July, 1978

ESL-P-833

LINEAR MULTIVARIABLE CONTROL. NUMERICAL CONSIDERATIONS.

by

Alan J. Laub

This research was partially supported by the Department of Energy, Division of Electric Energy Systems under grant ERDA-E(49-18)-2087.

LINEAR MULTIVARIABLE CONTROL. NUMERICAL CONSIDERATIONS.*

Alan J. Laub
Electronic Systems Laboratory
Room 35-331
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Outline

*Prepared for presentation at the American Mathematical Society Short Course on Control Theory, Providence, R.I., August 6-7, 1978.

0. Introduction

In this lecture we shall address some numerical issues arising in the study of models described by the linear system

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$z(t) = Dx(t) \ .$$

Here x is an n-vector of states, u is an m-vector of controls or inputs, and z is an r-vector of variables (outputs) to be regulated (to zero). This is one of the simplest models of a family of considerably more elaborate models. The interested reader is referred to [1] for an extended treatment and to [2] for a recent survey of geometric methods in linear multivariable control.

The emphasis here will be on numerical considerations pertinent to some of the theory discussed by Professor Wonham in an earlier lecture. Many important control and systems problems have been solved, frequently very elegantly, by this theory. But while these solutions are, for the most part, well-understood theoretically, very little is understood about their implementation on a digital computer. Indeed, the same can be said about almost all control/systems theory today.

Most of the difficulties in computation on a digital computer derive from the inherently finite word length used to represent real or complex numbers. While this may be the source of great frustration to the average user of computing facilities, it does provide an essentially limitless supply of challenging problems for the numerical analyst. Some very convenient mathematical properties that we sometimes take for granted in hand computation are no longer valid on a computer. For example, the associative law for addition of real numbers is no longer generally true. The interested reader is strongly urged to consult [3] for a brief introduction to the vagaries of floating point computation and [30] for the definitive treatment.

A particular number of which we shall make frequent use in the sequel is <u>machine epsilon</u>. This is defined to be the smallest positive number ε which, when added to 1 on our computing machine, gives a number greater than 1. In other words, any machine-representable number δ less than ε gets "rounded off" when added to 1 to give exactly 1 again as the rounded sum. The number ε varies, of course, depending on the kind of computer being used and the precision with which the computations are being done (single precision, double precision, etc.). But the fact that there exists such a positive number ε is entirely a consequence of finite word length.

While the problems to be discussed here have measure zero with respect to all reasonably interesting control and systems problems, they are representative in the sense of being both amenable to solution by some current methods of numerical analysis and of suggesting exciting new avenues of research. The development, in control and systems theory, of stable, efficient, and reliable algorithms and their embodiment in robust mathematical software is only now in its infancy. The situation is analogous to that which existed with respect to the algebraic eigenvalue problem in the 1950's. The Jordan canonical form was theoretically well-understood. But no one really knew much about actually computing eigenvalues and eigenvectors on a digital computer.[1] It is really only rather recently that some of the computational issues have been resolved or even understood. Some of the pivotal theoretical work was done by J. H. Wilkinson (see [4] for the state of the art in 1964) and quality mathematical software has only arrived on the scene in the 1970's (see [5],[6]). It has now even been demonstrated by Golub and Wilkinson [7] that the Jordan canonical form for general matrices cannot reliably be computed <u>in the presence of roundoff error</u>. While this comes as a bit of a surprise to some people, solace can be found in the fact that seldom in real computations is the Jordan canonical form itself really needed but rather

[1]A notable exception was the eigenvalue problem for real symmetric matrices where a great deal was known even in 1952 about Sturm sequence properties in the method of bisection. See [31] for details.

other more easily computed information may often be substituted. It is likely that the more stably computed Schur (upper triangular) canonical form (see, e.g., [7]) will replace the Jordan canonical form as a working tool in control and systems theory. The Jordan canonical form is even now only an appendix (albeit a crucially important one) in a recent book by Strang [8], one of the best sophomore-level books presently available on the topic of linear algebra. While it is a slow process, we are now just beginning to see some of the material (well-known to numerical analysts) presented in this lecture filter down to the undergraduate curriculum in mathematics and engineering. This process is certain to have a significant impact on the future directions and development of control and systems theory and applications.

Finally, let us close this introductory diatribe with a friendly, facetious, folk theorem: "If an algorithm can be used "easily" by hand, it's probably a poor method when implemented on a digital computer." For example, when confronted with the matrix $\begin{pmatrix} 2 & 1 \\ -1 & 4 \end{pmatrix}$ most people would find the characteristic polynomial and solve the resulting quadratic equation. But when implemented on a digital computer this turns out to be a very poor method for a variety of reasons (such as roundoff and overflow/underflow). Of course the preferred method now would generally be the QR algorithm (see [4],[5] for the messy details) but few of us would attempt that by hand -- even for 2×2 problems. It suffices to say that modern computer- and software-oriented numerical analysis has made great strides in the past ten or fifteen years and one would be well-advised to avail oneself, if possible, of this research before attempting any serious numerical computing.

Before proceeding any further we shall standardize here some notation to be used in this paper.

$\mathbb{F}^{n\times m}$	the set of all nxm matrices with coefficients in the field $\mathbb{F}$ ($\mathbb{F}$ will generally be $\mathbb{R}$ or $\mathbb{C}$)
$\mathbb{F}_r^{n\times m}$	the set of all nxm matrices of rank r with coefficients in the field $\mathbb{F}$
A^T	the transpose of $A \varepsilon \mathbb{R}^{n\times m}$
A^H	the conjugate transpose of $A \varepsilon \mathbb{C}^{n\times m}$
A^+	the Moore-Penrose pseudoinverse of A
$\|A\|$	the spectral norm of A (i.e., the matrix norm subordinate to the Euclidean vector norm: $\|A\| = \max_{\|x\|_2=1} \|Ax\|_2$)
$\text{diag}(a_1,\ldots,a_n)$	the diagonal matrix $\begin{pmatrix} a_1 & & 0 \\ & \ddots & \\ 0 & & a_n \end{pmatrix}$

1. Numerical Stability and Conditioning

In this section we give a brief discussion of two concepts of fundamental importance in numerical analysis: numerical stability and conditioning. While this material is standard in introductory textbooks such as [3], [9], or [10] and was covered in the AMS Short Course on Numerical Analysis in Atlanta in January 1978 we present it here both for completeness and because the two concepts are frequently confused in the control/systems literature.

