Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations

WILLIAM F. ARNOLD, III, MEMBER, IEEE, AND ALAN J. LAUB, SENIOR MEMBER, IEEE

Invited Paper

Numerical issues related to the computational solution of the algebraic matrix Riccati equation are discussed. The approach presented uses the generalized eigenproblem formulation for the solution of general forms of algebraic Riccati equations arising in both continuous- and discrete-time applications. These general forms result from control and filtering problems for systems in generalized (or implicit or descriptor) state space form. A Newton-type iterative refinement procedure for the generalized Riccati solution is given. The issue of numerical condition of the Riccati problem is addressed. Balancing to improve numerical condition is discussed. An overview of a software package, RICPACK, coded in portable, reliable Fortran is given. Results of numerical experiments are reported.

I. INTRODUCTION

One of the most deeply studied nonlinear matrix equations arising in mathematics and engineering is the Riccati equation. The generic term "Riccati equation" can mean any of a class of matrix "quadratic" algebraic or differential or difference equations of symmetric or nonsymmetric type arising in the study of continuous-time or discrete-time dynamical systems. In this paper we shall discuss algorithms and software for certain classes of algebraic Riccati equations. The resulting software package, called RICPACK, can be used as a module in a larger computer-aided control system design package or environment.

Riccati equations arise naturally in a rich variety of situations and their role and use in systems and control theory, in particular, has been well-established over the past 25 years. A representative but by no means exhaustive sample of such applications can be found in standard "classical" textbooks on optimal control (see [1]–[5] and the references therein) and filtering and prediction (see [3], [5]–[8] and the references therein). One of the finest mathematical treatises on Riccati equations is the book of Reid [9] which, in addition to control and estimation applications, discusses applications to partial differential equations, multiple uniform and nonuniform transmission lines, the Mycielski–Paszkowski diffusion problem, and neutron transport theory. The transport problem is an illustration of the intimate role played by Riccati equations in the method of invariant embedding (see [10], [11] and the references therein). Further details on the use of Riccati equations in solving two-point boundary value problems are discussed in [12] and various papers in [13], particularly those of Denman, Bramley, and Casti.

One aspect of Riccati equations that has always been significant, and which has received increasing attention over the past 5 years, is effective algorithms for their reliable numerical solution in the finite arithmetic environment of a digital computer. Reliable implementation of a numerical algorithm involves attention to at least the following three concerns:

1) the condition of the underlying problem,
2) the numerical stability of the algorithm being used to solve the problem,
3) the robustness of the actual software implementation.

Each of these will be illustrated further in the sequel in the context of solving algebraic Riccati equations.

Reliable numerical algorithms and software now exist for the solution of linear equations, singular value decomposition, linear least squares, and both standard and generalized eigenvalue problems [14]–[16]. This has not heretofore been the case for the solution of algebraic Riccati equations. This paper will outline, in a tutorial fashion, a class of algorithms which are quite generally reliable and are readily extendable to a variety of related problems. A Fortran software package, RICPACK, that implements the preferred algorithmic approach by building on and emulating [14]–[16] is described in detail.

Briefly, the rest of this paper is organized as follows. Section II contains general background information on Schur-type techniques for the solution of various types of algebraic Riccati equations. Section III gives algorithmic details for generalized eigenvalue methods for Riccati equation solution. Iterative refinement techniques are discussed in Section IV. General comments on mathematical software for Riccati equations along with details concerning the software package RICPACK are presented in Section V, and Section VI gives numerical results for example problems. Some concluding remarks are made in Section VII.

Manuscript received April 2, 1984. This work was supported in part by the U.S. Army Research Office under Contract DAAG29-81-K-0131.

W. F. Arnold, III is with the Naval Weapons Center, China Lake, CA 03555, USA.

A. J. Laub is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, USA.

0018-9219/84/1200-1746$01.00

II. Background

In this section we provide some background material on Schur-type techniques for the solution of various types of algebraic Riccati equations by means of certain associated generalized eigenvalue/eigenvector problems. The generalized eigenproblem framework provides a unifying methodology which facilitates the reliable numerical solution of very general classes of Riccati equations arising in optimal control or filtering problems (and elsewhere) including those with "nonstandard" features such as singular control weighting (or measurement noise covariance) matrices, cross-weighting (cross-correlation) matrices, and singular transition matrices (discrete-time). Generalized state space models can also be considered and give rise to "generalized" Riccati equations [12].

The basic algebraic Riccati equation (ARE) arising in continuous-time problems takes the form

$$A^TXE + E^TXA - (E^TXB + S)R^{-1}(B^TXE + S^T) + C^TQC = 0 \tag{1a}$$

or

$$\hat{A}^TXE + E^TX\hat{A} - E^TXBR^{-1}B^TXE + C^TQC - SR^{-1}S^T = 0 \tag{1b}$$

where

$$\hat{A} := A - BR^{-1}S^T.$$

This ARE arises in connection with finding a feedback control $u(t) = Kx(t)$ which solves the problem

$$\text{Min} \int_0^{+\infty} \frac{1}{2}\left[y^TQy + 2x^TSu + u^TRu\right] dt$$

$$\text{subject to:} \quad E\dot{x} = Ax + Bu, \qquad x \in \mathbf{R}^n,\ u \in \mathbf{R}^m$$

$$y = Cx, \qquad y \in \mathbf{R}^p.$$

Under mild technical conditions on the matrices, the minimizing control which is stabilizing (the generalized eigenvalues of $\lambda E - (A + BK)$ have negative real parts) is given by $K = -R^{-1}(B^TXE + S^T)$ where X is the unique nonnegative definite solution of (1). We are assuming E is nonsingular. This situation arises, for example, from first-order formulations of second-order models of the form

$$M\ddot{\xi} + D\dot{\xi} + K\xi = \bar{B}u$$

in which $M = M^T > 0$ but M^{-1} is not to be used—largely for numerical reasons—but possibly also to preserve and perhaps exploit the structure.

The more familiar form of (1) occurs when $S = 0$, $E = I$, and $C = I$

$$A^TX + XA - XBR^{-1}B^TX + Q = 0 \tag{2}$$

(or even $F^TX + XF - XGR^{-1}G^TX + H = 0$, according to taste). Equation (2) can be solved by finding an orthogonal matrix U such that

$$U^TMU = S$$

where

$$U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}, \qquad U_{ij} \in \mathbf{R}^{n\times n}$$

$$M = \begin{pmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{pmatrix}, \qquad \text{a Hamiltonian matrix}$$

$$\left(M^TJ = -MJ, \qquad \text{where } J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}\right)$$

and $S = \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix}$, a quasi-upper-triangular matrix with all eigenvalues of S_{11} in the strict left-half-plane. The n "Schur vectors" comprising $\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}$ span the stable invariant subspace and the solution of (2) is given by $X = U_{21}U_{11}^{-1}$. This so-called Schur method, which is described in detail in [17] and [18] has proved to be a rather reliable, general-purpose method for solving (2) for modest sized problems (say $n \lesssim 100$) where there is no usefully exploitable structure in the coefficient matrices.

