Towards a coordinated development of numerical CACSD software: the RASP/SLICOT compatibility concept

G. Grübel
and
A. Varga

DLR - Oberpfaffenhofen
Control Design Engineering
Institute for Robotics and System Dynamics
D-82230 Wessling, Germany
df03@master.df.op.dlr.de

A. van den Boom[†]
and
A. J. Geurts[‡]

Eindhoven University of Technology
[†]Department of Electrical Engineering
[‡]Department of Mathematics & Comp. Science
NL-5600 MB Eindhoven, The Netherlands
ersicavdb@er.ele.tue.nl

Abstract

A mutual compatibility concept is proposed for the further developments of RASP and SLICOT control libraries. The adoption of this concept will permit a coordinated development of both libraries leading to a reduction of software implementation and testing efforts. A first development along the new mutual compatibility concept is the recently developed model reduction library RASP-MODRED.

Keywords: Computer-aided design, control libraries, numerical software, model reduction.

1 Introduction

The development of efficient, reliable, and portable numerical software requires joining expertise in the application domain, 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.

Portable numerical software written in Fortran for computer-aided control systems design (CACSD) is provided by several subroutine libraries as for example RASP [5], SLICOT [2], BIMAS [18] and BIMASC [16], and LISPACK [8]. These libraries share many common features as for instance rigorous implementation along well established programming and documentation standards, numerical robustness of software implementations, similarities in the covered topics, and use of high performance algorithms. On the other hand they differ to some extent with respect to their main goals, organization, user interface, and size. For the purpose of this paper we restrict our attention to two of these libraries, RASP and SLICOT, which are apparently to date the only ones in active further developing.

The RASP routines cover a broad area of control engineering computations supporting frequency- and time-domain analysis and synthesis techniques, multi-criteria parameter optimization, simulation, and graphics [5]. RASP together with the engineering data-base and operating system RSYST [5] form the software infrastructure of the computer aided control engineering (CACE) environment ANDECS[1] (Analysis & Design of Controlled Systems) [6]. The organization of the library and the standardization of user interface of routines reflect the orientation of RASP towards control engineering applications. The numerical software supporting RASP are the linear algebra libraries BLAS (*Level 1*), LINPACK, EISPACK (and recently LAPACK [1]) as well as some libraries providing facilities for simulation and optimization of nonlinear systems (ODEPACK, MINPACK). The graphics in RASP is supported by the GKS and PHIGS libraries. The latest version RASP'92 consists of about 320 user callable routines (not counting those from the above mentioned standard libraries).

SLICOT can be primarily viewed as a mathematical library for control theoretical computations [2]. The library provides tools to perform many basic

[1] ANDECS® is a registered trademark of DLR

system analysis and synthesis tasks. A special emphasis in SLICOT is on providing maximum algorithmic flexibility to users, and on the use of rigorous implementation and documentation standards [19]. SLICOT is a product of the Benelux Working Group on Software (WGS) realized in cooperation with the Numerical Algorithms Group (NAG). The supporting numerical libraries for SLICOT are BLAS (*Level 1,2,3*) and similar additional NAG routines as well as some LINPACK and EISPACK equivalent routines from the NAG library [7]. The latest Release 2 of SLICOT contains about 90 user callable routines and Release 3 is under development.

The effort to develop both the SLICOT and the RASP libraries was very high. Taking into account that both libraries will continue to evolve, there are serious concerns to rationalize future developing efforts. In particular, it should be avoided to duplicate software pieces for which robust implementations are already available. In this paper we describe the new mutual compatibility concept adopted for the further development of the RASP library, and exemplify this by its latest developments. The main aim is to ensure full compatibility between the *newly* developed RASP programs and SLICOT. The basis for such a developing strategy is the exclusive use of LAPACK as the supporting library for linear algebra computations. In this way the new RASP routines are simultaneously usable for both libraries. SLICOT will use LAPACK as a supporting library, rather than the NAG library, starting with Release 3. Thus the new RASP routines can be included without any modifications in the new releases of SLICOT. It is expected that in the future also routines of SLICOT from Releases 1,2 will be based on LAPACK and thus be accessible for use without other supporting libraries.

