A NOVEL KRYLOV SUBSPACE METHOD FOR APPROXIMATING FRÉCHET DERIVATIVES OF LARGE-SCALE MATRIX FUNCTIONS[*]

DANIEL KRESSNER[†] AND PETER OEHME[†]

Abstract. We present a novel Krylov subspace method for approximating $L_f(A, E)\mathbf{b}$, the matrix-vector product of the Fréchet derivative $L_f(A, E)$ of a large-scale matrix function $f(A)$ in direction E, a task that arises naturally in the sensitivity analysis of quantities involving matrix functions such as centrality measures for networks. It also arises in the context of gradient-based methods for optimization problems that feature matrix functions, e.g., when fitting an evolution equation to an observed solution trajectory. In principle, the well-known identity

$$f\left(\begin{bmatrix} A & E \\ 0 & A \end{bmatrix}\right)\begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} L_f(A, E)\mathbf{b} \\ f(A)\mathbf{b} \end{bmatrix}$$

allows one to directly apply any standard Krylov subspace method, such as the Arnoldi algorithm, to address this task. However, this comes with the major disadvantage that the involved block-triangular matrix has unfavorable spectral properties, which impede the convergence analysis and, to a certain extent, also the observed convergence. To avoid these difficulties, we propose a novel modification of the Arnoldi algorithm that aims at better preserving the block-triangular structure. In turn, this allows one to bound the convergence of the modified method by the best polynomial approximation of the derivative f' on the numerical range of A. Several numerical experiments illustrate our findings.

Key words. matrix function, Fréchet derivative, Krylov subspace method

AMS subject classifications. 15A16, 65F50, 65F60

1. Introduction. This work is concerned with matrix functions $f(A)$, the title motif of Higham's monumental monograph [15], where $A \in \mathbb{C}^{n \times n}$ is a matrix and f is a scalar function analytic in an open region containing the eigenvalues of A. Well-known examples include the matrix exponential, the matrix square root, the matrix logarithm, and the inverse of a matrix. Consider the Fréchet derivative of f at A, defined as the unique linear operator $L_f(A, \cdot) : \mathbb{C}^{n \times n} \to \mathbb{C}^{n \times n}$ such that

$$(1.1) \qquad f(A + \varepsilon E) - f(A) = \varepsilon L_f(A, E) + \mathcal{O}(\varepsilon^2)$$

holds for every $E \in \mathbb{C}^{n \times n}$ and for sufficiently small $\varepsilon > 0$. Such derivatives appear naturally in the sensitivity analysis of quantities involving matrix functions such as centrality and communicability measures for networks [9, 12, 27, 29]. They also arise when using gradient-based methods for optimization problems that feature matrix functions; see [13, 23, 30, 31] for instances of this situation in different fields of application.

For larger n, the sheer size of their $n^2 \times n^2$ matrix representation makes Fréchet derivatives much too expensive to compute and work with. On the other hand, none of the above applications actually requires full access to this quantity. Typically, only the derivative evaluated in a specific direction E applied to a vector $\mathbf{b} \in \mathbb{C}^n$ is needed:

$$(1.2) \qquad L_f(A, E) \cdot \mathbf{b},$$

which coincides with the directional derivative of $f(A)\mathbf{b}$. Moreover, the matrices A and E usually feature some (data) sparsity that makes them cheap to multiply with vectors,

[†]Corresponding author: D. Kressner. Institute of Mathematics, EPFL, Switzerland (daniel.kressner@epfl.ch, oehme.pb@gmail.com).

prompting the use of Krylov subspace methods. While the development and analysis of Krylov subspace methods for approximating large-scale matrix functions $f(A)\mathbf{b}$ has been an active area of research during the last decades, significantly less is known about such methods for approximating (1.2).

