Using Semi-implicit Iterations in the Periodic QZ Algorithm

Vasile Sima[1,2,*] [a] and Pascal Gahinet[3] [b]

[1] *Modelling, Simulation, Optimization Department, National Institute for Research & Development in Informatics, Bd. Mareşal Averescu, Nr. 8–10, Bucharest, Romania*

[2] *Technical Sciences Academy of Romania, Romania*

[3] *MathWorks, 3 Apple Hill Drive, Natick, MA, U.S.A.*

Keywords: Eigenvalue Problem, Hamiltonian Matrix, Numerical Methods, Optimal Control, Periodic Systems.

Abstract: The periodic QZ (pQZ) algorithm is the key solver in many applications, including periodic systems, cyclic matrices and matrix pencils, and solution of skew-Hamiltonian/Hamiltonian eigenvalue problems, which, in turn, is basic in optimal and robust control, and characterization of dynamical systems. This algorithm operates on a formal product of matrices. For numerical reasons, the standard pQZ algorithm uses an implicit approach during the iterative process. The shifts needed to increase the convergence rate are implicitly defined and applied via an embedding, which essentially allows to reduce the processing to transformations of the data by Givens rotations. But the implicit approach may not converge for some periodic eigenvalue problems. A new, semi-implicit approach is proposed to avoid convergence failures and reduce the number of iterations. This approach uses shifts computed explicitly, but without evaluating the matrix product. The shifts are applied via a suitable embedding. The combination of the implicit and semi-implicit schemes proved beneficial for improving the behavior of the pQZ algorithm. The numerical results for several extensive tests have shown no convergence failures and a reduced number of iterations.

1 INTRODUCTION

The *periodic QZ algorithm* (pQZ algorithm), also called *periodic QR algorithm* (Van Loan, 1975; Bojanczyk et al., 1992; Sreedhar and Van Dooren, 1994) is used in many applications, such as, periodic linear systems, k-cyclic matrices and pencils encountered in the investigation of Markov chains and the solution of two-point boundary value problems (Bojanczyk et al., 1992). This algorithm is also the key solver for structured, *skew-Hamiltonian/Hamiltonian* (sHH) eigenvalue problems, see (Benner et al., 2002; Benner et al., 2007; Kressner, 2005; Benner et al., 2016) and the references therein. The sHH solver is core to the calculation of the H_∞-norm, based on (Bruinsma and Steinbuch, 1990), and to nonsmooth minimization of the $\mathcal{L}_\infty$-norm, which is central to the fixed-order controller tuning—systune—(Apkarian et al., 2014). It is applied in (Xia et al., 2017) for computing the R-index of quadratic sector bounds, which offer a characterization of, for instance, dynamical systems behavior, including passivity, dissipativity, and input/output gain. The sHH solver also comes into play in the new "safe" approach in robust control for finding μ upper bounds for real uncertainty. The use of this solver for the $\mathcal{L}_\infty$-norms computation and linear-quadratic and $\mathcal{H}_\infty$ optimization is presented in (Benner et al., 2012a; Benner et al., 2012b; Benner et al., 2013; Benner et al., 2016).

The pQZ and sHH algorithms have been implemented in the SLICOT Library (Benner et al., 1999) and have been also included in the releases R2012a or newer of MATLAB® (MathWorks®, 2012). Despite the extensive use of the sHH solver, some examples have been recently found for which the standard pQZ algorithm, which uses an *implicit* approach during the iterative process, either fails or requires too many iterations to converge (Sima and Gahinet, 2019). The cause of this undesirable behavior has been investigated and a solution has been proposed in (Sima and Gahinet, 2019) to avoid failures and improve the convergence speed. Essentially, if the implicit scheme does not converge in a certain fixed number of iterations for a subproblem, and the subproblem order, q, is small (e.g., $q \leq 6$), an *explicit* scheme is used

[a] https://orcid.org/0000-0003-1445-345X

[b] https://orcid.org/0000-0002-9485-5127

*Corresponding Member of the Technical Sciences Academy of Romania

Sima, V. and Gahinet, P.
Using Semi-implicit Iterations in the Periodic QZ Algorithm.
DOI: 10.5220/0009823300350046
In *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2020)*, pages 35-46
ISBN: 978-989-758-442-8

for at most ten consecutive iterations (for the same subproblem). The eigenvalues of the subproblem, λ_j, $j = 1:q$, are computed, and the shifts σ_i, $i = 1:2$, are chosen as the eigenvalues with maximum moduli, but disregarding all real eigenvalues, if there are complex conjugate ones. This combination of implicit and explicit approaches may improve the convergence of the algorithm. Indeed, no failures and less iterations have been reported in (Sima and Gahinet, 2019).

However, the explicit approach has a main drawback: the selected shifts will be inaccurate if q and k—the number of factors of the formal matrix product—are large, and/or some factors are ill-conditioned.

A new, *semi-implicit* approach is proposed in this paper. In this approach, after a certain number of unsuccessful iterations with the implicit scheme, explicitly computed shifts are used via a suitable embedding. Briefly speaking, the eigenvalues of the 2×2 trailing principal submatrix, F, of the current subproblem are found using a specialized pQZ algorithm for 2×2 problems. This variant is actually used in the general pQZ algorithm for the case when $q = 2$. It uses a very reliable and quite efficient single shifted complex pQZ algorithm, to be able to deal with complex eigenvalues. Therefore, F is not evaluated, and the eigenvalues λ_i, $i = 1:2$, are accurate. The shifts to be used are then chosen based on these eigenvalues. Specifically, if λ_i are complex conjugate, then $\sigma_i = \lambda_i$, $i = 1:2$. The same can be used if λ_i are real, but other definitions may be adopted, for instance, $\sigma_i = \lambda_*$, with $* \in \{m, M, c\}$, where m, M, and c refer to the real eigenvalue with minimum modulus, maximum modulus, or closer to the (q,q) element of the subproblem. In any case, there are several degrees of freedom, in comparison with the *purely* implicit approach, where the shifts are even not known.

Often, few iterations of the semi-implicit approach succeed to deflate the trailing 1×1 or 2×2 subproblem. This means that the eigenvalues of this subproblem can be taken as eigenvalues of the original problem, since all elements on the left of (and below) the corresponding submatrices are zero or negligible. Anyway, after some iterations other subdiagonal elements will become negligible and the subproblem can be split into smaller subproblems.

The paper is organized as follows. The next section introduces the essential matter related to the pQZ algorithm. Section 3 presents the semi-implicit approach, and summarizes the numerical results obtained using several possible options. No individual option provides the best results for all problems tried. Section 4 proposes to combine the implicit and semi-implicit approaches in order to get near optimal performance for all problems. More numerical results are presented, which confirm the improved behavior of the pQZ algorithm for several large sets of problems. No failures have been recorded and the number of iterations has been reduced compared to the previous versions of the pQZ solver.

