SIAM J. MATRIX ANAL. APPL.
Vol. 43, No. 2, pp. 638–660

MIXED PRECISION RECURSIVE BLOCK DIAGONALIZATION FOR BIVARIATE FUNCTIONS OF MATRICES*

STEFANO MASSEI† AND LEONARDO ROBOL‡

Abstract. Various numerical linear algebra problems can be formulated as evaluating bivariate function of matrices. The most notable examples are the Fréchet derivative along a direction, the evaluation of (univariate) functions of Kronecker-sum-structured matrices, and the solution of, Sylvester matrix equations. In this work, we propose a recursive block diagonalization algorithm for computing bivariate functions of matrices of small to medium size, for which dense linear algebra is appropriate. The algorithm combines a blocking strategy, as in the Schur–Parlett scheme, and an evaluation procedure for the diagonal blocks. We discuss two implementations of the latter. The first is a natural choice based on Taylor expansions, whereas the second is derivative-free and relies on a multiprecision perturb-and-diagonalize approach. In particular, the appropriate use of multiprecision guarantees backward stability without affecting the efficiency in the generic case. This makes the second approach more robust. The whole method has cubic complexity, and it is closely related to the well-known Bartels–Stewart algorithm for Sylvester matrix equations when applied to $f(x,y)=\frac{1}{x+y}$. We validate the performances of the proposed numerical method on several problems with different conditioning properties.

Key words. bivariate matrix functions, mixed precision, block diagonalization, perturb-and-diagonalize, multiprecision, function of matrices

AMS subject classification. 65F60

DOI. 10.1137/21M1407872

1. Introduction.

Matrix functions [14], such as the matrix inverse, the matrix exponential, the matrix square root, and many others, arise in an endless list of applications including analysis of complex networks [10], signal processing [17], solution of ODEs [18], and control theory [3]. The practical computation of univariate functions of matrices has been intensively analyzed from different angles such as the reduction to triangular form [9], polynomial and rational approximants [15], contour integrals [12], and projection on low dimensional subspaces [11].

The matrix function concept extends quite naturally to the bivariate setting. Given two square matrices $A \in \mathbb{C}^{m\times m}$ and $B \in \mathbb{C}^{n\times n}$ and a complex-valued function $f(x,y)$, the *bivariate matrix function* $f\{A,B\}$ [22] is a linear endomorphism on $\mathbb{C}^{m\times n}$. As in the univariate case, the definition of $f\{A,B\}$ can be given, equivalently, in terms of (bivariate) Hermite interpolation, power series expansion, and contour integration. We report the latter formulation which is the most useful for our work. Let Λ_A and Λ_B be the spectra of A and B, respectively, and let $f(x,y)$ be analytic in an open neighborhood of $\Lambda_A \times \Lambda_B$; $f\{A,B\}$ is defined as$f\{A,B\} : \mathbb{C}^{m\times n} \longrightarrow \mathbb{C}^{m\times n}$, where

$$f\{A,B\}(C) = -\frac{1}{4\pi^2}\iint_{\Gamma} f(x,y)(xI-A)^{-1}C(yI-B^T)^{-1}\,dxdy, \tag{1}$$

with $\Gamma := \Gamma_A \times \Gamma_B$ and Γ_A, Γ_B closed contours enclosing Λ_A and Λ_B, respectively.

*Received by the editors April 1, 2021; accepted for publication (in revised form) by D. J. Higham October 29, 2021; published electronically April 19, 2022.
https://doi.org/10.1137/21M1407872
Funding: The work of the authors was partially supported by the INdAM-GNCS project "Metodi low-rank per problemi di algebra lineare con struttura data-sparse."

†Centre for Analysis, Scientific Computing, and Applications, TU/e, Eindhoven, North Brabant, 1015, The Netherlands (massei.stef@gmail.com).
‡Department of Mathematics, University of Pisa, Pisa, 56127, Italy (leonardo.robol@unipi.it).

Although their usual formulations involve different frameworks, the following popular linear algebra problems correspond to evaluate a bivariate matrix function:

1. The solution of the *Sylvester equation*
$$AX + XB = C$$
is given by $X = f_1\{A, B\}(C)$, where $f_1(x, y) = \frac{1}{x+y}$.
2. Given a univariate matrix function $g(A)$, the *Fréchet derivative* of g at A in the direction C, i.e., $Dg\{A\}(C) := \lim_{t\to 0} \frac{1}{t}(g(A + tC) - g(A))$, verifies $Dg\{A\}(C) = f_2\{A, A^T\}(C)$, where $f_2(x, y)$ is the finite difference quotient
$$f_2(x, y) = \begin{cases} \frac{g(x)-g(y)}{x-y}, & x \neq y, \\ g'(x), & x = y. \end{cases}$$
3. Given a univariate function $h(x)$, a matrix with the Kronecker sum structure $\mathcal{A} = A \otimes I + I \otimes B$, and a vector v, we have that
$$h(\mathcal{A})v = \operatorname{vec}\left(f_3\{A, B\}(C)\right), \qquad f_3(x, y) = h(x + y),$$
where the matrix C verifies $\operatorname{vec}(C) = v$.

We stress that effective algorithms specialized for each of these case studies, or their subcases, already exist; see [25] for Sylvester equations, [1, 2] for the Frechét derivative, and [5, 24] for functions of Kronecker sums. Quite recently, computing the application of a generic bivariate matrix function to a low-rank matrix C has been addressed in [23], and a multivariate version of the Crouzeix–Palencia bound [7] has been proved in [6]. The ultimate goal of this work is to provide an algorithm for the computation of $f\{A, B\}(C)$ in the most general scenario, i.e., only requiring that the matrices A, B, C have appropriate sizes and that f is analytic on the Cartesian product of the spectra of A and B.

1.1. Diagonalization of A and/or B. If at least one among A and B is a normal matrix or has a well-conditioned eigenvector matrix, then evaluating $f\{A, B\}(C)$ simplifies considerably.

Note that if $D_A := \operatorname{diag}(\lambda_1^A, \dots, \lambda_m^A)$ and $D_B := \operatorname{diag}(\lambda_1^B, \dots, \lambda_n^B)$ are diagonal matrices, then $X := f\{D_A, D_B\}(C)$ is given by $(X)_{ij} = f(\lambda_i^A, \lambda_j^B) \cdot C_{ij}$. In addition, (1) implies the following property that describes the interplay between bivariate matrix functions and similarity transformations: Given invertible matrices $S_A \in \mathbb{C}^{m\times m}$ and $S_B \in \mathbb{C}^{n\times n}$, it holds that

$$f\{A, B\}(C) = S_A \cdot f\{S_A^{-1}AS_A, S_B^{-1}BS_B\}(S_A^{-1}CS_B^{-T}) \cdot S_B^T. \tag{2}$$

In the case $A = S_A D_A S_A^{-1}$, $B = S_B D_B S_B^{-1}$ for well-conditioned S_A and S_B (so that λ_i^A, λ_j^B are the eigenvalues of A, B), we make use of (2) to get

$$\begin{aligned} f\{A, B\}(C) &= S_A f\{D_A, D_B\}(\widetilde{C})S_B^T = S_A(F \circ \widetilde{C})S_B^T, \\ \widetilde{C} &:= S_A^{-1}CS_B^{-T}, \qquad (F)_{ij} := f(\lambda_i^A, \lambda_j^B), \end{aligned} \tag{3}$$

where $\circ$ indicates the componentwise Hadamard product of matrices. In particular, only evaluations of f on scalar entries are needed; we call the procedure based on (3) FUN2_DIAG, and we report it in Algorithm 1.

Algorithm 1

procedure FUN2_DIAG(f, A, B, C)
  $[S_A, D_A] = \texttt{eig}(A)$
  $[S_B, D_B] = \texttt{eig}(B)$
  $\widetilde{C} \leftarrow S_A^{-1} C S_B^{-T}$
  for $i = 1 \dots m$, $j = 1, \dots, n$ **do**
    $F(i,j) \leftarrow f(\lambda_i^A, \lambda_j^B)$
  end for
  return $S_A(F \circ \widetilde{C}) S_B^T$
end procedure

If only B is diagonalized, then it is convenient to rely on the following formula [22, section 5]:

$$f\{A,B\}(C) = f\{A, D_B\}(\widetilde{C}) S_B^T = [f_{\lambda_1^B}(A)\widetilde{c}_1 | \dots | f_{\lambda_n^B}(A)\widetilde{c}_n], \quad \widetilde{C} := C S_B^{-T} = [\widetilde{c}_1 | \dots | \widetilde{c}_n], \qquad f_{\lambda_j^B}(x) := f(x, \lambda_j^B). \tag{4}$$

Since the previous expression only involves the evaluation of univariate matrix functions, it is performed via the Schur–Parlett algorithm [9], which is implemented, for instance, in the `funm` MATLAB function. An analogous rowwise formula, involving the univariate functions $f_{\lambda_j^A}(y) := f(\lambda_j^A, y)$, applies to the case where only A is diagonalized. The resulting procedures are denoted by FUN2_DIAGA and FUN2_DIAGB and are reported in Algorithm 2 and Algorithm 3.

Algorithm 2

procedure FUN2_DIAGA(f, A, B, C)
  $[S_A, D_A] = \texttt{eig}(A)$
  $\widetilde{C} \leftarrow S_A^{-1} C$
  $D \leftarrow \mathbf{0}_{m \times n}$
  for $j = 1, \dots m$ **do**
    $D(j, :) \leftarrow \widetilde{C}(j, :) f_{\lambda_j^A}(B)$
  end for
  return $S_A D$
end procedure

Algorithm 3

procedure FUN2_DIAGB(f, A, B, C)
  $[S_B, D_B] = \texttt{eig}(B)$
  $\widetilde{C} \leftarrow C S_B^{-T}$
  $D \leftarrow \mathbf{0}_{m \times n}$
  for $j = 1, \dots n$ **do**
    $D(:, j) \leftarrow f_{\lambda_j^B}(A)\widetilde{C}(:, j)$
  end for
  return $D S_B^T$
end procedure

1.2. Contribution. From now on we will consider $f\{A, B^T\}(C)$ (instead of $f\{A,B\}(C)$) because this simplifies the exposition. We propose a numerically reliable method for computing $f\{A, B^T\}(C)$ for a general function $f(x, y)$ without requiring that A and/or B can be diagonalized with a well-conditioned similarity transformation. In complete analogy to the univariate scenario, our procedure computes the Schur decompositions $A = Q_A T_A Q_A^*$ and $B = Q_B T_B Q_B^*$ so that the task boils down to evaluate the bivariate function for triangular coefficients:

$$f\{A, B^T\}(C) = Q_A f\{T_A, T_B^T\}(\widetilde{C}) Q_B^*, \qquad \widetilde{C} := Q_A^* C Q_B.$$

A generalized block recurrence is applied to retrieve $f\{T_A, T_B^T\}(\widetilde{C})$; the recursion requires us to compute f on pairs of diagonal blocks of T_A and T_B^T and to solve Sylvester equations involving either diagonal blocks of T_A or of T_B. In view of the latter operation, we need to reorder the Schur forms of A and B such that distinct diagonal blocks have sufficiently separated eigenvalues. Finally, we evaluate f on the smallest

diagonal blocks of T_A and T_B^T, the so-called *atomic blocks*, via a truncated bivariate Taylor expansion or, in the spirit of [16], with a randomized approximate diagonalization technique combined with high precision arithmetic. As we discuss in section 3.1, the procedure can be interpreted as an implicit (recursive) block diagonalization strategy, where the eigenvectors matrices are not formed explicitly.