Many other solution techniques exist for (2). Most of these methods fall into one of the following broad categories:

a) methods based on certain special canonical forms
b) doubling and other direct integration (of the associated differential equation) techniques
c) Newton's method
d) parameter embedding methods
e) Chandrasekhar-type algorithms
f) methods based on use of the matrix sign function
g) spectral factorization techniques
h) "square root" formulations.

Each of these methods has interesting features but most do not have satisfactory numerical behavior in finite arithmetic. Moreover, many of the methods do not extend easily to the solution of (1) nor to certain related nonsymmetric Riccati equations.

For the solution of (1) we consider the $2n \times 2n$ matrix pencil

$$\lambda\begin{pmatrix} E & 0 \\ 0 & E^T \end{pmatrix} - \begin{pmatrix} A - BR^{-1}S^T & -BR^{-1}B^T \\ SR^{-1}S^T - C^TQC & -(A - BR^{-1}S^T)^T \end{pmatrix}. \tag{3}$$

The resulting generalized eigenvalue problem is then transformed by orthogonal matrices V and U to the form

$$V(\lambda L - M)U = \lambda\hat{L} - \hat{M} \tag{4}$$

where $\hat{L}$ is upper triangular and $\hat{M}$ is quasi-upper-triangular and the stable generalized eigenvalues are determined by the upper left $n \times n$ blocks. The Schur vectors $\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}$ span the stable deflating subspace and the feedback K is given by $K = -R^{-1}(B^TU_{21}U_{11}^{-1} + S^T)$. The solution of (1) is given by $X = W_{21}W_{11}^{-1}$ where $W = \begin{pmatrix} E & 0 \\ 0 & I \end{pmatrix}U$; see [12], [19].

Although the matrix R above is often diagonal or even the identity which makes R^{-1} quite trivial to determine, it may instead be nondiagonal and ill-conditioned with respect to inversion, or possibly even singular, in which case the $(2n + m) \times (2n + m)$ extended pencil

$$\lambda\begin{pmatrix} E & 0 & 0 \\ 0 & E^T & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} A & 0 & B \\ -C^TQC & -A^T & -S \\ S^T & B^T & R \end{pmatrix} \tag{5}$$

can be used. This pencil can be reduced or compressed to an equivalent $2n \times 2n$ pencil [20] whereupon a procedure similar to that described above is followed.

Of course, all the above has an analog for the discrete-time counterpart of the linear-quadratic problem

described above. In this case, the ARE corresponding to (1) takes the form

$$E^TXE = A^TXA - (A^TXB + S)(B^TXB + R)^{-1} \cdot (A^TXB + S)^T + C^TQC \tag{6}$$

or

$$E^TXE = \hat{A}^TX\hat{A} - \hat{A}^TXB(B^TXB + R)^{-1}B^TX\hat{A} + C^TQC - SR^{-1}S^T \tag{7}$$

where $\hat{A} := A - BR^{-1}S^T$. Equation (6) arises naturally in stochastic realization problems while form (7) arises naturally in LQG (linear-quadratic-Gaussian) problems.

In this case, the analog of (3) is the pencil

$$\lambda\begin{pmatrix} E & BR^{-1}B^T \\ 0 & (A - BR^{-1}S^T)^T \end{pmatrix} - \begin{pmatrix} A - BR^{-1}S^T & 0 \\ SR^{-1}S^T - C^TQC & E^T \end{pmatrix} =: \lambda L - M \tag{8}$$

while if R^{-1} is to be avoided (singular R being not uncommon in discrete-time systems) the analog of (5) is the extended pencil

$$\lambda\begin{pmatrix} E & 0 & 0 \\ 0 & A^T & 0 \\ 0 & -B^T & 0 \end{pmatrix} - \begin{pmatrix} A & 0 & B \\ -C^TQC & E^T & -S \\ S^T & 0 & R \end{pmatrix}. \tag{9}$$

The solution procedure involving (8) or a compressed version of (9) is essentially the same as that outlined above for the continuous-time case. In the special case, $S = 0$, $E = I$, and $C = I$, (8) takes the form

$$\lambda\begin{pmatrix} I & BR^{-1}B^T \\ 0 & A^T \end{pmatrix} - \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}$$

which was suggested in [18] and used in [21]–[24] to avoid the need for A^{-1} in the eigenproblem for the symplectic matrix (if A^{-1} exists)

$$\begin{pmatrix} I & BR^{-1}B^T \\ 0 & A^T \end{pmatrix}^{-1}\begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}. \tag{10}$$

A similar approach was discussed in [25].

III. Algorithmic Details

The main algorithmic issues associated with the Schur-type generalized eigenproblem method are the compression of the matrix pencil (5) or (9), the solution of the generalized eigenproblem (4), and the "ordering" of the generalized eigenvalues so that the stable generalized eigenvalues are contained in the upper left $n \times n$ blocks of $\lambda\hat{L} - \hat{M}$. Some details concerning these issues are given in this section. The related issues of numerical condition of the solution process and balancing to improve the condition are also discussed.

Should R in (5) or (9) be singular or nearly singular (and not easily invertible) then the following compression procedure due to Van Dooren [20] can be employed. Determine an orthogonal matrix $P \in \mathbf{R}^{(2n+m)\times(2n+m)}$ such that

$$\begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix}\begin{pmatrix} B \\ -S \\ R \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \bar{R} \end{pmatrix} \tag{11}$$

where $\bar{R} \in \mathbf{R}^{m\times m}$ and is nonsingular. P can be formed from a series of Householder transformations. The remainder of the compression technique will be illustrated using the pencil corresponding to the continuous-time problem (5). The discrete-time problem can be handled analogously as illustrated in [23]. Applying P to the pencil (5) we see that the "infinite generalized eigenvalues" are "deflated out" so that we need only work with the $2n \times 2n$ pencil

$$\lambda P_{11}\begin{pmatrix} E & 0 \\ 0 & E^T \end{pmatrix} - \left(P_{11}\begin{pmatrix} A & 0 \\ -C^TQC & -A^T \end{pmatrix} + P_{12}\begin{pmatrix} S^T & B^T \end{pmatrix}\right). \tag{12}$$

It can be shown [20] that the pencils (12) and (5) are equivalent. Therefore, the pencil (12) can be used for the Riccati solution and does not involve the explicit inversion of R. This extended pencil approach is used in [19] and [20] and is implicit in the work of Campbell and others (summarized in [26], [27]).

