table 9 displays the mean values of perturbations required to impose each of the 17 non-generic forms for 100 random examples. We measure the perturbations for each example and non-generic form as $\|(A - \tilde{A}, B - \tilde{B})\|_E$, where $\tilde{A} - \lambda \tilde{B}$ denotes a non-generic pencil. The matrices A and B are normalized such that $\|A\|_E = \|B\|_E$ and $\|(A, B)\|_E = 1$.

Columns 2 and 3 of Table 9 show the δ_i's in (4.11) computed by modified `GUPTRI` for the pencils $A - \lambda B$ and $B - \mu A$, respectively. Column 4 shows the explicit perturbations of tables 6, 7 and 8. The explicit perturbations that are proved to be the smallest possible are marked with the superscript *.

In Table 10 we display the smallest perturbations (measured as above) required to impose non-generic forms of each possible codimension for the same 100 random 2-by-3 examples. For example, we have three non-generic structures with codimension 2, so the smallest perturbations are in this case determined from 300 random examples. The singular structures (cases) that give the smallest perturbations are shown in columns directly following columns 2, 4 and 6 of Table 10.

Numbers in bold font in tables 9 and 10 indicate that the size of the perturbations (distances) computed by modified `GUPTRI` are the same as for the explicit perturbations, which for these cases also are shown to be the minimal

Table 9: Mean values of perturbations (measured as $\|(A-\tilde{A}, B-\tilde{B})\|_E$) required to impose each of the 17 non-generic forms for 100 random $A-\lambda B$ of size 2-by-3.

Case	$A-\lambda B$	$B-\mu A$	Explicit	Cod$(A-\lambda B)$	Comment
1	0.000	0.000	0.000	0	
$1'$	0.160	0.154	0.127	1	
2	**0.181**	0.394	0.181*	2	
6	0.378	**0.190**	0.190*	2	
5	0.235	0.227	0.140*	2	
$4'$	0.218	0.268	0.211	3	
$10'$	0.287	0.227	0.220	3	
4	*0.456*	0.533	0.461	4	
10	0.538	*0.481*	0.524	4	
7	0.437	0.434	0.281*	4	
$7'$	0.589	0.602	0.326*	5	
3	**0.707**	**0.707**	0.707*	6	$A=0_{2\times 3}$
11	**0.707**	**0.707**	0.707*	6	$B=0_{2\times 3}$
$9'$	0.399	0.399	0.353*	6	
9	0.466	0.460	0.390	7	
8	**0.737**	**0.737**	0.737*	8	$A=0_{2\times 3}$
12	**0.736**	**0.736**	0.736*	8	$B=0_{2\times 3}$
13	**1.000**	**1.000**	1.000*	12	$A=B=0_{2\times 3}$

Table 10: Minimum perturbations (measured as $\|(A-\tilde{A}, B-\tilde{B})\|_E$) required to impose non-generic forms of each possible codimension for 100 random $A-\lambda B$ of size 2-by-3.

Cod$(A-\lambda B)$	$A-\lambda B$	Case	$B-\mu A$	Case	Explicit	Case
0	0.000	1	0.000	1	0.000	1
1	$2\cdot 10^{-4}$	$1'$	$3\cdot 10^{-4}$	$1'$	$1\cdot 10^{-4}$	$1'$
2	0.011	2	0.010	5	0.009	5
3	**0.036**	$4'$	0.037	$4'$	0.036	$4'$
4	0.111	4	**0.106**	10	0.106	7
5	0.192	$7'$	**0.119**	$7'$	0.119	$7'$
6	0.163	$9'$	0.163	$9'$	0.153	$9'$
7	0.233	9	0.224	9	0.184	9
8	**0.707**	12	**0.707**	12	0.707	12
12	**1.000**	13	**1.000**	13	1.000	13

perturbations. Numbers marked in italic font in Table 9 indicate that modified GUPTRI computed smaller upper bounds than corresponding bounds for the explicit perturbations.

All explicit perturbations of the 100 2-by-3 random pencils turned out to be tractable. The results show that the smallest distance from $A - \lambda B$ to a non-generic structure with fixed codimension k increases with increasing k, in accordance with the Kronecker structure hierarchy in Figure 1. Case 1' with KCF $L_1 \oplus R_1$ is the closest non-generic pencil. Our explicit bound for case 1' is not proved to be the smallest possible.

5 Some Comments on the General Case

The complexity and the intricacies of the problems considered are well-exposed in sections 2 – 4. In the following we discuss some extensions to general m-by-n pencils. The number of different KCFs grows rapidly with increasing m and n. Some cases are displayed in Table 11. We have been able to generate 20098 structurally different KCFs for $m = 10, n = 20$. Notice that for a given m the number of different structures is fixed for $n \geq 2m$. For $m > n$ the number of KCFs are the same as for the transposed pencil. As an example we show all structurally different 3-by-4 Kronecker forms in Table 12, where we as before let R_2 denote a 2-by-2 regular block with any non-zero finite eigenvalues (see Section 2.1) and, similarly, we let R_3 denote a regular 3-by-3 block.

It is possible to extend Theorem 4.1 to general m-by-$(m+1)$ pencils.

Theorem 5.1 *For a generic m-by-$(m+1)$ pencil (A, B) the following inequalities hold:*

$$0 \equiv d_1(R[m]) < d_1(R[m-1]) \leq \ldots \leq d_1(R[0]), \tag{5.1}$$

$$d_1(R[m-1]) \leq d_1(A), \quad d_1(R[m-1]) \leq d_1(B), \tag{5.2}$$

$$d_1(R[m-1]) \leq d_1(L[m-1]) \leq \ldots \leq d_1(L[0]), \tag{5.3}$$

$$\left.\begin{array}{l} d_k(A) < d_{k+1}(A) \\ d_k(B) < d_{k+1}(B) \\ d_k(R[0]) < d_{k+1}(R[0]) \end{array}\right\} \quad k = 1, \ldots, m-1. \tag{5.4}$$

Proof. From Theorem 2.1 it follows that $d_1(R[m]) = 0$ for all m-by-$(m+1)$ pencils (generic or non-generic). A perturbation that lowers the column rank in $R[k-1]$ will always lower the rank in $R[k]$, since a dependence between columns

Table 11: Number of structurally different Kronecker forms of size m-by-n ($m \leq n$).

m	n: 1	2	3	4	5	6	7	8	9	10
1	4	5	5	5	5	5	5	5	5	5
2		14	18	19	19	19	19	19	19	19
3			41	54	58	59	59	59	59	59
4				110	145	159	163	164	164	164
5					271	358	397	411	415	416

Table 12: All 54 structurally different 3-by-4 pencils.

KCF		
L_3	$L_0 \oplus R_2 \oplus N_1$	$2L_0 \oplus L_2^T$
$L_2 \oplus N_1$	$L_0 \oplus R_3$	$2L_0 \oplus N_2 \oplus L_0^T$
$L_2 \oplus R_1$	$L_0 \oplus J_3$	$2L_0 \oplus N_1 \oplus L_1^T$
$L_2 \oplus J_1$	$L_0 \oplus J_2 \oplus N_1$	$2L_0 \oplus 2N_1 \oplus L_0^T$
$L_1 \oplus N_2$	$L_0 \oplus J_2 \oplus R_1$	$2L_0 \oplus R_1 \oplus L_1^T$
$L_1 \oplus 2N_1$	$L_0 \oplus J_1 \oplus N_2$	$2L_0 \oplus R_1 \oplus N_1 \oplus L_0^T$
$L_1 \oplus R_1 \oplus N_1$	$L_0 \oplus J_1 \oplus 2N_1$	$2L_0 \oplus R_2 \oplus L_0^T$
$L_1 \oplus R_2$	$L_0 \oplus J_1 \oplus R_1 \oplus N_1$	$2L_0 \oplus J_2 \oplus L_0^T$
$L_1 \oplus J_2$	$L_0 \oplus J_1 \oplus R_2$	$2L_0 \oplus J_1 \oplus L_1^T$
$L_1 \oplus J_1 \oplus N_1$	$L_0 \oplus J_1 \oplus J_2$	$2L_0 \oplus J_1 \oplus N_1 \oplus L_0^T$
$L_1 \oplus J_1 \oplus R_1$	$L_0 \oplus 2J_1 \oplus N_1$	$2L_0 \oplus J_1 \oplus R_1 \oplus L_0^T$
$L_1 \oplus 2J_1$	$L_0 \oplus 2J_1 \oplus R_1$	$2L_0 \oplus 2J_1 \oplus L_0^T$
$2L_1 \oplus L_0^T$	$L_0 \oplus 3J_1$	$2L_0 \oplus L_1 \oplus 2L_0^T$
$L_0 \oplus N_3$	$L_0 \oplus L_2 \oplus L_0^T$	$3L_0 \oplus L_0^T \oplus L_1^T$
$L_0 \oplus N_1 \oplus N_2$	$L_0 \oplus L_1 \oplus L_1^T$	$3L_0 \oplus N_1 \oplus 2L_0^T$
$L_0 \oplus 3N_1$	$L_0 \oplus L_1 \oplus N_1 \oplus L_0^T$	$3L_0 \oplus R_1 \oplus 2L_0^T$
$L_0 \oplus R_1 \oplus N_2$	$L_0 \oplus L_1 \oplus R_1 \oplus L_0^T$	$3L_0 \oplus J_1 \oplus 2L_0^T$
$L_0 \oplus R_1 \oplus 2N_1$	$L_0 \oplus L_1 \oplus J_1 \oplus L_0^T$	$4L_0 \oplus 3L_0^T$

in $R[k-1]$ will make the corresponding columns in

$$R[k] = \begin{bmatrix} A & 0 \\ B & R[k-1] \\ 0 & \end{bmatrix}$$

linearly dependent, proving (5.1). A perturbation that reduces the rank in A (or B) will cause a linear dependence among the m first (or last) rows of

$$R[m-1] = \begin{bmatrix} A & 0 \\ & \cdots \\ 0 & B \end{bmatrix}.$$

Since $R[m-1]$ is square $(m^2+m) \times (m^2+m)$, the row rank deficiency is equivalent to $R[m-1]$ being column rank deficient, which proves (5.2). The relations between $d_1(L[k]), k = 0, \ldots, m-1$ in (5.3) can be proved similarly as the corresponding relations between the $R[k]$-matrices in (5.1). For the first inequality in (5.3) we recall the fact that a row rank deficient $L[m-1]$ is equivalent to at least one L_k^T block ($k = 0, \ldots,$ or $m-1$) in the KCF. To match the dimensions of the pencil, the KCF must contain at least one L_i block ($i = 0, \ldots,$ or $m-2$) which is equivalent to $R[i]$ being column rank deficient. Hence $L[m-1]$ row rank deficient is equivalent to $R[i]$ being column rank deficient for some $i = 0, \ldots, m-2$. Now, the first inequality of (5.3) is obtained by applying (5.1) to the relation between $R[i]$ and $R[m-1]$. As in Theorem 4.1 the inequalities (5.4) follow from the definition of $d_k(\cdot)$. $\square$

We can see that the closest non-generic structure to a generic m-by-$(m+1)$ pencil is on distance $d_1(R[m-1])$. Notably, when all inequalities relating to $d_1(R[m-1])$ in Theorem 5.1 are strict, the equation (5.1) excludes any L_k blocks for $k < m-1$ in the KCF of any pencil on distance $d_1(R[m-1])$ from the generic case. Similarly, the equation (5.2) excludes any J_i or N_i blocks, and (5.3) the existence of L_k^T blocks. Altogether, this extends Corollary 4.1 to m-by-$(m+1)$ pencils.

Corollary 5.1 *If all inequalities relating to $d_1(R[m-1])$ in Theorem 5.1 are strict, the closest non-generic structure to a generic m-by-$(m+1)$ pencil is $L_{m-1} \oplus R_1$ (with codimension 1) on distance $d_1(R[m-1])$.*

Corollary 5.1 can be used to characterize the distance to uncontrollability for a single input single output linear system $E\dot{x}(t) = Fx(t) + Gu(t)$, where E and F are p-by-p matrices, G is p-by-1, and E is assumed to be nonsingular. The linear system is completely controllable (i.e., the dimension of the controllable subspace equals p) if and only if $A - \lambda B \equiv [G|F - \lambda E]$ is generic. Under the assumptions in Corollary 5.1 the closest uncontrollable system is on distance $d_1(R[p-1])$ corresponding to the non-generic structure $L_{p-1} \oplus R_1$ (with the eigenvalue of R_1 finite and non-zero but otherwise unspecified).

Since B has full row rank $A - \lambda B \equiv [G|F - \lambda E]$ can have neither infinite eigenvalues nor L_j^T blocks in its KCF. Therefore, it can only have finite eigenvalues and L_j blocks in its KCF (and GUPTRI form) and the number of L_j blocks is equal to the number of columns of G. For $p = 2$ the possible uncontrollable systems correspond to cases 1', 2, 5, 4', 4 and 3 of Table 3.

Generalizations of corollaries 4.2 and 4.3 to m-by-$(m+1)$ pencils are straightforward, but there are several more cases to distinguish. The formulations and technicalities are omitted here.

Some results for general matrix pencils relating to problems studied here are presented in [2]. Eigenvalue perturbation bounds are used to develop computational bounds on the distance from a given pencil to one with a qualitatively different Kronecker structure.

Acknowledgements

We are grateful to Alan Edelman and the referees for constructive comments, which have improved both the contents and the organization of the paper.

References

[1] T. Beelen and P. Van Dooren. An improved algorithm for the computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 105:9–65, 1988.

[2] D. L. Boley. Estimating the Sensitivity of the Algebraic Structure of Pencils with Simple Eigenvalue Estimates. *SIAM J. Matrix Anal. Appl.*, 11(4):632–643, October 1990.

[3] J. Demmel and A. Edelman. The Dimension of Matrices (Matrix Pencils) with Given Jordan (Kronecker) Canonical Forms. Report LBL-31839, Mathematics Department, Lawrence Berkeley Laboratories, University of California, Berkeley, CA 94720, 1992.

[4] J. Demmel and B. Kågström. Stably computing the Kronecker structure and reducing subspaces of singular pencils $A - \lambda B$ for uncertain data. In Jane Cullum and Ralph A. Willoughby, editors, *Large Scale Eigenvalue Problems*, pages 283–323. North-Holland, Amsterdam, 1986. Mathematics Studies Series Vol. 127, Proceedings of the IBM Institute Workshop on Large Scale Eigenvalue Problems, July 8-12, 1985, Oberlech, Austria.

[5] J. Demmel and B. Kågström. Computing stable eigendecompositions of matrix pencils. *Lin. Alg. Appl.*, 88/89:139–186, April 1987.

[6] J. Demmel and B. Kågström. Accurate solutions of ill-posed problems in control theory. *SIAM J. Mat. Anal. Appl.*, 9(1):126–145, January 1988.

[7] J. Demmel and B. Kågström. The Generalized Schur Decomposition of an Arbitrary Pencil $A - \lambda B$: Robust Software with Error Bounds and Applications. Part I: Theory and Algorithms. *ACM Trans. Math. Software*, Vol.19(No. 2):160–174, June 1993.

[8] J. Demmel and B. Kågström. The Generalized Schur Decomposition of an Arbitrary Pencil $A - \lambda B$: Robust Software with Error Bounds and Applications. Part II: Software and Applications. *ACM Trans. Math. Software*, Vol.19(No. 2):175–201, June 1993.

[9] F. Gantmacher. *The Theory of Matrices, Vol. I and II (transl.)*. Chelsea, New York, 1959.

[10] G. Golub and C. Van Loan. *Matrix Computations*. Second Edition. Johns Hopkins University Press, Baltimore, MD, 1989.

[11] B. Kågström. The generalized singular value decomposition and the general $A - \lambda B$ problem. *BIT*, 24:568–583, 1984.

[12] B. Kågström. RGSVD - an algorithm for computing the Kronecker canonical form and reducing subspaces of singular matrix pencils $A - \lambda B$. *SIAM J. Sci. Stat. Comp.*, 7(1):185–211, 1986.

[13] V. Kublanovskaya. AB-algorithm and its modifications for the spectral problem of linear pencils of matrices. *Num. Math.*, 43:329–342, 1984.

[14] V. Kublanovskaya and V. B. Chazanov. Spectral problems for matrix pencils: methods and algorithms, part I (in Russian). Preprint LOMI P-2-88, USSR Academy of Sciences, Leningrad, 1988.

[15] C. Paige. Properties of numerical algorithms related to computing controllability. *IEEE Trans. Autom. Contr.*, AC-26(1):130–138, 1981.

[16] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 27:103–141, 1979.

[17] P. Van Dooren. The generalized eigenstructure problem in linear system theory. *IEEE Trans. Autom. Contr.*, AC-26(1):111–129, 1981.

[18] P. Van Dooren. Reducing subspaces: Definitions, properties and algorithms. In B. Kågström and A. Ruhe, editors, *Matrix Pencils*, pages 58–73. Springer-Verlag, Berlin, 1983. Lecture Notes in Mathematics, vol. 973, Proceedings, Pite Havsbad, 1982.

[19] W. Waterhouse. The codimension of singular matrix pairs. *Lin. Alg. Appl.*, 57:227–245, 1984.

[20] J. H. Wilkinson. Linear differential equations and Kronecker's canonical form. In C. de Boor and G. Golub, editors, *Recent Advances in Numerical Analysis*, pages 231–265. Academic Press, 1978.

A Proof of Theorem 2.2

Proof. First we prove that each arc in the graph correspond to a closure relation, and then we prove that these are all arcs that can exist. We prove that one KCF is in the closure of the orbit of another KCF by showing that the one in the closure is just a special case of the one defining the closure. We show proofs for each arc starting from the zero pencil.

Before looking at each arc we note that there is a symmetry regarding row ranks and column nullities between the Kronecker structures with J_i and N_i blocks replaced (see Table 3). From this follows that some of the proofs below that are shown for J_i blocks can be done similarly for the corresponding case with N_i blocks. Typically we have to work with specific elements in A instead of B or vice versa. For these cases we will just mention this similarity without repeating the computations.

In the following, α, β, γ, δ, and ϵ are supposed to be non-zero elements when nothing else is stated.

- $3L_0 \oplus 2L_0^T$ is in the closure of orbit$(2L_0 \oplus J_1 \oplus L_0^T)$, since $3L_0 \oplus 2L_0^T$ is the special case $\alpha = 0$ of
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \end{bmatrix},$$
which is equivalent to $2L_0 \oplus J_1 \oplus L_0^T$ for all non-zero α.
- $3L_0 \oplus 2L_0^T$ is in the closure of orbit$(2L_0 \oplus N_1 \oplus L_0^T)$ follows from similar arguments based on the symmetry between J_i and N_i blocks.
- $2L_0 \oplus J_1 \oplus L_0^T$ is in the closure of orbit$(2L_0 \oplus R_1 \oplus L_0^T)$, since $2L_0 \oplus J_1 \oplus L_0^T$ is the special case $\alpha = 0$ of
$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & \beta \\ 0 & 0 & 0 \end{bmatrix},$$
which is equivalent to $2L_0 \oplus R_1 \oplus L_0^T$ for all non-zero α.
- $2L_0 \oplus N_1 \oplus L_0^T$ is in the closure of orbit$(2L_0 \oplus R_1 \oplus L_0^T)$ follows from similar arguments.
- $2L_0 \oplus J_1 \oplus L_0^T$ is in the closure of orbit$(L_0 \oplus 2J_1)$, since $2L_0 \oplus J_1 \oplus L_0^T$ is the special case $\alpha = 0$ of
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & \beta \\ 0 & \alpha & 0 \end{bmatrix},$$
which multiplied by a permutation matrix can be shown to be equivalent to
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & \alpha & 0 \\ 0 & 0 & \beta \end{bmatrix},$$

and this pencil is equivalent to $L_0 \oplus 2J_1$ for all non-zero α.

- $2L_0 \oplus N_1 \oplus L_0^T$ is in the closure of orbit$(L_0 \oplus 2N_1)$ follows from similar arguments.

- $2L_0 \oplus R_1 \oplus L_0^T$ is in the closure of orbit$(2L_0 \oplus L_1^T)$, since $2L_0 \oplus R_1 \oplus L_0^T$ is the special case $\beta = 0$ of

$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & \beta \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & \gamma \\ 0 & 0 & 0 \end{bmatrix},$$

which for non-zero β is shown to be equivalent to $2L_0 \oplus L_1^T$ by the following equivalence transformation

$$\begin{bmatrix} \frac{1}{\gamma} & \frac{-\alpha}{\beta\gamma} \\ 0 & \frac{1}{\beta} \end{bmatrix} \left(\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & \beta \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & \gamma \\ 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

- $2L_0 \oplus R_1 \oplus L_0^T$ is in the closure of orbit$(L_0 \oplus L_1 \oplus L_0^T)$, since $2L_0 \oplus R_1 \oplus L_0^T$ is the special case $\beta = 0$ of

$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & \beta & \gamma \\ 0 & 0 & 0 \end{bmatrix},$$

which for non-zero β is shown to be equivalent to $L_0 \oplus L_1 \oplus L_0^T$ by the following equivalence transformation

$$\begin{bmatrix} \frac{1}{\alpha} & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & \beta & \gamma \\ 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{\alpha}{\beta} & -\frac{\gamma}{\beta} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

- $2L_0 \oplus L_1^T$ is in the closure of orbit$(L_0 \oplus J_1 \oplus N_1)$, since $2L_0 \oplus L_1^T$ is the special case $\gamma = 0$ of

$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & \beta \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & \delta \\ 0 & 0 & 0 \end{bmatrix}. \tag{A.1}$$

This is shown by the following equivalence transformation:

$$\begin{bmatrix} \frac{1}{\delta} & \frac{-\alpha}{\beta\delta} \\ 0 & \frac{1}{\beta} \end{bmatrix} \left(\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & \beta \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & \delta \\ 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix},$$

which is identical to $2L_0 \oplus L_1^T$. That the pencil (A.1) is equivalent to $L_0 \oplus J_1 \oplus N_1$ for all non-zero γ follows from the equivalence transformation:

$$\begin{bmatrix} 1 & \frac{-\alpha}{\beta} \\ 0 & \frac{1}{\beta} \end{bmatrix} \left(\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & \beta \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & \delta \\ 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\gamma} & \frac{-\delta}{\gamma} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{A.2}$$

- $2L_0 \oplus L_1^T$ is in the closure of orbit$(L_0 \oplus J_2)$, since $2L_0 \oplus L_1^T$ is a permutation of
$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \beta \end{bmatrix},$$
which is the special case $\gamma = 0$ of
$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & 0 \\ 0 & 0 & \beta \end{bmatrix},$$
and this pencil is equivalent to $L_0 \oplus J_2$ for all non-zero γ.

- $2L_0 \oplus L_1^T$ is in the closure of orbit$(L_0 \oplus N_2)$ follows from similar arguments.

- $L_0 \oplus L_1 \oplus L_0^T$ is in the closure of orbit$(L_0 \oplus J_2)$, since $2L_0 \oplus L_1^T$ is the special case $\beta = 0$ of
$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & 0 \\ 0 & 0 & \beta \end{bmatrix},$$
which is equivalent to $L_0 \oplus J_2$ for all non-zero β.

- $L_0 \oplus L_1 \oplus L_0^T$ is in the closure of orbit$(L_0 \oplus N_2)$ follows from similar arguments.

- $L_0 \oplus L_1 \oplus L_0^T$ is in the closure of orbit$(L_0 \oplus J_1 \oplus N_1)$, since $L_0 \oplus L_1 \oplus L_0^T$ is the special case $\beta = 0$ of
$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & \beta \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & \delta \\ 0 & 0 & 0 \end{bmatrix}. \tag{A.3}$$
This follows from the equivalence transformation
$$\begin{bmatrix} \frac{1}{\alpha} & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & \delta \\ 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{\alpha}{\gamma} & -\frac{\delta}{\gamma} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$
That (A.3) is equivalent to $L_0 \oplus J_1 \oplus N_1$ for non-zero β is shown in (A.2).

- $L_0 \oplus 2J_1$ is in the closure of orbit$(L_0 \oplus J_2)$, since $L_0 \oplus 2J_1$ is the special case $\alpha = 0$ of
$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix},$$
which is equivalent to $L_0 \oplus J_2$ for all non-zero α.

- $L_0 \oplus 2N_1$ is in the closure of orbit$(L_0 \oplus N_2)$ follows from similar arguments.

- $L_0 \oplus J_2$ is in the closure of orbit$(L_0 \oplus J_1 \oplus R_1)$, since $L_0 \oplus J_2$ is the special case $\beta = 0$ of
$$\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & \beta \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & 0 \\ 0 & 0 & \delta \end{bmatrix},$$
which for non-zero β is shown to be equivalent to $L_0 \oplus J_1 \oplus R_1$ (with eigenvalue β/δ) by the following equivalence transformation
$$\begin{bmatrix} 1 & -\frac{\alpha}{\beta} \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 0 & 0 & \alpha \\ 0 & 0 & \beta \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & 0 \\ 0 & 0 & \delta \end{bmatrix} \right) \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\gamma} & \frac{\alpha}{\beta\gamma} \\ 0 & 0 & \frac{1}{\delta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{\beta}{\delta} \end{bmatrix} - \lambda \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

- $L_0 \oplus N_2$ is in the closure of orbit$(L_0 \oplus N_1 \oplus R_1)$ follows from similar arguments.

- $L_0 \oplus J_1 \oplus N_1$ is in the closure of orbit$(L_0 \oplus J_1 \oplus R_1)$, since $L_0 \oplus J_1 \oplus N_1$ is the special case $\gamma = 0$ of
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \alpha \end{bmatrix} - \lambda \begin{bmatrix} 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix},$$
which is equivalent to $L_0 \oplus J_1 \oplus R_1$ for all non-zero γ.

- $L_0 \oplus J_1 \oplus N_1$ is in the closure of orbit$(L_0 \oplus N_1 \oplus R_1)$ follows from similar arguments.

- $L_0 \oplus J_1 \oplus R_1$ is in the closure of orbit$(L_1 \oplus J_1)$, since $L_0 \oplus J_1 \oplus R_1$ is equivalent to $L_0 \oplus R_1 \oplus J_1$ which is the special case $\alpha = 0$ of
$$\begin{bmatrix} \alpha & \beta & 0 \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & 0 \\ 0 & 0 & \delta \end{bmatrix},$$
which for non-zero α is shown to be equivalent to $L_1 \oplus J_1$ by the following equivalence transformation
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \alpha & \beta & 0 \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & 0 \\ 0 & 0 & \delta \end{bmatrix} \right) \begin{bmatrix} -\frac{\beta}{\alpha\gamma} & \frac{1}{\alpha} & 0 \\ \frac{1}{\gamma} & 0 & 0 \\ 0 & 0 & \frac{1}{\delta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

- $L_0 \oplus N_1 \oplus R_1$ is in the closure of orbit$(L_1 \oplus N_1)$ follows from similar arguments.
- $L_0 \oplus J_1 \oplus R_1$ is in the closure of orbit$(L_0 \oplus R_2)$, since $L_0 \oplus J_1 \oplus R_1$ is the special case $\alpha = 0$ of
$$\begin{bmatrix} 0 & \alpha & 0 \\ 0 & 0 & \beta \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & 0 \\ 0 & 0 & \delta \end{bmatrix},$$
which is equivalent to $L_0 \oplus R_2$ for all non-zero α.
- $L_0 \oplus J_1 \oplus R_1$ is in the closure of orbit$(L_0 \oplus R_2)$ follows from similar arguments.
- $L_1 \oplus J_1$ is in the closure of orbit$(L_1 \oplus R_1)$, since $L_1 \oplus J_1$ is the special case $\beta = 0$ of
$$\begin{bmatrix} \alpha & 0 & 0 \\ 0 & 0 & \beta \end{bmatrix} - \lambda \begin{bmatrix} 0 & \gamma & 0 \\ 0 & 0 & \delta \end{bmatrix},$$
which is equivalent to $L_1 \oplus R_1$ for all non-zero β.
- $L_1 \oplus N_1$ is in the closure of orbit$(L_1 \oplus R_1)$ follows from similar arguments.
- $L_0 \oplus R_2$ is in the closure of orbit$(L_1 \oplus R_1)$, since $L_0 \oplus R_2$ is the special case $\alpha = 0$ of
$$\begin{bmatrix} \alpha & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix} - \lambda \begin{bmatrix} 0 & \delta & 0 \\ 0 & 0 & \epsilon \end{bmatrix},$$
which for non-zero α is shown to be equivalent to $L_1 \oplus R_1$ (with eigenvalue γ/ϵ) by the following equivalence transformation
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \alpha & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix} - \lambda \begin{bmatrix} 0 & \delta & 0 \\ 0 & 0 & \epsilon \end{bmatrix} \right) \begin{bmatrix} \frac{1}{\alpha} & -\frac{\beta}{\alpha\delta} & 0 \\ 0 & \frac{1}{\delta} & 0 \\ 0 & 0 & \frac{1}{\epsilon} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & \frac{\gamma}{\epsilon} \end{bmatrix} - \lambda \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$
- $L_1 \oplus R_1$ is in the closure of orbit(L_2), since L_2 spans the complete 12-dimensional space.