The paper is organized as follows; in section 2 we describe the various steps of the algorithm in detail. In particular, section 2.1 discusses the blocking procedure, section 2.2 contains the two implementations of the function evaluation of the atomic blocks, and section 2.3 provides further information about implementation aspects and complexity analysis. The focus of section 3 is on the connection of our method with block diagonalization and the Bartels–Stewart algorithm. Numerical results are reported in section 4, and conclusions are drawn in section 5.

2. Recursive block diagonalization for bivariate matrix functions. The univariate Schur–Parlett algorithm computes $f(A)$, for a triangular A, by exploiting that A and $f(A)$ commute. This property leads to a set of equations that allows us to retrieve the entries of $f(A)$ a superdiagonal at a time, starting with the diagonal elements.

A natural question is whether the triangular structure of A and B can be exploited in the bivariate framework. However, here the situation is a bit different because the goal is to compute the application of $f\{A, B^T\}$ to a matrix argument; the correspondent univariate operation is computing $f(A)v$ for a given vector v, for which the Schur–Parlett scheme is not applicable. Our strategy leverages the triangular structure of A and B to split the computation into smaller tasks. In order to show how the splitting works we state the following technical result.

LEMMA 1. *Let $A \in \mathbb{C}^{m\times m}$ be a triangular matrix block partitioned as*

$$A = \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix},$$

where A_{11} and A_{22} are square matrices with no eigenvalue in common. Then, $\forall z \in \mathbb{C} \setminus \Lambda_A$,

$$(zI - A)^{-1} = \begin{bmatrix} (zI - A_{11})^{-1} & (zI - A_{11})^{-1}V - V(zI - A_{22})^{-1} \\ & (zI - A_{22})^{-1} \end{bmatrix},$$

where V solves the Sylvester equation $A_{11}V - VA_{22} = A_{12}$.

Proof. Applying the block inverse formula we get

$$(zI - A)^{-1} = \begin{bmatrix} (zI - A_{11})^{-1} & (zI - A_{11})^{-1}A_{12}(zI - A_{22})^{-1} \\ & (zI - A_{22})^{-1} \end{bmatrix}.$$

Then, by imposing $(zI - A_{11})^{-1}V - V(zI - A_{22})^{-1} = (zI - A_{11})^{-1}A_{12}(zI - A_{22})^{-1}$ we get

$$A_{12} = V(zI - A_{22}) - (zI - A_{11})V \quad \iff \quad A_{12} = A_{11}V - VA_{22}. \qquad \square$$

We are now ready to state the result that is at the core of our recursion for evaluating bivariate matrix functions.

THEOREM 2. *Let $A \in \mathbb{C}^{m\times m}$ and $B \in \mathbb{C}^{n\times n}$ be triangular matrices block partitioned as*

$$A = \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix},$$

where $A_{11} \in \mathbb{C}^{(m-k_A)\times(m-k_A)}$ *and* $A_{22} \in \mathbb{C}^{k_A\times k_A}$ *have no eigenvalue in common and the same holds for* $B_{11} \in \mathbb{C}^{(n-k_B)\times(n-k_B)}$ *and* $B_{22} \in \mathbb{C}^{k_B\times k_B}$. *If* $f(x,y)$ *is a bivariate function analytic on* $\Lambda_A \times \Lambda_B$ *and* $C = \left[\begin{smallmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{smallmatrix}\right] \in \mathbb{C}^{m\times n}$ *is partitioned accordingly to* A *and* B, *then we have*

$$\begin{aligned} f\{A, B^T\}(C) = &\begin{bmatrix} I_{m-k_A} \\ 0 \end{bmatrix} f\{A_{11}, B_{11}^T\}\left(C_{11} + VC_{21}\right)\begin{bmatrix} I_{n-k_b} & W \end{bmatrix} \\ &+ \begin{bmatrix} -V \\ I_{k_A} \end{bmatrix} f\{A_{22}, B_{11}^T\}\left(C_{21}\right)\begin{bmatrix} I_{n-k_B} & W \end{bmatrix} \\ &+ \begin{bmatrix} I_{m-k_A} \\ 0 \end{bmatrix} f\{A_{11}, B_{22}^T\}\left(\begin{bmatrix} I_{m-k_A} & V \end{bmatrix} C \begin{bmatrix} -W \\ I_{k_B} \end{bmatrix}\right)\begin{bmatrix} 0 & I_{k_B} \end{bmatrix} \\ &+ \begin{bmatrix} -V \\ I_{k_A} \end{bmatrix} f\{A_{22}, B_{22}^T\}\left(C_{22} - C_{21}W\right)\begin{bmatrix} 0 & I_{k_B} \end{bmatrix}, \end{aligned}$$

where $V \in \mathbb{C}^{(m-k_A)\times k_A}$ *satisfies* $A_{11}V - VA_{22} = A_{12}$ *and* $W \in \mathbb{C}^{(n-k_B)\times k_B}$ *satisfies* $B_{11}W - WB_{22} = B_{12}$.

Proof. Let us indicate with $\mathfrak{L}_j(x) := (xI - A_{jj})^{-1}$ and $\mathfrak{R}_j(y) := (yI - B_{jj})^{-1}$, $j = 1, 2$, the resolvent functions associated with the diagonal blocks. By applying Lemma 1 we get $f\{A, B^T\}(C) = F$, where

$$\begin{aligned} F &= -\frac{1}{4\pi^2}\iint_\Gamma f(x,y)\begin{bmatrix} \mathfrak{L}_1(x) & \mathfrak{L}_1(x)V - V\mathfrak{L}_2(x) \\ & \mathfrak{L}_2(x) \end{bmatrix} C \begin{bmatrix} \mathfrak{R}_1(y) & \mathfrak{R}_1(y)W - W\mathfrak{R}_2(y) \\ & \mathfrak{R}_2(y) \end{bmatrix} dxdy \\ &= -\frac{1}{4\pi^2}\begin{bmatrix} I_{m-k_A} \\ 0 \end{bmatrix}\iint_\Gamma f(x,y)\mathfrak{L}_1(x)\left[C_{11} + VC_{21}\right]\mathfrak{R}_1(y)dx\,dy\begin{bmatrix} I_{n-k_b} & W \end{bmatrix} \\ &- \frac{1}{4\pi^2}\begin{bmatrix} -V \\ I_{k_A} \end{bmatrix}\iint_\Gamma f(x,y)\mathfrak{L}_2(x)C_{21}\mathfrak{R}_1(y)dx\,dy\begin{bmatrix} I_{n-k_B} & W \end{bmatrix} \\ &- \frac{1}{4\pi^2}\begin{bmatrix} I_{m-k_A} \\ 0 \end{bmatrix}\iint_\Gamma f(x,y)\mathfrak{L}_1(x)\begin{bmatrix} I_{m-k_A} & V \end{bmatrix} C \begin{bmatrix} -W \\ I_{k_B} \end{bmatrix}\mathfrak{R}_2(y)dx\,dy\begin{bmatrix} 0 & I_{k_B} \end{bmatrix} \\ &- \frac{1}{4\pi^2}\begin{bmatrix} -V \\ I_{k_A} \end{bmatrix}\iint_\Gamma f(x,y)\mathfrak{L}_2(x)\left(C_{22} - C_{21}W\right)\mathfrak{R}_2(y)dx\,dy\begin{bmatrix} 0 & I_{k_B} \end{bmatrix}. \end{aligned}$$

∎

In the 2×2 case we can leverage the previous result to state a generalization for the formula of univariate functions of 2×2 upper triangular matrices using divided differences. In the univariate case, we have [14, Theorem 4.11]

$$f\left(\begin{bmatrix} \lambda_1 & a_{12} \\ & \lambda_2 \end{bmatrix}\right) = \left(\begin{bmatrix} f(\lambda_1) & a_{12}D_x[\lambda_1, \lambda_2]f \\ & f(\lambda_2) \end{bmatrix}\right),$$

where D_x denotes the one dimensional divided difference

$$D_x[\lambda_1, \lambda_2]f = \begin{cases} \frac{f(\lambda_2) - f(\lambda_1)}{\lambda_2 - \lambda_1}, & \lambda_1 \neq \lambda_1, \\ f'(a_{11}), & \lambda_1 = \lambda_2. \end{cases}$$

We use the following definition of divided differences for bivariate functions:

$$D_x[x_1, x_2]f(x,y) := \frac{f(x_2,y) - f(x_1,y)}{x_2 - x_1}, \qquad D_y[y_1, y_2]f(x,y) := \frac{f(x,y_2) - f(x,y_1)}{y_2 - y_1}.$$

Note that $D_x[x_1, x_2]f(x,y)$ is a univariate function of y. Applying Theorem 2 yields the following formula that expresses $f\{A, B^T\}(C)$ in terms of $f(x,y)$ and its divided differences evaluated at all the possible pairs of eigenvalues of A and B.

COROLLARY 3. *Let*

$$A = \begin{bmatrix} \lambda_1 & a_{12} \\ & \lambda_2 \end{bmatrix}, \quad B = \begin{bmatrix} \mu_1 & b_{12} \\ & \mu_2 \end{bmatrix}, \quad C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix},$$

and $f(x, y)$ *such that* $f\{A, B^T\}(C)$ *is well defined. Then*

$$f\{A, B^T\}(C) = \begin{bmatrix} f(\lambda_1, \mu_1) & f(\lambda_1, \mu_2) \\ f(\lambda_2, \mu_1) & f(\lambda_2, \mu_2) \end{bmatrix} \circ C + \begin{bmatrix} c_{21} a_{12} D_x[\lambda_1, \lambda_2] f(x, \mu_1) & \Delta \\ & c_{21} b_{12} D_y[\mu_1, \mu_2] f(\lambda_2, y) \end{bmatrix},$$

where $\circ$ *denotes the Hadamard product, and*

$$\Delta := c_{22} a_{12} D_x[\lambda_1, \lambda_2] f(x, \mu_2) + c_{11} b_{12} D_y[\mu_1, \mu_2] f(\lambda_1, y) + c_{21} a_{12} b_{12} D_x[\lambda_1, \lambda_2] D_y[\mu_1, \mu_2] f(x, y).$$

Going back to the general framework, Theorem 2 splits the computation of $f\{A, B^T\}(C)$ into 4 bivariate functions of triangular coefficients with smaller sizes, the solution of 2 Sylvester equations, and some matrix multiplications and additions. Applying this procedure recursively reduces the problem to evaluate bivariate matrix functions on scalars or 2×2 triangular matrices via the formula in Corollary 3. In practice, it is convenient to stop the recursion at a larger block size in order to exploit BLAS 3 operations. We note that the Sylvester equations solved in the four branches generated by a recursion are pairwise identical since they only depend on A or B. Therefore, the most efficient implementation solves these equations before the recursive call. For readability, this is not done in Algorithm 4, but we discussed this step in further detail in section 2.3.

The implementation of this approach requires the availability of two additional procedures in the spirit of the univariate Schur–Parlett algorithm [9]:

- FUN2_ATOM evaluates the function for sufficiently small matrix arguments,
- BLOCKING produces the blocking pattern that ensures a sufficient separation between the spectra of the diagonal blocks; it returns the ordering permutation and the list of indices for each block $\mathcal{I}^A$ and $\mathcal{I}^B$, respectively.