2 ESSENTIALS ON THE PQZ ALGORITHM

The pQZ algorithm generalizes the QZ algorithm, dealt with, for instance, in (Golub and Van Loan, 2013). The main computational parts are the same: reduction to an upper Hessenberg-triangular form,[1] deflation, computation of the shifts, and the QZ step. The real case only is considered in this paper. The complex case is actually simpler. The pQZ algorithm operates on a *formal matrix product*,

$$P := A_1^{s_1} A_2^{s_2} \cdots A_k^{s_k}, \tag{1}$$

with $A_i \in \mathbf{R}^{n \times n}$, and $s_i = \pm 1$, $i = 1:k$. Any such product can be reduced using orthogonal transformations to a *similar* one (i.e., with the same eigenvalues) in which one factor is upper Hessenberg and all the other factors are upper triangular. This is the *Hessenberg-triangular form*. Therefore, without loss of generality, it is assumed that the factors are already in this form, and the factor with index h is upper Hessenberg. There is no restriction on the factor exponents, called *signatures*, stored in an integer vector s. A negative exponent s_i is associated with the "inverse" of the factor A_i. If such a factor is singular, P has one or more infinite eigenvalues. If $s_h = 1$, the product matrix P is also upper Hessenberg, but if $s_h = -1$, P might be a full matrix. In order to exploit the structure of P also for $s_h = -1$, the eigenvalues of P are computed as the reciprocals of the eigenvalues of P^{-1}, i.e., $\lambda_i(P) = 1/\lambda_i(P^{-1})$, $i = 1:n$. To simplify the implementation, an integer map p is defined so that $p_1 = h$, for all possible values of the signatures. The values p_j, $j \neq h$, are set so that to obtain equivalent eigenvalue problems. For instance, if $s_h = 1$, then

$$p_i = \mod(h+i-2,k)+1, \quad i = 1:k.$$

Then, $p_1 = h$, $p_2 = \mod(h,k)+1, \ldots, p_k = \mod(h-2,k)+1$. Therefore, the *spectrum* of P, i.e., its set of eigenvalues, can be written as

$$\Lambda(P) = \Lambda\left(A_{p_1} A_{p_2}^{s_{p_2}} \cdots A_{p_k}^{s_{p_k}}\right). \tag{2}$$

[1] A matrix M of order n is upper triangular if $m_{ij} = 0$ for $i = j+1:n$, $j = 1:n-1$, and it is upper Hessenberg if $m_{ij} = 0$ for $i = j+2:n$, $j = 1:n-2$; a MATLAB-style notation is used for index ranges.

If, for instance, $h = 3$ and $k = 5$, then

$$\Lambda(P) = \Lambda\left(A_3 A_4^{s_4} A_5^{s_5} A_1^{s_1} A_2^{s_2}\right).$$

For $h = 1$, $p_i = i$, $i = 1:k$.

If $s_h = -1$, then, defining $\bar{s}_i = -s_i$, then

$$P^{-1} = A_k^{\bar{s}_k} A_{k-1}^{\bar{s}_{k-1}} \cdots A_h A_{h-1}^{\bar{s}_{h-1}} \cdots A_2^{\bar{s}_2} A_1^{\bar{s}_1} \tag{3}$$

and

$$\Lambda\left(P^{-1}\right) = \Lambda\left(A_{p_1} A_{p_2}^{\bar{s}_{p_2}} \cdots A_{p_k}^{\bar{s}_{p_k}}\right), \tag{4}$$

which has the same form as (2), and where $p_1 = h$, $p_2 = h-1, \ldots, p_h = 1$, $p_{h+1} = k$, $p_{h+2} = k-1, \ldots, p_{k-1} = h+2$, $p_k = h+1$. For instance, if $h = 3$, $k = 5$, and $s_3 = -1$, then

$$\Lambda\left(P^{-1}\right) = \Lambda\left(A_3 A_2^{\bar{s}_2} A_1^{\bar{s}_1} A_5^{\bar{s}_5} A_4^{\bar{s}_4}\right).$$

The pQZ algorithm is an iterative process aiming to reveal the eigenvalues of P. The process preserves the Hessenberg-triangular form of the factors at each iteration, and ultimately it reduces the Hessenberg matrix to a *real Schur form*, that is, a block upper triangular matrix with 1×1 and 2×2 diagonal blocks corresponding to real and complex conjugate eigenvalues, respectively. All transformations applied in the algorithm can be defined by

$$\widetilde{A}_i^{s_i} := Q_i^T A_i^{s_i} Q_{i \oplus 1}, \tag{5}$$

where Q_i, $i = 1:k$, are orthogonal matrices, $i \oplus 1 := \mathrm{mod}(i,k)+1$, and the superscript T denotes the transposition. The matrices Q_i are chosen to preserve the Hessenberg-triangular form of the factors. Clearly, using (5) it follows that the product of transformed matrices, $\widetilde{P}$, is similar to P, since $\widetilde{P} = Q_1^T P Q_1$.

The algorithm uses a sequence of *periodic QZ steps* involving transformations defined in (5). After a number of such steps, one or more subdiagonal elements of the Hessenberg factor, and possibly some diagonal elements of the triangular factors, become negligible in comparison to the neighboring diagonal and super-diagonal elements, respectively (in case of a "cautious" strategy) or to the Frobenius norm of the corresponding factor (in case of an "aggressive" strategy). These negligible elements are set to zero, and the eigenvalue problem is decomposed into smaller subproblems, which are solved separately, starting from the bottom part. Zeros on the diagonal of a triangular factor are dealt with a special procedure to obtain zero or infinite eigenvalues. Therefore, the deflated subproblems are nonsingular. When subproblems of order 1 or 2 are deflated at the current bottom part, their eigenvalues are computed directly or using a specialized pQZ algorithm, respectively, and the order of the eigenvalue problem reduces by 1 or 2. Let $a_{pq}^{(i)}$ denote the (p,q) element of A_i. Assume that the last $n-l$, $l \le n$, eigenvalues have been found, and let $\underline{A}_i$ denote the submatrix formed by the rows and columns $j:l$ of A_i, with j, $1 \le j \le l \le n$, being the largest index so that

$$a_{j,j-1}^{(h)} = 0, \text{ if } j > 1,\ a_{r+1,r}^{(h)} \neq 0,\ r = j:l-1,$$
$$a_{l+1,l}^{(h)} = 0, \text{ if } l < n.$$

The submatrices $\underline{A}_i$, $i = 1:k$, define the *current (eigenvalue) subproblem*. Initially, $j = 1$ and $l = n$, hence $\underline{A}_i = A_i$, $i = 1:k$.

As in any QR-like algorithm, the convergence rate of the pQZ algorithm can be improved using suitable *shifts*, which are increasingly more accurate approximations of some eigenvalues of the matrix product. Since real matrices may have complex conjugate eigenvalues, two shifts are used simultaneously to keep the arithmetic real. The *double-shift Wilkinson polynomial*, with shifts σ_1 and σ_2, for the subproblem defined by the indices j and l, is given by

$$P_\sigma := \left(\underline{P} - \sigma_1 I_q\right)\left(\underline{P} - \sigma_2 I_q\right), \tag{6}$$

where $\underline{P}$ is obtained from (1) with A_i replaced by $\underline{A}_i$, $q = l - j + 1$, and I_q denotes the identity matrix of order q. Usually, σ_1 and σ_2 are the eigenvalues of the 2×2 trailing principal submatrix, F, of the matrix product $\underline{P}$. A simplified form is the *single shift polynomial* $P_\sigma = \underline{P} - \sigma_1 I_q$. For numerical reasons, it is essential that $\underline{P}$, σ_1, σ_2, and P_σ are not computed explicitly but implicitly in the pQZ algorithm.