Before discussing numerical methods for (1.2), it is useful to recall existing approaches for computing the whole $n \times n$ matrix $L_f(A, E)$. A common starting point of many methods is the identity

$$(1.3) \qquad f(\mathcal{A}) = \begin{bmatrix} f(A) & L_f(A, E) \\ 0 & f(A) \end{bmatrix}, \qquad \text{with} \qquad \mathcal{A} = \begin{bmatrix} A & E \\ 0 & A \end{bmatrix},$$

which is attributed to Najfeld and Havel [24] in [15]. This allows one to straightforwardly apply existing robust methods for computing matrix functions, such as scaling-and-squaring for matrix exponentials [3], to $\mathcal{A}$ in order to compute $L_f(A, E)$. However, it can be computationally beneficial to modify such a method by, e.g., exploiting the block-triangular structure of $\mathcal{A}$ or targeting $L_f(A, E)$ directly; see [2, 4, 16] for examples. Alternatively, the definition (1.1) can be used to proceed via finite-difference approximations such as $L_f(A, E) \approx \big(f(A + \varepsilon E) - f(A)\big)/\varepsilon$ for very small ε, but this limits the attainable accuracy due to numerical cancellation [15, p. 68]. This effect can often be mitigated by using the complex step approximation $L_f(A, E) \approx \text{Im}\big(f(A + \mathrm{i}\varepsilon E) - f(A)\big)/\varepsilon$ (see [1]), but this introduces complex arithmetic for real data and does not easily generalize when A itself is complex.

Each of the above approaches can be used to employ standard Krylov subspace methods, such as the Arnoldi method, for matrix functions to approximate $L_f(A, E)\mathbf{b}$:

Arnoldi for $\mathcal{A}$: The identity (1.3) implies

$$f(\mathcal{A}) \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} L_f(A, E)\mathbf{b} \\ f(A)\mathbf{b} \end{bmatrix},$$

which suggests to apply the Arnoldi method with the matrix $\mathcal{A}$ and the starting vector $\begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$ to compute $L_f(A, E)\mathbf{b}$ and $f(A)\mathbf{b}$ simultaneously; see [12, Section 4.1] for an example of this approach. A major disadvantage is that $\mathcal{A}$ typically has much less favorable spectral properties than A. For example, it is simple to verify that $\mathcal{A}$ generically (with respect to E) has Jordan blocks of size at least 2, even when A is symmetric! In turn, convergence bounds based on the numerical range, such as in [8], can deteriorate significantly; see [7] for a related situation. Also, one can no longer use the less expensive Lanczos method when A is symmetric.

Finite differences with Arnoldi: Given a finite difference approximation like

$$L_f(A, E)\mathbf{b} \approx \big(f(A + \varepsilon E)\mathbf{b} - f(A)\mathbf{b}\big)/\varepsilon$$

for small $\varepsilon > 0$, one can apply the Arnoldi method to approximate each of the individual terms $f(A)\mathbf{b}$ and $f(A + \varepsilon E)\mathbf{b}$, as suggested in [12, Section 4.1]. Recently, an approximation based on central finite differences was suggested and studied in [25, Section 2.2]. As discussed above, cancellation is again a concern. In order to attain an error tolerance tol, one needs to choose $\varepsilon \lesssim \sqrt{\text{tol}}$. Because of the appearance of ε in the denominator, a simple application of the triangle inequality *suggests* that each of the terms needs to be approximated with a significantly higher accuracy $\text{tol}^{3/2}$ instead of tol. In other words, convergence might be adversely affected by cancellation of the Arnoldi approximations.
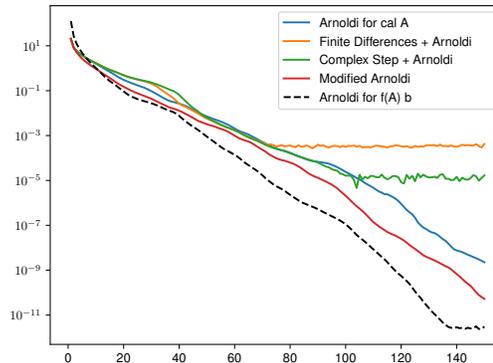
FIG. 1.1. *Error versus iterations of four different methods to approximate $L_f(A, E)\mathbf{b}$ for Example 1.1. The dashed curve corresponds to the Arnoldi method for approximating $f(A)\mathbf{b}$ and is shown for reference only.*

Complex step with Arnoldi: Combining the complex step approximation

$$L_f(A, E)\mathbf{b} \approx \mathrm{Im}\big(f(A + \mathrm{i}\varepsilon E)\mathbf{b} - f(A)\mathbf{b}\big)/\varepsilon$$

with the Arnoldi method comes (again) with the disadvantage that complex arithmetic is needed for real data. Also, the presence of ε is again a concern; at least we are not aware of theoretical results ruling out the cancellation effect of the Arnoldi approximations mentioned above.