In particular, FUN2_ATOM aims at computing $f\{A, B^T\}(C)$ for input arguments of size up to $n_{\min} \times n_{\min}$, where the choice of $n_{\min}$ depends on the conditioning of the problem or on the underlying computer architecture.

The recursion is constructed by repeatedly splitting the index partitionings $\mathcal{I}^A = \mathcal{I}^{A_1} \sqcup \mathcal{I}^{A_2}$ and $\mathcal{I}^B = \mathcal{I}^{B_1} \sqcup \mathcal{I}^{B_2}$ in two parts and applying Theorem 2 with $A_{ii} = A(\mathcal{I}^{A_i}, \mathcal{I}^{A_i})$ and $B_{jj} = B(\mathcal{I}^{B_j}, \mathcal{I}^{B_j})$. The purpose of BLOCKING is to ensure that the spectra of A_{11} and A_{22} (resp., B_{11} and B_{22}) are sufficiently separated at all steps of recursion; this is a necessary condition for solving accurately the Sylvester equations encountered in the process.

The detailed descriptions of FUN2M_ATOM and BLOCKING are postponed to the following sections. The algorithm obtained using this paradigm is reported in Algorithm 4. The pseudocode also makes use of the function SYLVESTER_TRI which solves Sylvester matrix equations with triangular coefficients; this can be done very efficiently, as described in [19]; in our code, we simply rely on the triangular Sylvester solver included in the LAPACK routine `*trsyl`.

Remark 4. We note that in Theorem 2 it is possible to only partition A or B instead of both matrices at once. This splits the problem into two subtasks. Formally,

this operation can be seen as a particular case of Theorem 2 where either A_{22} or B_{22} are empty matrices, and the associated terms in the expression of $f\{A, B^T\}(C)$ disappear.

Algorithm 4 Evaluates $f\{A, B^T\}(C)$

procedure FUN2M(f, A, B, C)
  if A and B are normal **then return** FUN2_DIAG(f, A, B, C)
  else if A is normal **then return** FUN2_DIAGA(f, A, B, C)
  else if B is normal **then return** FUN2_DIAGB(f, A, B, C)
  else
    $[Q_A, T_A] = \texttt{schur}(A)$
    $[Q_B, T_B] = \texttt{schur}(B)$
    $[P_A, \mathcal{I}^A] =$ BLOCKING(T_A)
    $[P_B, \mathcal{I}^B] =$ BLOCKING(T_B)
    $T_A \leftarrow P_A^* T_A P_A$, $T_B \leftarrow P_B^* T_B P_B$
    $\widetilde{C} \leftarrow P_A^* Q_A^* C Q_B P_B$
    $F \leftarrow$FUN2M_REC($f, T_A, T_B, \widetilde{C}, \mathcal{I}^A, \mathcal{I}^B$)
    return $Q_A P_A F P_B^* Q_B^*$
  end if
end procedure

procedure FUN2M_REC($f, A, B, C, \mathcal{I}^A, \mathcal{I}^B$)
  $\ell_A \leftarrow \texttt{length}(\mathcal{I}^A)$ ▷ $\mathcal{I}^A = \{I_1^A, \ldots, I_{\ell_A}^A\}$
  $\ell_B \leftarrow \texttt{length}(\mathcal{I}^B)$ ▷ $\mathcal{I}^B = \{I_1^B, \ldots, I_{\ell_B}^B\}$
  if ℓ_A or ℓ_B is zero **then return** [ ]
  else if ℓ_A and ℓ_B are both equal to 1 **then return** FUN2_ATOM(f, A, B, C)
  else
    Split $\mathcal{I}^A = \mathcal{I}^{A_1} \sqcup \mathcal{I}^{A_2}$ and $\mathcal{I}^B = \mathcal{I}^{B_1} \sqcup \mathcal{I}^{B_2}$ ▷ see section 2.3
    Partition A, B and C according to $\mathcal{I}^{A_1}, \mathcal{I}^{A_2}, \mathcal{I}^{B_1}, \mathcal{I}^{B_2}$:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix}, \qquad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

    $V \leftarrow$ SYLVESTER_TRI(A_{11}, A_{22}, A_{12}) ▷ V, W are precomputed; see section 2.3
    $W \leftarrow$ SYLVESTER_TRI(B_{11}, B_{22}, B_{12})
    $C_1 \leftarrow C_{11} + VC_{21}$, $C_2 \leftarrow C_{21}$
    $C_3 \leftarrow C_{12} - C_{11}W - VC_{21}W + VC_{22}$, $C_4 \leftarrow C_{22} - C_{21}W$
    $F_1 \leftarrow$ FUN2M_REC($f, A_{11}, B_{11}, C_1, \mathcal{I}^{A_1}, \mathcal{I}^{B_1}$)
    $F_2 \leftarrow$ FUN2M_REC($f, A_{22}, B_{11}, C_2, \mathcal{I}^{A_2}, \mathcal{I}^{B_1}$)
    $F_3 \leftarrow$ FUN2M_REC($f, A_{11}, B_{22}, C_3, \mathcal{I}^{A_1}, \mathcal{I}^{B_2}$)
    $F_4 \leftarrow$ FUN2M_REC($f, A_{22}, B_{22}, C_4, \mathcal{I}^{A_2}, \mathcal{I}^{B_2}$)
    return $\begin{bmatrix} F_1 - VF_2 & F_1W - VF_2W + F_3 - VF_4 \\ F_2 & F_2W + F_4 \end{bmatrix}$
  end if
end procedure

2.1. Block partitioning of the Schur forms. Algorithm 4 requires the solutions of two Sylvester equations at every recursive step. In order to avoid an excessive error propagation, we need to ensure a sufficient spectral separation between the coefficients A_{11}, A_{22}, or B_{11}, B_{22}. This is in complete analogy with the univariate Schur–Parlett algorithm, where only one matrix is involved. Hence, we rely on the same blocking procedure proposed in [9, Algorithm 4.1] that, chosen a parameter $\delta > 0$, returns two index partitionings

$$\mathcal{I}^A = \{I_1^A, \dots, I_{\ell_A}^A\}, \qquad \mathcal{I}^B = \{I_1^B, \dots, I_{\ell_B}^B\},$$

where $I_i^A \subseteq \{1, \dots, m\}$ and $I_j^B \subseteq \{1, \dots, n\}$, that identify diagonal blocks $A(I_i^A, I_i^A)$ and $B(I_j^B, I_j^B)$ with the following properties:

- Any block of size at least 2×2 is such that for each eigenvalue λ there exists another eigenvalue μ in the same block satisfying $|\lambda - \mu| \leq \delta$.
- Each pair of eigenvalues λ, μ that belong to different blocks in the same matrix (A or B) have distance at least $|\lambda - \mu| > \delta$.

The first property is useful to construct a polynomial approximant that is accurate on the spectrum of the block; for instance, a truncated Taylor expansion. We will use this fact in section 2.2.1, while this will not be relevant for the perturb-and-diagonalize approach in section 2.2.2.

In practice, BLOCKING interprets the eigenvalues as nodes in a graph, which are connected by an edge if their distance is less than δ; then, the blocking corresponds to identifying the connected components of this graph and to reordering the Schur form accordingly.

We remark that the condition $|\lambda - \mu| > \delta$ does not guarantee that the Sylvester equations solved in the recursion are well conditioned since their coefficients are nonnormal. Hence, we propose to verify this a posteriori and possibly cure the ill-conditioning by merging the blocks. This approach is described in detail in section 2.3; however, it might not be applicable when employing Taylor expansions for evaluating the function at the atomic blocks due to the potential loss of spectral clustering. The choice of δ may depend on the spectral properties of the matrices. In our experiments we noticed that $\delta = 0.1$ yields good results; however, a smaller value might be preferable if the blocking returns a partition with large blocks.

2.2. Evaluating the function at the atomic blocks. In this section we specify two implementations of FUN2_ATOM; the first is based on the evaluation of a truncated (bivariate) Taylor expansion and requires the availability of the partial derivatives of arbitrary orders; the second relies on the recent perturb-and-diagonalize approach developed in [16] which is derivative-free. In addition to the spectra separation for different blocks, the Taylor approach requires the blocking strategy to provide matrices with sufficiently clustered eigenvalues. For the perturb-and-diagonalize approach this is not necessary, and we choose the block size to be of the order of $n_{\min} = 4$ if this can be achieved along with the spectra separation condition.

Throughout this section, $\|\cdot\|$ denotes the spectral norm.

2.2.1. Bivariate Taylor expansion. Let us assume that A and B are triangular matrices with eigenvalues clustered around $\lambda = \mathrm{tr}(A)/m$ and $\mu = \mathrm{tr}(B)/n$, respectively; that is, $\Lambda_A \subset \mathcal{B}(\lambda, r_A) := \{|z - \lambda| < r_A\}$ and $\Lambda_B \subset \mathcal{B}(\mu, r_B) := \{|z - \mu| < r_B\}$ for $r_A, r_B > 0$.

We consider a truncated Taylor expansion of $f(x, y)$ centered at (λ, μ),

$$f(x, y) = \sum_{i+j \leq k} \frac{f^{(i,j)}(\lambda, \mu)}{i!j!} (x - \lambda)^i (y - \mu)^j + R_k(x, y),$$

which leads to the following approximation (see [22, section 2.2]):

$$f\{A, B^T\}(C) \approx \sum_{i+j \leq k} \frac{f^{(i,j)}(\lambda, \mu)}{i!j!} N_A^i C N_B^j, \tag{5}$$

where $A = \lambda I + N_A$ and $B = \mu I + N_B$. The value of k is chosen to ensure a small error in the approximation:

$$\left\| f\{A, B^T\}(C) - \sum_{i+j\le k} \frac{f^{(i,j)}(\lambda,\mu)}{i!j!} N_A^i C N_B^j \right\| = \|R_k\{A, B^T\}(C)\|. \tag{6}$$

The remainder in the above formula can be estimated using a straightforward generalization of [9, Theorem 2.5] to the bivariate case.

LEMMA 5. *The remainder of the approximation in* (6) *is bounded by*

$$\|R_k\{A, B^T\}(C)\| \le \max\{\|N_A\|, \|N_B\|\}^{k+1} \cdot \|C\| \cdot \max_{(\xi,\eta)\in\mathfrak{B}} \sum_{i+j=k+1} \frac{|f^{(i,j)}(\xi,\eta)|}{i!j!},$$

where $\mathfrak{B} = \mathcal{B}(\lambda, r_x) \times \mathcal{B}(\mu, r_y)$.