Now we have shown that all arcs in the graph are valid. It remains to show that there are no arcs missing. This can be done by examining the KCF:s that cannot be in the closure of each other.

First we remark that one necessary condition for a KCF to be in the closure of the orbit of another is that it must have higher codimension than the one defining the closure.

Since $L_0 \oplus N_1 \oplus R_1$, $L_0 \oplus N_2$ and $L_0 \oplus 2N_1$ all require that A has full rank $(= 2)$, none of them can be in the closure of orbit$(L_1 \oplus J_1)$, since that KCF requires A to have rank $= 1$. (Of course this also implies that none of these three KCF:s can be in the closure of the orbit of $L_0 \oplus J_1 \oplus R_1$, $L_0 \oplus 2J_1$ or any other KCF that is in the closure of $\text{orbit}(L_1 \oplus J_1)$.)

From the symmetry for J_i and N_i blocks, we have that none of $L_0 \oplus J_1 \oplus R_1$, $L_0 \oplus J_2$ and $L_0 \oplus 2J_1$ can be in the closure of $\text{orbit}(L_1 \oplus N_1)$, since they require B to have full rank and $L_1 \oplus N_1$ has $\text{rank}(B) = 1$.

Since $2L_0 \oplus J_1 \oplus L_0^T$ and $2L_0 \oplus R_1 \oplus L_0^T$ have a B of rank 1, none of them can be in the closure of $\text{orbit}(L_0 \oplus 2N_1)$ since that KCF requires a 2-dimensional rank deficiency in B. By similar arguments for the rank of A we see that $2L_0 \oplus N_1 \oplus L_0^T$ and $2L_0 \oplus R_1 \oplus L_0^T$ cannot be in the closure of $\text{orbit}(L_0 \oplus 2J_1)$. Since we have investigated all presumptive KCF:s the proof is complete. $\square$

Paper IV

A Geometric Approach to Perturbation Theory of Matrices and Matrix Pencils. Part I: Versal Deformations*

Alan Edelman†, Erik Elmroth‡, and Bo Kågström

Department of Computing Science, Umeå University
S-901 87 Umeå, Sweden.
E-mail: edelman@math.mit.edu, elmroth@cs.ume.se, bokg@cs.ume.se

Abstract

We derive versal deformations of the Kronecker canonical form by deriving the tangent space and orthogonal bases for the normal space to the orbits of strictly equivalent matrix pencils. These deformations reveal the local perturbation theory of matrix pencils related to the Kronecker canonical form. We also obtain a new singular value bound for the distance to the orbits of less generic pencils. The concepts, results and their derivations are mainly expressed in the language of numerical linear algebra. We conclude with experiments and applications.

Keywords: Jordan canonical form, Kronecker canonical form, generalized Schur decomposition, staircase algorithm, versal deformations, tangent and normal spaces, singularity theory, perturbation theory.

---

*Submitted to *SIAM Journal on Matrix Analysis and its Applications.*

†Department of Mathematics, Room 2–380, Massachusetts Institute of Technology, Cambridge, MA 02139, USA, Supported by NSF grant DMS-9120852 and an Alfred P. Sloan Foundation Research Fellowship.

‡In part supported by the Swedish National Board of Industrial and Technical Development under grant 89-02578P.

1 Introduction and Examples

1.1 Introduction

Traditionally, canonical structure computations take as their input some mathematical object, a matrix or a pencil say, and return an equivalent object that is perhaps simpler, or makes clear the structure of the equivalence relation. Some example equivalence relations and corresponding canonical forms are:

Structure	**Equivalence Relation**	**Canonical Form**
Square Matrices	$A \sim X^{-1}AX$	Jordan Canonical Form
Rectangular Matrices	$A \sim UAV$	Singular Values
Rectangular Matrices	$A \sim XA$	Reduced Echelon form
Matrix Pencils	$A - \lambda B \sim P^{-1}(A - \lambda B)Q$	Kronecker Canonical Form
Analytic real functions	$f(x) \sim f(\phi(x))$	$\pm x^k$

In the first three examples the input is a matrix, in the next example, the input is a pencil. In these cases, X, P, and Q are presumed nonsingular, and U and V are presumed orthogonal. We presume the real functions f are analytic in a neighborhood of zero, $f(0) = 0$, $\phi(0) = 0$ and $\phi(x)$ is monotonic and analytic near zero.

Canonical forms appear in every branch of mathematics. A few examples from control theory may be found in [20, 19, 25, 18]. However, researchers in singularity theory have asked the question what happens if you have not one object that you want to put into a normal form, but rather a whole family of objects nearby some particular object and you wish to put each member of the family into a canonical form in such a way that the canonical form depends smoothly on the deformation parameters.

For example, one may have, a one parameter matrix deformation of A_0 which is simply an analytic function $A(\lambda)$ for which $A(0) = A_0$. An n parameter deformation is defined the same way, except that $\lambda \in \mathbf{R}^n$. Similarly, one may have n parameter deformations of pencils or functions. Sticking with the matrix example, we say two deformations $A(\lambda)$ and $B(\lambda)$ are equivalent if $A(\lambda)$ and $B(\lambda)$ have the same Jordan canonical form for each and every λ. A deformation of a matrix is said to be versal if, loosely speaking, it captures all possible Jordan form behaviors, near the matrix. A deformation is said to be miniversal, if it does so with as few parameters as possible. A more formal discussion of these definitions may be found in Section 2.

Derivation of versal and miniversal deformations requires a detailed understanding of the perturbation theory of the objects under study. In particular, one needs to understand the tangent space of the equivalence relation, and how it is embedded in the entire space. In Section 2, we explain the mechanics of this perturbation theory.

While we believe that versal deformations are interesting mathematical objects, this work differs from other works on the subject in that our primary goal is not so much the versal deformation or the miniversal deformation, but rather the perturbation theory and how it influences the computation of the Kronecker canonical form. As such we tend to be interested more in metrical information than topological information. Therefore, we obtain new distance formulas to the space of less generic matrix pencils in Section 4. In Section 5, we derive an explicit orthogonal basis for the normal space of a Kronecker canonical form. For us a versal decomposition will be an explicit decomposition of a perturbation into its tangential and normal components, and we will not derive any miniversal deformations that may have simpler forms, but hide the metric information.

Versal deformations for function spaces are discussed in [17, 24, 4, 5]. The first application of these ideas for the matrix Jordan canonical form is due to Arnold [1]. Further references closely related to Arnold's matrix approach are [28] and [6]. The latter reference, [6], also includes applications to differential equations. Applications of the matrix idea towards an understanding of companion matrix eigenvalue calculations may be found in [13]. The only other work that we are aware of that considers versal deformations of the Kronecker canonical form is by Berg and Kwatny [3] who have independently derived some of the normal forms considered in this paper.

Our Section 2 contains a thorough explanation of versal deformations from a linear algebra perspective. Section 3 briefly reviews matrix pencils and canonical forms. Section 4 derives the geometry of the tangent and normal spaces to the orbits of matrix pencils. Section 5 derives the versal deformations, while Section 6 gives applications and illustrations.

1.2 Geometry of matrix space

Our guiding message is very simple: matrices should be seen in the mind's eye geometrically as points in n^2 dimensional space. A perfect vision of numerical computation would allow us to picture computations as moving matrices from point to point or manifold to manifold.

Abstractly, it hardly matters whether a vector is a column of numbers or a geometric point in space. However, without the interplay of these two representations, numerical linear algebra would not be the same. Imagine explaining how Householder reflections transform vectors without the geometric viewpoint.

By contrast, in numerical linear algebra we all know that matrices are geometric points in n^2 dimensional space, but it is far rarer that we actually *think* about them this way. Most often, matrices are thought of as either (sparse or dense) arrays of numbers, or they are operators on vectors.

The Eckart–Young (or Schmidt–Mirsky theorem) [27, p.210] gives a feel for the geometric approach. The theorem states that the smallest singular value of A is the Frobenius distance of A to the set of singular matrices. One can not help but to see a blob representing the set of singular matrices. This amorphous blob is most often thought of as an undesirable part of town, so unfortunately numerical analysts hardly ever study the set itself.

Demmel has helped to pioneer the development of geometric techniques [7] for the analysis of ill-conditioning of numerical analysis problems. Shub and Smale [26] are applying geometrical approaches towards the solution of polynomial systems.

We believe that if only we could better understand the geometry of matrix space, our knowledge of numerical algorithms and their failures would also improve. A general program for numerical linear algebra, then, is to transfer from pure mathematicians the technology to understand geometrically the high dimensional objects that arise in numerical linear algebra. This program may not be easy to follow. A major difficulty is that pure mathematicians pay a price for their beautiful abstractions – they do not always possess a deep understanding of the individual objects that we wish to study. This makes technology transfer difficult. Even when the understanding exists somewhere, it may be difficult to recognize or may be buried under a heavy layer of notation. This makes technology transfer time consuming. Finally, even after putting in the time for the excavation, the knowledge may still be difficult to apply towards the understanding or the improving of practical algorithms. This makes technology transfer from pure mathematics frustrating.

Nevertheless, our goal as researchers is the quest for understanding which we may then apply. In this paper, we follow our program for the understanding of the Jordan and Kronecker canonical forms of matrices and matrix pencils, respectively. Many of the ideas to be found in this paper have been borrowed from the pure mathematics literature with the goal of simplifying and applying to the needs of numerical linear algebraists.

While this is quite a general program for numerical linear algebra, this paper focuses on a particular goal. We analyze *versal deformations* from the numerical linear algebra viewpoint, and then compute normal deformations for the Kronecker canonical form. We consider both of these as stepping stones towards the far more difficult goal of truly understanding and improving upon staircase algorithms for the Jordan or Kronecker canonical form. These are algorithms used in systems and control theory. The structures of these matrices or pencils reflect important physical properties of the systems they model, such as controllability [10, 30].

The user chooses a parameter η to measure any uncertainty in the data. The existence of a matrix or pencil with a different structure within distance η of the input means that the actual system may have a different structure than the approximation supplied as input. These algorithms try to perturb their input by at most η so as to find a matrix or pencil with as high a codimension as possible. The algorithm is said to *fail* if there is another perturbation of size at most η which would raise the codimension even further. Therefore, we need to understand the geometry of matrix space in order to begin to understand how we can supply the correct information to the user. With this information, we believe that we would then be able to not only correctly provide the least generic solutions, but also understand how singularities hinder this process. Bad solutions may then be refined so as to obtain better solutions. As the next subsection illustrates, the geometry directly affects the perturbation theory.

1.3 Motivation: a singular value puzzle

Consider the following four nearly singular matrices:

$$M_1 = \begin{pmatrix} 0 & 1+\epsilon \\ 0 & 0 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 0 & 1 \\ \epsilon & 0 \end{pmatrix}, \quad M_3 = \begin{pmatrix} \epsilon & 1 \\ 0 & -\epsilon \end{pmatrix}, \quad M_4 = \begin{pmatrix} \epsilon & 1 \\ 0 & \epsilon \end{pmatrix}. \tag{1.1}$$

Each of these matrices are distance $O(\epsilon)$ from the Jordan block

$$J_2(0) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

What is the smaller of the two singular values of each of M_1, M_2, M_3 and M_4? The answer is

$$\sigma_{\min}(M_1) = 0, \quad \sigma_{\min}(M_2) = \epsilon, \quad \sigma_{\min}(M_3) \approx \epsilon^2 \quad \text{and} \quad \sigma_{\min}(M_4) \approx \epsilon^2.$$

A quick way to verify this algebraically is to notice that the larger singular value of each matrix is approximately 1 so that the smaller is approximately the (absolute) determinant of the matrix. Another approach that bounds the smallest singular value is the combination of the Eckart–Young theorem and the observation that these matrices are singular:

$$M_1' = M_1, \qquad M_2' = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad M_3' = \begin{pmatrix} \epsilon & 1 \\ -\epsilon^2 & -\epsilon \end{pmatrix}, \quad M_4' = \begin{pmatrix} \epsilon & 1 \\ \epsilon^2 & \epsilon \end{pmatrix}.$$

When $\epsilon = 0$ in (1.1) our four matrices become the singular 2×2 Jordan block $J_2(0)$. As ϵ varies from 0 each of the four forms in (1.1) traces out a line in matrix space. The geometric issue that is interesting here is that the line of matrices traced out as ϵ varies is { 1:In 2:Normal 3:Tangent 4:Tangent } to the set of singular matrices. Somehow, this feels like the "right" explanation for why the smaller singular values are { 1:0, 2:ϵ, 3:$\approx \epsilon^2$, 4:$\approx \epsilon^2$ }.

Let us take a closer look at the set of singular matrices. The four parameters found in a 2×2 matrix M are best viewed in a transformed coordinate system:

$$M = (x, y, z, w) = x \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + y \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + z \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + w \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} w+z & x \\ y & w-z \end{pmatrix}.$$

In this coordinate system, the singular matrices fall on the surface described by the equation $w^2 = z^2 + xy$. This is a three dimensional surface in four dimensional space. The traceless singular matrices ($w = 0$) fall on the cone $z^2 + xy = 0$ in three dimensional space.

Our matrix $J_2(0)$ may now be represented as $(1, 0, 0, 0)$ and the four lines of matrices mentioned above are

$$\begin{aligned}
l_1 &= \{ \quad (1+x, 0, 0, 0) \quad \} = \left\{ \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \epsilon \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \right\}, \\
l_2 &= \{ \quad (1, y, 0, 0) \quad \} = \left\{ \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \epsilon \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \right\}, \\
l_3 &= \{ \quad (1, 0, z, 0) \quad \} = \left\{ \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \epsilon \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \right\}, \\
l_4 &= \{ \quad (1, 0, 0, w) \quad \} = \left\{ \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \epsilon \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right\}.
\end{aligned}$$

The lines l_1, l_2 and l_3 are all traceless, i.e., the matrices on each of these lines may be viewed in the three dimensional space of the cone. The line l_1 is not only tangent to the cone, but in fact it lies in the cone. The line l_3 is tangent to one of the circular cross-sections of the cone.

Figure 1: Cone of traceless singular matrices with "stick" representing a tangent.

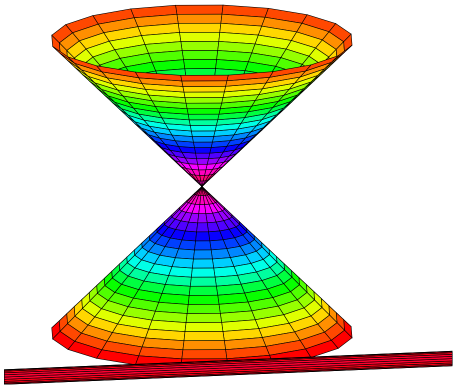
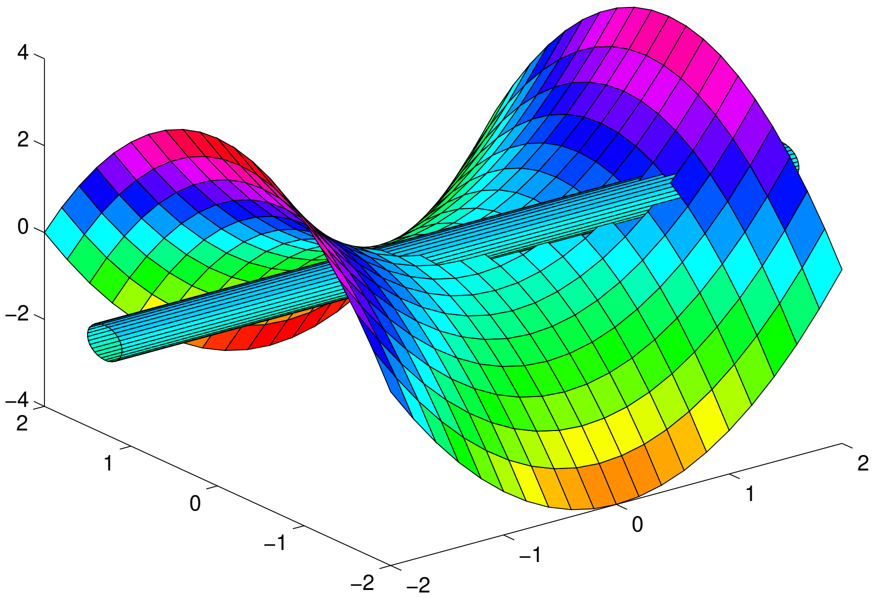


Figure 1 illustrates l_3 as a "stick" resting near the bottom of the cone. The line l_1 is a thin line on the cone through the same point.

The line l_4 is normal to the cone but it is also tangent to the manifold of singular matrices. One way to picture this in three dimensions is to take the three dimensional slice of $\{w^2 = z^2 + xy\}$ corresponding to $x = 1$, i.e., $\{w^2 - z^2 = y\}$. This is a hyperboloid with the Jordan block as a saddle point. The line is the tangent to the parabola $w^2 = y$ which rests in the plane $z = 0$. Figure 2 illustrates this line with a cylindrical stick whose central axis is the tangent. Lastly, the line l_2 is normal to the set of singular matrices.

Figure 2: Manifold of singular matrices. The axis of the cylindrical stick is tangent to the manifold.



If we move a distance ϵ away from a point on a surface along a tangent, our distance to the surface remains $O(\epsilon^2)$. This is what the singular value corresponding to l_3 and l_4 is telling us. Alternatively, if we move normal to the surface as in l_2, the singular value changes more rapidly: $O(\epsilon)$.

The cone of singular matrices with $w = 0$ is not only a slice of a large dimensional space, but it is also the (closure of) the set of matrices similar to $J_2(0)$ (which we denote orbit$(J_2(0))$ in Section 2.4). The matrices similar to $J_2(0)$ are singular and traceless. In fact, the only matrix that is singular and traceless that is not similar to $J_2(0)$ is the 0 matrix which is the vertex of the cone. We further explore this case in Section 2.5 after we have defined versal deformations.

We conclude that the geometry of the orbit and in particular the directions of the tangents and normals to the orbit directly influence the eigenvalue perturbation theory.

2 Introduction to Versal Deformations

This introduction is designed to be readable for general audiences, but we particularly target the numerical linear algebra community.

The ideas here may be thought of as a numerical analyst's viewpoint on ideas that were inspired by Arnold's work [1] on versal deformations of matrices. Further elaboration upon Arnold's versal deformations of matrices may be found in [6, Chapter 2.9 and 2.10] and [28]. These ideas fit into a larger context of differential topology and singularity theory. Bruce and Giblin [5] have written a wonderfully readable introduction to singularity theory emphasizing the elementary geometrical viewpoint. After reading this introduction, it is easy to be lulled into the belief that one has mastered the subject, but a whole further more advanced wealth of information may be found in [17, 24, 4]. Finally, what none of these references do very well is explain clearly that there is still much in this area that mankind does not yet fully understand.

Singularity theory may be viewed as a branch of the study of curves and surfaces, but its crowning application is towards the topological understanding of functions and their behavior under perturbations. Of course, numerical analysts are very interested in perturbations as well.

2.1 Characteristic polynomials give the "feel" of versal deformations

Let $A(\lambda)$ be a differentiable one parameter family of matrices through $A_0 \equiv A(0)$. This is just a curve in matrix space. If A_0 has a complicated Jordan canonical form, then very likely, the Jordan canonical form of $A(\lambda)$ is a discontinuous function of λ. (The Jordan canonical form, you will remember, can have nasty ones popping up unexpectedly on the superdiagonal.) It is even more desirable if that function can somehow describe the kinds of matrices that are near A_0.

Discontinuities are as unpleasant for pure mathematicians as they are for computers. Therefore Arnold [1] asks what kinds functions of λ are differentiable? (or many times differentiable, or analytic.)

One function that comes to mind is the characteristic polynomial $p_\lambda(t) \equiv \det(A(\lambda) - tI)$. The coefficients of p_λ are clearly differentiable functions of λ no matter how complicated a Jordan canonical form the matrix A_0 might have. In numerical linear algebra, we never compute the characteristic polynomial, because the eigenvalues are often very poorly determined by the coefficients of the characteristic polynomial. Mathematically, the characteristic polynomial is a nice function of a matrix because its coefficients, unlike the eigenvalues of the matrix, are analytic functions of the entries of the matrix.

The characteristic polynomial is a reasonable representation for the Jordan canonical form under the special circumstance that every matrix $A(\lambda)$ is *non-derogatory* (i.e., each matrix has exactly one Jordan block for each distinct eigenvalue). By a reasonable representation, we mean here that it actually encodes the Jordan canonical form of A_0. Theoretically, if you know the characteristic polynomial, then you know the eigenvalues with appropriate multiplicities. It follows that there is a unique non-derogatory Jordan canonical form. (See Wilkinson [33, pp.11–16 or Note 55, p.408]). To repeat, there is a one-to-one correspondence among the n eigenvalues of a non-derogatory matrix, the characteristic polynomial of a non-derogatory matrix, and the Jordan canonical form of a non-derogatory matrix, but only the characteristic polynomial is a differentiable function of the perturbation parameter λ. (The eigenvalues themselves can have first order perturbations with the non-differentiable form $\lambda^{1/n}$, for example, for an $n \times n$ matrix A_0 with only one Jordan block $J_n(\lambda)$. This is a well known example.)

In the language of numerical linear algebra, we would say that a non-derogatory matrix A_0 may be written in companion matrix form KCK^{-1}, in such a way that differentiable perturbations to the matrix A_0 lead to differentiable perturbations to the companion matrix C. Here the matrix K is a Krylov matrix. (See [16, p. 369]). Equivalently, first order perturbations to the matrix A_0 are manifested as first order perturbations to the companion matrix C. When A_0 is a companion matrix, this gives a first order perturbation theory for the characteristic polynomials of nearby matrices. This perturbation theory is computed in [13].

Our story would almost stop here if we were only interested in the Jordan form of non-derogatory matrices. We use "almost" because it would be a shame to stop here without explaining the ideas geometrically. Even if we did not discuss the geometry, we have reasons to continue on, since matrix space is enriched with the derogatory matrices, and also we wish to generalize these ideas about the Jordan canonical form to cover the more complicated case of the Kronecker canonical form.

2.2 The rational canonical form is not enough for derogatory matrices

In the previous subsection we saw that n parameters were sufficient to specify the Jordan canonical form of any matrix in a small neighborhood of a non-derogatory matrix. What happens if the matrix is derogatory? One obvious guess turns out to be wrong. The usual generalization of the companion matrix form for derogatory matrices is the rational canonical form. If A_0 is derogatory, it may be put in rational canonical form. This form may be thought of as the direct sum of companion matrices C_i with dimension $m_1 \geq m_2 \geq \ldots \geq m_k$. The characteristic polynomial of each C_i divides the characteristic polynomial of all the preceding $C_j, j < i$. Can any nearby matrix be expressed as the direct sum of companion matrices with dimension $m_1, m_2, \ldots, m_k$ in a nice differentiable manner? The answer is generally no; though good enough to specify the Jordan canonical form of a matrix, the rational canonical form fails to be powerful enough to specify the Jordan canonical forms of all matrices in a neighborhood. The reason is that there are just not enough parameters in the rational canonical form to cover all the possibilities. To have enough parameters we need a "versal deformation".

2.3 Versal deformation: the linearized theory

The "linearized" picture of a versal deformation is easy to understand. We therefore explain this picture before plunging into the global point of view. The general case may be nonlinear, but the linearized theory is all that really matters. For simplicity we assume that we are in real n dimensional Euclidean space, but this assumption is not so important.

We recall the elementary fact that if $\mathcal{S}$ and $\mathcal{T}$ are subspaces of $\mathbf{R}^n$ such that $\mathcal{S}+\mathcal{T}=\mathbf{R}^n$, then there exist linear projections $\pi_{\mathcal{S}}$ and $\pi_{\mathcal{T}}$ that map onto $\mathcal{S}$ and $\mathcal{T}$, respectively.

Consider a point $x \in \mathcal{S}$. We will investigate all possible perturbations y of x, but we will not be concerned with perturbations that are within $\mathcal{S}$ itself. Psychologically, we consider all the vectors in $\mathcal{S}$ to somehow be the same so there will be no need to distinguish them. Let $\mathcal{T}$ be any linear subspace such that $\mathcal{S}+\mathcal{T}=\mathbf{R}^n$, i.e., any vector may be written as the sum of an element of $\mathcal{T}$ and an element of $\mathcal{S}$ (not necessarily uniquely). Clearly if $t_1,\ldots,t_k$ span $\mathcal{T}$, then our perturbed vector $x+y$ may be written as

$$x+y=x+\sum_{i=1}^{k}\lambda_i t_i+(\text{something in }\mathcal{S}),$$

where the λ_i may be chosen as linear functions of y. We see here what will turn out to be the key idea of a versal deformation, every perturbation vector may be expressed in terms of the λ_i and vectors that we are considering to all be equivalent.

We now formally introduce the local picture of versal deformations.

Definition 2.1 *A* linear deformation *of the point x is a function defined on $\lambda \in \mathbf{R}^l$:*

$$A(\lambda)=x+T\lambda,$$

where $T=[t_1 t_2 \ldots t_l]$ are arbitrary directions.

The choice of the word "deformation" is meant to convey the idea that we are looking at small values of the λ_i, and these perturbations are small deformations of the starting point x.

Definition 2.2 *A linear deformation $A(\lambda)$ of the point x is* versal *if for all linear deformations $B(\mu)$ of the point x, it is possible to write*

$$B(\mu)=A(\phi(\mu))+\theta(\mu),$$

where $\phi(\mu)$ is a linear function from $\mu_1, \ldots, \mu_m$ to $\lambda_1, \ldots, \lambda_l$ with $\phi(0) = 0$, and θ is a linear function from μ into $\mathcal{S}$, with $S(0) = 0$.

We now explain why $A(\lambda) = x + \sum_{i=1}^{l} \lambda_i t_i$ is versal if and only if $\mathcal{S} + \mathcal{T} = \mathbf{R}^n$. Clearly $A(\phi(\mu)) + \theta(\mu) \in \mathcal{S} + \mathcal{T}$ and since B may be arbitrary, it is necessary that $\mathrm{span}(\{t_i\}) + \mathcal{S} = \mathbf{R}^n$. It is also sufficient, because we then obtain linear projections allowing us to write $B(\mu) = x + \pi_{\mathcal{S}} B(\mu) + \pi_{\mathcal{T}} B(\mu)$. The functions ϕ and θ may be obtained from $\pi_{\mathcal{S}}$ and $\pi_{\mathcal{T}}$.

Definition 2.3 *A linear deformation $A(\lambda)$ of the point x is* universal *or* miniversal *if it is versal, and has the fewest possible parameters needed for a versal deformation.*

The number of parameters in a miniversal deformation is exactly the codimension of $\mathcal{S}$. Numerical analysts might prefer taking the t_i to be an orthogonal basis for $\mathcal{S}^{\perp}$, the subspace perpendicular to $\mathcal{S}$. This provides one natural miniversal deformation. Arnold [1] does not insist on using $\mathcal{S}^{\perp}$, any basis for any subspace of dimension $n - \dim \mathcal{S}$ will do provided that it intersects $\mathcal{S}$ at zero only. From the topological point of view, this is exactly the same, though of course the numerical properties may be quite different.

2.4 Versal deformations the bigger picture

The previous subsection explained the linear or first order theory of versal deformations. At this point, the reader might wonder whether this is just a whole lot of jargon to merely extend a basis for a subspace to the entire space. At the risk of delaying the motivation until now, we decided to make sure that the linear theory is well understood.