Suppose we have some mathematically defined problem represented by f which acts on data $d \varepsilon \mathscr{D}$= some set of data to produce a solution $f(d) \varepsilon \mathscr{S}$= some set of solutions. These notions are kept deliberately vague for expository purposes. Given $d \varepsilon \mathscr{D}$ we desire to compute f(d). Frequently, only an approximation d* to d is known and the best we could hope for is to calculate f(d*). If f(d*) is "near" f(d) the <u>problem</u> is said to be <u>well-conditioned</u>. If f(d*) may potentially differ greatly from f(d) even when d* is near d, the problem is said to be ill-conditioned. Again the concept "near" cannot be made precise without further information about a particular problem.

Let f* denote the algorithm implemented to solve f. Given d, f*(d) represents the approximate computed solution. The <u>algorithm</u> f* is said to be <u>numerically stable</u> if for all $d \varepsilon \mathscr{D}$, there exists $d^* \varepsilon \mathscr{D}$ near d such that f(d*) (= the exact solution of a nearby problem) is near f*(d).

Of course, one can't expect a stable algorithm to solve an ill-conditioned problem any more accurately than the data warrant but an unstable algorithm can produce poor solutions even to well-conditioned problems. There are thus two separate factors to consider in determining the accuracy of a computed solution f*(d). First, if the algorithm is stable f*(d) is near f(d*) and second, if the problem is well-conditioned f(d*) is near f(d). Thus f*(d) is near f(d). Further details and examples may be found in [9].

Roundoff errors can cause unstable algorithms to give disastrous results. However, it would be virtually impossible to account for every roundoff error made at every arithmetic operation of a complex calculation. This would constitute a forward error analysis. To account for these errors, however, J. H. Wilkinson developed, to a fine degree, the notion of backward error analysis. Namely, for many problems, it is possible to show that what is actually computed is the exact solution of a nearby problem. One then attempts to show that the nearby problem is near enough which, if the problem is well-conditioned, can be translated into a quantitative statement regarding the accuracy of the solution.

For example, in the QR algorithm for finding the eigenvalues of a matrix A it can be proved that the computed eigenvalues are the exact eigenvalues of the matrix A + E where $\|E\| \leq c \cdot \|A\| \cdot \varepsilon$ (c = a modest constant involving the order of the matrix, $\|\cdot\|$ = an appropriate matrix norm, ε = machine epsilon). This is the statement of stability for the QR algorithm.

2. Singular Value Decomposition and Numerical Rank

One of the basic and most important tools of modern numerical analysis, particularly numerical linear algebra, is the singular value decomposition. We shall define it here and make a few comments about its properties and computation. In section 3 we shall see how the SVD can be used to reliably compute a number of the basic geometric objects of linear algebra. This is but one of many fields of application and it is likely that within five or ten years SVD will be one of the most important and fundamental working tools for the control/systems community, particularly in the area of linear systems.

We now state the SVD theorem. Square brackets will denote the complex case.

THEOREM 1: Let $A \in \mathbb{R}_r^{n\times m}$ $[\mathbb{C}_r^{n\times m}]$. Then there exist orthogonal [unitary] matrices $U \in \mathbb{R}^{n\times n}$ $[\mathbb{C}^{n\times n}]$ and $V \in \mathbb{R}^{m\times m}$ $[\mathbb{C}^{m\times m}]$ such that

$$A = U\Sigma V^T \quad [U\Sigma V^H]$$

where $\Sigma = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}$ and $S = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$ with

$\sigma_1 \geq \ldots \geq \sigma_r > 0$. ■

The proof of Theorem 1 is straightforward and can be found in, for example [9],[32]. The numbers $\sigma_1, \ldots, \sigma_r$ together with $\sigma_{r+1} = 0, \ldots, \sigma_m = 0$ are called the singular values of A and they are the positive square roots of the eigenvalues of $A^TA\,[A^HA]$. The columns of U are called the left singular vectors of A (the orthonormal eigenvectors of AA^T $[AA^H]$) while the columns of V are called the right singular vectors of A (the orthonormal eigenvectors of A^TA $[A^HA]$). The matrix $A^T[A^H]$ has n singular values, the positive square roots of the eigenvalues of AA^T $[AA^H]$. The r (= rank (A)) nonzero singular values of A and A^T $[A^H]$ are, of course, the same. The choice of A^TA $[A^HA]$ rather than $AA^T[AA^H]$ in the definition of singular values is arbitrary. Only the potentially nonzero singular values will be of any real interest.

It is not generally a good idea to compute the singular values of A by first finding the eigenvalues of A^TA (remember the F^3 Theorem), tempting as that is. Consider the following example with μ a real number with $|\mu| < \sqrt{\varepsilon}$ (so that $fl(1+\mu^2) = 1$ where $fl(\cdot)$ denotes floating point computation). Let $A = \begin{pmatrix} 1 & 1 \\ \mu & 0 \\ 0 & \mu \end{pmatrix}$. Then $fl(A^TA) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ so we compute $\hat{\sigma}_1 = \sqrt{2}$, $\hat{\sigma}_2 = 0$ leading to the (erroneous) conclusion that the rank of A is 1. Of course, if we could compute in infinite precision we would find $A^TA = \begin{pmatrix} 1+\mu^2 & 1 \\ 1 & 1+\mu^2 \end{pmatrix}$ with $\sigma_1 = \sqrt{2+\mu^2}$, $\sigma_2 = |\mu|$ and thus rank (A) = 2. The point is that by working with A^TA we have unnecessarily introduced μ^2 into the computations.

Fortunately, Golub and Reinsch [11] have developed an extremely efficient and stable algorithm for computing the SVD which does not suffer from the above defect. The computed U and V are orthogonal to approximately the working precision and the computed singular values can be shown to be the exact σ_i's for $A+E$ where $\frac{\|E\|}{\|A\|}$ is a modest multiple of ε. A fairly sophisticated implementation of this algorithm can be found in [6]. A word of warning! There are other SVD subroutines around and many have severe problems even if coded directly from [11]. One is probably best off using [6] or the version implemented in the forthcoming LINPACK from Argonne National Laboratories [12].

It is clear from the definition that the number of nonzero singular values of A determines its rank. While the question is not nearly as clear-cut in the context of computation on a digital computer, it is now generally acknowledged that the singular value decomposition is the only generally reliable method of determining rank numerically (see [7] for a more elaborate discussion).

Again, only rather recently has the problem of numerical determination of rank been well-understood. One of the best treatments of the subject, including a careful definition of numerical rank, is a paper by Golub, Klema, and Stewart [13]. The essential idea is as follows. We are going to look at the "smallest nonzero singular value" of a matrix A. Since that computed value is exact for a matrix near A it makes sense to consider the rank of all matrices in some δ-ball (w.r.t. the spectral norm $\|\cdot\|$, say) around A. The choice of δ may also be based on measurement errors incurred in estimating the coefficients of A or the coefficients may be uncertain because of roundoff errors incurred in a previous computation to get them. See [13] for further details.