Proof. We write the remainder in Lagrange form as follows:

$$\begin{aligned} R_k(x,y) &= \sum_{i+j\ge k+1} \frac{f^{(i,j)}(\lambda,\mu)}{i!j!}(x-\lambda)^i(y-\mu)^j \\ &= \sum_{i+j=k+1} \frac{f^{(i,j)}(\xi,\eta)}{i!j!}(x-\lambda)^i(y-\mu)^j, \end{aligned}$$

where (ξ,η) belongs to the segment that connects (λ,μ) with (x,y). Evaluating $R_k(x,y)$ at A and B applied to C yields

$$\begin{aligned} \|R_k\{A, B^T\}(C)\| &\le \sum_{i+j=k+1} \frac{f^{(i,j)}(\xi,\eta)}{i!j!} \max\{\|N_A\|, \|N_B\|\}^{k+1}\|C\| \\ &\le \max\{\|N_A\|, \|N_B\|\}^{k+1} \cdot \|C\| \cdot \max_{(\xi,\eta)\in\mathfrak{B}} \sum_{i+j=k+1} \frac{|f^{(i,j)}(\xi,\eta)|}{i!j!}. \quad \square \end{aligned}$$

In order to compute an approximation of the form (5) that yields an accuracy ϵ, we propose the following scheme:

1. Compute $\theta := \max\{\|N_A\|, \|N_B\|\}$, and define $\mathfrak{B}_{m,n} := \Lambda_A \times \Lambda_B$.
2. Identify the minimal integer k such that
$$\theta^{k+1} \cdot \|C\| \cdot \sum_{i+j=k+1} \frac{|f^{(i,j)}(\lambda,\mu)|}{i!j!} \le \epsilon.$$
3. Verify that the chosen k satisfies also the following inequality:
$$\theta^{k+1} \cdot \|C\| \cdot \max_{(\xi,\eta)\in\mathfrak{B}_{m,n}} \sum_{i+j=k+1} \frac{|f^{(i,j)}(\xi,\eta)|}{i!j!} \le \epsilon.$$
If not, increase k, checking again the previous conditions, until both are satisfied.
4. Using the computed derivatives, evaluate (5).

Note that the replacing $\mathfrak{B}$ with $\mathfrak{B}_{m,n}$ does not guarantee the upper bound for the remainder of the Taylor expansion, although it is in general a good heuristic. The procedure sketched above is reported in Algorithm 5.

Algorithm 5 Computes $f\{A, B^T\}(C)$ for triangular A, B with a Taylor expansion

1: **procedure** FUN2_ATOM_TAYLOR(f, A, B, C, ϵ)
2: Retrieve λ and μ from the diagonals of A and B
3: $N_A \leftarrow A - \lambda I$, $N_B \leftarrow B - \mu I$
4: $\theta \leftarrow \max\{\|N_A\|, \|N_B\|\}$
5: **for** $k = 1, \dots, k_{\max}$ **do**
6: $R \leftarrow \theta^{k+1} \cdot \|C\| \cdot \sum_{i+j=k+1} \frac{|f^{(i,j)}(\lambda,\mu)|}{i!j!}$
7: **if** $R \le \epsilon$ **then**
8: $R_2 \leftarrow \theta^{k+1} \cdot \max_{(\xi,\eta)\in\mathfrak{B}_{m,n}} \sum_{i+j=k+1} \frac{|f^{(i,j)}(\xi,\eta)|}{i!j!}$
9: **if** $R_2 \le \epsilon$ **then**
10: **break**
11: **end if**
12: **end if**
13: **end for**
14: **return** $\sum_{i+j\le k} \frac{f^{(i,j)}(\lambda,\mu)}{i!j!} N_A^i C N_B^j$
15: **end procedure**

To conclude, we specify how the bivariate polynomial at line 14 is evaluated. Given any polynomial $P(x, y)$ of total degree k we write it as follows:

$$P\{A, B^T\}(C) = \sum_{i+j\le k} p_{ij} A^i C B^j = \sum_{i=0}^{k} A^i C \underbrace{\sum_{j=0}^{k-i} p_{ij} B^j}_{P_i(B)}.$$

Then, we evaluate $P_i(B)$ for $i = 0, \dots, k$ using the Horner scheme, and finally we compute $\sum_{i=0}^{k} A^i C P_i(B)$ using again the Horner scheme with respect to the variable A:

$$P\{A, B^T\}(C) = A(\dots A(ACP_k(B) + CP_{k-1}(B)) + CP_{k-2}(B) + \cdots) + CP_0(B).$$

This approach requires $k(k-1)/2$ multiplications between $n \times n$ matrices, k multiplications between $m \times n$ and $n \times n$ matrices, and finally k multiplications between $m \times m$ and $m \times n$ matrices. This yields the total cost of $\mathcal{O}(k^2n^3 + km^2n + kmn^2)$. If $n > m$, it is convenient to swap the role of A and B, relying on an analogous formula.

2.2.2. Perturb-and-diagonalize. A derivative-free approach for the evaluation of $f(A)$, when A is highly nonnormal, has been proposed in [8]. The idea is to introduce a small random perturbation E to the matrix A so that $A+E$ is diagonalizable with probability 1. Then, $f(A+E) \approx f(A)$ is evaluated by diagonalization. The method has been recently improved in [16] and has been proposed for evaluating the atomic blocks in the Schur–Parlett scheme. In particular, in [16] it is suggested to first compute the Schur form, introduce a diagonal perturbation, and evaluate the function of the perturbed Schur form using a higher precision determined by estimating the condition number of its eigenvector matrix.

We propose to rely on the analogue scheme in the bivariate case. More specifically, consider A, B upper triangular matrices and E_A, E_B small diagonal perturbations. Let

$$\widetilde{A} := A + E_A = V_A D_A V_A^{-1}, \qquad \widetilde{B} := B + E_B = V_B D_B V_B^{-1}$$

be the the eigendecompositions of the perturbed matrices. Thanks to the triangular structure of $A + E_A$ and $B + E_B$ the eigenvalues can be read off the diagonal so that D_A and D_B can be considered as not affected by rounding errors. The eigenvector matrices V_A, V_B are also triangular and are determined by solving triangular shifted linear systems with $\widetilde{A}, \widetilde{B}$. As noted in [16] this allows us to estimate $\kappa(V_A)$ and $\kappa(V_B)$ from the entries of $\widetilde{A}, \widetilde{B}$ using

$$\kappa(V_A) \lesssim m\zeta(\zeta+1)^{m-2}, \qquad \zeta := \frac{\max_{i<j} |\widetilde{A}_{ij}|}{\min_{i\neq j} |\widetilde{A}_{ii} - \widetilde{A}_{jj}|}, \tag{7}$$

as well as for $\kappa(V_B)$. We remark that (7) can be pessimistic for moderate values of m. As in [16] we apply the following heuristic:

(i) We further partition A with BLOCKING using $\delta_1 < \delta$; in our experiments we adopt $\delta_1 = 5 \cdot 10^{-3}$.
(ii) We estimate $\kappa(V_A)$ by the maximum of the quantities as in (7) computed for its diagonal blocks.

In practice, the latter heuristic might fail for highly nonnormal matrices; therefore we verify it a posteriori as we describe later in this section.

A classic result for univariate matrix functions bounds the forward error of computing $f(A)$ by diagonalization with a small constant multiplied by $\kappa(V)u$, where u is the current unit roundoff and V is the eigenvector matrix of A [14, page 82]. We generalize the latter within the following result.

LEMMA 6. *Let $F = V_A f\{D_A, D_B\}(V_A^{-1} C V_B) V_B^{-1}$, with $D_A = \operatorname{diag}(\lambda_1, \dots, \lambda_m)$, $D_B = \operatorname{diag}(\mu_1, \dots, \mu_n)$, and let $\hat{F}$ be the corresponding quantity computed in floating point arithmetic. If the matrix multiplications are performed exactly and $f(\lambda_i, \mu_j)$ is computed with relative error bounded by u_h, then*

$$\|F - \hat{F}\| \leq \kappa(V_A)\kappa(V_B)\|C\| \max_{i,j} |f(\lambda_i, \mu_j)| u_h.$$

Proof. Under the assumptions, $\hat{F}$ is equal to

$$V_A \left[(G + E) \circ (V_A^{-1} C V_B)\right] V_B^{-1}, \qquad G_{ij} := f(\lambda_i, \mu_j),$$

where $|E_{ij}| \leq \max_{i,j} |f(\lambda_i, \mu_j)| u_h$. Then,

$$\begin{aligned} \|F - \hat{F}\| &\leq \|V_A\| \|E \circ (V_A^{-1} C V_B)\| \|V_B^{-1}\| \\ &\leq \|V_A\| \max_{i,j} |f(\lambda_i, \mu_j)| \|V_A^{-1} C V_B\| \|V_B^{-1}\| u_h \\ &\leq \kappa(V_A)\kappa(V_B)\|C\| \max_{i,j} |f(\lambda_i, \mu_j)| u_h. \end{aligned}$$

∎

In view of Lemma 6 we choose u_h to ensure that $\|F - \hat{F}\| \leq \|F\| u$, where u is the current machine roundoff. By assuming $\|F\| \approx \max_{i,j} |f(\lambda_i, \mu_j)| \|C\|$, similarly to what is done in [16], this can be achieved by setting

$$u_h \leq \frac{\|F\| u}{\kappa(V_A)\kappa(V_B)\|C\| \max_{i,j} |f(\lambda_i, \mu_j)|} \approx \frac{u}{\kappa(V_A)\kappa(V_B)}. \tag{8}$$

In practice, the quantities $\kappa(V_A)$ and $\kappa(V_B)$ are estimated, before computing V_A and V_B, using the right-hand side of (7). Then, V_A and V_B are computed with a relative accuracy u_h as in (8) or better, as pointed out in the following remark.

Remark 7. Assuming that the matrix multiplications are performed exactly up to the current precision simplifies the analysis and is required also in [16]. This is not particularly restrictive since it can be guaranteed by computing matrix multiplications temporarily working with the lower unit roundoff $u_h \cdot \max\{\kappa(V_A), \kappa(V_B)\}^{-1}$. The same argument applies when computing V_A, V_B by solving shifted linear systems with $A,\ B$.

Often, relying on u_h as in (8), where $\kappa(V_A)$ and $\kappa(V_B)$ are approximated via (7), yields a pessimistic estimate for the necessary working precision. Hence, once the triangular eigenvector matrices V_A and V_B are available, we propose to refine the estimates of their condition numbers and adjust the precision in the evaluation of the function of the atomic blocks. For efficiency reasons, we would like to avoid computing $\kappa(V_A), \kappa(V_B)$ with high precision if possible. Hence we suggest this greedy strategy, which we describe for a generic $V \in \{V_A, V_B\}$:

- Convert V to standard floating point precision, and compute $\|V\|$ and $\|V^{-1}\|$; if $\|V\| \cdot \|V^{-1}\| \leq 10^{14}$, then return this value as a sufficiently accurate estimate for $\kappa(V)$.
- Otherwise construct the matrix U

$$U_{ij} = \begin{cases} |V_{ij}|, & i = j, \\ -|V_{ij}|, & i \neq j \end{cases}$$

  for which the inverse can be computed entrywise in standard precision and satisfies $\|U^{-1}\| \geq \|V^{-1}\|$ [13, section 8.2]. If $\|U^{-1}\| \leq 10^4 \|V^{-1}\|$ (that is, the guaranteed estimate on the number of digits is not much more pessimistic than the previous one), then use $\|V\|\|U^{-1}\|$ as upper bound for $\kappa(V)$.
- Finally, if none of the previous points succeed, then compute $\kappa(V)$ using a unit roundoff u_h.

To sum up, we propose to evaluate $f\{A, B^T\}(C)$ with A, B upper triangular following these steps:

1. Lower the unit roundoff to u^2, and perturb A and B with diagonal matrices of norm $\|A\|u$ and $\|B\|u$, respectively.
2. Determine u_h using (7) and (8), and if $u_h < u^2$, set the unit roundoff to u_h.
3. Compute V_A and V_B using a unit roundoff $\frac{u_h}{\max\{\kappa(V_A),\kappa(V_B)\}}$, where $\kappa(V_A)$ and $\kappa(V_B)$ are estimated with (7).
4. Refine the estimates for $\kappa(V_A)$ and $\kappa(V_B)$ with the greedy strategy described above, and recompute the unit roundoff u_h. If the new estimates $\kappa(V_A)$ and $\kappa(V_B)$ are larger than the previous ones, we adjust the precision accordingly, and we go back to 3, using these values instead of (7).
5. Run Algorithm 1 to evaluate $f\{A, B^T\}(C)$ using the new u_h.