We are still in a finite dimensional Euclidean space $\mathbf{R}^n$, but $\mathcal{S}$ will no longer be a flat subspace. Instead, we wish to consider any equivalence relation $\sim$, such that the *orbit* of x ($\mathrm{orbit}(x) \equiv \{y | y \sim x\}$) is a smooth submanifold. As an example we might define $x \sim y$ to mean $\|x\| = \|y\|$, in which case the orbits are spheres. In this context the word "orbit" is quite natural. In n^2 dimensional space, points may be thought of as $n \times n$ matrices, and the orbit is the set of matrices with the same Jordan canonical form.

One final example that we must mention (because it explains the origins and significance of singularity theory) lives in an infinite dimensional space. The vector space is the set of analytic functions $f(x)$ for which $f(0) = 0$. We can define $f \sim g$, if $f(x)$ and $g(\phi(x))$ have the same Taylor expansion at $x = 0$, where ϕ is a monotonic analytic function with $\phi(0) = 0$. The orbit of any function is some complicated infinite dimensional manifold, but the codimension of the manifold happens to be finite.

Returning to $\mathbf{R}^n$, we can now cast everything into a nonlinear context.

Definition 2.4 *A* deformation *of the point x is any differentiable function*

$$A(\lambda_1, \ldots, \lambda_l)$$

satisfying $A(0) = x$.

Definition 2.5 *A deformation $A(\lambda)$ of the point x is* versal *if for all deformations $B(\mu)$, it is possible to write*

$$B(\mu) \sim A(\phi(\mu))$$

in an arbitrarily small neighborhood of 0, where $\phi(\mu)$ is a differentiable function from $\mu_1, \ldots, \mu_m$ to $\lambda_1, \ldots, \lambda_l$ for which $\phi(0) = 0$.

The good news is that the inverse function theorem lets us express this nonlinear notion in terms of the linear theory:

Theorem 2.1 *A deformation $A(\lambda)$ of x is versal if and only if $A_*(\lambda)$ is a linear deformation at the point x on the subspace* $\tan(\text{orbit}(x))$*, where A_* is the linearization of A near x (i.e. only first derivatives matter), and* tan *denotes the subspace tangent to the orbit at x.*

The rigorous proof may be found in [1], but the intuition should be clear: near the point x, only linear deformations matter, and the curvature of the orbit becomes unimportant: only the tangent plane matters. In other words $y \sim x$ only if y is in the orbit of x, but to first order, $y \sim x$ if (roughly speaking) $y = x + s$, where s is a small tangent vector to the orbit.

2.5 Versal deformations for the Jordan canonical form

We begin with deformations of the matrix $A_0 = J_2(0)$. The perturbation theory and the normal and tangent spaces were discussed in Section 1.3. We will use the same coordinate system here.

Four parameters $\mu = (\mu_1, \mu_2, \mu_3, \mu_4)$ are sufficient to describe the most general deformation of A_0:

$$A(\mu) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \mu_1 & \mu_2 \\ \mu_3 & \mu_4 \end{pmatrix}.$$

The equivalence relation is that of similar matrices, and it is easy to see by checking the trace and determinant, that for sufficiently small values of μ, we have the equivalence,

$$A(\mu) \sim B(\lambda) \equiv \begin{pmatrix} 0 & 1 \\ \lambda_1 & \lambda_2 \end{pmatrix},$$

where $\lambda = \phi(\mu)$ is defined by $\lambda_1 = \mu_3(1+\mu_2) - \mu_1\mu_4$ and $\lambda_2 = \mu_1 + \mu_4$. It is worth emphasizing that the equivalence relation does not work if $A(\mu)$ is derogatory, but this does not happen for small parameters μ.

We then see from Definition 2.5, that the two parameter deformation $B(\lambda)$ is versal. In fact, it is *miniversal*, in that one needs the two parameters. From the local theory pictured in Section 1.3, we saw that the orbit of $J_2(0)$ is the two dimensional cone, and therefore the tangent and normal spaces are each two dimensional. The number of parameters in a miniversal deformation is always the dimension of the normal space.

It is a worthwhile exercise to derive the similarity transformation $C(\mu)$ for which

$$A(\mu) = C(\mu)^{-1}B(\phi(\mu))C(\mu),$$

and then linearize this map for small values of μ to see which directions fall along the tangent space to the cone, and which directions are normal to the cone.

Now consider deformations of $A_0 = I_2$ or $A_0 = 0$. Both matrices are derogatory with 2 eigenvalues 1 and 0, respectively. The tangent space does not exist (i.e., it is zero dimensional). Any possible behavior may be found near I_2 (or 0) including a one dimensional space of derogatory matrices. The miniversal deformation of I_2 (or 0) is the full deformation requiring four parameters.

The general case has been worked out by Arnold [1]. The tangent vectors to the orbit of a matrix A_0 are those matrices that may be expressed as $XA_0 - A_0X$. The normal space is the adjoint of the centralizer, i.e., the set of matrices Z satisfying

$$A_0^H Z = Z A_0^H.$$

Let A_0 has p distinct eigenvalues $\lambda_i, i = 1 : p$ with p_i Jordan blocks each. Let $q_1(\lambda_i) \geq q_2(\lambda_i) \geq \ldots \geq q_{p_i}(\lambda_i)$ denote the sizes of the Jordan blocks corresponding to the eigenvalue λ_i. Then the dimension of the normal space of A_0 is

$$\sum_{i=1}^{p}\sum_{j=1}^{p_i}(2j-1)q_j(\lambda_i) = \sum_{i=1}^{p}(q_1(\lambda_i) + 3q_2(\lambda_i) + 5q_3(\lambda_i) + \ldots).$$

Notice that the values of the distint λ_i play no role in this formula. The dimension of the normal space of A_0 is determined only by the sizes of the Jordan blocks of A_0 associated with distinct eigenvalues. If the matrix is in Jordan canonical form, then the normal space consists of matrices made up of Toeplitz blocks, whose block structure is completely determined by the sizes of the Jordan blocks for different eigenvalues. The normal space is the same for all matrices with the same Jordan structure independent of the values of the distinct eigenvalues, so one may as well consider only Jordan blocks corresponding to a 0 eigenvalue. This form of the normal space for the zero eigenvalues is a special case in Theorem 5.1.

3 The Algebra of Matrix Pencils – Canonical Forms

We saw in Section 2.4 that to consider versal deformations, one needs a finite or infinite dimensional space, and an equivalence relation on this space. For the remainder of this paper, we consider the finite dimensional Euclidean space of matrix pencils endowed with the Euclidean metric (usually denoted the Frobenius metric in this context). The equivalence relation is that of the strict equivalence of pencils.

We consider a matrix pencil $A - \lambda B$, where A and B are arbitrary $m \times n$ matrices with real or complex entries. The pencil is said to be *regular* if $m = n$ and $\det(A-\lambda B)$ is not identically zero. Indeed, the zeros of $\det(A-\lambda B) = 0$ are the (generalized) eigenvalues of a regular pencil. Otherwise, i.e., if $\det(A-\lambda B)$ is identically zero or $m \neq n$, $A-\lambda B$ is called *singular.* Two $m \times n$ pencils $A_1 - \lambda B_1$ and $A_2 - \lambda B_2$ are *strictly equivalent* if there exist constant (independent of λ) invertible matrices P of size $m \times m$ and Q of size $n \times n$ such that

$$P^{-1}(A_1 - \lambda B_1)Q = A_2 - \lambda B_2.$$

Kronecker has shown that any matrix pencil is strictly equivalent to a canonical diagonal form that describes the structure elements of $A - \lambda B$ (including generalized eigenvalues and eigenspaces) in full detail (e.g. see [15]). This form is a generalization of the Jordan canonical form (JCF) to general matrix pencils.

3.1 Kronecker canonical form

The *Kronecker canonical form* (KCF) of $A - \lambda B$ exhibits the fine structure elements, including elementary divisors (Jordan blocks) and minimal indices, and is defined as follows [15]. Suppose $A, B \in \mathbf{C}^{m\times n}$. Then there exist nonsingular $P \in \mathbf{C}^{m\times m}$ and $Q \in \mathbf{C}^{n\times n}$ such that

$$P^{-1}(A - \lambda B)Q = S - \lambda T, \tag{3.1}$$

where $S = \text{diag}\,(S_{11}, \ldots, S_{bb})$ and $T = \text{diag}\,(T_{11}, \ldots, T_{bb})$ are block diagonal. $S_{ii} - \lambda T_{ii}$ is $m_i \times n_i$. We can partition the columns of P and Q into blocks corresponding to the blocks of $S - \lambda T$: $P = [P_1, \ldots, P_b]$ where P_i is $m \times m_i$, and $Q = [Q_1, \ldots, Q_b]$ where Q_i is $n \times n_i$. Each block $M_i \equiv S_{ii} - \lambda T_{ii}$ must be of one of the following forms: $J_j(\alpha)$, N_j, L_j or L_j^T. First we consider

$$J_j(\alpha) \equiv \begin{bmatrix} \alpha-\lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \alpha-\lambda \end{bmatrix} \quad \text{and} \quad N_j \equiv \begin{bmatrix} 1 & -\lambda & & \\ & \ddots & \ddots & \\ & & \ddots & -\lambda \\ & & & 1 \end{bmatrix} \tag{3.2}$$

$J_j(\alpha)$ is simply a $j \times j$ Jordan block, and α is called a *finite eigenvalue.* N_j is a $j \times j$ block corresponding to an *infinite eigenvalue* of multiplicity j. The $J_j(\alpha)$

and N_j blocks together constitute the *regular structure* of the pencil. All the $S_{ii} - \lambda T_{ii}$ are regular blocks if and only if $A - \lambda B$ is a regular pencil. $\sigma(A - \lambda B)$ denotes the eigenvalues of the regular part of $A - \lambda B$ (with multiplicities), and is called the *spectrum* of $A - \lambda B$.

The other two types of diagonal blocks are

$$L_j \equiv \begin{bmatrix} -\lambda & 1 & & \\ & \cdot & \cdot & \\ & & -\lambda & 1 \end{bmatrix} \quad \text{and} \quad L_j^T \equiv \begin{bmatrix} -\lambda & & \\ 1 & \cdot & \\ & \cdot & -\lambda \\ & & 1 \end{bmatrix} \tag{3.3}$$

The $j \times (j+1)$ block L_j is called a *singular block of right (or column) minimal index* j. It has a one dimensional right null space, $[1, \lambda, \ldots, \lambda^j]^T$, for any λ. The $(j+1) \times j$ block L_j^T is a *singular block of left (or row) minimal index* j, and has a one dimensional left null space for any λ. The left and right singular blocks together constitute the *singular structure* of the pencil, and appear in the KCF if and only if the pencil is singular. The regular and singular structures define the *Kronecker structure* of a singular pencil.

We also have a real KCF associated with real matrix pencils. If $A, B \in \mathbf{R}^{m \times n}$, there exist nonsingular $P \in \mathbf{R}^{m \times m}$ and $Q \in \mathbf{R}^{n \times n}$, where as before $P^{-1}(A - \lambda B)Q = S - \lambda T$ is block diagonal. The only difference with (3.1) is the Jordan blocks associated with complex conjugate pairs of eigenvalues. Let $\alpha = \mu + i\omega$, where μ, ω are real and $\omega \neq 0$. If α is an eigenvalue of $A - \lambda B$, then also $\bar{\alpha}$ is an eigenvalue. Let $J_j(\alpha, \bar{\alpha})$ denote a Jordan block of size $2j \times 2j$ associated with a complex conjugate pair of eigenvalues, here illustrated with the case $j = 3$:

$$J_3(\alpha, \bar{\alpha}) \equiv \begin{bmatrix} \mu - \lambda & \omega & 1 & 0 & 0 & 0 \\ -\omega & \mu - \lambda & 0 & 1 & 0 & 0 \\ 0 & 0 & \mu - \lambda & \omega & 1 & 0 \\ 0 & 0 & -\omega & \mu - \lambda & 0 & 1 \\ 0 & 0 & 0 & 0 & \mu - \lambda & \omega \\ 0 & 0 & 0 & 0 & -\omega & \mu - \lambda \end{bmatrix}. \tag{3.4}$$

The Jordan block $J_j(\alpha, \bar{\alpha})$ plays the same role in the real Jordan canonical form as $\mathrm{diag}(J_j(\alpha),\ J_j(\bar{\alpha}))$ does in the complex JCF. Notice that each pair of the $2j$ columns of the real P and Q associated with a $J_j(\alpha, \bar{\alpha})$ block form the real and imaginary parts of the (generalized) principal chains corresponding to the complex conjugate pair of eigenvalues.

3.2 Generalized Schur form and reducing subspaces

In most applications it is enough to transfer $A-\lambda B$ to a *generalized Schur form* (e.g. to GUPTRI form [11, 12])

$$P^H(A-\lambda B)Q = \begin{bmatrix} A_r-\lambda B_r & * & * \\ 0 & A_{reg}-\lambda B_{reg} & * \\ 0 & 0 & A_l-\lambda B_l \end{bmatrix}, \tag{3.5}$$

where P $(m \times m)$ and Q $(n \times n)$ are unitary and $*$ denotes arbitrary conforming submatrices. Here the square upper triangular block $A_{reg}-\lambda B_{reg}$ is regular and has the same regular structure as $A-\lambda B$ (i.e., contains all eigenvalues (finite and infinite) of $A-\lambda B$). The rectangular blocks $A_r-\lambda B_r$ and $A_l-\lambda B_l$ contain the singular structure (right and left minimal indices) of the pencil and are block upper triangular.

$A_r-\lambda B_r$ has only right minimal indices in its Kronecker canonical form (KCF), indeed the same L_j blocks as $A-\lambda B$. Similarly, $A_l-\lambda B_l$ has only left minimal indices in its KCF, the same L_j^T blocks as $A-\lambda B$. If $A-\lambda B$ is singular at least one of $A_r-\lambda B_r$ and $A_l-\lambda B_l$ will be present in (3.5). The explicit structure of the diagonal blocks in staircase form can be found in [12]. If $A-\lambda B$ is regular $A_r-\lambda B_r$ and $A_l-\lambda B_l$ are not present in (3.5) and the GUPTRI form reduces to the upper triangular block $A_{reg}-\lambda B_{reg}$. Staircase forms that reveal the Jordan structure of the zero and infinite eigenvalues are contained in $A_{reg}-\lambda B_{reg}$.

Given $A-\lambda B$ in GUPTRI form we also know different pairs of reducing subspaces [31, 11]. Suppose the eigenvalues on the diagonal of $A_{reg}-\lambda B_{reg}$ are ordered so that the first k, say, are in Λ_1 (a subset of the spectrum) and the remainder are outside Λ_1. Let $A_r-\lambda B_r$ be $m_r \times n_r$. Then the left and right reducing subspaces corresponding to Λ_1 are spanned by the leading m_r+k columns of P and leading n_r+k columns of Q, respectively. When Λ_1 is empty, the corresponding reducing subspaces are called *minimal*, and when Λ_1 contains the whole spectrum the reducing subspaces are called *maximal*.

Several authors have proposed (staircase-type) algorithms for computing a generalized Schur form (e.g. see [2, 21, 23, 22, 29, 34]). They are numerically stable in the sense that they compute the exact Kronecker structure (generalized Schur form or something similar) of a nearby pencil $A'-\lambda B'$. $\delta \equiv \|(A-A', B-B')\|_E$ is an upper bound on the distance to the closest $(A+\delta A, B+\delta B)$ with the KCF of (A', B'). Recently, robust software with error bounds for computing the GUPTRI form of a singular $A-\lambda B$ has been published [11, 12]. Some computational experiments that use this software will be discussed later.

3.3 Generic and non-generic Kronecker structures

Although, the KCF looks quite complicated in the general case, most matrix pencils have a quite simple Kronecker structure. If $A - \lambda B$ is $m \times n$, where $m \neq n$, then for almost all A and B it will have the same KCF, depending only on m and n. This corresponds to the *generic case* when $A - \lambda B$ has full rank for any complex (or real) value of λ. Accordingly, generic rectangular pencils have no regular part. The generic Kronecker structure for $A - \lambda B$ with $d = n - m > 0$ is

$$\operatorname{diag}(L_\alpha, \ldots, L_\alpha, L_{\alpha+1}, \ldots, L_{\alpha+1}), \tag{3.6}$$

where $\alpha = \lfloor m/d \rfloor$, the total number of blocks is d, and the number of $L_{\alpha+1}$ blocks is $m \bmod d$ (which is 0 when d divides m) [29, 8]. The same statement holds for $d = m - n > 0$ if we replace $L_\alpha, L_{\alpha+1}$ in (3.3) by $L_\alpha^T, L_{\alpha+1}^T$. Square pencils are generically regular, i.e., $\det(A - \lambda B) = 0$ if and only if λ is an eigenvalue. The generic singular pencils of size n-by-n have the Kronecker structures [32]:

$$\operatorname{diag}(L_j, L_{n-j-1}^T), \quad j = 0, \ldots, n-1. \tag{3.7}$$

Only if a singular $A - \lambda B$ is rank deficient (for some λ) may the associated KCF be more complicated and possibly include a regular part, as well as, right and left singular blocks. This situation corresponds to the *non-generic case*, which of course is the real challenge from a computational point of view.

The generic and non-generic cases can easily be couched in terms of reducing subspaces. For example, generic rectangular pencils have only trivial reducing subspaces and no generalized eigenvalues at all. Generic square singular pencils have the same minimal and maximal reducing subspaces. A non-generic case corresponds to that $A - \lambda B$ lies in a particular manifold of the matrix pencil space and that the pencil has nontrivial reducing subspaces. Moreover, only if it is perturbed so as to move continuously within that manifold do its reducing subspaces and generalized eigenvalues also move continuously and satisfy interesting error bounds [9, 11]. These requirements are natural in many control and systems theoretic problems such as computing controllable subspaces and uncontrollable modes.

4 The Geometry of Matrix Pencil Space

In this section we derive formulas for the tangent and normal spaces of the orbit of a matrix pencil that we will make use of in order to compute the versal form in the next section. We also derive new bounds for the distance to less generic pencils.

4.1 The orbit of a matrix pencil and its tangent and normal spaces

Any $m \times n$ matrix pair (A, B) (with real or complex entries) defines a manifold of *strictly equivalent* matrix pencils in the $2mn$ dimensional space $\mathcal{P}$ of m-by-n pencils:

$$\text{orbit}(A - \lambda B) = \{P^{-1}(A - \lambda B)Q : \det(P)\det(Q) \neq 0\}. \tag{4.1}$$

We may choose a special element of orbit$(A - \lambda B)$ that reveals the KCF of the pencil.

As usual the dimension of orbit$(A - \lambda B)$ is equal to the dimension of the tangent space to the orbit at $A - \lambda B$, here denoted $\tan(A - \lambda B)$. By considering the deformation $(I_m + \delta X)(A - \lambda B)(I_n - \delta Y)$ of $A - \lambda B$ to first order term in δ, where δ is a small scalar, we obtain $A - \lambda B + \delta(X(A - \lambda B) - (A - \lambda B)Y) + O(\delta^2)$, from which it is evident that $\tan(A - \lambda B)$ consists of the pencils that can be represented in the form

$$T_A - \lambda T_B = (XA - AY) - \lambda(XB - BY), \tag{4.2}$$

where X is an $m \times m$ matrix and Y is an $n \times n$ matrix.

Using Kronecker products we can represent the $2mn$-vectors $T_A - \lambda T_B \in \tan(A - \lambda B)$ as

$$\begin{bmatrix} \text{vec}(T_A) \\ \text{vec}(T_B) \end{bmatrix} = \begin{bmatrix} A^T \otimes I_m \\ B^T \otimes I_m \end{bmatrix} \text{vec}(X) - \begin{bmatrix} I_n \otimes A \\ I_n \otimes B \end{bmatrix} \text{vec}(Y).$$

In this notation, we may say that the tangent space is the range of the $2mn \times (m^2 + n^2)$ matrix

$$T \equiv \begin{bmatrix} A^T \otimes I_m & -I_n \otimes A \\ B^T \otimes I_m & -I_n \otimes B \end{bmatrix}. \tag{4.3}$$

We may define the *normal* space, nor$(A - \lambda B)$, as the space perpendicular to $\tan(A - \lambda B)$. Orthogonality in $\mathcal{P}$, the $2mn$ dimensional space of matrix pencils is defined with respect to a Frobenius inner product

$$< A - \lambda B, C - \lambda D > \equiv \text{tr}(AC^H + BD^H), \tag{4.4}$$

where $\text{tr}(X)$ denotes the trace of a square matrix X. Remembering that the space orthogonal to the range of a matrix is the kernel of the Hermitian transpose, we have that

$$\text{nor}(A - \lambda B) = \ker(T^H) = \ker \begin{bmatrix} \bar{A} \otimes I_m & \bar{B} \otimes I_m \\ -I_n \otimes A^H & -I_n \otimes B^H \end{bmatrix}.$$

In ordinary matrix notation, this states that $Z_A - \lambda Z_B$ is in the normal space of $A - \lambda B$ if and only if

$$Z_A A^H + Z_B B^H = 0 \quad \text{and} \quad A^H Z_A + B^H Z_B = 0. \tag{4.5}$$

The conditions on Z_A and Z_B can easily be verified and also be derived in terms of the Frobenius inner product, i.e.,

$$< T_A - \lambda T_B, Z_A - \lambda Z_B > = \mathrm{tr}(X(AZ_A^H + BZ_B^H) - (Z_A^H A + Z_B^H B)Y). \quad (4.6)$$

Verification: if the conditions (4.5) are satisfied, it follows from (4.6) that the inner product is zero. Derivation: if $< T_A - \lambda T_B, Z_A - \lambda Z_B > = 0$, then $\mathrm{tr}(X(AZ_A^H + BZ_B^H) - (Z_A^H A + Z_B^H B)Y) = 0$ must hold for any X (of size $m \times m$) and Y (of size $n \times n$). By choosing $X \equiv 0$, (4.6) reduces to $\mathrm{tr}((Z_A^H A + Z_B^H B)Y) = 0$, which holds for any Y if and only if $Z_A^H A + Z_B^H B = 0$. Similarly, we can chose $Y \equiv 0$, which gives that $AZ_A^H + BZ_B^H = 0$.

If $B = I$, this reduces to $Z_A \in$ nor(A) if and only if $Z_A^H \in$ centralizer(A), which is a well-known fact (e.g. see [1]). We will see in Section 5.3 that though the A-part of the normal space is very simple when $B = I$, obtaining an orthonormal basis for the B-part is particularly challenging. The requirement that $Z_B = -A^H Z_A$ when $B = I$ destroys any orthogonality one may have in a basis for the A-part.

We now collect our general statements and a few obvious consequences:

Theorem 4.1 *Let the $m \times n$ pencil $A - \lambda B$ be given. Define the $2mn \times (m^2 + n^2)$ matrix T as in (4.3). Then*

$$\tan(A - \lambda B) = \mathrm{range}(T) = \{(XA - AY) - \lambda(XB - BY)\},$$

where X and Y are compatible square matrices, and

$$\mathrm{nor}(A - \lambda B) = \ker(T^H) = \{Z_A - \lambda Z_B\},$$

where $Z_A A^H + Z_B B^H = 0$ and $\quad A^H Z_A + B^H Z_B = 0$.
The dimensions of these spaces are

$$\dim(\tan(A - \lambda B)) = m^2 + n^2 - \dim(\ker(T)), \quad (4.7)$$

and

$$\dim(\mathrm{nor}(A - \lambda B)) = \dim(\ker(T^H)) = \dim(\ker(T)) - (m - n)^2. \quad (4.8)$$

Of course, the tangent and normal spaces are complementary and span the complete $2mn$ dimensional space, i.e., $\mathcal{P} = \tan(A - \lambda B) \oplus \mathrm{nor}(A - \lambda B)$, so that the dimensions in (4.7) and (4.8) add up to $2mn$ as they should.

Theorem 4.1 leads to one approach for computing a basis for nor$(A - \lambda B)$ from the singular value decomposition (SVD) of T. Indeed, the left singular vectors corresponding to the zero singular value form such a basis. The dimension of the normal space is also known as the *codimension* of the orbit, here denoted cod$(A - \lambda B)$. Accordingly, we have the following "compact" characterization of the codimension of orbit$(A - \lambda B)$.

Corollary 4.1 *Let the $m \times n$ pencil $A - \lambda B$ be given. Then,*

$$\text{cod}(A - \lambda B) = \text{the number of zero singular values of } T. \tag{4.9}$$

The corresponding result for the (square) matrix case is

$$\text{cod}(A) = \text{the number of zero singular values of } I_n \otimes A - A^T \otimes I_n.$$

Although, the *SVD*-based method is simple and has nice numerical properties (backward stability), it is rather costly in the number of operations. Computing the *SVD* of T is an $O(m^3n^3)$ operation.

Knowing the Kronecker structure of $A - \lambda B$, it is also possible to compute the codimension of the orbit as the sum of separate codimensions [8]:

$$\text{cod}(A - \lambda B) = c_{\text{Jor}} + c_{\text{Right}} + c_{\text{Left}} + c_{\text{Jor,Sing}} + c_{\text{Sing}}. \tag{4.10}$$

The different contributions in (4.10) originate from the Jordan structure of all eigenvalues (including any infinite eigenvalue), the right singular blocks ($L_j \leftrightarrow L_k$), the left singular blocks ($L_j^T \leftrightarrow L_k^T$), interactions of the Jordan structure with the singular blocks (L_k and L_j^T) and interactions between the left and right singular structures ($L_j \leftrightarrow L_k^T$), respectively. Explicit expressions for these codimensions are derived in [8]. Assume that the given $A - \lambda B$ has $p \leq \min(m, n)$ distinct eigenvalues $\lambda_i, i = 1 : p$ with p_i Jordan blocks each. Let $q_1(\lambda_i) \geq q_2(\lambda_i) \geq \ldots \geq q_{p_i}(\lambda_i)$ denote the sizes of the Jordan blocks corresponding to the eigenvalue λ_i. Then the separate codimensions of (4.10) can be expressed as

$$c_{\text{Jor}} = \sum_{i=1}^{p} \sum_{j=1}^{p_i} (2j-1) q_j(\lambda_i) = \sum_{i=1}^{p} (q_1(\lambda_i) + 3q_2(\lambda_i) + 5q_3(\lambda_i) + \ldots),$$

$$c_{\text{Right}} = \sum_{j>k} (j-k-1), \quad c_{\text{Left}} = \sum_{j>k} (j-k-1), \quad c_{\text{Sing}} = \sum_{j,k} (j+k+2),$$

$$c_{\text{Jor,Sing}} = (\text{size of complete regular part}) \cdot (\text{number of singular blocks}).$$

Notice that if we do not wish to specify the value of an eigenvalue λ_i, the codimension count for this unspecified eigenvalue is one less, i.e.,

$$-1 + q_1(\lambda_i) + 3q_2(\lambda_i) + 5q_3(\lambda_i) + \ldots$$

This is sometimes done in algorithms for computing the Kronecker structure of a matrix pencil, where usually only the eigenvalues 0 and ∞ are specified and the remaining ones are unspecified.

It is possible to extract the Kronecker structure of $A - \lambda B$ from a generalized Schur decomposition in $O((\max(m, n))^3)$ operations. The most reliable *SVD*-approach for computing a generalized Schur decomposition of $A - \lambda B$ requires at most $O((\max(m,n))^4)$ operations, which is still small compared to computing the *SVD* of T (4.3) for already moderate values of m and n (e.g. when $m = n$).

For given m and n the generic pencil has codimension 0 (i.e., span the complete $2mn$ dimensional space) while the most non-generic matrix pair $(A, B) = (0_{m\times n}, 0_{m\times n})$ has codimension $= 2mn$ (i.e., defines a "point" in $2mn$ dimensional space). Accordingly, any $m \times n$ non-generic pencil different from the "zero pencil" has a codimension ≥ 1 and $< 2mn$.

4.2 A lower bound on the distance to a less generic pencil

The *SVD* characterization of the codimension of orbit$(A - \lambda B)$ in Corollary 4.1 leads to the following theorem from which we present an interesting special case as a corollary.

