Technical Sciences
Academy of Romania
www.jesi.astr.ro

Journal of Engineering Sciences and Innovation
Volume 10, Issue 2 / 2025, p. 221 - 232

F. Electrical, Electronics Engineering, Computer Sciences and Engineering

Received 15 January 2025 ***Accepted*** *10 June 2025*
Received in revised form *14 May 2025*

Performance improvement in computing the block diagonal form of matrix pencils

VASILE SIMA*
Technical Sciences Academy of Romania, Bucharest, Romania

Abstract. Techniques for computing the block diagonal form for matrix pencils are presented. The computation starts by reducing the matrix pencil to a generalized Schur form using unitary transformations. The off-diagonal blocks are then successively annihilated by well-conditioned non-unitary transformations, using solutions of generalized Sylvester equations. The ultimate reduction to diagonal or quasi-diagonal form, with blocks of order 1 and 2, is usually impossible, since this could lead to very inaccurate results. The basic techniques try to increase the granularity by selecting appropriate eigenvalues and reordering them to improve the conditioning of the problem. For high order matrix pencils, with large and dense clusters of eigenvalues, these techniques spend much computing time unsuccessfully attempting to split such clusters. Incorporating clustering techniques into the block diagonalization process allows to detect hard to split subproblems and to accept larger diagonal blocks. Numerical results are presented illustrating the performance and effectiveness of these techniques.

Keywords: descriptor system, linear multivariable systems, numerical methods, simulation, software.

1. Introduction

One important application in the analysis and design of a linear dynamical system is the computation of its response to various inputs, including simulated disturbances. Often, such a computation has to be performed many times. Therefore, it is important to be able to do such simulations as fast as possible. One approach to reduce the computational effort by a factor about two is to first transform the original system to a (generalized) Hessenberg or Schur form. This can be done using unitary equivalence transformations. But significantly much substantial speed improvement can be obtained using the *block diagonal form*, which decouples the state part of the system into smaller subsystems, preserving as

*Correspondence address: vasilesima@ymail.com

much as possible its dynamical behavior. This reduction, however, needs non-unitary transformations. To ensure an adequate accuracy, the numerical condition of these transformations should be bounded. The reduction involves solution of (generalized) Sylvester equations. For large-scale systems, another approach is *model reduction* (see, e.g., [1] and the references therein), that reduces the order of a system, approximating its essential dynamics. The associated techniques differ from those for block diagonalization, but their aim is the same, namely, to allow faster further computations.

When the orders of the diagonal blocks are small, powers of the transformed matrices can be computed easily, and this is useful to evaluate functions (defined by power series) of the original matrices [2]. The best reduced theoretical forms, like *Jordan form* of a matrix, or *Kronecker form* of a matrix pencil, are often not well determined numerically. Small perturbations of some matrix elements can make some blocks to split or coalesce [2]. In the second case, splitting such blocks may need very ill-conditioned transformation matrices. Although this usually happens for blocks with close eigenvalues, it may happen even for matrices with well separated eigenvalues. This is related to the sensitivity of the associated eigenvalues [3], [4].

Theoretically, the order of the diagonal blocks can be no smaller than the order of the blocks in the Kronecker canonical form [5]. In practice, they have to be larger, in order to limit the conditioning of the transformation matrices [6]. If there are clusters of close eigenvalues, then usually these eigenvalues cannot be separated by well-conditioned transformations, and it could be necessary to leave them in the same diagonl block.