The whole procedure is also summarized in Algorithm 6.

We remark that recomputing u_h ensures a significant performance gain when all the blocks are of small size, as it allows us to perform the calls to FUN2M_DIAG (which are $\mathcal{O}(mn)$) at a lower precision at the price of computing the condition numbers (which is only performed $\mathcal{O}(n+m)$ times).

2.2.3. Avoiding complex arithmetic. Whenever A, B, and C are real matrices and $f(x, y)$ has the property $\overline{f(x,y)} = f(\overline{x}, \overline{y})$ (and in particular $f(x, y)$ is real for real arguments) the bivariate matrix function $f\{A, B^T\}(C)$ is real as well. Indeed, we can select an integration path symmetric with respect to the real axis in definition (1) so that the imaginary part of the integral is guaranteed to vanish. Hence, it is

Algorithm 6 Computes $f\{A, B^T\}(C)$ for triangular A, B with a perturb-and-diagonalize approach

procedure FUN2_ATOM_DIAG(f, A, B, C)
  Set unit roundoff to u^2
  Generate random diagonal matrices E_A, E_B of norm $\|A\|u, \|B\|u$
  $\widetilde{A} \leftarrow A + E_A, \widetilde{B} \leftarrow B + E_B$
  Estimate $\kappa(V_A), \kappa(V_B)$ as in (7)
  $u_h \leftarrow u/(\kappa(V_A)\kappa(V_B))$
  Set the unit roundoff to $\min\{u^2, u_h\}$
  $[V_A, D_A] \leftarrow \text{EIG}(\widetilde{A})$, $[V_B, D_B] \leftarrow \text{EIG}(\widetilde{B})$ ▷ Can be precomputed; see section 2.3
  Refine the estimates of $\kappa(V_A), \kappa(V_B)$, and set the unit roundoff to the new u_h
  $F \leftarrow \text{FUN2M\_DIAG}(D_A, D_B, V_A^{-1} C V_B)$
  return $V_A F V_B^{-1}$
end procedure

appealing to use an evaluation procedure that preserves the real structure. To this end, we first reduce the matrices A and B to real Schur form so that the problem boils down to dealing with 2×2 blocks encoding complex conjugate eigenvalues. In fact, if the real structure is preserved by FUN2_ATOM, the recursion applied by FUN2M only requires solving Sylvester equations and matrix-matrix operations that do not introduce any complex arithmetic.

It is easy to see that the approach based on Taylor expansions preserves the real structure if the complex conjugate eigenvalues have small imaginary parts and thus can be put in the same block. Otherwise, for Taylor there is no straightforward alternative to working with complex arithmetic. In contrast, the perturb-and-diagonalize approach described in the previous section can be adapted to work directly with the real Schur form without particular assumptions. The random perturbations of the diagonal blocks are chosen as $\left[\begin{smallmatrix} \delta\alpha & \delta\beta \\ -\delta\beta & \delta\alpha \end{smallmatrix}\right]$ in order to match the structure of the Schur form. Then, the latter is block diagonalized, and $f\{A, B^T\}(C)$ is evaluated, where A and B are either 2×2 or 1×1. When one between A or B is a 1×1 block, the problem can be recast into the evaluation of either a scalar function or a univariate matrix function of a 2×2 block representing z and $\overline{z}$; in the latter case, the outcome can be expressed in terms of the block representing $g(z)$ and $\overline{g(z)}$, where $g(\cdot)$ is obtained by fixing the variable corresponding to the 1×1 block in $f(x, y)$.

The following result provides an explicit formula for the case where both A and B are 2×2 blocks.

THEOREM 8. *Let A, B, and C be 2×2 real matrices with A and B of the form*

$$A = \begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix}, \qquad B = \begin{bmatrix} \gamma & \delta \\ -\delta & \gamma \end{bmatrix},$$

If f is such that $f\{A, B^T\}(C)$ is well defined and $\overline{f(x, y)} = f(\overline{x}, \overline{y})$, then

$$f\{A, B^T\}(C) = \frac{1}{2} \begin{bmatrix} Q_1 + Q_2 & Q_3 + Q_4 \\ Q_3 - Q_4 & Q_1 - Q_2 \end{bmatrix},$$

where, denoting by $z = \alpha + \mathbf{i}\beta$ *and* $w = \gamma + \mathbf{i}\delta$*, we have*

$$
\begin{aligned}
Q_1 &:= (c_{21} - c_{12})\Im(f(z,w)) + (c_{11} + c_{22})\Re(f(z,w)),\\
Q_2 &:= (c_{12} - c_{21})\Im(f(z,\overline{w})) + (c_{11} - c_{22})\Re(f(z,\overline{w})),\\
Q_3 &:= (c_{22} - c_{11})\Im(f(z,\overline{w})) + (c_{12} + c_{21})\Re(f(z,\overline{w})),\\
Q_4 &:= (c_{11} + c_{22})\Im(f(z,w)) + (c_{12} - c_{21})\Re(f(z,w)).
\end{aligned}
$$

Proof. The matrices A and B are simultaneously diagonalized by means of the eigenvector matrix $\left[\begin{smallmatrix} 1 & 1 \\ \mathbf{i} & -\mathbf{i} \end{smallmatrix}\right]$. Then, applying formula (3) and exploiting that $f(z,w) = \overline{f(\overline{z},\overline{w})}$ and $f(z,\overline{w}) = \overline{f(\overline{z},w)}$ yield the claim. ∎

We remark that in this case one needs to adjust the blocking procedure to make sure that conjugate pairs are kept together. In practice, we perform the blocking by only looking at real parts of the eigenvalues and then use the perturb-and-diagonalize algorithm for the evaluation at the atomic blocks.

2.3. Splitting strategy and computational complexity. We have not specified yet the splitting strategy for the block index sets $\mathcal{I}^A$ and $\mathcal{I}^B$ returned by the BLOCKING procedure. Our code implements two different possibilities (which we call *balanced* and *single*) that we detail at the end of this section. Under minimal assumptions, any splitting strategy yields an algorithm with cubic cost in the sizes of A and B. From now on, we make the following assumption.

Assumption 2.1. The partitionings $\mathcal{I}^A$ and $\mathcal{I}^B$ are split in the same way in all branches of the recursion of Algorithm 4.

Assumption 2.1 implies that a given block in A or B is split in the same way in all branches of recursion. This allows us to look at A and B separately and precompute the solutions of all Sylvester equations before running the recursion in Algorithm 4. During this process, we also check the conditioning of the equations; if ill-conditioning is detected, we adjust the blocking, as described in the next subsection. Similarly, the eigendecompositions of the atomic blocks are precomputed when using FUN2M_ATOM_DIAG for the evaluations of the atomic blocks.

Note that this strategy identifies two trees describing the recursive partitioning of the index sets of A and B. We denote by d_A and d_B the depths of such trees.

2.3.1. Dealing with ill-conditioned Sylvester equations. The condition $|\lambda - \mu| > \delta$ obtained from the blocking strategy of section 2.1 does not necessarily guarantee that the Sylvester equations related with the diagonal blocks of A and B are well conditioned because their coefficients are not normal.

Nevertheless, it is in general a good heuristic, and we propose to check a posteriori whether the condition number is larger than expected by verifying the norm of the solution. More specifically, for a Sylvester equation $A_{11}V - VA_{22} = A_{12}$ we compute the ratio $r := \|V\|/\|A_{12}\|$; if $r > \gamma\delta^{-1}$, where γ is a moderate constant (in our case we set $\gamma = 10$), then we propose to discard the solution and consider the matrix $\left[\begin{smallmatrix} A_{11} & A_{12} \\ & A_{22} \end{smallmatrix}\right]$ as an atomic block.

However, this is not always viable because it could deteriorate the spectral clustering property of the blocks, making the method based on Taylor expansions not efficient. In contrast, the approach based on randomized diagonalization applies with no modifications, although high precision arithmetic has to be employed on a larger block, causing an increase in the computational cost.

In our implementation, the merging of the blocks is adopted only when relying on FUN2M_ATOM_DIAG; the potential accuracy loss of the Taylor approach without merging is visible in the first example in section 4.

2.3.2. Complexity. We now prove that, under Assumption 2.1, the cost of Algorithm 4 is $\mathcal{O}(m^3+n^3)$ independently on the splitting choice.

LEMMA 9. *Let $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times n}$, and $f\{A,B\}(C)$ be computed by means of* FUN2M *with a splitting strategy satisfying Assumption* 2.1. *If* FUN2_ATOM *applied with arguments of sizes $p\times p$ and $q\times q$ costs $\mathcal{O}(\max\{p,q\}pq)$, then* FUN2M *requires $\mathcal{O}(m^3+n^3)$ flops.*

Proof. We remark that the complexity of Algorithm 1 is dominated by the cost of the reduction to Schur forms of A and B (that requires $\mathcal{O}(m^3+n^3)$ flops), the calls to FUN2_ATOM, and the solution of the Sylvester equations.

Let us denote by $m_j = |I_j^A|$ and $n_j = |I_j^B|$ the sizes of the atomic blocks. Algorithm 4 calls FUN2_ATOM $\ell_A \cdot \ell_B$ times, and each call costs $O(\max\{m_i,n_j\}m_in_j)$, where ℓ_A and ℓ_B are the number of blocks in A and B, respectively. Then, the overall cost of these calls is

$$\begin{aligned}\sum_{i=1}^{\ell_A}\sum_{j=1}^{\ell_B}\max\{m_i,n_j\}m_in_j &\le \sum_{i=1}^{\ell_A}\sum_{j=1}^{\ell_B}(m_i+n_j)m_in_j\\ &= n\sum_{i=1}^{\ell_A}m_i^2 + m\sum_{j=1}^{\ell_B}n_j^2 \le m^2n+mn^2.\end{aligned}$$

The Sylvester equations are solved in a preprocessing step separately for A and B. We prove by induction on the depth d_A of the partitioning tree associated with A that the cost of solving all Sylvester equations is $\mathcal{O}(m^3)$. The result for B (that gives $\mathcal{O}(n^3)$) is analogous. When $d_A = 1$ there is no Sylvester equation to solve. When $d_A > 1$, let us suppose that the first splitting yields $\mathcal{I}^A = \mathcal{I}^{A_1} \sqcup \mathcal{I}^{A_2}$ with[1] $|\mathcal{I}^{A_1}| = m_1$ and $|\mathcal{I}^{A_2}| = m_2$. Then, we have to solve one Sylvester equation of size $m_1\times m_2$ and the Sylvester equations arising from the subtrees of depth d_A-1 associated with index sets of cardinality m_1, m_2, respectively. Solving the Sylvester equation costs $\mathcal{O}(m_1m_2\min\{m_1,m_2\})$; the induction step yields $\mathcal{O}(m_1^3)$ and $\mathcal{O}(m_2^3)$ for the subtrees. Summing these contributions we get $\mathcal{O}(m_1^3+m_2^3+m_1m_2\min\{m_1,m_2\}) \le O((m_1+m_2)^3) = \mathcal{O}(m^3)$. ∎

Although Lemma 9 ensures the same asymptotic complexity independently on the splitting strategy, different choices might be preferable based on the underlying computer architecture. We remark that the atomic blocks of the splitting procedure are determined by BLOCKING; any feasible partitioning tree has nodes given by the (ordered) union of such atomic index sets and the latter correspond to the leaf nodes. We describe two strategies for constructing a feasible tree:

- **Balanced:** Each node $\mathcal{I}_A$ is split as $\mathcal{I}^{A_1} \sqcup \mathcal{I}^{A_2}$ with subsets $\mathcal{I}^{A_i}$ of approximately the same cardinality.[1]
- **Single:** If $\mathcal{I}_A$ is composed by the atomic blocks $I_1^A,\dots,I_{\ell_A}^A$, then we consider the splitting $\mathcal{I}^{A_1} = \{I_1^A,\dots,I_{\ell_A-1}^A\}$, $\mathcal{I}^{A_2} = \{I_{\ell_A}^A\}$.

