NICONET Report 97-3

SLICOT — A Subroutine Library in Systems and Control Theory [1]

Peter Benner[2], Volker Mehrmann[3], Vasile Sima[4], Sabine Van Huffel[5], Andras Varga[6]

June 1997

[1] This paper presents research results of the European Community BRITE-EURAM III Thematic Networks Programme NICONET and is distributed by the Working Group on Software WGS. *WGS secretariat*: Mrs. Ida Tassens, ESAT - Katholieke Universiteit Leuven, K. Mercierlaan 94, 3001-Leuven-Heverlee, BELGIUM. This report is also available by anonymous ftp from *wgs.esat.kuleuven.ac.be* in the directory *pub/WGS/REPORTS/nic97-3.ps.Z*
[2] Zentrum für Technomathematik, Fachbereich 3 – Mathematik und Informatik, Universität Bremen, D–28334 Bremen, Germany
[3] Fakultät für Mathematik, TU Chemnitz–Zwickau, D–09107 Chemnitz, Germany
[4] Research Institute for Informatics, Bd. Maresal Averescu Nr. 8–10, 71316 Bucharest 1, Romania
[5] NICONET coordinator, ESAT - Katholieke Universiteit Leuven, K. Mercierlaan 94, 3001 Leuven, Belgium, tel 32/16/32 17 03, fax 32/16/32 19 70, WWW: http://www.esat.kuleuven.ac.be/sista, email: *sabine.vanhuffel@esat.kuleuven.ac.be*
[6] Institut für Robotik und Systemdynamik, DLR Forschungszentrum Oberpfaffenhofen, Postfach 1116, D-82230 Wessling, Germany

SLICOT — A Subroutine Library in Systems and Control Theory

Peter Benner [*] Volker Mehrmann [†] Vasile Sima [‡]
Sabine Van Huffel [§] Andras Varga [¶]

September 26, 1997

ABSTRACT This article describes the subroutine library SLICOT that provides Fortran 77 implementations of numerical algorithms for computations in systems and control theory. Around a nucleus of basic numerical linear algebra subroutines, this library builds methods for the design and analysis of linear control systems. A brief history of the library is given together with a description of the current version of the library and the on-going activities to complete and improve the library in several aspects.

1 Introduction

Systems and control theory are disciplines widely used to describe, control, and optimize industrial and economical processes. There is now a huge amount of theoretical results available which has lead to a variety of methods and algorithms used throughout industry and academia. Although based on theoretical results, these methods often fail when applied to real-life problems, which often tend to be ill-posed or of high dimensions. This failing is frequently due to the lack of numerical robustness when implemented in a finite-precision environment. Moreover, the users

[*] Zentrum für Technomathematik, Fachbereich 3 – Mathematik und Informatik, Universität Bremen, D–28334 Bremen (Germany)

[†] Fakultät für Mathematik, TU Chemnitz–Zwickau, D–09107 Chemnitz (Germany)

[‡] Research Institute for Informatics, Bd. Maresal Averescu Nr. 8–10, 71316 Bucharest 1 (Romania)

[§] Department of Electrical Engineering (ESAT), Katholieke Universiteit Leuven, Kardinaal Mercierlaan 94, B–3001 Leuven–Heverlee (Belgium)

[¶] Institut für Robotik und Systemdynamik, DLR Forschungszentrum Oberpfaffenhofen, Postfach 1116, D-82230 Wessling (Germany)

of these methods are often not aware of new algorithmic developments and rely on available software. This has led to several approaches to design and implement software packages for computer-aided control system design (CACSD).

The Fortran 77 **S**ubroutine **L**ibrary **i**n **Co**ntrol **T**heory (SLICOT) is a library of widely used control system design and analysis algorithms. The intention of this library is to form the basis of new design software packages, thus avoiding duplication of software development and allowing a concentration of effort in establishing a standard set of, as far as possible, numerically robust routines with known performance in terms of reliability and efficiency.

In order to achieve the latter objective it is necessary to take into account the latest developments in numerical analysis techniques, as applicable to control algorithms, and to consult with experts in the field of numerical analysis. The current approach to this topic is to make use as far as possible of the established standard packages available for numerical linear algebra computations, i.e., the **B**asic **L**inear **A**lgebra **S**ubroutines BLAS [25, 16, 15] and the **L**inear **A**lgebra **Pack**age LAPACK [2]. These libraries form the basic layer of SLICOT. On top of this, subroutines for the numerical solution of mathematical problems often encountered in CACSD are built such as linear and quadratic matrix equations, rational matrix factorization, computation of canonical forms, etc. Furthermore, the library contains other mathematical tools not contained in the BLAS and LAPACK packages such as discrete sine/cosine and Fourier transformations, etc.

The applicability of the library is substantially enhanced when its routines are embedded in a user-friendly and widely accepted standard environment like MATLAB[1] by an appropriate interface. Such an environment provides the user a flexible and easy way of combining and experimenting with the routines of the library and the tools of MATLAB. Moreover, the use of Fortran-written routines within MATLAB often has considerable advantages with respect to execution speed compared to similar genuine MATLAB m-files. Such an interface with MATLAB is described in [37].

The outline of this paper is as follows. First, in Section 2, we will motivate the usage and implementation of a low-level subroutine library like SLICOT in contrast to using exclusively a high-level environment like MATLAB. In Section 3, a brief account on the history of SLICOT and related system and control software developments is given. The design of the library is described in Section 4. This includes a description of the structure of the library, the criteria on which the included algorithms are chosen, the user manual, the implementation and documentation standards, and the design of benchmark examples on which the included subroutines can be tested

[1] MATLAB is a registered trademark of The MathWorks, Inc.

and evaluated. Section 5 gives an account of the contents of the library, presenting the current status of SLICOT and giving a brief outlook on the next release. A complete list of the user-callable SLICOT routines in the current release can be found in Appendix A. The examples in Section 6 demonstrate that the algorithms implemented in SLICOT often outperform their equivalents in other environments such as MATLAB with respect to execution time and necessary workspace as well as accuracy. The developments planned for the future which can basically be summarized by the integration of SLICOT in a *Network for development and evaluation of numerically reliable software in control engineering and its implementation in production technologies* (NICONET) are described in Section 7. Concluding remarks are given in Section 8 and information on how to access the library and corresponding software and reports is described in Appendix B.

2 Why Do We Need More Than MATLAB Numerics?

2.1 Limitations of MATLAB

MATLAB is an excellent tool for developing and testing new algorithmic ideas or new control analysis and synthesis methods. This is a main reason for its dominance in control education and research. Still, it is important to reemphasize the need to develop and maintain control libraries written in general programming languages such as Fortran or C. In this view, we just mention three reasons for a sometimes poor performance of MATLAB in a dedicated production-quality CACSD environment.

A first reason of poor performance is the use of the dense complex matrix as the main data structure for linear algebra computations. The need to use complex computations to solve computational CACSD problems leads to very inefficient implementations of several important computational algorithms. For instance, consider the following example.

Example: Algebraic Riccati equations (AREs) can be solved in MATLAB using the functions `care` for continuous-time systems and `dare` for discrete-time systems [26]. These are based on forming the deflating subspace approach described in [3, 41] and require the computation of the stable deflating subspace of the matrix pencils corresponding to the ARE. This is achieved by using the built-in implementation of the QZ algorithm in MATLAB.

We replaced the call of the MATLAB function `qz` to determine the complex generalized Schur form and the reordering function `qzexch` to separate the stable from the unstable deflating subspace by a call to a `.mex` file implementing equivalent computations using the real versions of the same functions as provided in LAPACK.

TABLE 1. Comparison between MATLAB and Fortran based implementations.

n	Times (sec)		Speed-up	Times (sec)		Speed-up
	`care`	`care_new`		`dare`	`dare_new`	
20	1.71	0.38	4.50	0.61	0.22	2.77
40	10.76	2.42	4.44	4.94	1.54	3.20
60	42.95	9.01	4.76	16.97	4.78	3.55
80	94.14	19.72	4.77	44.27	11.09	3.99
100	150.88	33.17	4.54	83.76	22.14	3.78
120	303.46	63.44	4.78	149.94	37.24	4.02

Table 1 displays the CPU times resulting from these computational sequences required by `care` and `dare` and the new routines `care_new` and `dare_new` using the `.mex` file. We used as test data randomly generated systems of various order n. Note that the corresponding generalized eigenvalue problems have orders $2n$. The results have been obtained on a Pentium 133 Mhz PC running MATLAB Version 5 under Windows 95.

The resulting speed-up arises mainly as the effect of using real computations instead of complex computations. The use of complex computations leads in this case to a large increase of up to almost five times of the computational cost as compared to performing the real version of the algorithm. For large problems this is an unacceptable situation, especially if such computations are performed in an iterative design optimization loop.

A second reason of poor performance is the necessary trade-off to balance the MATLAB matrix handling power with the possibility to exploit intrinsic structural aspects of problems. Exploiting the structural features of computational problems often has the paradoxical effect of larger execution times which are due to heavy overheads in the interpretational operation mode of MATLAB. Thus, high order control problems can be hardly tackled in a reasonably efficient way. In contrast, implementing algorithms in Fortran or C allows the use of appropriate data structures as well as the exploitation of structural features, and thus can drastically improve the performance of various algorithms (as will be shown by various examples in Section 6). This explains why the efficiency and robustness of many MATLAB functions provided in toolboxes is minor to that of similar implementations available in Fortran control libraries.

The third reason of poor performance is the lack of numerical robustness of the implemented algorithms in many of the MATLAB toolboxes. The popular Control Toolbox [26], faced with poorly scaled systems, is particularly fragile from a numerical point of view because practically all functions are completely unaware of the possible occurrence of poorly scaled system models. Poor scaling can lead not only to spectacular failures of some functions, but even to the more dangerous situation that an apparently correct execution of the software yields seemingly meaningful results, which however are completely wrong. The existence of poorly scaled system models is not seldom in practice due to an unreflected use of SI units (see the www-link in the footnote[2] for a real-world example illustrating malfunctioning and failure of some MATLAB functions).

