more expensive to implement (by a factor of at least ten) than the direct inverse approach.

The above remarks should be viewed only as suggested ways to approach the problem of computing robustness bounds rather than definitive algorithms. We have not even discussed the critical issues of scaling, equilibration, etc. These topics are currently under investigation.

References

[1] W. M. Wonham, *Linear Multivariable Control: A Geometric Approach*. Berlin: Springer-Verlag, 1974.
[2] G. W. Stewart, *Introduction to Matrix Computations*. New York: Academic, 1973.
[3] N. R. Sandell, "Robust stability of linear dynamic systems with application to singular perturbation theory," submitted to *Automatica*; also available from Electron. Syst. Lab., Massachusetts Inst. Technol., Cambridge, MA, Rep. ESL-P-837, Aug. 1978.
[4] A. J. Laub, "Linear multivariable control: Numerical considerations" (Invited Paper), presented at the Amer. Math. Soc. Short Course on Control and Systems Theory, Providence, RI, Aug. 1978; also available from Electron. Syst. Lab., Massachusetts Inst. Technol., Rep. ESL-P-833, July 1978.
[5] M. Safonov and M. Athans, "Gain and phase margins for multiloop LQG regulators," *IEEE Trans. Automat. Contr.*, vol. AC-22, pp. 173–179, 1977.
[6] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson, "An estimate for the condition number of a matrix," Argonne Nat. Lab., Appl. Math. Div., LINPACK Working Note 7, TM-310, July 1977.
[7] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, "Preliminary LINPACK user's guide," Argonne Nat. Lab., Appl. Math. Div., LINPACK Working Note 9, TM-313, Aug. 1977.

A Note on the Evaluation of Matrix Polynomials

CHARLES VAN LOAN

***Abstract*—The problem of evaluating a polynomial $p(x)$ in a matrix A arises in many applications, e.g., the Taylor approximation of e^A. The $0(\sqrt{q}\, n^3)$ algorithm of Paterson and Stockmeyer has the drawback that it requires $0(\sqrt{q}\, n^2)$ storage, where q is the degree of p and n is the dimension of A. An algorithm which greatly reduces this storage requirement without undue loss of speed is presented.**

One method for evaluating the matrix polynomial

$$p(A)=b_0I+b_1A+\cdots+b_qA^q \qquad b_i\in R,\ A\in R^{n\times n}$$

is to use Horner's rule:

$$S_0=b_qI.$$

For $k=1,2,\cdots,q$

$$\lfloor S_k=AS_{k-1}+b_{q-k}I$$

whereupon $S_q=p(A)$. Ignoring low order terms in n, this algorithm requires $2n^2$ storage and $(q-1)n^3$ multiplications.

In an interesting paper by Paterson and Stockmeyer [2], it is shown how this computation can be rearranged so that only $0[\sqrt{q}\, n^3]$ multiplications are needed. The main idea behind their algorithm is to apply Horner's rule to $p(A)$ regarded as a polynomial in A^s where $s\approx\sqrt{q}$. For example, if $q=9$, then

$$p(A)=(b_9I)(A^3)^3+(b_8A^2+b_7A+b_6I)(A^3)^2+(b_5A^2+b_4A+b_3I)(A^3)+(b_2A^2+b_1A+b_0I).$$

Once A^2 and A^3 are computed, then only $2n^3$ multiplications are required to evaluate $p(A)$ as evidenced by the following "Horner regrouping":

$$p(A)=A^3[A^3[b_9A^3+(b_8A^2+b_7A+b_6I)]+(b_5A^2+b_4A+b_3I)]+(b_2A^2+b_1A+b_0I).$$

Manuscript received September 27, 1978.
The author is with the Department of Computer Science, Cornell University, Ithaca, NY 14853.

Hence, by this technique, only $4n^3$ multiplications are required to evaluate $p(A)$. This is in contrast to the count of $8n^3$ associated with the "ordinary" application of Horner's rule.

In general, if s is any integer satisfying

$$1\leqslant s\leqslant q$$

and

$$r=[q/s]=\text{the integral part of } q/s$$

then

$$p(A)=\sum_{k=0}^{r}B_k(A^s)^k \tag{1}$$

where

$$B_k=b_{sk+s-1}A^{s-1}+\cdots+b_{sk+1}A+b_{sk}I \qquad (k=0,\cdots,r-1)$$

and

$$B_r=b_qA^{q-sr}+\cdots+b_{sr+1}A+b_{sr}I.$$

Applying Horner's rule to (1) we obtain

$$F_0=B_r.$$

For $k=1,\cdots,r$

$$\lfloor F_k=(A^s)F_{k-1}+B_{r-k}$$

where $F_r=p(A)$. The fact that there can be a significant reduction in work as a result of these manipulations is made clear when this iteration is spelled out in greater detail.