Theorem 4.2 *For a given $m \times n$ pencil $A - \lambda B$ with codimension c, a lower bound on the distance to the closest pencil $(A+\delta A)-\lambda(B+\delta B)$ with codimension $c + d$, where $d \geq 1$ is given by*

$$\|(\delta A, \delta B)\|_E \geq \frac{1}{\sqrt{m+n}} \left(\sum_{i=2mn-c-d+1}^{2mn} \sigma_i^2(T) \right)^{1/2}, \tag{4.11}$$

where $\sigma_i(T)$ denotes the i*th largest singular value of T $(\sigma_i(T) \geq \sigma_{i+1}(T) \geq 0)$.*

Proof. It follows from Corollary 4.1 that T has rank $= 2mn - c$ if and only if $A - \lambda B$ has codimension c and $(A + \delta A) - \lambda(B + \delta B)$ has codimension $c + d$, $(d \geq 1)$ if and only if $T + \delta T$, where δT is defined as

$$\delta T \equiv \begin{bmatrix} \delta A^T \otimes I_m & -I_n \otimes \delta A \\ \delta B^T \otimes I_m & -I_n \otimes \delta B \end{bmatrix}, \tag{4.12}$$

has rank $2mn - c - d$. From the construction, it follows that $\|\delta T\|_E = \sqrt{m+n}$ $\|(\delta A, \delta B)\|_E$ (each element δa_{ij} and δb_{ij} appears $m + n$ times in δT). The Eckart–Young and Mirsky theorem for finding the closest matrix of a given rank (e.g. see [16]), gives that the size of the smallest perturbation in Frobenius norm that reduces the rank in T from $2mn - c$ to $2mn - c - d$ is

$$\left(\sum_{i=2mn-c-d+1}^{2mn-c} \sigma_i^2(T) \right)^{1/2}. \tag{4.13}$$

Moreover, $A - \lambda B$ has codimension c implies that $\sigma_{2mn-c+1}(T) = \ldots = \sigma_{2mn}(T) = 0$. Since $\|\delta T\|_E$ must be larger than or equal to the quantity (4.13), the proof is complete. □

Corollary 4.2 *For a given generic $m \times n$ pencil $A - \lambda B$, a lower bound on the distance to the closest non-generic pencil $(A + \delta A) - \lambda(B + \delta B)$ is given by*

$$\|(\delta A, \delta B)\|_E \geq \frac{\sigma_{\min}(T)}{\sqrt{m+n}}, \tag{4.14}$$

where $\sigma_{\min}(T) = \sigma_{2mn}(T)$ denotes the smallest singular value of T, which is non-zero for a generic $A - \lambda B$.

We remark that the set of $m \times n$ matrix pencils does not include orbits of all codimensions from 1 to $2mn$.

One application of Corollary 4.2 is to characterize the distance to uncontrollability for a multiple input multiple output linear system $E\dot{x}(t) = Fx(t) + Gu(t)$, where E and F are p-by-p matrices, G is p-by-q ($p \geq q$), and E is assumed to be nonsingular. If $A - \lambda B \equiv [G|F - \lambda E]$ is generic, the linear system is controllable (i.e., the dimension of the controllable subspace equals p) and a lower bound on the distance to the closest uncontrollable system is given by (4.14).

5 Versal Deformations for the Kronecker Canonical Form

In this section, we derive versal deformations which for us will mean the decomposition of arbitrary perturbations into the tangent and normal spaces of the orbits of equivalent pencils.

5.1 An introductory example

We start with a small example before considering the general case. Let $A - \lambda B = L_1 \oplus L_4$ with codimension $= 2$. (This means that the manifold orbit$(A - \lambda B)$ has codimension 2 or dimension 68 in the 70 dimensional space of 5×7 pencils.) Since $A - \lambda B$ already is in KCF we know its block structure:

$$A - \lambda B = \left[\begin{array}{cc|ccccc} -\lambda & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -\lambda & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\lambda & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\lambda & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\lambda & 1 \end{array}\right].$$

From (4.2) the matrices in the tangent space are given by $T_A - \lambda T_B =$

$(XA - AX) - \lambda(XB - BX)$, where

$$T_A = \left[\begin{array}{cc|ccccc} -y_{21} & x_{11}-y_{22} & -y_{23} & x_{12}-y_{24} & x_{13}-y_{25} & x_{14}-y_{26} & x_{15}-y_{27} \\ \hline \boxed{\boxed{-y_{41}}} & x_{21}-y_{42} & -y_{43} & x_{22}-y_{44} & x_{23}-y_{45} & x_{24}-y_{46} & x_{25}-y_{47} \\ \boxed{-y_{51}} & \boxed{\boxed{x_{31}-y_{52}}} & -y_{53} & x_{32}-y_{54} & x_{33}-y_{55} & x_{34}-y_{56} & x_{35}-y_{57} \\ -y_{61} & \boxed{x_{41}-y_{62}} & -y_{63} & x_{42}-y_{64} & x_{43}-y_{65} & x_{44}-y_{66} & x_{45}-y_{67} \\ -y_{71} & x_{51}-y_{72} & -y_{73} & x_{52}-y_{74} & x_{53}-y_{75} & x_{54}-y_{76} & x_{55}-y_{77} \end{array}\right]$$

and

$$T_B = \left[\begin{array}{cc|ccccc} x_{11}-y_{11} & -y_{12} & x_{12}-y_{13} & x_{13}-y_{14} & x_{14}-y_{15} & x_{15}-y_{16} & -y_{17} \\ \hline x_{21}-y_{31} & -y_{32} & x_{22}-y_{33} & x_{23}-y_{34} & x_{24}-y_{35} & x_{25}-y_{36} & -y_{37} \\ \boxed{\boxed{x_{31}-y_{41}}} & -y_{42} & x_{32}-y_{43} & x_{33}-y_{44} & x_{34}-y_{45} & x_{35}-y_{46} & -y_{47} \\ \boxed{x_{41}-y_{51}} & \boxed{\boxed{-y_{52}}} & x_{42}-y_{53} & x_{43}-y_{54} & x_{44}-y_{55} & x_{45}-y_{56} & -y_{57} \\ x_{51}-y_{61} & \boxed{-y_{62}} & x_{52}-y_{63} & x_{53}-y_{64} & x_{54}-y_{65} & x_{55}-y_{66} & -y_{67} \end{array}\right].$$

By inspection we find the following two relations between elements in T_A and T_B:

$$\boxed{\boxed{}}: \ t^a_{21} + t^a_{32} = t^b_{31} + t^b_{42},$$

and

$$\boxed{}: \ t^a_{31} + t^a_{42} = t^b_{41} + t^b_{52}$$

where t^a_{ij} and t^b_{ij} denote the (i,j)-th elements of T_A and T_B, respectively. These two relations show clearly that the tangent space has codimension at least two. It may be verified that the other parameters may be chosen arbitrarily so that the codimension is exactly two.

We want to find $Z_A - Z_B$ that is orthogonal to $T_A - \lambda T_B$ with respect to the Frobenius inner product, i.e.,

$$0 \equiv < T_A - \lambda T_B, Z_A - \lambda Z_B > \equiv \mathrm{tr}(T_A Z_A^H + T_B Z_B^H) \equiv \sum_{i,j} t^a_{ij}\bar{z}^a_{ij} + t^b_{ij}\bar{z}^b_{ij}. \quad (5.1)$$

This inner product is most easily envisioned as the sum of the elementwise multiplication of the two pencils. Using this point of view it is obvious that the normal space consists of pencils of the form $Z_A - \lambda Z_B \in \mathrm{nor}(\mathrm{A} - \lambda\mathrm{B})$:

$$Z_A - \lambda Z_B = \left[\begin{array}{cc|ccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline p_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_2 & p_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & p_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] - \lambda \left[\begin{array}{cc|ccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -p_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -p_2 & -p_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -p_2 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

$$
= \left[\begin{array}{cc|ccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
p_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
p_2 + \lambda\, p_1 & p_1 & 0 & 0 & 0 & 0 & 0 \\
\lambda\, p_2 & p_2 + \lambda\, p_1 & 0 & 0 & 0 & 0 & 0 \\
0 & \lambda\, p_2 & 0 & 0 & 0 & 0 & 0
\end{array}\right],
$$

where p_1 and p_2 are arbitrary. Roughly speaking the parameter p_1 corresponds to the doubly boxed entries ($\boxed{\square}$) and the parameter p_2 corresponds to the singly boxed entries ($\boxed{}$).

Now, $A - \lambda B + Z_A - \lambda Z_B$ may be thought of as a versal deformation, or normal form, with minimum number of parameters (equal to the codimension of the original pencil). It follows that any (complex) pencil close to the given $A - \lambda B$ can be reduced to the 2-parameter normal form $A - \lambda B + Z_A - \lambda Z_B$, where $A - \lambda B$ is in Kronecker canonical form.

5.2 Notation: a glossary of Toeplitz and Hankel matrices

The example in the previous section shows that a non-zero block of $Z_A - \lambda Z_B$ has a structured form. Indeed, the $(2,1)$ block has a Toeplitz-like form with $j - i = 3$ non-zero diagonals starting from the $(1,1)$-element of the $(2,1)$ block. A closer look shows that the A-part has $i - j - 1 = 2$ non-zero diagonals and the B-part is just the same matrix negated and with the diagonals shifted one row downwards. In general, different non-zero blocks with Toeplitz or Hankel properties will show up in $Z_A - \lambda Z_B \in \mathrm{nor}(A - \lambda B)$. To simplify the proof of the general case we introduce some Toeplitz and Hankel matrices. Arrows and "stops" near the matrices make clear how the matrix is defined.

Let $S^L_{s\times t}$ be a *lower trapezoidal* s-by-t Toeplitz matrix with the first non-zero diagonal starting at position $(1,1)$:

$$
S^L_{s\times t} = \begin{array}{c} \downarrow \\ \\ \\ \\ \\ \perp \end{array}
\begin{bmatrix}
p_1 & 0 & 0 \\
\vdots & \ddots & 0 \\
\vdots & & p_1 \\
p_{s-t+1} & & \vdots \\
\vdots & \ddots & \vdots \\
p_s & \cdots & p_{s-t+1}
\end{bmatrix}
\quad \text{if } s \geq t, \text{ and } \quad
S^L_{s\times t} = \begin{array}{c} \downarrow \\ \\ \perp \end{array}
\begin{bmatrix}
p_1 & 0 & & & \cdots & 0 \\
\vdots & \ddots & \ddots & & & \vdots \\
p_s & \cdots & p_1 & 0 & \cdots & 0
\end{bmatrix},
$$

otherwise, and let $T^L_{s\times t}$ be a *lower trapezoidal* s-by-t Toeplitz matrix with the first non-zero diagonal's last element at position (s,t):

$$T^L_{s\times t}=\begin{array}{c}\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & & \vdots \\ p_1 & \ddots & \vdots \\ \vdots & \ddots & 0 \\ p_t & \cdots & p_1 \end{bmatrix} \\ \begin{array}{ccc} \vdash & \quad & \leftarrow \end{array}\end{array} \text{ if } s \geq t, \text{ and } T^L_{s\times t}=\begin{array}{c}\begin{bmatrix} p_{t-s+1} & \cdots & p_1 & 0 & 0 \\ \vdots & \ddots & & \ddots & 0 \\ p_t & \cdots & p_{t-s+1} & \cdots & p_1 \end{bmatrix} \\ \begin{array}{ccc} \vdash & \qquad\qquad\qquad & \leftarrow \end{array}\end{array}, \text{ otherwise.}$$

If $s<t$, the entries of the last $t-s$ columns of $S^L_{s\times t}$ are zero. Similarly, if $s \geq t$, the entries of the first $s-t$ rows of $T^L_{s\times t}$ are zero.

Let $S^B_{s\times t}$ be a *banded lower trapezoidal* s-by-t Toeplitz with last row 0:

$$S^B_{s\times t} = \begin{array}{c} \downarrow \\ \\ \\ \perp \\ \\ \\ \\ \end{array} \begin{bmatrix} p_1 & 0 & 0 \\ \vdots & \ddots & 0 \\ \vdots & & p_1 \\ p_{s-t} & & \vdots \\ 0 & \ddots & \vdots \\ \vdots & \ddots & p_{s-t} \\ 0 & \cdots & 0 \end{bmatrix} \text{ if } s>t, \text{ and } S^B_{s\times t}=0, \text{ otherwise,}$$

and let $T^B_{s\times t}$ be another *banded lower trapezoidal* s-by-t Toeplitz matrix, this time with last column 0:

$$T^B_{s\times t} = \begin{array}{c} \begin{array}{ccc} \vdash & \qquad & \leftarrow \end{array} \\ \begin{bmatrix} p_{t-s} & \cdots & p_1 & 0 & \cdots 0 \\ 0 & \ddots & & \ddots & \ddots \vdots \\ 0 & 0 & p_{t-s} & \cdots & p_1\ 0 \end{bmatrix}\end{array} \text{ if } s<t, \text{ and } T^B_{s\times t}=0, \text{ otherwise.}$$

Notice that the last row of $S^B_{s\times t}$ (if $s>t$) and the last column of $T^B_{s\times t}$ (if $s<t$) have all entries equal to zero.

Moreover, let $H^L_{s\times t}$ be a *lower trapezoidal* s-by-t Hankel matrix with the first non-zero diagonal starting at position $(1, t)$:

$$H^L_{s\times t} = \begin{bmatrix} 0 & 0 & p_1 \\ 0 & \iddots & \vdots \\ p_1 & & \vdots \\ \vdots & & p_{s-t+1} \\ \vdots & \iddots & \vdots \\ p_{s-t+1} & \cdots & p_s \end{bmatrix}\begin{matrix}\downarrow \\ \\ \\ \\ \\ \perp\end{matrix} \quad \text{if } s \geq t, \text{ and } \quad H^L_{s\times t} = \begin{bmatrix} 0 & \cdots & & 0 & p_1 \\ \vdots & & \iddots & \iddots & \vdots \\ 0 \cdots 0 & p_1 & \cdots & & p_s \end{bmatrix}\begin{matrix}\downarrow \\ \\ \perp\end{matrix},$$

otherwise, and let $H^U_{s\times t}$ be a similar *upper trapezoidal* s-by-t Hankel matrix:

$$H^U_{s\times t} = \begin{array}{c} \begin{matrix}\vdash & \quad & \leftarrow\end{matrix} \\ \begin{bmatrix} p_t & \cdots & p_1 \\ \vdots & \iddots & 0 \\ p_1 & \iddots & \vdots \\ 0 & & \vdots \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}\end{array} \quad \text{if } s \geq t, \text{ and } \quad H^U_{s\times t} = \begin{array}{c} \begin{matrix}\vdash & \qquad\qquad\qquad\qquad & \leftarrow\end{matrix} \\ \begin{bmatrix} p_t & \cdots & p_{t-s+1} & \cdots & p_1 \\ \vdots & \iddots & & \iddots & 0 \\ p_{t-s+1} & \cdots & p_1 & 0 & 0 \end{bmatrix}\end{array},$$

otherwise. If $s < t$, the entries of the first $t - s$ columns of $H^L_{s\times t}$ are zero. Similarly, if $s \geq t$, the entries of the last $s - t$ rows of $H^U_{s\times t}$ are zero.

Let $H_{s\times t}$ be a *dense* s-by-t Hankel matrix (with the first diagonal starting at position $(1, 1)$):

$$H_{s\times t} = \begin{bmatrix} p_1 & p_2 & p_3 \cdots & p_t \\ p_2 & \iddots & & \vdots \\ p_3 & & & \vdots \\ \vdots & & & \vdots \\ p_s & & \cdots & p_{s+t-1} \end{bmatrix},$$

for both the cases $s \geq t$ and $s < t$.

The *nilpotent* k-by-k matrix

$$C_k = \begin{bmatrix} 0 & I_{k-1} \\ 0 & 0 \end{bmatrix},$$

will be used as a shift operator. For a given k-by-n matrix X, the rows are shifted one row upwards and downwards by the operations $C_k X$ and $C_k^T X$, respectively. The columns are shifted one column rightwards and one column leftwards in an n-by-k matrix X by the operations $X C_k$ and $X C_k^T$, respectively. The k-by-$(k+1)$ matrices

$$G_k = [I_k \; 0] \text{ and } \hat{G}_k = [0 \; I_k],$$

will be used to pick all rows but one or all columns but one of a given matrix X in the following way. The first k and last k rows in a $(k+1)$-by-n matrix X are picked by $G_k X$ and $\hat{G}_k X$, respectively. The k first and k last columns in an n-by-$(k+1)$ matrix X are picked by XG_k^T and $X\hat{G}_k^T$, respectively.

Let $\hat{I}_k$ denote the k-by-k matrix obtained by reversing the order of the columns in the k-by-k identity matrix. It follows that for an n-by-k matrix X, the order of the columns is reversed by the multiplication $X\hat{I}_k$.

So far, the matrices introduced are rectangular Toeplitz and Hankel matrices with a special structure, e.g. lower trapezoidal (S^L, T^L, H^L), banded lower trapezoidal (S^B, T^B), upper trapezoidal (H^U) or dense (H). The matrices C and $G, \hat{G}$ that will be used as "shift" and "pick" operators, respectively, are Toeplitz matrices with only one non-zero diagonal. In the next section we will see that versal deformations for all combinations of different blocks in the KCF, except Jordan blocks with non-zero, finite eigenvalues, can be expressed in terms of these matrices. To cope with non-zero, finite Jordan blocks $J_k(\gamma), \gamma \neq 0$ we need to introduce three more matrices. First, two lower triangular Toeplitz matrices D^L and E^L which are involved in the case with two $J_k(\gamma)$ blocks. Finally, the "monstrous" matrix F^D, which captures the cases with a (left or right) singular block and a $J_k(\gamma)$ block.

Given $\gamma \neq \{0, \infty\}$, define two infinite sequences of numbers d_i and e_i by the recursion

$$\begin{bmatrix} d_i \\ \gamma e_i \end{bmatrix} = -\begin{bmatrix} 1 & 1 \\ 1 & 2-1/i \end{bmatrix} \begin{bmatrix} \overline{\gamma} d_{i-1} \\ e_{i-1} \end{bmatrix}, \tag{5.2}$$

starting with

$$\begin{bmatrix} d_1 \\ e_1 \end{bmatrix} = \begin{bmatrix} \gamma \\ 1 \end{bmatrix}.$$

Given sizes s and t, for $1 \le q \le \min\{s,t\}$, we define $D_{s\times t}[q]$ and $E_{s\times t}[q]$ as *lower triangular* Toeplitz matrices with q diagonals in terms of $d_1, \ldots, d_q$ and $e_1, \ldots, e_{q-1}$ and a *boundary* value $e_q^* = -\overline{\gamma} d_q$.

$$D_{s\times t}[q] = \begin{bmatrix} 0 & & \cdots & & & 0 \\ d_q & & & & & \\ d_{q-1} & \ddots & & & & \\ \vdots & \ddots & \ddots & & & \vdots \\ d_2 & & \ddots & \ddots & & \\ d_1 & d_2 & \cdots & d_{q-1} & d_q & 0 \end{bmatrix} \text{ and } E_{s\times t}[q] = \begin{bmatrix} 0 & & \cdots & & & 0 \\ e_q^* & & & & & \\ e_{q-1} & \ddots & & & & \\ \vdots & \ddots & \ddots & & & \vdots \\ e_2 & & \ddots & \ddots & & \\ e_1 & e_2 & \cdots & e_{q-1} & e_q^* & 0 \end{bmatrix}.$$

We take linear combinations with parameters p_j to form the matrices

$$D_{s\times t}^L = \sum_{i=1}^{\min\{s,t\}} p_j D_{s\times t}[i]\,\pi(i), \quad \text{and} \quad E_{s\times t}^L = \sum_{i=1}^{\min\{s,t\}} p_j E_{s\times t}[i]\,\pi(i), \tag{5.3}$$

where $j = \min\{s,t\} - i + 1$ and $\pi(i) = -\prod_{k=2}^{i-1} k\gamma/(1-2k)$ is defined to be $1/\gamma$ and -1 for $i = 1$ and $i = 2$, respectively. The parameter index j and the scaling function $\pi(i)$ are chosen to satisfy $D^L_{s\times t} = S^L_{s\times t}$ and $E^L_{s\times t} = -C_s^T S^L_{s\times t}$ for $\gamma = 0$ in Theorem 5.1 (see tables 1 and 2). By simplifying (5.3) using $i = j$ and $\pi(i) = 1$ this consistency will be lost, but we will still have valid expressions for the versal deformations.

The relations between the elements of $D^L_{s\times t}$ and $E^L_{s\times t}$ are most readily shown by an example:

$$D^L_{4,3} = \left[\begin{array}{ccc} 0 & 0 & 0 \\ p_1\left(\frac{2|\gamma|^4}{3}+\frac{4|\gamma|^2}{3}+1\right) & 0 & 0 \\ p_1\left(-\frac{2\gamma|\gamma|^2}{3}-\frac{2\gamma}{3}\right)+p_2\left(|\gamma|^2+1\right) & p_1\left(\frac{2|\gamma|^4}{3}+\frac{4|\gamma|^2}{3}+1\right) & 0 \\ p_1\frac{2\gamma^2}{3}-p_2\gamma+p_3 & p_1\left(-\frac{2\gamma|\gamma|^2}{3}-\frac{2\gamma}{3}\right)+p_2\left(|\gamma|^2+1\right) & p_1\left(\frac{2|\gamma|^4}{3}+\frac{4|\gamma|^2}{3}+1\right) \end{array}\right],$$

and

$$E^L_{4,3} = \left[\begin{array}{ccc} 0 & 0 & 0 \\ p_1\left(-\frac{2\overline{\gamma}|\gamma|^4}{3}-\frac{4\overline{\gamma}|\gamma|^2}{3}-\overline{\gamma}\right) & 0 & 0 \\ p_1\left(-\frac{2|\gamma|^2}{3}-1\right)+p_2\left(-\overline{\gamma}|\gamma|^2-\overline{\gamma}\right) & p_1\left(-\frac{2\overline{\gamma}|\gamma|^4}{3}-\frac{4\overline{\gamma}|\gamma|^2}{3}-\overline{\gamma}\right) & 0 \\ p_1\frac{2\gamma}{3}-p_2-p_3\overline{\gamma} & p_1\left(-\frac{2|\gamma|^2}{3}-1\right)+p_2\left(-\overline{\gamma}|\gamma|^2-\overline{\gamma}\right) & p_1\left(-\frac{2\overline{\gamma}|\gamma|^4}{3}-\frac{4\overline{\gamma}|\gamma|^2}{3}-\overline{\gamma}\right) \end{array}\right].$$

Let $F^D_{s\times t}$ (D for dense) be defined as

$$F^D_{s\times t} = \sum_{i=1}^{s} p_{s-i+1}\ F_{s\times t}[i],$$

where $F_{s\times t}[q]$ has the q last rows non-zero and defined as:

$$\begin{array}{ll} f_{s-q+1,j} = \overline{\gamma}^{j-1} & \text{for } j = 1,\ldots,t, \\ f_{i,j} = \overline{\gamma} f_{i,j-1} + f_{i-1,j-1} & \text{for } i = s-q+2,\ldots,s,\ j = 2,\ldots,t, \end{array}$$

and $f_{i,1}$ for $i = s-q+2,\ldots,s$ is defined as the solution to

$$< F_{s\times t}[q]G^T_{t-1} - \lambda F_{s\times t}[q]\hat{G}^T_{t-1},\ F_{s\times t}[s-i+1]G^T_{t-1} - \lambda F_{s\times t}[s-i+1]\hat{G}^T_{t-1} > \equiv 0.$$

Notice that $f_{i,1}$ is used as an unknown in the generation of elements in (5.4). In the definition of $F_{s\times t}[q]$, the solutions for $f_{i,1}$ for $i = s-q+2,\ldots,s$ ensure that $F_{s\times t}[q]G^T_{t-1} - \lambda F_{s\times t}[q]\hat{G}^T_{t-1}$ is orthogonal to $F_{s\times t}[\hat{q}]G^T_{t-1} - \lambda F_{s\times t}[\hat{q}]\hat{G}^T_{t-1}$ for $\hat{q} = 1,\ldots,q-1$.

Also here we show a small example to facilitate the interpretation of the definition:

$$F^D_{3\times 2} = \begin{bmatrix} p_1 & p_1\overline{\gamma} \\ p_2 - p_1\frac{(|\gamma|^2+1)\gamma}{|\gamma|^4+2|\gamma|^2+2} & p_2\overline{\gamma} + p_1\frac{|\gamma|^2+2}{|\gamma|^4+2|\gamma|^2+2} \\ p_3 - p_2\frac{\gamma}{|\gamma|^2+1} + p_1\frac{\gamma^2}{|\gamma|^4+2|\gamma|^2+2} & p_3\overline{\gamma} + p_2\frac{1}{|\gamma|^2+1} - p_1\frac{\gamma}{|\gamma|^4+2|\gamma|^2+2} \end{bmatrix}$$

5.3 Versal deformations – the general case

Without loss of generality assume that $A-\lambda B$ already is in Kronecker canonical form, $M = \mathrm{diag}(M_1, M_2, \ldots, M_b)$, where each M_k is either a Jordan block associated with a finite or infinite eigenvalue or a singular block corresponding to a left or right minimal index. A pencil $T_A - \lambda T_B = XM - MY$ in the tangent space can be partitioned conformally with the pencil M so that $T^A_{ij} - \lambda T^B_{ij} = X_{ij}M_j - M_iY_{ij}$, where M_k is m_k-by-n_k, X_{ij} is m_i-by-m_j and Y_{ij} is n_i-by-n_j:

$$\begin{bmatrix} X_{11} & \cdots & X_{1b} \\ \vdots & \ddots & \vdots \\ X_{b1} & \cdots & X_{bb} \end{bmatrix} \begin{bmatrix} M_1 & & \\ & \ddots & \\ & & M_b \end{bmatrix} - \begin{bmatrix} M_1 & & \\ & \ddots & \\ & & M_b \end{bmatrix} \begin{bmatrix} Y_{11} & \cdots & Y_{1b} \\ \vdots & \ddots & \vdots \\ Y_{b1} & \cdots & Y_{bb} \end{bmatrix}.$$

Since the blocks $T^A_{ij} - \lambda T^B_{ij}, i,j = 1,\ldots,b$ are mutually independent, we can study the different blocks of $T_A - \lambda T_B$ separately. Let $Z^A_{ij} - \lambda Z^B_{ij}$ be conformally sized blocks of $Z_A - \lambda Z_B$. From (4.5) we know that $Z_A - \lambda Z_B$ is in the normal space if and only if $A^H Z_A + B^H Z_B = 0$ and $Z_A A^H + Z_B B^H = 0$. We obtain a simple result since A and B are block diagonal.

Proposition 5.1 *Assume that* $M = A - \lambda B = \mathrm{diag}(A_1, A_2, \ldots, A_b) - \lambda\mathrm{diag}(B_1, B_2, \ldots, B_b)$ *is in Kronecker canonical form, where each block* $A_i - \lambda B_i \equiv M_i$ *represents one block in the Kronecker structure. Then* $Z_A - \lambda Z_B \in \mathrm{nor}(A - \lambda B)$ *if and only if*

$$A_j^H Z^A_{ji} = -B_j^H Z^B_{ji} \quad \text{and} \quad Z^A_{ji} A_i^H = -Z^B_{ji} B_i^H, \text{ for } i = 1,\ldots,b \text{ and } j = 1,\ldots,b.$$

The mutual independency of the (i,j) blocks of Z_A and Z_B implies that we only have to consider two M_k blocks at a time:

$$T_A[i,j] - \lambda T_B[i,j] =$$

$$\begin{bmatrix} X_{ii} & X_{ij} \\ X_{ji} & X_{jj} \end{bmatrix} \begin{bmatrix} M_i & 0 \\ 0 & M_j \end{bmatrix} - \begin{bmatrix} M_i & 0 \\ 0 & M_j \end{bmatrix} \begin{bmatrix} Y_{ii} & Y_{ij} \\ Y_{ji} & Y_{jj} \end{bmatrix} = \begin{bmatrix} T^A_{ii} & T^A_{ij} \\ T^A_{ji} & T^A_{jj} \end{bmatrix} - \lambda \begin{bmatrix} T^B_{ii} & T^B_{ij} \\ T^B_{ji} & T^B_{jj} \end{bmatrix},$$

and

$$Z_A[i,j] - \lambda Z_B[i,j] = \begin{bmatrix} Z^A_{ii} & Z^A_{ij} \\ Z^A_{ji} & Z^A_{jj} \end{bmatrix} - \lambda \begin{bmatrix} Z^B_{ii} & Z^B_{ij} \\ Z^B_{ji} & Z^B_{jj} \end{bmatrix}. \tag{5.4}$$

Notably, by interchanging the blocks $M_i = A_i - \lambda B_i$ and $M_j = A_j - \lambda B_j$ in the KCF, we only have to interchange the corresponding blocks in $Z_A - \lambda Z_B$ accordingly. For example, if $Z_A[i,j] - \lambda Z_B[i,j]$ in (5.4) belongs to $\mathrm{nor}(\mathrm{diag}(M_i, M_j))$, then

$$\begin{bmatrix} Z^A_{jj} & Z^A_{ji} \\ Z^A_{ij} & Z^A_{ii} \end{bmatrix} - \lambda \begin{bmatrix} Z^B_{jj} & Z^B_{ji} \\ Z^B_{ij} & Z^B_{ii} \end{bmatrix} \in \mathrm{nor}(\mathrm{diag}(M_j, M_i)).$$

This implies that given two blocks M_i and M_j, it is enough to consider the case $\mathrm{diag}(M_i, M_j)$. In the following we will order the blocks in the KCF so that $Z_A - \lambda Z_B$ is block lower triangular.