2.2 The need for production quality numerical software

Two recent applications using Fortran 77 implementations illustrate the high efficiency of employing dedicated production quality software to solve high order problems. The underlying numerical computations for both applications can certainly not be done efficiently by using MATLAB.

Helicopter ground resonance damping

The underlying control synthesis problem is the simultaneous output feedback stabilization of an $n = 24$ order aeromechanic helicopter multi-model with $N = 41$ vertex systems corresponding to 41 operating conditions of the rotor speed and thrust [47]. Gradient search techniques were used to find a local minimum for a quadratic performance index. The computational problem to determine the optimal output feedback control of the multi-model system was solved by turning the original problem into a high order stabilization problem (with Nn state variables) with a highly structured decentralized output feedback controller [47]. Each function and gradient evaluation involved the solution of N Lyapunov equations of order n. To solve the problem, dedicated numerical software was implemented using efficient Lyapunov solvers available in SLICOT, and a user-friendly interactive module has been implemented in ANDECS[3] to allow the efficient manipulation of a huge amount of system data stored in the hierarchical database of ANDECS [22].

Satellite attitude control

The control problem is to determine a simple controller which, making use of the periodicity of the earth magnetic field, ensures a stable operation of a satellite on an earth orbit [50]. The satellite system is described by a continuous-time linear periodic state space model of order $n = 4$, with $m = 1$ control inputs and $p = 2$ measurable outputs. The continuous-time periodic problem was turned into a discrete-time periodic problem by successive discretization of the continuous-time system over $N = 120$ time

[2]http://www.op.dlr.de/FF-DR-ER/research/num/scaling.html

[3]ANDECS (**An**alysis & **De**sign of **C**ontrolled **S**ystems) is a trademark of *Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V.* (DLR) Oberpfaffenhofen, Germany.

intervals. The proposed synthesis method relied on using an optimal periodic output feedback control law which minimizes an associated quadratic cost function [50]. A gradient search based optimization approach, specially developed for large order problems, was used to compute the N output feedback matrices of order m-by-p. Each function and gradient evaluation involved the numerical solution of a pair of discrete-time periodic Lyapunov equations over N periods. Turned into a standard problem, this computation is equivalent to solving a pair of standard discrete-time Lyapunov equations of order Nn (for our application this order is 480!). Without special algorithms this is a prohibitively expensive computational task with an approximate cost of $25(Nn)^3$ operations. Efficient and numerically reliable algorithms based on the periodic Schur decomposition have been developed for the solution of these equations [48]. The associated cost is about $25Nn^3$ operations and thus leaves the computational burden acceptable. It was shown that for all practical purposes the proposed periodic output feedback approach is a completely satisfactory alternative to existing spacecraft attitude control techniques. To solve the periodic output feedback control problem, dedicated numerical software was implemented in Fortran to compute the periodic Schur decomposition of a product of many matrices and to solve periodic discrete-time Lyapunov equations.

2.3 Low level reusability of Fortran libraries

Many sophisticated CACSD platforms available today like ANDECS [22], EASY5 [9], MATLAB [27], Scilab [11], or Xmath [24] rely on robust implementations of numerically reliable and computationally efficient algorithms. In the architecture of each CACSD platform we can explicitly identify and usually also access a basic computational layer consisting of subroutine libraries, RASP[4] in ANDECS [21], SLICOT in EASY5, or intrinsic functions and function toolboxes (in Scilab, MATLAB, and Xmath). This layer includes all computational routines for CACSD specific mathematical and control computations, simulation and optimization. An important advantage of developing libraries like RASP and SLICOT is that the development of the control library is not restricted by specific requirements of the CACSD platform operation, by the languages used for its implementation, or by the employed data structures. Moreover, such control libraries can serve the development of more than one platform or can be used within other dedicated engineering software systems. This *low-level reusability* has been achieved primarily by using a general purpose programming language like Fortran 77. This option of the implementors of RASP and SLICOT led to a true independence of the control libraries from any CACSD platform

[4]RASP is a product of the *Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V.* (DLR) Oberpfaffenhofen, Germany

and was enhanced by embedding or linking the high performance linear algebra packages BLAS, EISPACK, LINPACK, and recently LAPACK. For dedicated applications written in the C language, automated Fortran to C conversions are possible (by using for instance the `f2c` compiler[5]). Links to MATLAB and Xmath toolboxes are also possible by building appropriate gateways. Note that the low-level reusability resulting from the use of Fortran subroutine libraries can hardly be achieved in CACSD platforms like MATLAB or Xmath.

2.4 Structure preserving algorithms

A contemporary research streamline in numerical algorithms is the exploitation of any structural information of the underlying computational problem. The main advantage of developing structure-preserving algorithms is that the structural properties of a problem are preserved during finite precision computations. This allows the computed result to be interpreted as the exact solution of the original problem with perturbed input data which may not be the case if the properties of the problem are changed due to rounding errors. This not only increases the reliability of the returned results, but often also improves their accuracy. For instance, consider the following situation.

Example: The *controllability* and *observability Gramians* P_c, P_o, respectively, of a stable state-space realization (A, B, C) of a linear system are given by the solution of the stable *Lyapunov equations*

$$\begin{aligned} AP_c + P_cA^T &= -BB^T, \\ A^TP_o + P_oA &= -C^TC. \end{aligned}$$

As A is stable and the right-hand sides of the above equations are negative semidefinite, Lyapunov stability theory shows that P_c and P_o are positive semidefinite. The square roots of the eigenvalues of their product P_cP_o are known as the *Hankel singular values*. These eigenvalues play a fundamental role in finding balanced realizations and in model reduction.

As the eigenvalues of P_cP_o are all real nonnegative, so are the Hankel singular values. Solving the above Lyapunov equations without taking into account symmetry and semidefiniteness, roundoff errors can cause the computed Gramians to be non-symmetric and/or non-definite. This can then result in negative or complex Hankel singular values — a complete nonsense regarding the system-theoretic properties.

Besides this, exploiting the structure inherent to the considered problem usually results in a reduction of necessary computational operations and memory requirements. Without exploiting the problem structure, many

[5] `f2c` is available from netlib, `http://www.netlib.org/f2c/`

computational tasks are simply too expensive to be performed. However, exploiting problem structures also raises some difficulties associated with a more complex implementation of algorithms.

MATLAB is not appropriate to exploit intrinsic structural aspects of problems because of the necessary trade-off in balancing the higher level matrix handling power with the need for low level programming. Implementing structure preserving or structure exploiting algorithms is possible only in a general purpose programming language like Fortran 77 or C. Because of the full flexibility allowed by such programming languages, every algorithmic detail can be explicitly addressed, every optimization of the computational flow can be incorporated, every opportunity to minimize the used memory can be exploited. Thus, high quality robust numerical software can be created for numerical computations in control, similar to those for solving basic linear algebra problems in dedicated packages as EISPACK, LINPACK, or LAPACK.

Structure-preserving algorithms have been developed for many control domains and robust software implementations will be available in the forthcoming release of SLICOT; see Section 5.3. Some of the most recent developments are: balancing-free square-root methods for model reduction [46], periodic Schur methods for the solution of periodic Lyapunov equations [48], descriptor systems analysis procedures [28], symplectic methods to solve Riccati equations [1, 4, 32], and subspace identification methods [53, 44, 31].

3 Retrospect

3.1 Short history of control subroutine libraries

The development of efficient, reliable, and portable numerical software requires joining expertise in theory, in numerical mathematics, in numerical programming, and in numerical software engineering. Hence the development of tested, production-quality numerical software is a challenging and time-consuming task which involves cooperative efforts over a lasting period of time. The generally accepted form for such software is a subroutine library, i.e., a portable integrated collection of basic subroutines which can be used as building blocks for the construction of all sorts of complex interactive CACSD packages.

Several efforts have been initiated in the past to develop such control libraries. The Scandinavian control library [17, 54] and the Swiss control library AUTLIB [10] were the first initiatives in the field. These libraries did not reach maturity, which illustrates the difficulties encountered by such a project. Another British cooperation initiative, coordinated by Mike Denham at Kingston Polytechnic in the early eighties, led to the development of the control library SLICE which contains a set of almost 40 control routines [13, 12]. When this initiative stopped, the routines were further distributed by the Numerical Algorithms Group (NAG) from Oxford (UK) who issued a revised version of SLICE.

In the same period, the Working Group on Software (WGS) was founded as a Benelux cooperation between several academic institutes and industries that felt the necessity of combining software implementation efforts with respect to realising reliable control software. The objectives of the WGS were first to bring together the existing numerical software for control and systems theory in a widely available library and next to extend this library to cover as far as possible the area of industrial applications. It has a strong academic representation, which means that the members of the group are very active in the development and implementation of numerical methods for control systems design. Although the WGS started as a Benelux initiative, it now starts operating on a European level (see Section 7). First, an inventory was made of all available numerical control software [56] after which the group concentrated on the development of a standard control library, called SYCOT [55]. Important aspects of a software product are consistency in development and continuity in maintenance. These were not sufficiently covered in the early constitution of the WGS. Also, in order to produce a library that meets professional standards (documentation, implementation, portability, etc.), the WGS associated itself in the late eighties with NAG which produces the world renowned NAG Library for basic mathematical computations. NAG and WGS decided to combine their expertise and integrate their libraries, SLICE and SYCOT, into a new library, called SLICOT. This cooperation was quite effective and led to the first release of SLICOT in 1990 [35]. In a later phase a collaboration was pursued with the originators of the RASP library (an initiative from the German Aerospace Research Establishment DLR) and the resulting extension of routines in the SLICOT collection led to the second release of SLICOT in 1993 [36].

The development of the RASP library also started in the early eighties in Germany [20] and was partly based on the libraries EISPACK [34] and LINPACK [14]. Apart from routines, mostly for control, RASP also offers main programs for several chapters of control and systems theory and has found ample application in German industry. An interesting contribution from Romania is the mathematical library BIMAS [51, 52] which extends the capabilities of EISPACK and LINPACK. The control library BIMASC [45, 49] is based on BIMAS and provides many subroutines for analysis, modeling, design and simulation of multivariable systems. Another East-European control library with name LISPACK was developed by a Bulgarian group from Sofia. This library is included in the interactive SYSLAB package [30].

