NUMERICAL METHODS FOR LINEAR CONTROL SYSTEMS

by

Daniel Boley [1]
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455
e-mail: boley@cs.umn.edu

Biswa Nath Datta [2]
Department of Mathematical Sciences
Northern Illinois University
DeKalb, IL 60115
e-mail: dattab@math.niu.edu

Abstract

This paper gives a very brief overview of material of the short course entitled "**Numerical Methods for Linear Control Systems**" delivered by Biswa Nath Datta and Daniel Boley at the MTNS '96. As with the short course, the paper has two parts. Part I, written by B. Datta deals with computational aspects of dense linear algebra problems in control theory, and part II, written by D. Boley deals with large-scale solutions.

[1] The research of this author was supported by NSF under grant CCR-9405380.
[2] The work of this author was supported by NSF under grant DMS-9212629.

0. INTRODUCTION

The design and analysis of linear control systems:

$$\dot{x}(t) = Ax(t) + Bu(t),\ y = Cx(t) \tag{0.1}$$

and

$$x_{k+1} = Ax_k + Bu_k,\ y_k = Cx_k \tag{0.2}$$

give rise to many interesting linear algebra problems. Some of the important ones are: **controllability and observability problems, the problem of computing the exponential matrix e^{At}**, the matrix equations problems: **(Lyapunov equations, Sylvester equations, the algebraic Riccati equations), the pole-placement problems, stability problems, frequency response problems**, etc. These problems have been very widely studied in the literature. There exists a voluminous work both on theory and computations. Theory is very rich. Unfortunately, the same can not be remarked about computations. Many of the earlier methods were developed before the computer era, and are not based on numerically sound techniques. Fortunately, in the last twenty years or so, numerically effective techniques have been developed for most of these problems, and numerical analysis aspects of these methods (e.g. study of stability by round-off error analysis) and of the problems themselves (e.g. study of sensitivity) have been studied.

The purpose of this paper is: (a) First, to point out the difficulties in computational setting associated with many important theoretical methods, (b) Second, to outline some of the best ones from numerical view points, and (c) Third, to indicate the sensitivity issues of some of the problems that are available in the literature. Because of the lack of space, this is done only for a very few selected problems and mostly just for continuous-time models. The **survey is far from exhaustive**. Most of the methods for the continuous-time system (0.1) have discrete counterparts.

Some material of the paper has been taken from the book: **Numerical Methods for Linear Control Systems Design and Analysis**, currently being completed by one of the authors, Biswa Datta. The book will also have MATLAB based software implementing most of the algorithms described in the book, including algorithms for second-order control systems, not discussed in this paper.

PART I: NUMERICAL METHODS FOR DENSE PROBLEMS

by

Biswa Nath Datta

1. Introduction

This part of the paper is concerned with the computational methods for some selected problems of moderate size (about a few hundred or so) with no exploitable structure.

The selected problems include some of the important ones mentioned in the introduction. It is assumed that the readers are familiar with the control theory background. Most linear systems theory books such as Chen (1984), Kailath (1980), etc., have these material. As far as the numerical linear algebra background is concerned, as much as the author wanted to discuss it in details, was unable to do so due to space limitations. Only some crucial concepts such as **stability and conditioning**, and the methods for obtaining important canonical forms such as **the Hessenberg, Real Schur** and **Controller-Hessenberg** forms are briefly mentioned. For other linear algebra background, essential for understanding material of this paper, such as ***LU*** and ***QR* factorizations** and their applications to **solutions of linear systems and least square problems**, the **QR** and **QZ** algorithms for eigenvalue and generalized eigenvalue problems, and the **singular value decomposition**, etc., the readers are referred to the recent numerical linear algebra text of the author: **Numerical Linear Algebra and**

Applications (Datta (1995)) and the standard reference book in the subject: **Matrix Computations** (Golub and Van Loan (1989)). The recent reprint book "**Numerical Linear Algebra Techniques for Systems and Control**, edited by R.V. Patel, Alan J. Laub, and Paul M. Van Dooren (1993) contains most of the relevant papers.

2. Numerical Stability, Conditioning, Accuracy, and Efficiency

The numerical stability is a property of a computational algorithm, the conditioning is a property of the problem, but both have effects on the accuracy of the solution.

The numerical stability of an algorithm is established by means of round-off error analysis. There are two types of stability: **Forward stability** and **Backward stability**. The backward stability is widely used in practice in numerical linear algebra. **In this paper, by numerical stability, we will mean backward stability.**

An algorithm is said to be **backward stable** if the computed solution obtained by the algorithm is the exact solution of a nearby problem. For example, an algorithm for solving $Ax = b$ is backward stable, if the computed solution $\hat{x}$ satisfies $(A + E)\hat{x} = b + \delta b$, where the matrix E and the vector δb are small in some measure.

A problem (with respect to a given set of data) is said to be **ill-conditioned** if a small relative error in data of the problem causes a large relative error in the solution, regardless of the method used to solve the problem. Otherwise it is **well-conditioned**. Usually numerical analysts associate a number, called the **condition number**, with the problem that gives an indication of the conditioning. For example, the condition number of the linear systems problem $Ax = b$ is $\text{Cond}(A) = \|A\| \, \|A^{-1}\|$. **An important fact to note is that the backward stability of the algorithm does not guarantee an accurate solution.** However, when backward stable algorithm is applied to a well-conditioned problem, the solution should be accurate. **The condition numbers of only a few control problems have been identified. Some of them will be mentioned in this paper.**

Another important aspect of a numerical algorithm is its **efficiency**. A matrix algorithm involving computations with $n \times n$ matrices is said to be **efficient** if it does not require more than $O(n^3)$ floating point operations (**Flops**). **Note that an algorithm may be efficient but unstable**. The Gaussian elimination algorithm without pivoting is efficient, but unstable in general.

3. Canonical Forms

3.1 Ill-Conditioned Similarity Transformations and Potentially Numerically Dangerous Canonical Forms

A common strategy for solving control problems is:

1. Transform the problem by reducing the matrices A, B, and C to some convenient "**condensed**" forms using transformations that preserve the desirable properties of the problem in hand.
2. Solve the transformed problem by exploiting the structure of the condensed forms of the matrices A, B, and C obtained in step 1.
3. Recover the solution of the original problem from the solution of the transformed problem.

Extreme caution should be taken in implementing step 1 in floating point computations. Of course, steps 2 and 3 must be implemented also in a numerically stable way, but things might go wrong in the first place.

Let fl(s) denote the floating point computation of a quantity s, and let the matrix A be transformed by similarity to some condensed form. Then it can be shown (see Golub and Van Loan (1989), page 339-340), that:

$\mathrm{fl}(X^{-1}AX) = X^{-1}AX + E$, where $\|E\|_2 \approx= \mu \mathrm{Cond}_2(X)\|A\|_2$, ($\mu$ is the machine-precision). $\mathrm{Cond}_2(X)$ is the condition number with respect to the 2-norm.
Since the error matrix E clearly depends upon $\mathrm{Cond}_2(X)$, it follows from the above theorem that **we should avoid ill-conditioned transforming matrices in transforming A to a condensed form**.
Two condensed forms that have been used often in the past in control literature are:

1. The **Jordan Canonical Form** (JCF), and 2. The **Frobenius** (or **Block Companion**) Form (a variation of this is known as the **Luenberger Canonical Form**).

Exploitation of rich structures of these forms often makes it much easier to solve a problem.
Unfortunately, determination of both these forms might require very ill-conditioned similarity transformations. Thus, in these cases, the solutions obtained might be highly sensitive and therefore, the subsequent computations involving those solutions might be inaccurate.
It is difficult to determine exactly the Jordan structure of a matrix in case the matrix is defective or nearly defective. The eigenvector matrix X in this case is very ill-conditioned and this ill-conditioning contaminates the accurate determination of the JCF. The process of finding the companion form of a matrix A (or **equivalently finding the characteristic polynomial**) might be highly unstable. There are two stages: 1) Reduction of A to a Hessenberg matrix H 2) Further reduction of H to a companion matrix. **The first stage can be achieved in a numerically stable** way using Householder's or Givens' method (see Datta (1995)) with an orthogonal transformation (note that an orthogonal matrix is well-conditioned), however, **the second stage can be highly unstable**. The transforming matrix can be severely ill-conditioned if the product of the entries on the subdiagonal of the upper Hessenberg matrix H is small.

