nant for the cofactor C_{ij}, $i \neq j$, is

$$C_{ij} = \begin{array}{c} \\ 1 \\ 2 \\ \cdot \\ \cdot \\ \cdot \\ i \\ \cdot \\ \cdot \\ \cdot \\ n \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & \cdots & j & \cdots & i & \cdots & n \end{array} \\ \begin{bmatrix} s-a_{11} & -a_{12} & \cdots & -a_{1j} & \cdots & -a_{1i} & \cdots & -a_{1n} \\ -a_{21} & s-a_{22} & \cdots & -a_{2j} & \cdots & -a_{2i} & \cdots & -a_{2n} \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ 0 & 0 & \cdots & 1 & & 0 & \cdots & 0 \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ -a_{n1} & -a_{n2} & & -a_{nj} & \cdots & -a_{ni} & \cdots & s-a_{nn} \end{bmatrix} \end{array}. \tag{3}$$

Note that s must appear in all diagonals if the techniques for transformation to phase variable canonical form are to be employed. Equation (3) can be recast as

$$C_{ij} = \begin{array}{c} \\ 1 \\ 2 \\ \cdot \\ \cdot \\ \cdot \\ i \\ \cdot \\ \cdot \\ \cdot \\ n \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & \cdots & j & \cdots & i & \cdots & n \end{array} \\ \begin{bmatrix} s-a_{11} & -a_{12} & \cdots & -a_{1j} & \cdots & -a_{1i} & \cdots & -a_{1n} \\ -a_{21} & s-a_{22} & \cdots & -a_{2j} & \cdots & -a_{2i} & \cdots & -a_{2n} \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ 0 & 0 & & 1 & & s-s & \cdots & 0 \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ -a_{n1} & -a_{n2} & \cdots & -a_{nj} & \cdots & -a_{ni} & \cdots & s-a_{nn} \end{bmatrix} \end{array}. \tag{4}$$

A partial Laplacian expansion of (4) provides the following expanded form:

$$C_{ij} = \begin{bmatrix} s-a_{11} & -a_{12} & \cdots & -a_{1j} & \cdots & -a_{1i} & \cdots & -a_{1n} \\ -a_{21} & s-a_{22} & \cdots & -a_{2j} & \cdots & -a_{2i} & \cdots & -a_{2n} \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ 0 & 0 & \cdots & 1 & \cdots & s & \cdots & 0 \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ -a_{n1} & -a_{n2} & \cdots & -a_{nj} & \cdots & -a_{ni} & \cdots & s-a_{nn} \end{bmatrix}$$

$$- \begin{bmatrix} s-a_{11} & -a_{12} & \cdots & -a_{1j} & \cdots & -a_{1i} & \cdots & -a_{1n} \\ -a_{21} & s-a_{22} & \cdots & -a_{2j} & \cdots & -a_{2i} & \cdots & -a_{2n} \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ 0 & 0 & \cdots & 0 & \cdots & s & \cdots & 0 \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & & \cdot \\ -a_{n1} & -a_{n2} & \cdots & -a_{nj} & \cdots & -s_{ni} & \cdots & s-a_{nn} \end{bmatrix}. \tag{5}$$

Both determinants in (5) are now calculable by any of the techniques for transforming to phase variable canonical form. For a diagonal cofactor, the problem is easier, i.e.,

$$C_{ii} = \frac{1}{s} \begin{bmatrix} s-a_{11} & -a_{12} & \cdots & -a_{1i} & \cdots & -a_{1n} \\ -a_{21} & s-a_{22} & \cdots & -a_{2i} & \cdots & -a_{2n} \\ \cdot & \cdot & & \cdot & & \\ \cdot & \cdot & & \cdot & & \\ \cdot & \cdot & & \cdot & & \\ 0 & 0 & \cdots & s & \cdots & 0 \\ \cdot & \cdot & & \cdot & & \\ \cdot & \cdot & & \cdot & & \\ \cdot & \cdot & & \cdot & & \\ -a_{n1} & -a_{n2} & \cdots & -a_{ni} & \cdots & s-a_{nn} \end{bmatrix}. \tag{6}$$

Of course, the scheme will work with columns as well as rows [12], [13].