3.2 Standard libraries RASP and SLICOT: present status

Apparently, to date, only RASP and SLICOT are in active further development. The other initiatives failed to evolve due to a number of reasons, such as the lack of sufficient supporting and contributing research sites, the vast amount of work associated with the development of a more or less complete library, the absence of facilities of maintenance and the apparently small potential market of customers in the eighties.

As described in Section 5, SLICOT, a general purpose basic control library realised by WGS in cooperation with NAG, can primarily be viewed as a *mathematical library for control theoretical computations.* The library provides tools to perform basic system analysis and synthesis tasks. The main emphasis in SLICOT is on *numerical reliability* of implemented algorithms and the *numerical robustness and efficiency* of routines. Special emphasis is put on providing maximal algorithmic flexibility to users, and on the use of rigorous implementation and documentation standards (see [57, 59]). The previous and present releases of SLICOT contain about 90 user-callable routines (see [39, 36] and Table 15 in Appendix A), related to analysis, modeling, transformation and synthesis of systems. Future extensions will cover routines for descriptor systems, model reduction and subspace identification.

RASP covers a broad area of control engineering computations supporting frequency- and time-domain analysis and synthesis techniques, multicriteria parameter optimization, simulation and graphics. Special attention is given to the *numerical reliability* of the implemented algorithms. Currently, RASP consists of about 320 user-callable routines and is used in education and research at many universities and research sites in Germany. RASP and the engineering-database and -operating system RSYST [21], form together the software infrastructure of the computer aided control engineering environment ANDECS [22].

3.3 RASP/SLICOT mutual compatibility concept

Because both, RASP and SLICOT, will continue to evolve, there are serious concerns to rationalise future developing activities, that is, to avoid duplication of implementation of existing software of good quality. Recently, DLR and WGS have agreed to combine their activities in order to come to a joint library. This joint library is the first step towards a standard platform for computational tools. For the sake of this cooperation, the RASP/SLICOT mutual compatibility concept has been introduced [23], which enables a coordinated development of both libraries leading to a reduction of software implementation and testing efforts without giving up the identity of the separate libraries. Part of this agreement is the incorporation of the numerical linear algebra packages, BLAS [15] and LAPACK [2], in both libraries and in the future joint library. A first development along the lines of the mutual compatibility concept is the model reduction library RASP-MODRED [46] of about 20 user-callable routines. A more recent development in this direction is the public release of SLICOT; see Section 5.2.

The RASP/SLICOT cooperation is only a first step towards the realisation of a standard, generally accepted platform for computational control tools. Therefore, WGS, NAG, and DLR recently took the initiative to extend the scope of cooperation to a European level (see Section 7) and make SLICOT freely available in order to ensure a faster and wider distribution of these computational tools.

4 The Design of SLICOT

4.1 Structure of the library

The library is divided into several chapters, each devoted to a global area. Each of the global areas is then again divided into subchapters and sections devoted to more specific problem areas and specific problems, respectively. The structure of the library is given in [59].

The library contains two categories of routines accessible to users:

Fully documented (user-callable) routines which are each documented by a Library Routine Document in the Library Manual.

Supporting (lower-level) routines which are intended to be used, not by the generality of users, but by experts, by software developers, and especially by contributors to the library. The Library Manual contains a list of the names of such routines and their capabilities. More detailed documentation is provided by in-line documentation only.

The supporting routines are a privileged selection from the general collection of auxiliary routines. They give users access to separate components of the computations performed by the fully documented routines.

As a rough guide, a fully documented routine is designed to solve a complete problem (e.g., to solve a Lyapunov equation). A supporting routine would perform some part of the necessary computations (e.g., solve a Lyapunov equation where the coefficient matrix is in real Schur form).

In each chapter the first subchapter is reserved for service routines (auxiliary routines) to perform elementary or subsidiary computations.

The policy of the library is not to introduce an unnecessary number of routines into it. In order to perform a number of related tasks, or to perform a main task with a variety of subtasks, a single routine with a mode parameter is usually a satisfactory design, provided that most of the parameters are used for all option-settings.

4.2 Choice of algorithms

The main criteria for including an algorithm in the library are:

Usefulness: An algorithm must solve problems of practical utility to at least some section of the intended users of the library.

Robustness: An algorithm must either return reliable results, or it must return an error or warning indicator, if the problem has not been well-posed, or if the problem does not fall in the class to which the algorithm is applicable, or if the problem is too ill-conditioned to be solved in a particular computing environment. If this requirement is too stringent (as it may be in some chapters of the library), the documentation must make the possible risks very clear.

Numerical stability and accuracy: Algorithms are supposed to return results that are as good as can be expected when working at a given precision. If available at low cost, algorithms should provide a parameter which estimates the accuracy actually achieved. The documentation should give a clear simple statement of the accuracy to be expected, if possible as a rigorous upper bound, otherwise as a conservative estimate.

Speed: An algorithm should never be chosen for its speed if it fails to meet the usual standards of robustness, numerical stability, and accuracy, as described above. Speed is evaluated across a wide range of computing environments. Therefore, usually the number of floating-point operations, or the number and cost of iterations are considered.

Modern computer architectures: The requirements of modern computer architectures must be taken into account. As SLICOT aims at the same scope of computing environment as LAPACK, i.e., standard scalar processors as well as shared-memory vector and parallel processors, the differences in the various architectures may imply different choices of algorithms. If a reasonable compromise is not possible, more than one algorithm for the same problem may need to be included in the library.

4.3 User Manual

The SLICOT User Manual is organized in chapters. Each chapter consists of:

- an introduction to the problem area;
- a table of contents, with a list of all fully documented (user-callable) routines of the chapter;

- a Library Routine Document for each user-callable routine.

Currently, the user manual is available as on-line documentation in HTML format (filename `libindex.html`) at the world wide web location

`http://www.win.tue.nl/wgs/slicot.html`

using the link to the `FTP site` of freeware SLICOT or by directly linking to

`ftp://wgs.esat.kuleuven.ac.be/pub/WGS/SLICOT/`

A printed version of the user manual for the next SLICOT release is planned for the future and will be distributed by NAG.

4.4 Implementation and documentation standards

For the development of a unified library and in order to obtain a unified style in the implementation and documentation of the subroutines provided by many different experts in the field, a documentation and implementation standard was developed for SLICOT [57]. A general observation that was made over the years is that this standard also proved to be a very valuable tool in teaching students how to implement algorithms in the context of their studies.

With the development of the LAPACK library and the decision to base new releases of SLICOT as much as possible on basic routines from this library, it was necessary to adapt the standard to be compatible with the standards used in LAPACK. These new SLICOT Implementation and Documentation Standards 2.1 [59], were developed in the context of updating SLICOT from Release 2 to 3.0.

It is required that every new submission to the library is implemented and documented according to this standard. In order to help new contributors, a report with general guidelines for contributions was developed [58].

4.5 Benchmarks

In the analysis of numerical methods and their implementation as numerical software it is extremely important to be able to test the correctness of the implementation as well as the performance of the method. This validation is one of the major steps in the construction of a software library, in particular if this library is used in practical applications. To have a fair evaluation and a comparison of methods and software, there should be a standardized set of examples that allows an evaluation of the performance of a method as well as the implementation with respect to correctness, accuracy, and speed, and also to analyse the behaviour of the method in extreme situations, i.e., on problems where the limit of the possible accuracy is reached.

In many application areas therefore, benchmark collections have been created that can partially serve this purpose. Such collections are heavily used. It is one of the goals of future releases of SLICOT to create such testing and validation environments for the area of numerical methods in control and to accompany SLICOT with benchmark collections for each of the major problem areas.

Currently, SLICOT contains two such benchmark collections in the areas of linear quadratic control (Riccati equations) [5, 6, 7]. Other collections are currently being developed. See the WGS homepage for details.

5 Contents of SLICOT

5.1 Current contents of the library

A detailed presentation of the previous SLICOT Releases 1 and 2, including chapter-by-chapter summaries, and individual user-callable routine documents, can be found, for instance, in [36]. The HTML files for the on-line documentation of the current SLICOT version are available at the WGS ftp site[6]; see also Appendix B. This section summarizes the contents of the library, indicating the main functional abilities included. For a complete list of user-callable SLICOT routines, see Appendix A.

The main linear time-invariant system representations used by SLICOT routines are listed below:

State-space representations

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned}$$

where E, A, B, C, and D are matrices of appropriate dimensions, and λ represents either the differential operator $\mathrm{d}/\mathrm{d}t$, for a continuous-time system, or the advance shift operator z for a discrete-time system. This *descriptor model* reduces to the standard model when $E = I$.

Polynomial matrix representations

$$\begin{aligned} D_r(\lambda)x(t) &= u(t), \\ y(t) &= N_r(\lambda)x(t), \qquad \text{or} \\ D_l(\lambda)y(t) &= N_l(\lambda)u(t), \end{aligned}$$

where the matrices $N(\lambda)$ and $D(\lambda)$ are polynomial matrices, and λ has the same meaning as above.

[6] `ftp://wgs.esat.kuleuven.ac.be`, directory `pub/WGS/SLICOT/` and its subdirectories.

Rational matrix representations

$$Y(\lambda) = H(\lambda)U(\lambda),$$

where $H(\lambda)$ is a rational matrix of appropriate dimensions, λ is either s or z, the complex variables appearing in the Laplace- and z-transforms, respectively, and $Y(\lambda)$ and $U(\lambda)$ are the corresponding transforms of the output and input signals.

Time response representations are defined as pairs $(y(t_i), u(t_i))$, $i = 1, 2, \ldots, N$. For a state-space system (A, B, C, D) with a given input function $u(t)$, and zero initial state, $y(t_i)$ is computed as

$$y(t_i) = \int_0^{t_i} \left[C \exp^{At} B + D\right] u(t_i - t)\, \mathrm{d}t.$$

Non-zero initial states can also be handled.

A chapter-by-chapter short presentation of SLICOT follows.