3.2 Examples of Inaccurate Computations with the JCF and Companion Matrices

We give below a few examples to illustrate how the use of the JCF and companion matrices may give inaccurate solutions in certain cases.
Example 1 (Pole-Placement using the Characteristic Polynomial Approach)

$A = 12 \times 12$ randomly generated matrix with entries between 0 and 1.

$b = (0.999, 0, 0, \cdots, 0)^T$.

$s = (1, 2, \cdots, 12)$: The eigenvalues to be assigned.

The feedback vector k is obtained by using the well-known Ackerman formula:

$$k = e_n^T C_M^{-1} \phi(A),$$

where C_M is the controllability matrix, and $\phi(x)$ is the desired closed-loop characteristic polynomial, The computed eigenvalues of $(A - bk)$ are : $1, 2, 2.9999, 4.0049, 4.9552, 6.0032, 6.8524 \pm 0.8937i, 9.1970 \pm 2.1155i, 12.4686 \pm 1.4509i$. (**Note some of them are even complex**).
Example 2 (Solving the Lyapunov Equation using the Jordan Canonical Form).

$$A = \begin{pmatrix} 1 & 0 \\ 2 & 0.9999 \end{pmatrix}, \; C = \begin{pmatrix} 6 & 3.9999 \\ 3.9999 & 1.9998 \end{pmatrix}.$$

The Lyapunov equation $XA + A^TX = C$ was solved by reducing A to a diagonal matrix first.

The compute solution $\hat{X} = \begin{pmatrix} 1.0009 & 0.9995 \\ 0.9995 & 1.0009 \end{pmatrix}$, whereas the exact solution $X = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$.
Residual: $\|XA + A^TX - C\| = 0(10^{-2})$.
Suggestions: **Avoid the use of the JCF and Companion canonical forms in numerical computations, and use only canonical forms that can be obtained using well-conditioned transforming matrices, such as orthogonal transformations.** The Hessenberg form, the Hessenberg-Controller form, the Real Schur forms, etc., are examples of such canonical forms.

Three Important Canonical Forms
In this section, we introduce three important canonical forms: **Hessenberg, Real Schur** and **Controller-Hessenberg**, which are routinely used as tools in recently developed numerically viable algorithms for control problems.

3.3 Reduction to Hessenberg Form

An arbitrary $n \times n$ matrix H can be transformed to an upper Hessenberg matrix H_U by orthogonality similarity:

$$PAP^T = H_U$$

There are at least two numerically viable methods: **Householder's method** and **Given's method**. Householder's method is more efficient than Given's method, but both are numerically stable.

A matrix H is called **Householder matrix** if $H = I - \dfrac{2uu^T}{u^Tu}$, where u is a vector. ***H* is symmetric and orthogonal.**

In Householder Hessenberg reduction, the matrix P is constructed as the product of $(n-2)$ Householder matrices P_1 through P_{n-2}. P_1 is constructed to create zeros in the first column of A below the entry $(2,1), P_2$ is determined to create zeros below the entry $(3,2)$ of the second column of $P_1AP_1^T$, and so on. At the end of $(n-2)$ the step, one, therefore, has $H_U = PAP^T$, where $P = P_{n-2}P_{n-3}\cdots P_1$. **Neither the matrices $\boldsymbol{P_i}$ nor the products $\boldsymbol{P_iAP_i^T}$ need to be computed explicitly.**

The method requires $\dfrac{5n^3}{3}$ flops to compute H_U and another $\dfrac{2}{3}n^3$ flops to compute P. **The method is numerically stable**. For details, see Datta (1995, 146-152).

3.4 QR Factorization and the Reduction to Real Schur Form

A real matrix is said to be in **Real Schur Form** (RSF) if it is quasi-triangular - a triangular matrix with $1x1$ or 2×2 block matrices on the diagonal. The 2×2 blocks are referred to as **bumps**. **Every matrix A can be transformed to a RSF by orthogonal similarity:** $PAP^T = S$. This is routinely done using **QR iteration algorithm**. The QR iteration algorithm is based on **QR Factorization** of a matrix. A factorization of a matrix A in the form $A = QR$, where Q is orthogonal and R is upper triangular, is called QR factorization of A.

There are several ways to achieve this factorization: **Householder's method, Givens' method, the Classical Gram-Schmidt Process**. **Modified Gram-Schmidt process**. Again Householder's method is preferred in practice. To obtain a QR factorization of A, the Householder matrices H_1 through H_{n-1} are computed successively to create zeros below the diagonal in the first column of A, in the second column of H_1A, and so on. Thus, at the end of $(n-1)th$ step, we have $H_{n-1}H_{n-2}\cdots H_2H_1A = R$.
Taking $Q = H_1H_2\cdots H_{n-1}$, we have $A = QR$. Note that Q is orthogonal.
Again, neither the Householder matrices $\boldsymbol{H_i}$ nor the products $\boldsymbol{H_iA}$ need to be computed explicitly.

The process takes about $\dfrac{2}{3}n^3$ flops to compute R and another $\dfrac{2}{3}n^3$ flops to compute Q. **It is numerically stable.**

To obtain RSF, the process is used in an iterative fashion as follows:

$$\begin{array}{ccccc} A & = & A_0 & = & Q_0R_0 \\ A_1 & = & R_0Q_0 & = & Q_1R_1 \\ \vdots & & & & \\ A_k & = & R_{k-1}Q_{k-1} & = & Q_kR_k \end{array}$$

Each member of the sequence $\boldsymbol{\{A_k\}}$ is orthogonally similar to the previous one and therefore, to the original matrix $\boldsymbol{A}$. Under some mild conditions, the sequence $\{A_k\}$ can be shown to converge to a Real

Schur matrix S.

The above is the **basic QR iteration method**. It is not practical, because to carry out n iterations, $O(n^4)$ flops will be needed. To make it practical, the matrix A is first transformed to an upper Hessenberg matrix H_U once and for all, and then the above iterative process is applied to H_U. The convergence, even when H is first transformed to H_U, can be painfully slow in the presence of two nearly multiple eigenvalues. To speed up the convergence, the process is used with some suitable shifts which are approximate eigenvalues. In practice, double shifts are used at each iteration, and the matrices with shifts are constructed implicitly. The resulting process is known as the **implicit QR iteration algorithm with double-shifts**. For details, see Datta (1995, 435-450).

The method requires about $13n^3$ flops to obtain both P and S. **It is stable**.

3.5 Reduction to Controller-Hessenberg Form

Given the pair of matrices (A, B), there always exists an orthogonal matrix P such that

$$PB = R = \begin{bmatrix} \bar{B} \\ 0 \end{bmatrix}, PAP^T = H,$$

where H is in block Hessenberg form with $n_1 = \rho(B) \equiv \text{rank}(B)$. This is called the **controller-Hessenberg** form for the pair (A, B). The system represented by (A, B) is controllable if and only if the matrix H in the above reduction is an unreduced (block) Hessenberg matrix.

Rank decisions aside, the algorithm for this orthogonal reduction is simple:

Algorithm 3.1 The Controller-Hessenberg Decomposition

Input: $A \in R^{n\times n}$ and $B \in R^{n\times m}$

Output: The controller-Hessenberg form $(H, R) = (PAP^T, PB)$ of the pair (A, B).

```
Step 1 Compute the QR decomposition $R = Q_0 B$, set $n_1 = \rho(B)$, rank of $B$.
Step 2 Compute $A_1 = Q_0 A Q_0^T$.
Step 3 Compute the Block Hessenberg Decomposition $H = QA_1Q^T$ :
    For $i = 1, 2, \cdots$
        Compute the QR decomposition $R_i = Q_i B_i$, where
            $A_i = \begin{bmatrix} * & * \\ B_i & \tilde{A}_i \end{bmatrix}$
            and $B_i$ is $\left(n + 1 - \sum_{j=1}^{i-1} n_j\right) \times n_i$.
        Set $n_{i+1} = \rho(B_i)$.
        Compute $A_{i+1} = Q_i \tilde{A}_i Q_i^T$
        Compute $Q = \begin{bmatrix} I & 0 \\ 0 & Q_i \end{bmatrix} Q$
        If $n_{i+1} = 0$ or $\sum_{j=1}^{i+1} n_j = n$ Quit
Step 4 Compute $P = QQ_0$
```

See Paige (1981) for more details.