Each periodic QZ step starts by an initial transformation that "incorporates" the shifts. Actually, an orthogonal matrix Q_0 is built such that the first column of P_σ in (6) is reduced to a multiple ce_1 of $e_1 = [1, 0, \cdots, 0]^T \in \mathbf{R}^q$, $c \in \mathbf{R}$. Since $\underline{P}$ is upper Hessenberg, the only nonzeros in the first column of P_σ can be in the locations 1:3. Therefore, this column is reduced to ce_1 using two Givens rotations, G_1 and G_2, defined by the cosines c_i and sines s_i, $i = 1:2$, respectively, so that $Q_0 = \mathrm{diag}\left(\mathcal{G}, I_{q-3}\right)$, with

$$\mathcal{G} := \begin{bmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_2 & s_2 \\ 0 & -s_2 & c_2 \end{bmatrix}. \tag{7}$$

When Q_0 is applied to $\underline{P}$ (actually to $\underline{A}_1$) from the left, additional nonzeros appear in the locations (3,1), (4,1), and (4,2). The pQZ step then annihilates these nonzeros using suitably chosen matrices Q_i, $i = 1:k$, to restore the Hessenberg-triangular form. A well-known theorem (Golub and Van Loan, 2013) shows that if $Q_o \underline{P}$ is reduced to the Hessenberg form, the transformation is equivalent to the application of two pQZ steps with the shifts σ_1 and σ_2. For the single shift polynomial, only the rotation G_1 is needed.

The first version of the pQZ solver used the implicit embedding defined in (Kressner, 2001),

$$P_\sigma = \begin{bmatrix} \underline{A}_1 & I_q \end{bmatrix} \prod_{i=2}^{k} \begin{bmatrix} \underline{A}_i & 0 \\ 0 & a_{mm}^{(i)} I_q \end{bmatrix}^{s_i} \cdot \begin{bmatrix} -I_q & 0 \\ a_{mm}^{(1)} I_q & -a_{lm}^{(1)} I_q \end{bmatrix} \begin{bmatrix} -\underline{A}_1 & a_{lm}^{(1)} I_q & a_{ll}^{(1)} I_q \\ 0 & a_{mm}^{(1)} I_q & a_{ml}^{(1)} I_q \end{bmatrix} \cdot \prod_{i=2}^{k} \begin{bmatrix} \underline{A}_i & 0 & 0 \\ 0 & a_{mm}^{(i)} I_q & a_{ml}^{(i)} I_q \\ 0 & 0 & a_{ll}^{(i)} I_q \end{bmatrix}^{s_i} \begin{bmatrix} I_q \\ 0 \\ I_q \end{bmatrix}, \quad (8)$$

where $m := l-1$. This embedding corresponds to the assumption that $h = 1$, i.e., the Hessenberg matrix is the first factor of P in (1). But the shifts, implicitly defined by the eigenvalues of the submatrix

$$F := \begin{bmatrix} a_{mm}^{(1)} & a_{ml}^{(1)} \\ a_{lm}^{(1)} & a_{ll}^{(1)} \end{bmatrix} \prod_{i=2}^{k} \begin{bmatrix} a_{mm}^{(i)} & a_{ml}^{(i)} \\ 0 & a_{ll}^{(i)} \end{bmatrix}^{s_i}, \quad (9)$$

are not correct, since F differs from the trailing 2×2 submatrix of the product $\underline{A}_1 \underline{A}_2^{s_2} \cdots \underline{A}_k^{s_k}$ if $j \neq l-1$. Another implicit embedding has been proposed in (Sima and Gahinet, 2019),

$$P_\sigma = \begin{bmatrix} -I_q & I_q & 0 \end{bmatrix} \prod_{i=2}^{k} \begin{bmatrix} \underline{A}_i & 0 & 0 \\ 0 & a_{mm}^{(i)} I_q & a_{ml}^{(i)} I_q \\ 0 & 0 & a_{ll}^{(i)} I_q \end{bmatrix}^{s_i} \cdot \begin{bmatrix} \underline{A}_1 & 0 \\ a_{mm}^{(1)} I_q & a_{ml}^{(1)} I_q \\ a_{lm}^{(1)} I_q & a_{ll}^{(1)} I_q \end{bmatrix} \begin{bmatrix} -I_q & a_{ll}^{(1)} I_q \\ 0 & -a_{lm}^{(1)} I_q \end{bmatrix} \cdot \prod_{i=2}^{k} \begin{bmatrix} \underline{A}_i & 0 \\ 0 & a_{ll}^{(i)} I_q \end{bmatrix}^{s_i} \begin{bmatrix} \underline{A}_1 \\ I_q \end{bmatrix}, \quad (10)$$

which works for products where the Hessenberg matrix is the last factor. Then, clearly, the corresponding submatrix F is correct. Details on how the intermediate rotations are computed and applied in order to preserve the Hessenberg-triangular structure and to reduce the first column of the double-shift Wilkinson polynomial P_σ in (8) and (10) are given in (Sima, 2019). Essentially, partial QR factorizations of the first column of each factor of P_σ are found, starting from the last factor, and continuing step by step till the first factor.

3 SEMI-IMPLICIT APPROACH FOR THE pQZ ALGORITHM

The implicit embeddings (8) and (10) are appealing because they only involve the available data. The product $\underline{P}$, the shifts σ_1 and σ_2, and the polynomial P_σ are not evaluated, which is very important for numerical reasons. However, these embeddings have a main drawback: the eigenvalues implicitly used as shifts can be far from any eigenvalues of the product corresponding to the current subproblem. This sometimes causes convergence failures (Sima and Gahinet, 2019), which is very undesirable in many applications. Such failures may happen even for standard QR- and QZ-like algorithms, and to avoid them, exceptional shifts or transformations are used, e.g., after ten unsuccessful consecutive iterations for the same subproblem.

3.1 Purely Semi-implicit Approach

The *semi-implicit approach* has been devised with the intent to avoid as much as possible the convergence failures, which have been encountered for the implicit approach. The semi-implicit scheme differs from the explicit scheme proposed in (Sima and Gahinet, 2019) mainly in the fact that the shifts are determined based on the eigenvalues of F, but computed without evaluating the product of the factors. Moreover, the shifts are not used to build the corresponding Wilkinson polynomial, but an embedding-based implicit algorithm is employed. The embedding depends on the nature of the shifts σ_i, $i = 1:2$. If $\sigma_1 = \alpha$ and $\sigma_2 = \beta$ are real, then $P_\sigma = \left(\underline{P} - \alpha I_q\right)\left(\underline{P} - \beta I_q\right)$, with $\underline{P} = \underline{A}_2^{s_2} \cdots \underline{A}_k^{s_k} \underline{A}_1$, has an embedding defined by

$$P_\sigma = \begin{bmatrix} I_q & -\alpha I_q \end{bmatrix} \prod_{i=2}^{k} \begin{bmatrix} \underline{A}_i^{s_i} & 0 \\ 0 & I_q \end{bmatrix} \begin{bmatrix} \underline{A}_1 \\ I_q \end{bmatrix} \cdot \begin{bmatrix} I_q & -\beta I_q \end{bmatrix} \prod_{i=2}^{k} \begin{bmatrix} \underline{A}_i^{s_i} & 0 \\ 0 & I_q \end{bmatrix} \begin{bmatrix} \underline{A}_1 \\ I_q \end{bmatrix}. \quad (11)$$