In this work, we propose and analyze a modification of the Arnoldi method for $\mathcal{A}$ that better preserves the block-triangular structure of $\mathcal{A}$. This modification is inspired by existing Arnoldi methods for quadratic eigenvalue problems [5], and it allows us to mitigate the theoretical difficulties associated with the unfavorable spectral properties of $\mathcal{A}$. In particular, the results of Section 2.4 bound the convergence of the modified Arnoldi method for $L_f(A, E)\mathbf{b}$ by the best polynomial approximation of f' on the numerical range of A (instead of $\mathcal{A}$). This parallels existing results [7, 17] for $f(A)\mathbf{b}$ that bound the convergence of the Arnoldi method by the best polynomial approximation of f on the numerical range of A.

EXAMPLE 1.1. To provide some first insight into the numerical performance of the different methods for approximating $L_f(A, E)\mathbf{b}$, we consider $f(z) = \sqrt{z}$, $A = \mathrm{diag}(1, \ldots, 500)$, and random elements E, $\mathbf{b}$, generated using `randn` in Matlab. Figure 1.1 displays the error in the Euclidean norm versus the number of steps of the Arnoldi methods used for the approximation. Interestingly, the difficulties with the convergence theory mentioned above for existing methods show up in a much less pronounced way than one may expect. Still, our modified method converges fastest and at a rate comparable to the convergence of the Arnoldi method for approximating $f(A)\mathbf{b}$.

1.1. Low-rank directions. When the direction E has rank one or, more generally, low rank, then there are specialized methods for approximating the whole matrix $L_f(A, E)$ by a low-rank matrix. A (tensorized) Krylov subspace method has been proposed and analyzed in [19, 21]. The analysis in [10, 21] relates the convergence of this method also to the best polynomial approximation of f' on the numerical range of A, analogous to our analysis in Section 2.4. The close connection between [19, 21] and our work is also reflected by the following theorem, which shows that the computation of bilinear forms involving $L_f(A, E)$ can be effected by evaluating the Fréchet derivatives for rank-one directions. Note that $\langle \cdot, \cdot \rangle$ denotes the usual trace inner product defined by $\langle X, Y \rangle = \mathrm{trace}(X^*Y)$.

THEOREM 1.2. *Let $A \in \mathbb{C}^{n \times n}$ such that $L_f(A, \cdot)$ is well defined, $\mathbf{b}, \mathbf{c} \in \mathbb{C}^n$, and $E \in \mathbb{C}^{n \times n}$. Then it holds that $\mathbf{c}^* L_f(A, E)\mathbf{b} = \langle L_f(A, \mathbf{cb}^*), E \rangle$.*

Proof. It is well known that $L_f(A^*, \cdot)$ is the adjoint of the linear operator $L_f(A, \cdot) : \mathbb{C}^{n \times n} \to \mathbb{C}^{n \times n}$ with respect to the trace inner product; see, e.g., [15, p. 66]. Using the defining property of an adjoint, this implies the result

$$\mathbf{c}^* L_f(A, E)\mathbf{b} = \text{trace}(\mathbf{bc}^* L_f(A, E)) = \langle \mathbf{cb}^*, L_f(A, E) \rangle = \langle L_f(A^*, \mathbf{cb}^*), E \rangle. \qquad \square$$

For the special case of a rank-one matrix $E = \mathbf{xy}^*$, the result of Theorem 1.2 yields

$$(1.4) \qquad \mathbf{c}^* L_f(A, \mathbf{xy}^*)\mathbf{b} = \mathbf{y}^* L_f(A, \mathbf{cb}^*)^* \mathbf{x} = \overline{\mathbf{x}^* L_f(A, \mathbf{cb}^*)\mathbf{y}}.$$

This matches the result of [29, Theorem 2.3], which has been shown using vectorization and properties of Kronecker products.