Consider a linear time-invariant system,

$$E\lambda\big(x(t)\big) = Ax(t) + Bu(t), \;\; y(t) = Cx(t) + Du(t), \tag{1}$$

where $A, E \in \boldsymbol{R}^{n\times n}$, $B \in \boldsymbol{R}^{n\times m}$, $C \in \boldsymbol{R}^{p\times n}$, $D \in \boldsymbol{R}^{p\times m}$, $x(t) \in \boldsymbol{R}^{n}$ is the state vector, $y(t) \in \boldsymbol{R}^{p}$ is the output vector, $u(t) \in \boldsymbol{R}^{m}$ is the input vector, and $\lambda\big(x(t)\big)$ is the differential operator, $\mathrm{d}x(t)/\mathrm{d}t$, or the advance difference operator, $\lambda\big(x(t)\big) = x(t+1)$, for continuous- and discrete-time case, respectively. The input vector can include disturbance and control components, while the output vector can contain measured and regulated components. The matrix E is singular, for instance, when model (1) includes algebraic constraints. Such systems are referred to as *descriptor* (or *singular*) *systems*. It is assumed in the sequel that the matrix pencil $\lambda E - A$ is regular, that is, $\det(\lambda E - A) \not\equiv 0$. This matrix pencil can be reduced to a simpler form, using equivalence transformations [7], which are also applied to B and C,

$$\tilde{A} = Q^T A Z, \;\; \tilde{E} = Q^T E Z, \;\; \tilde{B} = Q^T B, \;\; \tilde{C} = CZ, \tag{2}$$

where Q, $Z \in \boldsymbol{R}^{n\times n}$. In theory, the system (2) has exactly the same dynamical behavior as (1), but this is not true in practice, due to limited precision numerical computations. The best practical results can be obtained using *unitary*

transformations, more specifically, *orthogonal*, satisfying $Q^TQ = QQ^T = I_n$ and $Z^TZ = ZZ^T = I_n$, where I_n is the identity matrix of order n. These transformations preserve the norm of any matrix on which they are applied, and the computed condition numbers of the original and transformed matrices are very close, since the singular values are minimally perturbed. In the complex case, the formulas are similar, but the transposition operator T is replaced by the conjugate transpose operator, H. Any regular real matrix pencil $\lambda E - A$ can be reduced via (2) to an equivalent one, $\lambda\tilde{E} - \tilde{A}$, with theoretically the same spectrum, by orthogonal matrices Q and Z, so that the real matrix $\tilde{E}$ is upper triangular and the real matrix $\tilde{A}$ is upper quasi-triangular, i.e., $\tilde{A}$ is block triangular, with 1×1 and 2×2 diagonal blocks. If the 1×1 and 2×2 diagonal blocks correspond to real and complex conjugate eigenvalues, respectively, the pair $(\tilde{A}, \tilde{E})$ is said to be in a *generalized (real) Schur form*. In the complex case, the matrices $\tilde{A}$ and $\tilde{E}$ are both complex upper triangular. If the matrix E is identity, then the matrix A is reduced to (real) Schur form, $\tilde{A} = Q^TAQ$, and $\tilde{B}$ and $\tilde{C}$ are defined by $\tilde{B} = Q^TB$, $\tilde{C} = CQ$.
The paper is organized as follows. Section 2 presents block diagonalization techniques. The real generalized case will be mainly considered, since the other cases are simpler. Section 3 deals with implementation details associated to the developed solvers. Section 4 shows some numerical results illustrating the performance of the generalized solver. Finally, Section 5 summarizes the conclusions.

2. Block diagonalization techniques

The algorithms for block diagonalization of matrix pencils start with data in generalized Schur form. For convenience, it will be assumed that A and E are already reduced to this form, and B and C are the corresponding matrices. The transformations which are further applied to A and E are optionally accumulated and finally applied to B and C.
The generalized Schur form can be recast to as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \; E = \begin{bmatrix} E_{11} & E_{12} \\ 0 & E_{22} \end{bmatrix}, \tag{3}$$

where initially A_{11} and E_{11} are the first pair of diagonal blocks, of order 1, in the complex case, but 1 or 2, in the real case. An attempt is made to compute the following transformation matrices, X and Y, partitioned as A and E,

$$X = \begin{bmatrix} I & V \\ 0 & I \end{bmatrix}, \; Y = \begin{bmatrix} I & W \\ 0 & I \end{bmatrix}, \tag{4}$$

where I are identity matrices of appropriate order, such that

$$X^{-1}AY = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}, \; X^{-1}EY = \begin{bmatrix} E_{11} & 0 \\ 0 & E_{22} \end{bmatrix} \tag{5}$$