One must essentially solve the appropriate generalized eigenproblem to transform the pencil to quasi-upper-triangular form as indicated in (4). Basic references for the numerical solution of the fundamental (unordered) generalized eigenvalue problem include [28] and [29]. The algorithm is a generalization of the QR algorithm used for the standard eigenvalue problem and is referred to as the QZ algorithm. Application of the QZ algorithm to the appropriate pencil yields the transformed matrices $\hat{L}$ and $\hat{M}$ as well as the orthogonal transformation U of (4). Unfortunately, the QZ algorithm does not result in $\hat{L}$ and $\hat{M}$ with the desired ordering of the generalized eigenvalues.

However, the eigenvalues can subsequently be reordered to any desired order. Algorithmic and numerical details of the "reordering problem" for the generalized eigenproblem were given in [20]. The reordering is accomplished by orthogonal transformations and the method is proved to be numerically stable.

An important aspect of the analysis of any numerical problem is its condition. That is, if the data of the problem are perturbed slightly is the resulting change in the solution "large" (an ill-conditioned problem) or "small" (a well-conditioned problem)? Understanding the conditioning of Riccati equations is, of course, crucial to their reliable numerical solution, but Riccati equation condition is apparently an exceedingly complex problem. This is borne out by both analysis and empirical study in [30]. Several proposed "condition numbers" for the Riccati problem are compared and all are shown to have deficiencies for some classes of problems. Among those compared are:

a) The condition of U_{11} or W_{11} with respect to inversion (see Section II) [12], [17] since the final Riccati solution derives from the solution of a linear system of the form $XU_{11} = U_{21}$ or $XW_{11} = W_{21}$. It is known that U_{11} or W_{11} are singular if the underlying model is unstabilizable and/or if E is singular. Thus near unstabilizability or near singularity of E can be expected to cause ill-conditioning, which it does. Unfortunately, a Riccati equation can still be ill-conditioned with a well-conditioned U_{11} or W_{11}.

b) A condition number in terms of the singular values of the singular value decomposition of the orthogonal U or W found in the reduction of the generalized eigenproblem. This result was developed in [31] and extended in [19], but turns out to be essentially the same as the one discussed in a); see [30].

c) Various condition numbers based on a first-order perturbation analysis of a Riccati equation. One such derivation is given in [32] while still others are developed in [30]. Examples can be shown (see [30]) where such analyses are inadequate in the sense of signaling ill-conditioning falsely or of not detecting ill-conditioning for a known ill-conditioned equation. Further pertinent discussions, results, and conjectures can be found in [17], and [30]–[32].

The coefficient matrices of many Riccati equations arising in practice contain numbers of widely different magnitudes. It is thus reasonable to expect that some sort of scaling and/or balancing strategy could be employed to improve the accuracy of a numerical solution of a Riccati equation in finite arithmetic. This expectation is, in fact, borne out in numerical experiments reported in [32] and [33]. The main technique employed in those experiments involved the balancing of the generalized eigenvalue problem prior to its solution. Since the Schur-type solution technique for Riccati equations involves solution of the generalized eigenproblem as a fundamental step, this balancing technique seems reasonable.

Ward [34] has developed a balancing algorithm designed especially for QZ numerical solution methods for generalized eigenproblems. The procedure consists of permutations and two-sided diagonal transformations to attempt to scale L and M in (4) so that their elements have magnitudes as close to unity as possible. This balancing strategy is relatively inexpensive, but can improve accuracy substantially and also increase the reliability of the condition of U_{11} or W_{11} with respect to inversion as an indicator of condition of the Riccati problem.

Another potential balancing scheme was investigated in [30]. There, as a preprocessing step before the Riccati solution, the underlying system matrices (A, B, C, etc.) are "balanced" in the system-theoretic sense [35], [36]. This is an attempt to give equal weight to both controllability and observability, but has the rather severe disadvantages of requiring open-loop stability of $A - \lambda E$ and of being relatively expensive computationally. Other state coordinate systems may offer computational advantages. A convenient algorithm for applying the associated change of coordinate transformation in the Schur-type Riccati solution method is given in [30].

IV. Iterative Refinement Techniques

Numerical implementation of the Schur-type solution methods of Section II is relatively straightforward. The proven stable algorithms discussed in Section III are coded into reliable Fortran software (see Section V). However, the Riccati problem may be ill-conditioned, and the resulting numerical solution may not be as accurate as desired. This section presents an iterative refinement procedure utilizing Newton's method. This procedure is presented for both the continuous- and discrete-time problems. The continuous-time method is based on Kleinman [37], and the discrete-time method is based on Hewer [38]. Other iterative solution methods are discussed briefly.

If X_k, $k = 0, 1, \cdots$ is the unique nonnegative definite solution of the linear algebraic equation

$$0 = (A - BK_k)^T X_k E + E^T X_k (A - BK_k) + C^T QC + K_k^T RK_k - SK_k - (SK_k)^T \quad (13)$$

where, recursively,

$$K_k = R^{-1}\left(B^T X_{k-1} E + S^T\right), \qquad k = 1, 2, \cdots \quad (14)$$

and K_0 is chosen such that $\lambda E - (A - BK_0)$ has generalized eigenvalues with negative real parts (stable), then it can be shown [30] that

a) $0 \leqslant X \leqslant X_{k+1} \leqslant X_k \leqslant \cdots \leqslant X_0$
b) $\lim_{k \to \infty} X_k = X$
c) in the vicinity of X, $\|X_{k+1} - X\| \leqslant C_2 \|X_k - X\|^2$

where X solves (1) and C_2 is a finite constant. This iterative procedure features monotonic convergence to the nonnegative definite solution to (1) when it exists, as long as the starting value K_0 is a stabilizing state feedback matrix. The global convergence can be quite slow, however, and result in excessive computation time if a poor K_0 is chosen. Numerical experience (see [30] for examples) has shown that when this iterative method is applied to an initial solution value produced by the direct Schur-type method, the convergence is quadratic or faster. This hybrid solution method is quite useful for improving the Riccati solution accuracy in the presence of certain sources of problem ill-conditioning. For the case $E = I$ and $S = 0$, the above result becomes the familiar procedure established by Kleinman [37].

An analogous result exists for the discrete-time equations (6) or (7) with the analogs of (13) and (14) being [30]

$$E^T X_k E = (A - BK_k)^T X_k (A - BK_k) + K_k^T RK_k + C^T QC - SK_k - (SK_k)^T \quad (15)$$

and

$$K_k = \left(R + B^T X_{k-1} B\right)^{-1}\left(B^T X_{k-1} A + S^T\right), \qquad k = 1, 2, \cdots \quad (16)$$

with the requirement that K_0 is chosen such that $\lambda E - (A - BK_0)$ has generalized eigenvalues with magnitude less than unity. For the case $E = I$ and $S = 0$, this result is the same as that established by Hewer [38].