Chapter A (Analysis Routines) includes routines devoted to the analysis of a dynamical system given in some of the abovementioned model representations. While the scope of this chapter would include polynomial or rational matrix analysis and frequency or time response analysis, the current release covers state-space analysis only, namely: computation of structural indices, zeros, and special subspaces for standard problems. Certain intermediate constructions (such as canonical forms) or computations (such as change of basis, interconnections of subsystems, dual systems, zeros of multivariable systems, etc.) are also included.

Chapter D (Data Analysis) includes routines for the computation of specific characteristics of a signal or data set: covariances, sine/cosine transform of a signal, fast Fourier transform of a signal, anti-aliasing window. Other calculations, like determining statistical properties, trend removal, z-transforms, prediction, or filter design, are planned for a future release of the library.

Chapter F (Filtering) includes routines performing certain filter operations or designing specific Kalman filters for discrete-time systems described by state-space models. Both time-varying and time-invariant square root covariance and information filters are dealt with. Special fast implementations, exploiting the lower observer or upper controller Hessenberg forms provided by other routines, are available for the time-invariant case. LPC filters and fast recursive least-square filters are possible candidates for future library additions.

Chapter M (Mathematical Routines) contains routines for basic mathematical operations, not included in BLAS or LAPACK collections. Most routines are viewed as auxiliary routines, but they can serve as building blocks for solving other problems in systems and control theory and related domains. A set of subroutines performs numerical linear algebra operations: solution of special linear systems (e.g., with triangular or complex upper Hessenberg matrices), total least squares using (partial) singular value decomposition [43], etc. Another set of subroutines is included for algebraic operations involving scalar polynomials or polynomial matrices.

Chapter S (Synthesis Routines) includes routines for the design of a system with some desired behaviour and following some prescribed rules. Routines for the solution of Sylvester, Lyapunov, and Riccati equations are also provided. Typical state-space design techniques, e.g., observer design, pole assignment, feedback design, deadbeat control, optimal regulation, etc., are based on the solution of such equations. Polynomial or rational matrix synthesis techniques, as well as frequency or time response synthesis, are currently empty sections to be filled in the future.

Chapter T (Transformation Routines) contains routines for transforming one representation of a linear time-invariant system to another. More specifically, routines for transforming state-space, polynomial matrix, rational matrix, or time response representations, to any other representation are included. Frequency response representations will also be considered in the future.

Chapter U (Utility Routines) contains utility routines.

5.2 Development of the public release of SLICOT

Converting the NAG SLICOT library to a freely available software package, and adapting it to the new implementation and documentation standards, has been a time consuming activity, which also offered the opportunity to improve the codes as much as reasonably possible. This process, which resulted in *Release 3.0* of SLICOT, has evolved in several steps (not necessarily taken in sequence):

1. Removing the dependence of SLICOT routines on proprietary NAG routines, by replacing NAG routine calls by equivalent BLAS or LAPACK routine calls, and by implementing new mathematical routines, e.g., for solving linear systems with complex upper Hessenberg coefficient matrices, or computing QR- and LQ-decompositions of some structured or partitioned matrices.

2. Adapting the codes to the new LAPACK-like implementation and documentation standards. This was achieved by implementing new user interfaces (including CHARACTER-type option parameters) and new error handling schemes.

3. Improving the *modularity*, by restructuring the user-callable routines, functional overloading, etc.

4. Improving the *functionality*, by adding new features to increase the flexibility of usage.

5. Improving the *performance*, by reprogramming virtually all routines, turning the BLAS 1 and BLAS 2 calls into BLAS 3 calls whenever possible, using LAPACK block algorithms, exploiting any special problem structure, etc. The use of upper level BLAS routines resulted also in improved clarity and compactness of the source codes.

6. Improving the *reliability*, by replacing less stable calculations by mathematically equivalent, numerically stable ones.

The SLICOT Release 3.0 routines have been checked using adaptations of the previous test programs, data and results. Part of the routines have also been tested by using the NAGWare Gateway Generator [37], which facilitates the integration of SLICOT into MATLAB. New MATLAB test programs have been written, which call the Fortran routines via this gateway. Equivalent computations have been performed in MATLAB, so that it was possible to compare the efficiency and accuracy of Fortran and MATLAB implementations; see Section 6.

5.3 In the queue

In the future, several new developments are planned for the next releases of SLICOT. Several new subroutines in the context of model reduction will be incorporated. The section on factorization of transfer functions will be extended and several new methods concerning the solution of optimal control problems will be added. It is planned to fill the new sections and subsections on subspace identification, descriptor systems, and periodic systems. Some of these new developments will be developed in the context of the NICONET project and some are already available or have to be transformed to the new standard.

6 Performance Results

This section reports preliminary performance results (efficiency, reliability, and accuracy) for some components of the new SLICOT release.

The results reported below have been obtained on a SUN Ultra 2 Creator 2200 workstation with 128 M RAM and operating system SunOS 5.5, by calling from MATLAB the gateways produced by the NAGWare Gateway Generator for the corresponding SLICOT codes. (The SLICOT routines have been compiled with `f77` using options `-O4 -native -u`.) These results show that SLICOT routines often outperform MATLAB calculations. While the accuracy is practically the same, or better, the gain in efficiency by calling SLICOT routines can be significant for large problems. Note that the figures have been obtained by timing in MATLAB the equivalent computations. Even better efficiency is to be expected by calling the SLICOT Fortran routines only (not through the gateway), and similar accuracy/efficiency improvements are possible for other SLICOT computations, especially for large problems, due to the incorporated calls to upper level BLAS routines, and performant LAPACK blocked (and unblocked) algorithms.

The following tables indicate the accuracy (measured by either the relative errors or relative residuals, when available), and the time spent exclusively for equivalent SLICOT and MATLAB computations, for some routines in Chapters A, D, F, and M of the library.

The **main conclusions** of the tests are:

1. The SLICOT gateways are usually several times faster than MATLAB.
2. The accuracy of SLICOT routines is at least as good as, or better than, that for MATLAB calculations.
3. Less memory is required by SLICOT routines for equivalent calculations, because the problem structure is fully exploited.

Typical results

(i) Tables 2 and 3 display comparative results for the SLICOT Fast Fourier transform routines (`DG01MD` for complex sequences, and `DG01ND` for real sequences, respectively) and the corresponding MATLAB function `fft`. For this purpose, random sequences `X` of length n were generated. Besides better efficiency, the accuracy of SLICOT routines shows an improvement which can be two orders of magnitude. The accuracy was measured by computing the distance between the original sequence and the inverse Fourier transform of the transformed sequence by

```
norm(X - ifft( fft( X ) ))/norm(X);
```

(which should theoretically be zero), and similarly for the SLICOT gateways. Note that `DG01ND` is 2–4 times faster than MATLAB.

(ii) Table 4 and 5 display comparative results for the SLICOT linear systems solver `MB02OD` for triangular matrices, which includes a condition number estimator, and the MATLAB operation `A\B`. Triangular sets of linear systems, defined by $AX = B$, with $A \in \mathbb{R}^{n \times n}$, and $B \in \mathbb{R}^{n \times m}$ matrices were solved, where

```
A = triu(rand(n,n)) + eye(n)/2;  X = rand(n,m);  B = A*X;
```

Timing results, and relative error and residual norms of `MB02OD` and MATLAB results are given. Note that the relative errors are the "true"

TABLE 2. Comparison between `DG01MD` and MATLAB results.

n	Time		Relative error	
	`DG01MD`	MATLAB	`DG01MD`	MATLAB
1024	0.00	0.00	`7.54e-16`	`8.14e-15`
2048	0.01	0.02	`1.46e-15`	`6.77e-15`
4096	0.02	0.03	`1.36e-15`	`2.49e-14`
8192	0.03	0.06	`2.33e-15`	`2.02e-14`
16384	0.08	0.13	`2.32e-15`	`1.10e-13`
32768	0.21	0.34	`3.35e-15`	`2.96e-13`
65536	0.80	1.22	`5.04e-15`	`4.77e-13`

TABLE 3. Comparison between `DG01ND` and MATLAB results.

n	Time		Relative error	
	`DG01ND`	MATLAB	`DG01ND`	MATLAB
1024	0.00	0.01	`6.44e-16`	`6.56e-15`
2048	0.00	0.01	`1.06e-15`	`6.56e-15`
4096	0.01	0.02	`1.44e-15`	`2.00e-14`
8192	0.02	0.05	`1.95e-15`	`2.13e-14`
16384	0.03	0.11	`2.46e-15`	`8.21e-14`
32768	0.07	0.27	`2.81e-15`	`2.46e-13`
65536	0.23	0.88	`4.22e-15`	`4.26e-13`

errors, because A and X have first been chosen and then used to compute $B = AX$, so the true results were known. For large matrices, the SLICOT gateway can be more than 5 times faster than MATLAB. The accuracy of `MB02OD` and MATLAB results is the same. The value 1 for the warning/error indicator `info` (an output parameter of `MB02OD` routine) shows that matrix A can be considered as numerically singular.

Tables 6 and 7 give, for comparison purposes, some results obtained using the MATLAB function `inv` for matrix inversion, for other random realizations. Note that the SLICOT gateway can be more than 10 times faster than MATLAB (for large matrices), and it obtains better relative error and residual norms. Many MATLAB codes call `inv(.)` instead of using `A\B`, especially when solving $XA = B$.

(iii) Tables 8 and 9 display comparative results for some SLICOT Kalman filter routines (Chapter F), and equivalent MATLAB calculations. The Chapter F routines update some covariance/information square root factors after one iteration of the Kalman filter algorithm; special

TABLE 4. Comparison between MB02OD and MATLAB results: timing.

			Time	
n	m	info	MB02OD	MATLAB
64	16	0	0.01	0.02
128	32	0	0.03	0.12
256	64	0	0.17	0.99
512	128	1	1.69	9.42

TABLE 5. Comparison between MB02OD and MATLAB results: accuracy.