If $\sigma_1 = \alpha + \imath\beta$ and $\sigma_2 = \alpha - \imath\beta$ are complex conjugate, then

$$P_\sigma = \begin{bmatrix} \underline{P} - \alpha I_q & \beta I_q \end{bmatrix} \begin{bmatrix} \underline{P} - \alpha I_q \\ \beta I_q \end{bmatrix}$$

has an embedding defined by

$$P_\sigma := \begin{bmatrix} I_q & -\alpha I_q & \beta I_q \end{bmatrix} \cdot \prod_{i=2}^{k} \begin{bmatrix} \underline{A}_i^{s_i} & 0 & 0 \\ 0 & I_q & 0 \\ 0 & 0 & I_q \end{bmatrix} \begin{bmatrix} \underline{A}_1 & 0 \\ I_q & 0 \\ 0 & I_q \end{bmatrix} \cdot \begin{bmatrix} I_q & -\alpha I_q \\ 0 & \beta I_q \end{bmatrix} \prod_{i=2}^{k} \begin{bmatrix} \underline{A}_i^{s_i} & 0 \\ 0 & I_q \end{bmatrix} \begin{bmatrix} \underline{A}_1 \\ I_q \end{bmatrix}. \quad (12)$$

These embeddings are simplified versions of (10) and retain a similar structure. Therefore they could work equally well for computing the Givens rotations that

will map $P_\sigma e_1$ to a multiple of e_1. Fewer rotations are needed for processing (11) or (12) than for (10).

The major difference between the implicit and semi-implicit approaches is in their degrees of freedom. The implicit approach offers no degree of freedom, since the shifts are not even known. On the other hand, the semi-implicit approach allows to make different selections of the shifts, possibly taking additional information into account. Indeed, if, for instance, the computed eigenvalues of F are both real, it is possible to use them as shifts, or use two identical shifts set to one of those eigenvalues, or even use only one of them. Sometimes, when the true eigenvalues, λ_{l-1} and λ_l, are complex conjugate, but the current eigenvalues of F are real, then a selection $\sigma_1 = \sigma_2$ could provide a better approximation of λ_{l-1} and λ_l if their imaginary parts have small magnitude compared to the real parts. The low-level routine implementing the basic computations of the semi-implicit approach has an input argument, SHFT, for specifying the number and type of the shifts employed by the polynomial P_σ, namely:

SHFT = 'C': two complex conjugate shifts;

SHFT = 'D': two real identical shifts;

SHFT = 'R': two real shifts;

SHFT = 'S': one real shift.

When the eigenvalues are complex conjugate, this argument must be set to 'C'. These options offer full flexibility and allow to perform various tests, but no such option provides the best results for all these tests, as can be seen in the next subsection.

The eigenvalues of the trailing principal submatrix F are computed in the pQZ solver, and the needed information about these eigenvalues reduces to SHFT and two real values, w_1 and w_2. The value w_1 is the real part of the first eigenvalue. The value w_2 is the second eigenvalue, if both eigenvalues are real; otherwise, w_2 is the imaginary part (positive or negative) of the complex conjugate pair.

The computed eigenvalues, $\lambda_1, \lambda_2 \in \Lambda(F)$, can be directly used as shifts, by setting SHFT = 'C' or SHFT = 'R'. Theoretically, this is equivalent to using the implicit scheme, but numerically, it is not. The implicit scheme does not compute and use these eigenvalues, as they do not appear in the embeddings (8) or (10). On the other hand, the new embeddings (11) and (12) resort to the eigenvalues. The specialized pQZ algorithm for 2×2 problems used to find $\Lambda(F)$ converts the real eigenvalue problem into an equivalent complex one with initial zero imaginary parts. The algorithm then performs at most 80 iterations using complex single shifts, trying to deflate the problem, if possible. Clearly, the obtained eigenvalues could sometimes differ from the shifts implicitly used by the embeddings (8) or (10), which only resort to the given data. It could even be possible that structurally different shifts, i.e., complex instead of real ones, be used by the two schemes for the same subproblem.

It is convenient to make a distinction between the purely semi-implicit and semi-implicit approaches. The *purely semi-implicit approach* computes the eigenvalues λ_1 and λ_2 via the specialized pQZ algorithm and uses the option SHFT = 'C', for complex conjugate eigenvalues, or one of the prespecified options 'R', 'D, or 'S', for two real eigenvalues. The *semi-implicit approach* computes λ_1 and λ_2 and uses a heuristic for choosing the shifts, involving two or more of the options above, for instance, 'C' or 'R' with 'D', since none of them taken alone ensure (fast) convergence for all problems. Moreover, the *combined implicit and semi-implicit* approach uses the implicit approach in the first iterations of the pQZ algorithm for each new subproblem, but switches to the semi-implicit approach if the subproblem is not deflated after a specified number of iterations.

Neither the implicit, nor the purely semi-implicit scheme alone converges for all problems. Often, convergence failures occur for subproblems of order three or four with one pair or two pairs of complex conjugate eigenvalues, respectively, if two real eigenvalues are obtained for the trailing 2×2 subproblem. See (Sima and Gahinet, 2019) for such a case. If the shifts used in many consecutive iterations are far away from any of the true eigenvalues of the subproblem, the process may converge slowly or fail. The same behavior happens with the QR or QZ algorithm without "exceptional" transformations. For instance, consider a simple problem, with $n = 4$, $k = 1$, defined by the Hessenberg matrix

$$A = \begin{bmatrix} 1085 & 17 & 0 & -2 \\ 15 & 876 & 0 & 0 \\ 0 & 2 & 1077 & 44 \\ 0 & 0 & 43 & 884 \end{bmatrix},$$

whose eigenvalues are $1086.2818649127 \pm 0.2300965203294685\imath$, and $874.7181350873004 \pm 0.2300965203293698\imath$. The eigenvalues of $F = A(3:4,3:4)$ are $\lambda_1^0 = 1086.350129900723$ and $\lambda_2^0 = 874.6498700992767$, which are close to the real parts of the two pairs of complex eigenvalues. Using a simple QR algorithm with explicit shifts, shown below in MATLAB, which updates the matrix A at each iteration, the convergence does not occur in `maxit` $= 480$ iterations.

```
n = size( A, 1 );  maxit = 120*n;
for ii = 1 : maxit
    ev2 = eig( A(n-1:n,n-1:n) );
    if isreal( ev2 ),
```

```
        sm = sum( ev2 );  pr = prod( ev2 );
    else
        sm = 2*real( ev2(1) );
        pr = abs( ev2(1) )^2;
    end
    [ Q, ~ ] = qr( A^2 - sm*A + pr*eye( n ) );
    A = Q'*A*Q;
    [ ~, l ] = min( abs( A(2:n+1:n*n) ) );
    if abs( A(l+1,l) ) < eps*( abs( A(l,l) ) +
                           abs( A(l+1,l+1) ) )
        break
    end
end
```

The same happens if the shifts are set to λ_1^0 and λ_2^0. However, using a double shift set to the eigenvalue with either the maximum or minimum magnitude in `ev2`, the convergence takes place after three iterations. If a single shift set to the current value a_{44} is used, then seven iterations are needed.