Other iterative solution methods such as those based on the matrix sign function are also potential candidates for hybrid-type methods for solving Riccati equations. Whatever iterative method is chosen, two basic advantages make them well worth including in a Riccati equation solution package:

a) Accuracy of computed solutions of ill-conditioned equations can be improved (even without computing residuals in extended precision).
b) For sufficiently small perturbations in the coefficient matrices, a solution of the perturbed equation can be found more efficiently by a few steps of an iterative method rather than redoing a full-ordered generalized eigenproblem.

V. Numerical Software for Riccati Equations

Even for problems which are apparently "simple" from a mathematical point of view, a vast myriad of little details must be attended to in order to enable a robust or reliable algorithmic implementation in finite arithmetic. These details can become so overwhelming that the only effective means of successfully communicating an algorithm is

through its embodiment as mathematical software. Mathematical or numerical software simply means an implementation on a computing machine of an algorithm for solving a mathematical problem. Ideally, such software must be reliable, portable, and unaffected by the machine or system environment in which it is used.

There are many characteristics that can be listed to characterize "good" or robust mathematical software. State-of-the-art discussions are to be found in [39]–[51]. Of course, one of the key features is portability. Inevitably, numerical algorithms are strengthened when their mathematical software is made portable since their widespread use is greatly facilitated. Furthermore, reliable and portable software is usually faster than poorer codes. Portability and speed largely account for the preponderance of good mathematical software being coded in Fortran although a strong case could now be made for Ada™, at least on portability grounds.

A careful examination of the modern mathematical software literature should lead one to the conclusion that serious evaluation of mathematical software is a highly nontrivial task. Clearly the quality of software is largely a function of its operational specifications. It must also reflect the numerical aspects of the algorithm being implemented. The language used and the compiler (e.g., optimizing or not) used for that language will both have an enormous impact on quality, both perceived and real, as will the underlying hardware and arithmetic. Different implementations of an algorithm can have markedly different properties and behavior, even of the same good underlying algorithm.

Many aspects of systems, control, and estimation theory are ready for the research and design that is necessary to produce reliable, portable mathematical software that performs successfully in finite arithmetic. Certainly many of the underlying linear algebra tools (for example, EISPACK [14], [15] and LINPACK [16]) are considered sufficiently reliable as to be used as black—or at least gray—boxes by control engineers. Much of the theory and methodology used in the production of prototypical mathematical software such as EISPACK and LINPACK can and has been carried over to problems such as the solution of Riccati equations. However, much of the work done in engineering, particularly the design and synthesis aspects, is not amenable to nice, "clean" algorithms and the ultimate software must have the capability to enable a dialog between the computing machine and the engineer (or scientist), but with the latter probably still making the final engineering decisions. Most applications software will not ultimately look like EISPACK or LINPACK. To even attempt that would be futile. Instead, a better analogy would be to emulate a good ordinary differential equation or partial differential equation package.

Schur-type methods for the generalized algebraic Riccati equations discussed in Section II have been implemented by the authors utilizing the algorithms of Section III with the Newton iterative refinement procedure of Section IV in a Fortran software package, RICPACK [33]. RICPACK was developed as a research tool to aid in the study of the numerical conditioning of algebraic Riccati equations. The package consists of modularly designed Fortran subroutines (approximately 60) together with a Fortran driving program for use in an interactive "terminal"-type environment. The driver prompts for all necessary input and convenient input default options exist not only for ease of data input, but also for exploitation by the subroutines to reduce the number and complexity of the computations. These options are also designed to speed the input of more "standard" type problems.

Highlights of RICPACK capabilities include:

a) Choice of calculation of the stabilizing (nonnegative definite), antistabilizing (nonpositive definite), or just any (possibly) indefinite solution to a generalized ARE (see Section II).
b) Coordinate or system balancing of the system model [35], [36].
c) Ward's balancing [34] of the generalized eigenproblem (prior to the QZ transformation).
d) Direct handling of singular control weighting or singular measurement noise covariance by compression of the expanded pencils (5) or (9).
e) Direct handling of cross weighting or noise correlation, i.e., $S \neq 0$.
f) Provision for robustness recovery procedure, i.e., to replace the C^TQC term with $Q + \Upsilon C^TC$ and iterate on Υ, the driver program need only modify one block of the matrix pencil at each iteration.
g) Iterative refinement (or new solutions for small parameter perturbations) by Newton's method and Sylvester equations, i.e., iteratively solving equations of the form

$$\text{(Continuous)} \qquad \bar{A}_k^T X_{k+1} E + E^T X_{k+1} \bar{A}_k + \bar{C}_k = 0$$

$$\text{(Discrete)} \qquad E^T X_{k+1} E - \bar{A}_k^T X_{k+1} \bar{A}_k + \bar{C}_k = 0$$

for X which is optionally either

1) the new solution at each step or
2) the required change to the solution at each step.

h) Model unstabilizability detection as indicated by the condition of U_{11} or W_{11} with respect to inversion (see discussion after (4)); U_{11} or W_{11} is singular for an unstabilizable model.
i) Calculation of unique stabilizing solution for stabilizable models with undetectable modes.
j) Residual calculation of the form

$$r = \frac{\|\text{Residual}\|_1}{\|X\|_1}. \tag{17}$$

k) Condition estimates for the Riccati problem which provide information on the following sources of ill-conditioning: model unstabilizability; small separation of closed-loop spectrum, near singularity of R (continuous-time case).

The authors chose to implement the algorithms as high-quality mathematical software because to do less would be to evade a major responsibility, for to leave software implementation to the algorithm user/reader has the potential to lead to disastrously bad "versions" of a perfectly good algorithm. This software will be usable as a module or "tool" in a wide variety of Computer-Aided Control System Design (CACSD) environments ranging from small packages developed by individuals to large commercial packages such as MATRIX$_X$ (Integrated Systems, Inc.) or CTRL-C (Systems Control Technology, Inc.).

Fig. 1 illustrates the hierarchy of the software routines. That is, the routines on the upper levels employ the routines of the lower levels. At the lowest level we have the basic matrix manipulation routines like add, subtract, multiply,

™Ada is a trademark of the U.S. Department of Defense.

