MULTISHIFT ALGORITHM FOR POLE ASSIGNMENT OF SINGLE-INPUT SYSTEMS

A. Varga
DLR Oberpfaffenhofen
German Aerospace Research Establishment
Institute of Robotics and System Dynamics
Control Design Engineering
D-82234 Wessling, Germany
Andreas.Varga@dlr.de

Keywords: Pole assignment, numerical methods, implicit shift, multishift methods

Abstract

A new numerically reliable algorithm is proposed for the pole assignment of single-input linear time-invariant systems. The proposed algorithm is based on an implicit multishift QR-like technique, using exclusively orthogonal transformations to compute a feedback matrix which assigns the desired poles. The new algorithm allows for a dynamic adjustment of the number of shifts. This feature is particularly useful for implementation on high performance computers. An $n-2$ shift variant of the proposed method (n is the system order) uses minimum additional storage and is faster and simpler than all of existing numerically stable methods.

1 Introduction

We consider the following *eigenvalue assignment problem* (EAP): given the controllable matrix pair (A, b), where $A \in \mathbb{R}^{n,n}$ and $b \in \mathbb{R}^{n,1}$, determine the feedback matrix $f \in \mathbb{R}^{1,n}$ such that the closed-loop state matrix $A+bf$ has all its eigenvalues at desired locations $\Gamma = \{\lambda_1, \ldots, \lambda_n\}$ in the complex plane. We assume that Γ is symmetric with respect to the real axis. This assumption guarantees that the resulting f is real. There exist several numerically reliable algorithms which can be used to solve the EAP. It is commonly accepted that the most reliable methods are the so-called *Hessenberg methods* based on explicit or implicit QR-like techniques. Explicit shift methods have been proposed by Miminis and Paige [7] and Petkov, Christov and Konstantinov [12]. Implicit versions of the algorithm of [7] has been proposed by Miminis [6] and by Patel and Misra [11]. All these methods are numerically backward stable [3, 8] and generalizations of them for the multi-input case have been also proposed [11, 13, 8]. A discussion of Hessenberg methods is done in [1], where a stable variant of the algorithm of [4] is also developed. An alternative to the above methods is the so-called *Schur method* proposed by Varga [16]. Although computationally more involved than the Hessenberg methods, the Schur method has the attractive feature to allow a partial pole assignment, *i.e.* it is possible to alter only those eigenvalues of A which are unsatisfactory for the closed-loop system dynamics and to keep unmodified the rest of eigenvalues.

The Hessenberg methods fit in the following algorithmic template: the pair (A, b) is first transformed to the *controller-Hessenberg form* (CHF), the feedback is then computed for the reduced problem, and then finally the solution is recovered in the original coordinate system. Recall that a pair (A, b) is in CHF if b has all but its first component zero and A is in an unreduced Hessenberg form, *i.e.* all its elements on the first sub-diagonal are nonzero. The existence of the CHF is guaranteed by the assumption of the controllability of the pair (A, b). This template is common to all Hessenberg methods and has the following main steps:

1. Reduce the pair (A, b) by using an orthogonal matrix Q to the CHF
   $(H, \beta e_1) = (Q^T A Q, Q^T b)$.

2. Compute $h \in \mathbb{R}^{1,n}$ such that $\lambda(H + \beta e_1 h) = \Gamma$.

3. Compute $f = hQ^T$.

The computation of the CHF can be done by using a sequence of $n-1$ orthogonal Householder reflectors. The reduction technique is standard (see for example [14], [15]) and is not discussed further here. The transformation matrix Q used at step 1 is not necessary to be explicitly computed. It can be stored in a factored form, by retaining only the elements of the Householder reflectors. The

minimal necessary storage is only $n^2/2 + 0(n)$ storage locations. The application of these reflectors at step 3 to h requires only $0(n^2)$ *floating-point operations* (*flops*) and thus steps 1 and 3 require together roughly $(2/3)n^3$ *flops*, representing the cost to reduce A to the Hessenberg form without accumulating the orthogonal transformations.

The Hessenberg methods differ in the ways of computing h at step 2. In this paper we propose a multi-step multishift implicit QR-like Hessenberg method to solve the EAP. The proposed algorithm can be viewed as a generalization of the explicit and implicit double shift methods proposed in [7] and [11], respectively. A particular one-step multishift variant of the proposed method is very well suited for computer implementations. Besides its simplicity, this variant is computationally very efficient, easy to implement and requires minimal additional storage. From all these points of view it compares favourably with the existent Hessenberg methods.