Conclusion

A major goal in this work was to develop an accurate numerical technique for cofactor computation, applicable to high-order systems. By casting the cofactor into an equivalent representation, the task is accomplished by any phase variable canonical transformation method. The authors used Danilevskii's method routinely in this capacity for systems up to 26th order while employing a Univac 1106. It appears that the technique is applicable to much higher order systems.

Although the thrust of the paper has been cofactor computation, some of the spin-off applications are calculation of $[sI-A]^{-1}$ and $\partial\lambda_k/\partial a_{ij}$.

References

[1] J. B. Nail, J. R. Mitchell, and W. L. McDaniel, Jr., "Determination of pole sensitivities by Danilevskii's method," *AIAA J.*, vol. 15, Oct. 1977.

[2] D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra*. San Francisco, CA: Freeman, 1963.

[3] Y. P. Kakad and J. Mahig, "On using Faddeeva's algorithm for a class of semidefinite systems," in *Proc. 9th Annu. Southeastern Symp. on System Theory*, Univ. of North Carolina, Charlotte, Mar. 7–8, 1977.

[4] B. S. Morgan, Jr., "Sensitivity analysis and synthesis of multivariable systems," *IEEE Trans. Automat. Contr.*, vol. AC-11, July 1966.

[5] C. D. Johnson and W. M. Wonham, "A note on transformation to canonical (phase variable) form," *IEEE Trans. Automat. Contr.*, vol. AC-9, July 1964.

[6] I. H. Mufti, "On the reduction of a system to canonical (phase variable) form," *IEEE Trans. Automat. Contr.*, vol. AC-10, Apr. 1965.

[7] M. R. Chidambara, "The transformation to (phase-variable) canonical form," *IEEE Trans. Automat. Contr.*, vol. AC-10, Oct. 1965.

[8] W. G. Tuel, Jr., "On transformation to (phase variable) canonical form," *IEEE Trans. Automat. Contr.*, vol. AC-11, July 1966.

[9] D. S. Rane, "A simplied transformation to canonical form," *IEEE Trans. Automat. Contr.*, vol. AC-11, July 1966.

[10] C. D. Johnson and W. M. Wonham, "Another note on transformation to canonical (phase-variable) form," *IEEE Trans. Automat. Contr.*, vol. AC-11, July 1966.

[11] B. Ramaswami and K. Ramar, "Transformation to phase-variable canonical form," *IEEE Trans. Automat. Contr.*, vol. AC-13, Dec. 1968.

[12] J. B. Nail, J. R. Mitchell, and W. L. McDaniel, Jr., "An application of Danilevskii's method to the determination of pole sensitivities," in *Conf. Rec. 10th Asilomar Conf. on Circuits, Systems, and Computers*, Naval Postgraduate School, Monterey, CA, 1976.

[13] J. R. Mitchell, W. L. McDaniel, Jr., and J. B. Nail, "Control system design by pole encouragement," NASA, MSFC, Final Rep., Contract NAS8-31568, Sept. 3, 1976.

Exact Reduction of a Polynomial Matrix to the Smith Normal Form

VIJAYA RAMACHANDRAN

***Abstract*—A finite field transform method is described for the exact reduction of a single-variable polynomial matrix to its Smith form, without the accompanying problem of coefficient growth.**

I. Introduction

The reduction of single-variable polynomial matrices to the Smith normal form is very useful in many areas of system theory. In particular, this reduction is used in one method of spectral factorization of rational

Manuscript received February 23, 1979.

The author is with the School of Automation, Indian Institute of Science, Bangalore 560012, India.

0018-9286/79/0800-0638$00.75 ©1979 IEEE

polynomial matrices (Youla [1]). Gantmacher [2] gives a straightforward method for this reduction and many algorithms have appeared recently which are variations of Gantmacher's method [3], [4]. However, all of these methods suffer from numerical instability and coefficient growth. Numerical inaccuracies, especially, are to be avoided in this case, since the Smith normal form of a matrix may become quite different if, say, a double root is taken as two distinct roots close to each other.