The adoption of this developing strategy led us to the idea of a coordinated future development of RASP and SLICOT libraries in order to reduce the implementation efforts by avoiding work duplications. A first step in this direction is the development of a library for model reduction called RASP-MODRED which fulfills the requirements for mutual compatibility. Details on the mutual compatibility concept and on the contents of MODRED are given later in the paper. MODRED contains 57 new routines and is intended to fill important hitherto empty chapters in both RASP and SLICOT libraries. Another potential candidate for extending SLICOT is a smaller package called RASP-DESCRIPT for the analysis and modeling of descriptor systems [12]. DESCRIPT contains about 40 routines. A part of these RASP routines could be easily modified in order to satisfy the mutual compatibility requirement.

2 RASP/SLICOT compatibility issues

There exist many potential sources for incompatibilities between numerical libraries for CACSD. Strict incompatibilities between two portable libraries which prevent their simultaneous usage arise only if two subroutines which belong to different libraries have identical names, or if `COMMON` blocks with the same names are used by the routines of both libraries. Usually the removing of such incompatibilities is difficult. The need to rename a routine or a `COMMON` block can yield an unexpected chain of modifications in all other routines calling the renamed routine or accessing the renamed `COMMON` block. Fortunately, because RASP and SLICOT use different naming conventions (see [4] and [19]), and since both the RASP and SLICOT computational routines do not use `COMMON` blocks, there exist no such conflicts between the two libraries.

RASP and SLICOT use different storage schemes of matrices. In RASP the emphasis on an easy integration of routines in an interactive CACE environment as ANDECS led to the adoption of memory saving solutions and of simpler parameter lists for user callable subroutines. The matrices are stored compactly (one-dimensional storage) and the information on leading dimensions are not necessary in the parameter lists. In contrast, SLICOT uses a conventional two-dimensional storage of matrices. This has the obvious advantage of an easier handling of operations on submatrices. However, the routines have longer parameter lists because of the need to transfer information about the leading dimensions of arrays. The different storage modes do not rise incompatibility problems from the side of RASP.

In the future, the development of both libraries should explicitly pay attention to external compatibility issues related to the storage mode of matrices in Fortran when the routines are intended to be called by interactive environments written in C. A seemingly satisfactory solution to C – Fortran interfacing is offered by the public-domain CFORTRAN tools-kit of the CERN Program Library[2]. CFORTRAN provides the necessary tools to create easy-to-use and machine independent interfaces between C and Fortran routines and global data. It is proposed that both libraries rely on this interfacing solution.

[2] available by anonymous ftp at: zebra.desy.de [131.169.2.244].

Another aspect which causes differences in using the routines of the two libraries is the employment of different error handling methods. In SLICOT practically no error handling is provided and this task has to be done explicitly by the user on the basis of returned error flags. RASP offers a more advanced error handling feature by emulating the ADA exception handling concept for error messaging. This type of error handling permits to automatically report errors bottom up through the entire software hierarchy, to yield a defined program stop or to continue the program execution as in case of an error-free run. This approach is once again along the lines of promoting an easy integration of routines in interactive software environments.

A somewhat different compatibility aspect between the two libraries is due to the use of different supporting linear algebra software. As mentioned before, SLICOT partly resides on a reduced set of NAG library routines. Employed NAG routines originating from standard libraries as LINPACK and EISPACK are renamed according to NAG's naming conventions. Routines from BLAS, LINPACK and EISPACK are also called in RASP but with their original names (if they were not modified). An unpleasant aspect in using the two control libraries simultaneously is the unnecessary but unfortunately unavoidable code duplication, which arises by including the same routines twice under different names.

The further development of the interactive CACE environment ANDECS will continue to rely on RASP. But routines from SLICOT could be used also in that environment with appropriate interfacing. Hence we intend to ensure that the users can benefit from the advantages offered by the next releases of both libraries. In order to achieve this goal we promote a mutual compatibility concept for implementing new RASP routines as detailed in the next section.

3 The RASP/SLICOT mutual compatibility concept

The achievement of a mutual compatibility for the newly developed RASP routines is of mutual advantage for both libraries RASP and SLICOT. The main advantage which we see is the possibility of a coordinated development which yields a sharing of efforts as well as results by both sides. For the SLICOT side an additional advantage is a guarantee for substantial renewment and extension of this library. For the RASP side the main advantage lies in the possibility to freely use the SLICOT routines in forthcoming software developments. Another advantage is of equal value and resides on the use of the well disciplined implementation standards of SLICOT [19]. A coordinated development of both libraries also has the important side effect of a supplementary qualified testing of all new subroutines in both libraries.

