A Multishift Hessenberg Method for Pole Assignment of Single-Input Systems

Andras Varga

Abstract— **A new algorithm is proposed for the pole assignment of single-input linear time-invariant systems. The proposed algorithm belongs to the family of Hessenberg methods and is based on an implicit multishift QR-like technique. The new method compares favorably in many respects (speed, memory usage) with existing numerically stable methods. Its improved vectorizability guarantees good opportunities for parallel implementation on high performance computers.**

I. 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 [1] and Petkov, Christov and Konstantinov [2]. Implicit versions of the algorithm of [1] has been proposed by Miminis [3] and by Patel and Misra [4]. All these methods are numerically backward stable [5], [6] and generalizations of them for the multi-input case have been also proposed [4], [7], [6]. Hessenberg methods are discussed in [8], where a stable variant of the algorithm of [9] is also developed. An alternative to the above methods is the so-called *Schur method* proposed by Varga [10]. 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 [11], [12]) and is not discussed further here. The transformation matrix Q used at Step 1) need not 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 [1] and [4], respectively. A particular one-step multishift variant of the proposed method is very well suited for computer implementation. 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.

II. Multishift Hessenberg Method

In this section we describe an implicit variable-multishift Hessenberg method for pole assignment for a pair $(H, \beta e_1)$ 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.

The derivation of the multishift pole assignment algorithm relies on the following result of Miminis and Page [13] on implicit multishift QR algorithm for eigenvalues determination.

Lemma 1: Let H be an unreduced Hessenberg matrix, let $\lambda_1, \ldots, \lambda_k \in \Lambda(H)$ and

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

Define the *implicit k-shift procedure* by the following two steps:

1) Choose a Householder reflector P_1 such that $e_n^T N P_1 = e_n^T \rho$
2) Compute $H_2 = P_1 H P_1$ and the Householder reflectors P_i such that

$$H_{i+1} = P_i H_i P_i, \quad i = 2, \ldots, n-k$$

with trailing i rows and columns in Hessenberg form.

Then, with $P = P_1 P_2 \cdots P_{n-k}$

$$\begin{aligned} H_{n-k+1} &= P^T H P = \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix} \\ \Lambda(H_{11}) &= \{\lambda_1, \ldots, \lambda_k\}. \end{aligned}$$

To apply this result to the EAP, we will assume that we already determined h such that $\Lambda(H + \beta e_1 h) = \Gamma$, and we apply the above procedure to $H' = H + \beta e_1 h$. Because H and H' differ only in their first rows, the matrices N and N', where

$$N' = (H' - \lambda_1 I)(H' - \lambda_2 I) \cdots (H' - \lambda_k I),$$

will have the same last $n-k$ rows. Thus the Householder transformation P_1 computed for H' can be computed by only using H. We also observe that the rest of Householder matrices P_i, $i = 2, \ldots, n-k$ computed for H' can be also computed by only

Manuscript received September 8, 1995; revised December 4, 1995.

The author is with the German Aerospace Research Establishment, DLR – Oberpfaffenhofen, Institute for Robotics and System Dynamics, D-82234 Wessling, Germany (e-mail: andreas.varga@dlr.de).

Publisher Item Identifier S 0018-9286(96)08436-X

using H, and thus also P is the same. We obtain by using Lemma 1 that

$$P^T H' P = \left[\begin{array}{c|c} H'_{11} & H'_{12} \\ \hline 0 & H'_{22} \end{array}\right], \quad \Lambda(H'_{11}) = \{\lambda_1, \ldots, \lambda_k\}.$$

By applying the same transformations to H and βe_1 we obtain

$$P^T H P = \left[\begin{array}{c|c} H_{11} & H_{12} \\ \hline \sigma & \\ \hline 0 & H_{22} \end{array}\right], \quad P^T \beta e_1 = \left[\begin{array}{c} b_1 \\ \hline \beta' \\ \hline 0 \end{array}\right].$$