In any case it can easily be shown that all matrices B lying strictly inside the σ_r-ball around A have rank $\geq$ r. The matrix $B = U\hat{\Sigma}V^T$ where $\hat{\Sigma} = \begin{pmatrix} \hat{S} & 0 \\ 0 & 0 \end{pmatrix}$ with $\hat{S} = \text{diag}(\sigma_1,\ldots,\sigma_{r-1})$ is a matrix with rank r-1 and $\|B-A\| = \sigma_r$. Thus if we choose as some "zero threshold" a number $\delta < \sigma_r$, we will consider A to have numerical rank r. There can sometimes be real difficulties in determining a "gap" between the computed last nonzero singular value and what should effectively be considered "zero". Further details are found in [13].

The key quantity in rank determination is obviously σ_r. Moreover, this number gives a dependable measure of how far (in the $\|\cdot\|$ sense) a matrix is from matrices of lesser rank. But σ_r alone is clearly sensitive to scale so that a better measure is $\frac{\sigma_r}{\|A\|}$. But $\|A\| = \sigma_1$ so the important quantity is $\frac{\sigma_r}{\sigma_1}$ which turns out to be the reciprocal of the number

$\kappa(A) = \|A\| \cdot \|A^{+}\|$, the so-called "condition number of A w.r.t. pseudo-inversion". In the case when A is invertible, $\kappa(A) = \|A\| \cdot \|A^{-1}\|$ is the usual spectral condition number w.r.t. inversion. The use of $\kappa(A)$ as a condition number in the general case is actually somewhat more complicated. For a discussion of this and related matters the reader is urged to consult a nice survey paper by Stewart [14].

In solving the linear system $Ax = b$, the condition number $\kappa(A) = \|A\| \cdot \|A^{-1}\|$ gives a measure of how much errors in A and/or b may be magnified in the computed solution. Moreover, if $A \in \mathbb{R}_n^{n\times n}$, $\sigma_n(A)$ gives a measure of the "nearness" of A to singularity. A standard introduction to the solution of linear systems is the classical book of Forsythe and Moler [15] to which we further refer the interested reader.

That other methods of rank determination are potentially unreliable is demonstrated by the following example which is a slight modification of a classical example due to Kahan. Consider the matrix $A \in \mathbb{R}^{n\times n}$ whose diagonal elements are all -1, whose upper triangle elements are all +1, and whose lower triangle elements are all 0. This matrix is clearly of rank n, i.e., is invertible. It has a good "solid" upper triangular shape. All of its eigenvalues (all = -1) are well away from zero. Its determinant is $(-1)^n$-- definitely not close to zero. But this matrix is, in fact, very near singular and gets more nearly so as n increases. Notice, for example, that

$$\begin{pmatrix} -1 & +1 & \cdots & +1 \\ & \ddots & \ddots & \vdots \\ & 0 & \ddots & +1 \\ & & & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1/2 \\ \vdots \\ 1/2^{n-1} \end{pmatrix} = \begin{pmatrix} -1/2^{n-1} \\ -1/2^{n-1} \\ \vdots \\ -1/2^{n-1} \end{pmatrix} \sim \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Moreover, adding $\frac{1}{2^{n-1}}$ to every element in the first column of A gives an exactly singular matrix. Arriving at such a matrix by, say Gaussian elimination, would give no hint as to the near-singularity. However, it is easy to check that $\sigma_n(A)$ behaves as $\frac{1}{2^n}$. A corollary for control theory: eigenvalues don't necessarily give a reliable measure of "stability margin".

Rank determination, in the presence of roundoff error, is a highly nontrivial problem. And, of course, all the same difficulties arise in any problem equivalent to or involving rank determination such as determining the independence of vectors, finding a basis for Ker A, etc. We turn now to some of these problems which naturally arise in the geometric theory of linear multivariable control.

3. Calculation of the Basic Geometric Objects

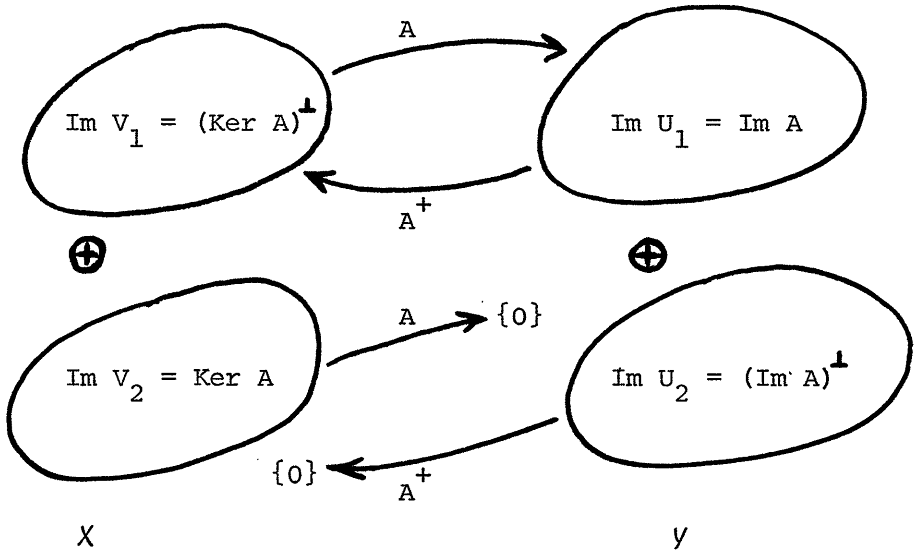
In this section we outline the application of singular value techniques to the computation of various subspaces and maps that arise in the geometric theory. No proofs will be given nor will the list of objects considered be exhaustive. Rather we shall attempt to impart only the flavor of singular value analysis to the subject, leaving technical matters and the details of software implementation for consideration elsewhere. Notation used will be consistent with Wonham [1].

3.1 The four fundamental subspaces

Consider a linear map A: $\mathcal{X} \to \mathcal{Y}$ between two finite-dimensional real vector spaces $\mathcal{X}$ and $\mathcal{Y}$ with $d(\mathcal{X}) = m$, $d(\mathcal{Y}) = n$. Identifying A with an nxm real matrix A suppose A has an SVD given by

$$A = U \Sigma V^T = (U_1, U_2) \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}$$

where $S = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ with $\sigma_1 \geq \ldots \geq \sigma_r > 0$ and U and V are partitioned compatibly (U_1 is n x (n-r), etc.). Then the U_i and V_i provide orthonormal bases for the four fundamental subspaces [8] in the following self-explanatory diagram:


A
$\mathrm{Im}\, V_1 = (\mathrm{Ker}\, A)^\perp$
$\mathrm{Im}\, U_1 = \mathrm{Im}\, A$
A^+
$\oplus$
$\oplus$
A
{0}
$\mathrm{Im}\, V_2 = \mathrm{Ker}\, A$
$\mathrm{Im}\, U_2 = (\mathrm{Im}\, A)^\perp$
{0}
A^+
$\mathcal{X}$
$\mathcal{Y}$

As discussed in Section 2 we have the computational problem of intelligently deciding what is σ_r and hence the rank of A. But that decision directly affects each of the above subspaces and hence their calculation. Since SVD is the only generally reliable way of calculating rank it follows that it is the only generally reliable way of calculating bases for these subspaces.