Flop-count and Stability. In case the system is controllable, the algorithm requires about $\frac{5}{3}n^3 + n^2m + \frac{7}{2}n^2$ flops. **The algorithm is backward stable**.

The counterpart of the Controller-Hessenberg decomposition is the observer-Hessenberg decomposition for observability.

4. Solving the State Equation: Computation of e^{At}

It is well-known that the solution $x(t)$ of the continuous -time control system (0.1) can be written as

$$x(t) = e^{At}x_0 + \int_0^t e^{(t-s)A}Bu(s)ds.$$

The major problem here, is, therefore, computing the exponential matrix e^{At}.
There is a wide variety of methods to compute the exponential matrix. Several of these methods have been carefully analyzed, with respect to efficiency, and numerical stability, in an authoritative paper on the subject by Moler and Van Loan (1978). The methods can be broadly classified as: **Series Methods, Ordinary Differential Equation Methods, and Matrix Decomposition Methods, etc.**

Out of these methods, the method based on Padé approximation with scaling and squaring and the one based on the reduction of A to a Real Schur form are numerically attractive and widely used. ODE Methods are useful for large and sparse problems.

The drawbacks to the series method, which is a natural choice for the problem, is that a large number of terms is needed for convergence, and even when convergence occurs, the answer can be totally wrong.
Consider the following example from Moler and Van Loan (1978). $A = \begin{pmatrix} -49 & 24 \\ -64 & 31 \end{pmatrix}$
A total of $k = 59$ terms was required for convergence and the computed output was

$$e^A \approx \begin{pmatrix} -22.25880 & -1.432766 \\ -61.49931 & -3.474280 \end{pmatrix},$$

which is nowhere close to the true answer (to 6 decimal places)

$$e^A \approx \begin{pmatrix} -0.735759 & 0.551819 \\ -1.471518 & 1.103638 \end{pmatrix}.$$

The reason is that catastrophic cancellation took place in the evaluation of $\frac{A^{16}}{16!} + \frac{A^{17}}{17!}$. These two terms have almost the same magnitude but are of opposite signs.
Among the matrix-decomposition methods, the ones based on diagonalization of A, and the reduction of A to a companion matrix, should be avoided, dus to reasons mentioned earlier. The **Schur-method** based on the reduction of A to a triangular matrix can certainly be used. If $P^TAP = R$ is upper triangular, then the matrix e^R is easily computed by exploiting the triangular structure of R. Unfortunately, **large round-off errors might occur if there are multiple or near multiple eigenvalues.**
Sensitivity of the Matrix Exponential Problem
The matrix exponential problem can be highly ill-conditioned unless A is a normal matrix. It can be shown that there exists a perturbation matrix E for which the relative error can be as large as

$$\kappa(A,t)\frac{\|E\|_2}{\|A\|_2}, \text{ where } \kappa(A,t) = \max_{\|E\|_2=1} \| \int_0^t e^{A(t-s)}Ee^{As}ds\|_2 \ \frac{\|A\|_2}{\|e^{At}\|_2}.$$

5. Controllability, observability and Distance to Uncontrollability

The well-known mathematical criteria of controllability and observability such as the **Kalman rank criterion, the Hautus spectrum criterion, the disjoint spectrum criterion**, criteria based on computing the **controllability and observability Grammians,** etc. (see Kailath (1980), Chen (1984), etc), do not yield numerically effective tests for controllability and observability. For example, if $A = \text{diag}\left(1, 2^{-1}, \cdots, 2^{1-n}\right)$, and $B = (1, 1, \cdots, 1)^T$, then the pair (A, B) is obviously controllable. However, with $n = 10$, the controllability matrix $(B, AB, \cdots, A^9 B)$ will have three **small singular values**: $0.613 \times 10^{-12}, 0.364 \times 10^{-9}$ 0.712×10^{-7}. (Note that the most **numerically viable way to find the rank of a matrix is to compute its singular values**). Thus, using the Kalman criterion on a computer whose machine precision is no smaller than 10^{-12}, one will erroneously conclude that the pair is uncontrollable.

Similar examples can be given with the other criteria. (see Datta (1997)).

The most numerically effective way to determine the controllability of the pair (A, B) is to transform it to the controller-Hessenberg form, described previously.

Thus, (A, B) is controllable if in the pair $(H, \bar{B}), H$ is an unreduced block Hessenberg matrix.

Since observability is a dual concept of controllability, the test can also be applied to determine the observability of the pair (A, C). Thus (A, C) is observable if in the pair $(H, \bar{C})$, H is a block unreduced upper Hessenberg matrix, where $PAP^T = H,\ CP^T = (0, \bar{C})$

The controller-Hessenberg and observer-Hessenberg forms also give controllability and observability indices.

Distance from an Uncontrollable System

The concepts of controllability and observability are generic ones. Since determining if a system is controllable depends upon whether or not a certain matrix (or matrices) has full rank, it is immediately obvious that a controllable system may be very close to an uncontrollable system. For example, (Eising (1984)): if A is an upper Hessenberg matrix with unit superdiagonal and $B = e_1$, then the pair (A, b) is obviously controllable. However, it is easily verified that if we add $(-2^{1-n}, -2^{1-n}, \cdots, -2^{1-n})$ to the last row of $[A, b]$, we obtain an uncontrollable system. Clearly, when n is large, the perturbation 2^{1-n} is small, implying that the original controllable system (A, b) is close to an uncontrollable system. **Thus, what is important in practice is knowledge of how close a controllable system is to an uncontrollable one rather than determining if a system is controllable or not.** To this end, we introduce, following Paige (1981), **a measure of controllability:**

$$\begin{aligned}\mu(A, B) \triangleq \quad & \text{minimum } \|\Delta A, \Delta B\|_2 \text{ such that the system} \\ & \text{defined by } (A + \Delta A, B + \Delta B) \text{ is uncontrollable.}\end{aligned}$$

Here ΔA and ΔB are complex.

The quantity $\mu(A, B)$ gives us a measure of the distance of a controllable pair (A, B) to the nearest uncontrollable pair. **If this distance is small, then the original controllable system is close to an uncontrollable system. If this distance is large, then the system is far from an uncontrollable system.**

Here is a well-known result on $\mu(A, B)$ (see Eising (1984)).

Theorem 5.1 $\mu(A, B) = \min \sigma_n(sI - A,\ B)$, where $\sigma_n(sI - A,\ B)$ is the smallest singular value of $[sI - A, B]$ and s runs over all complex numbers.

Algorithms for Computing $\mu(A, B)$

Based on the above theorem, several algorithms (see Patel, Laub, and Van Dooren (1993)), have been developed in the last few years to compute $\mu(A, B)$. Another recent algorithm for computing $\mu(A, B)$ is due to Elsner and He (1991). The algorithm is a Newton-type algorithm and is based on finding zeros of a certain function.

Let's denote $\sigma_n\left[sI - A, B\right]$ by $\sigma(s)$. The problem of finding $\mu(A, B)$ is clearly then the problem of minimizing $\sigma(s)$ over the complex plane.
To this end, define

$$f(s) = v_n^*(s) \begin{pmatrix} u_n(s) \\ 0 \end{pmatrix},$$

where $u_n(s)$and $v_n(s)$ are the normalized nth columns of U and V in the **singular value decomposition** of $[A - sI, B]$; that is, $(A - sI, B) = U\Sigma V^T$.
It has been shown that the zeros of $f(s)$ are the critical points of the function $\sigma(s)$.
Thus, some well-established root finding methods, such as the **Newton method**, or the **Bisection method** can be used. For details see Elsner and He (1991).

6. Numerical Methods for Matrix Equations

In this section, we consider the well-known matrix equations in control theory:

> **Lyapunov Equation**: $XA + A^TX = C$, **Sylvester Equation**: $AX + XB = C$, **Sylvester Observer Equation**: $XA - FX = DC$, **and Algebraic Riccati Equation**: $XA + A^TX - XBR^{-1}B^TX + Q = 0$.