A very simple example for which the sequence above does not converge is the matrix

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

For this example, `ev2` is zero. At the odd iterations the updated matrix A has the same structure as A, but the elements a_{13} and a_{32} have values -1. At the even iterations, the matrix coincides to the original A. Therefore, the shifts are zero at each iteration and the matrix cannot be deflated. Exceptional shifts are needed to deflate it.

This subsection is concluded by a summary of the similar, LAPACK approach (Anderson et al., 1999). The LAPACK basic eigensolver routines, DLAHQR for the QR algorithm (applied to a Hessenberg matrix A), or DHGEQZ for the QZ algorithm (applied to a Hessenberg-triangular pair (A,B)), use refined procedures to find the eigenvalues of the trailing 2×2 part. If the eigenvalues are complex conjugate for subproblems of order at least three, they are used as shifts. For subproblems of order two (i.e., with $j = l-1$), the 2×2 blocks are standardized (so that, e.g., for QZ, the corresponding submatrix in B is diagonal with the first element non-negative); in DHGEQZ, the eigenvalues are then recomputed and if they have accidentally moved onto the real line, a real single shift is applied. If the eigenvalues are real, a double shift, taken as the eigenvalue closest to a_{ll} or to a_{ll}/b_{ll}, is used by the QR or QZ algorithm, respectively. The shifts are applied in an implicit manner. The QR algorithm uses exceptional shifts at the iterations 10 and 20 for each subproblem. The newer versions (from LAPACK 3.2 on) use exceptional shifts computed based on the leading or trailing 2×2 principal submatrix of the subproblem, at iteration 10 or iteration 20, respectively. The QZ algorithm uses single exceptional shifts at each tenth iteration for a deflated subproblem. The exceptional shift is usually set to the ratio $a_{l,l-1}/b_{l-1,l-1}$, but it can be bigger than that value if $|a_{l,l-1}| \gg |b_{l-1,l-1}|$. The maximum number of iterations for the latest version of DLAHQR is $30\max(10,n)$. Older versions used the value 30 for each subproblem (e.g., in LAPACK 3.1) or $30n$ for the whole problem (e.g., in LAPACK 3.0, or in the DHGEQZ routine). Negligible subdiagonal elements in the QR algorithm are those with a magnitude less than or equal to the machine precision times the sum of the magnitudes of the neighboring elements. In such a case, a more conservative small subdiagonal deflation criterion in (Ahues and Tisseur, 1997) is used. Negligible (sub)diagonal elements in the QZ algorithm are those with a magnitude less than or equal to the machine precision times the Frobenius norm of the matrix. The LAPACK routines also check for two consecutive small subdiagonal elements in the Hessenberg submatrix (corresponding to the product AB^{-1} in DHGEQZ), and if this happens, the size of the subproblem is further reduced. For the QZ algorithm, this test is performed only if a negligible diagonal element in the triangular matrix was found. A single shift is used in this case, chosen as the eigenvalue closest to the last element of the current product. Such a test cannot be used for the pQZ algorithm since it will involve diagonal and subdiagonal elements of the product. Moreover, exceptional shifts or transformations with an effectiveness comparable to that of the standard QR and QZ algorithms are difficult to obtain for general formal matrix products (1). The previous version of the solver just used some random rotations G_1 and G_2.

3.2 Numerical Results using the Purely Semi-implicit Approach

One of the strategies tried has been to use the *purely semi-implicit approach* at each iteration of the pQZ algorithm.

The tests summarized below have been performed for various type of problems and different options. These tests are briefly described and the performance results, represented by the maximum, mean, and median number of iterations, are summarized in the associated tables. The behavior when the shifts at each iteration are chosen based on the computed eigenvalues of the trailing 2×2 principal submatrix of the current subproblem has been investigated. The best results are printed with bold characters. The notation 'C'/'R' means that the two complex conjugate or real

eigenvalues have been used as shifts, while 'S'-min or 'D'-min mean that the single/double real shift(s) has/have been set to the real eigenvalue with the minimum modulus; similarly, the suffix max refers to the real eigenvalue with the maximum modulus.

Test 1 comprises 144 matrix products in Hessenberg-triangular form, with factors generated from a pseudorandom standard uniform distribution, i.e., in the interval (0,1), using the MATLAB function `rand`, and parameters defined by $k = 1:8$, $n = 3:20$, $h = 1$, and $s_i = 1, i = 1:k$. The convergence results for various options are presented in Table 1.

Table 1: Performance results for Test 1 set using purely semi-implicit approach.

option	max	mean	median
'C'/'R'	**57**	**30.67**	31
'S'-min	198	39.96	36
'S'-max	444	33.92	**28.5**
'D'-min	433	53.42	49
'D'-max	231	39.70	37

Test 2 involves 8015 skew-Hamiltonian/Hamiltonian eigenvalue problems, with matrices generated either from a pseudorandom standard uniform distribution (4015 problems) or standard normal distribution (4000 problems), with $k = 4$, $h = 1$, and $s = \begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}$. The first 15 problems had $n = 10$, while the remaining problems had varying dimension $n = 1:20$, and 100 problems have been generated for each value of n. The performance results are shown in Table 2. Except for the 'C'/'R' option, the other options recorded a number of non-converging cases, part of them converging after a second call of the pQZ solver with the "aggressive" deflation strategy. Those numbers appear in the fifth column (with heading "no conv") to the left and right of the minus sign.

Table 2: Performance results for Test 2 set using purely semi-implicit approach.

option	max	mean	median	no conv
'C'/'R'	**61**	**21.62**	**22**	0
'S'-min	3959	213.91	31	440 − 18
'S'-max	2707	122.74	25	238 − 30
'D'-min	2530	88.13	37	114 − 8
'D'-max	2475	51.11	29	56 − 10

Test 3 includes the same 8015 eigenvalue problems described in Test 2 above, but with a different initialization of the random number generator, and 16 additional singular sHH problems with n equal to 9 (for 11 problems), 12, 3, 7, 21, and 21. The skew-Hamiltonian matrix of these 16 problems has been chosen diagonal with two distinct values, one nonzero and the other zero. The results are shown in Table 3. The option 'D'-max gave the same results as 'S'-max.

Table 3: Performance results for Test 3 set using purely semi-implicit approach.

option	max	mean	median	no conv
'C'/'R'	**53**	**21.50**	**22**	0
'S'-min	2604	205.78	31	460 − 18
'S'-max	2530	73.70	37	82 − 70
'D'-min	2511	113.91	25	208 − 124

Test 4 comprises 15 sHH sensitive eigenvalue problems which trigger a warning that some of the computed eigenvalues might be inaccurate. The results obtained with previous version of the solver have been analyzed in detail. Therefore, they served as a reference. The performance is shown in Table 4. The option 'D'-max gave the same results as 'S'-max.