and the elements of X and Y do not exceed, in magnitude, a given value τ, $\tau > 1$. A typical value for τ is 5000, but it can be as lower as 100 for problems with well-conditioned spectra. The special structure of X ensures that X^{-1} is easily obtained, since it has the same form as X, but V is replaced by $-V$. Using (4) in (5), it follows that

$$X^{-1}AY = \begin{bmatrix} I & -V \\ 0 & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \begin{bmatrix} I & W \\ 0 & I \end{bmatrix} = \begin{bmatrix} A_{11} & A_{11}W + A_{12} - VA_{22} \\ 0 & A_{22} \end{bmatrix}, \quad (6)$$

and $X^{-1}EY$ has a similar formula. The block diagonal form in (5) is obtained if V and W satisfy the following *generalized Sylvester equation* [8]

$$A_{11}W - VA_{22} = -\sigma A_{12}\,, \quad E_{11}W - VE_{22} = -\sigma E_{12}\,, \quad (7)$$

where $0 \leq \sigma \leq 1$ is a scaling factor, used for instance by the LAPACK solver [9], to avoid overflow in the computations. Usually, $\sigma = 1$, and smaller values indicate possible ill-conditioning.

If all elements of V and W have magnitude less than or equal to τ, the blocks A_{11} and E_{11} are accepted and the matrices then have the following form

$$A = \begin{bmatrix} A_{11} & 0 & 0 \\ 0 & \hat{A}_{11} & \hat{A}_{12} \\ 0 & 0 & \hat{A}_{22} \end{bmatrix}, \quad \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ 0 & \hat{A}_{22} \end{bmatrix} \coloneqq A_{22},$$

$$E = \begin{bmatrix} E_{11} & 0 & 0 \\ 0 & \hat{E}_{11} & \hat{E}_{12} \\ 0 & 0 & \hat{E}_{22} \end{bmatrix}, \quad \begin{bmatrix} \hat{E}_{11} & \hat{E}_{12} \\ 0 & \hat{E}_{22} \end{bmatrix} := E_{22}\,. \quad (8)$$

The new transformation matrices will act on the part with hat accents, that is, their form will be

$$\hat{X} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & \hat{V} \\ 0 & 0 & I \end{bmatrix}, \quad \hat{Y} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & \hat{W} \\ 0 & 0 & I \end{bmatrix}, \quad (9)$$

where $\hat{V}$ and $\hat{W}$ will solve, if possible, the equations in (7) for submatrices with hat accents. Hence, the essential part of each individual transformation has the form in (4). If $\hat{V}$ and $\hat{W}$ are accepted, then the current transformation matrices will be $X \coloneqq X\hat{X}$ and $Y \coloneqq Y\hat{Y}$, with

$$X = \begin{bmatrix} I & V_{11} & V_{12} \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & I & \hat{V} \\ 0 & 0 & I \end{bmatrix} = \begin{bmatrix} I & V_{11} & V_{11}\hat{V} + V_{12} \\ 0 & I & \hat{V} \\ 0 & 0 & I \end{bmatrix}, \quad [V_{11} \quad V_{12}] \coloneqq V,$$

and similarly for Y. It follows that updating X and Y for new individual transformations only involves the matrix operations of the form $V_{11}\hat{V} + V_{12}$ and $W_{11}\hat{W} + W_{12}$.

But if any of the elements of V and W exceeds τ in magnitude, or if the linear system corresponding to (7) is (almost) singular (that happens when the pairs (A_{11}, E_{11}) and (A_{22}, E_{22}) have common or very close eigenvalues), the blocks A_{11} and E_{11} are not accepted, since the transformations X and Y are considered to be too ill-conditioned. However, it could be possible to extend A_{11} and E_{11} by including other diagonal block(s) from A_{22} and E_{22} , and find acceptable transformations. There are strategies to choose suitable blocks so that the conditioning of the extended problem is acceptable. One such strategy finds a pair of 1×1 (or 2×2) diagonal blocks of A_{22} and E_{22} whose eigenvalue(s) are the closest to the mean of eigenvalues of the pair (A_{11}, E_{11}). This block pair is moved by orthogonal equivalence transformations to the leading position of A_{22} and E_{22} [7]. The moved diagonal blocks are then appended to A_{11} and E_{11}, increasing their size by 1 (or 2), and another attempt is made to solve the corresponding generalized Sylvester equation (7). If the new transformation matrices have all elements with magnitude at most τ, the current blocks A_{11} and E_{11} are accepted, the corresponding matrices X and Y postmultiply the current transformation matrices and the same procedure is applied to the blocks A_{22} and E_{22}. Other block selection strategies will be discussed in Section 3. This approach can be seen as being of „bottom-up" type.