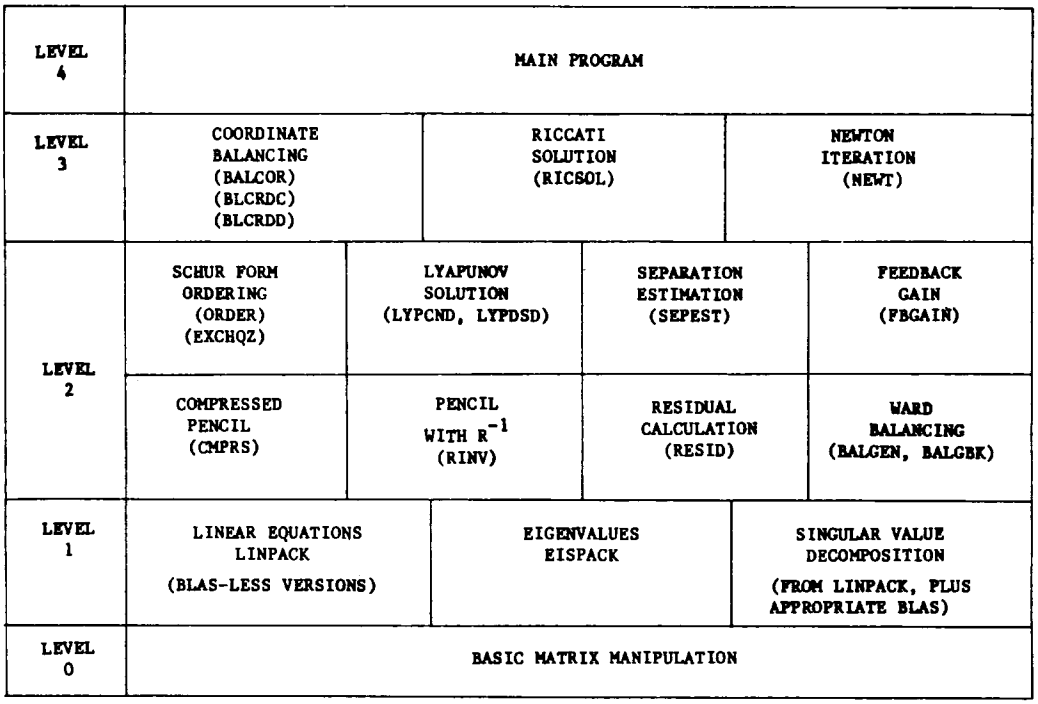

Fig. 1. Hierarchy of subroutines in software package RICKPACK.

etc., and some simple combinations of these basic operations. The next level consists of standard routines for linear equations, eigenvalues, and singular value decomposition (SVD). Most routines in this level are from LINPACK or EISPACK, or are slight modifications to routines from LINPACK and EISPACK. Subroutines that are modified have the modifications noted in the comment documentation included in the subroutine. A list of the Level 0 and 1 routines is given in Table 1. The SVD routine is listed separately with the BLAS routines that it requires from LINPACK, as it is the only routine to require the BLAS and could be modified to eliminate said BLAS.

Subroutines at Levels 2 and 3 perform more specialized tasks. The task title is given in Fig. 1, and the main subroutines performing the task are shown in parentheses.

Of course, the main driver program is at the highest level, Level 4. This main program would be rewritten, or at least extensively modified for most applications. This driver was written as a research tool and as such performs calculations not relevant to many analysis and design applications. A higher level language would be more appropriate to interface the Level 0 through Level 3 subroutines with a larger CACSD package and perform the necessary input and output.

Table 1 Subroutine List for Levels 0 and 1

Level 0	Level 1		DSVDC
BCORBK	BALANC	MLINEQ	DAXPY
D1NRM	DGECOM	ORTHES	DDOT
MADD	DGEFAM	ORTRAN	DNRM2
MMUL	DGESLM	QZHESW	DROT
MOUT	DSTSLV	QZITW	DROTG
MQF	ELMHES	QZVAL	DSCAL
MQFA	GIV	REBAKB	DSWAP
MQFWO	GRADBK	REDUCE	
MSCALE	GRADEQ	REDUC2	
MSUB	HQR	ROTC	
MULA	HQRORT	ROTR	
MULB	IMTQL2	SCALBK	
MULWOA	LINEQ	SCALEG	
MULWOB		SYMSLV	
PERMUT		TRED2	
SAVE			
SEQUIV			
TRNATA			
TRNATB			

VI. Numerical Examples

In this section we give a few examples to illustrate various points discussed previously and to provide some numerical results for comparison with other approaches. We also note here that all of the examples given in [17], [22], and [23] were solved using RICPACK and the solutions obtained were as accurate as the published results.

The following simple continuous-time example is used to illustrate the numerical properties of RICPACK when stabilizability is the key factor.

Example 1:

$$\dot{x} = \begin{pmatrix} 1 & 0 \\ 0 & -2 \end{pmatrix} x + \begin{pmatrix} \epsilon \\ 0 \end{pmatrix} u$$

$$y = (1 \quad 1)\, x$$

$$\text{minimize} \int_0^\infty (y^T y + u^T u)\, dt.$$

This system is stabilizable for $\epsilon \neq 0$ and completely reconstructible. The applicable ARE is

$$A^T X + XA - XBB^T X + C^T C = 0.$$

The "true" solution for X can be hand-calculated for comparison purposes as

$$X = \begin{bmatrix} \dfrac{1+\sqrt{1+\epsilon^2}}{\epsilon^2} & \dfrac{1}{2+\sqrt{1+\epsilon^2}} \\ \dfrac{1}{2+\sqrt{1+\epsilon^2}} & \dfrac{1}{4} - \dfrac{\epsilon^2}{4(2+\sqrt{1+\epsilon^2})^2} \end{bmatrix} > 0.$$

Table 2 Numerical Results for Example 1, $\epsilon = 10^{-N}$

N	$\kappa(U_{11})$	r (17)	Acc*	Newton Iterations	r (17)	Acc*
0	10^{0}	10^{-18}	17	...	...	...
2	10^{4}	10^{-14}	14	...	...	...
4	10^{8}	10^{-10}	10	2	10^{-18}	17
6	10^{12}	10^{-8}	6	3	10^{-20}	17
8	10^{16}	10^{-2}	2	4	10^{-34}	17
9	10^{18}	10^{-1}	0	6	10^{-18}	17
10	10^{20}	10^{0}	0	...	...	...
The following data include Ward balancing effects						
0	10^{0}	10^{-18}	17	...	...	...
5	10^{7}	10^{-15}	15	...	...	...
10	10^{12}	10^{-9}	9	2	10^{-18}	17
11	10^{16}	10^{-7}	7	3	0	17
12	10^{17}	10^{-7}	7	3	0	17
13	10^{17}	10^{-6}	7	3	10^{-18}	17
14	singular	10^{1}	0	...	...	...

*Accuracy in correct significant digits.

Note that as $\epsilon \to 0$ the system approaches unstabilizability and the (1,1) element of X tends to infinity.

The solution to this problem was numerically computed on a DEC KL-10 (under TOPS-20) in double precision using RICPACK. The machine precision is near 10^{-18} in this case. The results of interest are summarized in Table 2. The only measure of condition included in the table is the condition of U_{11} with respect to inversion because other measures did not give an indication that the solution accuracy was degenerating as $\epsilon \to 0$. We note here that the data in Table 2 and in the succeeding tables that are expressed as a power of 10 are rounded to the nearest power of 10.