3.2 Projections

The four fundamental orthogonal projections are given by:

$$\begin{aligned}
U_1U_1^T &= P_{\operatorname{Im}A} &&= AA^+ \\
U_2U_2^T &= P_{\operatorname{Ker}A^T} &&= I-AA^+ \\
V_1V_1^T &= P_{\operatorname{Im}A^T} &&= A^+A \\
V_2V_2^T &= P_{\operatorname{Ker}A} &&= I-A^+A
\end{aligned}$$

For the case of oblique projections, suppose $\operatorname{Im} R = \mathcal{R}$ and $\operatorname{Im} S = \mathcal{S}$ where $\mathcal{R}$ and $\mathcal{S}$ are two subspaces of $\mathcal{X}$ such that $\mathcal{R} \oplus \mathcal{S} = \mathcal{X}$. Notice that R and S need not have linearly independent columns. Then with the obvious choice of notation we have for the projection on $\mathcal{R}$ along $\mathcal{S}$

$$P_{\mathcal{R},\mathcal{S}} = (U_{1R},0)(U_{1R},U_{1S})^{-1}$$

with similar expressions for $P_{\mathcal{S},\mathcal{R}}$, $P_{\mathcal{R},\mathcal{S}^\perp}$, etc. Notice that in the special case of $\mathcal{S} = \mathcal{R}^\perp$ we do indeed have

$$\begin{aligned}
P_{\mathcal{R},\mathcal{R}^\perp} &= (U_{1R},0)(U_{1R},U_{2R})^{-1} \\
&= (U_{1R},0)\begin{pmatrix} U_{1R}^T \\ U_{2R}^T \end{pmatrix} \\
&= U_{1R}U_{1R}^T = RR^+ = P_{\operatorname{Im}R} = P_{\mathcal{R}}.
\end{aligned}$$

3.3 The factoring of a linear map

Suppose A: $\mathcal{X} \to \mathcal{Y}$, $\mathcal{R} \subseteq \operatorname{Ker} A \subseteq \mathcal{X}$ (here $\subseteq$ denotes "is a subspace of"). Let P: $\mathcal{X} \to \frac{\mathcal{X}}{\mathcal{R}}$ be the canonical projection. Then there exists a unique linear map Q: $\frac{\mathcal{X}}{\mathcal{R}} \to \mathcal{Y}$ such that the following diagram commutes

$$\begin{tikzcd} \mathcal{X} \arrow[r, "P"] \arrow[dr, "A"'] & \frac{\mathcal{X}}{\mathcal{R}} \arrow[d, dashed, "Q"] \\ & \mathcal{Y} \end{tikzcd}$$

To compute P and Q, suppose Im R = $\mathcal{R}$ and let R have SVD

$$R = (U_1, U_2)\begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}.$$

Then P is given by $P = U_2^T$ while the induced Q is given by $Q = AU_2$. Note that Ker P = $\mathcal{R} \subseteq \operatorname{Ker} A$ is equivalent to $ARR^+ = 0$ whence $A = A(I-RR^+) = AU_2U_2^T = QP$. In the case when $\mathcal{R} = \operatorname{Ker} A$ we have $AU_1 = 0$ and $A = QP = (AU_2)\cdot U_2^T$ factors A into the product of injective and surjective maps.

3.4 The induced map in the factor space

Suppose A: $\mathcal{X} \to \mathcal{X}$, and $\mathcal{R} \subseteq \mathcal{X}$ is A-invariant, i.e., $A\mathcal{R} \subseteq \mathcal{R}$. Then A induces a unique endomorphism $\bar{A}$ of the factor space $\frac{\mathcal{X}}{\mathcal{R}}$ which makes the following diagram commute

$$\begin{tikzcd} \mathcal{X} \arrow[r, "A"] \arrow[d, "P"'] & \mathcal{X} \arrow[d, "P"] \\ \frac{\mathcal{X}}{\mathcal{R}} \arrow[r, dashed, "\bar{A}"] & \frac{\mathcal{X}}{\mathcal{R}} \end{tikzcd}$$

P = the canonical projection

Again suppose R is any matrix with $\mathrm{Im}R = \mathcal{R}$ and let R have SVD $R = (U_1,U_2)\begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}$. Then P is given by U_2^T while $\bar{A}$ is given by $\bar{A} = U_2^T A U_2$. Note that $A\mathcal{R} \subseteq \mathcal{R}$ is equivalent to

$$RR^+AR = AR$$

which implies $U_2^T AU_1 = 0$ and $U_2^T A\,(I-U_2U_2^T) = 0$. Thus $\bar{A}P = U_2^T AU_2U_2^T = U_2^T A = PA$. Also note that under the orthogonal change of basis U, A becomes

$$U^TAU = \begin{pmatrix} U_1^TAU_1 & U_1^TAU_2 \\ 0 & U_2^TAU_2 \end{pmatrix} = \begin{pmatrix} A|\mathcal{R} & U_1^TAU_2 \\ 0 & \bar{A} \end{pmatrix}.$$

3.5 Subspaces

Suppose $\mathcal{R} \subseteq \mathcal{X}$ and $A: \mathcal{X} \to \mathcal{X}$. Let R be any matrix with $\mathrm{Im}\ R = \mathcal{R}$ and let the SVD of R be as above. Then $d(\mathcal{R})$ = the number of nonzero singular values of R and the columns of U_1 give an orthonormal basis for $\mathcal{R}$ while the columns of U_2 give an orthonormal basis for $\mathcal{R}^\perp \approx \left(\frac{\mathcal{X}}{\mathcal{R}}\right)'$. A basis for $A\mathcal{R}$ can be obtained from the U_1 of the SVD of AR. To get a basis for $A^{-1}\mathcal{R}$ we need the SVD of A. Then

$$\begin{aligned} A^{-1}\mathcal{R} &= \mathrm{Im}\ A^+R + \mathrm{Ker}\ A \\ &= \mathrm{Im}\ [V\Sigma^+ U^TR,\ V_2] \\ &= \mathrm{Im}\ [V_1S^{-1}U_1^TR,\ V_2] \end{aligned}$$

3.6 The calculus of subspaces

Given two subspaces $\mathcal{R}$ and $\mathcal{T}$ of $\mathcal{X}$ it must frequently be determined if $\mathcal{R} \subseteq \mathcal{T}$. Suppose $\mathrm{Im}\ R = \mathcal{R}$ and $\mathrm{Im}\ T = \mathcal{T}$. Then $\mathcal{R} \subseteq \mathcal{T}$ if and only if $TT^+R = R$ so using U_1 from the SVD of T one can check if $U_1U_1^TR = R$.