2. Krylov subspace approximation of $L_f(A, E)\mathbf{b}$.

2.1. Preliminaries. We let $\mathcal{K}_k(A, \mathbf{b}) = \text{span}\{\mathbf{b}, A\mathbf{b}, \ldots, A^{k-1}\mathbf{b}\}$ denote the usual Krylov subspace, and, to simplify the discussion, we always assume that it has maximal dimension k. Let $U_k \in \mathbb{C}^{n \times k}$ contain an orthonormal basis of $\mathcal{K}_k(A, \mathbf{b})$. Then the Arnoldi method for matrix functions returns the approximation

$$(2.1) \qquad f(A)\mathbf{b} \approx U_k f(U_k^* A U_k) U_k^* \mathbf{b}.$$

One usually has $k \ll n$, and thus, the evaluation of the $k \times k$ matrix function $f(U_k^* A U_k)$ is relatively cheap. The convergence of (2.1) is closely related to the polynomial approximation of f. In fact, when f itself is a polynomial of degree at most $k-1$, then the approximation (2.1) is exact [14, 28]. The following lemma provides a slight variation of this polynomial exactness result, allowing the subspace to be larger than $\mathcal{K}_k(A, \mathbf{b})$:

LEMMA 2.1. *Let p be a polynomial of degree at most $k - 1$, and let U be an orthonormal basis of a subspace $\mathcal{U}$ such that $\mathcal{K}_k(A, \mathbf{b}) \subseteq \mathcal{U}$. Then it holds that*

$$p(A)\mathbf{b} = U p(U^* A U) U^* \mathbf{b}.$$

Proof. It suffices to show the claim for the monomials x^j, $j = 0, 1, \ldots, k-1$. We observe that

$$(2.2) \qquad U(U^* A U)^j U^* \mathbf{b} = (UU^* A)^j UU^* \mathbf{b}.$$

By assumption we know that $A^i \mathbf{b} \in \mathcal{U}$, whence $UU^* A^i \mathbf{b} = A^i \mathbf{b}$ for every $0 \leq i \leq j$. In turn,

$$(UU^* A)^j UU^* \mathbf{b} = (UU^* A)^j \mathbf{b} = (UU^* A)^{j-1} UU^* A\mathbf{b} = (UU^* A)^{j-1} A\mathbf{b}$$
$$= (UU^* A)^{j-2} A^2 \mathbf{b} = \cdots = A^j \mathbf{b}.$$

Combined with (2.2), this proves the claim $U(U^* A U)^j U^* \mathbf{b} = A^j \mathbf{b}$. $\qquad \square$

2.2. Modified Arnoldi algorithm: basic version. As explained in Section 1, directly applying the approximation (2.1) to the block-triangular embedding (1.3) in order to approximate $L_f(A, E)\mathbf{b}$ comes with theoretical difficulties. To address this issue, we consider a basis $\mathcal{V}_k \in \mathbb{C}^{2n \times k}$ for the relevant Krylov subspace $\mathcal{K}_k \left(\mathcal{A}, \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} \right)$ and tear it into two parts, letting

$U_k \in \mathbb{C}^{n \times k}$ and $V_k \in \mathbb{C}^{n \times k}$ contain orthonormal bases for the top n rows and bottom n rows of $\mathcal{V}_k$, respectively. Trivially,

$$\mathcal{K}_k \left(\mathcal{A}, \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} \right) \subset \operatorname{span}(\mathcal{U}_k), \qquad \text{with} \quad \mathcal{U}_k = \begin{bmatrix} U_k & 0 \\ 0 & V_k \end{bmatrix}.$$

By Lemma 2.1, the corresponding approximation

$$(2.3) \qquad \mathbf{f}_k = \mathcal{U}_k f(\mathcal{U}_k^* \mathcal{A} \mathcal{U}_k) \mathcal{U}_k^* \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$$

still satisfies polynomial exactness, and it comes with the advantage that the compressed matrix

$$(2.4) \qquad \mathcal{U}_k^* \mathcal{A} \mathcal{U}_k = \begin{bmatrix} U_k^* A U_k & U_k^* E V_k \\ 0 & V_k^* A V_k \end{bmatrix}$$

preserves the block-triangular structure of $\mathcal{A}$. By (1.3), the first n entries of $\mathbf{f}_k$ approximate $L_f(A, E)\mathbf{b}$, while its bottom n entries approximate $f(A)\mathbf{b}$.