The Sylvester-observer equation is a variation of the Sylvester equation, and the Lyapunov equation is a special case of the algebraic Riccati equation.
There exists a wide variety of methods for the Lyapunov and the Sylvester equations: finite and infinite series solutions methods, methods based on reduction of the matrices A and B to convenient forms such as companion or Schwarz, Hessenberg Forms, methods based on computing the **explicit expressions** for solutions of these equations, etc. **Out of these, the Schur method for the Lyapunov equation**, based on the reduction of A to RSF, devised by Bartels and Stewart (1972), and the **Hessenberg-Schur method** for the **Sylvester equation**, based on the reductions of A to Hessenberg and B^T to RSF (assuming that B is of smaller dimension than A), devised by Golub, Nash, and Van-Loan (1979), are **the best from the numerical point of view and are widely used**. There also exists an important variation of the Schur method by Hammarling (1982) for constructing the Cholesky factor of the symmetric positive definite solution of the Lyapunov matrix equation, without computing the solution itself.

6.1 The Hessenberg-Schur Method for the Sylvester Equation

Let A be $m \times m$ and B be $n \times n$; $n < m$. The matrix equation $AX + XB = C$ is first reduced to the equation $HY + YS^T = C'$, where $H = U^TAU$, $V^TB^TV = S$, and $C' = U^TCV$. H is upper Hessenberg, S is in RSF, and U and V are orthogonal. The reduced Hessenberg-Schur problem is then solved for Y by finding one or two columns at a time. If $s_{k-1,k} = 0$, then the kth column of Y is found by solving the $\boldsymbol{m \times m}$ **Hessenberg system:** $(H + s_{kk}I)y_k = c'_k - \sum_{j=k+1}^{n} s_{kj}y_j$. If $s_{k-1,k} \neq 0$, the two columns y_{k-1} and y_k are obtained by solving the $\boldsymbol{2m \times 2m}$ **system:**

$$H\left(y_{k-1}, y_k\right) + \left(y_{k-1}, y_k\right) \begin{pmatrix} s_{k-1,k-1} & s_{k,k-1} \\ s_{k-1,k} & s_{kk} \end{pmatrix} = \left(c'_{k-1}, c'_k\right) - \sum_{j=k+1}^{n} \left(s_{k-1,j}y_j, s_{kj}y_j\right)$$

Finally, the solution X is recovered from $X = UYV^T$.

The Schur-Method for the Lyapunov Equation

The Schur-method for the Lyapunov equation can be viewed as a special case of the Hessenberg-Schur method, in which only the A is decoposed into RSF.

Flop-Count The Hessenberg-Schur method for the Sylvester equation requires about $\frac{5}{3}m^3 + 13n^3 + 5m^2n + \frac{5}{2}mn^2$ flops.

The Schur method for the Lyapunov equation requires about $13n^3 + \frac{7}{2}n^3$ flops.
Stability of the Schur and Hessenberg-Schur Algorithms
The round-off error analysis performed by Golub, Nash, and Van Loan (1979), and more recently in detail, by Higham (1996) shows that **these algorithms are at best conditionally backward stable**; which might be satisfactory for all practical purposes.

Hessenberg methods for the Sylvester and Lyapunov Equations

Since the reduction of a matrix to Real Schur form is rather expensive, and on the other hand, as an initial step to the Schur-reduction, one needs to transform the matrix to a Hessenberg matrix, it is natural to wonder if methods can be devised that are based solely on Hessenberg reductions. Such a method has been proposed by Datta and Datta (1987) for certain Lyapunov equations. **This algorithm followed by some iterative refinement procedure might produce acceptable results**, but the algorithms itself is not guaranteed to be stable.

6.2 The Sylvester-observer Matrix Equation

The matrix equation $XA - FX = DC$, where A and C are given such that (A, C) is observable, and X, F, and D have to be found such that (i)F has an arbitrary spectrum and (ii) (F, D) is controllable, is called the **Sylvester-observer equation**.

The equation is so-called, because, it is a variation of the classical Sylvester equation considered in the last section, and this equation arises in the design of Luenberger observer (see Datta (1994) for more details on this equation).

The numerical methods for solving this equation include a method by Van Dooren (1984), a parallel/highperformance algorithm by Bischof, Datta and Purkayastha (1996), and Arnoldi-type methods for constructing **orthogonal solutions** to the equation by Datta and Saad (1991), and Datta and Hetti (1996).

The Van Dooren method is based on orthogonal reduction of the observable pair (A, C) to the observer-Hessenberg form. Thus if P is an orthogonal matrix such that $PAP^T = H$, an unreduced block Hessenberg matrix, and $CP^T = \tilde{C} = (0, C_1)$, then the Sylvester-observer equation is reduced to $YH - FY = D\tilde{C}$, where $Y = XP^T$. Once this reduced equation is solved, the solution of the original equation can be recovered from $X = YP$. It has been shown that by choosing Y as the unit-upper triangular matrix and F as lower triangular matrix with arbitrary diagonal entries, the remaining entries of Y and D can be computed simultaneously, by finding one row of each matrix at a time.

Neither the Van Dooren method nor the Hessenberg-Schur method for the Sylvester equation constructs, in general, an orthogonal solution. On the other hand, since the solution to a Sylvester-observer equation is used in subsequent computations in the design process of a control system, it is highly desirable, from numerical and practical view points, to have a well-conditioned solution. The Aroldi type methods of Datta and Saad (1991) and Datta and Hetti (1996) construct such solutions. The Datta-Saad method is a projection method; on the other hand the Datta-Hetti method is based on a new generalization of the classical Arnoldi-recursion.
The Bischof-Datta-Purkayastha (1996) algorithm constructs the matrix Y with a block of rows at a time by choosing the matrix F as a block bidiagonal matrix with arbitrarily chosen diagonal blocks.

6.3 The Algebraic Riccati Equation

The continuous-time algebraic Riccati equation (CARE)

$$XA + A^T X - XSX + Q = 0,$$

where $S = BR^{-1}B^T$, arises in **linear quadratic regular problem, H-infinity control problems, filter theory**, etc.

The solution of most practical interests in these applications is the **symmetric positive semidefinite stabilizing solution**.
The numerical methods for solving the CARE include (i) **The Eigenvector Methods**, (ii) **The Schur-methods** (Laub (1979), and Byers (1986)), (iii) **The Matrix Sign Function Method** (Roberts (1980)), the **generalized eigenvalue-eigenvector methods** (Van Dooren (1981), Arnold and Laub (1984), etc.), etc.

The eigenvector and Schur methods are based on constructing a basis for an appropriate invariant subspace of the associated Hamiltonian matrix

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}.$$

The eigenvector method can be highly unstable if there are multiple or near multiple eigenvalues of $\boldsymbol{H}$. This difficulty in the eigenvector method can be overcome by using the Schur-vectors (the vectors of the orthogonal transforming matrix that transforms H to Real Schur form), resulting in the Schur method for the problem.

The ordinary Schur method, however, fails to exploit the special Hamiltonian structure of H. The Hamiltonian-Schur method due to Byers (1986) is based on the transformation of H to **Hamiltonian-Schur form**, which preserves the special rich structure of H. The method is, however, restricted to the single-input case only.

The matrix sign function method is based on constructing the matrix sign function of the Hamiltonian matrix H, and is convenient for use on parallel machines. It is not stable as such. But, when used in conjunction with an iterative refinement technique, produces satisfactory results.

The generalized eigenvector and Schur methods are based on finding the eigenvectors and Schur vectors of the pencil $M - \lambda N$, where

$$M = \begin{pmatrix} A & 0 & B \\ -Q & -A^T & 0 \\ 0 & B^T & R \end{pmatrix}, N = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

The methods are, therefore, suitable when R is singular or ill-conditioned.

The pencil $M - \lambda N$ can be further compressed into an $n \times n$ pencil using an orthogonal reduction of the matrix $\begin{pmatrix} B \\ R \end{pmatrix}$. **Again, for reasons of numerical instability, the generalized Schur method is preferred**.

Newton's method is naturally an iterative method. It is, therefore, suitable for refining an approximate solution obtained by another method.

In fact, **a common practice to solve the CARE is to use the Schur method (or the generalized Schur method) followed by Newton's method.**

6.3.1 The Schur Method for the Stabilizing Solution of the CARE

The Hamiltonian matrix H is first transformed to the **ordered** RSF: $UHU^T = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$, where the n eigenvalues of H with negative real parts are contained in T_{11}. (For a program on reordering RSF, see Stewart (1976)). The matrix $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$ is then conformably partitioned, and finally X is computed from $X = U_{21}U_{11}^{-1}$.