By comparing $P^T H' P$ and $P^T H P$ it follows that

$$h = [\, -\sigma/\beta' \;\; h' \,] P^T \tag{1}$$

where h' is chosen such that $\Lambda(H_{22} + \beta' e_1 h') = \{\lambda_{k+1}, \ldots, \lambda_n\}$, that is h' solves a pole assignment problem for the rest of eigenvalues. If we separate h as $h = h_1 + h_2$, where

$$h_1 = [\, -\sigma/\beta' \;\; 0 \,] P^T, \quad h_2 = [\, 0 \;\; h' \,] P^T$$

it is clear that h_1 can be used as a partial feedback to deflate the pole assignment problem. Then an $n-k$ order problem, of the same form as the original problem, can be solved for the pair $(H_{22}, \beta' e_1)$ to determine h' which assigns the rest of the eigenvalues.

We illustrate the deflation process for $n = 7$ and $k = 3$ by considering the allocation of a symmetric subset of eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$ for the pair $(A_1, b_1) := (H, \beta e_1)$ by a feedback h_1. The last row of the matrix

$$N = (A_1 - \lambda_1 I)(A_1 - \lambda_2 I)(A_1 - \lambda_3 I)$$

has only the last $k+1 = 4$ elements nonzero, that is we have

$$e_n^T N = \left[\begin{array}{ccccccc} 0 & 0 & 0 & \times & \times & \times & \times \end{array}\right]$$

According to the theory of implicit shifting we choose a Householder transformation P_1 to annihilate all but the last nonzero element of $e_n^T N$ and we compute $P_1 A_1 P_1$. The matrix $P_1 A_1 P_1$ has the form

$$P_1 A_1 P_1 = \left[\begin{array}{ccccccc} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times & \times \\ 0 & 0 & + & \times & \times & \times & \times \\ 0 & 0 & + & + & \times & \times & \times \\ 0 & 0 & + & + & + & \times & \times \end{array}\right]$$

where $+$ stands for nonzero elements created by applying P_1. The nonzero elements introduced by P_1 can be now chased upwards, by a series of $n-k-1$ Householder transformations P_i, $i = 2, \ldots, n-k$ such that the trailing principal submatrix of order $(n-k)$ of $P^T A_1 P$, where $P = P_1 P_2 \cdots P_{n-k}$, 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 forms

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

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 $\bar{h}_1$ of the form

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

From the partitioning of the pair $(\bar{A}_1 + \bar{b}_1 \bar{h}_1, \bar{b}_1)$ in the form

$$\bar{A}_1 + \bar{b}_1 \bar{h}_1 := \left[\begin{array}{cc} \bar{A}_{11} & \bar{A}_{12} \\ 0 & A_2 \end{array}\right], \qquad \bar{b}_1 := \left[\begin{array}{c} * \\ b_2 \end{array}\right] \tag{2}$$

we have that $\Lambda(\bar{A}_{11}) = \{\lambda_1, \lambda_2, \lambda_3\}$ and A_2 is in unreduced upper Hessenberg form. 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) defined in (2), in HCF, with a new set of eigenvalues to be assigned. To assign the last two eigenvalues, explicit formulas must 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 2, compute h_i such that $\Lambda(A_i + b_i h_i) = \Gamma$, $h \leftarrow h + [\, \underbrace{0 \cdots 0}_{n-r} \; h_i \,] P^T$ and *Stop*.
3) Choose a symmetric set of $k_i \leq r-2$ 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 P_1 to annihilate the first $r-1$ components of y and compute $A_i \leftarrow P_1 A_i P_1$ and $P \leftarrow P \mathrm{diag}(I_{n-r}, P_1)$.
5) Choose $r - k_i - 1$ Householder reflectors P_j, $j = 2, \ldots, r - k_i$ such that the trailing principal block of order $(r-k_i)$ of $U_i^T A_i U_i$, where $U_i = P_2 \cdots P_{r-k_i}$, is in an upper Hessenberg form. Compute $A_i \leftarrow U_i^T A_i U_i$, $b_i \leftarrow U_i^T b_i$ and $P \leftarrow P \mathrm{diag}(I_{n-r}, U_i)$.
6) Choose h_i such that the first k_i elements in row $k_i + 1$ of $A_i + b_i h_i$ are zero. Define
$$A_i + b_i h_i := \left[\begin{array}{cc} * & * \\ 0 & A_{i+1} \end{array}\right], \qquad b_i := \left[\begin{array}{c} * \\ b_{i+1} \end{array}\right]$$
7) Set $h \leftarrow h + [\, \underbrace{0 \cdots 0}_{n-r} \; h_i \,] P^T$, $\Gamma \leftarrow \Gamma \setminus \Gamma_i$, $r \leftarrow r - k_i$, $i \leftarrow i+1$ and *go to* Step 2).