Algorithm 1: Modified Arnoldi Algorithm (Basic Version).

Data: $A, E \in \mathbb{C}^{n \times n}, \mathbf{b} \in \mathbb{C}^n, k$
Result: Approximation $\mathbf{f}_k \in \mathbb{C}^{2n}$ defined in (2.3)

1 Compute a basis $\mathcal{V}_k$ of $\mathcal{K}_k \left(\mathcal{A}, \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} \right)$.

2 Split $\mathcal{V}_k = \begin{bmatrix} C \\ D \end{bmatrix}$, and orthogonalize $U_k = \operatorname{orth}(C), V_k = \operatorname{orth}(D)$.

3 Form $\mathcal{U}_k = \begin{bmatrix} U_k & 0 \\ 0 & V_k \end{bmatrix}$, and compute $\mathbf{f}_k = \mathcal{U}_k f(\mathcal{U}_k^* \mathcal{A} \mathcal{U}_k) \mathcal{U}_k^* \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$.

Algorithm 1 summarizes the procedure described above. In this basic form, Algorithm 1 clearly suffers from inefficiencies because the (stable) computation of $\mathcal{V}_k$ already requires orthogonalization, and then its bottom and top parts need to be orthogonalized again. Also, the explicit computation of the compressed matrix $\mathcal{U}_k^* \mathcal{A} \mathcal{U}_k$ requires additional matrix-vector products with A and E.

2.3. Modified Arnoldi algorithm: separate orthogonalization. To reduce the expense of orthogonalization, we recursively construct (non-orthonormal) Krylov subspace bases of the form

$$(2.5) \qquad \begin{bmatrix} U_i R_i \\ V_i \end{bmatrix} \in \mathbb{C}^{2n \times (i+1)},$$

such that $U_i \in \mathbb{C}^{n \times i}$, $V_i \in \mathbb{C}^{n \times (i+1)}$ are orthonormal and $R_i \in \mathbb{C}^{i \times (i+1)}$ is strictly upper triangular. In the following, we explain how the Arnoldi method applied to the Krylov subspace $\mathcal{K}_k \left(\mathcal{A}, \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} \right)$ is modified to yield bases of the form (2.5), for $i = 0, 1, \ldots, k$.

Initialization: For $i = 0$, the Krylov subspace is spanned by the vector $\begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$, and a basis of the form (2.5) is obtained by setting $V_0 = [\mathbf{v}_0]$ with $\mathbf{v}_0 = \mathbf{b}/\|\mathbf{b}\|_2$. Note that $U_0 \in \mathbb{C}^{n \times 0}$ and $R_0 \in \mathbb{C}^{0 \times 1}$ are empty matrices.

Iteration from $i - 1$ to i: Given a basis of the form (2.5) for $i - 1$, we partition its last vector into the two components $\begin{bmatrix} \mathbf{w}_{i-1} \\ \mathbf{v}_{i-1} \end{bmatrix}$. Note that $\mathbf{w}_{i-1}$ is not available explicitly but needs to be extracted from the matrix product $U_{i-1}R_{i-1}$. Multiplying the last basis vector with $\mathcal{A}$,

$$(2.6) \qquad \begin{bmatrix} A & E \\ 0 & A \end{bmatrix} \begin{bmatrix} \mathbf{w}_{i-1} \\ \mathbf{v}_{i-1} \end{bmatrix} = \begin{bmatrix} A\mathbf{w}_{i-1} + E\mathbf{v}_{i-1} \\ A\mathbf{v}_{i-1} \end{bmatrix},$$

and appending it to the current basis yields a basis for the next Krylov subspace provided that no breakdown in the Arnoldi process occurs. We now aim at transforming this new basis into the desired form

$$(2.7) \qquad \begin{bmatrix} U_i R_i \\ V_i \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} U_{i-1} & \mathbf{u}_i \end{bmatrix} R_i \\ \begin{bmatrix} V_{i-1} & \mathbf{v}_i \end{bmatrix} \end{bmatrix}.$$