Theorem 5.1 *Let $A - \lambda B = \mathrm{diag}(A_1, A_2, \ldots, A_b) - \lambda \mathrm{diag}(B_1, B_2, \ldots, B_b)$ be in KCF with the structure blocks $M_i = A_i - \lambda B_i$ ordered as follows: L_k, $J_k(0)$, $J_k(\gamma)$ (for $\gamma \neq \{0, \infty\}$), N_k, and L_k^T, where the ordering within each block-type is in increasing order of size, except for the L_k^T blocks which are ordered by decreasing order of size.*

For all i and j, let the $(i,j), (j,i)$ and $(i,i), (j,j)$ blocks of $Z_A - \lambda Z_B$ corresponding to diag(M_i, M_j) be built from Table 1 and Table 2, respectively.

Then $Z_A - \lambda Z_B$ gives an orthogonal basis for $\mathrm{nor}(A - \lambda B)$ *with minimum number of parameters.*

The superscripts B, L, U, and D of the matrices in tables 1 and 2 are parts of the matrix definitions in Section 5.2. The superscript T is matrix transpose. All subscripts, e.g. $\alpha \times \beta$, refer to the sizes of the matrices.

Notice that the diagonal blocks (i,i) and (j,j) of $Z_A - \lambda Z_B$ can also be obtained from Table 1 by setting $i = j$. For clarity we also display the expressions for the (i,i) and (j,j) blocks of $Z_A - \lambda Z_B$ corresponding to all kinds of structure blocks M_i in Table 2. Of course, the (j,j) blocks corresponding to M_j are read from Table 2 by substituting α with β.

The proof of Theorem 5.1 consists of three parts:

1. The blocks of $Z_A - \lambda Z_B$ displayed in Table 1 fulfill the conditions in Proposition 5.1, which imply that $Z_A - \lambda Z_B \in \mathrm{nor}(A - \lambda B)$ is orthogonal to an arbitrary $T_A - \lambda T_B \in \tan(A - \lambda B)$.

2. The number of independent parameters in $Z_A - \lambda Z_B$ is equal to the codimension of $\mathrm{orbit}(A - \lambda B)$, which implies that the parameterized normal form has minimum number of parameters.

3. Each block in Table 1 defines an orthogonal basis, i.e., the basis for each parameter p_i is orthogonal to the basis for each other parameter p_j, $i \neq j$.

We start by proving part 3, followed by proving parts 1 and 2 for the 16 different cases $\mathrm{diag}(M_i, M_j)$ corresponding to different combinations of structure blocks in the KCF. In Table 3 we display the codimension for these 16 cases and

Table 1: Blocks in $Z_A - \lambda Z_B \in \mathrm{nor}(A - \lambda B)$, where it for $L_\alpha \oplus L_\beta$, $J_\alpha(0) \oplus J_\beta(0)$, $J_\alpha(\gamma) \oplus J_\beta(\gamma)$, and $N_\alpha \oplus N_\beta$, is assumed that $\alpha \leq \beta$. For $L_\alpha^T \oplus L_\beta^T$, $\alpha \geq \beta$ is assumed. Also $\gamma_1 \neq \gamma_2$ is assumed.

KCF : $M_i \oplus M_j$	Z_{ij}^A	Z_{ij}^B	Z_{ji}^A	Z_{ji}^B
$L_\alpha \oplus L_\beta$	0	0	$S^B_{\beta\times(\alpha+1)}$	$-C_\beta^T S^B_{\beta\times(\alpha+1)}$
$L_\alpha \oplus J_\beta(0)$	0	0	$S^L_{\beta\times(\alpha+1)}$	$-C_\beta^T S^L_{\beta\times(\alpha+1)}$
$L_\alpha \oplus J_\beta(\gamma)$	0	0	$F^D_{\beta\times(\alpha+2)} G^T_{\alpha+1}$	$-F^D_{\beta\times(\alpha+2)} \hat{G}^T_{\alpha+1}$
$L_\alpha \oplus N_\beta$	0	0	$C_\beta^T H^L_{\beta\times(\alpha+1)}$	$-H^L_{\beta\times(\alpha+1)}$
$L_\alpha \oplus L_\beta^T$	0	0	$G_{\beta+1} H_{(\beta+2)\times(\alpha+1)}$	$\hat{G}_{\beta+1} H_{(\beta+2)\times(\alpha+1)}$
$J_\alpha(0) \oplus J_\beta(0)$	$S^L_{\alpha\times\beta}$	$-C_\alpha^T S^L_{\alpha\times\beta}$	$T^L_{\beta\times\alpha}$	$-C_\beta^T T^L_{\beta\times\alpha}$
$J_\alpha(0) \oplus L_\beta^T$	0	0	$H^U_{(\beta+1)\times\alpha}$	$-H^U_{(\beta+1)\times\alpha} C_\alpha^T$
$J_\alpha(\gamma) \oplus J_\beta(\gamma)$	$D^L_{\alpha\times\beta}$	$E^L_{\alpha\times\beta}$	$D^L_{\beta\times\alpha}$	$E^L_{\beta\times\alpha}$
$J_\alpha(\gamma) \oplus L_\beta^T$	0	0	$G_{\beta+1}(\hat{I}_\alpha F^D_{\alpha\times(\beta+2)})^T$	$-\hat{G}_{\beta+1}(\hat{I}_\alpha F^D_{\alpha\times(\beta+2)})^T$
$N_\alpha \oplus N_\beta$	$C_\alpha^T S^L_{\alpha\times\beta}$	$-S^L_{\alpha\times\beta}$	$C_\beta^T T^L_{\beta\times\alpha}$	$-T^L_{\beta\times\alpha}$
$N_\alpha \oplus L_\beta^T$	0	0	$T^L_{(\beta+1)\times\alpha} C_\alpha^T$	$-T^L_{(\beta+1)\times\alpha}$
$L_\alpha^T \oplus L_\beta^T$	0	0	$T^B_{(\beta+1)\times\alpha}$	$-T^B_{(\beta+1)\times\alpha} C_\alpha$
$J_\alpha(0) \oplus J_\beta(\gamma)$	0	0	0	0
$J_\alpha(0) \oplus N_\beta$	0	0	0	0
$J_\alpha(\gamma_1) \oplus J_\beta(\gamma_2)$	0	0	0	0
$J_\alpha(\gamma) \oplus N_\beta$	0	0	0	0

Table 2: The diagonal blocks in $Z_A - \lambda Z_B \in \mathrm{nor}(A - \lambda B)$.

KCF:M_i	Z_{ii}^A	Z_{ii}^B
L_α	0	0
$J_\alpha(0)$	$S_{\alpha\times\alpha}^L$	$-C_\alpha^T S_{\alpha\times\alpha}^L$
$J_\alpha(\gamma)$	$D_{\alpha\times\alpha}^L$	$E_{\alpha\times\alpha}^L$
N_α	$C_\alpha^T S_{\alpha\times\alpha}^L$	$-S_{\alpha\times\alpha}^L$
L_α^T	0	0

the number of parameters in the $(i,i),(i,j),(j,i)$ and (j,j) blocks of $Z_A - \lambda Z_B$. The codimensions are computed from (4.10), which is the minimum number of parameters required to span the corresponding normal space. For the ordering and the sizes of the blocks in $A - \lambda B$ we have made the same assumptions in Table 3 as in Table 1. Notice that the codimension counts for $L_\alpha \oplus L_\beta$ and $L_\alpha^T \oplus L_\beta^T$ are 0 if $\alpha = \beta$. The number of parameters required in each of the $(i,i),(i,j),(j,i)$ and (j,j) blocks of $Z_A - \lambda Z_B$ follows from the proof given below.

Proof of part 3. We show that each matrix pencil block in Table 1 has all its parameters in orthogonal directions. This is trivial for blocks built from the structured Toeplitz and Hankel matrices S^L, S^B, H, H^L, H^U, T^L, or T^B (possibly involving some kind of shift). Remember that the Frobenius inner product can be expressed in terms of the sum of all results from elementwise multiplications as shown in (5.1). For each of these matrices, the elementwise multiplication of the basis for one parameter p_i and the basis for another parameter $p_j, j \neq i$ only results in multiplications where at least one of the two elements is zero. Obviously, these bases are orthogonal. For the matrix pencil blocks built from the F^D matrix, the orthogonality follows from construction, since some of the elements are explicitly chosen so that the Frobenius inner product is zero.

For the proof for the blocks of type $D^L - \lambda E^L$ we define s_q in terms of the d_i and e_i in (5.2) to be

$$s_q = \sum_{i=1}^{q} i|d_i|^2 + \sum_{i=1}^{q-1} i|e_i|^2 - q\overline{\gamma} d_q \overline{e}_q.$$

Independent of s and t, the number s_q is the inner product of the qth basis vector with the rth, where $q < r$.

We show by induction that $s_q = 0$ for $q = 1, 2, \ldots$. Clearly $s_1 = |\gamma|^2 - \gamma\overline{\gamma} = 0$.

Table 3: The number of parameters in the (i,i), (i,j), (j,i), and (j,j) blocks of $Z_A - \lambda Z_B \in \mathrm{nor}(M_i \oplus M_j)$.

KCF: $M_i \oplus M_j$	$\mathrm{cod}(M_i \oplus M_j)$	(i,i)	(i,j)	(j,i)	(j,j)
$L_\alpha \oplus L_\beta$	$\beta-\alpha-1$	0	0	$\beta-\alpha-1$	0
$L_\alpha \oplus J_\beta(0)$	2β	0	0	β	β
$L_\alpha \oplus J_\beta(\gamma)$	2β	0	0	β	β
$L_\alpha \oplus N_\beta$	2β	0	0	β	β
$L_\alpha \oplus L_\beta^T$	$\alpha+\beta+2$	0	0	$\alpha+\beta+2$	0
$J_\alpha(0) \oplus J_\beta(0)$	$\beta+3\alpha$	α	α	α	β
$J_\alpha(0) \oplus L_\beta^T$	2α	α	0	α	0
$J_\alpha(\gamma) \oplus J_\beta(\gamma)$	$\beta+3\alpha$	α	α	α	β
$J_\alpha(\gamma) \oplus L_\beta^T$	2α	α	0	α	0
$N_\alpha \oplus N_\beta$	$\beta+3\alpha$	α	α	α	β
$N_\alpha \oplus L_\beta^T$	2α	α	0	α	0
$L_\alpha^T \oplus L_\beta^T$	$\alpha-\beta-1$	0	0	$\alpha-\beta-1$	0
$J_\alpha(0) \oplus J_\beta(\gamma)$	$\alpha+\beta$	α	0	0	β
$J_\alpha(0) \oplus N_\beta$	$\alpha+\beta$	α	0	0	β
$J_\alpha(\gamma_1) \oplus J_\beta(\gamma_2)$	$\alpha+\beta$	α	0	0	β
$J_\alpha(\gamma) \oplus N_\beta$	$\alpha+\beta$	α	0	0	β

We now show that $s_{q+1} - s_q = 0$ from which the result follows.

$$
\begin{aligned}
&q\overline{\gamma} d_q \overline{e}_q + (q+1)|d_{q+1}|^2 + q|e_q|^2 - (q+1)\overline{\gamma} d_{q+1}\overline{e}_{q+1} = \\
&q\overline{e}_q(\overline{\gamma} d_q + e_q) + (q+1)d_{q+1}(\overline{d}_{q+1} - \overline{\gamma}\,\overline{e}_{q+1}) = \\
&d_{q+1}((q+1)(\overline{d}_{q+1} - \overline{\gamma}\,\overline{e}_{q+1}) - q\overline{e}_q) = \\
&d_{q+1}((q+1)(-\gamma\overline{d}_q - \overline{e}_q + \gamma\overline{d}_q + 2\overline{e}_q - \frac{\overline{e}_q}{q+1}) - q\overline{e}_q) = \\
&d_{q+1}((q+1)\overline{e}_q - \overline{e}_q - q\overline{e}_q) = 0.
\end{aligned}
$$

Since $Z_A - \lambda Z_B$ is built from b^2 mutually independent blocks in Table 1, each associated with c_i parameters, it follows that $Z_A - \lambda Z_B$ is an orthogonal basis for a $c_1 + c_2 + \ldots + c_{b^2}$ dimensional space, with one parameter for each dimension. □

Proof of parts 1 and 2. Now, it remains to show that $Z_A - \lambda Z_B$ is orthogonal to $\tan(A - \lambda B)$ and that the number of parameters in $Z_A - \lambda Z_B$ is equal to $\mathrm{cod}(A - \lambda B)$. Since the number of parameters in orthogonal directions cannot exceed the codimension, it is sufficient to show that we have found them all. The orthogonality between $Z_A - \lambda Z_B$ and $\tan(A - \lambda B)$ is shown by proving that each pair of blocks fulfills the conditions $A_j^H Z_{ji}^A = -B_j^H Z_{ji}^B$ and $Z_{ji}^A A_i^H = -Z_{ji}^B B_i^H$ in Proposition 5.1. In the following we refer to these as the *first* and *second* conditions, respectively.

We carry out the proofs for all 16 cases $M_i \oplus M_j$ in Table 1, starting with blocks where M_i and M_j are of the same kind.

$\mathbf{J}_\alpha(\mathbf{0}) \oplus \mathbf{J}_\beta(\mathbf{0})$: We note that $J_k(0) = C_k - \lambda I_k$. *First* condition for the (j,i) block:

$$A_j^H Z_{ji}^A = C_\beta^T T_{\beta\times\alpha}^L = I_\beta C_\beta^T T_{\beta\times\alpha}^L = -B_j^H Z_{ji}^B.$$

Second condition for the (j,i) block:

$$Z_{ji}^A A_i^H = T_{\beta\times\alpha}^L C_\alpha^T = T_{\beta\times\alpha}^L C_\alpha^T I_\alpha = C_\beta^T T_{\beta\times\alpha}^L I_\alpha = -Z_{ji}^B B_i^H,$$

where we used that $T_{\beta\times\alpha}^L C_\alpha^T = C_\beta^T T_{\beta\times\alpha}^L$, for $\beta \geq \alpha$. Similarly for the (i,j) block:

$$A_i^H Z_{ij}^A = C_\alpha^T S_{\alpha\times\beta}^L = I_\alpha C_\alpha^T S_{\alpha\times\beta}^L = -B_i^H Z_{ij}^B,$$

and

$$Z_{ij}^A A_j^H = S_{\alpha\times\beta}^L C_\beta^T = S_{\alpha\times\beta}^L C_\beta^T I_\beta = C_\alpha^T S_{\alpha\times\beta}^L I_\beta = -Z_{ij}^B B_j^H.$$

Here we used that $S_{\alpha\times\beta}^L C_\beta^T = C_\alpha^T S_{\alpha\times\beta}^L$, for $\beta \geq \alpha$.

Since the (i,i), (i,j), and (j,i) blocks of $Z_A - \lambda Z_B$, have α parameters each and the (j,j) block has β parameters, the total number of parameters in $Z_A - \lambda Z_B$ is equal to $\operatorname{cod}(J_\alpha(0) \oplus J_\beta(0)) = \beta + 3\alpha$.

$\mathbf{N}_\alpha \oplus \mathbf{N}_\beta$: Since there is a symmetry between $J_k(0) = C_k - \lambda I_k$ and $N_k = I_k - \lambda C_k$ and there is a corresponding symmetry between blocks in $Z_A - \lambda Z_B$ for $J_k(0)$ and N_k blocks, the proof for $N_\alpha \oplus N_\beta$ is similar to the case $J_\alpha(0) \oplus J_\beta(0)$.

$\mathbf{J}_\alpha(\gamma) \oplus \mathbf{J}_\beta(\gamma)$: Here the (j,i) block and the (i,j) block are defined similarly (see Table 1), and therefore it is sufficient to prove one of them with no constraints on α and β. We note that $J_k(\gamma) = \gamma I_k + C_k - \lambda I_k$. We show that the first and second conditions hold for $Z_{ji}^A = D_{\beta\times\alpha}[q]$ and $Z_{ji}^B = E_{\beta\times\alpha}[q]$ for $q = 1, \ldots, \min\{\alpha, \beta\}$.
First condition:

$$A_j^H Z_{ji}^A = (\gamma I_\beta + C_\beta)^H D_{\beta\times\alpha}[q] = \overline{\gamma} D_{\beta\times\alpha}[q] + C_\beta^T D_{\beta\times\alpha}[q].$$

Remember that $D_{\beta\times\alpha}[q]$ has all elements zero, except for the q lower left diagonals, where all elements in each diagonal are identical and defined by the element in the first column. For $q = 1$ the proof is trivial. For $q > 1$, $A_j^H Z_{ji}^A$ gives the following matrix. All diagonals starting at position $(u,1)$ for $1 \leq u \leq \beta - q$ are zero. The elements in the diagonal starting at position $(\beta - q + 1, 1)$ are $\overline{\gamma} d_q$ which by definition is equal to $-e_q^*$, which in turn defines the corresponding diagonal in $-E_{\beta\times\alpha}[q]$. The elements in the diagonals starting at positions $(\beta - u + 1, 1)$, where $1 \leq u < q$ are equal to $\overline{\gamma} d_u + d_{u+1}$. Since d_{u+1} is defined as $-\overline{\gamma} d_u - e_u$, the elements in these diagonals are equal to $-e_u$, which defines the elements in the corresponding diagonals in $-E_{\beta\times\alpha}[q]$. Since $-E_{\beta\times\alpha}[q] = -B_j^H Z_{ji}^B$, we have proved the first condition.
Second condition: Since $D_{\beta\times\alpha}[q]$ only has $q \leq \min\{s,t\}$ non-zero diagonals in the lower left corner of the matrix, a shift of rows downwards gives the same

result as a shift of columns leftwards, i.e., $C_\beta^T D_{\beta\times\alpha}[q] = D_{\beta\times\alpha}[q]C_\alpha^T$. Using information from the first part we obtain

$$\begin{aligned} Z_{ji}^A A_i^H &= D_{\beta\times\alpha}[q](\gamma I_\alpha + C_\alpha)^H = \overline{\gamma} D_{\beta\times\alpha}[q] + D_{\beta\times\alpha}[q]C_\alpha^T \\ &= \overline{\gamma} D_{\beta\times\alpha}[q] + C_\beta^T D_{\beta\times\alpha}[q] = A_j^H Z_{ji}^A = -E_{\beta\times\alpha}[q] = -Z_{ji}^B B_i^H, \end{aligned}$$

since B_i is the identity matrix.

Also here, the number of parameters in $Z_{jj}^A - \lambda Z_{jj}^B$ is β and it is α parameters in each of the other three blocks, giving $\beta + 3\alpha$ in total.

Even though the (i,i), (j,i), (i,j), and (j,j) blocks look rather complicated, they reduce for $\gamma = 0$ to the corresponding blocks for $J_\alpha(0) \oplus J_\beta(0)$ in Table 1.

$\mathbf{L}_\alpha \oplus \mathbf{L}_\beta$: Here we use $L_k = \hat{G}_k - \lambda G_k$. *First* condition for the (j,i) block:

$$\begin{aligned} A_j^H Z_{ji}^A &= \hat{G}_\beta^T S_{\beta\times(\alpha+1)}^B = \begin{bmatrix} 0 \\ S_{\beta\times(\alpha+1)}^B \end{bmatrix} = \begin{bmatrix} C_\beta^T S_{\beta\times(\alpha+1)}^B \\ 0 \end{bmatrix} \\ &= G_\beta^T C_\beta^T S_{\beta\times(\alpha+1)}^B = -B_j^H Z_{ji}^B. \end{aligned}$$

Second condition for the (j,i) block:

$$\begin{aligned} Z_{ji}^A A_i^H &= S_{\beta\times(\alpha+1)}^B \hat{G}_\beta^T = \begin{bmatrix} 0 \\ S_{\beta\times(\alpha+1)}^B \end{bmatrix} = \begin{bmatrix} C_\beta^T S_{\beta\times(\alpha+1)}^B \\ 0 \end{bmatrix} \\ &= C_\beta^T S_{\beta\times(\alpha+1)}^B G_\beta^T = -Z_{ji}^B B_i^H. \end{aligned}$$

Since the contribution from $L_\alpha \oplus L_\beta$ to the codimension is $\beta - \alpha - 1$ and the (j,i) block has $\beta - \alpha - 1$ independent parameters we deduce that all other blocks in $Z_A - \lambda Z_B$ are zero.

$\mathbf{L}_\alpha^{\mathbf{T}} \oplus \mathbf{L}_\beta^{\mathbf{T}}$: Since this case is just the transpose of $L_\alpha \oplus L_\beta$ the proof is almost the same, and therefore we omit the technical details here.

So far, we have proved all cases where both blocks are of the same type. Since the diagonal blocks in $Z_A - \lambda Z_B$ always correspond to such cases (see Table 3 for the number of parameters in these blocks), we from now on only have to consider the (i,j) and (j,i) blocks, where $i \neq j$ for the remaining cases.

$\mathbf{L}_\alpha \oplus \mathbf{J}_\beta(\mathbf{0})$: *First* condition for the (j,i) block:

$$A_j^H Z_{ji}^A = C_\beta^T S_{\beta\times(\alpha+1)}^L = I_\beta C_\beta^T S_{\beta\times(\alpha+1)}^L = -B_j^H Z_{ji}^B.$$

Second condition for the (j,i) block:

$$Z_{ji}^A A_i^H = S_{\beta\times(\alpha+1)}^L \hat{G}_\alpha^T = C_\beta^T S_{\beta\times(\alpha+1)}^L G_\alpha^T = -Z_{ji}^B B_i^H.$$

The (i,i) and (j,j) blocks contribute with zero and β parameters, respectively. Since the (j,i) block gives another β parameters, we have found all 2β parameters, and therefore it follows that $Z_{ij}^A = \lambda Z_{ij}^B = 0$.

$\mathbf{L}_\alpha \oplus \mathbf{J}_\beta(\gamma)$: *First* condition for the (j,i) block:

$$A_j^H Z_{ji}^A = (\gamma I_\beta + C_\beta)^H F_{\beta\times(\beta+2)}^D G_{\alpha+1}^T = \overline{\gamma} F_{\beta\times(\beta+2)}^D G_{\alpha+1}^T + C_\beta^T F_{\beta\times(\beta+2)}^D G_{\alpha+1}^T.$$

By inspection we see that the (u,v)-element of this matrix is $\overline{\gamma} f_{u,v}^d + f_{u-1,v}^d$ if $u > 1$ and $\overline{\gamma} f_{u,v}^d$ if $u = 1$ (where $f_{u,v}^d$ denotes the (u,v)-element of F^D). The right hand side of the same condition is

$$-B_j^H Z_{ji}^B = I_\beta F_{\beta\times(\beta+2)}^D \hat{G}_{\alpha+1}^T,$$

which simply is the β leftmost columns of $F_{\beta\times(\beta+2)}^D$. The (u,v)-element of this matrix is then $f_{u,v+1}^d$, which is defined as $\overline{\gamma} f_{u,v}^d + f_{u-1,v}^d$ if $u > 1$ and $\overline{\gamma} f_{u,v}^d$ if $u = 1$.
Second condition for the (j,i) block:

$$\begin{aligned} Z_{ji}^A A_i^H \hat{G}_\alpha^T &= F_{\beta\times(\alpha+2)}^D G_{\alpha+1}^T \hat{G}_\alpha^T = F_{\beta\times(\alpha+2)}^D \begin{bmatrix} 0 \\ I_\alpha \\ 0 \end{bmatrix} \\ &= F_{\beta\times(\alpha+2)}^D \hat{G}_{\alpha+1} G_\alpha^T = -Z_{ji}^B B_i^H. \end{aligned}$$

As in the previous case, the (i,i) and (j,j) blocks contribute with zero and β parameters, respectively. Since the (j,i) block gives the remaining β parameters, the (i,j) block is the zero pencil.

Notably, for $\gamma = 0$, the "monstrous" (j,i) block reduces to the (j,i) block for $L_\alpha \oplus J_\beta(0)$ in Table 1.

$\mathbf{L}_\alpha \oplus \mathbf{N}_\beta$: *First* condition for the (j,i) block:

$$A_j^H Z_{ji}^A = I_\beta C_\beta^T H_{\beta\times(\alpha+1)}^L = C_\beta^T H_{\beta\times(\alpha+1)}^L = -B_j^H Z_{ji}^B.$$

Second condition for the (j,i) block:

$$Z_{ji}^A A_i^H \hat{G}_\alpha^T = C_\beta^T H_{\beta\times(\alpha+1)}^L = \begin{bmatrix} 0 \\ H_{(\beta-1)\times\alpha}^L \end{bmatrix} = H_{\beta\times(\alpha+1)}^L G_\alpha^T = -Z_{ji}^B B_i^H.$$

Also here, the (i,i) and (j,j) blocks contribute with zero and β parameters, respectively. Since the (j,i) block gives the remaining β parameters, the (i,j) block is the zero pencil.

$\mathbf{L}_\alpha \oplus \mathbf{L}_\beta^{\mathbf{T}}$: For this case the (i,i) and (j,j) blocks are zero pencils. *First* condition for the (j,i) block:

$$\begin{aligned} A_j^H Z_{ji}^A &= \hat{G}_\beta G_{\beta+1} H_{(\beta+2)\times(\alpha+1)} = [0 \ I_\beta \ 0] H_{(\beta+2)\times(\alpha+1)} \\ &= G_\beta \hat{G}_{\beta+1} H_{(\beta+2)\times(\alpha+1)} = -B_j^H Z_{ji}^B. \end{aligned}$$

Second condition for the (j,i) block:

$$Z_{ji}^A A_i^H = G_{\beta+1} H_{(\beta+2)\times(\alpha+1)} \hat{G}_\alpha^T,$$

which is a matrix consisting of the $\beta+1$ first rows and α last columns of $H_{(\beta+2)\times(\alpha+1)}$. This matrix is identical to the one given by the $\beta+1$ last rows and α first columns of $H_{(\beta+2)\times(\alpha+1)}$, i.e.,

$$\hat{G}_{\beta+1}H_{(\beta+2)\times(\alpha+1)}G_\alpha^T = -Z_{ji}^B B_i^H .$$

Since this block has all $\alpha+\beta+2$ parameters, it follows that the (i,j) block is the zero pencil.

$\mathbf{J}_\alpha(\mathbf{0}) \oplus \mathbf{L}_\beta^\mathbf{T}$: *First* condition for the (j,i) block:

$$A_j^H Z_{ji}^A = \hat{G}_\beta H_{(\beta+1)\times\alpha}^U ,$$

which simply is the last β rows in $H_{(\beta+1)\times\alpha}^U$. Another way to construct this matrix is to shift the columns in $H_{(\beta+1)\times\alpha}^U$ one column leftwards and pick the β first columns of that matrix, which can be written as

$$G_\beta H_{(\beta+1)\times\alpha}^U C_\alpha^T = -B_j^H Z_{ji}^B .$$

Second condition for the (j,i) block:

$$Z_{ji}^A A_i^H = H_{(\beta+1)\times\alpha}^U C_\alpha^T = H_{(\beta+1)\times\alpha}^U C_\alpha^T I_\alpha = -Z_{ji}^B B_i^H .$$

The (i,i) and (j,j) blocks contribute with α and zero parameters, respectively. Since the (j,i) block gives another α parameters, we conclude that the (i,j) block is the zero pencil.