In the numerical experiments in section 4 we adopt the *balanced* approach. The *single* approach is used in section 3.2 to discuss the connection with the Bartels–Stewart

[1] By a slight abuse of notation, we write $|\mathcal{I}^{A_i}|$ to denote the sum of the cardinality of the index sets in $\mathcal{I}^{A_i}$ and analogously for $\mathcal{I}^{B_j}$.

algorithm. We remark that both choices satisfy Assumption 2.1, and hence provide a cubic algorithm.

2.4. Evaluating the function when one between A and B is small. We conclude with a discussion about the evaluation of $f\{A, B^T\}(C)$ when $m \gg n$ so that the outcome is a tall and thin matrix; the case $n \gg m$ is analogous. Similar considerations can be found also in [22]. In the case $n = 1$, the problem reduces to computing a univariate function of the triangular matrix A multiplied by the vector C. This can be done by relying on a Krylov method [11], and the cost depends on performing matrix-vector operations with A. When $n > 1$, we consider the following cases:

(i) The eigenvalues of B are clustered around y_0 so that we can find a low-degree univariate Taylor approximant of $f(x, y) \approx \sum_{j=0}^{k} \frac{\partial^j f(x,y_0)}{\partial y^j} \frac{(y-y_0)^j}{j!}$ centered at y_0. Then

$$f\{A, B^T\}(C) \approx \sum_{j=0}^{k} g_j(A) C (B - y_0 I)^j, \qquad g_j(x) := \frac{1}{j!} \frac{\partial^j f(x, y_0)}{\partial y^j},$$

and the problem is recast as computing univariate functions of A times (block) vectors and multiplications by (shifted) B.

(ii) If the eigenvalues are not clustered as in (i), then we foresee two options. The first one is to block partition B, as described in section 2.1, in order to retrieve the property on its atomic blocks. Finally, apply the same strategy as in Algorithm 4 blockwise. The second is to perturb and diagonalize A and then use the formula (4) to evaluate the univariate matrix functions at a higher precision in order to compensate for the condition number of the eigenvector matrix of A.

Note that, thanks to the triangular structure of A, rational Krylov subspace methods have the same asymptotic iteration cost of the standard polynomial Krylov method for the evaluation of $f(A)b$. Hence, unless a specific choice of shift parameters is known in advance (e.g., if a good rational approximant is known) the *extended Krylov method* might be a good choice.

3. Relation with other approaches. Algorithm 4 is closely related with other known approaches for evaluating functions of matrices. In this section, we point out some of these connections and differences.

3.1. Recursive block diagonalization. The presented algorithm may be alternatively described avoiding Theorem 2 as a recursive block diagonalization procedure, where the similarity transformations are kept implicit. Indeed, given block triangular matrices A and B, V and W as in Theorem 2, we have

$$\underbrace{\begin{bmatrix} I & V \\ & I \end{bmatrix}}_{\widetilde{V}} f\{A, B^T\}(C) \underbrace{\begin{bmatrix} I & -W \\ & I \end{bmatrix}}_{\widetilde{W}^{-1}} = f\left\{ \begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix}, \begin{bmatrix} B_{11}^T & \\ & B_{22}^T \end{bmatrix} \right\} (\widetilde{V} C \widetilde{W}^{-1}).$$

Multiplying on the left by $\widetilde{V}^{-1}$ and on the right by $\widetilde{W}$ and working out the relations on the blocks yield the same recursion obtained in Theorem 2.

Working with transformations of this type allows us to maintain the diagonal blocks, relying on the blocking procedure to have well-conditioned Sylvester equations.

3.2. Algorithm 4 and the Bartels–Stewart algorithm. In this section we will see that the celebrated Bartels–Stewart algorithm [4] for solving Sylvester equations is closely related to a particular case of Algorithm 4 applied to the function $f(x,y) = \frac{1}{x+y}$. We start by illustrating the relation between block diagonalization and the backsubstitution method for solving a triangular linear systems; then, we show that Algorithm 4 and the Bartels–Stewart algorithm verify a bivariate version of the latter relation.

Given a triangular matrix A, let us consider the linear system $Ax = b$ partitioned as

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

Applying the (block) backsubstitution procedure means to first compute $x_2 = A_{22}^{-1} b_2$ and then $x_1 = A_{11}^{-1}(b_1 - A_{12}x_2)$. On the other hand, we might compute $A^{-1}b$ by first applying the similarity transformation

$$\underbrace{\begin{bmatrix} I & V \\ & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} I & -V \\ & I \end{bmatrix}}_{= \left[\begin{smallmatrix} A_{11} & \\ & A_{22} \end{smallmatrix}\right]} \begin{bmatrix} x_1 + Vx_2 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 + Vb_2 \\ b_2 \end{bmatrix},$$

where $A_{11}V - VA_{22} = A_{12}$. This approach would result in the following steps:

1. compute $x_2 = A_{22}^{-1} b_2$,
2. compute $\widetilde{x}_1 = A_{11}^{-1}(b_1 + Vb_2)$,
3. compute $x_1 = \widetilde{x}_1 - Vx_2$.

In particular, both procedures solve two triangular systems whose coefficient matrices are the diagonal blocks of A. However, the one based on block diagonalization needs corrections that require the solution of $A_{11}V - VA_{22} = A_{12}$. In particular, the block diagonalization requires a cubic cost, whereas backsubstitution is quadratic.

Let us recall the procedure by Bartels and Stewart by using the notation introduced in section 2. Given the Sylvester equation $AX + XB = C$ with A and B upper triangular (possibly after the computation of the Schur forms) we consider the partitioning

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} + \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}, \tag{9}$$

where A_{22} and B_{11} are scalars. The Bartels–Stewart algorithm retrieves the blocks X_{ij} as follows:

1. solve the scalar equation associated to the $(2,1)$ block, $X_{21} = \frac{C_{21}}{A_{22}+B_{11}}$,
2. solve the triangular linear system $(A_{11} + B_{11}I)X_{11} = C_{11} - A_{12}X_{21}$,
3. solve the triangular linear system $X_{22}(B_{22} + A_{22}I) = C_{22} - X_{21}B_{12}$,
4. recursively solve the Sylvester equation

$$A_{11}X_{12} + X_{12}B_{22} = C_{12} - A_{12}X_{22} - X_{12}B_{12}.$$

We now analyze the relation between the previous steps and the four quantities in Theorem 2, applied with $k_A = 1$ and $k_B = n - 1$. Given $f(x,y) = \frac{1}{x+y}$, we remark that when at least one of the arguments of the bivariate matrix function $f\{A, B\}$ is a scalar the associated operator is the resolvent of a scalar equation or a linear system. In our setting we have

$$f\{A_{22}, B_{11}^T\}(C_{21}) = \frac{C_{21}}{A_{22} + B_{11}},$$

which is equivalent to step 1 of Bartels–Stewart and

$$f\{A_{11}, B_{11}^T\}(C_{11} + VC_{21}) = (A_{11} + B_{11}I)^{-1}(C_{11} + VC_{21}),$$
$$f\{A_{22}, B_{22}^T\}(C_{22} - C_{21}W) = (C_{22} - C_{21}W)(B_{22} + A_{22}I)^{-1},$$

where the column and row vectors V and W are given by

$$V = (A_{11} - A_{22}I)^{-1}A_{12},$$
$$W = B_{12}(B_{11}I - B_{22})^{-1}.$$

Then, X_{11} and X_{22} are computed by

$$X_{11} = f\{A_{11}, B_{11}^T\}(C_{11}+VC_{21})-VX_{21}, \quad X_{22} = f\{A_{22}, B_{22}^T\}(C_{22}-C_{21}W)-X_{21}W.$$

Finally, the X_{12} is computed recursively by

$$X_{12} = f\{A_{11}, B_{22}^T\}\left(\begin{bmatrix} I & V \end{bmatrix} C \begin{bmatrix} -W \\ I \end{bmatrix}\right) + (X_{11} + VX_{21})W - VX_{21}W - V(X_{22} + X_{21}W),$$

and X_{12} is computed by removing the three rank one corrections. In contrast with the univariate case, in this case both approaches have the same asymptotic complexity, even though Bartels–Stewart is more efficient since it does not need to apply the corrective terms.

However, this further optimization is viable only because of the special features of $f(x, y) = \frac{1}{x+y}$. Indeed, one can verify that

$$f\{A_{11}, B_{11}^T\}(C_{11} + VC_{21}) + VX_{21} = f\{A_{11}, B_{11}^T\}(C_{11} - A_{12}X_{21})$$

only holds for $f(x, y) = \frac{1}{x+y}$ and similarly for the relations for X_{22} and X_{12}. These are the key properties that allow us to avoid computing the terms V and W explicitly in the Bartels–Stewart algorithm.

4. Numerical results. In this section we test the performances of FUN2M and of the various choices that can be made in its implementation for computing $f\{A, B^T\}(C)$. We note that the choice of the matrix C does not affect the behavior of FUN2M. Everywhere, we set C equal to a *randomly generated complex matrix*; the latter indicates that both real and imaginary parts have $N(0, 1)$-distributed entries throughout this section. For simplicity we also assume $m = n$ in all our tests. The value of δ is set to 0.05. Concerning the choice of A and B we introduce the following test cases:

- `rand-eig`: Both A and B are of the form VDV^{-1}, where D is a randomly generated diagonal matrix whose entries have a real part uniformly distributed on $[1, 2]$ and Gaussian distributed imaginary parts; the matrix V is a randomly generated complex matrix with both real and imaginary parts of its entries Gaussian distributed.
- `randn`: Both A and B are randomly generated complex matrices.
- `jordbloc`: Both A and B are of the form QJQ^*, where Q is a randomly generated unitary matrix (obtained by means of the QR factorization of a randomly generated complex matrix) and J is the direct sum of a 8×8 Jordan block with eigenvalue 0.1 and a randomly generated complex matrix of size $n - 8$ shifted by the identity; B is generated analogously.