One step of the Gram–Schmidt procedure is used to orthonormalize the bottom part of (2.6), $\tilde{\mathbf{v}}_i = A\mathbf{v}_{i-1}$, versus the existing basis V_{i-1}:

$$\mathbf{h} = V_{i-1}^* \tilde{\mathbf{v}}_i, \qquad \tilde{\mathbf{v}}_i \leftarrow \tilde{\mathbf{v}}_i - V_{i-1}\mathbf{h}, \qquad \beta = \|\tilde{\mathbf{v}}_i\|_2, \qquad \mathbf{v}_i = \tilde{\mathbf{v}}_i/\beta.$$

This amounts to the transformation

$$\begin{bmatrix} V_{i-1} & \mathbf{v}_i \end{bmatrix} = \begin{bmatrix} V_{i-1} & A\mathbf{v}_{i-1} \end{bmatrix} \begin{bmatrix} \mathrm{id} & -\mathbf{h}/\beta \\ 0 & 1/\beta \end{bmatrix}.$$

In order to preserve the basis property for the Krylov subspace, we need to apply the same transformation to the top part, absorbing it into the triangular factor:

$$\tilde{R}_i = \begin{bmatrix} R_{i-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{id} & -\mathbf{h}/\beta \\ 0 & 1/\beta \end{bmatrix} = \begin{bmatrix} R_{i-1} & -R_{i-1}\mathbf{h}/\beta \\ 0 & 1/\beta \end{bmatrix}.$$

Similarly, we orthonormalize $\tilde{\mathbf{u}}_i = A\mathbf{w}_{i-1} + E\mathbf{v}_{i-1}$ versus U_{i-1} by computing

$$\mathbf{g} = U_{i-1}^* \tilde{\mathbf{u}}_i, \qquad \tilde{\mathbf{u}}_i \leftarrow \tilde{\mathbf{u}}_i - U_{i-1}\mathbf{g} \qquad \alpha = \|\tilde{\mathbf{u}}_i\|_2, \qquad \mathbf{u}_i = \tilde{\mathbf{u}}_i/\alpha.$$

This time the corresponding transformation

$$\begin{bmatrix} U_{i-1} & \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} U_{i-1} & A\mathbf{w}_{i-1} + E\mathbf{v}_{i-1} \end{bmatrix} \begin{bmatrix} \mathrm{id} & -\mathbf{g}/\alpha \\ 0 & 1/\alpha \end{bmatrix}$$

cannot be applied to the other part because it would destroy the orthonormality of V_i. Instead, we compensate it by applying the inverse transformation to the triangular factor:

$$R_i = \begin{bmatrix} \mathrm{id} & \mathbf{g} \\ 0 & \alpha \end{bmatrix} \tilde{R}_i = \begin{bmatrix} R_{i-1} & -R_{i-1}\mathbf{h}/\beta + \mathbf{g}/\beta \\ 0 & \alpha/\beta \end{bmatrix}.$$

In turn, we have constructed a Krylov subspace basis taking the form (2.7).

Algorithm 2: Separate Orthonormalization.

Data: $A, E \in \mathbb{C}^{n \times n}, \mathbf{b} \in \mathbb{C}^n, k$
Result: $U_k \in \mathbb{C}^{n \times k}, V_k \in \mathbb{C}^{n \times (k+1)}, R_k \in \mathbb{C}^{k \times (k+1)}$