Some useful observations can be made on these data. One can see that for this example, $\kappa(U_{11})$ and the residual (r) are both good indicators of the numerical accuracy. Since machine precision is near 10^{-18}, one would expect about 17 correct digits for a well-conditioned problem (which is the case for $\epsilon = 1$). The data indicate that one digit of accuracy is lost for each power of 10 change in $\kappa(U_{11})$ and r. This is desirable behavior of a condition estimate. Note that Ward balancing improves the condition of U_{11} and reduces the value of r for the same value of ϵ. Ward balancing enables solution calculation for smaller values of ϵ, but the accuracy is not as smooth a function of $\kappa(U_{11})$. However, the residual is still a good indicator of accuracy. Note that in all cases with a reasonable starting guess a few iterations of Newton's method restores full accuracy. The generalized eigenvalue solution was used as a starting guess and was considered reasonable if $\kappa(U_{11}) <$ 1./(machine precision). When this condition was not satisfied, the Newton iteration failed to converge to the desired solution.

Table 2 illustrates that stabilizability of the model does indeed influence the numerical accuracy of the Riccati solution. Also, $\kappa(U_{11})$ and r are good indicators of solution accuracy as the model approaches unstabilizability. However, $\kappa(U_{11})$ may not be a good indicator in other situations as the following example will show.

Example 2:

$$\dot{x} = \begin{pmatrix} -\epsilon & 1 & 0 & 0 \\ -1 & -\epsilon & 0 & 0 \\ 0 & 0 & \epsilon & 1 \\ 0 & 0 & -1 & \epsilon \end{pmatrix} x + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} u$$

$$y = (1 \quad 1 \quad 1 \quad 1)\, x$$

$$\text{minimize} \int_0^\infty (y^T y + u^T u)\, dt.$$

The model is completely controllable and observable. The open-loop poles are at $\pm\epsilon \pm j$, and the applicable ARE is

$$A^T X + XA - XBB^T X + C^T C = 0.$$

The solution to this problem was numerically computed on a UNIVAC 1100/83 in double precision using RICPACK. The machine precision is near 10^{-18} in this case. The results of interest are summarized in Table 3. Although an exact

Table 3 Numerical Results for Example 2, $\epsilon = 10^{-N}$

N	CLP	$\kappa(U_{11})$	$\kappa A_c(X)$ (18)	$\kappa B(X)$ (19)	r (17)	Newton Iterations	r (17)
0	10^{0}	10^{1}	10^{0}	10^{2}	10^{-16}	1	10^{-16}
3	10^{-6}	10^{0}	10^{6}	10^{7}	10^{-12}	1	10^{-14}
5	10^{-10}	10^{0}	10^{10}	10^{11}	10^{-8}	1	10^{-12}
7	10^{-14}	10^{0}	10^{18}	10^{15}	10^{-8}	2	10^{-16}
8	10^{-16}	10^{0}	10^{16}	10^{17}	10^{-1}	2	10^{-16}
9	10^{-18}	10^{0}	10^{18}	10^{18}	10^{-1}	...	...

hand solution was not possible in this case, the behavior of the residual can be used to judge the solution accuracy. This example was designed to assess the effect of the separation of the closed-loop spectrum on solution accuracy and the ability of condition estimates to detect degrading accuracy. The column CLP in Table 3 indicates the position (real part) of the closed-loop pole nearest the imaginary axis in the complex plane.

One can see from this example that $\kappa(U_{11})$ provides no indication of loss of accuracy in the solution. However, in this example we employ two other measures of condition defined in [30] and [32], respectively, as

$$\kappa A_c(X) = \frac{\kappa(E)\kappa(E^T)\|C^T QC - SR^{-1}S^T\|}{\|X\|\|E\|\|E^T\|\text{SEP}\left[A_c E^{-1}, -\left(A_c E^{-1}\right)^T\right]} \quad (18)$$

where

$$A_c = (A - BR^{-1}S^T - BR^{-1}B^T XE)$$

and

$$\kappa B(X) = \frac{\|C^T QC\| + 2\|A\|\|X\| + \|BR^{-1}B^T\|\|X\|^2}{\|X\|\text{SEP}\left[A_c^T, -A_c\right]} \quad (19)$$

where

$$A_c = A - BR^{-1}B^T X.$$

Note that $\text{SEP}(F, G) = \inf_{\|P\|=1}\|PF - GP\|$, is a measure of the separation of the closed-loop spectrum in the previous definitions. For a detailed definition of SEP(F, G) and a discussion of its properties see Stewart [52], [53].

The results in Table 3 show that $\kappa A_c(X)$ and $\kappa B(X)$ correlate directly with the behavior of the residual, and thus, the solution accuracy. Ward balancing of the eigenproblem had no noticeable effect on solution accuracy for a given value of ϵ. Newton iterations did significantly improve solution accuracy, as measured by the residual, until the condition $\kappa A_c(X) = \kappa B(X) = 1./(\text{machine precision})$. At this point, the iterations failed to converge to the desired solution, as was the case in Example 1.

This example shows that separation of the closed-loop

spectrum does indeed influence the numerical accuracy of the Riccati solution and that the condition estimates in which separation is a factor provide good indicators of solution degeneracy.

The following example illustrates the effect of ill-conditioning of the R weighting matrix, with respect to inversion, on the numerical solution for the continuous-time case. Recall that ill-conditioning of R is not necessarily a problem in the discrete-time case since its inverse is not explicitly required.

Example 3:

$$\dot{x} = \begin{pmatrix} -0.1 & 0 \\ 0 & -0.02 \end{pmatrix} x + \begin{pmatrix} 0.1 & 0 \\ 0.001 & 0.01 \end{pmatrix} u$$

$$y = (10. \quad 100.)\, x$$

$$\text{minimize} \int_0^\infty \left(y^T y + u^T \begin{pmatrix} 1+\epsilon & 1 \\ 1 & 1 \end{pmatrix} u \right) dt.$$

The system is completely controllable and observable. As $\epsilon \to 0$, the R matrix approaches singularity. The applicable ARE is

$$A^T X + XA - XBR^{-1}B^T X + C^T C = 0.$$

The solution to this problem was numerically computed on a UNIVAC 1100/83 in double precision using RICPACK. The machine precision is near 10^{-18} in this case. The results of interest are summarized in Tables 4 and 5. Ward balancing was employed in the calculations for Table 4, and coordinate balancing [35], [36] was employed for Table 5. Results of calculations where no balancing was applied were nearly identical to those of Table 5 for coordinate balancing.