2 Multishift Hessenberg Method

In this section we describe an implicit variable-multishift Hessenberg method for pole assignment. We assume that the pair $(A_1, b_1) := (H, \beta e_1)$ is already in the CHF. For the computation of the feedback h such that $\lambda(H + \beta e_1 h) = \Gamma$, a recursive deflation technique based exclusively on orthogonal transformations is used. At each deflation step a number of k eigenvalues are allocated, where k can vary from step to step. In order to keep the computations in real domain, we impose that the set of k eigenvalues to be assigned at each step is symmetric with respect to the real axis.

We illustrate the deflation process by considering the allocation of a symmetric subset of $k \leq n-2$ eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ for the pair (A_1, b_1) by a feedback h_1. First compute the last row of the matrix

$$N = (A_1 - \lambda_1 I)(A_1 - \lambda_2 I) \cdots (A_1 - \lambda_k I).$$

Notice that because of the form of (A_1, b_1), the last row of N is the same as the last row of $(A_1 + b_1 h_1 - \lambda_1 I) \cdots (A_1 + b_1 h_1 - \lambda_k I)$ and has only the last $k+1$ elements nonzero and real. That is, we have

$$e_n^T N = [\, 0, \ldots, 0, \gamma_{n-k}, \ldots, \gamma_n \,]$$

and according to the theory of implicit shifting [9], [2] we choose a Householder transformation $\mathcal{H}_1$ to annihilate all but the last nonzero element of $e_n^T N$ and we compute $\mathcal{H}_1 A_1 \mathcal{H}_1$. For $n = 7$ and $k = 3$, the matrix $\mathcal{H}_1 A_1 \mathcal{H}_1$ has the form

$$\mathcal{H}_1 A_1 \mathcal{H}_1 = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times & \times \\ 0 & 0 & + & \times & \times & \times & \times \\ 0 & 0 & + & + & \times & \times & \times \\ 0 & 0 & + & + & + & \times & \times \end{bmatrix}$$

where + stands for nonzero elements resulted by applying $\mathcal{H}_1$. The nonzero elements introduced by $\mathcal{H}_1$ can be now chased upwards, by a series of $n-k-1$ Householder transformations $\mathcal{H}_i$, $i = 2, \ldots, n-k$ such that the trailing principal submatrix of order $(n-k)$ of $P^T A_1 P$, where $P = \mathcal{H}_1 \mathcal{H}_2 \cdots \mathcal{H}_{n-k}$, is in an upper Hessenberg form. For the example considered above, the matrices $\bar{A}_1 = P^T A_1 P$ and $\bar{b}_1 = P^T b_1$ have the form

$$\bar{A}_1 = \left[\begin{array}{ccc|cccc} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ \hline * & * & * & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & \times & \times \end{array}\right], \quad \bar{b}_1 = \left[\begin{array}{c} \times \\ \times \\ \times \\ \hline \times \\ 0 \\ 0 \\ 0 \end{array}\right]$$

The $(k+1)$-th element of $\bar{b}_1$ is nonzero and thus the first k elements in row $k+1$ of $\bar{A}_1$ can be annihilated by choosing a suitable feedback $\bar{h}_1$. From the partitioning of the pair $(\bar{A}_1 + \bar{b}_1 \bar{h}_1, \bar{b}_1)$ in the form

$$\bar{A}_1 + \bar{b}_1 \bar{h}_1 = \begin{bmatrix} \bar{A}_{11} & \bar{A}_{12} \\ 0 & A_2 \end{bmatrix}, \qquad \bar{b}_1 = \begin{bmatrix} * \\ b_2 \end{bmatrix} \tag{1}$$

we have that $\lambda(\bar{A}_{11}) = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ and A_2 is upper Hessenberg. For our example, we choose $\bar{h}_1$ of the form

$$\bar{h}_1 = \left[\begin{array}{ccc|cccc} \times & \times & \times & 0 & 0 & 0 & 0 \end{array}\right]$$

where the nonzero elements denoted with $\times$ are chosen to annihilate the elements of $\bar{A}_1$ denoted by $*$. The feedback matrix h_1 which assigns k eigenvalues of $A_1 + b_1 h_1$ is given by $h_1 = \bar{h}_1 P^T$. The deflation process continues by applying the same procedure to the pair (A_2, b_2) in HCF defined in (1) with a new set of eigenvalues to be assigned. To assign the last two eigenvalues, explicit formulas should be used. The following algorithm summarizes the above steps. We assume that H is an unreduced Hessenberg matrix and $\beta \neq 0$.