1 Compute $\beta = \|\mathbf{b}\|_2, \mathbf{v}_0 = \mathbf{b}/\beta$.
2 Set $V_0 = [\mathbf{v}_0], U_0 = [\,], R_0 = [\,]$.
3 **for** $i = 1, 2, 3, \ldots, k$ **do**
    `// Orthonormalization for `$A\mathbf{v}_{i-1}$`.`
4   Compute $\tilde{\mathbf{v}}_i = A\mathbf{v}_{i-1}$ and $\mathbf{h} = V_{i-1}^* \tilde{\mathbf{v}}_i$.
5   Orthogonalize $\tilde{\mathbf{v}}_i \leftarrow \tilde{\mathbf{v}}_i - V_{i-1}\mathbf{h}$.
6   Set $\beta = \|\tilde{\mathbf{v}}_i\|_2, \mathbf{v}_i = \tilde{\mathbf{v}}_i/\beta, V_i = [V_{i-1}, \mathbf{v}_i]$.
    `// Orthonormalization for `$AU_{i-1}R_{i-1}[:,-1] + E\mathbf{v}_{i-1}$`.`
7   Compute $\tilde{\mathbf{u}}_i = AU_{i-1}R_{i-1}[:,-1] + E\mathbf{v}_{i-1}$ and $\mathbf{g} = U_{i-1}^* \tilde{\mathbf{u}}_i$.
8   Orthogonalize $\tilde{\mathbf{u}}_i \leftarrow \tilde{\mathbf{u}}_i - U_{i-1}\mathbf{g}$.
9   Set $\alpha = \|\tilde{\mathbf{u}}_i\|_2, \mathbf{u}_i = \tilde{\mathbf{u}}_i/\alpha, U_i = [U_{i-1}, \mathbf{u}_i]$.
10   Update $R_i = \begin{bmatrix} R_{i-1} & -R_{i-1}\mathbf{h}/\beta + \mathbf{g}/\beta \\ 0 & \alpha/\beta \end{bmatrix}$.

Algorithm 2 summarizes the described procedure. Note that for $i = 1$, line 7 simplifies to $\tilde{\mathbf{u}}_1 = E\mathbf{v}_0$, $\mathbf{g}$ remains void, and line 10 simplifies to $R_1 = \begin{bmatrix} 0 & \alpha/\beta \end{bmatrix}$.

Note that Algorithm 2 already carries out all the required matrix-vector products with A and E for forming the compressed matrix $\mathcal{U}_k^* \mathcal{A} \mathcal{U}_k$; see (2.4). We can thus avoid the need for extra large-scale matrix-vector products by letting Algorithm 2 store and return the matrices $AU_k, V_k^* AV_k$, and EV_k. Compared to the naïve orthogonalization in Algorithm 1, we have thus not only removed the additional orthogonalizations in line 2, but we have also avoided the block-matrix multiplication in line 3. This mitigates the computational inefficiencies discussed above at the expense of $\mathcal{O}(nk)$ additional memory requirements for storing AU_k and EV_k.

2.4. Convergence analysis. Lemma 2.1 combined with the block-structure preservation (2.4) allow us to analyze the convergence of Algorithm 1 (with or without the separate orthogonalization performed by Algorithm 2). The following theorem is the main theoretical result of this work, relating convergence to the best polynomial approximation of the derivative of f on the numerical range $W(A) = \{x^* Ax : \|x\|_2 = 1\}$ of A.

THEOREM 2.2. *Given an analytic function* $f : \Omega \to \mathbb{C}$ *for some domain* $\Omega \subseteq \mathbb{C}$ *and* $A \in \mathbb{C}^{n \times n}$, *assume that* $W(A) \subset \Omega$. *For some* $E \in \mathbb{C}^{n \times n}$, $\mathbf{b} \in \mathbb{C}^n$, *partition the output of Algorithm 1 into* $\mathbf{f}_k = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}$ *such that* $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{C}^n$. *Letting* Π_{k-2} *denote the set of all polynomials of degree at most* $k - 2$, *for* $k \geq 2$, *it holds that*

$$\|L_f(A, E)\mathbf{b} - \mathbf{v}_1\|_2 \leq 2C \cdot \|\mathbf{b}\|_2 \|E\|_F \cdot \inf_{q \in \Pi_{k-2}} \sup_{z \in W(A)} |f'(z) - q(z)|,$$