			Relative error	Relative residual
n	m	info	MB02OD, MATLAB	MB02OD, MATLAB
64	16	0	4.75e-14	6.74e-17
128	32	0	1.39e-12	5.80e-17
256	64	0	1.97e-08	6.57e-17
512	128	1	1.14e-02	6.60e-17

cases of system matrices in lower observer or upper controller Hessenberg forms are dealt with by separate routines. The updates are obtained by applying either LQ or QR factorizations to some structured "pre-arrays", using the new routines MB04LD (and, possibly, MB04JD) or MB04KD (and, possibly, MB04ID), respectively. The routines FB01SD and FB01TD considered in Table 8 and 9, respectively, update the information square roots for general or upper Hessenberg matrices, respectively. The relative errors reported are the differences between SLICOT and MATLAB results. Note that SLICOT codes were about 2–4 times faster than MATLAB calculations, at comparable or better accuracy.

(iv) Tables 10 and 11 display comparative results for the SLICOT subroutines MB05MD and MB05OD, for the computation of the matrix exponential, and the equivalent MATLAB functions expm3 and expm1, respectively. While the algorithms implemented in SLICOT and MATLAB are basically the same, the implementation details differ significantly.

We have used randomly generated matrices of order n for comparison purposes. The relative errors reported are the differences between the results obtained from SLICOT and MATLAB. Note that the SLICOT codes were usually faster (up to four times) than MATLAB calculations, at a comparable accuracy. Moreover, for upper triangular matrices, MB05OD can be much faster than expm1. The main reason for this is that the balancing strategy implemented in MB05OD can reduce

TABLE 6. Comparison between `MB02OD` and MATLAB `inv(.)` results: timing.

			Time	
n	m	`info`	`MB02OD`	MATLAB
64	16	0	0.00	0.01
128	32	0	0.02	0.11
256	64	0	0.20	1.60
512	128	1	1.79	20.07

TABLE 7. Comparison between `MB02OD` and MATLAB `inv(.)` results: accuracy.

			Relative error		Relative residual	
n	m	`info`	`MB02OD`	MATLAB	`MB02OD`	MATLAB
64	16	0	9.19e-14	1.06e-13	5.25e-17	1.77e-15
128	32	0	1.01e-11	1.02e-11	5.91e-17	9.95e-14
256	64	0	1.71e-08	1.89e-08	6.98e-17	1.10e-10
512	128	1	1.64e-02	2.00e-02	6.22e-17	1.17e-04

the matrix norm for triangular matrices (and save matrix multiplications), while `expm1` cannot. Table 12 illustrates the speed-up for matrices generated by

```
A = triu(rand(n,n),n/2);
```

An increase of over 30 times in speed has been achieved for $n = 256$.

(v) The last example in this section demonstrates the behaviour of the SLICOT routine AB01ND for computing the controllability staircase form of a linear system in state-space form given by a matrix pair (A, B), $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$.

We tested randomly generated examples of increasing order. Relative errors were computed using the following formula for the SLICOT subroutines:

```
max(norm(z'*A*z - a)/norm(A),norm(z'*B - b)/norm(B));
```

where A and B are the given system matrices, `a`, `b`, are the computed matrices in controllability staircase form, and `z` is the computed orthogonal transformation matrix.

The SLICOT computations were compared to the MATLAB function `ctrbf` provided by the Control Toolbox [26]. In both approaches, the built-in tolerances were used for numerical rank decisions. Note that

TABLE 8. Comparison between FB01SD and MATLAB results.

			Time		Relative error norms
n	m	p	FB01SD	MATLAB	FB01SD–MATLAB
16	4	8	0.01	0.00	4.74e-16
32	8	16	0.01	0.01	9.99e-16
64	16	32	0.05	0.11	1.55e-15
128	32	64	0.36	0.76	2.10e-15
256	64	128	3.40	7.16	1.84e-15

TABLE 9. Comparison between FB01TD and MATLAB results.

			Time		Relative error norms
n	m	p	FB01TD	MATLAB	FB01TD–MATLAB
16	4	8	0.00	0.01	1.81e-16
32	8	16	0.01	0.01	3.10e-16
64	16	32	0.03	0.11	2.25e-16
128	32	64	0.17	0.76	1.86e-15
256	64	128	1.48	6.48	1.67e-15

using the smaller SLICOT tolerance in `ctrbf` results in even larger execution times for `ctrbf`. It should be mentioned that `ctrbf` uses singular value decompositions for rank determination while `AB01ND` employs rank-revealing QR factorizations.

Relative errors for MATLAB computations are not given as `ctrbf` accumulates the computed transformation matrix and applies it to the original system. Using the above error formula, this results in a zero relative error.

Further note that `ctrbf` also applies the transformation to the output matrix C of the linear system. This matrix is chosen here as `eye(1,n)` such that the additional time required for computing `c=Cz` is negligible in this context.

Tables 13 and 14 show the behaviour for multi-input and single-input systems, respectively. For the given examples, the SLICOT routines are up to ten times faster than `ctrbf` in the multi-input case and more than **250** (!) times faster in the single-input case.

TABLE 10. Comparison between `MB05MD` and MATLAB `expm3` results.

	Time		Relative error norms
n	MB05MD	MATLAB	MB05MD–MATLAB
16	0.01	0.01	5.84e-15
32	0.03	0.04	1.08e-14
64	0.18	0.33	1.99e-14
128	1.21	2.60	1.82e-14
256	9.29	34.04	1.43e-13
512	103.71	426.76	1.21e-13

TABLE 11. Comparison between `MB05OD` and MATLAB `expm1` results.

	Time		Relative error norms
n	MB05OD	MATLAB	MB05OD–MATLAB
16	0.00	0.00	1.57e-15
32	0.01	0.02	1.43e-14
64	0.14	0.11	3.61e-14
128	1.28	0.80	7.73e-14
256	11.64	19.14	1.66e-13

7 The Future — NICONET

7.1 Objectives and exploratory phase of NICONET

It is clear that the efforts to develop SLICOT are very intensive. Therefore, as mentioned in Section 3, a coordinated future development of RASP and SLICOT into a joint library has been established to reduce the implementation efforts. In order to extend these RASP/SLICOT coordination efforts to other European software development initiatives in the area of numerical control, a thematic “Numerics in Control” network, entitled *Network for development and evaluation of numerically reliable software in control engineering and its implementation in production technologies*, has been set up. The Commission of the European Union has given an exploratory award to WGS from October 1996 till April 1997 in order to start up this Numerics in Control Network – with acronym NICONET[7] – in the field of Industrial and Materials Technologies. The objectives of NICONET can be summarized as follows:

- to *intensify* the research in and collaboration on Numerics in Con-

[7]For more information, see the NICONET world wide web homepage `http://www.win.tue.nl/wgs/niconet.html`

TABLE 12. Comparison between `MB05OD` and MATLAB `expm1` results: `A = triu(rand(n,n),n/2)`.

	Time		Relative error norms
n	MB05OD	MATLAB	MB05OD–MATLAB
16	0.00	0.00	4.84e-17
32	0.00	0.02	7.67e-17
64	0.02	0.11	1.60e-16
128	0.08	0.86	2.14e-16
256	0.61	19.01	1.99e-16

TABLE 13. Comparison between `AB01ND` and MATLAB results.

		Time		Relative errors
n	m	AB01ND	MATLAB	AB01ND
16	2	0	0.01	4.75e-16
32	4	0.01	0.04	3.92e-16
64	8	0.04	0.22	6.05e-16
128	16	0.32	1.44	6.46e-16
256	32	2.56	25.10	1.35e-15

trol in which European teams play a prominent role on a world scale. This is achieved by stimulating the collaboration of control specialists and numerical scientists, by exchanging specialists between academic and industrial research units and by the organisation of information exchange between academic and industrial research units within Europe.

- to *integrate* the SLICOT and RASP control libraries into a joint library, to *extend, improve* and *benchmark* it and to *adapt* it for easy implementation in general purpose CACSD packages.

- to *ensure the transfer* of information technology related to control of industrial processes to industry. To *facilitate access* to high technology software and convince industrial developers of the feasibility of this software, and the benefits in using it.

The results of a questionnaire performed during the NICONET exploratory phase confirm the present need for performant numerical software and greatly support coordination of this activity on a European level. These results reflect the opinion of 55 universities or research centers and 17 industrial companies spread over Europe and can be summarized as follows

TABLE 14. Comparison between AB01ND and MATLAB results, $m = 1$.

	Time		Relative errors
n	AB01ND	MATLAB	AB01ND
16	0.01	0.04	6.26e-16
32	0.01	0.14	4.16e-16
64	0.04	1.29	5.61e-16
128	0.30	25.55	1.09e-15
256	2.48	639.57	1.05e-15

(for more details, see [60, 42]). The expansion of the present WGS network (NICONET partners in the exploratory phase) to a European level is strongly encouraged in order to obtain a wider base of software developers and potential users. However, quality control on the software made available must be imposed. Therefore, feedback from others than the developers is absolutely necessary. The network will focus on the development of numerically reliable and efficient control related software freely available and embedded in a user-friendly environment such as MATLAB in order to guarantee its widespread use in both academia and industry. A homogeneous user-interface is of high importance: most software users want to have flexible and powerful, but easy-to-use tools, and are willing to sacrifice speed for ease of use. However, people do want reliable answers, and therefore reliable software is needed. In addition, the library should be benchmarked and validated by means of real industrial examples. Commercial support should be provided too, especially for industry. The design of a library for large-scale applications will complement SLICOT. In this context high-performance computer tools are appropriate (software packages like BLAS, LAPACK, and new parallel packages should be used). In addition, the use of electronic means is widely accepted as the most flexible and user-friendly way to enhance information exchange and cooperation within the network. Finally, it should be noted that all but one replier of the questionnaire use MATLAB which is mainly due to the user-friendliness of the product. Also, the MATLAB toolboxes are very popular (used by more than 90% of the repliers). More than 30 software contributions have been reported in diverse control areas, in particular in system identification, optimal control, model reduction, time and frequency response. Despite these contributions, almost all repliers expressed their needs for new software (preferably in MATLAB or Fortran 77) in all areas of systems and control and pointed out the lack of tools to increase the efficiency of software development in process control applications. To realize these aims, NICONET will focus on the activities described in the following sections.