Alternatively (but more expensively) one can check if the number of nonzero singular values of T equals the number of nonzero singular values of the matrix [R,T]. To check if $\mathcal{R} = \mathcal{T}$ both $\mathcal{R} \subseteq \mathcal{T}$ and $\mathcal{T} \subseteq \mathcal{R}$ can be verified.

Upon computing the SVD of [R,T], the columns of U_1 are an orthonormal basis for $\mathcal{R} + \mathcal{T}$ while the columns of U_2 are an orthonormal basis for $\mathcal{R}^{\perp} \cap \mathcal{T}^{\perp}$. The columns of V_1 and V_2 do not appear to span anything particularly interesting. The extension of this procedure to arbitrary finite sums of subspaces is obvious.

The "dual calculation" of $\mathcal{R} \cap \mathcal{T}$ is not quite so easy. One can use $\mathcal{R} \cap \mathcal{T} = (\mathcal{R}^{\perp} + \mathcal{T}^{\perp})^{\perp}$ and do two SVD's to get bases for $\mathcal{R}^{\perp}, \mathcal{T}^{\perp}$ then proceed as above with one further SVD. This procedure extends obviously to arbitrary finite intersections of subspaces but is clearly expensive. An alternate procedure (but only for the case of two subspaces) is to do an SVD of [R,T]. Then from the partitioning

$$0 = [R,T]V_2 = [R,T]\begin{pmatrix} V_{2R} \\ V_{2T} \end{pmatrix}$$

choose RV_{2R} or TV_{2T} (according as $d(\mathcal{R})$ or $d(\mathcal{T})$ is smaller) as the appropriate basis.

4. Transmission Zeros

We now turn to the numerical determination of the transmission zeros of the system

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$z(t) = Dx(t).$$

For ease of exposition we shall consider here only the "square" case, i.e., the case where u and z are both m-vectors. A complete discussion of this and related problems (including the "nonsquare" case) can be found in [16]. We shall simply highlight the results of that paper.

As we have already seen in Professor Wonham's lecture, the regulator problem with internal stability is well-posed if and only if no internal system transmission zeros coincide with any exosystem poles. Transmission zeros also play a role in other areas of regulation, decoupling, and servomechanism design. They are essentially the multivariable analogue of the numerator zeros of the classical single-input, single-output transfer function. Very roughly speaking these zeros are certain complex frequencies at which "transmission" through the system may be blocked.

The problem can be solved very straightforwardly by an application of the famous QZ algorithm [17]. This algorithm has the advantages of being efficient, numerically stable, and, most important, widely available in a reliable implementation [6]. To varying degrees, this is in definite contrast to various other proposed algorithms. A number of examples illustrating the potential difficulties with other approaches are given in [16].

The set $\mathcal{T}$ of transmission zeros of the system described above is the set of complex numbers λ (including multiplicities) such that $\det(L-\lambda M) = 0$ where

$$L = \begin{pmatrix} A & B \\ D & 0 \end{pmatrix} , \quad M = \begin{pmatrix} I_n & 0 \\ 0 & 0_m \end{pmatrix} .$$

The definition becomes somewhat more complicated in the nonsquare case (in which generically there are no transmission zeros anyway) and we refer the reader to [16] and [18] for details. For the square case considered here, we thus want to solve the generalized eigenvalue problem: Find all finite λ for which there exist nontrivial solutions for the equation

$$Lz = \lambda Mz.$$

The generalized eigenvectors z corresponding to $\lambda \varepsilon T$ will play a role in computing bases for supremal (A,B)-invariant and controllability subspaces (see Section 5).

The theoretical aspects of this and the analogous "rectangular" problem have been studied extensively. But the first definitive numerical treatment which squarely addressed the problem of singular M was published by Moler and Stewart in 1973 [17]. Their algorithm is based on the following theorem [19] which helps explain its desirable numerical properties.

THEOREM 2: There exist unitary matrices Q and Z such that QLZ and QMZ are both upper triangular. ■

Error and stability analysis of the QZ algorithm as well as a notion of condition number for the generalized eigenvalue problem are developed in [17], [19], and [20].

Since unitary transformations are used the computed generalized eigenvalues are the exact generalized eigenvalues of the "slightly perturbed" problem $(L+G)z = \lambda(M+H)z$ where G and H are perturbation matrices whose norms can usually be bounded by a modest multiple of the machine precision (ε). Moreover, all well-conditioned generalized eigenvalues are computed accurately independently of the singularity of M.

The QZ algorithm does not actually compute the λ_i but rather determines α_i and β_i, the diagonal elements of QLZ and QMZ, respectively. All the important information in the problem is contained in the α_i and the β_i and it is our responsibility as users to judiciously compute the λ_i from them. For example, if the elements of L are determined experimentally and are known exactly only to within, say 10^{-3}, then, since we are using unitary transformations, we may wish to call any λ_i corresponding to a $\beta_i < 10^{-3}$ and $\alpha_i \geq 10^{-3}$ an infinite generalized eigenvalue while $\lambda_i = \frac{\alpha_i}{\beta_i}$ corresponding to α_i and $\beta_i \geq 10^{-3}$ would constitute a finite generalized eigenvalue. Details are discussed below.

The QZ algorithm is an obvious generalization of the QR algorithm and, just as in the QR algorithm, the QZ algorithm can be implemented in real arithmetic by orthogonal transformations with 2x2 blocks on the diagonal and first subdiagonal of QLZ in the case of complex pairs of generalized eigenvalues. Details are found in [6].

Clearly the application of the QZ algorithm is straightforward. There is also one unexpected bonus for this particular application of QZ and that is that balancing may be applied (e.g. subroutine BALANC in EISPACK) to the first n rows and columns of L because of the special form of M. This is cheap to implement and can occasionally enhance the accuracy of the computed solution and should therefore generally be used. The problem of balancing for the general generalized eigenvalue problem is still an open area of research. We would emphasize the fact that no preprocessing (rank tests, matrix multiplication, inversions, etc.) of the system matrices is required. The transmission zeros are readily determined once one decides what to numerically use as "zero" for the α_i's and β_i's. Typically one would call zero anything less than some δ related to the precision to which the system data are known (or, from purely numerical considerations, one could conservatively use a δ on the order of the square root of ε). The point is--and this is standard for orthogonal similarity algorithms--that "a quantity may be set to zero if a perturbation of the same size can be tolerated in the original matrix" [17]. There are then three cases to consider:

CASE (1): $\beta_i \geq \delta$.