$\mathbf{J}_\alpha(\gamma) \oplus \mathbf{L}_\beta^\mathbf{T}$: Since the proof for this case is similar to the one for the case $L_\alpha \oplus J_\beta(\gamma)$, we omit the technical details here. It follows that for $\gamma=0$, the (j,i) block reduces to the (j,i) block for $J_\alpha(0) \oplus L_\beta^T$ in Table 1.

$\mathbf{N}_\alpha \oplus \mathbf{L}_\beta^\mathbf{T}$: *First* condition for the (j,i) block:

$$A_j^H Z_{ji}^A = \hat{G}_\beta T_{(\beta+1)\times\alpha}^L C_\alpha^T ,$$

which is the last β rows in $T_{(\beta+1)\times\alpha}^L$ shifted one column leftwards. This matrix is identical to the one given by the β first rows in $T_{(\beta+1)\times\alpha}^L$, which is

$$G_\beta T_{(\beta+1)\times\alpha}^L = -B_j^H Z_{ji}^B .$$

Second condition for the (j,i) block:

$$Z_{ji}^A A_i^H = T_{(\beta+1)\times\alpha}^L C_\alpha^T I_\alpha = T_{(\beta+1)\times\alpha}^L C_\alpha^T = -Z_{ji}^B B_i^H .$$

The (i,i) and (j,j) blocks in $Z_A - \lambda Z_B$ contribute with α and zero parameters, respectively. Since the (j,i) block gives another α parameters, we conclude that the (i,j) block is the zero pencil.

$\mathbf{J}_\alpha(\mathbf{0}) \oplus \mathbf{J}_\beta(\gamma)$, $\mathbf{J}_\alpha(\mathbf{0}) \oplus \mathbf{N}_\beta$, $\mathbf{J}_\alpha(\gamma_1) \oplus \mathbf{J}_\beta(\gamma_2)$, and $\mathbf{J}_\alpha(\gamma) \oplus \mathbf{N}_\beta$: In these four cases the (i,i) and (j,j) blocks contribute with α and β parameters, respectively, and therefore the (j,i) and (i,j) blocks are zero pencils.

Since we have considered all possible cases of M_i and M_j blocks the proof is complete. $\square$

6 Applications and Examples

6.1 Some examples of versal deformations of matrix pencils in KCF

In the following we show three examples of versal deformations of matrix pencils. For the 7×8 pencil $A - \lambda B = L_2 \oplus J_2(0) \oplus J_3(0)$ with codimension 14, the 14-parameter versal deformation $A - \lambda B + Z_A - \lambda Z_B$, where $Z_A - \lambda Z_B \in \mathrm{nor}(A - \lambda B)$ is given by

$$Z_A = \left[\begin{array}{ccc|cc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline p_1 & 0 & 0 & p_6 & 0 & p_{10} & 0 & 0 \\ p_2 & p_1 & 0 & p_7 & p_6 & p_{11} & p_{10} & 0 \\ \hline p_3 & 0 & 0 & 0 & 0 & p_{12} & 0 & 0 \\ p_4 & p_3 & 0 & p_8 & 0 & p_{13} & p_{12} & 0 \\ p_5 & p_4 & p_3 & p_9 & p_8 & p_{14} & p_{13} & p_{12} \end{array}\right],$$

and

$$Z_B = \left[\begin{array}{ccc|cc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -p_1 & 0 & 0 & -p_6 & 0 & -p_{10} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -p_3 & 0 & 0 & 0 & 0 & -p_{12} & 0 & 0 \\ -p_4 & -p_3 & 0 & -p_8 & 0 & -p_{13} & -p_{12} & 0 \end{array}\right].$$

For the 3×4 pencil $A - \lambda B = L_1 \oplus J_2(\gamma)$ with codimension 4, the 4-parameter versal deformation $A - \lambda B + Z_A - \lambda Z_B$, where $Z_A - \lambda Z_B \in \mathrm{nor}(A - \lambda B)$ is given by

$$Z_A = \left[\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ \hline p_1 & p_1\overline{\gamma} & p_3(|\gamma|^2+1) & 0 \\ p_2 - p_1\frac{2\gamma}{|\gamma|^2+1} & p_2\overline{\gamma} - p_1\frac{|\gamma|^2-1}{|\gamma|^2+1} & -p_3\gamma + p_4 & p_3(|\gamma|^2+1) \end{array}\right],$$

and

$$Z_B = \left[\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ \hline -p_1\overline{\gamma} & -p_1\overline{\gamma}^2 & -p_3(|\gamma|^2\overline{\gamma}+\overline{\gamma}) & 0 \\ -p_2\overline{\gamma} + p_1\frac{|\gamma|^2-1}{|\gamma|^2+1} & -p_2\overline{\gamma}^2 - p_1\frac{2\overline{\gamma}}{|\gamma|^2+1} & -p_3 - p_4\overline{\gamma} & -p_3(|\gamma|^2\overline{\gamma}+\overline{\gamma}) \end{array}\right].$$

For the 11×11 pencil $A - \lambda B = L_1 \oplus J_3(0) \oplus N_4 \oplus L_2^T$ with codimension 26, the 26-parameter versal deformation $A - \lambda B + Z_A - \lambda Z_B$, where $Z_A - \lambda Z_B \in$ nor$(A - \lambda B)$ is given by

$$
Z_A = \left[\begin{array}{cc|ccc|cccc|cc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
p_1 & 0 & p_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
p_2 & p_1 & p_{14} & p_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
p_3 & p_2 & p_{15} & p_{14} & p_{13} & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & p_4 & 0 & 0 & 0 & p_{19} & 0 & 0 & 0 & 0 & 0 \\
p_4 & p_5 & 0 & 0 & 0 & p_{20} & p_{19} & 0 & 0 & 0 & 0 \\
p_5 & p_6 & 0 & 0 & 0 & p_{21} & p_{20} & p_{19} & 0 & 0 & 0 \\
\hline
p_8 & p_9 & p_{18} & p_{17} & p_{16} & p_{23} & 0 & 0 & 0 & 0 & 0 \\
p_9 & p_{10} & p_{17} & p_{16} & 0 & p_{24} & p_{23} & 0 & 0 & 0 & 0 \\
p_{10} & p_{11} & p_{16} & 0 & 0 & p_{25} & p_{24} & p_{23} & 0 & 0 & 0
\end{array}\right],
$$

and

$$
Z_B = \left[\begin{array}{cc|ccc|cccc|cc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-p_1 & 0 & -p_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-p_2 & -p_1 & -p_{14} & -p_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & -p_4 & 0 & 0 & 0 & -p_{19} & 0 & 0 & 0 & 0 & 0 \\
-p_4 & -p_5 & 0 & 0 & 0 & -p_{20} & -p_{19} & 0 & 0 & 0 & 0 \\
-p_5 & -p_6 & 0 & 0 & 0 & -p_{21} & -p_{20} & -p_{19} & 0 & 0 & 0 \\
-p_6 & -p_7 & 0 & 0 & 0 & -p_{22} & -p_{21} & -p_{20} & -p_{19} & 0 & 0 \\
\hline
-p_9 & -p_{10} & -p_{17} & -p_{16} & 0 & -p_{24} & -p_{23} & 0 & 0 & 0 & 0 \\
-p_{10} & -p_{11} & -p_{16} & 0 & 0 & -p_{25} & -p_{24} & -p_{23} & 0 & 0 & 0 \\
-p_{11} & -p_{12} & 0 & 0 & 0 & -p_{26} & -p_{25} & -p_{24} & -p_{23} & 0 & 0
\end{array}\right].
$$

6.2 Versal deformations of the set of 2-by-3 matrix pencils

In [14], the algebraic and geometric characteristics of the set of 2-by-3 matrix pencils were examined in full detail, including the complete closure hierarchy. There, all non-zero and finite eigenvalues were considered as unspecified. R_2 was used to denote a 2-by-2 block with non-zero finite eigenvalues, i.e., any of the three structures $J_1(\alpha) \oplus J_1(\beta)$, $J_1(\alpha) \oplus J_1(\alpha)$, and $J_2(\alpha)$, where $\alpha, \beta \neq \{0, \infty\}$. However, in the context of versal deformations all these forms are considered separately and with the eigenvalues specified (known). Consequently, we now have 20 different Kronecker structures to investigate. For example, the versal deformation of $A - \lambda B = L_0 \oplus J_2(\gamma)$, $\gamma \neq \{0, \infty\}$, is found by computing $Z_A - \lambda Z_B =$

$$
\begin{bmatrix}
p_1 + \lambda \overline{\gamma} p_1 & p_3(|\gamma|^2 + 1) + p_3(|\gamma|^2 \overline{\gamma} + \overline{\gamma}) & 0 \\
p_2 - \frac{p_1 \gamma}{|\gamma|^2 + 1} + \lambda(p_2 \overline{\gamma} + \frac{p_1}{|\gamma|^2 + 1}) & -p_3 \gamma + p_4 + \lambda(p_3 + p_4 \overline{\gamma}) & p_3(|\gamma|^2 + 1) + p_3(|\gamma|^2 \overline{\gamma} + \overline{\gamma})
\end{bmatrix}. \tag{6.1}
$$

In Table 4 we show the versal deformations for all different Kronecker structures for this set of matrix pencils. The different structures are displayed in increasing codimension order.

6.2.1 Using GUPTRI in a random walk in tangent and normal directions of non-generic pencils

In order to illustrate how perturbations in the tangent space and in the normal space affect the Kronecker structure computed by a staircase algorithm, we have performed a set of tests on non-generic 2-by-3 matrix pencils. Since the staircase algorithm considers all non-zero finite eigenvalues as unspecified, we have not included these cases in the test.

For the remaining 12 non-generic cases a random perturbation $E_A - \lambda E_B$, with entries e^a_{ij}, e^b_{ij}, has been decomposed into two parts, $T_A - \lambda T_B \in \tan(A - \lambda B)$, and $Z_A - \lambda Z_B \in \mathrm{nor}(A - \lambda B)$, such that

$$E_A = T_A + Z_A \quad \text{and} \quad E_B = T_B + Z_B.$$

We illustrate the decomposition of $E_A - \lambda E_B$ with $A - \lambda B = L_0 \oplus J_2(0)$. From Table 4 we get

$$Z_A = \begin{bmatrix} p_1 & p_3 & 0 \\ p_2 & p_4 & p_3 \end{bmatrix}, \quad Z_B = \begin{bmatrix} 0 & 0 & 0 \\ -p_1 & -p_3 & 0 \end{bmatrix}.$$

Let $T_A - \lambda T_B = (E_A - \lambda E_B) - (Z_A - \lambda Z_B)$. Now, the parameters p_i are determined by computing the component of $E_A - \lambda E_B$ in each of the four orthogonal (but not orthonormal) directions that span the normal space:

$$\begin{aligned}
Z_1 &= \frac{1}{2}\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}\right) \\
Z_2 &= 1\left(\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) \\
Z_3 &= \frac{1}{3}\left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}\right) \\
Z_4 &= 1\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right).
\end{aligned}$$

We conclude that

$$p_1 = \frac{e^a_{11} - e^b_{21}}{2}, \quad p_2 = e^a_{21}, \quad p_3 = \frac{e^a_{12} + e^a_{23} - e^b_{22}}{3}, \quad p_4 = e^a_{22}.$$

It is easily verified that $< T_A - \lambda T_B, Z_A - \lambda Z_B > = 0$.

`GUPTRI` [11, 12] has been used to compute the Kronecker structure of the perturbed pencils $A - \lambda B + \epsilon(E_A - \lambda E_B)$, $A - \lambda B + \epsilon(Z_A - \lambda Z_B)$, and $A -$

Table 4: Versal deformations $A - \lambda B + Z_A - \lambda Z_B$ of 2-by-3 matrix pencils.

KCF	$A - \lambda B$	$Z_A - \lambda Z_B$
L_2	$\begin{bmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
$L_1 \oplus J_1(\gamma)$	$\begin{bmatrix} -\lambda & 1 & 0 \\ 0 & 0 & \gamma - \lambda \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ p_1 + \lambda \overline{\gamma} p_1 & \overline{\gamma} p_1 + \lambda \overline{\gamma}^2 p_1 & p_2 + \lambda \overline{\gamma} p_2 \end{bmatrix}$
$L_1 \oplus J_1(0)$	$\begin{bmatrix} -\lambda & 1 & 0 \\ 0 & 0 & -\lambda \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ p_1 & 0 & p_2 \end{bmatrix}$
$L_1 \oplus N_1$	$\begin{bmatrix} -\lambda & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & \lambda p_1 & \lambda p_2 \end{bmatrix}$
$L_0 \oplus J_1(\gamma_1) \oplus J_1(\gamma_2)$	$\begin{bmatrix} 0 & \gamma_1 - \lambda & 0 \\ 0 & 0 & \gamma_2 - \lambda \end{bmatrix}$	$\begin{bmatrix} p_1 + \lambda \overline{\gamma}_1 p_1 & p_3 + \lambda \overline{\gamma}_1 p_3 & 0 \\ p_2 + \lambda \overline{\gamma}_2 p_2 & 0 & p_4 + \lambda \overline{\gamma}_2 p_4 \end{bmatrix}$
$L_0 \oplus J_2(\gamma)$	$\begin{bmatrix} 0 & \gamma - \lambda & 1 \\ 0 & 0 & \gamma - \lambda \end{bmatrix}$	See (6.1)
$L_0 \oplus 2J_1(\gamma)$	$\begin{bmatrix} 0 & \gamma - \lambda & 0 \\ 0 & 0 & \gamma - \lambda \end{bmatrix}$	$\begin{bmatrix} p_1 + \lambda \overline{\gamma} p_1 & p_3 + \lambda \overline{\gamma} p_3 & p_5 + \lambda \overline{\gamma} p_5 \\ p_2 + \lambda \overline{\gamma} p_2 & p_4 + \lambda \overline{\gamma} p_4 & p_6 + \lambda \overline{\gamma} p_6 \end{bmatrix}$
$L_0 \oplus J_1(0) \oplus J_1(\gamma)$	$\begin{bmatrix} 0 & -\lambda & 0 \\ 0 & 0 & \gamma - \lambda \end{bmatrix}$	$\begin{bmatrix} p_1 & p_3 & 0 \\ p_2 + \lambda \overline{\gamma} p_2 & 0 & p_4 + \lambda \overline{\gamma} p_4 \end{bmatrix}$
$L_0 \oplus J_1(\gamma) \oplus N_1$	$\begin{bmatrix} 0 & \gamma - \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} p_1 + \lambda \overline{\gamma} p_1 & p_3 + \lambda \overline{\gamma} p_3 & 0 \\ \lambda p_2 & 0 & \lambda p_4 \end{bmatrix}$
$L_0 \oplus J_2(0)$	$\begin{bmatrix} 0 & -\lambda & 1 \\ 0 & 0 & -\lambda \end{bmatrix}$	$\begin{bmatrix} p_1 & p_3 & 0 \\ p_2 + \lambda p_1 & p_4 + \lambda p_3 & p_3 \end{bmatrix}$
$L_0 \oplus N_2$	$\begin{bmatrix} 0 & 1 & -\lambda \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \lambda p_1 & \lambda p_3 & 0 \\ p_1 + \lambda p_2 & p_3 + \lambda p_4 & \lambda p_3 \end{bmatrix}$
$L_0 \oplus J_1(0) \oplus N_1$	$\begin{bmatrix} 0 & -\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} p_1 & p_3 & 0 \\ \lambda p_2 & 0 & \lambda p_4 \end{bmatrix}$
$L_0 \oplus L_1 \oplus L_0^T$	$\begin{bmatrix} 0 & -\lambda & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ p_1 + \lambda p_2 & p_3 + \lambda p_4 & p_4 + \lambda p_5 \end{bmatrix}$
$L_0 \oplus 2J_1(0)$	$\begin{bmatrix} 0 & -\lambda & 0 \\ 0 & 0 & -\lambda \end{bmatrix}$	$\begin{bmatrix} p_1 & p_3 & p_5 \\ p_2 & p_4 & p_6 \end{bmatrix}$
$L_0 \oplus 2N_1$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \lambda p_1 & \lambda p_3 & \lambda p_5 \\ \lambda p_2 & \lambda p_4 & \lambda p_6 \end{bmatrix}$
$2L_0 \oplus L_1^T$	$\begin{bmatrix} 0 & 0 & -\lambda \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} p_1 + \lambda p_2 & p_4 + \lambda p_5 & 0 \\ p_2 + \lambda p_3 & p_5 + \lambda p_6 & 0 \end{bmatrix}$
$2L_0 \oplus J_1(\gamma) \oplus L_0^T$	$\begin{bmatrix} 0 & 0 & \gamma - \lambda \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} p_1 + \lambda \overline{\gamma} p_1 & p_4 + \lambda \overline{\gamma} p_4 & p_7 + \lambda \overline{\gamma} p_7 \\ p_2 + \lambda p_3 & p_5 + \lambda p_6 & p_8 + \lambda \overline{\gamma} p_8 \end{bmatrix}$
$2L_0 \oplus J_1(0) \oplus L_0^T$	$\begin{bmatrix} 0 & 0 & -\lambda \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} p_1 & p_4 & p_7 \\ p_2 + \lambda p_3 & p_5 + \lambda p_6 & p_8 \end{bmatrix}$
$2L_0 \oplus N_1 \oplus L_0^T$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \lambda p_1 & \lambda p_4 & \lambda p_7 \\ p_2 + \lambda p_3 & p_5 + \lambda p_6 & \lambda p_8 \end{bmatrix}$
$3L_0 \oplus 2L_0^T$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} p_1 + \lambda p_2 & p_5 + \lambda p_6 & p_9 + \lambda p_{10} \\ p_3 + \lambda p_4 & p_7 + \lambda p_8 & p_{11} + \lambda p_{12} \end{bmatrix}$

$\lambda B + \epsilon(T_A - \lambda T_B)$, for $\epsilon = 10^{-16}, 10^{-15}, \ldots, 10^0$. We investigate how far we can move in the tangent and normal directions before GUPTRI reports the generic Kronecker structure.

The procedure has been repeated for all cases and for 100 random perturbations (E_A, E_B), where $\|(E_A, E_B)\|_F = 1$ and $\|E_A\|_F = \|E_B\|_F$. The entries of (E_A, E_B) are uniformly distributed in $(-0.5, 0.5)$. For each case and for each perturbation $E_A - \lambda E_B$ we record the size of ϵ when GUPTRI reports the generic Kronecker structure. In Table 5 we display the smallest, median, and maximum values of ϵ for the 100 random perturbations.

Table 5: How far we can move in tangent and normal directions before non-generic 2-by-3 matrix pencils turn generic.

		$\epsilon(Z_A - \lambda Z_B)$			$\epsilon(T_A - \lambda T_B)$		
$A - \lambda B$	$\mathrm{cod}(A - \lambda B)$	$\epsilon_{\min}$	$\epsilon_{\mathrm{median}}$	$\epsilon_{\max}$	$\epsilon_{\min}$	$\epsilon_{\mathrm{median}}$	$\epsilon_{\max}$
$L_1 \oplus J_1(0)$	2	10^{-4}	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{-1}
$L_1 \oplus N_1$	2	10^{-4}	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}
$L_0 \oplus J_2(0)$	4	10^{-4}	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}
$L_0 \oplus N_2$	4	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{-1}
$L_0 \oplus J_1(0) \oplus N_1$	4	10^{-4}	10^{-4}	10^{-2}	10^{-2}	10^{-1}	10^{0}
$L_0 \oplus L_1 \oplus L_0^T$	5	10^{-4}	10^{-4}	10^{-2}	10^{-2}	10^{-1}	10^{0}
$L_0 \oplus 2J_1(0)$	6	10^{-16}	10^{-16}	10^{-16}	+	+	+
$L_0 \oplus 2N_1$	6	10^{-16}	10^{-16}	10^{-16}	+	+	+
$2L_0 \oplus L_1^T$	6	10^{-4}	10^{-4}	10^{-2}	+	+	+
$2L_0 \oplus J_1(0) \oplus L_0^T$	8	10^{-5}	10^{-4}	10^{-1}	+	+	+
$2L_0 \oplus N_1 \oplus L_0^T$	8	10^{-4}	10^{-4}	10^{-3}	+	+	+
$3L_0 \oplus 2L_0^T$	12	10^{-16}	10^{-16}	10^{-16}	+	+	+

Entries marked + in Table 5, represent that the generic structure was not found for any size of the perturbations. All these results were for perturbations in $\tan(A - \lambda B)$, and they indicate that for these Kronecker structures there are no or only small curvatures in the orbit at this point (pencil). Here the tangent directions are very close to $\mathrm{orbit}(A - \lambda B)$.

Notably, the results for the perturbations $\epsilon(E_A - \lambda E_B)$ are, except for one case, similar to the results for $\epsilon(Z_A - \lambda Z_B)$. This is natural since the perturbation $E_A - \lambda E_B$ implies a translation both in the tangent space as well as the normal space directions. The structure changes appear more rapidly in the normal space, i.e., for smaller ϵ. Our computational results extend the cone example in Section 1.3 to 2-by-3 matrix pencils.

Why is the smallest perturbation $10^{-16}(Z_A - \lambda Z_B)$ enough to find the generic structure for the three cases $L_0 \oplus 2J_1(0), L_0 \oplus 2N_1$ and $3L_0 \oplus 2L_0^T$? The explanation is connected to the procedure for determining the numerical rank of matrices.

GUPTRI has two input parameters, EPSU and GAP, which are used to make rank decisions in order to determine the Kronecker structure of an input pencil $A - \lambda B$. Inside GUPTRI the absolute tolerances $\texttt{EPSUA} = \|A\|_E \cdot \texttt{EPSU}$ and $\texttt{EPSUB} = \|B\|_E \cdot \texttt{EPSU}$ are used in all rank decisions, where the matrices A and B, respectively, are involved. Suppose the singular values of A are computed in increasing order, i.e., $0 \leq \sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_k \leq \sigma_{k+1} \leq \ldots$; then all singular values $\sigma_k < \texttt{EPSUA}$ are interpreted as zeros. The rank decision is made more robust in practice: if $\sigma_k < \texttt{EPSUA}$ but $\sigma_{k+1} \geq \texttt{EPSUA}$, GUPTRI insists on a gap between the two singular values such that $\sigma_{k+1}/\sigma_k \geq \texttt{GAP}$. If $\sigma_{k+1}/\sigma_k < \texttt{GAP}$, σ_{k+1} is also treated as zero. This process is repeated until an appreciable gap between the zero and non–zero singular values is obtained. In all of our tests we have used $\texttt{EPSU} = 10^{-8}$ and $\texttt{GAP} = 1000.0$.

For the most non-generic case $3L_0 \oplus 2L_0^T$, both the A-part and the B-part are zero matrices giving $\texttt{EPSUA} = \texttt{EPSUB} = 0$, which in turn lead to the decision that a full rank perturbation $E_A - \lambda E_B$ times a very small ϵ is interpreted as a generic pencil. For the other two cases, either the A-part or the B-part is full rank and the other part is a zero matrix, which accordingly is interpreted to have full rank already for the smallest perturbation.

6.2.2 Versal deformations and minimal perturbations for changing a non-generic structure

In the following we illustrate how versal deformations are useful in the understanding of the relations between the different structures, by looking at requirements on perturbations to (A, B) for changing the Kronecker structure. Assume that we have the following matrix pencil with the Kronecker structure $L_1 \oplus J_1(0)$:

$$A - \lambda B = \begin{bmatrix} -\epsilon_1\lambda & \epsilon_2 & 0 \\ 0 & 0 & -\epsilon_3\lambda \end{bmatrix} \quad \text{and} \quad Z_A - \lambda Z_B = \begin{bmatrix} 0 & 0 & 0 \\ p_1 & 0 & p_2 \end{bmatrix}. \tag{6.2}$$

It was shown in [14] that $L_1 \oplus J_1(0)$ with codimension 2 is in the closure of $\mathrm{orbit}(L_1 \oplus J_1(\gamma))$ ($\gamma \neq \{0, \infty\}$ but otherwise unspecified) with codimension 1, which in turn is in the closure of $\mathrm{orbit}(L_2)$ (the generic KCF) with codimension 0. Notice in Table 4, since γ is assumed specified, $L_1 \oplus J_1(\gamma)$ has two parameters (and codimension $= 2$). In the discussion that follows we assume that γ is finite, non-zero but unspecified.

We will now, for this example, illustrate how perturbations in the normal space directions can be used to find more generic Kronecker structures (going upwards in the Kronecker structure hierarchy), and how we can perturb the elements in $A - \lambda B$ to find less generic matrix pencils. Since the space spanned by $Z_A - \lambda Z_B$ is the normal space, we must always first hit a more generic pencil when we move infinitesimally in normal space directions.

The KCF remains unchanged as long as $p_1 = p_2 = 0$, but for $p_1 = 0$ and $p_2 \neq 0$, the KCF is changed into $L_1 \oplus J_1(\gamma)$ (with $\gamma = p_2$). That is, by adding a component in a normal space direction, we find a more generic pencil in the closure hierarchy. Notably, the size of the required perturbation is equal to the smallest size of an eigenvalue to be interpreted as non-zero. By choosing p_1 non-zero (and p_2 arbitrary), the resulting pencil will be generic with the KCF L_2.

To find a less generic structure, we may proceed in one of the following ways:

1. Find a less generic structure in the closure of orbit$(L_1 \oplus J_1(0))$.

2. Go upwards in the closure hierarchy, to a more generic structure and then look in that orbit's closure for a less generic structure.

We know from the investigation in [14] that all structures with higher codimension than $A - \lambda B = L_1 \oplus J_1(0)$ include an L_0 block in their Kronecker structures, which in turn imply that A and B must have a common column nullspace of at least dimension 1. Therefore, the smallest perturbation that turns $L_1 \oplus J_1(0)$ less generic is the smallest perturbation that reduces the rank of

$$\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} 0 & \epsilon_2 & 0 \\ 0 & 0 & 0 \\ \epsilon_1 & 0 & 0 \\ 0 & 0 & \epsilon_3 \end{bmatrix}.$$

The size of the smallest rank-reducing perturbation is equal to the smallest of the singular values ϵ_1, ϵ_2, and ϵ_3. By just deleting one ϵ_i, the corresponding perturbed pencil is a less generic pencil within the closure of orbit$(L_1 \oplus J_1(0))$. These three cases correspond to approach 1 above. We summarize these perturbations and the perturbations in the normal space in Table 6. Notice that approach 2 will always require a perturbation larger than $\min\{\epsilon_i\}$.

Which of the non-generic structures displayed in Table 6 is obtained by the smallest perturbation to $L_1 \oplus J_1(0)$? Mathematically, it is easy to see that the perturbations in the normal space always can be made smaller than a rank-reducing perturbation ϵ_i, since p_1 and p_2 are parameters that can be chosen arbitrary small, e.g. smaller than $\min\{\epsilon_i\}$.

However, in finite precision arithmetic, it is not clear that the smallest perturbation required to find another structure is in the normal direction. This can be illustrated by using `GUPTRI` to compute the Kronecker structures for $A - \lambda B$ as in (6.2) and perturbed as in Table 6. For `EPSU` $= 10^{-8}$, $\epsilon_2 = 1$ and $\epsilon_1 = \epsilon_3 = 10^{-10}$, `GUPTRI` uses different tolerances `EPSUA` $= 10^{-8}$ and `EPSUB` $= 10^{-18}$ for making rank decisions in A and B, respectively. It follows that for p_1 and p_2 of order 10^{-6}, `GUPTRI` still computes the Kronecker structure $L_1 \oplus J_1(0)$. However, if $p_1 = p_2 = 0$ and the B-part of the pencil is perturbed by ϵ_1 or ϵ_3, `GUPTRI` computes the less generic structures, just as shown in Table 6.

Table 6: Perturbing $A - \lambda B$ (defined in 6.2) yields the pencil $\tilde{A} - \lambda\tilde{B}$ with more or less generic structures. The codimension of the original orbit is 2.