Algorithm 1. Multi-step variable-multishift single-input pole assignment.

1. Set $A_1 = H$, $b_1 = \beta e_1$, $P = I$, $h = 0$, $r = n$, $i = 1$.
2. If $r = 1$ or $r = 2$, compute h_i such that
$$\lambda(A_i + b_i h_i) = \Gamma.$$
Put $h \leftarrow h + [\, 0 \; h_i] P^T$ and *Stop*.
3. Choose a symmetric set of $k_i < r - 2$ eigenvalues $\Gamma_i = \{\lambda_1, \lambda_2, \ldots, \lambda_{k_i}\} \subset \Gamma$ and compute
$$y = e_r^T (A_i - \lambda_1 I)(A_i - \lambda_2 I) \cdots (A_i - \lambda_{k_i} I)$$
4. Choose a Householder reflector $\mathcal{H}_1$ to annihilate the first $r-1$ components of y. Compute $A_i \leftarrow \mathcal{H}_1 A_i \mathcal{H}_1$ and $P \leftarrow P \mathrm{diag}(I_{n-r}, \mathcal{H}_1)$.

5. Choose $r-k_i-1$ Householder reflectors $\mathcal{H}_j$, $j=2,\ldots,r-k_i$ such that the trailing principal block of order $(r-k_i)$ of $P_i^TA_iP_i$, where $P_i=\mathcal{H}_2\cdots\mathcal{H}_{r-k_i}$, is in an upper Hessenberg form. Put $A_i\leftarrow P_i^TA_iP_i$, $b_i\leftarrow\mathcal{H}_{r-k_i}b_i$ and $P\leftarrow P\mathrm{diag}(I_{n-r},P_i)$.

6. Choose h_i such that the first k_i elements in row k_i+1 of $A_i+b_ih_i$ are zero. Put $A_i\leftarrow A_i+b_ih_i$. Define

$$A_i:=\begin{bmatrix}*&*\\0&A_{i+1}\end{bmatrix},\qquad b_i:=\begin{bmatrix}*\\b_{i+1}\end{bmatrix}$$

7. Set $h\leftarrow h+[\ 0\ h_i]P^T$, $\Gamma\leftarrow\Gamma\setminus\Gamma_i$, $r\leftarrow r-k_i$, $i\leftarrow i+1$ and *go to* Step 2.

Remark 1. The matrix $P^T(H+\beta e_1h)P$, which can be optionally computed in place of H, results in an upper block-triangular form, with successive diagonal blocks of orders k_i. If k_i is always two, then the above algorithm is identical to the implicit double-shift variant of the Miminis and Paige algorithm described in [11]. In this case the above matrix is in a quasi-triangular form with diagonal blocks of order at most two corresponding to the assigned eigenvalues. □

Remark 2. By using the arguments of Arnold [1], it is very likely that the proposed multishift algorithm is backward numerically stable. The accuracy of method however depends crucially on the accuracy of shift information contained in the Householder reflector $\mathcal{H}_1$ computed at step 4. It is worth mentioning that roundoff errors can create difficulties even when one is working with a single shift as observed by Parlett in the case of the QR-algorithm [10]. Provided the shifts information are sufficiently accurate, it is to be expected that for moderate values of k_i (say $k_i\leq 10$) the multishift algorithm has the same accuracy as the simple or double-shift methods. □

Remark 3. The number of performed operation depends on the values of k_i. If $k_i=k$ is constant at each step, then Algorithm 1 performs about $(2/3)n^3+k^3/6+(2k+1)n^3/(3k)$ *flops*. The necessary additional storage when the intermediary transformations are stored in factored form is about $(k+2)n(n-k)/(2k)+0(n)$ storage locations. □