In this paper, we give an exact method for Smith form reduction based on finite field transforms [12], [14], which overcomes this difficulty without the accompanying problem of coefficient growth. This appears to be the first successful application of exact computation techniques to Smith form reduction. A Fortran program (for IBM 360/44 PS) has been developed for this purpose and the results are included.

II. Definitions and Motivation

A polynomial matrix (or λ-matrix) is a rectangular matrix $A(\lambda)$ whose elements are polynomials in λ with coefficients from the field of rational or real numbers. A λ-matrix is *unimodular* if its determinant is a nonzero constant. Two λ-matrices $A(\lambda)$ and $B(\lambda)$ are said to be *equivalent* if they are of the same dimension and if there exist unimodular matrices $M(\lambda)$ and $N(\lambda)$ such that

$$M(\lambda)A(\lambda)N(\lambda)=B(\lambda).$$

Smith Form: Any arbitrary rectangular polynomial matrix $A(\lambda)$ is equivalent to a canonical diagonal matrix (Smith form) given by

$$S(\lambda)=\begin{bmatrix} a_1(\lambda) & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & a_2(\lambda) & \cdots & & & & \vdots \\ \vdots & 0 & \cdots & a_s(\lambda) & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}$$

where $a_1(\lambda)|a_2(\lambda)|\cdots|a_s(\lambda)$, and each $a_i(\lambda)$ is monic, $i=1,2,\cdots,s$ (a polynomial is *monic*, if the coefficient of the highest degree term is 1). The polynomials $a_1(\lambda),a_2(\lambda),\cdots,a_s(\lambda)$ are uniquely determined by $A(\lambda)$.

A constructive proof of the uniqueness of the Smith form is given in Gantmacher [2]. (The treatment there is for integer matrices, but the extension to the polynomial case is straightforward.) This construction gives a basic algorithm for Smith form reduction and many other algorithms based on this have been proposed with the view to improve efficiency (Bradley [3], Pace and Barnett [4]). The chief drawback of Gantmacher's algorithm (and the succeeding ones) is the problem of coefficient growth. Since the Smith normal form of a matrix depends on the roots of its polynomial elements, it need not be a continuous function of its elements. Hence, if floating-point arithmetic is used in the computation, the cumulative errors introduced may result in a completely different Smith form.

For example, consider two matrices given below:

$$A(\lambda)=\begin{bmatrix} (\lambda-1) & 0 \\ 0 & (\lambda-1) \end{bmatrix},\quad B(\lambda)=\begin{bmatrix} (\lambda-1) & 0 \\ 0 & (\lambda-1+\epsilon) \end{bmatrix},\quad \epsilon\to 0.$$

The Smith form of $A(\lambda)$ is $A(\lambda)$ itself, while the Smith form of $B(\lambda)$ is

$$S_B(\lambda)=\begin{bmatrix} 1 & 0 \\ 0 & (\lambda-1)(\lambda-1+\epsilon) \end{bmatrix}.$$

Hence, it is clear that the use of floating-point arithmetic could easily result in a wrong Smith form.

To obviate this difficulty, the coefficients of the polynomial entries are to be maintained in rational form. This can be done using an *exact computation procedure*, which has been suggested for the inversion of integer and polynomial matrices (Rao *et al.* [6], Krishnamurthy [5], Bareiss [7], Munro and Zakian [8], Downs [9]). In this case, the numerators and denominators of the coefficients grow very large during the computation, especially in the absence of an explicit cancellation procedure. This is the well-known problem of coefficient growth, also known as "intermediate expression swell" (McClellan [10]). The problem of handling such large rational numbers (not reduced to the lowest form) is a very real one, and finite field transforms have been suggested as a possible method to overcome this difficulty. Chief among these techniques are the residue system (Rao *et al.* [6], McClellan [10], Young and Gregory [11]), and the finite field coding system (p-adic system) (Krishnamurthy *et al.* [12], [13], Gregory [14]) for representing rationals. In this paper, the results obtained by applying the finite field transform method for Smith form reduction are presented.