$\|(\Delta A, \Delta B)\|_F$	$\tilde{A} - \lambda\tilde{B}$	KCF	$\mathrm{cod}(\tilde{A} - \lambda\tilde{B})$
p_1	$\begin{bmatrix} -\epsilon_1\lambda & \epsilon_2 & 0 \\ p_1 & 0 & -\epsilon_3\lambda \end{bmatrix}$	L_2	0
p_2	$\begin{bmatrix} -\epsilon_1\lambda & \epsilon_2 & 0 \\ 0 & 0 & p_2 - \epsilon_3\lambda \end{bmatrix}$	$L_1 \oplus J_1(p_2)$	1 (2)
ϵ_1	$\begin{bmatrix} -0 & \epsilon_2 & 0 \\ 0 & 0 & -\epsilon_3\lambda \end{bmatrix}$	$L_0 \oplus J_1(0) \oplus N_1$	4
ϵ_3	$\begin{bmatrix} -\epsilon_1\lambda & \epsilon_2 & 0 \\ 0 & 0 & -0 \end{bmatrix}$	$L_0 \oplus L_1 \oplus L_0^T$	5
ϵ_2	$\begin{bmatrix} -\epsilon_1\lambda & 0 & 0 \\ 0 & 0 & -\epsilon_3\lambda \end{bmatrix}$	$L_0 \oplus 2J_1(0)$	6

7 Conclusions

In this paper, we have obtained not only versal deformations for deformations of Kronecker canonical forms, but more importantly for our purposes, metrical information for the perturbation theory of matrix pencils relevant to the Kronecker canonical form. In Part II of this paper, we will explore the stratification theory of matrix pencils with the goal of making algorithmic use of the lattice of orbits under the closure relationship.

Acknowledgements

We would like to thank Jim Demmel for conveying the message that the geometry of matrix and matrix pencil space influences perturbation theory and numerical algorithms. We further thank him for many helpful discussions. In addition, the first author would like to thank the second two authors for their kind hospitality and support at Umeå University where this work and fruitful collaboration began.

References

[1] V. I. Arnold. On Matrices Depending on Parameters. *Russian Math. Surveys*, 26:29–43, 1971.

[2] T. Beelen and P. Van Dooren. An improved algorithm for the computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 105:9–65, 1988.

[3] J.M. Berg and H.G. Kwatny. A canonical parameterization of the Kronecker form of a matrix pencil. Preprint, 1994. To appear in *Automatica*.

[4] T. Bröcker and L. Lander. *Differential Germs and Catastrophes.* Cambridge University Press, 1975.

[5] J.W. Bruce and P.J. Giblin. *Curves and Singularities.* Cambridge University Press, 1991.

[6] S.-N. Chow, C. Li, and D. Wang. *Normal Forms and Bifurcation of Planar Vector Fields.* Cambridge University Press, 1994.

[7] J. Demmel. On structured singular values. In *Proceedings of the 27th Conference on Decision and Control*, Austin, TX, Dec 1988. IEEE.

[8] J. Demmel and A. Edelman. The Dimension of Matrices (Matrix Pencils) with Given Jordan (Kronecker) Canonical Forms. Report LBL-31839, Mathematics Department, Lawrence Berkeley Laboratories, University of California, Berkeley, CA 94720, 1992. To appear in *Lin. Alg. Appl.*

[9] J. Demmel and B. Kågström. Computing stable eigendecompositions of matrix pencils. *Lin. Alg. Appl.*, 88/89:139–186, April 1987.

[10] J. Demmel and B. Kågström. Accurate solutions of ill-posed problems in control theory. *SIAM J. Mat. Anal. Appl.*, 9(1):126–145, January 1988.

[11] J. Demmel and B. Kågström. The Generalized Schur Decomposition of an Arbitrary Pencil $A - \lambda B$: Robust Software with Error Bounds and Applications. Part I: Theory and Algorithms. *ACM Trans. Math. Software*, Vol.19(No. 2):160–174, June 1993.

[12] J. Demmel and B. Kågström. The Generalized Schur Decomposition of an Arbitrary Pencil $A - \lambda B$: Robust Software with Error Bounds and Applications. Part II: Software and Applications. *ACM Trans. Math. Software*, Vol.19(No. 2):175–201, June 1993.

[13] A. Edelman and H. Murakami. Polynomial Roots from Companion Matrix Eigenvalues. *Mathematics of Computations, to appear*, 1995.

[14] E. Elmroth and B. Kågström. The Set of 2-by-3 Matrix Pencils – Kronecker Structures and Their Transitions Under Perturbations. Report UMINF-93.22, Department of Computing Science, Umeå University, S-901 87 Umeå, Sweden, November, 1993. Revised October, 1994. To appear in *SIAM J. Matrix Anal. Appl.*

[15] F. Gantmacher. *The Theory of Matrices, Vol. I and II (transl.).* Chelsea, New York, 1959.

[16] G. Golub and C. Van Loan. *Matrix Computations.* Second Edition. Johns Hopkins University Press, Baltimore, MD, 1989.

[17] M. Golubitsky and V. Guillemin. *Stable Mappings and their Singularities.* Springer Verlag, 1973.

[18] U. Helmke and D. Hinrichsen. Canonical forms and orbit spaces of linear systems. *IMA J. Mathematical Control & Information*, pages 167–184, 1986.

[19] D. Hinrichsen and D. Prätzel-Wolters. A wild quiver in linear systems theory. *Linear Algebra Applications*, pages 143–175, 1987.

[20] D. Hinrichsen and D. Prätzel-Wolters. A Jordan canonical form for reachable linear systems. *Linear Algebra Applications*, pages 489–524, 1989.

[21] B. Kågström. RGSVD - an algorithm for computing the Kronecker canonical form and reducing subspaces of singular matrix pencils $A - \lambda B$. *SIAM J. Sci. Stat. Comp.*, 7(1):185–211, 1986.

[22] V. B. Khazanov and V. Kublanovskaya. Spectral problems for matrix pencils. Methods and algorithms. I. *Sov. J. Numer. Anal. Math. Modelling*, 3:337–371, 1988.

[23] V. Kublanovskaya. AB-algorithm and its modifications for the spectral problem of linear pencils of matrices. *Num. Math.*, 43:329–342, 1984.

[24] Y.-C. Lu. *Singularity Theory and an Introduction to Catastrophe Theory.* Springer Verlag, 1976.

[25] D. Prätzel-Wolters. Canonical forms for linear systems. *Linear Algebra Applications*, pages 437–473, 1983.

[26] M. Shub and S. Smale. Complexity of Bezout's theorem II: volumes and probabilities. In F. Eyssette and A. Galligo, editors, *Computational Algebraic Geometry*, Progress in Mathematics Vol. 109, pages 267–285. Birkhauser, 1993.

[27] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory.* Academic Press, New York, 1990.

[28] L. Stolovitch. On the computation of a versal family of matrices. *Numerical Algorithms*, 4:25–56, 1993.

[29] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 27:103–141, 1979.

[30] P. Van Dooren. The generalized eigenstructure problem in linear system theory. *IEEE Trans. Autom. Contr.*, AC-26(1):111–129, 1981.

[31] P. Van Dooren. Reducing subspaces: Definitions, properties and algorithms. In B. Kågström and A. Ruhe, editors, *Matrix Pencils*, pages 58–73. Springer-Verlag, Berlin, 1983. Lecture Notes in Mathematics, vol. 973, Proceedings, Pite Havsbad, 1982.

[32] W. Waterhouse. The codimension of singular matrix pairs. *Lin. Alg. Appl.*, 57:227–245, 1984.

[33] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford Science Publications, 1965.

[34] J. H. Wilkinson. Linear differential equations and Kronecker's canonical form. In C. de Boor and G. Golub, editors, *Recent Advances in Numerical Analysis*, pages 231–265. Academic Press, 1978.

Paper V

On the Stratification of the Kronecker Canonical Form

Erik Elmroth*

Department of Computing Science, Umeå University
S-901 87 Umeå, Sweden.
E-mail: elmroth@cs.ume.se

Abstract

The understanding of which Kronecker structures that are close to a given structure is revealed by the Kronecker structure hierarchy, i.e., the stratification of the Kronecker canonical form. For a given matrix pencil $A - \lambda B$, the Kronecker structure hierarchy shows all structures that are within the closure of orbit$(A - \lambda B)$, and each structure, whose orbit's closure contains $A - \lambda B$. In order to gain new insight in the problem of stratification, we give new interpretations of important results by Pokrzywa, for determining closure relations among orbits of Kronecker structures. This is partly done by generalizing classical theorems by Gantmacher. The results are used to derive an algorithm for computation of the complete Kronecker structure hierarchy, or the Kronecker structure hierarchy above or below a given structure. The algorithm is presented in terms of the rank-decisions required in a staircase algorithm, in order to compute the Kronecker structure hierarchy.

Keywords: Kronecker canonical form, Kronecker structure hierarchy, stratification, staircase algorithm, perturbation theory.

*In part supported by the Swedish National Board of Industrial and Technical Development under grant 89-02578P.

1 Introduction

Any algorithm for computation of the Kronecker Canonical form of an m-by-n matrix pencil $A - \lambda B$, computes the exact Kronecker structure of a nearby pencil $A' - \lambda B'$. The distance $\delta \equiv \|(A - A', B - B')\|_E$ is an upper bound on the distance to the closest $(A + \delta A, B + \delta B)$ with the KCF of (A', B'). These algorithms are supposed to compute the most non-generic KCF within distance δ from $A - \lambda B$, and they are said to fail if there is another more non-generic pencil $\hat{A} - \lambda \hat{B}$ within distance δ from the original pencil.

The computation of a KCF can be seen as moving pencils from point to point or manifold to manifold in the $2mn$-dimensional matrix pencil space. During the computation and in the interpretation of the result, it is of great interest to know which structures that are close to a given KCF.

Given $A-\lambda B$, we know mathematically that an arbitrary small perturbation outwards from orbit$(A - \lambda B)$ (the manifold of pencils strictly equivalent to $A - \lambda B$) results in a more generic pencil above $A - \lambda B$ in the closure hierarchy. That is, a structure that is more generic than $A - \lambda B$ and whose orbit's closure includes $A - \lambda B$. By travelling downwards in the closure hierarchy, we find less generic structures, that are in the closure orbit$(A - \lambda B)$.

In order to make the best decisions during the computation of the KCF it is important to understand how the manifolds of different structures relate to each other. Recent contributions to this understanding can be found in [2, 6, 7]. In this contribution we show for the general case how to generate the complete Kronecker structure hierarchy and how to generate the Kronecker structure hierarchy above or below a given KCF.

In Section 2, we briefly review the Kronecker canonical form and the concepts of orbits and their codimension. Section 3 contains a summary of important results by Pokrzywa [11] on the stratification of orbits, i.e., the characterization of closure relations of matrix pencils and necessary conditions on consecutive pencils. In Section 4, we give new interpretations of these results, partly by extending some classical theorems by Gantmacher [8]. The necessary conditions on consecutive pencils are in Section 5 interpreted in terms of rank-decisions of a staircase algorithm for computation of Kronecker structures. In Section 6, our results are put together in an algorithm for computing the complete Kronecker structure hierarchy, or the hierarchy above or below a given structure.

The results in this contribution should be seen as preliminary results for the second part of [6].

2 The Kronecker Canonical Form

Any m-by-n pencil $A - \lambda B$ can by equivalence transformations be transformed into the Kronecker canonical form

$$P^{-1}(A-\lambda B)Q = \mathrm{diag}(L_{\epsilon_1}, \ldots, L_{\epsilon_p}, J_{j_1}(\mu_1), \ldots, J_{j_k}(\mu_k), N_{i_1}, \ldots, N_{i_k}, L^T_{\eta_1}, \ldots, L^T_{\eta_q}),$$

where $J_j(\mu)$ corresponds to a j-by-j Jordan block for the zero or non-zero finite eigenvalue μ and N_j corresponds to a j-by-j Jordan block for the infinite eigenvalue:

$$J_j(\mu) \equiv \begin{bmatrix} \mu-\lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \mu-\lambda \end{bmatrix} \quad \text{and} \quad N_j \equiv \begin{bmatrix} 1 & -\lambda & & \\ & \ddots & \ddots & \\ & & \ddots & -\lambda \\ & & & 1 \end{bmatrix}.$$

The L_j and L_j^T blocks are *singular blocks of right* (column) and *left* (row) *indices of grade* j. These blocks are of size j-by-$(j+1)$ and $(j+1)$-by-j, respectively, and have the form

$$L_j \equiv \begin{bmatrix} -\lambda & 1 & & \\ & \ddots & \ddots & \\ & & -\lambda & 1 \end{bmatrix} \quad \text{and} \quad L_j^T \equiv \begin{bmatrix} -\lambda & & \\ 1 & \ddots & \\ & \ddots & -\lambda \\ & & 1 \end{bmatrix}.$$

The singular blocks have no eigenvalues and there exists a right singular (column) vector that for each λ zeroes out the L_j block identically:

$$\begin{bmatrix} -\lambda & 1 & & \\ & \ddots & \ddots & \\ & & -\lambda & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \lambda \\ \lambda^2 \\ \vdots \\ \lambda^j \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Similarly, there exists a left singular (row) vector that zeroes out the L_j^T block identically:

$$\begin{bmatrix} 1 & \lambda & \lambda^2 & \cdots & \lambda^j \end{bmatrix} \begin{bmatrix} -\lambda & & \\ 1 & \ddots & \\ & \ddots & -\lambda \\ & & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}.$$

If $A - \lambda B$ is regular, the L_j and L_j^T blocks are not present in the Kronecker canonical from.

However, most applications do not require $A - \lambda B$ to be transformed into Kronecker canonical form. Most often it is enough to transfer $A - \lambda B$ to a *generalized Schur form* or similar [1, 3, 4, 9, 10, 12, 14], which reveals the complete Kronecker structure.

If $A - \lambda B$ is m-by-n, where $m \neq n$, then for almost all A and B it will have the same KCF, depending only on m and n (the *generic case*). The generic

Kronecker structure for $A - \lambda B$ with $d = n - m > 0$ is

$$\mathrm{diag}(L_\alpha, \ldots, L_\alpha, L_{\alpha+1}, \ldots, L_{\alpha+1}), \tag{2.1}$$

where $\alpha = \lfloor m/d \rfloor$, the total number of blocks is d, and the number of $L_{\alpha+1}$ blocks is $m \bmod d$ (which is 0 when d divides m) [12, 2]. The same statement holds for $d = m - n > 0$ if we replace $L_\alpha, L_{\alpha+1}$ in (2.1) by $L_\alpha^T, L_{\alpha+1}^T$. Square pencils are generically regular, i.e., $\det(A - \lambda B) = 0$ if and only if λ is an eigenvalue. The generic singular pencils of size n-by-n have the Kronecker structures [13]:

$$\mathrm{diag}(L_j, L_{n-j-1}^T), \quad j = 0, \ldots, n-1.$$

We define an *orbit* to be the set of *strictly equivalent* pencils in $2mn$-dimensional space:

$$\mathrm{orbit}(A - \lambda B) = \{P^{-1}(A - \lambda B)Q : \det(P)\det(Q) \neq 0\}.$$

That is, an orbit defines a manifold of pencils with identical Kronecker structures. The dimension of $\mathrm{orbit}(A - \lambda B)$ is equal to the dimension of the tangent space, $\tan(A - \lambda B)$, to the orbit of $A - \lambda B$. The tangent space is defined as

$$f(X, Y) = X(A - \lambda B) - (A - \lambda B)Y, \tag{2.2}$$

where X is an $m \times m$ matrix and Y is an $n \times n$ matrix [2]. Since (2.2) maps a space of dimension $m^2 + n^2$ linearly into a space of dimension $2mn$, the dimension of the tangent space is $m^2 + n^2 - d$, where d is the number of (linearly) independent solutions of $f(X, Y) = 0$.

The codimension is the dimension of the space complementary to the tangent space, i.e.,

$$\mathrm{cod}(A - \lambda B) = 2mn - \dim(\tan(A - \lambda B)) = d - (m - n)^2.$$

As shown in [6], the codimension of $A - \lambda B$ can be computed as the number of zero singular values of

$$T \equiv \left[\begin{array}{cc} A^T \otimes I_m & -I_n \otimes A \\ B^T \otimes I_m & -I_n \otimes B \end{array} \right].$$

We may also define the normal space, $\mathrm{nor}(A - \lambda B)$, as the space perpendicular to $\tan(A - \lambda B)$. Then $\mathrm{cod}(A - \lambda B)$ is the dimension of $\mathrm{nor}(A - \lambda B)$.

Since the codimension of an orbit depends only on its Kronecker structure, $\mathrm{cod}(A - \lambda B)$ can also be computed by summing the contributions to the codimension from different blocks in the KCF [2].

3 Stratification of Orbits

Given two $m \times n$ matrix pencils $\mathcal{P}_1 = A_1 - \lambda B_1$ and $\mathcal{P}_2 = A_2 - \lambda B_2$ we are interested to know when the closure of orbit$(\mathcal{P}_1)$ includes the closure of orbit$(\mathcal{P}_2)$, i.e., $\overline{\text{orbit}}(\mathcal{P}_1) \supset \overline{\text{orbit}}(\mathcal{P}_2)$. As in the matrix case we define $\mathcal{P}_1$ and $\mathcal{P}_2$ to be *consecutive* if $\overline{\text{orbit}}(\mathcal{P}_1) \supset \overline{\text{orbit}}(\mathcal{P}_2)$ and there is no pencil $\mathcal{P}$ such that $\overline{\text{orbit}}(\mathcal{P}_1) \supset \overline{\text{orbit}}(\mathcal{P}) \supset \overline{\text{orbit}}(\mathcal{P}_2)$.

Since the KCF may include both Jordan blocks associated with finite as well as infinite eigenvalues and Kronecker blocks corresponding to left and right minimal indices, the task to get a complete characterization of all possible $m \times n$ Kronecker forms is much more intricate than for the matrix case, where we only have to consider Jordan blocks.

3.1 Characterization of the closure of orbit$(\mathbf{A} - \lambda\mathbf{B})$

An important contribution to the understanding of the stratification for the Kronecker canonical form was presented by Pokrzywa [11] in 1986. We start by reviewing some of his results (using our notation).

The following theorem gives a characterization of $\overline{\text{orbit}}(\mathcal{P}_1) \supset \overline{\text{orbit}}(\mathcal{P}_2)$ in terms of their Kronecker structures. For a given pencil $\mathcal{P} = A - \lambda B$ let $r_k(\mathcal{P})$ and $l_k(\mathcal{P})$ be the number of L_k (right singular) and L_k^T (left singular) blocks, respectively, and let $r(\mathcal{P})$ be the total number of right singular blocks in the KCF of $\mathcal{P}$. Moreover, let $j_k(\mathcal{P}, \mu)$ be the number of $k \times k$ Jordan blocks associated with the eigenvalue μ of $\mathcal{P}$.

Theorem 3.1 *[11] A pencil $\mathcal{P}_2$ is in the closure of* orbit$(\mathcal{P}_1)$ *if and only if the following inequalities hold:*

$$\sum_k (i-k)_+ \cdot r_k(\mathcal{P}_1) \;\le\; \sum_k (i-k)_+ \cdot r_k(\mathcal{P}_2), \tag{3.1}$$

$$\sum_k (i-k)_+ \cdot l_k(\mathcal{P}_1) \;\le\; \sum_k (i-k)_+ \cdot l_k(\mathcal{P}_2), \tag{3.2}$$

$$i \cdot r(\mathcal{P}_1) + \sum_k \min(i,k) \cdot j_k(\mathcal{P}_1, \mu) \;\le\; i \cdot r(\mathcal{P}_2) + \sum_k \min(i,k) \cdot j_k(\mathcal{P}_2, \mu), \tag{3.3}$$

for all $i = 1, 2, \ldots$ and $\mu \in \overline{\mathbf{C}}$, where $\overline{\mathbf{C}}$ is the complex plane including the point at infinity and $(x)_+ = \max(0, x)$.

Theorem 3.1 gives us necessary and sufficient conditions to decide whether one pencil is in the closure of another. However, to get a complete understanding of the closure hierarchy we also need to be able to decide whether two pencils are consecutive or not, or given a pencil $\mathcal{P}_1$ we want to know how to construct all possible consecutive pencils (above and below in the closure hierarchy).

An instrument for this purpose is the following characterization of two consecutive pencils [11]. If $\overline{\text{orbit}}(\mathcal{P}_1) \supset \overline{\text{orbit}}(\mathcal{P}_2)$ and $\mathcal{P}_1$ and $\mathcal{P}_2$ are consecutive pencils, then $\mathcal{P}_1 = P_1^{(1)} \oplus P_1^{(2)}$ and $\mathcal{P}_2 = P_2^{(1)} \oplus P_2^{(2)}$, where $P_1^{(1)}$ and $P_2^{(1)}$ are equivalent, and $P_1^{(2)}$ and $P_2^{(2)}$ are equivalent to one of the ordered pairs of pencils $(\overline{\text{orbit}}(P_1^{(2)}) \supset \overline{\text{orbit}}(P_2^{(2)}))$ of the following lemma.

Lemma 3.1 *[11] The following closure relations hold:*

(i) $\overline{\text{orbit}}(L_j \oplus L_k) \supset \overline{\text{orbit}}(L_{j-1} \oplus L_{k+1}),\ 1 \le j \le k.$

(ii) $\overline{\text{orbit}}(L_j^T \oplus L_k^T) \supset \overline{\text{orbit}}(L_{j-1}^T \oplus L_{k+1}^T),\ 1 \le j \le k.$

(iii) $\overline{\text{orbit}}(L_{j+1} \oplus J_k(\mu)) \supset \overline{\text{orbit}}(L_j \oplus J_{k+1}(\mu)),\ j,k = 0,1,2,\dots$ and $\mu \in \overline{\mathbf{C}}$.

(iv) $\overline{\text{orbit}}(L_{j+1}^T \oplus J_k(\mu)) \supset \overline{\text{orbit}}(L_j^T \oplus J_{k+1}(\mu)),\ j,k = 0,1,2,\dots$ and $\mu \in \overline{\mathbf{C}}$.

(v) $\overline{\text{orbit}}(J_{j-1}(\mu)) \oplus J_{k+1}(\mu)) \supset \overline{\text{orbit}}(J_j(\mu)) \oplus J_k(\mu)),\ 1 \le j \le k$ and $\mu \in \overline{\mathbf{C}}$.

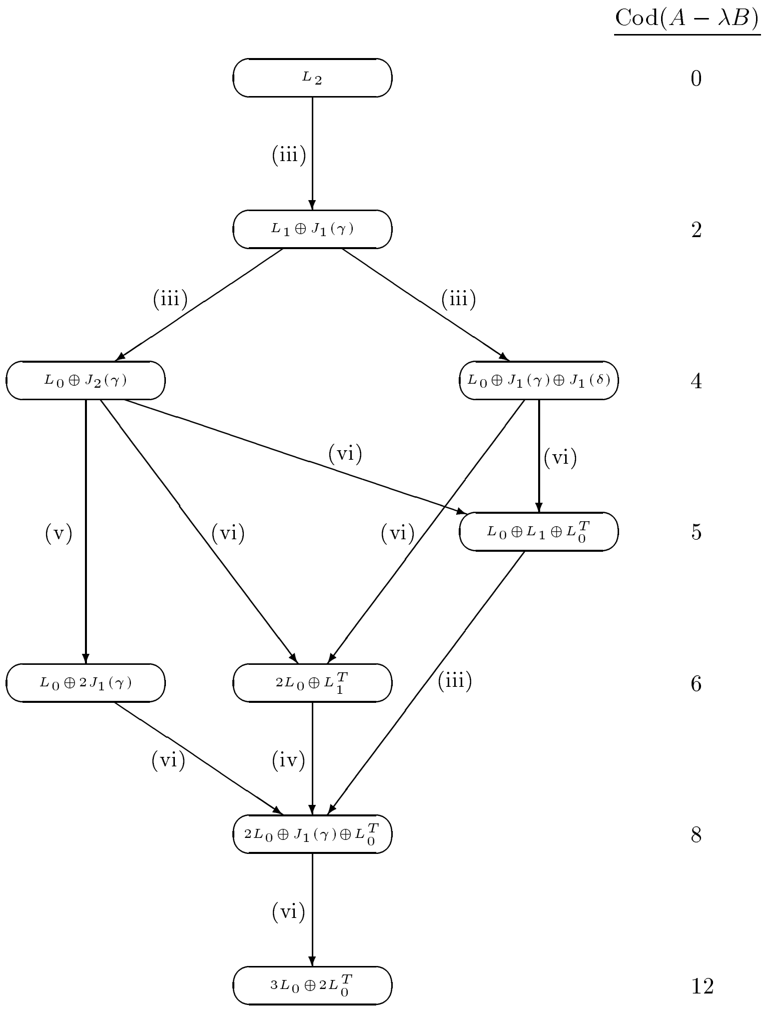
(vi) $\overline{\text{orbit}}(\bigoplus_{i=1}^t J_{k_i}(\lambda_i)) \supset \overline{\text{orbit}}(L_p \oplus L_q^T)$, if $\lambda_i \ne \lambda_j$ for $i \ne j$, $\lambda_i \in \overline{\mathbf{C}}$ and $p + q + 1 = \sum_{i=1}^t k_i$.

Given a pencil $\mathcal{P}_1$ and its KCF we can use Lemma 3.1 to find a pencil $\mathcal{P}_2$ in $\overline{\text{orbit}}(\mathcal{P}_1)$ (i.e., $\mathcal{P}_2$ is below in the hierarchy) such that $\mathcal{P}_1$ and $\mathcal{P}_2$ fulfill necessary conditions for two consecutive pencils. Similarly, using Lemma 3.1 we can find another pencil $\mathcal{P}_2$ such that $\mathcal{P}_1$ is in $\overline{\text{orbit}}(\mathcal{P}_2)$ (i.e., $\mathcal{P}_2$ is above in the hierarchy). In both cases $\mathcal{P}_1$ and $\mathcal{P}_2$ fulfill necessary but not sufficient conditions for two consecutive pencils. Notice that it is allowed to have $J_0(\mu)$ blocks in the left hand side of items (iii), (iv), and (v) [11]. For example, in item (iii) this means that $\overline{\text{orbit}}(L_{j+1}) \supset \overline{\text{orbit}}(L_j \oplus J_1(\mu))$.

Recently, Elmroth and Kågström derived the closure hierarchy (or stratification) of the set of 2×3 pencils [7]. By considering zero, non-zero and infinite eigenvalues separately (which is motivated from an algorithmic point of view) there are 20 structurally different Kronecker structures. From a topological point of view it is natural to only consider different eigenvalues without any special attention to zero and infinite eigenvalues, resulting in only 9 structurally different Kronecker structures. The closure hierarchy graph corresponding to these is shown in Figure 1.

Following [7] we display closure graphs such that orbits (nodes) with the same codimension are displayed on the same horizontal level. The generic case (L_2) is at the highest level and the most non-generic pencil ($3L_0 \oplus 2L_0^T$ which is the 2×3 zero pencil) is at the lowest level. Moreover, along each arc in Figure 1 we display the item in Lemma 3.1 for the necessary condition for consecutive pencils that is fulfilled. From this closure hierarchy it is possible to find examples where the necessary conditions of Lemma 3.1 are not sufficient conditions for consecutive pencils. For example, $\mathcal{P}_1 = L_1 \oplus J_1(\gamma)$ and $\mathcal{P}_2 = L_0 \oplus 2J_1(\gamma)$ fulfill the necessary condition (iii), $\mathcal{P}_1 = L_1 \oplus J_1(\gamma)$ and $\mathcal{P}_2 = L_0 \oplus L_1 \oplus L_0^T$ fulfill the necessary condition (vi), as $\mathcal{P}_1 = L_0 \oplus J_1(\gamma) \oplus J_1(\delta)$ and $\mathcal{P}_2 = 2L_0 \oplus J_1(\gamma) \oplus L_0^T$ do, while in all three cases there exists a pencil $\mathcal{P}$ such that $\overline{\text{orbit}}(\mathcal{P}_1) \supset \overline{\text{orbit}}(\mathcal{P}) \supset \overline{\text{orbit}}(\mathcal{P}_2)$.