Remark 4. It has been observed that the accuracy of the shift vector y computed at step 3 is sensitive to the order in which the shifts enter in the computations. In order to avoid the incorporation of some sorting procedure in the implementation of the algorithm, we can use the scheme proposed in [5] to compute first the coefficients of the polynomial

$$\begin{aligned}p(\lambda)&=(\lambda-\lambda_1)(\lambda-\lambda_2)\cdots(\lambda-\lambda_{k_i})\\&=c_{k_i}\lambda^{k_i}+\ldots+c_1\lambda+c_0\end{aligned}$$

and then to evaluate the vector y as $y=e_r^Tp(A_i)$. This computation can be done efficiently by using a Horner-like polynomial evaluation scheme as for instance

1. Set $y=0$ and $f=e_r^T$.

2. For $i=1,\ldots,k_i+1$
   $y\leftarrow c_{k_i-i+1}f+yA_i$

The operations above are column oriented and can exploit efficiently the Hessenberg structure of A_i. □

3 One-step Multishift Method

An one-step variant of Algorithm 1 can be easily derived by performing an implicit multishift for $n-2$ eigenvalues. The resulting algorithmic variant is particularly advantageous for computer implementation primarily because its simplicity. It can be implemented with the least additional storage and requires the least number of operations among all known techniques. Therefore we present it explicitly bellow.

Algorithm 2. One-step multishift single-input pole assignment.

1. Set $A_1=H$, $b_1=\beta e_1$.

2. If $n=1$ or $n=2$, compute h such that $\lambda(A_1+b_1h)=\Gamma$ and *Stop*.

3. Compute

$$y=e_n^T(A_1-\lambda_1I)(A_1-\lambda_2I)\cdots(A_1-\lambda_{n-2}I).$$

4. Choose a Householder reflector $\mathcal{H}_1$ to annihilate the first $n-1$ components of y.
   Compute $A_1\leftarrow\mathcal{H}_1A_1\mathcal{H}_1$.

5. Choose a Householder reflector $\mathcal{H}_2$ to annihilate the first $n-2$ elements in the last row of A_1. Compute $A_1\leftarrow\mathcal{H}_2A_1\mathcal{H}_2$ and $b_1\leftarrow\mathcal{H}_2b_1$.

6. Compute $h_{1,i}=-a_{n-1,i}/b_{n-1,1}$ for $i=1,\ldots,n-2$ and choose the last two components of h such that λ_{n-1} and λ_n are the eigenvalues of the matrix

$$\begin{bmatrix}a_{n-1,n-1}&a_{n-1,n}\\a_{n,n-1}&a_{n,n}\end{bmatrix}+\begin{bmatrix}b_{n-1,1}\\0\end{bmatrix}\begin{bmatrix}h_{1,n-1}&h_{1,n}\end{bmatrix}$$

7. Set $h\leftarrow h\mathcal{H}_2\mathcal{H}_1$.

Remark 4. The evaluation of the shift vector y at step 3 amounts to about $n^3/6$ *flops*. All other computations requires $0(n^2)$ *flops* and thus are practically negligible. The necessary additional storage is $2n$ locations. Thus, Algorithm 2 is very efficient from both the points of views of computational effort as well of storage requirements. However the numerical stability of Algorithm 2 is questionable because its resemblance with Datta's method [4]. □

4 Comparisons with Other Methods

It is interesting to compare the performances of the proposed methods with those of existing ones. Table 1 contains the number of operations N_{op} and the necessary storage M_{loc} for different methods, including also the reduction of the given system to the CHF. It is apparent from this table that for $k > 2$ Algorithm 1 is more efficient than all previously proposed numerically stable methods with respect to both performance criteria. Notice that Algorithm 2 is much simpler to be implemented and has even better performance figures than Algorithm 1.

Table 1.