In practice, the mutual compatibility concept means:

- *The top level user callable RASP routines are implemented according to RASP programming and documentation standards.*
  This means that in accordance with the actual RASP standards the new top level routines basically operate on one-dimensional stored matrices, the error handling is performed by using the facilities of the RASP error handling package, and the list of parameters is ordered according to the RASP conventions. The decision to further maintain the compact storage at this level is along the lines to ensure an easy integration of routines in an interactive environment like ANDECS. The user callable RASP routines are documented according to the RASP documentation standard [4].

- *Each top level user callable RASP routine calls a functionally equivalent SLICOT driver routine. All called lower level routines are on their turn fully SLICOT compatible, that is implemented according to SLICOT standards. Direct calls to SLICOT routines are also possible.*
  In this way the full flexibility of SLICOT is taken care of by the use of the two-dimensional storage of matrices and by the error handling based on error-flags. On the other hand, the user-callable RASP routines augment functionally equivalent SLICOT routines by a more advanced 'bottom-up' error handling procedure which is necessary for an engineering-efficient use in integrated CACE environments like ANDECS. This purpose is also served by the storage saving array handling of the RASP routines. The documentation for all SLICOT compatible routines is provided by in-line comments. These comments also serve for the elaboration of the SLICOT documentation according to the existent standards [19].

- *The supporting linear algebra libraries for future RASP and SLICOT implementations are BLAS and LAPACK.*

LAPACK is now a *de-facto* standard for linear algebra computations and it is expected that in the future LAPACK will completely replace the LINPACK and EISPACK libraries. LAPACK covers most of the linear algebra computational problems appearing in CACSD problems with a rich set of driver, expert, and computation routines. It should be pointed out here that for the robust implementation of future numerical CACSD software many low level LAPACK routines are also very useful. The use of BLAS (*Level 1,2,3*) routines on which the implementation of LAPACK is actually based will be also strongly encouraged.

Adopting this approach rises of course some delicate questions. For example, some of the existing RASP or SLICOT routines could become obsolete or should be rewritten. But in the long range this approach seems to be the only one which can guarantee affordable developing efforts. An example of using the mutual compatibility concept in newly developed RASP software is dealt with in the next section.

4 The model reduction library RASP-MODRED

The model reduction subroutines library RASP-MODRED was implemented along the lines of the mutual compatibility concept described in the previous section. MODRED is among the first libraries developed by using the new linear algebra standard library LAPACK and certainly the first library written in Fortran which provides a rich set of computational facilities for model reduction. The subroutines of MODRED cover order reduction of both continuous-time and discrete-time systems. (Note that the existing MATLAB toolboxes basically provide only tools for continuous-time systems.) All algorithms implemented in MODRED are numerically reliable and computationally efficient and represent the latest developments in the field of numerical methods for model reduction.

The reduction of stable systems can be performed by using several alternative methodologies related to balancing techniques: balance & truncate, singular perturbation approximation, Hankel-norm approximation, balanced stochastic truncation. The core model reduction routines are based on recently developed *square-root* and *balancing-free* accuracy enhancing algorithms [10], [9], [17]. For the reduction of unstable systems the available tools for the reduction of stable system can be used in conjunction with the coprime factor model reduction technique or the additive spectral decomposition approach. Several new algorithms for computing stable coprime factorizations of transfer-function matrices are implemented in MODRED [11], [15]. For performing frequency-weighted model or controller reductions, tools are provided to compute efficiently and in a numerically reliable way the necessary stable projections [14]. Additional tools are available for computing Hankel and L^2 norms of transfer-function matrices [13].

In its present state of development MODRED consists of 77 routines from which 20 are user callable RASP routines, 20 are the functionally equivalent SLICOT compatible driver routines, and the rest are various computational routines. About 90 various LAPACK and BLAS routines are called by the routines of MODRED.

5 Conclusions

It is possible to combine apparently different libraries if certain compatibility conditions are fulfilled. For the control libraries RASP and SLICOT the mutual compatibility concept is introduced which enables the user to share a large collection of numerically reliable routines. This concept also is expected to lead to a substantial saving of time and energy for the further realization and development of control libraries.