The main advantage of the multishift QR-like algorithms in comparison with single or double shift methods is that by taking larger values for the shift parameter k the vectorizability of the implemented code is improved. The main operations can be expressed as matrix-vector products, and level 2 BLAS [14] can be used. It is also possible to organize the algorithm so that several columns are chased at a time, using the WY representation of reflectors [15]. This allows to express the basic computations as matrix-matrix products, and the algorithm can be coded using level 3 BLAS [16]. This increases the parallelism of computations and the efficiency of hierarchical memory usage. Such a version was implemented by Bai and Demmel [17] for the QR algorithm.

The matrix $P^T(H + \beta e_1 h)P$, 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 matrix is in a quasi-triangular form with diagonal blocks of order at most two corresponding to the assigned eigenvalues. In this case the above algorithm is identical to the implicit double-shift variant of the Miminis and Paige algorithm [1] described in [4]. Because the multishift algorithm is an extension of the implicit double-shift method, it is to be expected that for moderate values of k_i (say $k_i \leq 10$) the proposed multishift algorithm has similar numerical stability properties as the numerically stable double-shift methods of [1], [4]. The proof of numerical stability is still under investigation.

The number of performed operation depends on the values of k_i. If $k_i = k$ is constant at each iteration, then in one iteration Algorithm 1 performs about $k^3/6 + (2k+1)r(r-k)$ *flops*, where the first term accounts for the evaluation of the shift vector y at Step 3) and the second term accounts for the application of the *modified* Householder transformations to A_i at Steps 4) and 5). We assumed that the intermediary Householder transformations are stored in factored form and thus are not accumulated. The following table shows the total number of operations N^1_{op} and the required additional memory M^1_{loc} for some particular values of k, considering that in average n/k iterations are performed by the algorithm.

k	$k \ll n$	2	10	$n/2$	n-2
N^1_{op}	$\frac{2k+1}{3k}n^3$	$\frac{5}{6}n^3$	$\frac{2}{3}n^3$	$\frac{1}{2}n^3$	$\frac{1}{6}n^3$
M^1_{loc}	$\frac{k+1}{2k}n^2$	$\frac{3}{4}n^2$	$\frac{1}{2}n^2$	$\frac{1}{4}n^2$	$2n$

Notice that both the number of operations and the necessary memory decrease with increasing values of k. For instance, for $k = 10$ Algorithm 1 needs approximately $n^3/6$ less operations than for $k = 2$.

Remark. A distinct non-recursive one-step variant of Algorithm 1 can be derived by performing an implicit multishift for $n-2$ eigenvalues [18]. The resulting algorithmic variant is particularly advantageous for computer implementation primarily because its simplicity and efficiency. Only two orthogonal reflectors must be determined: P_1, which provides the shift information, and P_2 which annihilates the first $n-2$ elements in the last row of $P_1A_1P_1$. Then h is computed in the form (1), where $P = P_1P_2$. This algorithm can be implemented with $0(n)$ additional storage and requires only $n^3/6$ *flops* (to evaluate the shift vector y). Thus this algorithmic variant is one of the most efficient algorithms available to solve the single-input pole assignment problem. □

III. Computation of Shift Information

The accuracy and the roundoff properties of the proposed implicit multishift method depend crucially on the accuracy of shift information contained in the Householder reflector P_1 computed at Step 4) of Algorithm 1. Provided the shift information are sufficiently accurate, it is to be expected that the multishift algorithm has the same accuracy as the single or double-shift methods. It is worth mentioning in this context that roundoff errors can create difficulties (forward instability) even when one is working with a single shift as observed by Parlett in the case of the QR-algorithm [19].