Figure 1: Closure hierarchy graph for the set of 2-by-3 pencils.



4 Characterizing Closure Relations Using Gantmacher Matrices

Let the Gantmacher matrix $R[A,B,i]$ of size $(i+2)m \times (i+1)n$ be defined by

$$R[A,B,i] = \begin{bmatrix} A & 0 & \cdots & 0 \\ B & A & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & B & A \\ 0 & \cdots & 0 & B \end{bmatrix},$$

where A and B are $m \times n$ matrices. When it is clear from context we use the abbreviated notation $R[i]$, or $R[\mathcal{P}, i]$ where $\mathcal{P} = A - \lambda B$, for $R[A,B,i]$.

Gantmacher showed that if $R[i-1]$ has full column rank but $R[i]$ does not, then there exists an L_i block in the Kronecker structure of $A - \lambda B$ [8]. Here we extend this result in theorems 4.1–4.2 and we use these extensions in Theorem 4.3 to characterize closure relations in terms of Gantmacher matrices. The objective is to gain more insight in the problem of stratification of orbits of Kronecker structures.

Theorem 4.1 *If $R[A,B,i-1]$ has full column rank and $R[A,B,i]$ has a k-dimensional column nullspace, then there exist k L_i blocks in the KCF of $A - \lambda B$.*

Proof. Since $R[i-1]$ has full column rank but $R[i]$ does not (since it has a k-dimensional column nullspace), we know from Gantmacher's results that $A - \lambda B$ is equivalent to the pencil

$$\begin{bmatrix} L_i & 0 \\ 0 & A' - \lambda B' \end{bmatrix}. \tag{4.1}$$

By rewriting $R[i]$ using this equivalent pencil and the notation $L_i = L_i^A - \lambda L_i^B$ we have,

$$R[A,B,i] = \left[\begin{array}{cc|cc|cc|cc} L_i^A & 0 & & & & & & \\ 0 & A' & & & & & & \\ \hline L_i^B & 0 & L_i^A & & & & & \\ 0 & B' & 0 & A' & & & & \\ \hline & & \ddots & & \ddots & & & \\ & & & \ddots & & \ddots & & \\ \hline & & & & L_i^B & 0 & L_i^A & 0 \\ & & & & 0 & B' & 0 & A' \\ \hline & & & & & & L_i^B & 0 \\ & & & & & & 0 & B' \end{array}\right], \tag{4.2}$$

where the empty blocks denote zeros. By using

$$U = \left[\begin{array}{c|c|c|c}
I_i \quad 0 & & & \\ \hline
 & I_i \quad 0 & & \\ \hline
 & & \ddots & \\ \hline
 & & & I_i \quad 0 \\ \hline
0 \quad I_{m-i} & & & \\ \hline
 & 0 \quad I_{m-i} & & \\ \hline
 & & \ddots & \\ \hline
 & & & 0 \quad I_{m-i}
\end{array}\right], \tag{4.3}$$

and

$$V = \left[\begin{array}{c|c|c|c|c|c|c|c}
\begin{matrix} I_{i+1} \\ 0 \end{matrix} & & & & \begin{matrix} 0 \\ I_{n-i-1} \end{matrix} & & & \\ \hline
 & \begin{matrix} I_{i+1} \\ 0 \end{matrix} & & & & \begin{matrix} 0 \\ I_{n-i-1} \end{matrix} & & \\ \hline
 & & \ddots & & & & \ddots & \\ \hline
 & & & \begin{matrix} I_{i+1} \\ 0 \end{matrix} & & & & \begin{matrix} 0 \\ I_{n-i-1} \end{matrix}
\end{array}\right], \tag{4.4}$$

for row and column permutations, respectively, we have

$$U\,R[A,B,i]\,V = \begin{bmatrix} R[L_i^A, L_i^B, i] & 0 \\ 0 & R[A', B', i] \end{bmatrix}. \tag{4.5}$$

It is easily shown that $R[L_i^A, L_i^B, i]$ has full row rank and since it is of dimension $(i^2 + 2i) \times (i^2 + 2i + 1)$, the matrix $R[L_i^A, L_i^B, i]$ has a 1-dimensional column nullspace. Since $R[A, B, i]$ has a k-dimensional column nullspace, the matrix $R[A', B', i]$ must have a $(k-1)$-dimensional column nullspace. If $k - 1 \geq 1$, then there is at least one L_i block in the KCF of $A' - \lambda B'$. By repeating the procedure above, we are in total able to extract k L_i blocks from $A - \lambda B$. $\Box$

Theorem 4.2 *If $R[A, B, i]$ has a k-dimensional column nullspace given by k L_i blocks in the KCF of $A - \lambda B$, then these k L_i blocks induce a $k(p - i + 1)$-dimensional column nullspace in $R[A, B, p]$ for $p \geq i$.*

Proof. Let $R[A, B, i]$ have a k-dimensional column nullspace given by k L_i blocks. The total nullspace may of course be larger due to existence of other right singular blocks. By using similar reorganizations as in (4.1)–(4.5) of the matrix $R[A, B, i]$, we can obtain a reorganized matrix with k $R[L_i^A, L_i^B, i]$ matrices in its block structure. Similarly, a reorganized $R[A, B, p]$ has k $R[L_i^A, L_i^B, p]$ matrices in its block structure. Since each $R[L_i^A, L_i^B, p]$ matrix has full row rank and dimensions $(p+2)i \times (p+1)(i+1)$, each $R[L_i^A, L_i^B, p]$ has a $(p-i+1)$-dimensional column nullspace for $p \geq i$. It follows that the k matrices $R[L_i^A, L_i^B, i]$ in total give a $k(p - i + 1)$-dimensional column nullspace in $R[A, B, p]$ for $p \geq i$. $\Box$

From Theorem 3.1 we have that the right singular structure of a pencil $\mathcal{P} = A - \lambda B$ is characterized by the expression

$$\sum_k (i-k)_+ \cdot r_k(\mathcal{P}), \text{ for } i = 1, 2, \ldots, \tag{4.6}$$

where $r_k(\mathcal{P})$ is the number of L_k blocks in the KCF of $\mathcal{P}$. For a given value i the sum (4.6) is equal to

$$i \cdot r_0 + (i-1) \cdot r_1 + \ldots + 1 \cdot r_{i-1}.$$

From theorems 4.1–4.2 we know that $(i-j) \cdot r_j$ is the dimension of the part of the column nullspace of $R[A, B, i-1]$ that is induced by r_j L_j blocks. It follows that we get an explicit expression for $\mathrm{n}_c(R[i])$, the dimension of the column nullspace of $R[i]$.

Corollary 4.1 *Let $r_k(\mathcal{P})$ be the number of right singular blocks in $\mathcal{P} = A - \lambda B$. Then*

$$\sum_k (i-k)_+ \cdot r_k(\mathcal{P}) \equiv \mathrm{n}_c(R[i-1]), \tag{4.7}$$

for $i = 1, 2, \ldots$.

To identify the existence of L_i^T blocks in the KCF, Gantmacher stated a dual form of his theorem. It says that if the $(i+1)m \times (i+2)n$ matrix

$$L[A, B, i] = \begin{bmatrix} A & B & 0 & \cdots & 0 \\ 0 & A & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A & B \end{bmatrix},$$

does not have full row rank but $L[A, B, i-1]$ does, then there exists an L_i^T block in the KCF of $A - \lambda B$.

As before, we also use the abbreviated notations $L[i]$ or $L[\mathcal{P}, i]$ for $L[A, B, i]$. It is straightforward to formulate dual versions of theorems 4.1 and 4.2, resulting in similar explicit expressions for $\mathrm{n}_r(L[i])$, the dimension of the row nullspace of $L[i]$.

Corollary 4.2 *Let $l_k(\mathcal{P})$ be the number of left singular blocks in $\mathcal{P} = A - \lambda B$. Then*

$$\sum_k (i-k)_+ \cdot l_k(\mathcal{P}) \equiv \mathrm{n}_r(L[i-1]), \tag{4.8}$$

for $i = 1, 2, \ldots$.

We have now showed how to interpret the first two inequalities in Theorem 3.1, using Gantmacher matrices. Our interpretation of the third inequality is partly based on some final remarks by Pokrzywa [11]. He notes that if we consider the n-by-n pencil $A - \mu I$, then

$$\sum_k \min(i,k)\cdot j_k(\mathcal{P},\mu) = \mathrm{n}_c((A-\mu I)^i),$$

which we recognize as the sum of the i first elements in the Weyr characteristics corresponding to the eigenvalue μ [5]. (Since A is square, we also have $\mathrm{n}_c((A - \mu I)^i) = \mathrm{n}_r((A-\mu I)^i)$). For general $A - \lambda B$, let w_j for the eigenvalue μ denote the number of $J_k(\mu)$ blocks for $k \geq j$. Then

$$\sum_k \min(i,k)\cdot j_k(\mathcal{P},\mu) = \sum_{j=1}^{i} w_j.$$

We are now able to rewrite Theorem 3.1 in terms of Weyr characteristics and Gantmacher matrices.

Theorem 4.3 *A pencil $\mathcal{P}_2$ is in the closure of* $\mathrm{orbit}(\mathcal{P}_1)$ *if and only if the following inequalities hold:*

$$\mathrm{n}_c(R[\mathcal{P}_1,i]) \;\leq\; \mathrm{n}_c(R[\mathcal{P}_2,i]), \tag{4.9}$$

$$\mathrm{n}_r(L[\mathcal{P}_1,i]) \;\leq\; \mathrm{n}_r(L[\mathcal{P}_2,i]), \tag{4.10}$$

$$i\cdot r(\mathcal{P}_1) + \sum_{j=1}^{i} w_j^{(1)} \;\leq\; i\cdot r(\mathcal{P}_2) + \sum_{j=1}^{i} w_j^{(2)}, \tag{4.11}$$

for all $i = 0,1,2,\ldots$ and $w_j^{(1)}$ and $w_j^{(2)}$, $j = 1,2,\ldots$, denote the Weyr characteristics corresponding to the eigenvalue $\mu \in \overline{\mathbf{C}}$, for $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively.

Note that we here start form $i = 0$ in order to obtain $R[i]$ and $L[i]$ in (4.10) and (4.10) instead of $R[i-1]$ and $L[i-1]$, as in (4.7) and (4.8). The inequality (4.11) is not affected by this change, since $i = 0$ makes both the left hand side and the right hand side equal to zero.

One immediate observation is that since $w_i = 0$ for i large enough, $r(\mathcal{P}_1) \leq r(\mathcal{P}_2)$ must hold in order to satisfy the third inequality for all i. Indeed, by examining the first two inequalities using Theorem 4.2, we see that they require $\mathcal{P}_2$ to have at least as many L_k and L_k^T blocks as $\mathcal{P}_1$.

5 Characterizing Consecutive Pencils Using the Staircase Algorithm

Using the staircase algorithm (in infinite precision arithmetic) it is possible to compute the (exact) Kronecker structure of a given $m \times n$ pencil. One phase of the algorithm extracts, for example, the Jordan structure of the zero eigenvalue and the right singular structure of $A - \lambda B$ using a finite sequence of orthogonal (unitary) equivalence transformations. In step k $(= 0, 1, \ldots)$ of the first phase, the GUPTRI algorithm [3, 4] (one variant of the staircase algorithm) determines m_k = dimension of the column nullspace of $A^{(k)}$ and $m_k - s_k$ = dimension of the common column nullspace of $A^{(k)}$ and $B^{(k)}$. Here, $A^{(0)} = A$ and $B^{(0)} = B$ and $(A^{(k)}, B^{(k)})$ for $k \geq 1$ correspond to the deflated matrix pair obtained after the equivalence transformation in step $k - 1$. The structure indices display the Kronecker structure as follows:

- $m_k - s_k$ = number of L_k blocks.
- $s_k - m_{k+1}$ = number of $J_{k+1}(0)$ blocks.

Applying the same algorithm to $B - \mu A$ results in the Jordan structure of the infinite eigenvalue and the right singular structure. The Jordan structure (and structure indices) associated with a finite but non-zero eigenvalue is obtained by applying the algorithm to a shifted pencil. One way to find the left singular structure is to apply the same algorithm to the transposed pencil. Another way is to directly determine the sizes of the corresponding row nullspaces as done in the GUPTRI algorithm, resulting in the Jordan structure of the infinite eigenvalue and the left singular structure. Then $m_k - s_k$ is the number of L_k^T blocks and $s_k - m_{k+1}$ equals the number of $N_{k+1} \equiv J_{k+1}(\infty)$ blocks.

In Corollary 5.1, we formulate Lemma 3.1 in terms of the structure indices computed by the staircase algorithm. Given a pencil $\mathcal{P}_1$, we show prerequisites on its structure indices for each item in Lemma 3.1 and how they must change in order to characterize a pencil $\mathcal{P}_2$ such that $\overline{\text{orbit}}(\mathcal{P}_1) \supset \overline{\text{orbit}}(\mathcal{P}_2)$ and $\mathcal{P}_1$ and $\mathcal{P}_2$ fulfill necessary conditions for two consecutive pencils. We use an arrow $(\rightarrow)$ to show how one block in the KCF is transferred to another.

Corollary 5.1 *Items* (i)–(vi) *show prerequisites on the m_i and s_i indices of a pencil $\mathcal{P}_1$ and the changes required in these indices to find a pencil $\mathcal{P}_2$, such that $\overline{\text{orbit}}(\mathcal{P}_1) \supset \overline{\text{orbit}}(\mathcal{P}_2)$ and $\mathcal{P}_1$ and $\mathcal{P}_2$ fulfill necessary conditions for two consecutive pencils.*

(i) *For $1 \leq j \leq k$ there exist at least one L_j and L_k in the KCF of $\mathcal{P}_1$ if either $m_j > s_j$, $m_k > s_k$ and $j < k$, or $m_j > s_j + 1$ and $j = k$. $\mathcal{P}_2$ is characterized by the changes $m_j := m_j - 1$, $s_{j-1} := s_{j-1} - 1$ $(L_j \rightarrow L_{j-1})$, $m_{k+1} := m_{k+1} + 1$, and $s_k := s_k + 1$ $(L_k \rightarrow L_{k+1})$.*

(ii) *Similar to item* (i) *but with the m_k and s_k indices replaced by the indices representing the corresponding left singular structure.*

(iii) *For $j, k \geq 0$ there exist at least one L_{j+1} and $J_k(\mu)$ in the KCF of $\mathcal{P}_1$ if $m_{j+1} > s_{j+1}$ and $s_{k-1} > m_k$. The last relation is not applicable if $k = 0$. $\mathcal{P}_2$ is characterized by the changes $m_{j+1} := m_{j+1} - 1$, $s_j := s_j - 1$ $(L_{j+1} \rightarrow L_j)$, $m_k := m_k + 1$, and $s_k := s_k + 1$ $(J_k(\mu) \rightarrow J_{k+1}(\mu))$.*

(iv) *Similar to item* (iii) *but with the* m_k *and* s_k *indices replaced by the indices representing the corresponding left singular structure and Jordan structure.*

(v) *For* $1 \le j \le k$ *there exist at least one* $J_{j-1}(\mu)$ *and* $J_{k+1}(\mu)$ *in the KCF of* $\mathcal{P}_1$ *if* $s_{j-2} > m_{j-1}$, $s_k > m_{k+1}$. *The first relation is not applicable if* $j = 1$. $\mathcal{P}_2$ *is characterized by the changes* $m_{j-1} := m_{j-1} + 1$, $s_{j-1} := s_{j-1} + 1$ $(J_{j-1}(\mu) \to J_j(\mu))$, $m_k := m_k - 1$, *and* $s_k := s_k - 1$ $(J_{k+1}(\mu) \to J_k(\mu))$.

(vi) *There exist* t *Jordan blocks* $J_{k_i}(\mu_i), (k_i \ge 1)$ *where each Jordan block corresponds to a different eigenvalue* μ_i. *The structure indices for each eigenvalue must fulfill* $s_{k_i - 1} > m_{k_i}$. $\mathcal{P}_2$ *is characterized by the changes* $s_{\delta-1} := s_{\delta-1} - 1$, $m_\delta := m_\delta - 1$, *for* $\delta = 1, \ldots, k_i$ *for the indices corresponding to each of the eigenvalues. Simultaneously, the indices for* L_j *blocks must be changed as* $m_\alpha := m_\alpha + 1$ *for* $\alpha = 0, \ldots, p$, $s_\alpha := s_\alpha + 1$ *for* $\alpha = 0, \ldots, p-1$ *and the indices for* L_j^T *blocks must be changed as* $m_\beta := m_\beta + 1$ *for* $\beta = 0, \ldots, q$, $s_\beta := s_\beta + 1$ *for* $\beta = 0, \ldots, q-1$, *where* $p + q + 1 = \sum_{i=1}^{t} k_i$. *These operations replace the regular part corresponding to the* t *Jordan blocks with a generic square singular pencil* $L_p \oplus L_q^T$.

Similarly, given $\mathcal{P}_1$ it is possible to characterize a pencil $\mathcal{P}_2$ such that $\overline{\mathrm{orbit}}(\mathcal{P}_2) \supset \overline{\mathrm{orbit}}(\mathcal{P}_1)$ and $\mathcal{P}_1$ and $\mathcal{P}_2$ fulfill necessary conditions for two consecutive pencils. Of course, this will impose different prerequisites on $\mathcal{P}_1$'s structure indices. The details are omitted here.

Applying the GUPTRI algorithm in finite precision arithmetic means that all rank decisions for computing the structure indices are made with respect to a user supplied tolerance which reflects the relative accuracy of the data [3, 4]. Assuming a fixed accuracy of the input data it is possible to increase or decrease the tolerance for rank decisions such that a more non-generic pencil or a more generic pencil, respectively, is computed. Alternatively, given a Kronecker structure computed by the staircase algorithm we can impose a more non-generic Kronecker structure by applying any of the applicable structure index changes in Corollary 5.1. A modified GUPTRI algorithm can deliver an upper bound on the size of the distance from the pencil $\mathcal{P}_1$ we started with to the pencil $\mathcal{P}_2$ we imposed such that $\mathcal{P}_2$ is in $\overline{\mathrm{orbit}}(\mathcal{P}_1)$. The other way around, we can start with a pencil $\mathcal{P}_1$ and construct a more generic pencil $\mathcal{P}_2$ by adding perturbations (whose sizes depend on the rank decision tolerance) such that $\mathcal{P}_1$ is in $\overline{\mathrm{orbit}}(\mathcal{P}_2)$ and the necessary conditions for consecutive pencils are fulfilled.

In infinite precision arithmetic we can always go upwards in the closure hierarchy by adding arbitrary small perturbations. This is normally not the case for going downwards in the hierarchy. See [7] for computable normwise bounds of the smallest perturbations for going downwards (or upwards) in the closure hierarchy of the set of 2×3 pencils.

6 An Algorithm for Generating Closure Hierarchies

In the following we give an algorithm for deriving the complete closure hierarchy below a given m-by-n pencil $\mathcal{P}$. By choosing $\mathcal{P}$ as the generic pencil, the algorithm derives the complete closure hierarchy for m-by-n pencils. We call $\mathcal{P}_i$ a successor of $\mathcal{P}$ if and only if $\mathcal{P}$ and $\mathcal{P}_i$ are consecutive and $\overline{\text{orbit}}(\mathcal{P}) \supset \overline{\text{orbit}}(\mathcal{P}_i)$. Algorithm 6.1 generates the closure hierarchy by finding all successors of each KCF.

Algorithm 6.1 *Let $\mathcal{P}$ denote the starting KCF and let Q denote a queue of generated structures, whose successors are not yet determined. Initially Q is empty.*

1. *Generate all structures, $\mathcal{P}_1, \mathcal{P}_2, \ldots$, that are candidates for being successors of $\mathcal{P}$, by applying all appropriate items (i) - (vi) of Corollary 5.1, on all appropriate combination of blocks in $\mathcal{P}$. That is, change the m_i and s_j according to each item where the prerequisites on the indices are fulfilled.*

2. *Compare all pairs $(\mathcal{P}_i, \mathcal{P}_j)$ of candidate successors of $\mathcal{P}$, using Theorem 4.3 (or Theorem 3.1) to detect if $\overline{\text{orbit}}(\mathcal{P}_i) \supset \overline{\text{orbit}}(\mathcal{P}_j)$. If so, the $\mathcal{P}_j$ and $\mathcal{P}$ are not consecutive, and $\mathcal{P}_j$ is discarded from being a successor of $\mathcal{P}$. All other $\mathcal{P}_i$s are registered as successors of $\mathcal{P}$.*

3. *All remaining $\mathcal{P}_i$s that are not already in Q are inserted in order of increasing codimension.*

4. *If Q is not empty, remove the first KCF in Q and denote it $\mathcal{P}$. Repeat from 1.*

We illustrate the execution of Algorithm 6.1 by showing how it generates the complete closure hierarchy for the set of 2-by-3 matrix pencils in Table 1. The algorithm starts with $\mathcal{P} = L_2$, i.e., the generic 2-by-3 Kronecker structure. For each $\mathcal{P}$, the table shows all candidate successors as generated from step 1 in the algorithm. Candidate successors that are discarded in step 2 are overcrossed in Table 1. The graph displaying the complete closure hierarchy in Figure 1 is obtained by drawing arrows from each structure $\mathcal{P}$ in the table, to all its successors (but not to the overcrossed ones).

To obtain an algorithm for computing the structure hierarchy above a given pencil we have to do the following. In step 1 in Algorithm 6.1, the use of Corollary 5.1 should be replaced by use of a corresponding corollary, that given a pencil $\mathcal{P}_1$ gives the prerequisites and the required changes in the m_i and s_j indices for finding a pencil $\mathcal{P}_2$ such that $\overline{\text{orbit}}(\mathcal{P}_2) \supset \overline{\text{orbit}}(\mathcal{P}_1)$ and $\mathcal{P}_1$ and $\mathcal{P}_2$ fulfill necessary conditions for consecutive pencils, according to Lemma 3.1. In step 2, $\mathcal{P}_i$ should be discarded from being successor of $\mathcal{P}$ if $\overline{\text{orbit}}(\mathcal{P}_i) \supset \overline{\text{orbit}}(\mathcal{P}_j)$.

Table 1: Successors in the closure hierarchy for the set of 2-by-3 matrix pencils, in order as generated by Algorithm 6.1.

$\mathcal{P}$	Successors
L_2:	$L_1 \oplus J_1(\gamma)$
$L_1 \oplus J_1(\gamma)$:	$L_0 \oplus J_1(\gamma) \oplus J_1(\delta)$, $\cancel{L_0 \oplus 2J_1(\gamma)}$, $L_0 \oplus J_2(\gamma)$, $\cancel{L_0 \oplus L_1 \oplus L_1^T}$
$L_0 \oplus J_1(\gamma) \oplus J_1(\delta)$:	$\cancel{2L_0 \oplus J_1(\gamma) \oplus L_0^T}$, $2L_0 \oplus L_1^T$, $L_0 \oplus L_1 \oplus L_1^T$
$L_0 \oplus J_2(\gamma)$:	$L_0 \oplus 2J_1(\gamma)$, $2L_0 \oplus L_1^T$, $L_0 \oplus L_1 \oplus L_1^T$
$L_0 \oplus L_1 \oplus L_1^T$:	$2L_0 \oplus J_1(\gamma) \oplus L_0^T$
$2L_0 \oplus L_1^T$:	$2L_0 \oplus J_1(\gamma) \oplus L_0^T$
$L_0 \oplus 2J_1(\gamma)$:	$2L_0 \oplus J_1(\gamma) \oplus L_0^T$
$2L_0 \oplus J_1(\gamma) \oplus L_0^T$:	$3L_0 \oplus L_0^T$
$3L_0 \oplus L_0^T$:	

In step 3, Q should be sorted in decreasing order of codimension. Finally, to derive the complete closure hierarchy, we should now start with the most non-generic pencil, i.e., the zero pencil.

We now from [7] that that number of different Kronecker structures grows rapidly for increasing m and n. In Algorithm 6.1, this will not only cause a lot of more work for generation of all true successors, but there will also be a large growth of the number of generated candidate successors that in step 2 are discarded. Therefore, the amount of redundant work in the algorithm will increase with increasing m and n.

7 Conclusions

We have given new interpretations of important results by Pokrzywa [11], for determining closure relations among orbits of Kronecker structures. This has partly been done by generalizing classical theorems by Gantmacher [8]. These results have been used to derive an algorithm for computation of the complete Kronecker structure hierarchy, or the complete hierarchy above or below a given Kronecker structure. The algorithm is presented in terms of the rank-decisions required in a staircase algorithm, in order to compute the Kronecker structure hierarchy.

From a mathematical point of view, we know that the Kronecker structures that are closest to a pencil $\mathcal{P}$ with a given structure are the ones that are above it in the hierarchy. We know that these structures are the ones that are found by adding perturbations in the normal space of orbit($\mathcal{P}$) in the point given by $\mathcal{P}$. How to actually compute these perturbations, or the *versal deformation* of a KCF, is shown in [6]. For our purposes, it is not the versal deformation itself that is most important. We are more interested in metrical information for the perturbation theory relevant to the Kronecker canonical form. The results in this contribution should be seen in this context and as preliminary results for the second part of [6].

Acknowledgements

I would like to thank Bo Kågström for helpful discussions and constructive comments during the work and the preparation of this paper. I would also like to thank Alan Edelman for helpful discussions and for encouraging the study of the stratification of the Kronecker canonical form.

References

[1] T. Beelen and P. Van Dooren. An improved algorithm for the computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 105:9–65, 1988.

[2] J. Demmel and A. Edelman. The Dimension of Matrices (Matrix Pencils) with Given Jordan (Kronecker) Canonical Forms. Report LBL-31839, Mathematics Department, Lawrence Berkeley Laboratories, University of California, Berkeley, CA 94720, 1992. To appear in *Lin. Alg. Appl.*

[3] J. Demmel and B. Kågström. The Generalized Schur Decomposition of an Arbitrary Pencil $A - \lambda B$: Robust Software with Error Bounds and Applications. Part I: Theory and Algorithms. *ACM Trans. Math. Software*, Vol.19(No. 2):160–174, June 1993.

[4] J. Demmel and B. Kågström. The Generalized Schur Decomposition of an Arbitrary Pencil $A - \lambda B$: Robust Software with Error Bounds and Applications. Part II: Software and Applications. *ACM Trans. Math. Software*, Vol.19(No. 2):175–201, June 1993.

[5] C.C. Mac Duffee. *The Theory of Matrices.* Chelsea Publishing Company, New York, 1956.

[6] A. Edelman, E. Elmroth, and B. Kågström. A Geometric Approach To Perturbation Theory of Matrices and Matrix Pencils. Part I: Versal Deformations. Report UMINF-95.09, Department of Computing Science, Umeå University, S-901 87 Umeå, Sweden, March, 1995.

[7] E. Elmroth and B. Kågström. The Set of 2-by-3 Matrix Pencils – Kronecker Structures and Their Transitions Under Perturbations. Report UMINF-93.22, Department of Computing Science, Umeå University, S-901 87 Umeå,

Sweden, November, 1993. Revised October, 1994. To appear in *SIAM J. Matrix Anal. Appl.*

[8] F. Gantmacher. *The Theory of Matrices, Vol. I and II (transl.).* Chelsea, New York, 1959.

[9] B. Kågström. RGSVD - an algorithm for computing the Kronecker canonical form and reducing subspaces of singular matrix pencils $A - \lambda B$. *SIAM J. Sci. Stat. Comp.*, 7(1):185–211, 1986.

[10] V. B. Khazanov and V. Kublanovskaya. Spectral problems for matrix pencils. Methods and algorithms. I. *Sov. J. Numer. Anal. Math. Modelling*, 3:337–371, 1988.

[11] A. Pokrzywa. On perturbations and the equivalence orbit of a matrix pencil. *Lin. Alg. Appl.*, 82:99–121, 1986.

[12] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 27:103–141, 1979.

[13] W. Waterhouse. The codimension of singular matrix pairs. *Lin. Alg. Appl.*, 57:227–245, 1984.

[14] J. H. Wilkinson. Linear differential equations and Kronecker's canonical form. In C. de Boor and G. Golub, editors, *Recent Advances in Numerical Analysis*, pages 231–265. Academic Press, 1978.