$\lambda_i = \frac{\alpha_i}{\beta_i}$ is a transmission zero.

CASE (2): $\beta_i < \delta$, $|\alpha_i| \geq \delta$.

This corresponds to a generalized eigenvalue at infinity. There will be $m+s (s \geq 0)$ of these: m of them arise because the rank deficiency of M is m (and they usually appear with "hard zeros" for β_i), while the other s correspond to transmission zeros at infinity.

CASE (3): $\beta_i < \delta$, $|\alpha_i| < \delta$.

This is the degenerate case where $\det(L-\lambda M) \equiv 0$ and all complex λ are in $\mathcal{T}$. Note that in the near-degenerate case ($|\alpha_i|$ and β_i simultaneously small, i.e., near δ, but, for example, β_i somewhat greater than δ) the computed $\lambda_i = \frac{\alpha_i}{\beta_i}$ is "ill-conditioned, however reasonable it may appear" [17]. Of course, a decision has still been made concerning δ but, again, the point is that since the α_i's and β_i's are derived via orthogonal similarities, an intelligent decision--related to the original data--can be made.

The test for degeneracy in Case (3) is thus also a very reliable and stable way of determining left or right invertibility of a system.

The major advantage of the QZ approach is reliability. The QZ algorithm computes transmission zeros about as accurately as their numerical conditioning will allow. The most significant benefits derive from the determination of the α_i and β_i (by unitary similarities), the ratios of which determine the finite transmission zeros, if any. There are no controllability or observability assumptions; there are no initial rank assumptions which need to be checked; degeneracy (or, just as important, near-degeneracy) is detected "automatically" (i.e., without a separate test) in a stable way. In short, no preliminary analysis of the system matrices is needed at all. The algorithm proceeds directly on the raw system data. All the difficult (if done properly) programming and analysis has been done by the specialists.

We might also mention that while a superficial examination of the problem might indicate that the QZ approach would be slightly more CPU-time consuming, in some cases, than other theoretical approaches, in practice it is usually faster because of its reliability and direct applicability. An approximate ballpark figure for CPU time is $20(n+m)^3$ microseconds on an IBM 370/165 system (an average "medium" speed system) using the FORTRAN H Extended (Optimize = 2) compiler with double precision arithmetic.

We next turn to an examination of the eigenvectors produced by the QZ approach and see how they give important system-theoretic information.

5. Computation of Supremal (A,B) - Invariant and Controllability Subspaces

In this section we investigate the computation of two fundamental subspaces critical to the synthesis of feedback controls via the geometric theory for the system considered in Section 4. Specifically, these subspaces are $\mathcal{V}^* =$ the supremal (A,B) - invariant subspace contained in Ker D and $\mathcal{R}^* =$ the supremal (A,B) - controllability subspace contained in Ker D. This section is based on [21] which should be consulted for details.

The essence of our approach to this problem is a blend of numerical analysis and control theory with an eye toward design and applications. For design purposes, knowledge solely of $\mathcal{V}^*$, $\mathcal{R}^*$ is just not sufficient; certain subspaces of $\mathcal{V}^*$, $\mathcal{R}^*$ may be unsuitable for true design applications. Examples of the consequences of design based on these unreliable parts of $\mathcal{V}^*$, $\mathcal{R}^*$ are given in [21].

The final word on the "best" way to go about computing bases for $\mathcal{V}^*$, $\mathcal{R}^*$ is still open for investigation. What we shall outline here, instead, are prototypical algorithms which reflect attention to the kinds of numerical considerations discussed in Sections 0, 1, and 2. The power of this approach lies in the information generated to enable identification of the reliable components of $\mathcal{V}^*$, $\mathcal{R}^*$.

For the system

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$z(t) = Dx(t)$$

we will assume that x is an n-vector, u is an m-vector, and z is an r-vector. We will furthermore assume, without loss of generality, that rank (D) = r. Throughout the discussion, we shall make a number of convenient assumptions for the purpose of not clouding the basic issues with fussy technicalities. This approach does sidestep some of the nontrivial numerical considerations but should be sufficient to allow the reader adequate insight into the types of problems to be encountered.

Let $P(\lambda) = L - \lambda M$ where L and M are defined in the previous section and may now be nonsquare. We shall assume throughout the rest of this section that $m \geq r$ and $P(\lambda)$ is full rank, i.e., $P(\lambda)$ has a nonidentically-vanishing $(n+r) \times (n+r)$ minor. The set of transmission zeros is defined to be all complex numbers λ which reduce the rank of $P(\lambda)$. We shall make the simplifying assumption that there are no multiple transmission zeros. The case $m < r$ is not "well-posed" (see [1]) and will not be discussed further.

Recall from Professor Wonham's lecture that the space $\mathcal{V}^*$ is the largest subspace in Ker D (and thus "unobservable" at z) which can be made (A + BF)-invariant for some feedback map F ($u(t) = Fx(t)$ is the feedback controller). The space $\mathcal{R}^* \subseteq \mathcal{V}^*$ is the largest such subspace with the spectral assignability property: namely, the spectrum of $(A+BF)|\mathcal{R}^*$ can be freely assigned by suitable choice of $F \in \underline{F}(\mathcal{V}^*) = \{F: (A+BF)\mathcal{V}^* \subseteq \mathcal{V}^*\}$. In other words, over the class $\underline{F}(\mathcal{V}^*)$, the spectrum of $(A+BF)|\mathcal{V}^*$ can be partitioned into an assignable set Λ_A and a fixed set Λ_F. The set Λ_A will be involved in deriving a basis for $\mathcal{R}^*$ while Λ_F will be involved in a basis for $\frac{\mathcal{V}^*}{\mathcal{R}^*}$. The combined bases will provide a basis for $\mathcal{V}^*$. Further characterizations and properties of these subspaces are found in [1] and [21]. Their numerical determination will be based on the following two theorems [21].

THEOREM 3: Let $d_r = d(\mathcal{R}^*)$ (Note: $d_r \leq (n-r) = d(\text{Ker } D)$) and let $\Lambda = \{\lambda_1,\ldots,\lambda_k\} \subseteq \mathbb{C}$ be such that

(i) $\Lambda = \bar{\Lambda}$ and $\Lambda \cap \mathbb{R} \neq \phi$;

(ii) $k \geq d_r$;

(iii) no element of Λ is a transmission zero or has as its real part a transmission zero.