In the p-adic number system (which is a generalization of the residue system), a modulus of the form p^r is chosen, where p is a prime and r is a positive integer. Any rational number a/b (where a and b are integers) is written in the form

$$\frac{a}{b}=\frac{c}{d}\cdot p^n$$

where c and d are integers, relatively prime to p. The p-adic representation of a/b is given by

$$\begin{cases} \cdot a_0a_1\cdots a_{r-1}, & n=0 \\ \cdot \underbrace{0\ 0\cdots 0}_{n\text{ zeros}}a_0a_1\cdots a_{r-n-1}, & n>0 \\ a_0a_1\cdots a_{n-1}\cdot a_n\cdots a_{r-1}, & n<0 \end{cases}$$

where

$$a_0+a_1p+a_2p^2+\cdots+a_{r-1}p^{r-1}=c\cdot d^{-1}\bmod p^r.$$

(d^{-1} is the multiplicative inverse of $d \bmod p^r$ when the gcd $(d,p)=1$.) This system can uniquely encode every order-N Farey fraction (i.e., every fraction of the form a/b with $0\leq|a|,|b|\leq N$, $b\neq 0$), where $N<\sqrt{(p^r-1)/2}$.

Arithmetic operations within the p-adic system are similar to those within a residue system and are explained in detail in Krishnamurthy *et al.* [12] and Gregory [14]. Although every order-N Farey fraction has a unique p-adic code, an efficient decoding algorithm is not available to reconstruct the rational. Hence, to facilitate decoding, a FACTOR term is included in the computation [14]. At the start of the computation, all values are made integral by multiplying each value by the lcm of all denominators. Then, during the procedure, any multiplication by a factor of the form a^{-1} (a integral) will result in the FACTOR being replaced by FACTOR$*a$. At the end of the computation, FACTOR contains the lcm of the denominators, and the numerator of any term x is obtained by the formula

$$x*\text{FACTOR} \bmod p^r.$$

As mentioned earlier, handling the coefficients in rational form during computation (in order to avoid numerical instability) will result in unnecessary coefficient growth in the absence of an explicit cancellation procedure to maintain the rationals in lowest reduced form. In the p-adic system, however, multiplication of the numerator and denominator by the same factor does not affect the p-adic representation of the rational. Hence, the p-adic code for any rational corresponds to the code for that rational in its lowest reduced form. Thus, this system ensures minimum growth in coefficient size during computation.

The p-adic coding and computation have been applied to the coefficients of the polynomial elements of the matrix in the Smith form reduction procedure of the next section.

III. Exact Smith Form Algorithm

In this section, we present the Smith form algorithm, using exact p-adic computation. The algorithm is basically that of Gantmacher [2], and incorporates many of the modifications suggested by Pace and Barnett [4]. The algorithm is written in "Pidgin Algol," a high-level language described in Aho, Hopcroft, and Ullman [15]. A Fortran implementation of the algorithm was used to compute the examples.

Algorithm 3.1: SNF

Input: An $m\times n$ single-variable polynomial matrix $A(\lambda)$, with integer coefficients, a bound on the polynomial degree d, and a bound on the coefficient size k, i.e., for every coefficient $a/b(a,b$, relatively prime) likely to occur in the procedure, $k>|a|,|b|$.

Output: An $m \times n$ diagonal polynomial matrix $S(\lambda)$, which is the Smith form of $A(\lambda)$.

Algorithm

begin$_0$

1) choose a prime p and a positive integer r such that $(p^r-1)/2 \geq k^2$.

2) *initialize*:
$S(\lambda) \leftarrow A(\lambda)$, with coefficients in p-adic code
FACTOR$\leftarrow$1
$i \leftarrow 1$

end$_0$

3) *while* $i < \min(m,n)$ *do*
comment all arithmetic operations are in the p-adic system
begin$_1$

4) *while* at least one nondiagonal element in ith row or ith column is nonzero *do*
begin$_2$

5) *comment* bring lowest degree term to the (i,i) position.
a) interchange s_{ii} with s_{pq}, a nonzero element of smallest degree in $S(\lambda)$ in the block including rows $i, i+1, \cdots, m$ and columns $i, i+1, \ldots, n$.
comment The search for s_{pq} is conducted in such a way that if such an element is present in the (i,i) position, it is not interchanged with another element of the same degree in the block.