Optionally, the transformations matrices can be initialized on input to the solver, and they can be updated during the block diagonalization process. This option is useful, for instance, when the original system matrices A and E are not in a generalized Schur form; in this case, X and Y are initialized by the left and right, respectively, orthogonal matrices generated in the reduction to this Schur form. At the end of the diagonalization, X and Y will contain the matrices that would reduce the original data A and E to the block diagonal form. Therefore, X and Y can then be applied to B and C, ensuring the equivalence between the original and the reduced system.

The block diagonalization solver in the SLICOT Library [10], developed by the author, delivers the transpose of the final matrix X. This ensures the compatibility with the reduction to the generalized Schur form, performed by the LAPACK routines, since the reduced matrices are obtained in the same form: $\tilde{A} = Q^T A Z$, $\tilde{E} = Q^T E Z$ and $\tilde{A} = X^T A Y$, $\tilde{E} = X^T E Y$, for Schur reduction and block diagonalization, respectively. In each successful step of the block diagonalization, only the rows and columns involved in that step need to be updated in X and Y. (Note that the MATLAB command `qz` returns the matrix Q^T instead of Q.)

In the standard case, only the matrix A has to be transformed, $W = V$, and V is obtained by solving a Sylvester equation [11], given by the first formula in (7) with $W = V$. Adaptations of the solvers for standard Sylvester equation [11] and generalized Sylvester equation [8], for controlling the magnitude of the individual elements of the computed solution [2], are used to obtain V and W.

The algorithm usually requires $O(n^3)$ operations, but $O(n^4)$ are possible in the worst case, when the matrix pencil cannot be block diagonalized by well-conditioned transformations. The individual non-orthogonal transformation matrices used in the reduction of A and E to a block diagonal form have condition numbers of the order τ. This does not guarantee that their product is well-conditioned enough. But it can be ensured that the transformations X and Y have condition numbers, $\text{cond}(X)$ and $\text{cond}(Y)$, respectively, that are not much latger than τ, where $\text{cond}(M) := \|M\|\|M^{-1}\|$ for any square matrix M. This can be obtained by scaling the rows and columns of each individual transformation matrices like in (9), so that $\|X_{i,:}\| = 1$ and $\|Y_{:,i}\| = 1$. Diagonal scaling matrices, D_X and D_Y are used, where $D_X = \text{diag}(I, D^V)$ and $D^V_{ii} = 1/(1 + \|V_{i,:}\|^2)^{1/2}$ for each row i of V, and similarly for Y. If $\|V\|$, hence $\|X\|$ are large, which would imply an ill-conditioned matrix X, it follows that $\|D_X X\| \approx 1$ and $\|(D_X X)^{-1}\| \approx \sqrt{1 + \|V\|^2} \approx \|V\|$, hence $\text{cond}(D_X X) = \|D_X X\|\|(D_X X)^{-1}\| \approx \|V\|$; without this normalization, one would obtain $\text{cond}(X) \approx \|V\|^2$. Clearly, the scaling used for normalization is also applied to the corresponding parts of A and E in order to ensure the equivalence between the original and the reduced matrix pencils. Normalization of the columns of Y which are no longer modified, and updating of A and E, is done during the reduction process. Normalization of all columns of X (not rows, due to the computation of the transposed matrix), and the corresponding update of A and E are done at the end of the computational process.