Flops-count. Assuming that the average number of QR iterations is 1.5,, about $75n^3$ flops will be necessary to execute the algorithm.

Stability of the method. The method is not always stable. These are cases where the method introduces more error than the data warrants. In such cases, a proper scaling is recommended; however, such scaling requires a prior knowledge of the solution matrix X.

6.3.2 Newton's Method for the CARE

Step 1. Choose K_0 such that $(A - BK_0)$ is asymptotically stable.

Step 2. Construct the sequence of solutions $\{X_i\}$ as follows:

```
For $i = 0, 1, 2, \cdots$ do until convergence occurs
    (a)  Compute $A_i = A - BK_i$.
    (b)  Compute $Q_i = Q + K_i^T RK_i$.
    (c)  Compute the unique positive semidefinite solution $X_i$ of the Lyapunov equation
         $A_i^T X_i + X_i A_i + Q_i = 0$.
    (d)  Stop if a criterion of convergence is satisfied.
    (e)  Compute $K_{i+1} = R^{-1} B^T X_i$.
```

It can be shown that $\lim_{k\to\infty} X_k = X$.

6.3.4. Sensitivity of the Riccati and Lyapunov Equations

A perturbation analysis [see Byers (1985) and Petkov et al (1991)] shows that if the data matrices in the CARE have relative errors of size μ, an upper bound of the relative error in the solution X is μ times

$$\frac{2\|A\|_F + \dfrac{\|Q\|_F}{\|X\|_F} + \|S\|_F \|X\|_F}{sep\left((A - SX)^T, (A - SX)\right)},$$

where $sep\,(A, -B) = \sigma_{\min}\left(I \bigotimes A + B^T \bigotimes I\right), \sigma_{\min}$ stands for the minimum singular value and $\bigotimes$ denotes the Kronecker product. **Thus the CARE is ill-conditioned if the norm of the solution matrix $\boldsymbol{X}$ is very small or very large and/or $\boldsymbol{sep}\left((\boldsymbol{A} - \boldsymbol{SX})^{\boldsymbol{T}}, -(\boldsymbol{A} - \boldsymbol{SX})\right)$ is small.**

Sensitivity of the Sylvester Equation

The number $\dfrac{\|A\|_F + \|B\|_F}{sep(A, -B)}$ is considered as a **condition number** for the **Sylvester equation** . The condition number of the Lyapunov equation is obtained as a special case.

Linpack-style Estimator for $\boldsymbol{sep(A, B)}$

There is a **LINPACK-style estimator** for $sep(A, B)$ by Byers (1984) that avoids computing the Kronecker product and the singular values.

7. Pole-placement problems

Given an $n \times n$ matrix A, and $n \times m$ matrix $B(m \leq n)$, and a set $S = \{s_1, \cdots, s_n\}$ of numbers, closed under complex conjugation, the problem of finding a matrix K such that $A - BK$ has the spectrum S, is known as the **pole-placement** or the **eigenvalue assignment** problem.

The problem is central to the design of a control system. There exists a wide variety of methods for the problem. Many of the earlier developed methods are based on computing the characteristic polynomial of A or equivalently on the reduction of A to companion or Luenberger canonical form are discarded for reasons of numerical instability. **The most recent numerically attractive algorithms are based on the orthogonal reduction**

of $(\boldsymbol{A}, \boldsymbol{B})$ to the controller-Hessenberg form $(\boldsymbol{H}, \bar{\boldsymbol{B}})$.

The different algorithms differ in the way the reduced Hessenberg problem is solved. Such existing algorithms include recursive algorithms of Datta (1987), Arnold and Datta (1990), QR algorithms of Miminis and Paige (1988), Patel and Misra (1984), Petkov et al (1984), Arnold and Datta (1995); the Schur-methods of Varga (1981), the matrix equation method of Bhattacharyya and DeSouza (1982), etc.

The QR and Schur methods are believed to be numerically stable. In fact, stability of some of these algorithms have been proven by detailed round-off error analysis (Miminis and Paige (1988), Cox and Moss (1989), Arnold and Datta (1995), etc).

The results of these papers show that the feedback matrices K for the respective algorithms are exact for slightly perturbed Datta matrices A and B, and the set S. **Note that this does not guarantee that the assigned eigenvalues will be accurate.**

The efficiency of all these algorithms, except the recursive ones, are comparable. **The recursive algorithm of Datta (1987) and that of Arnold and Datta (1990) are, the fastest algorithms developed so far, respectively, for the single-input and multi-input problems.**

Furthermore, it has been shown in Arnold (1993) and Arnold and Datta (1995) that all the single-input QR algorithms are related to each other - this relationship has been exposed via RQ (rather than QR) factorizations of the deflated matrices at each step. The RQ version of the single-input recursive algorithms has provided such a unified framework.

7.1 A Single-input Recursive Algorithm for Pole-placement

Step 1. Construct a set of normalized vectors, $\{\ell_k\}$:

```
$\ell_1 = (0, 0, \cdots, 1)^T$.
For $i = 1, 2, \cdots, n-1$ do
    $\hat{\ell}_{i+1} = \left(H^T - \lambda_i I\right) \ell_i$
    $\ell_{i+1} = \dfrac{\hat{\ell}_{i+1}}{\|\hat{\ell}_{i+1}\|}$
END
$\ell_{n+1} = (H^T - \lambda_n I)\ell_n$.
```

Step 2. Compute $f = \dfrac{\ell_{n+1}}{\alpha}$, where α is the first entry of ℓ_n.
Theorem 7.1 $\Omega(H - e_1 f^T) = \{\lambda_1, \cdots, \lambda_n\}$ where $\Omega(M)$ = spectrum of M.

Efficiency The algorithm requires $\dfrac{n^3}{6}$ flops.

Stability The round-off error analysis performed in Arnold and Datta (1995) shows that the algorithm is not guaranteed to be stable; however it is **reliable in the sense that its stability can be monitored or controlled by monitoring or controlling the condition number of $\boldsymbol{L}$**, which can be computed rather cheaply. L is the matrix of the normalized vectors obtained from the above algorithm. However, the **RQ version of the algorithm has been proven to be backward stable** (Arnold and Datta (1995)).

7.2 A Multi-input Recursive Algorithm

A straightforward generalization of the algorithm to the multi-input case appears in Arnold and Datta (1990).

7.3 Robust Pole Assignment

A robust pole assignment for a **multi-input problem** is concerned with assigning not only eigenvalues, but also a set of linearly independent eigenvectors associated with the required eigenvalues such that the matrix of eigenvectors is as well-conditioned as possible. The idea is to find a feedback matrix F such that the assigned poles are as insensitive to perturbations as possible. Mathematically, given (A, B) and the set $s = \{\lambda_1, \cdots, \lambda_n\}$, find a matrix F and a non-singular matrix X satisfying

$$(A + BF)\,X = X\Lambda,$$

where $\Lambda = \text{diag}\,(\lambda_1, \cdots, \lambda_n)$, such that the eigenproblem is as well-conditioned as possible. Kautsky, Nichols, and Van Dooren (1984) have given **several iterative schemes** for choosing such an X, and an explicit expression for F, when it exists:

$$F = Z^{-1} U_0^T \left(X\Lambda X^{-1} - A \right),$$

where

$$B = [U_0, U_1] \begin{bmatrix} Z \\ 0 \end{bmatrix}.$$

7.4 A New Arnoldi-type Algorithm

In a recent thesis, Hetti (1995), has given an algorithm for constructing a feedback matrix F for the multi-input problem using an **orthogonal solution** to a certain Sylvester-observer matrix equation. The matrix equation is solved using a **generalization** of the **classical Arnoldi scheme**. The feedback matrix F obtained this way is expected to be **robust** (well-conditioned).

7.5 Conditioning of the pole-placement and the Feedback problems

The pole-placement is intrinsically an ill-conditioned problem. For even moderate order systems (say $n \geq 15$) the eigenvector matrix of the closed-loop matrix is usually highly ill-conditioned. This phenomenon has been thoroughly studied in a recent paper by He, Laub, and Mehrmann (1995). The condition number for the **feedback problem** has recently been identified by Arnold (1993), who has also given several cheap estimators for the condition number.