6) *comment* make s_{ii} monic
a) multiply ith row of $S(\lambda)$ by a_0^{-1}, the inverse of the coefficient of the highest degree term in a_{ii}.
b) FACTOR$\leftarrow$FACTOR$*a_0$

7) *comment* reduce the degree of each nonzero element in ith column
for $j = i+1, i+2, \cdots, m$ *do*
begin$_3$
a) $R_1 \leftarrow a^{-1}$, where a is the coefficient of the highest degree term in a_{ij}
b) $R_2 \leftarrow \lambda^u$, where u = degree of a_{ij}—degree of a_{ii}
c) multiply jth row of $S(\lambda)$ by R_1
d) subtract R_2 times the ith row of $S(\lambda)$ from the jth row of $S(\lambda)$
e) FACTOR$\leftarrow$FACTOR$*a$
end$_3$

8) *comment* reduce the degree of each nonzero element in ith row
for $j = i+1, i+2, \cdots, n$ *do*
begin$_4$
a) $R_1 \leftarrow a^{-1}$, where a is the coefficient of the highest degree term in a_{ji}
b) $R_2 \leftarrow \lambda^u$, where u = degree of a_{ji} − degree of a_{ii}
c) multiply jth column of $S(\lambda)$ by R_1
d) subtract R_2 times the ith column of $S(\lambda)$ from the jth column of $S(\lambda)$
e) FACTOR$\leftarrow$FACTOR$*a$
end$_4$

end$_2$

9) find the gcd g(g monic) of s_{ii} and $s_{i-1,i-1}$ with premultipliers a and b such that $a \cdot s_{ii} + b \cdot s_{i-1,i-1} = g$

10) *comment* check if $s_{i-1,i-1} | s_{ii}$
if $g = s_{i-1,i-1}$ *then* $i \leftarrow i+1$
else
comment make $s_{i-1,i} = g$ and decrement i
begin$_5$
a) add a times the ith row of $S(\lambda)$ to row $(i-1)$
b) add b times column $(i-1)$ of $S(\lambda)$ to column i of $S(\lambda)$
c) $i \leftarrow i-1$

end$_5$

end$_1$

begin$_6$

11) *write* FACTOR

12) *for* $i = 1, 2, \cdots, m$ *do*

13) *for* $j = 1, 2, \cdots, n$ *do*
begin$_7$

14) *write* s_{ij}*FACTOR
end$_7$

end$_6$

Examples:

1)

$$A(X) = \begin{bmatrix} -2X^2 & -2X & 0 \\ 2X & 4-2X^2 & -X \\ 0 & X & -2X^2 \end{bmatrix}$$

prime $p = 8209$; $r = 1$.

Result:

FACTOR = 8

$$S(X) = \text{diag}[8, 8X^2, -6X^2+8X^4].$$

2)

$$A(X) = \begin{bmatrix} X & -1 & -1 & 0 & 1 \\ 0 & X & 0 & -1 & 0 \\ 0 & 0 & -1+X & 0 & 0 \\ 0 & 0 & 0 & -2+X & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$p^r = 8209$.

Result:

FACTOR = 1

$$S(X) = \text{diag}[1, 1, 1, 1, 2X-3X^2-X^3].$$

3)

$$A(X) = \begin{bmatrix} X & -1 & 0 & 0 & 0 \\ 0 & -1+X & 0 & 0 & 1 \\ 0 & 0 & -1+X & 0 & 0 \\ 0 & 0 & 0 & -1+X & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$p^r = 8209$.

Result:

FACTOR = 1

$$S(X) = \text{diag}[1, 1, 1, -1+X, -X+X^2].$$

4)

$$A(X) = \begin{bmatrix} -1+X & 0 & 0 & -1 & 1 & 0 \\ 0 & -1+X & 2 & -3 & 3 & 0 \\ 0 & 0 & 1+X & -2 & 2 & 0 \\ -1 & 1 & -1 & X & -1 & 0 \\ -1 & 1 & -1 & 1 & -2+X & 0 \\ 0 & 0 & 0 & 0 & 0 & X \end{bmatrix}$$

$p^r = 8209$.

Result:

FACTOR = −1

$$S(X) = \text{diag}[-1, -1, -1, -1, -1+2X-X^2, -X+X^2+X^3-X^4].$$

5)

$$A(X) = \begin{bmatrix} x & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1+X & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1+X & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1+X & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$p^r = 8209$.