Table 4: Performance results for Test 4 set using purely semi-implicit approach.

option	max	mean	median
'C'/'R'	**209**	**54.58**	**22**
'S'-min	662	93.74	26
'S'-max	222	73.37	43
'D'-min	437	73.00	23

Test 5 includes 10^6 sHH scaled eigenvalue problems used in (Sima and Gahinet, 2019). The results are shown in Table 5. All non-converging problems converged with the "aggressive" deflation strategy.

Table 5: Performance results for Test 5 set using purely semi-implicit approach.

option	max	mean	median	no conv
'C'/'R'	5317	104.09	85	129 – 129
'S'-min	125	94.98	96	0
'S'-max	134	108.18	108	0
'D'-min	102	**71.37**	**71**	0
'D'-max	**93**	79.70	80	0

As can be seen from the tables, the best option is not the same for all tests: it is 'C'/'R' for the first four tests, but 'D' for the last test. Moreover, as the purely implicit approach, the purely semi-implicit approach did not converge in some cases, see Table 2, Table 3, and Table 5. Note that the mean of the number of iterations, and not its maximum value, is considered as the key indicator of the convergence performance. The best median value is usually encountered for the same option as the mean value of a test set.

The main reason for an occasionally slow convergence or failure of the purely semi-implicit approach is that the shifts used might not be suitable for the subproblem eigenstructure. This happened, for instance, when running an example from the Test 2 set with $n = 19$. (The corresponding sHH problem

has order 38.) Using the option 'D'-min, the last 15 eigenvalues have been found in 71 iterations, but the remaining 4×4 leading subproblem could not be solved in the next 2209 iterations. (The maximum allowed number of iterations is set to $120n = 2280$.) The shifts are chosen as $\sigma_1 = \sigma_2 = \lambda_m$, where λ_m is the real eigenvalue of F (if any) with minimum modulus. The initial submatrix F has the eigenvalues 1.513627055196251 and 5.480905879466530, so the first one has been used as a double shift. This selection rule proved to be unsuitable, since the iterations continued with very slow reduction of the subdiagonal entries of the Hessenberg matrix. During the process, the minimum absolute value of the element $a_{32}^{(1)}$ was about 1.2897. Also, $a_{43}^{(1)}$ varied in the interval $[-1.6818 \cdot 10^{-6}, 2.7095 \cdot 10^{-6}]$, with minimum absolute value of about $2.0942 \cdot 10^{-8}$. The variation of $a_{43}^{(1)}$ is shown in Figure 1. It is a slowly damped oscillation. The algorithm may converge, but in a very large number of iterations. For this example, a fast convergence is obtained when using the option 'D'-max.

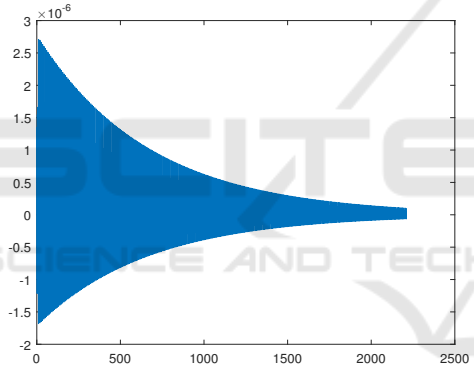

Figure 1: The variation of the $a_{43}^{(1)}$ element of an example during the iterations of the pQZ algorithm with a double real shift.

A rather good global behavior is visible in Figure 2, for the Test 5 set of problems. For the specific run on which this histogram is based, the maximum number of iterations, the mean, and the median have been 120, 84.51, and 84, respectively. The distribution is not symmetric and few problems needed a number of iterations significantly larger than the mean, for instance, more than 100 iterations.

4 COMBINING IMPLICIT AND SEMI-IMPLICIT APPROACHES

As seen in the Subsection 3.2, the convergence behavior of the purely semi-implicit pQZ algorithm can be strongly influenced by the selection of the shifts. Since no selection strategy or approach seems to be efficient for all problems, better results may be expected for a combined approach. For instance, several choices for the argument SHFT may be successively tried, or even use interleaved implicit and semi-implicit (or even explicit) approaches. Actually, the combination of implicit and semi-implicit schemes might work better than any of them, taken separately, since the shifts they use may differ, and this might prevent the algorithm in using indefinitely shifts with wrong nature or values.

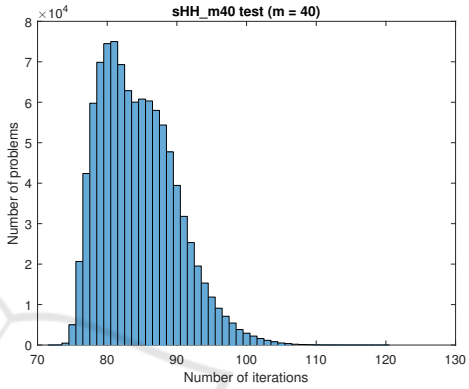

Figure 2: The histogram of the number of iterations for one run of the 10^6 scaled sHH problems.

4.1 Influence of the Selection Rules for Shifts

Some example problems are very sensitive to the way in which the shifts are chosen. Such an example is a problem of order 12 in Hessenberg-triangular form, whose non-zero part has been generated randomly from a uniform distribution in the interval (0,1), with $h = k = 5$ and $s = \begin{bmatrix} -1 & -1 & 1 & 1 & -1 \end{bmatrix}$. Therefore, the Hessenberg matrix has a negative signature. The factors have the following approximate condition numbers 265.05, 60906.88, 242.35, 192035.36, and 344.19. Evaluation of the matrix product P in (1) would give an inaccurate result. Note that for this example the solver operates with the inverse of the product, but it delivers the eigenvalues of P (see Section 2). Table 6 shows the number of iterations needed with various selection rules for the shifts σ_i. As before, q denotes the order of the current subproblem. In all cases, the parameters n_i and n_e, defining the number of iterations with implicit and explicit shifts, respectively, had the values $n_i = 1$ and $n_e = 4$. The solver has two counters, c_i and c_e, associated with these parameters, which are reset to zero for each new subproblem to be solved. If the eigenvalues of a subproblem are not obtained in n_i implicit pQZ iterations, at most n_e

semi-implicit pQZ iterations are performed, and if the subproblem is still not solved, other n_i iterations, possibly followed by n_e iterations, are similarly taken, and so on. More complicate logic has also been tried. The values $n_i = 1$ and $n_e = 4$ represent a rather inconvenient selection, but it has been used to check the capabilities of the solver in difficult settings.

Let $\lambda_{1,2} \in \Lambda(F)$ be the real eigenvalues, if any, of the 2×2 trailing submatrix F of $\underline{P} \in \mathbf{R}^{q \times q}$, and let p_{ll} be the trailing element of $\underline{P}$. Let λ_m and λ_M be the eigenvalue in $\Lambda(F)$ with minimum or maximum modulus, respectively, and λ_c be the eigenvalue in $\Lambda(F)$ with minimum distance to p_{ll}, i.e., $\lambda_c = \lambda_r, r = \arg\min\{|\lambda_i - p_{ll}|, i = 1:2\}$. Note that the selection rules in Table 6 only apply if the eigenvalues of the current submatrix F are real. Otherwise, the shifts are set to the complex conjugate eigenvalues of F. It is apparent that small changes in the selection rules for the shifts can imply large differences in the number of iterations needed for convergence. However, using larger values for n_i provides less sensitivity and a more stable behavior of the algorithm.