Algorithm I

1) $Y_k=A^k \qquad (k=0,1,\cdots,s)$.
2) $F_0=b_qY_{q-sr}+b_{q-1}Y_{q-sr-1}+\cdots+b_{sr+1}Y_1+b_{sr}Y_0$.
3) For $k=1,\cdots,r$

$$\lfloor F_k=Y_sF_{k-1}+b_{s(r-k)+s-1}Y_{s-1}+\cdots+b_{s(r-k)}Y_0.$$

This is essentially Algorithm B in [2] applied to the evaluation of matrix polynomials. Noting that $(s-1)n^3$ multiplications are required to compute the Y_k and that there is one matrix–matrix multiplication per iteration in step 3), we conclude that for a given s, Algorithm I requires $w_1(s)n^3$ multiplications where

$$w_1(s)=\begin{cases}s+[q/s]-2 & \text{if } s \text{ divides } q\\ s+[q/s]-1 & \text{otherwise.}\end{cases}$$

Setting $s=1$ in Algorithm I renders the ordinary version of Horner's rule. Setting $s=[\sqrt{q}\,]$ approximately minimizes $w_1(s)$, and we then find $w_1([\sqrt{q}\,])\approx 2\sqrt{q}$.

A discouraging feature of Algorithm I is that it requires $(s+1)n^3$ storage. (Arrays are required for $A,A^2,\cdots,A^s$, and the "current" F_k.) However, a slightly less efficient algorithm which requires only $3n^2$ storage results if $p(A)$ is computed one column at a time. To derive this algorithm, let e_j denote the jth column of the identity matrix. From (1) we have

$$p(A)e_j=\sum_{k=0}^{r}(A^s)^k(B_ke_r)$$

which may be computed as follows:

$$f_0^{(j)}=B_re_j.$$

For $k=1,\cdots,r$

$$\lfloor f_k^{(j)}=(A^s)f_{k-1}^{(j)}+B_{-k}e_j$$

where $f_r^{(j)}=p(A)e_j$, the jth column of $p(A)$. Notice that once the vectors $Ae_j,A^2e_j,\cdots,A^{s-1}e_j$ are computed, then the vectors B_ke_j, $k=0,\cdots,r$ are "free":

0018-9286/79/0400-0320$00.75 ©1979 IEEE

$$B_k e_j = b_{sk+s-1}\left(A^{s-1}e_j\right) + \cdots + b_{sk+1}(Ae_j) + b_{sk}e_j \qquad (k=0,\cdots,r-1)$$
$$B_r e_j = b_q A^{q-sr}e_j + \cdots + b_{sr+1}Ae_j + b_{sr}e_j.$$

We therefore obtain the following algorithm.

ALGORITHM II

1) $Y = A^s$.
2) For $j = 1,\cdots,n$

$$\left\lfloor \begin{array}{l} y_0^{(j)} = e_j. \\ \text{For } k=1,\cdots,s-1 \\ \left\lfloor y_k^{(j)} = Ay_{k-1}^{(j)} \right. \\ f_0^{(j)} = b_q y_{q-sr}^{(j)} + \cdots + b_{sr+1}y_1^{(j)} + b_{sr}y_0^{(j)}. \\ \text{For } k=1,\cdots,r \\ \left\lfloor f_k^{(j)} = Yp_{k-1}^{(j)} + b_{s(r-k)+s-1}y_{s-1}^{(j)} + \cdots + b_{s(r-k)}y_0^{(j)} \right. \end{array} \right.$$

whereupon $p(A) = [f_r^{(1)} | f_r^{(2)} | \cdots | f_r^{(n)}]$.

Since arrays are needed for only A, A^s, and $p(A)$, Algorithm II requires $3n^2$ storage. A careful operation count for $s>1$ shows that $w_2(s)n^3$ multiplications are required where

$$w_2(s) = \begin{array}{ll} 2(s-1)+[q/s]-2 & \text{if } s \text{ divides } q \\ 2(s-1)+[q/s]-1 & \text{otherwise.} \end{array}$$

This work count is approximately minimized by setting $s=[\sqrt{q/2}\,]$ whereupon $w_2(s)\approx 2\sqrt{2q}$ —about 1.5 times the work associated with Algorithm I.

At the expense of one extra $n\times n$ array, A^s can be computed via its binary expansion

$$A^s = (A^{\beta_0 2^0})(A^{\beta_1 2^1})\cdots(A^{\beta_m 2^m}) \qquad s = \sum_{k=0}^{m}\beta_k 2^k$$

in $\alpha\log_2(s)n^3$ multiplications where $\alpha\in[1], [2]$ depends upon the number of nonzero β_i's. (If s is an exact power of two, the extra storage is not required.) When this feature is incorporated in Algorithm II, the work count becomes $\hat{w}_2(s)n^3$ where

$$\hat{w}_2(s) = \alpha\log_2(s) - 1 + w_1(s).$$

These ideas have been incorporated in a subroutine for approximating the matrix exponential e^A with a truncated Taylor series

$$e^A \approx \sum_{k=0}^{q} A^k/k! = T_q(A).$$

For reasons based on the machine precision used (10^{-15}), the norm of A, and the desired accuracy, q was chosen to be 16. (See [1] for an analysis of Taylor approximation to e^A.) Setting $s=4$ and using binary squaring to compute A^4, Algorithm II requires $7n^3$ multiplications and $3n^2$ storage to evaluate $T_{16}(A)$. This is less than half the work associated with traditional Horner evaluation and only marginally slower than the $6n^3$ multiplications needed by Algorithm I which requires $5n^2$ storage for this problem.