7.2 Development of performant numerical software for CACSD

In the past, WGS essentially relied on the expertise and software developed by its own members. Each contributor had its own focus, so that the present library only covers part of the whole field. Therefore the list of NICONET participants has been expanded to a representative European network and new research centers and universities with complementary expertise in software development in numerics of control have been selected which jointly cover almost the whole discipline of systems and control theory. In this way, the resulting joint library is potentially able to approach a "mature" status with respect to *size, completeness, and quality.* The present SLICOT subroutine collection is too restricted for broad industrial use and therefore NICONET will fill these gaps by starting to complete the library in industrially relevant areas for which the partners of NICONET have readily available prototype software and/or algorithms. These are basic numerical tools for control, model reduction, subspace identification, robust control, and nonlinear systems.

The future development of SLICOT will be accompanied by the development of a parallel version of SLICOT (with working title *PSLICOT*) for distributed memory computing environments. Parallelisation clearly opens new perspectives for solving large scale problems which frequently arise in industrial practice. As not every control problem requires the computing power of such machines, only those subroutines will be contained that can be used to solve control problems capable of taking advantage of distributed memory parallel computing environments. The set up of this parallel SLICOT version involves the extension of the SLICOT Standard [59] for high-performance computations on parallel architectures and the selection of standard communication kernels such as MPI [29] or PVM [19] and parallel numerical linear algebra libraries such as ScaLAPACK [8] or PLAPACK [38].

7.3 Integration of software in a user-friendly environment

The main aim of WGS is to see the library be used by as many scientists and engineers in industry as possible, so that the careful efforts of the contributors bear fruit. This requires a wider distribution, and in order to guarantee this, a better integration of SLICOT in a user friendly environment is needed. If the software is not easy to use it will not be used, certainly not in industry! Those environments are chosen which are most commonly used by European industrial companies and research centers, namely MATLAB [27], Scilab [11], and ANDECS [22]. Scilab has the advantage to be public domain. Very often industrial enterprises are not willing to pay additional library licences. These may be very expensive (especially for industry). For those companies Scilab can be a useful tool. A first step in this direction is the use of a compiler that automatically passes the function parameters from the CACSD environment, such as Matlab, to any Fortran routine of SLICOT and back (e.g., the already mentioned NAG-Ware Gateway Generator), which clearly makes the routines of a Fortran library available to a broader group of users. Therefore, NICONET plans the development of Matlab toolboxes based on calls of SLICOT routines by means of this tool.

7.4 Benchmarking and testing the software in an industrial environment

There is a definite need for more and adequate benchmarks for control methods and their implementation [18]. These benchmarks should be practically oriented. Carefully chosen benchmarks give insight in the state of the art with respect to the performance of methods in the language of the control system analyst or control engineer; see, e.g., [5, 6, 7] and Section 4.5. The need for this kind of insight is rapidly increasing due to today's widespread availability of a wealth of methods and implementations. Therefore, a SLICOT benchmark library and accompanying standards will be set up and made available through the WGS ftp site. To assess the performance, reliability, and versatility of SLICOT in industry, industrial enterprises specialised in the development of industrial software for implementation in production technology have been selected which jointly cover a broad spectrum of industrial applications, such as aerospace and automotive technology, robotics, and manufacturing. By integrating the new tools, provided through NICONET, into their products, these partners would be able to improve traditional production processes and make the much needed software available to industry. In addition, this facilitates largely future implementations of advanced solutions into production systems.

7.5 Information dissemination and access to control software

In order to facilitate dissemination of the SLICOT software and its transfer to a wide range of users, WGS, DLR and NAG decided to make SLICOT freely available. The use of electronic means is most appropriate to ensure easy and worldwide access to the library and therefore the software, together with the documentation, has been made available on the WGS ftp site (see Appendix B for details). It will also be available on CD-Rom or tape. Because of using the highly standardized Fortran 77 as programming language, SLICOT can be used under all major operating systems and platforms. Also, its manual should be available in paper form. These services, as well as commercial support, will be provided by NAG. As pointed out in the results of the NICONET questionnaire, this is of major importance for a lot of industrial users who want better services for maintenance of the product, advice for troubleshooting and software guidance.

Information exchange will be realized by the issue of an electronic NICONET newsletter. These issues, containing information on NICONET, SLICOT, and other CACSD software, are made available on the ftp and web sites and will be announced via appropriate electronic newsletters (such as the NA-NET News Digest[8] and the E-letter on Systems, Control, and Signal Processing[9]). The establishment of these electronic services, together with the set up and maintenance of electronic mail reflectors, the NICONET world wide web pages, the electronic repository of NICONET related reports, are major assets for the partners of NICONET and provide ready information exchange, up-to-date sources of information and software, and a means to publish and make available important software and results.

In addition, workshops around the topic of control algorithms and software, as well as tutorials and training courses, will be organized.

At present, NICONET consists of 17 European partners and a new proposal for the final implementation of the network with a detailed workplan over the next 4 years has been submitted to the European Communities. The achievement of the above aims and activities will depend on the available funding.

8 Concluding Remarks

We have presented developments around the Subroutine Library in Control Theory SLICOT. This library, maintained and developed in the last decade by the Working Group on Software WGS, provides implementations of the basic mathematical tools required for computations in control and systems theory. It relies now on the widely used linear algebra packages BLAS and LAPACK and on this basis has recently been turned into public domain software. Using the NAGWare Gateway Generator, it is possible to embed SLICOT routines into MATLAB. The so produced MATLAB function often outperform the available MATLAB functions from the MATLAB toolboxes or even built-in functions.

The future development is connected to the European "Numerics in Control" network NICONET. Its aim is to improve and complete SLICOT in such a way that it can be applied to a wide range of industrial applications of control and systems theory. This will be accompanied by the development of special routines for large-scale problems frequently encountered in practice and by embedding SLICOT in user-friendly environments such as MATLAB, Scilab, and ANDECS.

[8]For information about NA-NET (editor: C. Moler): mail to "na.help@na-net.ornl.gov".

[9]For information about this E-letter (editors: A. Stoorvogel and S. Weiland): send an (empty) e-mail message to "eletter@win.tue.nl" carrying the subject "info".

Acknowledgments

This paper presents research results of the European Community IMT Thematic Networks Programme (project BRRT-CT96-0038), the Belgian Programme on Interuniversity Poles of Attraction (IUAP Phase IV/2 & 24), initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture, and of a Concerted Research Action (GOA) project of the Flemish Community, entitled "Model-based Information Processing Systems".

S. Van Huffel is a Research Associate with the F.W.O. (Fund for Scientific Research – Flanders).

P. Benner, V. Mehrmann, and A. Varga were supported by the *Deutsche Forschungsgemeinschaft*, Research Grants Me 790/7-1 and Me 790/7-2.

9 References

[1] G.S. Ammar, P. Benner, and V. Mehrmann. A multishift algorithm for the numerical solution of algebraic Riccati equations. *Electr. Trans. Num. Anal.*, 1:33–48, 1993.

[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*. SIAM, Philadelphia, PA, second edition, 1994.

[3] W.F. Arnold, III and A.J. Laub. Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proc. IEEE*, 72:1746–1754, 1984.

[4] P. Benner. *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*. Dissertation, Fakultät für Mathematik, Technische Universität Chemnitz–Zwickau, D–09107 Chemnitz, Germany, February 1997.

[5] P. Benner, A. Laub, and V. Mehrmann. A collection of benchmark examples for the numerical solution of algebraic Riccati equations I: Continuous-time case. Technical Report SPC 95_22, Fak. f. Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1995. Available from `http://www.tu-chemnitz.de/sfb393/spc95pr.html`.

[6] P. Benner, A. Laub, and V. Mehrmann. A collection of benchmark examples for the numerical solution of algebraic Riccati equations II: Discrete-time case. Technical Report SPC 95_23, Fak. f. Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1995. Available from `http://www.tu-chemnitz.de/sfb393/spc95pr.html`.

[7] P. Benner, A. J. Laub, and V. Mehrmann. Benchmarks for the numerical solution of algebraic Riccati equations. *IEEE Control Systems Magazine*, 1997. To appear.

[8] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley. *ScaLAPACK Users' Guide*. SIAM, Philadelphia, PA, 1997.

[9] The Boeing Company, Seattle, WA. *EASY5 User's Guide*, 1996.

[10] F. E. Cellier, P. O. Grepper, D. F. Ruffer, and J. Toedtli. Educational aspects of development and application of a subprogram package for control. In *Prepr. IFAC Symposium on Trends in Automatic Control Education, Barcelona, Spain*, pages 151–159, 1977.

[11] F. Delebecque and S. Steer. *Integrated Scientific Computing with Scilab*. Birkhäuser, Boston, MA, 1997.

[12] M. J. Denham. A software library and interactive design environment for computer-aided control system design. In M. Jamshidi and C. J. Herget, editors, *Computer-aided Control Systems Engineering*. North Holland, Amsterdam, 1985.

[13] M. J. Denham and C. J. Benson. Implementation and documentation standards for the software library in control engineering (SLICE). Technical Report 81/3, Kingston Polytechnic, Control Systems Research Group, Kingston, United Kingdom, 1981.

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

[15] J.J. Dongarra, J. Du Croz, I.S. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.*, 16:1–17, 1990.

[16] J.J. Dongarra, J. Du Croz, S. Hammarling, and R.J. Hanson. An extended set of FORTRAN Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.*, 14:1–17, 1988.

[17] H. Elmqvist, A. Tysso, and J. Wieslander. Scandinavian control library. Programming. Technical report, Dept. of Aut. Control, Lund Inst. of Technology, Lund, Sweden, 1976.

[18] D. K. Frederick. Benchmark problems for computer aided control system design. In *Proc. 4th IFAC Symposium on Computer-Aided Control Systems Design*, pages 1–6, Bejing, China, 1988.

[19] A. Geist, A. Beguelin, J. J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing.* The MIT Press, 1994.