Table 6: Number of iterations of the pQZ algorithm for a problem of order 12 under various selection rules for the shifts.

iterations	selection rules for the shifts
65	$\sigma_i = \lambda_i$, $i = 1:2$, if q is even, and $\sigma_1 = \lambda_m$, if q is odd.
69	$\sigma_i = \lambda_i$, $i = 1:2$, if q is even, and $\sigma_1 = \lambda_c$, if q is odd.
76	$\sigma_i = \lambda_i$, $i = 1:2$, if $c_e \le n_e/2$, and $\sigma_1 = \lambda_M$, if $n_e/2 < c_e \le n_e$.
80	$\sigma_i = \lambda_i$, $i = 1:2$.
98	$\sigma_i = \lambda_i$, $i = 1:2$, if $c_e \le n_e/2$, and $\sigma_1 = \lambda_m$, if $n_e/2 < c_e \le n_e$.
65	$\sigma_i = \lambda_M$, $i = 1:2$.
65	$\sigma_i = \lambda_M$, $i = 1:2$, if $c_e \le n_e/2$, and $\sigma_1 = \lambda_M$, if $n_e/2 < c_e \le n_e$.
70	$\sigma_i = \lambda_m$, $i = 1:2$, if $c_e \le n_e/2$, and $\sigma_1 = \lambda_m$, if $n_e/2 < c_e \le n_e$.
1200	$\sigma_i = \lambda_m$, $i = 1:2$.

4.2 Numerical Results using the Combined Approach

Various values for the parameters involved in the combined implicit/semi-implicit approach have been tried. Some performance results are summarized below for selected parameter values. The solver converged for all problems. The following five tables present the results for the tests described in Subsection 3.2. Other tests are then introduced and the corresponding results are given. The tables include the performance values for two choices of the main parameters n_i and n_e, which proved to be almost optimal for all the tests performed. In addition, the best results obtained for all trials are also given (but not highlighted), for comparison, since no selection strategy is optimal for all tests. Over a dozen, and sometimes much more, of different choices of the parameters and selection rules have been tried. The results for the two pairs of parameters, $n_i = 20$, $n_e = 10$, and $n_i = 4$, $n_e = 1$, have been obtained using the following shift selection rule for real eigenvalues λ_i, $i = 1:2$:

$$\begin{aligned} \sigma_i &= \lambda_c,\ i = 1:2, \text{ if } c_e \le \max(1, n_e/2); \\ \sigma_1 &= \lambda_c, \text{ if } \max(1, n_e/2) < c_e \le n_e. \end{aligned} \qquad (13)$$

Therefore, if the subproblem has not been deflated after n_i iterations with implicit shifts, explicitly computed double shifts are used for at most $c_e \le \max(1, n_e/2)$ iterations, and then, if needed, at most $n_e/2$ single shifts are further used. In both cases, the shifts are set to the eigenvalue which is closest to the trailing element of the product, p_{ll}. Clearly, no single shifts are used for $n_e = 1$.

Table 7, ..., Table 11 show the results obtained for Test 1, ..., Test 5 sets of problems, respectively. The best results are now better than any of the results from the tables in the Subsection 3.2. Figure 3 shows the histogram of the number of iterations when running the 10^6 scaled sHH problems using the combined implicit and semi-implicit approaches, with parameters $n_i = 4$, $n_e = 1$. The histogram is thinner than that in Figure 2 and much more problems needed less iterations. The performance parameters for this run are given in Table 11.

Table 7: Performance results for Test 1 set using combined approach.

option	max	mean	median
$n_i = 20$, $n_e = 10$	57	30.65	**31**
$n_i = 4$, $n_e = 1$	**55**	**30.46**	**31**
best	55	30.25	30

Table 8: Performance results for Test 2 set using combined approach.

option	max	mean	median
$n_i = 20$, $n_e = 10$	**50**	21.59	**22**
$n_i = 4$, $n_e = 1$	**50**	**21.51**	**22**
best	35	21.50	22

Test 6 comprises 6400 matrix products in Hessenberg-triangular form, with factors generated from a pseudorandom standard uniform distribution (3200 problems) and normal distribution (3200 problems), using the MATLAB functions `rand` and `randn`, and parameters defined by $k = 5$, $n = 1:20$, $h = 1:k$,

Table 9: Performance results for Test 3 set using combined approach.

option	max	mean	median
$n_i = 20, n_e = 10$	53	21.48	**22**
$n_i = 4, n_e = 1$	**49**	**21.42**	**22**
best	49	21.37	22

Table 10: Performance results for Test 4 set using combined approach.

option	max	mean	median
$n_i = 20, n_e = 10$	**209**	**54.74**	**22**
$n_i = 4, n_e = 1$	**209**	55.26	**22**
best	198	53.53	21

and s taking all $2^k = 32$ possible distinct signature values for its k elements. The performance is shown in Table 12.

Similarly, **Test 7** includes 9600 matrix products in Hessenberg-triangular form, with the same parameters defined for Test 6, but with $n = 1 : 30$. The performance is shown in Table 13.

Also, **Test 8** includes matrix products in Hessenberg-triangular form, with the same parameters defined for Test 6, but with $n = 1 : 100$. The performance is shown in Table 14. Figure 4 shows the histogram of the number of iterations for one run of the Test 8 problems using the combined implicit and semi-implicit approaches, with parameters $n_i = 20$, $n_e = 10$. The appearance of the histogram strongly differs from that in Figure 2 and Figure 3 since the order n varies and the problems with large order need more iterations than problems with small order. The performance parameters for this run are given in Table 14.

The results reported for the two choices of the parameters n_i and n_e are close to the best results obtained so far.

5 CONCLUSIONS

A new, semi-implicit approach has been proposed for the periodic QZ algorithm and its convergence behavior has been investigated. It has been shown that combining the implicit and semi-implicit approaches and using a carefull selection of the main parameters involved contribute to an improved convergence behavior, measured by the maximum, mean, and median number of iterations needed for solving ample sets of test problems. This improvement is achieved without reducing the reliability and the accuracy of the results. The "cautious" deflation strategy has been used; the "aggressive" strategy has never been invoked in the combined approach. The latest results obtained using

Table 11: Performance results for Test 5 set using combined approach.

option	max	mean	median
$n_i = 20, n_e = 10$	168	88.47	85
$n_i = 4, n_e = 1$	**106**	**80.23**	**80**
best	86	71.24	71

Table 12: Performance results for Test 6 set using combined approach.

option	max	mean	median
$n_i = 20, n_e = 10$	69	**26.37**	**27**
$n_i = 4, n_e = 1$	**64**	26.80	**27**
best	65	24.14	24

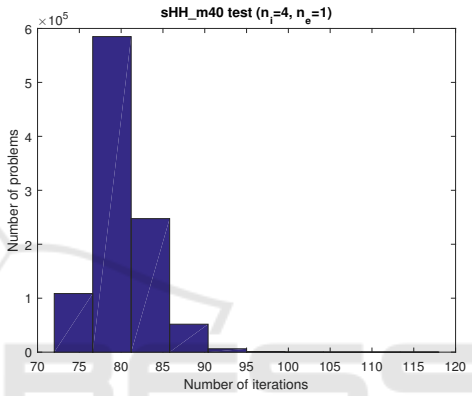

Figure 3: The histogram of the number of iterations for one run of the 10^6 scaled sHH problems using the combined implicit and semi-implicit approaches, with parameters $n_i = 4$, $n_e = 1$.