Methods	N_{op}	M_{loc}	Numerical Stability
Miminis & Paige [7]	$\frac{2}{3}n^3 + \frac{5}{6}n^3$	$\frac{3}{2}n^2$	yes
Petkov *et al.* [12]	$\frac{2}{3}n^3 + \frac{5}{3}n^3$	n^2	yes
Patel & Misra [11]	$\frac{2}{3}n^3 + \frac{5}{6}n^3$	$\frac{3}{2}n^2$	yes
Datta [4]	$\frac{2}{3}n^3 + \frac{1}{6}n^3$	$\frac{1}{2}n^2$	no
Modified Datta [1]	$\frac{2}{3}n^3 + \frac{5}{6}n^3$	$\frac{3}{2}n^2$	yes
Schur method [15]	$14n^3$	n^2	likely
Algorithm 1	$\frac{2}{3}n^3 + \frac{1}{6}k^3 + \frac{2k+1}{3k}n^3$	$\frac{1}{2}n^2 + \frac{k+2}{2k}n(n-k)$	very likely
Algorithm 2	$\frac{2}{3}n^3 + \frac{1}{6}n^3$	$\frac{1}{2}n^2$	questionable

We performed several tests to assess the accuracy of proposed methods. As it is to be expected, for moderate values of the shift parameter k ($k \leq 10$) the accuracy of Algorithm 1 is similar to the accuracy of equivalent Hessenberg methods [8, 12, 11]. We also compared the accuracy of its extreme variant (Algorithm 2), where practically all eigenvalues are assigned in a single step by an appropriately chosen implicit multiple shift. Two other methods were included in the test runs, namely the method of Miminis and Paige [7] and the Schur method [15]. All tests have been performed on an IBM RS 6000 computer, in double precision, by using MATLAB implementations of the mentioned algorithms.

In a first run, we generated randomly the matrices A and b for different values of n and we chose as desired eigenvalues the eigenvalues of A. The resulting feedback f should be zero, and thus the resulting deviation from zero can be viewed as a measure of the accuracy of the particular methods. In Table 2. we included the resulting values for $\max_i |f_i|$ for different values of n. It can be observed that for moderate values of n, ($n \leq 50$) all methods manifest approximately the same accuracy, but for values of $n \geq 50$ the accuracy of the one-step multishift method is progressively worse than for the other three methods. Notice that for this kind of test, the Schur method performed systematically better than the Hessenberg methods.

Table 2. Values of $\max_i |f_i|$.

Methods	$n = 5$	$n = 10$	$n = 20$	$n = 50$	$n = 100$
Miminis–Paige Method	$4.1 \cdot 10^{-15}$	$1.3 \cdot 10^{-14}$	$1.2 \cdot 10^{-14}$	$1.5 \cdot 10^{-13}$	$7.0 \cdot 10^{-13}$
Schur Method	$8.6 \cdot 10^{-16}$	$4.3 \cdot 10^{-15}$	$2.0 \cdot 10^{-15}$	$4.4 \cdot 10^{-14}$	$1.7 \cdot 10^{-13}$
Algorithm 1 ($k = 5$)	$2.9 \cdot 10^{-15}$	$1.1 \cdot 10^{-14}$	$1.3 \cdot 10^{-14}$	$9.1 \cdot 10^{-14}$	$8.9 \cdot 10^{-13}$
Algorithm 2	$2.9 \cdot 10^{-15}$	$8.2 \cdot 10^{-15}$	$1.7 \cdot 10^{-14}$	$1.4 \cdot 10^{-12}$	$2.4 \cdot 10^{-8}$

In a second test run we compared for randomly generated eigenvalues, the accuracy of the eigenvalues resulting after pole assignment. For each value of n we computed the quantities $\max_i |\bar{\lambda}_i - \lambda_i|$, where $\bar{\lambda}_i$ is the i-th eigenvalue of $A + bf$. The obtained results are contained in Table 3 and also confirms that for moderate values of n ($n \leq 50$) the accuracy of all four methods is very similar. The best accuracy for the assigned eigenvalues has been achieved this time systematically with the Miminis–Paige method and with Algorithm 1.

Table 3. Values of $\max_i |\bar{\lambda}_i - \lambda_i|$.

Methods	$n = 5$	$n = 10$	$n = 20$	$n = 50$	$n = 100$
Miminis–Paige Method	$2.8 \cdot 10^{-15}$	$7.5 \cdot 10^{-15}$	$6.2 \cdot 10^{-15}$	$3.1 \cdot 10^{-13}$	$3.1 \cdot 10^{-11}$
Schur Method	$1.3 \cdot 10^{-15}$	$6.1 \cdot 10^{-15}$	$1.0 \cdot 10^{-14}$	$1.1 \cdot 10^{-11}$	$1.2 \cdot 10^{-9}$
Algorithm 1 ($k = 5$)	$2.6 \cdot 10^{-15}$	$6.5 \cdot 10^{-15}$	$8.2 \cdot 10^{-15}$	$3.6 \cdot 10^{-13}$	$3.7 \cdot 10^{-11}$
Algorithm 2	$2.6 \cdot 10^{-15}$	$1.0 \cdot 10^{-14}$	$9.9 \cdot 10^{-14}$	$2.2 \cdot 10^{-12}$	$6.6 \cdot 10^{-7}$