8. Control Problems for Second-order Systems

There have been some fine developments in numerical solutions for control problems associated with the second-order control system: $M\ddot{x} + D\dot{x} + K = Bu$.

The traditional engineering approach is **Modal Space Control Approach**, which requires complete knowledge of eigenvalues and eigenvectors of the associated quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$, and is not practical for large and sparse systems. Nonmodal and partial modal approaches for the feedback stabilization, eigenvalue assignment, and partial eigenvalue assignment have been developed, which are suitable for large and sparse problems (e.g. those arising in the design of large flexible structures). The readers are referred to the papers: Datta and Rincon (1993), Datta, Elhay, and Ram (1995, 1996) and Chu and Datta (1995).

PART II: KRYLOV SPACE METHODS FOR CONTROL PROBLEMS
by
Daniel L. Boley

9. Krylov Space Methods

We give an overview of various Lanczos/Krylov space methods and how they are being used for solving certain problems in Control Systems Theory based on state-space models. The matrix methods used are based on Krylov sequences and are closely related to modern iterative methods for standard matrix problems such as sets of linear equations and eigenvalue calculations. We show how these methods can be applied to problems in Control Theory such as controllability, observability and model reduction. All the methods are based on the use of state-space models, which may be very sparse and of high dimensionality.
A block Krylov sequence is generated by an $n \times n$ matrix A and an $n \times p$ matrix B as follows

$$K(A, B, k) \equiv (\, B, AB, A^2B, \cdots, A^{k-1}B \,)\,,$$

and the corresponding column space is called the *k-th Krylov space* and is denoted by $\mathcal{K}(A, B, k)$. When working with problems with very large dimensionality, it is necessary and useful to project the large problem onto a smaller subspace, and it turns out that the Krylov space is an excellent choice for such a smaller subpace. There are two main reasons for this: (a) the resulting projections tend to inherit most of the useful properties from the original operator, such as the extreme eigenvalues or the leading Markov moments, and (b) there are very efficient incremental algorithms for generating good bases for this space. We first sketch some of the principal methods for generating these spaces, then we sketch some examples where these methods have found great usefulness. Space does not permit a detailed description, but many of the details can be found in (Boley (1994)).

9.1 Methods

The Arnoldi method [Arnoldi (1951), Boley and Golub (1991), and Wilkinson (1965)] is used to generate an orthogonal basis $\mathbf{X}$ for the Krylov Space, starting with an $n \times n$ matrix A and a set of p starting vectors B. The algorithm starts by orthogonalizing the starting vectors by means of a QR decomposition: $X_1R = B$. The algorithm then proceeds by recursively filling in the first few columns in the relation

$$A\mathbf{X} = \mathbf{X}\mathbf{H} \text{ subject to } \mathbf{X}^T\mathbf{X} = I, \tag{9.1}$$

where $\mathbf{X} = (X_1, X_2, \ldots)$ is an $n \times r_{\max}$ matrix of orthonormal columns spanning the Krylov space of maximal rank $r_{\max}$ and $\mathbf{H}$ is an $r_{\max} \times r_{\max}$ block upper Hessenberg matrix. At each step j, the algorithm expands the Krylov space by forming the product AX_j, then orthogonalizes the product against all the blocks $(X_1, \ldots, X_j)$ computed so far, and finally orthonormalizes the result using the QR decomposition (with pivoting if necessary), yielding the new block of vectors X_{j+1}. The coefficients from the orthogonalizing steps are collected into the block upper Hessenberg matrix $\mathbf{H}$. When we start with only a single starting vector, some of these steps are simplified: each X_i consists of a single vector, and $\mathbf{H}$ is "scalar" upper Hessenberg.
Unless the starting vectors are deficient in certain eigendirections, $r_{\max} = n$. However, the Arnoldi algorithm is never carried out this far, but is stopped after $r \leq r_{\max}$ steps, in which case we have the orthonormal basis $\mathbf{X}_r = (X_1, \ldots, X_r)$ for the partial Krylov space $\mathcal{K}(A, B, r)$. This algorithm is the basic method behind the popular GMRES method [Saad and Schultz (1986)] for solving large sparse system of linear equations.
As a special case, if the matrix A is symmetric, then so will be the block upper Hessenberg matrix $\mathbf{H}$, hence it will be block tridiagonal. For the algorithm, this implies that at each step j, the product AX_j need be orthogonalized against only X_j and X_{j-1}. In this way it is possible to drop all the previous history, saving only the most recent blocks of vectors, at least in exact arithmetic. The resulting algorithm is called the symmetric Lanczos algorithm Lanczos (1950). It is possible to extend this savings to the nonsymmetric case, but only by sacrificing another desirable property, namely orthogonality. The resulting algorithm is usually called the non-symmetric Lanczos

algorithm, and involves two Krylov spaces: the right space $\mathcal{K}(A,B,r_{\rm r})$, and the left space $\mathcal{K}(A^T,C,r_{\rm l})$. When starting with multiple starting vectors, it is called the the block nonsymmetric Lanczos algorithm.
In the block nonsymmetric Lanczos algorithm, we generate two sets of vectors $\mathbf{X},\mathbf{Y}$, and two block upper Hessenberg matrices of orthogonalization coefficients $\mathbf{H},\mathbf{G}$ satisfying

$$A\mathbf{X} = \mathbf{XH} \tag{9.2a}$$
$$A^T\mathbf{Y} = \mathbf{YG} \tag{9.2b}$$
$$\mathbf{Y}_i^T\mathbf{X}_j = 0 \text{ if } i \neq j \tag{9.2c}$$
$$\mathbf{Y}_i^T\mathbf{X}_i = D_i \text{ is nonsingular, for } i \leq l \tag{9.2d}$$

and for all $k = 1,2,3,\ldots$

$$\text{COLSP}\,(\mathbf{x}_1,\ldots,\mathbf{x}_k) = \text{ the span of the first } k \text{ independent columns in } \mathcal{K}(A,B,\infty) \tag{9.2e}$$
$$\text{COLSP}\,(\mathbf{y}_1,\ldots,\mathbf{y}_k) = \text{ the span of the first } k \text{ independent columns in } \mathcal{K}(A^T,C,\infty)\ , \tag{9.2f}$$

In the above, we have grouped the columns of $\mathbf{X},\mathbf{G}$ into l clusters: $X_1,X_2,\ldots,X_l$, $Y_1,Y_2,\ldots,Y_l$, but these clusters do not correspond to those obtained in the block Arnoldi algorithm (9.1). Instead these clusters are normally just single vectors, even when starting with multiple starting vectors. Normally the *bi-orthogonality condition* (9.2c) is enforced between individual vectors, but if $\mathbf{D} = \mathbf{G}^T\mathbf{H}$ is singular it is impossible to do that, so we group the vectors into *look-ahead clusters* such that the bi-orthogonality condition is enforced only between different clusters. From (9.2c), we see that $\mathbf{D} = \mathbf{G}^T\mathbf{H}$ is block diagonal, and from that it follows that $\mathbf{H},\mathbf{G}$ must both be block tridiagonal. The look-ahead clusters are defined by enforcing the condition (9.2d) for all the clusters, except for the last one which may be incomplete due to termination of the algorithm.
A brief sketch of the block non-symmetric Lanczos process is as follows, in which the vectors sequences $\mathbf{X},\mathbf{Y}$ are computed one column at a time (even in the block case). We refer the reader to (Aliaga et al (1996)) for the precise details. We sketch the situation at the k-th stage. The process is initialized by applying the two-sided Gram-Schmidt process (Parlett (1992)) to the starting vectors B,C. Suppose we have k right vectors and k left vectors:

$$\mathbf{X}_{r_{\rm r}} = (X_1,\ldots,X_l) = (\mathbf{x}_1,\ldots,\mathbf{x}_k) \text{ and } \mathbf{Y}_{r_{\rm l}} = (Y_1,\ldots,Y_l) = (\mathbf{y}_1,\ldots,\mathbf{y}_k)$$

partitioned into l clusters as indicated, where the last cluster is either empty or incomplete (meaning that the l-th diagonal block of $\mathbf{D}$, $D_l = Y_l^TX_l$, is singular), but $\mathbf{D}_{l-1} = (Y_1,\ldots,Y_{l-1})^T(X_1,\ldots,X_{l-1})$ is nonsingular.