3. Implementation issues

The computation of block diagonal forms for matrices and matrix pencils, with real or complex elements, can be performed using subroutines from the SLICOT Library [10], available on GitHub, https://github.com/SLICOT/SLICOT-Reference. The data matrices are assumed to be in Schur or generalized Schur form, which can be obtained by LAPACK subroutines. There are options to specify the desired strategy and the bound τ on the magnitude of the elements of the individual transformations. The tolerance θ, used by the strategies for selecting the blocks to be moved, can be specified, but a default value can be set instead. The optimal size of the real working array can be computed by the generalized solver using a special call with this size set to -1; the returned value can then be given as input argument in a second solver call. The use of this feature could reduce the computing time for systems with large order, due to calls to the needed efficient BLAS 3 routines [12], instead of the slower, but some memory saving BLAS 2 routines. To conserve memory, at each successful step of the reduction, the matrices V and W are stored in the memory space for A_{12} and E_{12}. However, A_{12} and E_{12} have to be saved before calling the generalized Sylvester solver, since it may use this memory space before an element of magnitude larger than τ is found; in such a case the calculations must be redone for a larger block pair, that should include the previous A_{12} and E_{12}. Actually, A_{12}^T and E_{12}^T are saved in the

corresponding zero lower triangular part of A and E, respectively. After a successful reduction step and updating X and Y (if desired), the space for A_{12}, E_{12}, A_{12}^T and E_{12}^T is set to zero.

The current block diagonalization routines included in SLICOT do not use advanced clustering information. Several strategies are available for selecting a new block pair to be added to the already reduced leading block diagonal form. The desired strategy is chosen by specifying an input argument, SORT, of the routine. The „closest to the mean" strategy, discussed in Section 2, is selected by setting SORT = 'N'. In a variation of this strategy, used by setting SORT = 'S', the diagonal blocks of the generalized real Schur form are reordered before each step of the reduction, so that each cluster of generalized eigenvalues, defined as specified in the description of the tolerance θ below, appears in adjacent blocks. The blocks for each cluster are merged together, and the procedure described in Section 2 is applied to the larger blocks. Using the option SORT = 'S' will usually provide better efficiency than the standard option (SORT = 'N'), proposed in [2], because there could be no or few unsuccessful attempts to compute individual transformation matrices X and Y of the form (4) or (9). However, the resulting dimensions of the blocks are usually larger; this could make subsequent calculations less efficient.

For other two strategies, chosen by setting SORT = 'C' or 'B', the procedure is similar to that for SORT = 'N' or 'S', respectively, but the blocks of A_{22} and E_{22} whose eigenvalue(s) is (are) the closest to those of (A_{11}, B_{11}) (not to their mean) are selected and moved to the leading position of A_{22} and E_{22}. This is called the „closest-neighbour" strategy.

If SORT is set to 'S' or 'B', the tolerance θ is used for reordering the diagonal blocks of the block upper triangular matrix pair. If $\theta > 0$, then the given value of θ is used as an absolute tolerance: a pair of blocks i and a temporarily fixed pair of blocks α (the first pair of blocks of the current trailing pair of submatrices to be reduced) are considered to belong to the same cluster if their eigenvalues satisfy the following „distance" condition $|\mu_\alpha - \mu_i| \leq \theta$, where μ_α and μ_i denote the eigenvalues of two block pairs. If $\theta < 0$, then the given value of θ is used as a relative tolerance: the pairs of blocks i and α are considered to belong to the same cluster if their eigenvalues satisfy, for finite eigenvalues μ_j, $|\mu_\alpha - \mu_i| \leq |\theta| \times \max\{|\mu_j|, j = 1, \dots, n\}$. If $\theta = 0$, then an implicitly computed, default tolerance, defined by $\theta = \varepsilon_M^{1/4}$ is used instead, as a relative tolerance, where ε_M is the machine precision, $\varepsilon_M \approx 2.22 \times 10^{-16}$. The approximate symmetric *chordal metric* is used as „distance" of two complex, possibly infinite numbers, x and y. This metric is given by the formula

$$d(x, y) = \min\left(|x - y|, \left|\frac{1}{x} - \frac{1}{y}\right|\right), \tag{10}$$

taking into account the special cases of infinite or NaN values. If SORT = 'N' or 'C', the tolerance θ is not used.