Table 4 Numerical Results for Example 3, Ward Balancing, $\epsilon = 10^{-N}$

N	$\kappa(R)$	$\kappa(U_{11})$	$\kappa A_c(X)$ (18)	$\kappa B(X)$ (19)	r (17)	Newton Iterations	r (17)
0	10^1	10^1	10^0	10^2	10^{-17}	1	10^{-17}
2	10^2	10^2	10^2	10^6	10^{-17}	1	10^{-17}
4	10^4	10^3	10^3	10^{10}	10^{-14}	2	10^{-14}
6	10^6	10^4	10^3	10^{11}	10^{-12}	10	10^{-12}
8	10^8	10^5	10^3	10^{13}	10^{-10}	10	10^{-10}
10	10^{10}	10^6	10^3	10^{15}	10^{-8}	10	10^{-8}
12	10^{12}	10^7	10^3	10^{17}	10^{-6}	10	10^{-8}
14	10^{14}	10^8	10^3	10^{19}	10^{-5}	10	10^{-5}
16	10^{16}	10^9	10^3	10^{21}	10^{-3}	10	10^{-3}

Table 5 Numerical Results for Example 3, System Balancing, $\epsilon = 10^{-N}$

N	$\kappa(R)$	$\kappa(U_{11})$	$\kappa A_c(X)$ (18)	$\kappa B(X)$ (19)	r (17)	Newton Iterations	r (17)
0	10^1	10^3	10^0	10^3	10^{-15}	2	10^{-17}
2	10^2	10^3	10^2	10^6	10^{-15}	2	10^{-17}
4	10^4	10^3	10^3	10^9	10^{-12}	7	10^{-14}
6	10^6	10^3	10^3	10^{11}	10^{-12}	10	10^{-12}
8	10^8	10^3	10^3	10^{13}	10^{-9}	10	10^{-11}
10	10^{10}	10^3	10^3	10^{15}	10^{-6}	10	10^{-9}
12	10^{12}	10^3	10^3	10^{17}	10^{-6}	10	10^{-6}
14	10^{14}	10^3	10^3	10^{19}	10^{-2}	10	10^{-4}
16	10^{16}	10^3	10^3	10^{21}	10^{-1}	10	10^{-2}

The data in Table 4 indicate that $\kappa(R)$ with respect to inversion accurately reflects the behavior of the residual. Also, $\kappa(U_{11})$ with respect to inversion is too optimistic in its estimation of the problem condition and $\kappa B(X)$ is too pessimistic. $\kappa A_c(X)$ provides no information in this case. A maximum of 10 Newton iterations was allowed, and where 10 appears in the table, convergence did not occur. The value for the residual in that case is the residual associated with the solution at the tenth iteration. One can see that the ill-conditioning of R begins to dominate the numerical accuracy when $\kappa(R) > 10^6$. Since R^{-1} is involved in the Newton iteration calculations, iterative improvement does not improve accuracy when $\kappa(R)$ dominates. The convergence criteria used for the Newton iterations do not recognize this fact and stop the iterations.

The data in Table 5 are essentially the same as those in Table 4 except for one important aspect. $\kappa(U_{11})$ with respect to inversion provides no information on the problem conditioning once $\kappa(R) \geq \kappa(U_{11})$. This may indicate that Ward balancing will cause other sources of problem ill-conditioning, beside unstabilizability of the model and singularity of E, to be reflected in $\kappa(U_{11})$.

The preceding examples illustrate that none of the potential measures of conditioning are reliable indicators by themselves. However, numerical experience to date has shown that in all cases at least one of the measures will detect the degeneracy of numerical accuracy as it occurs. These examples were of small order because the problem condition is not directly related to size, so simple examples were sufficient to illustrate the numerical effects of an ill-conditioned problem. However, one may wonder if the size of the problem will affect the numerical results, even for well-conditioned problems. The following example shows that size is not a factor.

Example 4:

It is difficult to construct examples of large order for which the exact solution is known, or whose condition can be inferred. For this reason, two examples were taken from the literature. The first is a 64th-order example involving circulant matrices that appeared in [17] as example 5. The example was solved using RICPACK on a VAX-11/780 (under VMS) in double precision. The machine precision is near 10^{-17} in this case. The resulting solution and closed-loop eigenvalues agreed to within the least significant digit published in [17]. RICPACK indicated the condition of the problem was on the order 10^1 and the r value for the solution was on the order 10^{-14}. The other large example, also solved on the VAX, appeared as example 3 in [22]. In this case, an 80th-order discrete-time problem was solved. The closed-loop eigenvalues were accurate to 16 significant digits and the Riccati solution was accurate to 13. RICPACK indicated the condition was on the order 10^2 and the r value for the solution was on the order of 10^{-15}.

VII. Concluding Remarks

While significant progress can be documented already on the numerical solution of Riccati equations, substantial numbers of questions remain to be investigated. The ubiquitous nature of these equations in mathematics and engineering together with the rapid and fundamental changes in computing environments of the 1980s offer important benefits to be gained in their further study. It is our contention that further progress is necessary but can only be made through an interdisciplinary approach blending systems theory, numerical analysis, computer science, and mathematical software. The research described in this paper is directed towards that goal.

REFERENCES

[1] B. D. O. Anderson and J. B. Moore, *Linear Optimal Control.* Englewood Cliffs, NJ: Prentice-Hall, 1971.

[2] M. Athans and P. L. Falb, *Optimal Control.* New York: McGraw-Hill, 1966.

[3] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems.* New York: Wiley, 1972.

[4] W. M. Wonham, *Linear Multivariable Control: A Geometric Approach*, 2nd ed. New York: Springer-Verlag, 1979.

[5] *IEEE Trans. Automat. Contr.*, Special Issue on Linear Quadratic Gaussian Control, vol. AC-16, Dec. 1971.

[6] B. D. O. Anderson and J. B. Moore, *Optimal Filtering.* Englewood Cliffs, NJ: Prentice-Hall, 1979.

[7] A. H. Jazwinski, *Stochastic Processes and Filtering Theory.* New York: Academic Press, 1970.

[8] K. Åström, *Introduction to Stochastic Control Theory.* New York: Academic Press, 1970.

[9] W. T. Reid, *Riccati Differential Equations.* New York: Academic Press, 1972.

[10] M. R. Scott, *Invariant Imbedding and its Applications to Ordinary Differential Equations.* Reading, MA: Addison-Wesley, 1973.

[11] W. H. Vandevender, "On the stability of an invariant imbedding algorithm for the solution of two-point boundary value problems," Sandia Laboratories Rep. SAND77-1107, Aug. 1977.

[12] A. J. Laub, "Schur techniques in invariant imbedding methods for solving two-point boundary value problems," in *Proc. 21st IEEE CDC* (Orlando, FL), pp. 55–61, Dec. 1982.

[13] B. Childs, M. Scott, J. W. Daniel, E. Denman, and P. Nelson Eds., *Codes for Boundary-Value Problems in Ordinary Differential Equations*, Lec. Notes in Computer Sci., vol. 76. New York: Springer-Verlag, 1979.

[14] B. T. Smith *et al.*, *Matrix Eigensystem Routines—EISPACK Guide*, 2nd ed., Lec. Notes in Computer Sci., vol. 6. New York: Springer-Verlag, 1976.