1. Expand both the left and right Krylov spaces, i.e. compute vector $\tilde{\mathbf{x}} = A\mathbf{x}_i$ (respectively $\tilde{\mathbf{y}} = A^T\mathbf{y}_j$), where $\mathbf{x}_i$ (respectively $\mathbf{y}_j$) is the first vector in $\mathbf{X}$ (respectively $\mathbf{Y}$) not yet expanded in this fashion.

2. Biorthogonalize $\tilde{\mathbf{x}},\tilde{\mathbf{y}}$ against all previous *complete* clusters $Y_1,Y_2,\ldots,Y_{l-1}$ and $X_1,X_2,\ldots,X_{l-1}$, respectively, using an incremental two-sided Gram-Schmidt process, filling in the relations (9.2a), (9.2b). Actually, since the coefficient matrices $\mathbf{H},\mathbf{G}$ are block tridiagonal, only the most recent clusters must be used.

3. If the current look-ahead cluster X_l,Y_l is non-empty, orthogonalize the resulting vectors from the previous step against the vectors in the same cluster. This is done to ensure linear independence and to enhance the numerical stability.

4. If the right vector resulting from the previous step is zero, then obtain another by repeating those parts of steps 1-3 related to the right Krylov vector $\tilde{\mathbf{x}}$. If the left vector is zero, repeat those parts of the previous steps related to the left Krylov vector $\tilde{\mathbf{y}}$.

5. The resulting nonzero right and left vectors obtained, after suitable scaling, become the new vectors $\mathbf{x}_{k+1},\mathbf{y}_{k+1}$, respectively.

6. We include the vectors $\mathbf{x}_{k+1},\mathbf{y}_{k+1}$ in their current respective look-ahead cluster, X_l,Y_l, and increment k. If the resulting diagonal block $D_l = Y_l^TX_l$ is nonsingular, then the current look-ahead cluster is complete, so we close it and open a new empty look-ahead cluster X_{l+1},Y_{l+1} and increment l.

9.2 Relation to Dynamical Systems

We mention some of the principal ways the Krylov space methods have been used in control and dynamical systems. For reasons of space, we limit this discussion to the case of single starting vectors ("single input single output") though most of these are easily extendible to the multiple vector case.

9.2.1 Moment Matching

A popular use of Krylov space methods is in model reduction. Suppose we have a dynamical system represented by $(A, \mathbf{b}, \mathbf{c})$ whose transfer function is

$$f(z) = \mathbf{c}^T(zI - A)^{-1}\mathbf{b} = \sum_{i=0}^{\infty} \frac{\mathbf{c}^T A^i \mathbf{b}}{z^{i+1}} = \sum_{i=0}^{\infty} \frac{m_i}{z^{i+1}},$$

where m_i is the i-th high frequency moment (or impulse response if in discrete time). Suppose also that we have used the scalar Lanczos to generate k vectors: $A\mathbf{X}_k = \mathbf{X}_k\mathbf{H}_k + \mathbf{x}_k\mathbf{e}_k^T\beta_k$, $A^T\mathbf{Y}_k = \mathbf{Y}_k\mathbf{G}_k + \mathbf{y}_k\mathbf{e}_k^T\gamma_k$, where we have included the remainder term after the k-th step. Then one can show that the first $2k$ moments of the original system $f(z)$ are preserved in the reduced k-th order system represented by the matrices $(\mathbf{H}_k, \mathbf{e}_1, D_1^T\mathbf{e}_1)$, where the quantities $D_1, \mathbf{H}_K$ are exactly those arising from the Lanczos algorithm, and $\mathbf{e}_1$ is the first coordinate unit vector. This idea is the basis for the model reduction techniques for space structures in (Kim and Craig (1988), (1990)) and in the Padé-Lanczos waveform analysis of circuits in (Feldmann and Freund (1995)). In these applications, it is also necessary to find the low frequency moments $\mathbf{c}^T A^{-k}\mathbf{b}$, $k = 1, 2, \ldots$, which involves an implicit inversion of the system matrix A.
We note that if the Lanczos process is carried to completion so that $\mathbf{X}_k$ is a basis for the entire Krylov space $\mathcal{K}(A, \mathbf{b}, \infty)$, then the "reduced" order system $(\mathbf{H}_k, \mathbf{e}_1, D_1^T\mathbf{e}_1)$ is the *minimal realization* of the original system, and $\mathbf{Y}_k$ is a basis for the orthogonal complement to the unobservable space. Furthermore the look-ahead clustering from the Lanczos also yields directly bases for the controllable-observable, controllable-unobservable, and uncontrollable-observable spaces, namely, $(X_1, \ldots, X_{l-1})$, X_l, and Y_l, respectively, as fully discussed in (Boley and Golub (1991), Boley (1994)).

9.2.2 Balanced Realization, Lyapunov and Riccati Equations

Another popular approach for model reduction is via the balanced realization. It is well known that in order to obtain the balanced realization of a system represented by the matrices (A, B, C), it is necessary to obtain the reachability and observability grammians: $\mathbf{W}_{\text{c}}, \mathbf{W}_{\text{o}}$. In the time-invariant infinite horizon case, these are obtained by solving the respective Lyapunov equations $A\mathbf{W}_{\text{c}} + \mathbf{W}_{\text{c}}A^T = -BB^T$ and $A^T\mathbf{W}_{\text{o}} + \mathbf{W}_{\text{o}}A = -C^TC$, in continuous time, and $A\mathbf{W}_{\text{c}}A^T - \mathbf{W}_{\text{c}} = -BB^T$ and $A^T\mathbf{W}_{\text{o}}A - \mathbf{W}_{\text{o}} = -C^TC$, in discrete time. The Arnoldi method has been proposed as a way of projecting the original system to a lower order system in which the corresponding projected Lyapunov equations can be easily solved (Saad (1990), Jaimoukha and Kasenally (1994)) for continuous time; (Boley (1994)) for discrete time). For example, in discrete time the reachability Lyapunov equation is projected to $\mathbf{H}_k G \mathbf{H}_K^T + \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{pmatrix} R^T & 0 \end{pmatrix} = G$. Then the approximate solution is $\widetilde{\mathbf{W}}_{\text{c}} = \mathbf{X}_k G \mathbf{X}_k^T$, where $R, \mathbf{X}_k, \mathbf{H}_k$ are the matrices obtained from the block Arnoldi algorithm. It has been shown that this solution satisfies an algebraic Galerkin condition and also satisfies an error bound which converges to zero as the dimension of the Arnoldi approximation increases (Boley (1994)).
Similar techniques have been proposed for the more general Sylvester equation $AX - XB = C$ in (Hu and Reichel (1992)), in which the approximate solutions can be recursively generated as the Arnoldi process advances. These ideas have even been proposed as a method for solving large order Riccati equations (Hodel and Poolla (1988)), but the theory still needs further development.

REFERENCES

Aliaga J., D. Boley, R. Freund, and V. Hernández, (1996). *A Lanczos-type method for multiple starting vectors*, (Report in preparation).

Arnold M. (1993). *Algorithms and Conditioning for Eigenvalue Assignment.* Ph.D. Dissertation, Northern Illinois University.

Arnold M. and B.N. Datta. (1990). *An algorithm for the multi-input eigenvalue assignment problem.* IEEE Trans. Automat. Control 35(10):1149-1152.

Arnold M. and B.N. Datta. (1995). *The single-input eigenvalue assignment algorithm : A close-look.* To appear in SIAM J. Matrix Anal. Appl.

Arnoldi W.E. (1951).*The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 , pp. 17–29.

Arnold, W.F. and A.J. Laub (1984) 'Generalized eigenproblem algorithms and software for algebraic Riccati equations', *Proc. IEEE,* 72, pp. 1746-54.

Bhattacharyya, S.P. and E. DeSouza. (1982). *Pole Assignment via Sylvester's Equation.* Syst. and Contr., vol. 1,no. 4,261-263.

Bischof C.H., B.N. Datta and A. Purkayastha. (1996). *A Parallel Algorithm for the Sylvester Observer Equation.* SIAM J. Scientific Computing, 17, 686-698.