The further development of the RASP package will be based on the RASP/SLICOT mutual compatibility concept described in this paper. Priority areas for further developments of both RASP and SLICOT are: H_∞- and μ-synthesis, signal processing, systems identification [3], descriptor systems [12], closed-loop controller reduction. We hope that a common future development of both libraries can be achieved by joining the skills and efforts of many contributors in a coordinated software development cooperation. Only such a scheme can guarantee the continuous renewment and extension of libraries.

References

[1] E. Anderson, Z. Bai, J. Bishop, J. Demmel, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, 1992.

[2] A. van den Boom, A. Brown, 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 *Prepr. 5th IFAC/IMACS Symp. CADCS'91, Swansea, UK*, pp. 89–94. Pergamon Press, Oxford, 1991.

[3] A. van den Boom and M. Verhaegen. Design of an identification chapter for a control library in a CACSD environment. In *Proc. 2-nd Intern. Conf. Automation, Robotics and Computer Vision, ICARV'92, Singapore*, vol. 3, pp. INV 10.4.1–5, 1992.

[4] G. Grübel and H.-D. Joos. The control systems engineering numerical subroutine library RASP. Technical Report TR R14–90, DLR - German Aerospace Research Establishment, D-82230 Wessling, 1990.

[5] 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*, pp. 101–106. Pergamon Press, Oxford, 1991.

[6] 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.

[7] NAG. The NAG Fortran Library Manual. Technical report, NAG Ltd., Oxford, UK, 1990.

[8] P. Hr. Petkov, N. D. Christov, and M. M. Konstantinov. SYSLAB: An interactive system for analysis and design of linear multivariable systems. In *Proc. of 3-rd IFAC/IFIP Symp. CADCE'85, Copenhagen, Denmark*, pp. 167–171. Pergamon Press, Oxford, 1986.

[9] A. Varga. Balancing-free square-root algorithm for computing singular perturbation approximations. In *Proc. of 30th IEEE CDC, Brighton, UK*, pp. 1062–1065, 1991.

[10] A. Varga. Efficient minimal realization procedure based on balancing. In A. El Moudni, P. Borne, and S. G. Tzafestas, editors, *Proc. of IMACS/IFAC Symp. on Modelling and Control of Technological Systems, Lille, France*, vol. 2, pp. 42–47, 1991.

[11] A. Varga. Coprime factors model reduction based on square-root balancing-free techniques. In A. Sydow, editor, *Computational System Analysis 1992, Proc. 4-th Int. Symp. Systems Analysis and Simulation, Berlin, Germany*, pp. 91–96. Elsevier, Amsterdam, 1992.

[12] A. Varga. Numerical algorithms and software tools for analysis and modelling of descriptor systems. In *Prepr. of 2nd IFAC Workshop on System Structure and Control, Prague, Czechoslovakia*, pp. 392–395, 1992.

[13] A. Varga. On computing 2-norms of transfer function matrices. In *Proc. 1992 American Control Conference, Chicago, Illinois*. Elsevier, Amsterdam, 1992.

[14] A. Varga. Explicit formulas for an efficient implementation of the frequency-weighted model reduction approach. In *Proc. 1993 European Control Conference, Groningen, NL*, pp. 693–696, 1993.

[15] A. Varga. A Schur method for computing coprime factorizations with inner denominators and applications in model reduction. In *Proc. 1993 American Control Conference, San Francisco, CA*, pp. 2130–2131, 1993.

[16] A. Varga and A.Davidoviciu. BIMASC - A package of Fortran subprograms for analysis, modelling, design and simulation of control systems. In *Proc. of 3-rd IFAC/IFIP Symp. CADCE'85, Copenhagen, Denmark*. Pergamon Press, Oxford, 1986.

[17] A. Varga and K. H. Fasol. A new square-root balancing-free stochastic truncation model reduction algorithm. In *Prepr. of 12th IFAC World Congress, Sydney, Australia*, 1993.

[18] A. Varga and V. Sima. BIMAS - A basic mathematical package for computer aided systems analysis and design. In J. Gerter and L. Keviczky, editors, *Proc. of 9-th IFAC World Congress, Budapest, Hungary*, 1985.

[19] WGS. Implementation and Documentation Standards. Report 90-1, Working Group on Software, 1990.