Result:

$$\text{FACTOR} = 3$$
$$S(X) = \text{diag}[3,3,3,3,3,0,0].$$

IV. Discussion

As mentioned earlier, p and r should be chosen such that

$$\sqrt{\frac{p^r-1}{2}} \quad (=K)$$

is greater than the absolute value of the numerator or denominator of any rational (reduced to its lowest form) likely to occur during computation. However, considering the complex nature of the computation, it is not possible, in general, to compute an *a priori* bound for K. We suggest that some "reasonable" values can be chosen initially for p and r. After computation, the determinant of the largest nonzero principal minor in the Smith form can be compared with the gcd of the determinants of all nonzero minors of the same dimension in the input matrix. If the two do not agree to within a constant factor, then either p or r can be increased, and the computation repeated.

Although we have made no complexity analysis on this algorithm, and although we will not be surprised if Smith form reduction and determinant evaluation are proved to be of the same time complexity, we are quite certain that determinant evaluation is cheaper than Smith form reduction by at least a large constant factor. Hence, the above method of "pick and verify" is quite feasible especially since one should be able to predict a bound for K after some experience, if one is dealing with a particular class of matrices.

This method will take more time than floating-point computation in rational form, and this is mainly due to the p-adic inverse computation during division. On the other hand, this procedure is not subject to the errors of floating-point arithmetic and it requires less space than that required for explicit rational form computation. This is a simple example of time-space tradeoff. Thus, applications of Smith form reduction which cannot tolerate numerical instabilities and for which memory space is at a premium will find our procedure useful.

V. Conclusion

In this paper, we have presented an exact method for computing the Smith normal form of a polynomial matrix. This algorithm was developed with the view to mechanize completely the spectral factorization technique presented in Youla [1]. The subsequent step of factoring a unimodular polynomial matrix is under consideration. The algorithm presented in this paper, however, can be used in any application requiring the Smith form of a polynomial matrix. For instance, the Smith–Macmillan form (which is an extension of the Smith form for a matrix with rational polynomial entries) is useful in the analysis and minimal realization of transfer function matrices of time-invariant linear dynamical systems (Kalman [16], Rosenbrock [17]). The Smith form reduction is useful in determining if the closed-loop spectrum of a linear dynamical system can be freely assigned with decentralized control, and provides the unassignable polynomial when free spectrum assignment is not possible (Corfmat and Morse [18]). The Smith form can also be used to prove the existence of a solution to an integer programming problem (Barnett [19]).

A listing of the Fortran program of the algorithm is available with the author.

Acknowledgment

The author is grateful to Prof. E. V. Krishnamurthy for suggestions. She is also thankful to the Indian Institute of Science for the award of a research fellowship.