[20] G. Grübel. Die regelungstechnische Programmbibliothek RASP. *Regelungstechnik*, 31:75–81, 1983.

[21] G. Grübel and H.-D. Joos. RASP and RSYST - two complementary program libraries for concurrent control engineering. In *Prepr. 5th IFAC/IMACS Symp. CADCS'91, Swansea, UK*, pages 101–106. Pergamon Press, Oxford, 1991.

[22] G. Grübel, H.-D. Joos, M. Otter, and R. Finsterwalder. The ANDECS design environment for control engineering. In *Prepr. of 12th IFAC World Congress, Sydney, Australia*, 1993.

[23] G. Grübel, A. Varga, A. J. W. van den Boom, and A. J. Geurts. Towards a coordinated development of numerical CACSD software: the RASP/SLICOT compatibility concept. In *Proc. CACSD'94 Symposium*, pages 499–504, Tucson, AZ, 1994.

[24] Integrated Systems, Inc., Santa Clara, CA. *Xmath Basics, Version 5.0*, 1997.

[25] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for FORTRAN usage. *ACM Trans. Math. Software*, 5:303–323, 1979.

[26] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, MA 01760. *Control System Toolbox User's Guide*, 1996.

[27] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, MA 01760. *Using* MATLAB, 1996.

[28] P. Misra, P. Van Dooren, and A. Varga. Computation of structural invariants of generalized state-space systems. *Automatica*, 30:1921–1936, 1994.

[29] P. Pacheco. *Parallel Programming with MPI.* Morgan Kaufmann Publishers Inc., 1997.

[30] P. Hr. Petkov, N. D. Christov, and M. M. Konstantinov. SYSLAB: an interactive system for analysis and design of linear multivariable systems. In *Prepr. 3th IFAC/IFIP Int. Symposium on computer aided design in control and engineering systems (CADCE '85), Copenhagen, Denmark*, pages 140–145. Pergamon Press, Oxford, July 31-August 2 1985.

[31] V. Sima. Algorithms and LAPACK-Based software for subspace identification. In *Proc. CACSD'96 Symposium, Dearborn, MI*, pages 182–187, 1996.

[32] V. Sima. *Algorithms for Linear-Quadratic Optimization*, volume 200 of *Pure and Applied Mathematics: A Series of Monographs and Textbooks.* Marcel Dekker, Inc., New York, 1996.

[33] V. Sima. High-performance numerical software for control systems, and subspace-based system identification. Technical Report WGS-report 97-2, The Working Group on Software: WGS, 1997.

[34] 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, New York, second edition, 1976.

[35] The Numerical Algorithms Group, Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, U.K. *NAG SLICOT Library Manual, Release 1*, 1991.

[36] The Numerical Algorithms Group, Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, U.K. *NAG SLICOT Library Manual, Release 2*, 1993. Updates Release 1 of May 1990.

[37] The Numerical Algorithms Group, Wilkinson House, Jordan Hill Road, Oxford OX2 8DR, UK. *NAGWare Gateway Generator, Release 2.0*, 1994.

[38] R. A. van de Geijn. *Using PLAPACK: Parallel Linear Algebra Package.* The MIT Press, Cambridge, MA, 1997.

[39] A. J. W. van den Boom, A. Brown, F. Dumortier, A. Geurts, S. Hammarling, R. Kool, M. Vanbegin, P. Van Dooren, and S. Van Huffel. SLICOT, a subroutine library in control and systems theory. In *Proc. of the 5th IFAC Symposium on Computer Aided Design in Control Systems, Swansea, UK*, pages 89–94, Oxford, UK, July 1991. Pergamon Press.

[40] A. J. W. van den Boom and S. Van Huffel. Developments around the freeware standard control library SLICOT. In *Proc. CACSD'96 Symposium*, pages 473–476, Dearborn, MI, 1996.

[41] P. Van Dooren. A generalized eigenvalue approach for solving Riccati equations. *SIAM J. Sci. Statist. Comput.*, 2:121–135, 1981.

[42] S. Van Huffel and A. J. W. van den Boom. NICONET: network for performant numerical software development in control engineering. In *Proc. 7th IFAC Symposium on Computer-Aided Control Systems Design*, pages 197–202, Ghent, Belgium, April 28–30 1997.

[43] S. Van Huffel and J. Vandewalle. *The total least squares problem: computational aspects and analysis*. SIAM, Philadelphia, PA, 1991.

[44] P. Van Overschee and B. De Moor. N4SID: Two subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30:75–93, 1994.

[45] A. Varga. BIMASC - general description. Technical Report ICI, TR-10.83, Central Inst. for Management and Informatics, Bucharest, Romania, 1983.

[46] A. Varga. Numerical methods and software tools for model reduction. In I. Troch and F. Breitenecker, editors, *Proc. of 1st MATHMOD Conf., Vienna*, volume 2, pages 226–230, 1994.

[47] A. Varga. Optimal output feedback control: a multi-model approach. In *Proc. CACSD'96 Symposium, Dearborn, MI*, pages 327–332, 1996.

[48] A. Varga. Periodic Lyapunov equations: some applications and new algorithms. *Int. J. Control*, 67:69–87, 1997.

[49] A. Varga and A. Davidoviciu. BIMASC - a package of fortran subprograms for analysis, modelling, design and simulation of control systems. In *Prepr. 3th IFAC/IFIP Int. Symposium on computer aided design in control and engineering systems (CADCE '85), Copenhagen, Denmark*, pages 151–156. Pergamon Press, Oxford, 1985.

[50] A. Varga and S. Pieters. A computational approach for optimal periodic output feedback control. Technical Report TR R199-96, DLR-Oberpfaffenhofen, Institute for Robotics and System Dynamics, March 1996. (submitted for publication).

[51] A. Varga and V. Sima. BIMAS - general description. Technical Report ICI, TR-03.82, Central Inst. for Management and Informatics, Bucharest, Romania, 1982.

[52] A. Varga and V. Sima. BIMAS - a basic mathematical package for computer aided systems analysis and design. In *Prepr. 9th IFAC World Congress, Budapest, Hungary*, volume 8, pages 202–207, 1985.

[53] M. Verhaegen. Identification of the deterministic part of MIMO state space models given in innovations form from input-output data. *Automatica*, 30:61–74, 1994.

[54] J. Wieslander. Scandinavian control library. A subroutine library in the field of automatic control. Technical report, Dept. of Aut. Control, Lund Inst. of Technology, Lund, Sweden, 1977.

[55] The Working Group on Software: WGS. *Implementation and Documentation Standards for the basic subroutine library SYCOT Systems and Control Tools. Towards a computer aided control systems design package*, 1983. WGS-report 83-1.

[56] The Working Group on Software: WGS. *An inventory of basic software for computer aided control system design*, 1985. WGS-report 85-1.

[57] The Working Group on Software: WGS. *SLICOT Implementation and Documentation Standards*, 1990. WGS-report 90-1.

[58] The Working Group on Software: WGS. *SLICOT Contributor's Kit 2.1*, 1994. WGS-report 96-2.

[59] The Working Group on Software: WGS. *SLICOT Implementation and Documentation Standards 2.1*, 1996. WGS-report 96-1.

[60] The Working Group on Software: WGS. *Results of the NICONET Questionnaire*, 1997. WGS-report 97-1.

A Contents of SLICOT Release 3.0

Table 15 lists the user-callable SLICOT routines, briefly stating their function. The routine names have been changed (usually only the fifth letter) when upgrading from Release 2.0.

TABLE 15: Contents of SLICOT: user-callable routines.

Routine	Function
AB01MD	Orthogonal controllability form for single-input system.
AB01ND	Orthogonal controllability staircase form for multi-input system.
AB01OD	Staircase form for multi-input system.
AB04MD	Discrete-time $\leftrightarrow$ continuous-time conversion by bilinear transformation.
AB05MD	Cascade interconnection of two systems in state-space form.
AB05ND	Feedback interconnection of two systems in state-space form.
AB05OD	Rowwise concatenation of two systems in state-space form.
AB05PD	Parallel interconnection of two systems in state-space form.
AB05QD	Appending two systems in state-space form.
AB05RD	Closed-loop system for a mixed output and state feedback control law.
AB05SD	Closed-loop system for an output feedback control law.

TABLE 15: (continued)

Routine	Function
AB06MD	Minimal block Hessenberg realization for a state-space representation.
AB07MD	Dual of a given state-space representation.
AB08MD	Invariant zeros of a multivariable system given by a state-space representation.
AB08ND	Construction of a regular pencil corresponding to a multivariable system given by a state-space representation such that the invariant zeros of the system are the generalized eigenvalues of the pencil.
DE01OD	Convolution or deconvolution of two signals.
DF01MD	Sine transform or cosine transform of a real signal.
DG01MD	Discrete Fourier transform of a complex signal.
DG01ND	Discrete Fourier transform of a real signal.
DK01MD	Anti-aliasing window applied to a real signal.
FB01QD	One iteration of the time-varying square root covariance filter (dense matrices).
FB01RD	One iteration of the time-invariant square root covariance filter (Hessenberg form).
FB01SD	One iteration of the time-varying square root information filter (dense matrices).
FB01TD	One iteration of the time-invariant square root information filter (Hessenberg form).
FB01VD	One iteration of the conventional Kalman filter.
MB01PD	Matrix scaling.
MB01RD	Computation of matrix expression $\alpha R + \beta \mathtt{op}(A) X \mathtt{op}(A)^T$, where R and X are symmetric matrices, and $\mathtt{op}(A) = A$, or $\mathtt{op}(A) = A^T$.
MB02MD	Solution of the Total Least Squares (TLS) problem using a Singular Value Decomposition (SVD) approach.
MB02ND	Solution of the Total Least Squares (TLS) problem using a Partial Singular Value Decomposition (PSVD) approach.
MB02OD	Solution of $\mathtt{op}(A)X = \alpha B$, or $X \mathtt{op}(A) = \alpha B$, where A is triangular (with condition estimation).
MB02RZ	Solution of a linear system with complex upper Hessenberg matrix.
MB02SZ	LU factorization of a complex upper Hessenberg matrix.
MB02TZ	Condition number of a complex upper Hessenberg matrix.