These bottom-up strategies are very efficient for matrix pencils with relatively small order and well separated eigenvalues, in which case the solver often succeeds to obtain diagonal blocks of order at most two. For large order problems with clustered eigenvalues, the solution time can be high, due to the possibility to have a big number of unsuccessful attempts to split the blocks. A preliminary analysis of the clustered structure of the spectrum, followed by an appropriate reordering of the eigenvalues, could improve the efficiency. Specifically, starting by the most separated eigenvalues, the solver could quickly decouple them, leaving to the end all possibly big clusters of eigenvalues. Few failed attempts to split such a cluster could signal that there is no reason to continue the computations and therefore, finish the process with one or more large blocks. These strategies could be considered as being of „top-down" type. In implementation, the $n_p(n_p-1)/2$ pairwise distances between all n_p eigenvalues with nonnegative imaginary parts are computed, since the eigenvalues with negative imaginary parts should be considered together with their complex conjugate counterparts. Euclidean distance is used if there are only finite eigenvalues; otherwise, chordal metric (10) can be computed, in order to deal in the same way with finite and infinite eigenvalues. (Alternatively, all infinite eigenvalues can be included in the same cluster and can be separated from the beginning.) The distance information is used to build a *linkage* matrix, which shows how the eigenvalues should be grouped into clusters. The size of this matrix is $(n_p-1)\times 3$. The third column contains a selected list of distances, in decreasing order, while the first two columns specify the objects which are grouped together into binary clusters. The objects are eigenvalues or detected groups of close eigenvalues. The newly formed objects are linked together and to other objects into bigger clusters, until all n_p eigenvalues are linked in a binary tree. The clusters that contain complex eigenvalues are extended with the corresponding eigenvalues with negative imaginary parts; the eigenvalues in a complex conjugate pair appear successively, with eigenvalue having positive imaginary part in the first position. The number of clusters to be considered can be specified. This number should be smaller than n_p, possibly much smaller.

4. Numerical results

Extensive testing has been performed to evaluate the standard and generalized block diagonalization solvers. The computations have been done in double precision on an Intel Core i7-3820QM portable computer (2.7 GHz, 16 GB RAM). Some results obtained with the generalized solver are presented below. An executable MEX-file has been built using the solver source code, SLICOT routines and MATLAB-provided optimized LAPACK and BLAS routines.

Example 1. Consider first a random example generated using the following MATLAB commands

```
A = rand( n );  E = rand( n );   p = 10^3*rand( n, 1 );  q = rand( n, 1 )/10^2;
P = diag( p );  Q = diag( q );   A = P*A*Q;  E = P*E*Q;
```

The random sequence has been initialized using the command rng('default'), for reproductibility of the results. The executable solver has been called with all four options for SORT, with either original matrices or in their generalized Schur form, and with transformation matrices computed or not. Hence, there are 16 calls for each value of n. For $n = 50$, $\tau = 100$, the maximum block size for all calls has been 2, and the total CPU time has been 0.079542 s (seconds), resulting a mean time of 0.0049714 s. The mean and standard deviation of the 24 relative errors between initial and final eigenvalues (16 errors) and between the initial and transformed matrix pairs (8 errors) have been 2.1649e-16 and 7.56342e-17, respectively. The relative errors are computed as follows

$$e_A = \left\|X^T\tilde{A}Y - \hat{A}\right\|/\max\left(1, \left\|\tilde{A}\right\|\right), \quad e_E = \left\|X^T\tilde{E}Y - \hat{E}\right\|/\max\left(1, \left\|\tilde{E}\right\|\right),$$
$$e_\mu = \|\tilde{\mu} - P\hat{\mu}\|/\max\left(1, \|\tilde{\mu}\|\right), \qquad (11)$$

where P is a permutation matrix (chosen to reorder $\hat{\mu}$ in agreement to $\tilde{\mu}$), and the variables with tilde and hat accents correspond to the generalized Schur form and the computed results, respectively.