[15] B. S. Garbow *et al.*, *Matrix Eigensystem Routines—EISPACK Guide Extension*, Lec. Notes in Computer Sci., vol. 51. New York: Springer-Verlag, 1977.

[16] J. Dongarra *et al.*, *LINPACK User's Guide.* Philadelphia, PA: SIAM, 1979.

[17] A. J. Laub, "A Schur method for solving algebraic Riccati equations," *IEEE Trans. Automat. Contr.*, vol. AC-24, pp. 913–921, Dec. 1979.

[18] ———, "A Schur method for solving algebraic Riccati equations," LIDS Rep. LIDS-R-859, LIDS, MIT, Oct. 1978.

[19] K. H. Lee, "Generalized eigenproblem structures and solution methods for Riccati equations," Ph.D. dissertation, Dept. EE-Systems, USC, Jan. 1983.

[20] P. Van Dooren, "A generalized eigenvalue approach for solving Riccati equations," *SIAM J. Sci. Stat. Comp.*, vol. 2, pp. 121–135, 1981.

[21] T. Pappas, "Solution of discrete-time LQG problems with singular transition matrix," B. S. thesis, Dept. of Elec. Engrg., MIT, May 1979.

[22] T. Pappas, A. J. Laub, and N. R. Sandell, "On the numerical solution of the discrete-time algebraic Riccati equation," *IEEE Trans. Automat. Contr.*, vol. AC-25, pp. 631–641, Aug. 1980.

[23] A. Emami-Naeini and G. F. Franklin, "Deadbeat control and tracking of discrete-time systems," *IEEE Trans. Automat. Contr.*, vol. AC-27, pp. 176–181, Feb. 1982.

[24] R. A. Walker, A. Emami-Naeini, and P. Van Dooren, "A general algorithm for solving the algebraic Riccati equation," in *Proc. 21st IEEE CDC*, pp. 68–72, 1982.

[25] A. Emami-Naeini and G. F. Franklin, "Design of steady state quadratic loss optimal digital controls for systems with a singular system matrix," in *Proc. 13th Asilomar Conf. Circ. Syst. Comp.*, pp. 370–374, Nov. 1979.

[26] S. L. Campbell, *Singular Systems of Differential Equations.* Marshfield, MA: Pitman, 1980.

[27] ———, *Singular Systems of Differential Equations II.* Marshfield, MA: Pitman, 1982.

[28] C. B. Moler and G. W. Stewart, "An algorithm for generalized matrix eigenvalue problems," *SIAM J. Numer. Anal.*, vol. 10, pp. 241–256, 1973.

[29] R. C. Ward, "The combination shift QZ algorithm," *SIAM J. Numer. Anal.*, vol. 12, pp. 835–853, 1975.

[30] W. F. Arnold, "On the numerical solution of algebraic matrix Riccati equations," Ph.D. dissertation, Dept. EE-Systems, USC, Dec. 1983.

[31] C. C. Paige and C. F. Van Loan, "A Schur decomposition for Hamiltonian matrices," *Linear Algebra and Its Applications*, vol. 14, pp. 11–32, 1981.

[32] R. Byers, "Hamiltonian and symplectic algorithms for the algebraic Riccati equation," Ph.D. dissertation, Dept. Comp. Sci., Cornell University, Ithaca, NY, 1983.

[33] W. F. Arnold and A. J. Laub, "A software package for the solution of generalized algebraic Riccati equations," in *Proc. 22nd IEEE CDC* (San Antonio, TX), pp. 415–417, Dec. 1983.

[34] R. C. Ward, "Balancing the generalized eigenvalue problem," *SIAM J. Sci. Stat. Comput.*, vol. 2, pp. 141–152, 1981.

[35] B. C. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Trans. Automat. Contr.*, vol. AC-26, pp. 17–32, Feb. 1981.

[36] A. J. Laub, "On computing 'Balancing' transformations," in *Proc. 1980 JACC* (San Francisco, CA), pp. FA8–E, 1980.

[37] D. L. Kleinman, "On an iterative technique for Riccati equation computations," *IEEE Trans. Automat. Contr.*, vol. AC-13, pp. 114–115, Feb. 1968.

[38] G. A. Hewer, "An iterative technique for the computation of the steady state gains for the discrete optimal regulator," *IEEE Trans. Automat. Contr.*, vol. AC-16, pp. 382–384, Aug. 1971.

[39] T. J. Aird, *The Fortran Converter User's Guide.* Cambridge, MA: IMSL, MIT, 1975.

[40] J. Boyle and K. Dritz, "An automated programming system to aid the development of quality mathematical software," in *IFIP Proceedings.* New York: North-Holland, 1974, pp. 542–546.

[41] W. Cowell, *Portability of Numerical Software, Oak Brook, 1976*, Lec. Notes in Comp. Sci., vol. 57. New York: Springer-Verlag, 1977.

[42] W. Cowell and L. J. Osterweil, "The Toolpack/IST programming environment," Argonne National Lab., Appl. Math. Div., Rep. ANL/MCS-TM7, 1983.

[43] H. Crowder, R. S. Dembo, and J. M. Mulvey, "On reporting computational experiments with mathematical software," *ACM Trans. Math. Software*, vol. 5, pp. 193–203, 1979.

[44] J. Dorrenbacher, D. Paddock, D. Wisneski, and L. D. Fosdick, "POLISH, a Fortran program to edit Fortran programs," Dept. Comp. Sci., Univ. of Colorado (Boulder), Rep. CU-CS-050-74, 1974.

[45] L. D. Fosdick, Ed., *Performance Evaluation of Numerical Software.* New York: North-Holland, 1979.

[46] M. A. Hennel and L. M. Delves, Eds., *Production and Assessment of Numerical Software.* New York: Academic Press, 1980.

[47] P. Messina and A. Murli, Eds., *Problems and Methodologies in Mathematical Software Production*, Lec. Notes in Comp. Sci., no. 142. New York: Springer-Verlag, 1982.

[48] J. K. Reid, Ed., *The Relationship Between Numerical Computation and Programming Languages.* New York: North-Holland, 1982.

[49] J. R. Rice, *Matrix Computations and Mathematical Software.* New York: McGraw-Hill, 1981.

[50] ———, *Numerical Methods, Software, and Analysis.* New York: McGraw-Hill, 1983.

[51] B. G. Ryder, "The PFORT verifier: User's guide," CS. Tech. Rep. 12, Bell Labs, 1975; also *Software Practice and Experience*, vol. 4, pp. 359–377, 1974.

[52] G. W. Stewart, "Error bounds for approximate invariant subspaces of closed linear operators," *SIAM J. Numer. Anal.*, vol. 8, pp. 769–808, Dec. 1971.

[53] ———, "Error and perturbation bounds for subspaces associated with certain eigenvalue problems," *SIAM Review*, vol. 15, pp. 727–764, Oct. 1973.