- `grcar`: A and B are equal to the `grcar` matrix of the MATLAB gallery.
- `smoke`: A and B are equal to the Schur form of the `smoke` matrix of the MATLAB gallery.
- `kahan`: A and B are equal to the `kahan` matrix of the MATLAB gallery.
- `lesp`: The matrices $-A$ and $-B$ are equal to the direct sum of the Schur form of the `lesp` matrix of the MATLAB gallery of dimension 32 with a randomly generated matrix of size $n-32$. The latter is obtained by generating a complex random matrix, rescaling it to have unit spectral norm, and subtracting the identity.
- `sampling`: The matrices A and B are equal to the direct sum of the `sampling` matrix of the MATLAB gallery of dimension 32 with a randomly generated matrix of size $n-32$. The latter is obtained by generating a complex random matrix, rescaling it to have unit spectral norm, and adding the identity.
- `grcar-rand`: A is equal to the `grcar` matrix of the MATLAB gallery, and B is as in `rand-eig`.

When experimenting with FUN2M we indicate in brackets the method used for evaluating the atomic blocks, i.e., FUN2_ATOM_DIAG or FUN2_ATOM_TAYLOR. The other considered computational approaches are labeled as follows:

- DIAG: The evaluation of $f\{A, B^T\}(C)$ is performed diagonalizing A and B in floating point arithmetic, regardless of the conditioning of the eigenvector matrices.
- DIAG_HP: The evaluation of $f\{A, B^T\}(C)$ is performed diagonalizing A and B in high precision, estimating the required digits as in FUN2_ATOM_DIAG.

We expect the first method to be fast with no guarantee on its accuracy. In contrast, the second approach is the most accurate although it can be significantly more expensive than both DIAG and FUN2M. In the tables, the columns labeled as n_A and n_B denote the number of atomic blocks in A and B, respectively. The label "Digits" refers to the maximum number of digits used in the multiprecision computation of the functions of the atomic blocks. The column "Max deg" contains the maximum of the degrees of the Taylor expansions used for the atomic blocks. Residual errors in the spectral norm are evaluated with respect to a benchmark quantity computed as in DIAG_HP, where the number of digits is fixed to 128. The latter value is in all cases much higher than the number of digits employed by FUN2M and DIAG_HP. Finally, for each example we provide a very rough estimate κ_f of the condition number of evaluating $f\{A, B^T\}(C)$. The latter is defined as

$$\lim_{h\to 0} \sup_{\frac{\|\Delta A\|}{\|A\|}, \frac{\|\Delta B\|}{\|B\|} \le h} \frac{\|f\{A+\Delta A, B^T+\Delta B^T\}(C) - f\{A, B^T\}(C)\|}{h}.$$

We compute κ_f by evaluating the above fraction in higher precision (128 digits) for $h = 10^{-32}$ and randomly generated complex matrices ΔA and ΔB scaled to have norm $h\|A\|$ and $h\|B\|$, respectively. This estimate is quite rough in general, but it yields a guaranteed lower bound and usually captures the order of magnitude of the condition number, which is sufficient to assess the accuracy of our results. In the tables we report $\kappa_f \cdot u$, where u is the unit roundoff in double precision, which gives an indication of the accuracy attainable by a backward stable method.

The algorithms have been implemented in a Julia package named `BivMatFun` and are available at https://github.com/numpi/BivMatFun. The implementation of FUN2M may be further optimized relying on the recursive Sylvester triangular solver `recsy` [20] and the BLAS 3 reordering of the Schur form in [21]. For simplicity we have

used what is available in LAPACK through the interfaces in Julia. The experiments have been run using Julia 1.5.3 on a dual CPU server with two Intel(R) Xeon(R) CPU E5-2643 version4 CPUs running at 3.40 GHz and 240 GB of RAM.

4.1. Perturbation and diagonalization versus Taylor expansion. In the first numerical test we compare the two proposed implementation for the functions of the atomic blocks, i.e., FUN2_ATOM_TAYLOR and FUN2_ATOM_DIAG. We have tested two cases: `rand-eig` and `grcar-rand`. In the first, both eigenvector matrices are sufficiently well conditioned, and the accuracy achieved by the methods is similar. Moreover the blocking procedure allows us to form atomic blocks of small sizes so that the cost of employing high precision arithmetic does not impact it at all. Indeed, the timings are in favor of FUN2_ATOM_DIAG. In the second test case, the eigenvector matrix of A is severely ill-conditioned, and this is reflected in the magnitude of the solutions of the Sylvester equations computed in Algorithm 4. When using FUN2_ATOM_DIAG this issue is circumvented by merging the blocks of A into a single one at the price of an increased computational cost caused by the use of higher precision arithmetic on larger matrices. This procedure cannot be applied by FUN2_ATOM_TAYLOR because this would cause a lack of the convergence for the Taylor expansion of $\frac{1}{\sqrt{x+y}}$. Hence, the timings of the Taylor-based approach are similar, but the outcome is not reliable. These remarks are confirmed by the results reported in Figure 1.

Finally, we mention that when estimating the eigenvector condition number for `grcar` and $n = 160$, the heuristic estimate based on (7) fails, and our procedure detects this with the a posteriori check and repeats the computation of the eigenvector with the appropriate accuracy.

4.2. Highly nonnormal A and B. Here, we consider seven test cases of fixed size $n = 64$ that involve both A and B with ill-conditioned eigenvector matrices. We compare the performances of FUN2M with DIAG and DIAG_HP on four different bivariate functions: $\sqrt{x+y}$, $\frac{1}{\sqrt{x+y}}$, $\frac{\exp(x+y)}{x+y}$, $\exp(\sqrt{x+y})$. In view of the considerations made in the previous experiment we only rely on FUN2_ATOM_DIAG for evaluating

Test = `randn-shift`, $f(x,y) = \frac{1}{\sqrt{x+y}}$

	FUN2M (FUN2_ATOM_DIAG)					FUN2M (FUN2_ATOM_TAYLOR)					
Size	Err	Time	nA	nB	Digits	Err	Time	nA	nB	Max deg	$\kappa_f \cdot u$
32	$4.1 \cdot 10^{-15}$	0.01	8	8	18	$9.2 \cdot 10^{-15}$	0.01	30	32	7	$9.2 \cdot 10^{-16}$
64	$9.2 \cdot 10^{-15}$	0.02	18	17	18	$9.7 \cdot 10^{-15}$	0.08	57	62	9	$1.2 \cdot 10^{-14}$
96	$4.0 \cdot 10^{-13}$	0.07	32	32	19	$4.0 \cdot 10^{-13}$	0.22	88	85	10	$1.2 \cdot 10^{-12}$
128	$5.7 \cdot 10^{-13}$	0.13	37	36	19	$5.7 \cdot 10^{-13}$	0.43	106	112	11	$2.6 \cdot 10^{-13}$
160	$9.9 \cdot 10^{-15}$	0.2	59	60	18	$1.9 \cdot 10^{-14}$	0.61	129	140	11	$4.5 \cdot 10^{-15}$

Test = `grcar-rand`, $f(x,y) = \frac{1}{\sqrt{x+y}}$

	FUN2M (FUN2_ATOM_DIAG)					FUN2M (FUN2_ATOM_TAYLOR)					
Size	Err	Time	nA	nB	Digits	Err	Time	nA	nB	Max deg	$\kappa_f \cdot u$
32	$3.0 \cdot 10^{-15}$	0.09	1	8	22	$9.3 \cdot 10^{-12}$	0.01	32	32	0	$3.9 \cdot 10^{-16}$
64	$4.4 \cdot 10^{-15}$	0.57	1	17	28	$3.6 \cdot 10^{-4}$	0.08	56	62	9	$1.0 \cdot 10^{-15}$
96	$7.0 \cdot 10^{-15}$	1.67	1	32	35	$3.2 \cdot 10^{5}$	0.22	68	85	13	$2.5 \cdot 10^{-15}$
128	$9.7 \cdot 10^{-15}$	3.83	1	36	36	$5.7 \cdot 10^{9}$	0.41	74	112	16	$4.8 \cdot 10^{-15}$
160	$8.1 \cdot 10^{-15}$	6.93	1	60	38	$1.7 \cdot 10^{28}$	0.65	61	140	22	$1.6 \cdot 10^{-15}$

FIG. 1. *Performances of the diagonalize-and-perturb and of the bivariate Taylor approximation on well-conditioned and ill-conditioned test cases.*

$f(x,y) = \sqrt{x+y}$, Size $= 64$

Test	FUN2M (FUN2_ATOM_DIAG) Err	Time	nA	nB	Digits	DIAG Err	Time	DIAG_HP Time	Err	Digits	$\kappa_f \cdot u$
jordbloc	$7.9 \cdot 10^{-10}$	0.02	15	15	48	$2.0 \cdot 10^{0}$	0.008	1.64	$7.9 \cdot 10^{-10}$	50	$1.1 \cdot 10^{-10}$
grcar	$1.1 \cdot 10^{-13}$	1.5	1	1	40	$4.5 \cdot 10^{-7}$	0.007	1.49	$1.1 \cdot 10^{-13}$	40	$3.4 \cdot 10^{-10}$
smoke	$8.2 \cdot 10^{-14}$	1.53	1	1	35	$1.3 \cdot 10^{-8}$	0.001	1.42	$6.5 \cdot 10^{-14}$	35	$2.3 \cdot 10^{-5}$
kahan	$2.5 \cdot 10^{-16}$	1.35	1	1	43	$4.0 \cdot 10^{-4}$	0.001	1.38	$6.6 \cdot 10^{-16}$	43	$2.4 \cdot 10^{-8}$
lesp	$2.6 \cdot 10^{-15}$	0.23	9	9	35	$1.4 \cdot 10^{0}$	0.003	1.29	$2.3 \cdot 10^{-15}$	36	$2.6 \cdot 10^{-16}$
sampling	$4.5 \cdot 10^{-8}$	1.45	1	9	49	$3.5 \cdot 10^{-1}$	0.005	2.17	$4.5 \cdot 10^{-8}$	49	$9.7 \cdot 10^{-9}$
grcar-rand	$1.6 \cdot 10^{-12}$	0.4	1	16	29	$1.2 \cdot 10^{-7}$	0.008	1.48	$1.6 \cdot 10^{-12}$	31	$5.4 \cdot 10^{-7}$

$f(x,y) = \frac{1}{\sqrt{x+y}}$, Size $= 64$

Test	FUN2M (FUN2_ATOM_DIAG) Err	Time	nA	nB	Digits	DIAG Err	Time	DIAG_HP Time	Err	Digits	$\kappa_f \cdot u$
jordbloc	$2.0 \cdot 10^{-9}$	0.02	15	15	48	$3.9 \cdot 10^{-1}$	0.008	1.65	$2.0 \cdot 10^{-9}$	51	$3.2 \cdot 10^{-10}$
grcar	$1.5 \cdot 10^{-13}$	1.53	1	1	40	$7.7 \cdot 10^{-8}$	0.008	1.54	$1.5 \cdot 10^{-13}$	40	$1.0 \cdot 10^{-9}$
smoke	$3.5 \cdot 10^{-9}$	1.46	1	1	35	$1.1 \cdot 10^{-8}$	0.002	1.45	$1.8 \cdot 10^{-9}$	35	$5.0 \cdot 10^{-1}$
kahan	$3.4 \cdot 10^{-16}$	1.37	1	1	43	$6.8 \cdot 10^{-7}$	0.002	1.36	$4.5 \cdot 10^{-16}$	43	$1.4 \cdot 10^{-7}$
lesp	$4.4 \cdot 10^{-15}$	0.23	9	9	35	$1.6 \cdot 10^{-1}$	0.003	1.32	$3.5 \cdot 10^{-15}$	36	$1.9 \cdot 10^{-15}$
sampling	$1.0 \cdot 10^{-7}$	0.41	10	9	49	$2.2 \cdot 10^{-2}$	0.006	2.04	$1.0 \cdot 10^{-7}$	49	$8.2 \cdot 10^{-8}$
grcar-rand	$5.2 \cdot 10^{-12}$	0.39	1	16	29	$7.8 \cdot 10^{-8}$	0.009	1.45	$5.2 \cdot 10^{-12}$	31	$3.7 \cdot 10^{-6}$