ACKNOWLEDGMENT

The author greatly profited from discussions he had with Dr. L. Berman who pointed out [2].

REFERENCES

[1] C. B. Moler and C. Van Loan, "Nineteen dubious ways to compute the matrix exponential," *SIAM Rev.*, vol. 20, pp. 801–836, 1978.
[2] M. S. Paterson and L. Stockmeyer, "On the number of nonscalar multiplications necessary to evaluate polynomials," *SIAM J. Comput.*, vol. 2, pp. 60–66, 1973.

Subsystem Simplification in Large-Scale Systems Analysis

R. A. EL-ATTAR AND M. VIDYASAGAR

***Abstract*—A new procedure for stabilizing an unstable system by stable feedback is presented. The procedure is based on using a lower order model for the given system. Stabilization conditions for the original system are given in terms of the error in the approximation. The procedure is extended to large-scale systems whereby stabilization conditions with decentralized feedback are presented.**

PROBLEM FORMULATION AND SOLUTION

Let $A^{n\times n}$ be the set of $n\times n$ matrices whose elements have entries of the form [1]

$$f(t) = \begin{cases} f_a(t) + \sum_{i=0}^{\infty} f_i\delta(t-t_i), & \text{for } t \geqslant 0 \\ 0, & \text{for } t<0 \end{cases}$$

where

$$\int_0^{\infty} |f_a(t)|\,dt < \infty$$

and

$$\sum_{i=0}^{\infty} |f_i| < \infty.$$

Suppose $G\notin A^{n\times n}$; the objective is to find $H\in A^{n\times n}$ such that $\hat{G}(I+\hat{H}\hat{G})^{-1}\in\hat{A}^{n\times n}$ where $\hat{\cdot}$ denotes the Laplace transform. In other words, the objective is to stabilize the unstable system whose impulse response matrix is $G(t)$ by means of a stable feedback H. It is now known [2] that if $\hat{G}(s)$ is a proper rational matrix in s, then under some conditions it is possible to find a suitable H. The technique in [2] requires constructing a state-variable realization of the transfer function $\hat{G}(s)$; hence, the larger the order of such a realization, the more complicated it is to find a suitable H.

As an alternative, we propose the following approach: First, replace $\hat{G}$ by a "simpler" transfer function matrix $\hat{G}_0$, and design $\hat{H}\in\hat{A}^{n\times n}$ such that $(I+\hat{G}_0\hat{H})^{-1}\hat{G}_0\in\hat{A}^{n\times n}$; then implement the same feedback with the original $\hat{G}$. The question then becomes, under what condition does H stabilize the original G? A simple answer is contained in the following lemma.

Lemma 1: Suppose $H\in A^{n\times n}$ and $(I+\hat{G}_0\hat{H})^{-1}\hat{G}_0\in\hat{A}^{n\times n}$. Under these conditions $(I+\hat{G}_0\hat{H})^{-1}\in\hat{A}^{n\times n}$. Moreover, whenever $G-G_0\in A^{n\times n}$ and

$$\|(I+\hat{G}_0\hat{H})^{-1}\|\cdot\|\hat{G}-\hat{G}_0\|\cdot\|\hat{H}\| < 1 \qquad (1)$$

we have that $\hat{G}(I+\hat{H}\hat{G})^{-1}\in\hat{A}^{n\times n}$, $(I+\hat{H}\hat{G})^{-1}\in\hat{A}^{n\times n}$.

The proof is superficially similar to that of the small gain theorem, but some care is required because $G\notin A^{n\times n}$ and $G_0\notin A^{n\times n}$.

Lemma 1 can be interpreted as follows. Suppose we replace G by the simpler system G_0 in such a way that $G-G_0\in A^{n\times n}$ (in other words, G_0 faithfully reproduces the "unstable part" of G), and suppose we find $H\in A^{n\times n}$ that stabilizes G_0. Then H also stabilizes G provided (1) holds. Lemma 1 can also be interpreted, not in terms of system *simplification*, but also in terms of system *uncertainty*. Suppose the system G is not precisely known, and is nominally represented by G_0. If H stabilizes G_0, then H also stabilizes all G such that

$$\|\hat{G}-\hat{G}_0\| < \left(\|(I+\hat{H}\hat{G}_0)^{-1}\|\cdot\|\hat{H}\|\right)^{-1}. \qquad (2)$$

Manuscript received December 10, 1977.
The authors are with the Department of Electrical Engineering, Concordia University, Montreal, P.Q., Canada.

0018-9286/79/0400-0321$00.75