In this section we discuss two approaches to compute the implicit shift information. In the first approach we evaluate explicitly the shift vector y and then we compute the orthogonal reflector P_1 which annihilate the first $r-1$ components of y. In order to avoid potential problems with overflow or underflow during computing the shift vector y, it is recommended to compute instead of y the quantity $z = y/\|y\|$ using an appropriate norm. This can be done efficiently by the recursion $z \leftarrow z(A_i - \lambda_j I)$, $z \leftarrow z/\|z\|$, $j = 1, \cdots, k_i$, starting with $z = e_r^T$. It has been however observed that the accuracy of the shift vector y computed in this way 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 above recurrence, we can use an alternative scheme recently proposed in [20] to compute first the coefficients of the polynomial

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

and then to evaluate the vector y as $y = e_r^T p(A_i)$. This computation can be done efficiently by using a Horner-like polynomial evaluation scheme as for instance by evaluating recursively $y \leftarrow c_{k_i - i + 1}e_r^T + yA_i$ for $i = 1, \ldots, k_i + 1$ started with $y = 0$. These operations are column oriented and can exploit efficiently the Hessenberg structure of A_i. Both techniques to evaluate y are used in practical implicit multishift QR-like techniques for eigenvalue computations.

For $k_i = n-2$ both of above techniques lead to an algorithm with certain resemblance to the unstable method of Datta [9]. To enhance, especially for large values of k_i, the numerical accuracy in the computation of the shift information, we propose a second approach by which an orthogonal matrix $\widetilde{P}_1$, holding the same information as the orthogonal reflector P_1, is implicitly determined without performing any explicit matrix-vector multiplication. Assume that all eigenvalues $\lambda_1, \ldots, \lambda_{k_i}$ to be simultaneously assigned are real and denote $A_i^{(0)} = A_i$. Let $\mathcal{H}_1$ be the Householder reflector which annihilates the first $r-1$ elements of the last row of $A_i^{(0)} - \lambda_1 I$ and let

$$R^{(1)} = (A_i^{(0)} - \lambda_1 I)\mathcal{H}_1 := \begin{bmatrix} R_1 & r_1 \\ 0 & \rho_1 \end{bmatrix}.$$

It is easy to observe that

$$\begin{aligned} y &= e_r^T(A_i^{(0)} - \lambda_1 I)(A_i^{(0)} - \lambda_2 I)\cdots(A_i^{(0)} - \lambda_{k_i} I) \\ &= e_r^T R^{(1)}(A_i^{(1)} - \lambda_2 I)\cdots(A_i^{(1)} - \lambda_{k_i} I)\mathcal{H}_1 \\ &= \rho_1 e_r^T(A_i^{(1)} - \lambda_2 I)\cdots(A_i^{(1)} - \lambda_{k_i} I)\mathcal{H}_1 \end{aligned}$$

where

$$A_i^{(1)} = \mathcal{H}_1 A_i \mathcal{H}_1.$$

Notice that because $A_i^{(0)} - \lambda_1 I$ is an unreduced upper Hessenberg matrix of order r, the first $r-3$ columns of the resulting $A_i^{(1)}$ are also in upper Hessenberg form.

Next we compute $\mathcal{H}_2$ which annihilates the first $r-1$ elements of the last row of $A_i^{(1)} - \lambda_2 I$ and let

$$R^{(2)} = (A_i^{(1)} - \lambda_2 I)\mathcal{H}_2 := \begin{bmatrix} R_2 & r_2 \\ 0 & \rho_2 \end{bmatrix}.$$

We evaluate further y as

$$\begin{aligned} y &= \rho_1 e_r^T(A_i^{(1)} - \lambda_2 I)\cdots(A_i^{(1)} - \lambda_{k_i} I) \\ &= \rho_1 e_r^T R^{(2)}(A_i^{(2)} - \lambda_3 I)\cdots(A_i^{(2)} - \lambda_{k_i} I)\mathcal{H}_2\mathcal{H}_1 \\ &= \rho_1\rho_2 e_r^T(A_i^{(2)} - \lambda_3 I)\cdots(A_i^{(2)} - \lambda_{k_i} I)\mathcal{H}_2\mathcal{H}_1 \end{aligned}$$

where

$$A_i^{(2)} = \mathcal{H}_2\mathcal{H}_1 A_i \mathcal{H}_1\mathcal{H}_2.$$

Notice that the first $r-4$ columns of the resulting $A_i^{(2)}$ are in upper Hessenberg form.