A word of caution is necessary here. The results of the previous test are meaningful only if the eigenvalue problem for $A + bf$ is well-conditioned. This is the case for our randomly generated data. It is however not difficult to generate low order examples leading to very ill-conditioned eigenvalue problems for $A + bf$. Because of numerical stability, the results computed by either of above algorithms could be correct to full accuracy (exact results can be generated symbolically). However the accuracy of numerically computed eigenvalues of $A + bf$ could be very poor even for the exact symbolically computed feedback f.

5 Conclusion

A numerically reliable multishift QR-like algorithm for single-input pole assignment has been proposed. The basic method is a multi-step variable-multishift algorithm whose numerical performances (operation count, memory usage, accuracy) are at least as good as those of other existing numerically stable methods. The method is well suited for implementation on high performance computers. A very simple to implement one-step variant of the basic method has been explicitly described. This variant requires the least computational effort and storage among all existing algorithms and is well suited for moderate order systems ($n \leq 50$). The proposed multishift methods can be readily extended to assign poles of multi-input systems [11] as well as of generalized state space systems.

We believe that the proposed multishift algorithm for pole assignment is a viable alternative to existing explicit or implicit double-shift methods. Especially on high performance computers, we expect similar performances for this algorithm to that of multishift methods for eigenvalue computations [2].

References

[1] M. Arnold. *Algorithms and Conditioning for Eigenvalue Assignment*. PhD thesis, Northern Illinois University, Dekalb, Illinois, 1992.

[2] Z. Bai and J. W. Demmel. On a block implementation of the Hessenberg multishift QR iteration. *Int. J. High-Speed Comp.*, 1:97–112, 1989.

[3] C. L. Cox and W. F. Moss. Backward error analysis for a pole assignment algorithm. *SIAM J. Matrix Anal. Appl.*, 10:446–456, 1989.

[4] B. N. Datta. An algorithm to assign eigenvalues in a Hessenberg matrix: single input case. *IEEE Trans. Autom. Control*, AC-32:414–417, 1987.

[5] A. A. Dubrulle and G. H. Golub. A multishift QR iteration without computation of the shifts. Numer. Anal. Project, NA-93-04, Computer Science Dept., Stanford University, 1993.

[6] G. Miminis. *Numerical Algorithms for Controllability and Eigenvalue Allocation*. M. Sc. thesis, School of Computer Science, McGill University, 1981.

[7] G. S. Miminis and C. C. Paige. An algorithm for pole assignment of time invariant linear systems. *Int. J. Control*, 35:341–354, 1982.

[8] G. S. Miminis and C. C. Paige. A direct algorithm for pole assignment of time-invariant multi-input linear systems using state feedback. *Automatica*, 24:343–356, 1988.

[9] G. S. Miminis and C. C. Paige. Implicit shifting in the QR and related algorithms. *SIAM J. Matrix Anal. Appl.*, 12:385–400, 1991.

[10] B. N. Parlett and J. Le. Forward instability of tridiagonal QR. *SIAM J. Matrix Anal. Appl.*, 14:279–316, 1993.

[11] R. V. Patel and P. Misra. Numerical algorithm for eigenvalue assignment by state feedback. *Proc. IEEE*, 72:1755–1764, 1984.

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

[13] P. Hr. Petkov, N. D. Christov, and M. M. Konstantinov. A computational algorithm for pole assignment of linear multiinput systems. *IEEE Trans. Autom. Control*, AC-31:1755–1764, 1986.

[14] P. Van Dooren. The generalized eigenstructure problem in linear systems theory. *IEEE Trans. Autom. Control*, AC-26:111–129, 1981.

[15] A. Varga. Numerically stable algorithm for standard controllability form determination. *Electron. Lett.*, 17:74–75, 1981.

[16] A. Varga. A Schur method for pole assignment. *IEEE Trans. Autom. Control*, AC-26:517–519, 1981.