For $n = 100$, $\tau = 100$, the solver could not split the spectrum, so the maximum block size for all calls has been 100. The total CPU time has been 0.565525 s and its mean value 0.035345 s. The mean and standard deviation of the relative errors have been 1.9846e-15 and 2.2303e-15, respectively. The spectrum of the matrix pencil is shown in Fig.1. Although there are several well separated eigenvalues, these could not be split. However, using $\tau = 5000$ for the same example, a perfect block diagonalization is obtained, that is, the number of blocks of size 2 equals the number of complex conjugate eigenvalues. The total CPU time has been 0.25866 s, resulting a mean time of 0.016166 s. The mean and standard deviation of all 24 relative errors in (11) have been 1.3076e-16 and 5.8876e-17, respectively. A similar behavior also appears for $\tau = 100$, but without scaling of A and E in the MATLAB commands above.

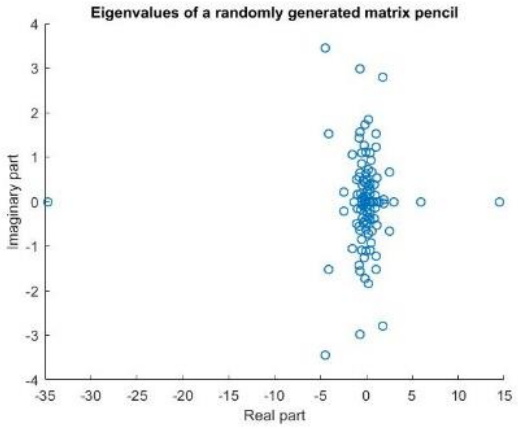

Fig.1. The spectrum of a randomly generated matrix pencil of order 100.

Using several strategies for a preliminary reordering of the eigenvalues $\tilde{\mu}$, for 6 clusters (of eigenvalues with nonnegative imaginary parts), perfect block diagonalization results have been obtained, and the CPU times for those strategies varied between 0.0056 s and 0.011479 s. Moreover, the same results have been produced even with $\tau = 100$, when the CPU times varied between 0.004889 s and 0.012211 s. Therefore, the mean CPU time has been reduced by a factor of about 2 compared to the results using botom-up strategies. The performance statistics are summarized in Table 1, where b denotes the vector of diagonal block orders and $e \in \mathbf{R}^{24}$ contains the eight values $\max(e_A, e_E)$ and 16 values e_μ for all 16 solver calls.

Table 1. Example 1 performance statistics.

n	τ	$\max(b)$	CPU time (s)	mean(CPU time)	mean(e)	std(e)
50	100	2	0.079542	0.0049714	2.1649e-16	7.5634e-17
100	100	100	0.565525	0.035345	1.9846e-15	2.2303e-15
100	5000	2	0.25866	0.016166	1.3076e-16	5.8876e-17

Example 2. A matrix pencil of order 999 has been investigated. It has 107 real are 446 complex conjugate eigenvalues. The eigenvalues with nonnegative imaginary parts are displayed in Fig.2. Without using clustering information, the block diagonalization solver, with SORT = 'N' and $\tau = 5000$, obtained a solution with 135 blocks in about 62.68 s. The largest block, of order 734, appeared in the 89-th position; there are three 1×1 and 131 2×2 diagonal blocks. The remaining 104 real eigenvalues are included in the largest block. The same block structure and comparable CPU times have been recorded for the other values of the parameter SORT. The existence of such a large block is due to a great number of badly separated eigenvalues, both real and complex conjugate.

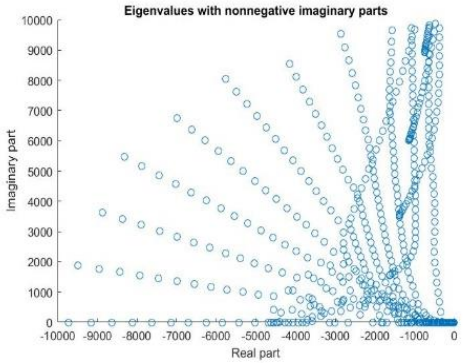

Fig. 2. The eigenvalues with nonnegative imaginary parts of a matrix pencil of order 999.