$f(x,y) = \frac{\exp(x+y)}{x+y}$, Size $= 64$

Test	FUN2M (FUN2_ATOM_DIAG) Err	Time	nA	nB	Digits	DIAG Err	Time	DIAG_HP Time	Err	Digits	$\kappa_f \cdot u$
jordbloc	$1.2 \cdot 10^{-14}$	0.02	17	15	48	$2.5 \cdot 10^{9}$	0.008	1.67	$8.9 \cdot 10^{-15}$	50	$2.3 \cdot 10^{-16}$
grcar	$7.9 \cdot 10^{-15}$	1.53	1	1	40	$6.9 \cdot 10^{2}$	0.008	1.53	$7.9 \cdot 10^{-15}$	40	$3.9 \cdot 10^{-16}$
smoke	$4.9 \cdot 10^{-17}$	1.49	1	1	35	$8.4 \cdot 10^{-1}$	0.002	1.48	$4.9 \cdot 10^{-17}$	35	$2.3 \cdot 10^{-16}$
kahan	$4.7 \cdot 10^{-17}$	1.34	1	1	43	$2.2 \cdot 10^{7}$	0.001	1.35	$4.7 \cdot 10^{-17}$	43	$1.7 \cdot 10^{-16}$
lesp	$4.5 \cdot 10^{-17}$	0.23	9	10	35	$2.4 \cdot 10^{-13}$	0.003	1.33	$4.5 \cdot 10^{-17}$	36	$2.3 \cdot 10^{-13}$
sampling	$1.6 \cdot 10^{-8}$	1.4	1	9	49	$4.9 \cdot 10^{-2}$	0.006	2.09	$1.6 \cdot 10^{-8}$	49	$1.3 \cdot 10^{-9}$
grcar-rand	$1.8 \cdot 10^{-14}$	0.36	1	16	29	$4.8 \cdot 10^{-7}$	0.009	1.5	$1.8 \cdot 10^{-14}$	31	$7.0 \cdot 10^{-16}$

$f(x,y) = \exp(\sqrt{x+y})$, Size $= 64$

Test	FUN2M (FUN2_ATOM_DIAG) Err	Time	nA	nB	Digits	DIAG Err	Time	DIAG_HP Time	Err	Digits	$\kappa_f \cdot u$
jordbloc	$2.0 \cdot 10^{-10}$	0.02	15	15	48	$7.6 \cdot 10^{0}$	0.009	1.77	$2.0 \cdot 10^{-10}$	51	$3.9 \cdot 10^{-10}$
grcar	$1.1 \cdot 10^{-13}$	1.61	1	1	40	$1.5 \cdot 10^{-6}$	0.008	1.6	$1.1 \cdot 10^{-13}$	40	$6.9 \cdot 10^{-10}$
smoke	$1.7 \cdot 10^{-13}$	1.56	1	1	35	$2.1 \cdot 10^{-8}$	0.002	1.54	$1.1 \cdot 10^{-13}$	35	$4.0 \cdot 10^{-5}$
kahan	$1.4 \cdot 10^{-14}$	1.43	1	1	43	$1.0 \cdot 10^{-2}$	0.001	1.36	$1.3 \cdot 10^{-14}$	43	$5.5 \cdot 10^{-7}$
lesp	$2.5 \cdot 10^{-16}$	0.23	9	9	35	$4.1 \cdot 10^{-1}$	0.003	1.4	$2.4 \cdot 10^{-16}$	36	$2.3 \cdot 10^{-16}$
sampling	$5.0 \cdot 10^{-8}$	2.19	1	1	49	$3.8 \cdot 10^{2}$	0.006	2.17	$5.0 \cdot 10^{-8}$	49	$1.0 \cdot 10^{-8}$
grcar-rand	$6.3 \cdot 10^{-13}$	0.4	1	16	29	$8.5 \cdot 10^{-8}$	0.009	1.57	$6.3 \cdot 10^{-13}$	31	$1.6 \cdot 10^{-5}$

FIG. 2. *Numerical results on highly nonnormal matrices of sizes* $n = m = 64$.

the atomic blocks in FUN2M. The results reported in Figure 2 confirm that DIAG is the fastest and least reliable method. On the other hand, FUN2M and DIAG_HP are equally accurate with FUN2M outperforming DIAG_HP apart from the cases where the partitioning is trivial—$n_A = n_B = 1$—where the two algorithms coincide. We note that in many cases the residual obtained by FUN2M and DIAG_HP is significantly below the estimate given by $\kappa_f \cdot u$. This is motivated by the fact that the matrices are (partially) upper triangular and the condition number with respect to perturbations sharing the same sparsity pattern is smaller.

4.3. Asymptotic cost. In this final example we test the computational cost of FUN2M on well-conditioned test cases where the number of atomic blocks in A and B grows linearly with the size n. More specifically, we consider the test case `randn` with exponentially increasing sizes 2^j for $j = 6, \ldots, 12$, and we measure the computational

Test = `randn`, $f(x,y) = \frac{1}{\sqrt{x+y}(x-y)}$, Size= 64

Size	FUN2M (FUN2_ATOM_DIAG) Time	nA	nB	DIAG Time
64	0.01	16	16	0.01
128	0.08	32	32	0.06
256	0.27	64	64	0.2
512	1.42	128	128	1.07
1,024	5.47	256	256	4.09
2,048	29.12	512	512	22.01
4,096	243.41	1,024	1,024	132.48

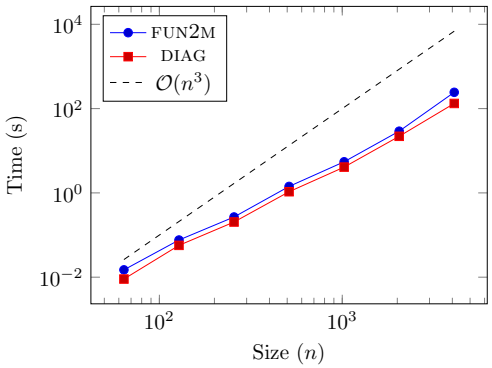

FIG. 3. *Timings of* FUN2M *and* DIAG *for well-conditioned* A *and* B.

time. The performances are compared with the ones of DIAG in Figure 3. The methods have comparable costs with DIAG being faster. Both approaches scale quadratically up to dimension 2048, and we start to see the expected cubic growth only on the last test. We mention that the measured accuracies are comparable, and since this is a well-conditioned case we refrain from reporting them.

5. Conclusions. We have proposed a novel block diagonalization approach for the evaluation of bivariate matrix functions. By relying on the synergy of multiprecision and a blocking strategy analogous to the one used in the Schur–Parlett scheme for univariate functions, the method guarantees backward stable results. We have validated the stability properties by testing the algorithm on a wide range of ill-conditioned cases. The asymptotic complexity is $\mathcal{O}(m^3 + n^3)$, where m and n correspond to the size of the two square matrix arguments, independently on the blocking strategy applied. In the ideal situation of well-conditioned eigenvector matrices the performances are comparable to evaluating the function by diagonalization. The algorithm extends naturally to the multivariate case although the number of atomic blocks grows exponentially with the number of variables.

REFERENCES

[1] A. H. AL-MOHY AND N. J. HIGHAM, *Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation*, SIAM J. Matrix Anal. Appl., 30 (2009), pp. 1639–1657.

[2] A. H. AL-MOHY, N. J. HIGHAM, AND S. D. RELTON, *Computing the Fréchet derivative of the matrix logarithm and estimating the condition number*, SIAM J. Sci. Comput., 35 (2013), pp. C394–C410.

[3] A. C. ANTOULAS, *Approximation of Large-Scale Dynamical Systems*, SIAM, Philadelphia, 2005.

[4] R. H. BARTELS AND G. W. STEWART, *Solution of the matrix equation* $AX + XB = C$[F4], Commun. ACM, 15 (1972), pp. 820–826.

[5] M. BENZI AND V. SIMONCINI, *Approximation of functions of large matrices with Kronecker structure*, Numer. Math., 135 (2017), pp. 1–26.

[6] M. CROUZEIX AND D. KRESSNER, *A Bivariate Extension of the Crouzeix-Palencia Result with an Application to Fréchet Derivatives of Matrix Functions*, preprint, arXiv:2007.09784, 2020.

[7] M. CROUZEIX AND C. PALENCIA, *The numerical range is a* $(1+\sqrt{2})$*-spectral set*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 649–655.

[8] E. B. DAVIES, *Approximate diagonalization*, SIAM J. Matrix Anal. Appl., 29 (2008), pp. 1051–1064.

[9] P. I. Davies and N. J. Higham, *A Schur-Parlett algorithm for computing matrix functions*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 464–485.
[10] E. Estrada and D. J. Higham, *Network properties revealed through matrix functions*, SIAM Rev., 52 (2010), pp. 696–714.
[11] S. Güttel, *Rational Krylov Methods for Operator Functions*, Ph.D. thesis, Technische Universität Bergakademie Freiberg, 2010.
[12] N. Hale, N. J. Higham, and L. N. Trefethen, *Computing A^α, $\log(A)$, and related matrix functions by contour integrals*, SIAM J. Numer. Anal., 46 (2008), pp. 2505–2523.
[13] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 2002.
[14] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, 2008.
[15] N. J. Higham, *The scaling and squaring method for the matrix exponential revisited*, SIAM Rev., 51 (2009), pp. 747–764.
[16] N. J. Higham and X. Liu, *A multiprecision derivative-free Schur–Parlett algorithm for computing matrix functions*, SIAM J. Matrix Anal. Appl., to appear.
[17] A. Hjørungnes, *Complex-Valued Matrix Derivatives: With Applications in Signal Processing and Communications*, Cambridge University Press, Cambridge, UK, 2011.
[18] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numer., 19 (2010), pp. 209–286.
[19] I. Jonsson and B. Kågström, *Recursive blocked algorithms for solving triangular systems–Part* I*: One-sided and coupled Sylvester-type matrix equations*, ACM Trans. Math. Software, 28 (2002), pp. 392–415.
[20] I. Jonsson and B. Kågström, *RECSY–a high performance library for sylvester-type matrix equations*, in Eur-Par 2003 Parallel Processing, Springer, Berlin, 2003, pp. 810–819.
[21] D. Kressner, *Block algorithms for reordering standard and generalized Schur forms*, ACM Trans. Math. Software, 32 (2006), pp. 521–532.
[22] D. Kressner, *Bivariate matrix functions*, Oper. Matrices, 8 (2014), pp. 449–466.
[23] D. Kressner, *A Krylov subspace method for the approximation of bivariate matrix functions*, in Structured Matrices in Numerical Linear Algebra, Springer, Cham, 2019, pp. 197–214.
[24] S. Massei and L. Robol, *Rational Krylov for Stieltjes matrix functions: Convergence and pole selection*, BIT, (2020), pp. 1–37.
[25] V. Simoncini, *Computational methods for linear matrix equations*, SIAM Rev., 58 (2016), pp. 377–441.