For all $i \in \underline{k}$ $(=\{1,\ldots,k\})$ let $\begin{pmatrix} V_i \\ W_i \end{pmatrix}$ be a matrix whose columns form a basis for Ker $P(\lambda_i)$, i.e.,

$$\begin{pmatrix} A-\lambda_i I & B \\ D & 0 \end{pmatrix} \begin{pmatrix} v_i \\ w_i \end{pmatrix} = 0 .$$

Then $\mathcal{R}^* = \text{Span}_{\mathbb{R}} \{v_i, i \in \underline{k}\}$. ■

THEOREM 4: Let $z_1, \ldots, z_q$ be the transmission zeros (distinct). For $i \in \underline{q}$, let $\begin{pmatrix} v_i \\ w_i \end{pmatrix}$ span Ker $P(z_i)$, i.e.,

$$\begin{pmatrix} A-z_i I & B \\ D & 0 \end{pmatrix} \begin{pmatrix} v_i \\ w_i \end{pmatrix} = 0 .$$

Then $\mathcal{V}^* = \mathcal{R}^* + \text{Span}_{\mathbb{R}} \{v_i, i \in \underline{q}\}$. ■

We now outline prototype algorithms for $\mathcal{V}^*$, $\mathcal{R}^*$ based on these two theorems. The algorithms are referred to as prototype for two reasons. First, we assume that the dimension of $\mathcal{V}^*$ is n-r and that $\mathcal{V}^* = \mathcal{R}^*$ whenever $m > r$. It is clear that this is a valid assumption if A,B,D arise from measured data (for then we are almost surely in the generic situation). The algorithms may be modified easily to remove this assumption. Second, the reader will find that there are straight-forward modifications which reduce the number of necessary operations.

The important point is that the algorithms provide information which is essential for practical engineering design.

<u>To compute basis vectors for $\mathcal{R}^*$</u> (m>r):

<u>Step 1</u>: Determine the transmission zeros of the system (see [16]). Generically, of course, there won't be any. We assume this here and refer to [21] otherwise.

<u>Step 2</u>: Select $\{\lambda_1, \ldots, \lambda_{n-r}\}$ to satisfy the conditions of Theorem 3. The elements of the set should represent acceptable closed loop eigenvalues for $(A+BF)|\mathcal{R}^*$.

Step 3: For i ε $\underline{n-r}$, compute V_i, W_i as in Theorem 3 by SVD.

Step 4: Compute the SVD of $V_{\mathcal{R}^*} = [V_1,\ldots,V_{n-r}]$. The left singular vectors corresponding to the singular values of $V_{\mathcal{R}^*}$ which are greater than some threshold (possibly related to the data uncertainty) then form a basis for the "reliable" part of $\mathcal{R}^*$. The left singular vectors corresponding to the "small" singular values correspond to the "unreliable" part of $\mathcal{R}^*$.

To compute basis vectors for $\mathcal{V}^*$(m=r):

Step 1: Determine the transmission zeros z_i and associated generalized eigenvectors $\begin{pmatrix} v_i \\ w_i \end{pmatrix}$ satisfying

$$\begin{pmatrix} A-z_iI & B \\ D & 0 \end{pmatrix}\begin{pmatrix} v_i \\ w_i \end{pmatrix} = 0$$

by the QZ algorithm [16].

Step 2: Compute the SVD of $V_{\mathcal{V}^*} = [v_1,\ldots,v_{n-r}]$. The left singular vectors corresponding to the singular values of $V_{\mathcal{V}^*}$ which are greater than some threshold form a basis for the "reliable" part of $\mathcal{V}^*$. Transmission zeros of too large magnitude or those which have nonnegative real part are frequently excluded from consideration, also, so that only a subset of finite, stable transmission zeros are used to determine a working subspace $\mathcal{V}^*_{fs}$.

Finally, a word on the "generic" situations (see [1]). In case m>r we have:

$$\mathcal{R}^* = \text{Ker } D \qquad \text{(g)} \qquad \text{("(g)" denotes "generically")}$$

$$\mathcal{V}^* = \mathcal{R}^* \qquad \text{(g)}$$

$$\mathcal{T} = \phi \qquad \text{(g)}$$

In case m=r we have:

$$\mathcal{R}^* = 0 \qquad (g)$$

$$\mathcal{V}^* = \text{Ker } D \qquad (g)$$

$$\mathcal{T} = \{z_1, \dots, z_{n-m}\} \qquad (g)$$

Ker D may be seen to be generically (A,B) -invariant if $m \geq r$ since $d(\text{Ker } D) = n-r$, $d(\text{Im } B) = m$ implies $\text{Ker } D + \text{Im } B = \mathcal{X}$ (g) so that, trivially,

$$A \text{ Ker } D \subseteq \text{Ker } D + \text{Im } B \qquad (g)$$

(recall $\mathcal{V}$ is (A,B)-invariant if $A\mathcal{V} \subseteq \mathcal{V} + \text{Im } B$). There are obviously difficult numerical issues to be addressed in the nongeneric (but nonetheless important) cases since arbitrary small perturbations cause the problem to slide into (albeit just barely) the generic situation. However, in many situations there are certain "hard zeros" in the problem so that arbitrary dense perturbations may not be relevant. This remains an open area for investigation.

6. Mathematical Software[1]

The last, but probably the most important topic of all, is the implementation of control theory and numerical algorithms in mathematical software. The software should be reliable, portable, and unaffected by the machine or system environment in which it is used. In saying that the software is reliable we mean that computation will continue as long as meaningful results can be obtained. If meaningful computation cannot continue sufficient information will be given to the user to enable him to diagnose the trouble with his problem.

At the present time mathematical software serves as a major vehicle of communication among numerical analysts, algorithmists (frequently the numerical analyst and the algorithmist must be one and the same), users, and potential users of the software. In the not too distant past software was looked upon primarily as development work rather than research. However, recent rigorous work on mathematical software has served to stimulate research in numerical analysis. Examples are the condition estimate of a linear system of equations [22], the convergence analysis for iteratively reweighted least squares which is a part of robust estimation [23], nonlinear least squares [24], and unconstrained optimization [25].

The prototypical work on reliable, portable mathematical software for the standard eigenproblem was started in 1968. EISPACK [5],[6] Editions I and II were an outgrowth of that work. Since that time many pre-processors

[1]It is a pleasure to acknowledge the collaboration of my colleague Virginia C. Klema in the preparation of most of this section.

that are, themselves, portable software have been designed and implemented to assist in instituting and verifying portability. An excellent reference on portability of mathematical software is [26]. Among such machine aids that are in use are the PFORT verifier [27] from Bell Labs. and the FORTRAN Converter from International Mathematical and Statistical Libraries, Inc. [28]. Reliable software that has been constructed using these machine aids includes ROSEPACK [29] and NL2SOL [24]. ROSEPACK is mathematical software that includes an interactive driver to do iteratively reweighted least squares. NL2SOL is a system of subroutines to do nonlinear least squares computations.