Boley D.L. and G.H. Golub (1984). *The Lanczos algorithm and controllability*, Systems & Control letters, 4, 317-324.

Boley D.L. (1994). *Krylov space methods on state-space control models*, Circ. Syst. & Signal Proc., 13 (1994), pp. 733–758.

Byers R. (1984) 'A LINPACK-style condition estimator for the equation $AX - XB^T = C$', *IEEE Trans. Autom. Control*, AC-29, pp. 926-8.

Byers R. (1985) 'Numerical condition of the algebraic Riccati equation', *Contemporary Math.*, 47, pp. 35-49.

Byers R. (1986) 'A Hamiltonian QR algorithm', *SIAM J. Sci. Stat. Comput.*, 7, pp. 212-29.

Chen C.T. (1984). **Linear System Theory and Design**. CBS College Publishing, New York.

Chu, E. and B.N. Datta (1995). *Numerical Robust Pole Placement for Second Order Systems.* To appear in Int. J. Control.

Cox C.L. and W.F. Moss (1989).*Backward error analysis for a pole assignment algorithm*, SIAM J. Matrix Anal. Appl. 10, 446-456.

Datta B.N. (1997). **Numerical Methods for Linear Control Systems Design and Analysis**, to be published by Academic Press.

Datta B.N. (1987). *An Algorithm to Assign Eigenvalues in a Hessenberg Matrix: Single-Input Case.* IEEE Trans. Auto. Control, vol. AC-32, no. 5, 414-417.

Datta B.N. (1994). *Linear and Numerical Linear Algebra in Control Theory: Some Research Problems.* Linear Algebra and Its Applications 197, 198:755-790.

Datta B.N. (1995). *Numerical Linear Algebra and Applications.* Brooks/Cole Publishing Company.

Datta B.N. and C. Hetti (1996). *An Arnoldi-type method for the Sylvester-observer equation*, to appear in the proceedings of the European Control Conference.

Datta B.N. and F. Rincon (1993). *Feedback Stabilization of the Second-Order Model: A Nonmodal Approach*, Lin Alg. Appl., 188, 138-161.

Datta B.N. and Y. Saad. (1991). *Arnoldi Methods for Large Sylvester-Like Observer Matrix Equations, and an Associated Algorithm for Partial Spectrum Assignment.* Linear Algebra and its Applications, 154-156:225-244.

Datta, B.N. and K. Datta (1987). *A Hessenberg method for the positive semidefinite Lyapunov equation*, Proc. 26th IEEE Conference on Decision and Control.

Datta, B.N., S. Elhay, and Y. Ram (1996). *An algorithm for the multi-input pole placement of second-order system*, Proc. IEEE Conf. Decision and Control.

Datta, B.N., S. Elhay, and Y. Ram (1995). *Orthogonality and Partial Pole Placement for a Quadratic Pencil.* Linear Algebra and Applications. (To appear).

Eising, R. (1984b). Between controllable and uncontrollable, *Systems & Control Lett.*, 4, pp. 263-4.

Elsner, L. and C. He, (1991). *An algorithm for computing the distance to uncontrollability*, Systems & Control Letters, 17, 453-464.

P. Feldmann and R. Freund, (1995)*Efficient linear circuit analysis by Padé approximation via the Lanczos process*, IEEE Trans. CAD of Integr. Circ. and Syst., 14 , pp. 639-649.

Golub G., S. Nash and C. Van Loan. (1979). *A Hessenberg Schur method for the problem $AX + XB = C$.*IEEE Transactions in Automatic Control, 24(6) : 209-213.

Golub G. and C. Van Loan. (1989). **Matrix Computations**. The Johns Hopkins University Press.

Hammarling, S.J. (1982) 'Numerical solution of the stable, nonnegative definite Lyapunov equation', *IMA J. Numer. Anal.*, 2, pp. 303-23.

He C., A. Laub, and V. Mehrmann, (1995) *Placing plenty of poles is pretty preposterous*, IEEE Trans. Automatic Control, to appear (Technical Report).

Hetti, C. (1996). On Numerical Solutions of the Sylvester-observer equation and The Multi-input Eigenvalue Assignment Problem, Ph.D. Dissertation, Northern Illinois University, 1996.

Higham N.J. (1996). *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia.

Hodel A. and K. Poolla, (1988). *Heuristic approaches to the solution of very large sparse Lyapunov and algebraic Riccati equations*, in 27th IEEE Conf. Dec. Contr., 1988, pp. 2217–2222.

Hu D.Y. and L. Reichel. (1992). *Krylov Subspace Methods for the Sylvester Equation.* Lin. Alg. Appl. and Appl. 172:283 - 313.

Jaimoukha I.M. and E. M. Kasenally, (1994) *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31 , pp. 227–251.

Kailath, T. (1980). **Linear systems**, Prentice Hall, Inc. Englewood Cliffs, N.J.

Kautsky J., Nichols N.K. and P. Van Dooren. (1985). *Robust pole assignment in linearstate feedback.* Int. J. Control, vol. 41, no. 5, 1129-1155.

Kim H.M. and R. R. Craig Jr. (1988). *Structural dynamics analysis using an unsymmetric block Lanczos algorithm*, International Journal for Numerical Methods in Engineering, 26 (1988), pp. 2305–2318.

Kim, H.M. and R.R. Craig Jr. (1990). *Computational enhancement of an Unsymmetric block Lanczos algorithm*, Int. J. Numerical Methods in Engineering, 30, pp. 1083-1089.

Lanczos C.(1950). *An iteration method for the solution of the eigenvalue problem linear differential and integral operators*, J. Res. Natl. Bur. Stand., 45 , pp. 255–282.

Laub A.J. (1979). A Schur method for solving algebraic Riccati equations, *IEEE Trans. Autom. Control*, AC-24, pp. 913-21.

Miminis G.S. and C.C. Paige (1988). A direct algorithm for pole assignment of time-invariant multi-input linear systems using state feedback, *Automatica*, 24, pp. 343-56.

Moler, C., and C. Van Loan (1978). *Nineteen dubious ways to compute the exponential of a matrix*, SIAM Review, 20, 801-836.

Paige C.C. (1981). *Properties of Numerical Algorithms Related to Computing Controllability.* IEEE Trans. Aut. Control, AC-26, 130-138.

Parlett B.N (1992). *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matr. Anal., 13 , pp. 567–593.

Patel R.V. and P. Misra (1984). *Numerical algorithms for eigenvalue assignment by state feedback*, Proc. IEE, 72, 1755-1764.

Patel, R.V., A. Laub, and P. Van Dooren, (1993). *Numerical Linear Algebra Techniques for Systems and Control*, IEEE Press Piscataway, N.J. (Reprint volume).

Petkov P.Hr., N.D. Christov, and M.M. Konstantinov. (1984). *A computational algorithm for pole assignment of linear single-input systems.* IEEE Trans. Autom. Control, AC-29: pp. 1045-1048.

Petkov P.Hr., N.D. Christov, and M.M Konstantinov (1991). **Computational Methods for Linear Control Systems**, Prentice Hall International, U.K.

Roberts, J.D. (1980) Linear model reduction and solution of the algebraic Riccati equation by use of the sign function, *Int. J. Control*, 32, pp. 677-87.

Saad Y. (1990). *Numerical Solutions of Large Lyapunov Equations. In Signal Processing, Scattering, Operator Theory, and Numerical Methods*, edited by M.A. Kaashoek, J. H. Van Schuppen and A. C. Ran, 503-511. Boston:Birkhauser.

Saad Y. and M. H. Schultz (1986). *GMRES: A generalized minimal residual algorithm for solving unsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7, pp. 856–869.

Stewart, G.W. (1976) *Algorithm 506: HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix*, *ACM Trans. Math. Software*, 2, pp. 275-80.

Van Dooren P. (1981). *A generalized eigenvalue approach for solving Riccati equations.* SIAM J. Sci. Stat. Comput, 2, 121-135.

Varga A. (1981). *A Schur method for pole assignment.* IEEE Trans. Autom. Control, AC-26:517-519.

Walker H.F. (1988). *Implementation of the GMRES method using Householder transformations.* SIAM J. Sci. Stat. Comput., 9:152-163.

Wilkinson J.H. (1965). **The Algebraic Eigenvalue Problem**. Oxford University Press, Oxford.