TABLE 15: (continued)

Routine	Function
MB03MD	Computation of an upper bound θ using a bisection method such that a bidiagonal matrix has precisely ℓ singular values greater than or equal to θ plus a given tolerance.
MB03ND	Computation of the number of singular values of a bidiagonal matrix which are smaller than or equal to a given value θ.
MB03OD	Matrix rank determination by incremental condition estimation.
MB04ID	QR factorization of a matrix with a lower left zero triangle.
MB04JD	LQ factorization of a matrix with an upper right zero triangle.
MB04KD	QR factorization of a special structured block matrix.
MB04LD	LQ factorization of a special structured block matrix.
MB04SD	Reducing a rectangular matrix to column echelon form by unitary row permutations and column transformations.
MB04TD	Computation of orthogonal transformations Q and Z such that the transformed pencil $Q^T(sE-A)Z$ is in upper block triangular form, where E is an $m \times n$ matrix in column echelon form and A is an $m \times n$ matrix.
MB04XD	Computation of a basis for the left and/or right singular subspace of a matrix corresponding to its smallest singular values.
MB04YD	Partial diagonalization of an upper bidiagonal matrix using QR or QL iterations so that it is split into unreduced bidiagonal submatrices whose singular values are either all larger than a given bound or are all smaller than (or equal to) this bound.
MB05MD	Computation of $\exp(A\delta)$ where A is a real $n \times n$ non-defective matrix and δ is a real scalar using an eigenvalue/eigenvector decomposition.
MB05ND	Computation of matrix exponential and integral using Padé approximation.
MB05OD	Computation of $\exp(A\delta)$ where A is a real $n \times n$ matrix and δ is a real scalar using Padé approximation. The minimal number of accurate digits in the 1-norm of $\exp(A\delta)$, including its value at 95 % confidence level, are also returned.
MC01MD	Computation, for a real given polynomial $P(x)$ and a real scalar α, of the leading k coefficients of the shifted polynomial $P(x) = \sum_{i=1}^{k} q_i(x-\alpha)^{i-1}$ using Horner's algorithm.

TABLE 15: (continued)

Routine	Function
MC01ND	Computation of the value of the real polynomial $P(x)$ at a given complex point $x = x_0$ using Horner's algorithm.
MC01OD	Computation of the coefficients of a complex polynomial from its zeros.
MC01PD	Computation of the coefficients of a real polynomial from its zeros.
MC01QD	Computation of the quotient polynomial $Q(x)$ and the remainder polynomial $R(x)$ of $A(x)$ divided by $B(x)$ for two given real polynomials $A(x)$ and $B(x)$.
MC01RD	Computation of the coefficients of the polynomial $P(x) = P_1(x)P_2(x)+\alpha P_3(x)$, where $P_1(x)$, $P_2(x)$ and $P_3(x)$ are real polynomials and α is a real scalar.
MC01SD	Scaling the coefficients of the real polynomial $P(x)$ such that the coefficients of the scaled polynomial $Q(x) = sP(tx)$ have minimal variation, where s and t are real scalars.
MC01TD	Checking stability of a given polynomial $P(x)$ with real coefficients, either in the discrete-time or continuous-time case.
MC01VD	Roots of a quadratic equation with real coefficients.
MC01WD	Computation of the quotient polynomial $Q(x)$ and the linear remainder polynomial $R(x)$ for a given real polynomial $P(x)$ and a quadratic polynomial $B(x)$.
MC03MD	Computation of the coefficients of the real polynomial matrix $P(x) = P_1(x)P_2(x) + \alpha P_3(x)$, where $P_1(x)$, $P_2(x)$ and $P_3(x)$ are given real polynomial matrices and α is a real scalar.
MC03ND	Computation of the coefficients of a minimal polynomial basis $K(s)$ for the right nullspace of the polynomial matrix $P(s)$ (solution of the polynomial matrix equation $P(s)K(s) = 0$).
SB01MD	Pole assignment for a linear time-invariant single-input system.
SB02MD	Solution of the continuous-time algebraic Riccati equation $A^T X + XA + C - XDX = 0$ using Laub's real Schur vector method.
SB02ND	Optimal feedback matrix F for a standard continuous-time or discrete-time optimal control problem.

TABLE 15: (continued)

Routine	Function
SB02OD	Solution of either the continuous-time algebraic Riccati equation $Q+A^TX+XA-XBR^{-1}B^TX=0$ or the discrete-time algebraic Riccati equation $X=A^TXA-A^TXB(R+B^TXB)^{-1}B^TXA+Q$ using the method of deflating subspaces, where $Q=C^TC$, $R=D^TD$ and $C^TD=0$.
SB03MD	Solution of either the continuous-time Lyapunov equation $A^TX+XA=C$ or the discrete-time Lyapunov equation $A^TXA-X=C$ using Bartels/Stewart or Barraud's methods, respectively.
SB03OD	Solution (for $X=U^TU$) of either the stable non-negative definite continuous-time Lyapunov equation $A^TX+XA=-B^TB$ or the convergent non-negative definite discrete-time Lyapunov equation $A^TXA-X=-B^TB$ where A is a square matrix whose eigenvalues have negative real parts or lie inside the unit circle, respectively, and U is an upper triangular matrix, using a variant of the Bartels/Stewart method.
SB04MD	Solution of the continuous-time Sylvester equation $AX+XB=C$ where A, B, and C are general matrices.
SB04ND	Solution of the continuous-time Sylvester equation $AX+XB=C$, with at least one of the matrices A or B in Schur form and the other in Hessenberg or Schur form (both either upper or lower).
SB04OD	Solution (for R and L) of the generalized Sylvester equation $AR-LB=C$, $DR-LE=F$ where A and D are $m\times m$ matrices, B and E are $n\times n$ matrices and C, F, R and L are $m\times n$ matrices.
SB06ND	Minimum norm feedback matrix for "deadbeat control" of a state-space representation.
SB08MD	Spectral factorization of a real polynomial $A(s)$ arising from continuous optimality problems, i.e., computation of the real polynomial $E(s)$ such that $E(-s)E(s)=A(-s)A(s)$ and $E(s)$ is stable.
SB08ND	Spectral factorization of a real polynomial $A(z)$ arising from discrete optimality problems, i.e., computation of the real polynomial $E(z)$ such that $E(1/z)E(z)=A(1/z)A(z)$ and $E(z)$ is stable in the discrete-time sense.
SB09MD	Comparison of two multivariable sequences $M_1(k)$ and $M_2(k)$ for $k=1,2,\ldots,n$, and evaluation of their closeness.
TB01MD	Upper/lower controller Hessenberg form.

TABLE 15: (continued)

Routine	Function
TB01ND	Upper/lower observer Hessenberg form.
TB01QD	Transfer matrix of a state-space representation.
TB01RD	Frequency response matrix of a state-space representation.
TB01SD	Left/right polynomial matrix representation of a state-space representation.
TB01TD	Balancing state-space representation by permutations and scalings.
TC01MD	Transfer matrix of a left/right polynomial matrix representation.
TC01ND	State-space representation for left/right polynomial matrix representation.
TC01OD	Dual of a left/right polynomial matrix representation.
TD01MD	Evaluation of a transfer function for a specified frequency.
TD01ND	Left/right polynomial matrix representation for a proper transfer matrix.
TD01OD	Minimal state-space representation for a proper transfer matrix.
TF01MD	Output response of a linear discrete-time system.
TF01ND	Output response of a linear discrete-time system (Hessenberg matrix).
TF01OD	Block Hankel expansion of a multivariable parameter sequence.
TF01PD	Block Toeplitz expansion of a multivariable parameter sequence.
TF01QD	Markov parameters of a system from transfer function matrix.
TF01RD	Markov parameters of a system from state-space representation.
UD01MD	Printing an $m \times n$ real matrix A row by row.

The subroutines corresponding to AB08ND, DF01MD, FB01QD, FB01RD, FB01SD, FB01TD, FB01VD, MC01MD, MC01ND, MC01OD, MC01PD, MC01QD, MC01RD, MC01SD, MC01TD, MC01VD, MC01WD, MC03MD, MC03ND, SB08MD, and SB08ND have been added in Release 2 of the library. Moreover, AB05PD, AB05QD, AB05RD, AB05SD, MB01PD, MB01RD, MB04ID, MB04JD, MB04KD, and MB04LD have been added in the latest release. Releases 1 and 2 also contained several routines performing basic linear algebra computations, which are now available in the BLAS and LAPACK packages; these routines are not listed in Table 15.

B Electronic Access to the Library and Related Literature

The SLICOT routines can be downloaded from the WGS ftp site,

`wgs.esat.kuleuven.ac.be`

(directory `pub/WGS/SLICOT/` and its subdirectories) in compressed (gzipped) tar files. On line `.html` documentation files are also provided there. It is possible to browse through the documentation on the WGS homepage at the World Wide Web URL

`http://www.win.tue.nl/wgs/`

after linking from there to the SLICOT web page and clicking on the `FTP site` link in the freeware SLICOT section. The SLICOT index is also operational. Each functional "module" can be copied to the user's current directory, by clicking on an appropriate location in the `.html` image. A "module" is a compressed (gzipped) tar file, which includes the following files: source code for the main routine and its test program, test data, execution results, the associated `.html` file, as well as the source code for the called SLICOT routines. This involves duplicating some routines, but it can be convenient for a user needing only a single function. There is also a file, called `slicot.tar.gz`, in the directory `/pub/WGS/SLICOT/`, which contains the entire library. The tree structure created after applying `gzip -d slicot.tar` and `tar xvf slicot.tar` is:

```
./slicot/       - for routine source files;
./slicot/doc/   - for html files;
./slicot/tests/ - for test programs/data/results files.
```

Some of the references cited in this paper and the recent WGS reports can be downloaded as compressed postscript files from the World Wide Web URL

`http://www.win.tue.nl/wgs/reports.html`

These are the following references:

- WGS reports 96-1 [59], 96-2 [58], 97-1 [60], 97-2 [33].
- proceedings papers [23, 31, 40, 42, 46].