Inevitably numerical algorithms are strengthened when their mathematical software is made portable. Furthermore such software has been shown to be markedly faster by factors ranging from 10 to 50 than earlier and less reliable code.

The documentation for portable mathematical software, and the above-mentioned machine aids that assist in the design of the software are sufficient for users who have not been associated with the software to use, extend, or modify it. The portable software must be modular in design, well-structured, and the comments within the program or subroutine must be sufficient to inform the user about input parameters, output parameters, temporary storage requirements, error exits, and the algorithm that the program implements.

Experience has shown that applications subsystems themselves that include input-output and on-line documentation in the form of "help" commands can be constructed as portable software that operates efficiently in an interactive environment [29]. In short, a research environment can be constructed--in fact, has been constructed, that serves the research worker and the applications user.

It is clear that many aspects of control and estimation theory are ready for the research and design that is necessary to produce reliable, portable mathematical software that performs in bounded arithmetic, that is, the finite precision arithmetic of computing machines.

7. Concluding Remarks

We have outlined above a number of numerical problems associated with one branch of control theory. There are countless more such control and estimation problems and a great deal of numerical groundwork remains to be completed before significant progress can be made on the numerical aspects of such problems. Indeed, some synthesis and design problems may not yet even be well-posed in the context of finite precision computation. However, the ultimate goal is to solve real problems and reliable tools (mathematical software) and experience must be available to effect real solutions or strategies. This is not generally the case at the present time, partly because there is frequently no way of determining, in complex problems, whether current techniques and software are reliable. Serious interdisciplinary research combining control and estimation theory, numerical analysis, and mathematical software can make a valuable contribution to improving the present state of affairs.

8. References

[1] Wonham, W. M., Linear Multivariable Control: A Geometric Approach, Springer-Verlag, New York, 1974.

[2] Wonham, W. M., Geometric Methods in the Structural Synthesis of Linear Multivariable Controls, Proc. 1977 Joint Automatic Control Conf., San Francisco, CA, June 1977, pp. 594-600.

[3] Dahlquist, G. and Å. Björck, Numerical Methods, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[4] Wilkinson, J. H., The Algebraic Eigenvalue Problem, Oxford University Press, London, 1965.

[5] Smith, B. T., et al., Matrix Eigensystem Routines--EISPACK Guide, Second Edition, Lect. Notes in Comp. Sci., Vol. 6, Springer-Verlag, New York, 1976.

[6] Garbow, B. S., et al., Matrix Eigensystem Routines--EISPACK Guide Extension, Lect. Notes in Comp. Sci., Vol. 51, Springer-Verlag, New York, 1977.

[7] Golub, G. H. and J. H. Wilkinson, Ill-Conditioned Eigensystems and the Computation of the Jordan Canonical Form, SIAM Review, 18(1976), 578-619.

[8] Strang, G., Linear Algebra and Its Applications, Academic Press, New York, 1976.

[9] Stewart, G. W., Introduction to Matrix Computations, Academic Press, New York, 1973.

[10] Forsythe, G. E., M. A. Malcolm, and C. B. Moler, Computer Methods for Mathematical Computations, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[11] Golub, G. H., and C. Reinsch, Singular Value Decomposition and Least Squares Solutions, Numer. Math., 14(1970), 403-420.

[12] Dongarra, J. J., et al., LINPACK Working Note #9, Preliminary LINPACK User's Guide, Argonne National Lab., Appl. Math. Div., TM-313, Aug. 1977.

[13] Golub, G. H., V. C. Klema, and G. W. Stewart, Rank Degeneracy and Least Squares Problems, Tech. Rep't. STAN-CS-76-559, Computer Science Dept., Stanford University, Aug. 1976.

[14] Stewart, G. W., On the Perturbation of Pseudo-Inverses, Projections, and Linear Least Squares Problems, SIAM Rev., 19(1977), 634-662.

[15] Forsythe, G. E., and C. B. Moler, Computer Solution of Linear Algebraic Systems, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[16] Laub, A. J. and B. C. Moore, Calculation of Transmission Zeros Using QZ Techniques, Automatica, (to appear, Nov. 1978).

[17] Moler, C. B., and G. W. Stewart, An Algorithm for Generalized Matrix Eigenvalue Problems, SIAM J. Numer. Anal., 10(1973), 241-256.

[18] Francis, B. A., and W. M. Wonham, The Role of Transmission Zeros in Linear Multivariable Regulators, Int. J. Control, 22(1975), 657-681.

[19] Stewart, G. W., On the Sensitivity of the Eigenvalue Problem $Ax = \lambda Bx$, SIAM J. Numer. Anal., 9(1972), 669-686.

[20] Stewart, G. W., Gershgorin Theory for the Generalized Eigenvalue Problem $Ax = \lambda Bx$, Math. Comp., 29(1975), 600-606.

[21] Moore, B. C., and A. J. Laub, Computation of Supremal (A,B) - Invariant and Controllability Subspaces, IEEE Trans. Aut. Contr. (to appear).

[22] Cline, A. K., C. B. Moler, G. W. Stewart, and J. H. Wilkinson, An Estimate for the Condition Number of a Matrix, LINPACK Working Note #7, Argonne Na. Lab., TM-310, July 1977.

[23] Dennis, John E., Jr., Private communication, (1978).

[24] Dennis, John E., Jr., D. M. Gay, and R. E. Welsch, An Adaptive Nonlinear Least Squares Algorithm, NBER Working Paper 196, (1977), submitted to ACM Transactions on Mathematical Software.

[25] Dennis, John E., Jr., Nonlinear Least Squares and Equations, in The State of the Art of Numerical Analysis, edited by D. Jacobs, Academic Press, London, 1977.

[26] Goos, G. and J. Hartmanis, Portability of Numerical Software, Oak Brook, Ill., Lecture Notes in Computer Science, Vol. 57, Springer-Verlag, New York, 1977.

[27] Ryder, B. G., The PFORT Verifier: User's Guide, CS Tech. Report 12, Bell Labs., 1975.

[28] Aird, T. J., The FORTRAN Converter User's Guide, IMSL, 1975.

[29] Coleman, David, Paul Holland, Neil Kaden, Virginia Klema, and Stephen C. Peters, A System of Subroutines for Iteratively Reweighted Least Squares Computations, NBER Working Paper 189, 1977, submitted to Transactions on Mathematical Software.

[30] Wilkinson, J.H., Rounding Errors in Algebraic Processes, Prentice-Hall, Englewood Cliffs, N.J., 1963.

[31] Givens, W., Numerical Computation of the Characteristic Values of a Real Symmetric Matrix, Oak Ridge National Lab., ORNL-1574, March, 1954.

[32] Golub, G.H., and W. Kahan, Calculating the Singular Values and Pseudo-Inverse of a Matrix, SIAM J. Numer. Anal., 2 (1965), 205-224.