References

[1] D. C. Youla, "On the factorization of rational matrices," *IRE Trans. Inform. Theory*, vol. IT-7, pp. 172–189, July 1961.
[2] F. R. Gantmacher, *The Theory of Matrices*, vol. 1. New York: Chelsea, 1959.
[3] G. H. Bradley, "Algorithms for Hermite and Smith normal matrices and linear diophantine equations," *Math. Comput.*, vol. 25, pp. 897–907, 1971.
[4] I. S. Pace and S. Barnett, "Efficient algorithms for linear systems calculations, Part I—Smith form and common divisors of polynomial matrices," *Int. J. Syst. Sci.*, vol. 5, pp. 403–411, 1974.
[5] E. V. Krishnamurthy, "Exact inversion of a rational polynomial matrix using finite field transforms," *SIAM J. Appl. Math.*, vol. 35, pp. 453–464, Nov. 1978.
[6] T. M. Rao, K. Subramanian, and E. V. Krishnamurthy, "Residue arithmetic algorithms for exact computation of g-inverses of matrices," *SIAM J. Numer. Anal.*, vol. 13, pp. 155–171, Apr. 1976.
[7] E. H. Bareiss, "Sylvester's identity and multistep integer preserving Gaussian elimination," *Math. Comput.*, vol. 22, pp. 565–578, 1968.
[8] N. Munro and V. Zakian, "Inversion of rational polynomial matrices," *Electron. Lett.*, vol. 6, pp. 629–630, 1970.
[9] T. Downs, "Some properties of the Souriau–Frame algorithm with application to the inversion of rational matrices," *SIAM J. Appl. Math.*, vol. 28, pp. 237–251, 1975.
[10] M. T. McClellan, "The exact solution of systems of linear equations with polynomial coefficients," *J. Ass. Comput. Mach.*, vol. 20, pp. 563–588, 1973.
[11] D. M. Young and R. T. Gregory, *A Survey of Numerical Mathematics*. Reading, MA: Addison-Wesley, 1973.
[12] E. V. Krishnamurthy, T. M. Rao, and K. Subramanian, "Finite segment p-adic number systems with applications to exact computation," *Proc. Indian Acad. Sci.*, vol. LXXXI, sec. A, no. 2, pp. 58–79, 1975.
[13] E. V. Krishnamurthy, T. M. Rao, and K. Subramanian, "p-adic arithmetic procedures for exact matrix computations," *Proc. Indian Acad. Sci.*, vol. LXXXII, sec. A, no. 5, pp. 165–175, 1975.
[14] R. T. Gregory, "The use of finite-segment p-adic arithmetic for obtaining exact computational results," Dep. Comput. Sci., Univ. of Tennessee, Knoxville, Tech. Rep. CS-77-26, Dec. 1977.
[15] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.
[16] R. E. Kalman, "Irreducible realizations and the degree of a rational matrix," *J. Soc. Indust. Appl. Math.*, vol. 13, pp. 520–544, June 1965.
[17] H. H. Rosenbrock, *State-Space and Multivariable Theory*. London, England: Nelson, 1970.
[18] J. P. Corfmat and A. S. Morse, "Decentralized control of linear multivariable systems," *Automatica*, vol. 12, pp. 479–495, 1976.
[19] S. Barnett, *Matrices in Control Theory*. London, England: Van Nostrand Reinhold, 1971.

A Direct Way to Stabilize Continuous-Time and Discrete-Time Linear Time-Varying Systems

V. H. L. CHENG

***Abstract*—We derive, in a parallel fashion, state-feedback control laws for continuous-time and discrete-time linear time-varying systems; these laws assure that the zero solution of the closed-loop system is (globally) uniformly exponentially stable at a prescribed rate.**

Introduction

The problem of stabilizing "uniformly controllable" linear time-varying systems has been studied in [3], [7] for the continuous-time case, and in [4] for the discrete-time case. Their time-invariant counterparts can be found in [1] and [2], respectively. Without recourse to the linear quadratic regulator problem, we give here state-feedback control laws respectively for the continuous-time and discrete-time cases, which make the zero solution of the closed-loop systems (globally) uniformly exponentially stable (u.e.s.).[1] In contrast to the existing literature, the proofs of closed-loop stability given here do not involve the notion of

Manuscript received December 4, 1978. This work was supported by the National Science Foundation under Grant ENG76-84522.

The author is with the Department of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California, Berkeley, CA 94720.

[1]The zero solution of $\dot{x}(t) = A(t)x(t)$, $t \in \mathbf{R}_+$, is said to be u.e.s. on $\mathbf{R}_+$ at a rate (at least) μ iff $\exists \mu > 0$, $\exists M > 0$ such that $\forall t_0 \in \mathbf{R}_+$, $\forall x(t_0) \in \mathbf{C}^n$, $|x(t)| \leq |x(t_0)| M e^{-\mu(t-t_0)}$, $\forall t > t_0$.

The zero solution of $x_{i+1} = A_i x_i$, $i \in \mathbf{N} := \{0, 1, 2, \cdots\}$, is said to be u.e.s. on $\mathbf{N}$ at a rate (at least) μ iff $\exists \mu > 1$, $\exists M > 0$ such that $\forall i_0 \in \mathbf{N}$, $\forall x_{i_0} \in \mathbf{C}^n$, $|x_i| < |x_{i_0}| M (1/\mu)^{i-i_0}$ $\forall i \in \mathbf{N}$ with $i > i_0$.

0018-9286/79/0800-0641$00.75