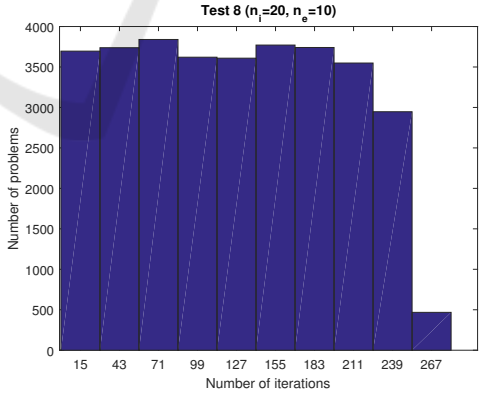

Figure 4: The histogram of the number of iterations for one run of the Test 8 problems using the combined implicit and semi-implicit approaches, with parameters $n_i = 20$, $n_e = 10$.

two sets of values for the main parameters of this approach have been always either identical or very close to the best performance results found by any other strategy or parameter values. No convergence failures have been produced.

Table 13: Performance results for Test 7 set using combined approach.

option	max	mean	median
$n_i = 20$, $n_e = 10$	**89**	**38.70**	**39**
$n_i = 4$, $n_e = 1$	96	39.30	**39**
best	89	35.51	35

Table 14: Performance results for Test 8 set using combined approach.

option	max	mean	median
$n_i = 20$, $n_e = 10$	**281**	**126.14**	**125**
$n_i = 4$, $n_e = 1$	290	128.08	127
best	280	114.29	114

Based on the numerical experience, the following findings are summarized:

- The explicit approach is bad for numerical reasons.
- Purely implicit or purely semi-explicit approach is prone to slow convergence or lack thereof.
- Good convergence is obtained when implicit approach is used by default with semi-explicit approach invoked temporarily when lack of convergence is detected.
- The semi-implicit approach uses the eigenvalues of the trailing 2×2 subproblem to select the shifts.
- When these eigenvalues are real, a good shift selection rule is to set a double shift $\sigma_1 = \sigma_2 = \lambda_c$, where λ_c is the closest eigenvalue to the last element of the product for the current subproblem.

REFERENCES

Ahues, M., Tisseur, F. (1997). A New Deflation Criterion for the QR Algorithm. LAPACK Working Note 122, UT-CS-97-353.

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999). *LAPACK Users' Guide: Third Edition*. Software · Environments · Tools. SIAM, Philadelphia.

Apkarian, P., Gahinet, P. M., and Buhr, C. (2014). Multimodel, multi-objective tuning of fixed-structure controllers. In *Proceedings 13th European Control Conference (ECC)*, 24–27 June 2014, Strasbourg, France, pages 856–861. IEEE.

Benner, P., Byers, R., Losse, P., Mehrmann, V., and Xu, H. (2007). Numerical solution of real skew-Hamiltonian/Hamiltonian eigenproblems. Technical report, Technische Universität Chemnitz, Chemnitz.

Benner, P., Byers, R., Mehrmann, V., and Xu, H. (2002). Numerical computation of deflating subspaces of skew Hamiltonian/Hamiltonian pencils. *SIAM J. Matrix Anal. Appl.*, 24(1):165–190.

Benner, P., Mehrmann, V., Sima, V., Van Huffel, S., and Varga, A. (1999). SLICOT — A subroutine library in systems and control theory. In Datta, B. N., editor, *Applied and Computational Control, Signals, and Circuits*, volume 1, chapter 10, pages 499–539. Birkhäuser, Boston, MA.

Benner, P., Sima, V., and Voigt, M. (2012a). $\mathcal{L}_\infty$-norm computation for continuous-time descriptor systems using structured matrix pencils. *IEEE Trans. Automat. Contr.*, AC-57(1):233–238.

Benner, P., Sima, V., and Voigt, M. (2012b). Robust and efficient algorithms for $\mathcal{L}_\infty$-norm computations for descriptor systems. In Bendtsen, J. D., editor, *7th IFAC Symposium on Robust Control Design (ROCOND'12)*, pages 189–194. IFAC.

Benner, P., Sima, V., and Voigt, M. (2013). FORTRAN 77 subroutines for the solution of skew-Hamiltonian/Hamiltonian eigenproblems. Part I: Algorithms and applications. SLICOT Working Note 2013-1. Available from `www.slicot.org`.

Benner, P., Sima, V., and Voigt, M. (2016). Algorithm 961: Fortran 77 subroutines for the solution of skew-Hamiltonian/Hamiltonian eigenproblems. *ACM Transactions on Mathematical Software (TOMS)*, 42(3):1–26.

Bojanczyk, A. W., Golub, G., and Van Dooren, P. (1992). The periodic Schur decomposition: Algorithms and applications. In Luk, F. T., editor, *Proc. of the SPIE Conference Advanced Signal Processing Algorithms, Architectures, and Implementations III*, volume 1770, pages 31–42.

Bruinsma, N. A. and Steinbuch, M. (1990). A fast algorithm to compute the H_∞-norm of a transfer function. *Systems Control Lett.*, 14(4):287–293.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, fourth edition.

Kressner, D. (2001). An efficient and reliable implementation of the periodic QZ algorithm. In *IFAC Proceedings Volumes (IFAC Workshop on Periodic Control Systems)*, volume 34, pages 183–188.

Kressner, D. (2005). *Numerical Methods for General and Structured Eigenvalue Problems*, volume 46 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin.

MathWorks® (2012). Control System Toolbox™, Release R2012a.

Sima, V. (2019). Computation of initial transformation for implicit double step in the periodic QZ algorithm. In Precup, R.-E., editor, *Proceedings of the 2019 23th International Conference on System Theory, Control and Computing (ICSTCC)*, 9-11 October, 2019, Sinaia, Romania, pages 7–12. IEEE.

Sima, V. and Gahinet, P. (2019). Improving the convergence of the periodic QZ algorithm. In Gusikhin, O., Madani, K., and Zaytoon, J., editors, *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics (ICINCO-2019)*, 29–31 July, 2019, Prague, Czech Republic, volume 1,

pages 261–268. SciTePress — Science and Technology Publications, Lda, Portugal.

Sreedhar, J. and Van Dooren, P. (1994). Periodic Schur form and some matrix equations. In Helmke, U., Mennicken, R., and Saurer, J., editors, *Systems and Networks: Mathematical Theory and Applications, Proceedings of the Symposium on the Mathematical Theory of Networks and Systems (MTNS'93)*, Regensburg, Germany, 2–6 August 1993, volume 1, pages 339–362. John Wiley & Sons.

Van Loan, C. F. (1975). A general matrix eigenvalue algorithm. *SIAM J. Numer. Anal.*, 12:819–834.

Xia, M., Gahinet, P. M., Abroug, N., and Buhr, C. (2017). Sector bounds in control design and analysis. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Melbourne, Australia, pages 1169–1174.