After performing k_i similar steps we have

$$y = \rho_1\rho_2\cdots\rho_{k_i} e_r^T \mathcal{H}_{k_i}\cdots\mathcal{H}_2\mathcal{H}_1$$

TABLE I
COMPARISON OF DIFFERENT METHODS

Methods	N_{op}	M_{loc}	Numerical Stability
Miminis & Paige [1]	$\frac{2}{3}n^3+\frac{5}{6}n^3$	$\frac{3}{2}n^2$	yes
Petkov *et al.* [2]	$\frac{2}{3}n^3+\frac{5}{3}n^3$	n^2	yes
Patel & Misra [4]	$\frac{2}{3}n^3+\frac{5}{6}n^3$	$\frac{3}{2}n^2$	yes
Datta [9]	$\frac{2}{3}n^3+\frac{1}{6}n^3$	$\frac{1}{2}n^2$	no
Modified Datta [8]	$\frac{2}{3}n^3+\frac{5}{6}n^3$	$\frac{3}{2}n^2$	yes
Schur method [10]	$14n^3$	n^2	likely
Algorithm 1 ($k \ll n$)	$\frac{2}{3}n^3+\frac{2k+1}{3k}n^3$	$\frac{1}{2}n^2+\frac{k+1}{2k}n^2$	likely
Algorithm 1'($k \ll n$)	$\frac{2}{3}n^3+\frac{2k+1}{3k}n^3$	$\frac{1}{2}(n^2+k^2)+\frac{k+1}{2k}n^2$	very likely

TABLE II
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 ($k=5$)	$2.9\ {\cdot}10^{-15}$	$1.1{\cdot}10^{-14}$	$1.3{\cdot}10^{-14}$	$9.1{\cdot}10^{-14}$	$8.9{\cdot}10^{-13}$

TABLE III
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 ($k=5$)	$2.6{\cdot}10^{-15}$	$6.5{\cdot}10^{-15}$	$8.2{\cdot}10^{-15}$	$3.6{\cdot}10^{-13}$	$3.7{\cdot}10^{-11}$

and thus

$$\widetilde{P}_1 = \mathcal{H}_1\mathcal{H}_2\cdots\mathcal{H}_{k_i}$$

annihilates the first $r-1$ components of y. The final matrix

$$A_i^{(k_i)} = \mathcal{H}_{k_i}\cdots\mathcal{H}_2\mathcal{H}_1 A_i \mathcal{H}_1\mathcal{H}_2\cdots\mathcal{H}_{k_i} = \widetilde{P}_1^T A_i \widetilde{P}_1$$

has exactly the same structure as the matrix A_i resulted at Step 4) of Algorithm 1.

The above technique can be also used in the case when not all eigenvalues to be assigned are real, by working with unitary Householder reflectors. We can avoid operations with complex numbers by combining (similarly as in the case of QR method) two complex steps in a single double-step. If $\{\lambda_j, \lambda_{j+1}\}$ is a pair of complex conjugated eigenvalues to be assigned, then we determine the Householder reflector $\mathcal{H}_j$ which annihilates the first $r-1$ elements of the last row of the real matrix $(A_i^{(j-1)} - \lambda_j I)(A_i^{(j-1)} - \lambda_{j+1} I)$ and apply it as in the case of a single real eigenvalue.

It can be seen that if we successively apply the computed transformations $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_{k_i}$ to A_i, then the explicit computation of $\widetilde{P}_1$ is no more necessary. Thus a numerically enhanced version of Algorithm 1 can be devised which determines the shift information without any explicit matrix-vector multiplications. This algorithmic variant is numerically more robust than Algorithm 1 and should be preferred for the implementation of the multishift technique. We give bellow only those steps of the enhanced algorithm which differ from the original version.