To investigate the advantages of exploiting the clustering information, several values for the number of clusters have been tried. The best results have been obtained for 120 clusters. There are 141 diagonal blocks, with the largest block, of order 719, appearing in the last position; the other 140 diagonal blocks are 2×2. All real eigenvalues are included in the largest block. The total CPU time for reordering the eigenvalues $\tilde{\mu}$ and block diagonalization has been 4.6905 s, that is, this execution was 13.36 times faster than the execution without using clustering information. Moreover, more blocks have been found, and the largest block has a smaller order. This illustrates the benefits of considering clustering techniques in the block diagonalization algorithm. The performance statistics are summarized in Table 2. The numbers of 1×1 and 2×2 blocks are also given.

Table 2. Example 2 performance statistics.

size(b)	max(b)	position	# 1×1	# 2×2	CPU time (s)	# clusters
135	734	89	3	131	62.68	-
141	719	141	0	140	4.69	120

5. Conclusions

Techniques for computing the block diagonal form for matrix pencils are presented. Such a form is very useful, for instance, for fast simulation of linear time-invariant descriptor systems, since their state dynamics can often be very well approximated by a series of decoupled subsystems of much lower sizes. Starting by reducing the matrix pencil to a generalized Schur form by unitary transformations, the off-diagonal blocks are then successively annihilated by well-conditioned non-unitary transformations, using solutions of generalized Sylvester equations. The ultimate reduction to diagonal or quasi-diagonal form, with blocks of order 1 (and 2, in the real case), is often impossible, since this could drastically diminish the accuracy of the results. The basic techniques attempt to increase the granularity as much as possible, by selecting appropriate eigenvalues and reordering them to improve the conditioning of the problem. For high order matrix pencils, with large and dense clusters of eigenvalues, these techniques may spend much computing time unsuccessfully trying to split such clusters into smaller ones. Further investigation will be devoted for incorporating advanced clustering techniques into the block diagonalization process for detecting subproblems hard to split, and therefore quickly accepting larger order diagonal blocks. Numerical results are presented which illustrate the performance and effectiveness of these techniques; for instance, using clustering information the computing effort was reduced by a factor bigger than 10 for a matrix pencil of order 999.

Acknowledgements

Acknowledgements are addressed to Pascal Gahinet for valuable suggestions.

References

[1] Mehrmann V., Stykel T., *Balanced truncation model reduction for large-scale systems in descriptor form*, Benner, P., Mehrmann, V., D. Sorensen (eds.), Dimension Reduction of Large-Scale Systems, vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin, Heidelberg, New York, 2005, ch. 3, p. 89–116.

[2] Bavely C.A., Stewart G.W., *An algorithm for computing reducing subspaces by block diagonalization*, SIAM J. Numer. Anal., **16**, 2, 1979, p. 359–367.

[3] Wilkinson J.H., *The Algebraic Eigenvalue Proble,* Clarendon Press, Oxford, UK, 1965.

[4] Stewart, G.W., *On the sensitivity of the eigenvalue problem* $Ax = \lambda Bx$, SIAM J. Numer. Anal., **9**, 1972, p. 669–686.

[5] Wilkinson J.H., *Kronecker's canonical form and the QZ algorithm*, Lin. Alg. Appl., **28**, 1979, p. 285–303.

[6] Demmel J., *The condition number of equivalence transformations that block diagonalize matrix pencils*, SIAM J. Numer. Anal., **20**, 1983, p. 599–610.

[7] Golub, G.H., Van Loan, C.F., *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 4th edition, 2013.

[8] Kagström B., Westin L., *Generalized Schur methods with condition estimators for solving the generalized Sylvester equation*, IEEE Trans. Auto. Contr., **34**, 1989, p. 745–751.

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

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

[11] Bartels R.H., Stewart G.W., *Algorithm 432: Solution of the matrix equation* $AX + XB = C$, Comm. ACM, **15**, 9, 1972, p. 820–826.

[12] Dongarra J.J., Du Croz J., Duff I.S., Hammarling S., *Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms*, ACM Trans. Math. Softw., **16**, 1990, p. 1–17, 18–28.