Algorithm 1'. *Enhanced multi-step variable-multishift single-input pole assignment.*

...

3) Choose a symmetric set of $k_i < r-2$ eigenvalues $\Gamma_i = \{\lambda_1, \lambda_2, \ldots, \lambda_{k_i}\} \subset \Gamma$, where complex conjugated pairs appear in consecutive positions.
4) For $j = 1, \ldots, k_i$
   a) Compute $y = e_r^T(A_i - \lambda_j I)$ for λ_j real or $y = e_r^T(A_i - \lambda_j I)(A_i - \bar{\lambda}_j I)$ for λ_j complex. Choose a Householder reflector $\mathcal{H}_j$ to annihilate the first $r-1$ components of y.
   b) Compute $A_i \leftarrow \mathcal{H}_j A_i \mathcal{H}_j$, $P \leftarrow P\text{diag}(I_{n-r}, \mathcal{H}_j)$. If λ_j is complex then $j \leftarrow j+1$.

...

To compute the shift information the enhanced algorithm performs slightly more operations then Algorithm 1 (about $(2/3)k^2n + (1/2)kn^2$ *flops* instead $(1/6)k^2n$) and needs some additional storage (about $(1/2)k^2$ locations) to store the Householder transformations $\mathcal{H}_j$, $j = 1, \ldots, k$. For $k \ll n$ the in-

creases in the operation number as well in the storage requirements are negligible.

IV. Comparisons with Other Methods

It is interesting to compare the performances of the proposed methods with those of existing ones. Table 1 contains the total number of operations N_{op} and the necessary storage M_{loc} for different methods, including for Hessenberg methods 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.

We performed several tests to assess the accuracy of proposed methods. As 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 [6], [2], [4]. Two other methods were included in the test runs, namely the method of Miminis and Paige [1] and the Schur method [12]. 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 chosed as desired eigenvalues the eigenvalues of A. The resulting feedback f must 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 all methods manifest approximately the same accuracy. Notice however that for this kind of test, the Schur method performed systematically slightly better than the Hessenberg methods.

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. This test also confirms that the accuracy of all three methods is very similar. The best accuracy for the assigned eigenvalues has been achieved this time systematically with the two Hessenberg methods. A word of caution is however necessary here. The results of such a 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 even low order examples leading to very ill-conditioned eigenvalue problems for $A + bf$.

V. 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 similar with or better than the performances of existing numerically stable methods. The proposed multishift method 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 [17]. The multishift technique can be readily extended to assign poles of multi-input systems as well as of generalized state space systems.

References

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

[2] 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*, vol. AC-29, pp. 1045–1048, 1984.

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

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

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

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

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

[8] M. Arnold, *Algorithms and Conditioning for Eigenvalue Assignment*, Ph.D. thesis, Northern Illinois University, Dekalb, Illinois, 1992.

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

[10] A. Varga, "A Schur method for pole assignment," *IEEE Trans. Autom. Control*, vol. AC-26, pp. 517–519, 1981.

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

[12] A. Varga, "Numerically stable algorithm for standard controllability form determination," *Electron. Lett.*, vol. 17, pp. 74–75, 1981.

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

[14] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. Hanson, "An extended set of Fortran basic linear algebra subprograms," *ACM Trans. Math. Software*, vol. 14, pp. 1–17, 1988.

[15] C. Bischof and C. Van Loan, "The WY representation for products of Householder matrices," *SIAM J. Sci. Stat. Comput.*, vol. 8, pp. s2–s13, 1987.

[16] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff, "A set of level 3 basic linear algebra subprograms," *ACM Trans. Math. Software*, vol. 16, pp. 1–17, 1990.

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

[18] A. Varga, "Multishift algorithm for pole assignment of single-input systems," in *Proc. 1995 European Control Conference, Rome, Italy*, 1995, vol. 4, pp. 3348–3352.

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

[20] 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.