with $C = 1$ *if* A *is normal and* $C = (1 + \sqrt{2})^2$ *otherwise.*

Proof. For arbitrary $p \in \Pi_{k-1}$, let us denote $\mathbf{p}_k = \mathcal{U}_k p(\mathcal{U}_k^* \mathcal{A} \mathcal{U}_k) \mathcal{U}_k^* \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$, where $\mathcal{U}_k$ is the block-diagonal basis constructed by Algorithm 1. Using Lemma 2.1, we may write

$$(2.8) \qquad f(\mathcal{A}) \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} - \mathbf{f}_k = e(\mathcal{A}) \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} - \mathbf{e}_k,$$

with $e = f - p$ and $\mathbf{e}_k = \mathbf{f}_k - \mathbf{p}_k$. Recalling (1.3), it holds that

$$e(\mathcal{A}) = \begin{bmatrix} e(A) & L_e(A, E) \\ 0 & e(A) \end{bmatrix}.$$

Corollary 5.1 in [10] gives the upper bound

$$\|L_e(A, E)\|_F \leq C\|E\|_F \cdot \sup_{z \in W(A)} |e'(z)|.$$

To address the second term in (2.8), we first note that

$$e(\mathcal{U}_k^* \mathcal{A} \mathcal{U}_k) = e\left(\begin{bmatrix} U_k^* A U_k & U_k^* E V_k \\ 0 & V_k^* A V_k \end{bmatrix}\right) = \begin{bmatrix} e(U_k^* A U_k) & E_{12} \\ 0 & e(V_k^* A V_k) \end{bmatrix},$$

for some $E_{12} \in \mathbb{C}^{k \times k}$; see, e.g., [20] and [15]. Because the numerical ranges of $U_k^* A U_k$ and $V_k^* A V_k$ are both contained in $W(A)$, we can apply [6, Lemma 4.1], which gives

$$\|E_{12}\|_F \leq C\|E\|_F \cdot \sup_{z \in W(A)} |e'(z)|.$$

Inserting these relations into (2.8) gives

$$L_f(A, E)\mathbf{b} - \mathbf{v}_1 = L_e(A, E)\mathbf{b} - U_k E_{12} V_k^* \mathbf{b}$$

and the bound

$$\|L_f(A, E)\mathbf{b} - \mathbf{v}_1\|_2 \leq (\|L_e(A, E)\|_F + \|E_{12}\|_F)\|\mathbf{b}\|_2 \leq 2C \cdot \|\mathbf{b}\|_2 \|E\|_F \cdot \sup_{z \in W(A)} |e'(z)|.$$

Noting that $e' = f' - p'$ and setting $q = p'$ concludes the proof. $\square$

3. Numerical experiments. To test our newly proposed algorithm, all numerical experiments were run on a MacBook Air with 16 GB of RAM, an Apple M2 processor, and version 26.1 of macOS. The Python source code is publicly available at github.com/peoe/fAb-frechet.

3.1. Centrality measure sensitivity in network analysis. Let A be the adjacency matrix of a graph. There are multiple centrality measures for individual graph nodes and edges, some of which can be expressed as matrix functions. We focus on the following functions:

Total Network Communicability: $\quad f_{\text{TN}}(A) := \mathbf{1}^T \exp(A)\mathbf{1},$

Subgraph Centrality: $\quad f_{\text{SC}}(A, \ell) := \mathbf{e}_\ell^T \exp(A)\mathbf{e}_\ell,$

Estrada Index: $\quad f_{\text{EI}}(A) := \text{trace}(\exp(A)).$

Most applications of these quantities focus on their sensitivities with respect to changes of the adjacency matrix A. Thus, one wishes to compute the Fréchet derivative and the subsequent sensitivity matrices $S^{(f)}$ as

$$S_{ij}^{(TN)}(A) := \mathbf{1}^T L_{\exp}(A, \mathbf{e}_i \mathbf{e}_j^T)\mathbf{1},$$
$$S_{ij}^{(SC)}(A, \ell) := \mathbf{e}_\ell^T L_{\exp}(A, \mathbf{e}_i \mathbf{e}_j^T)\mathbf{e}_\ell,$$
$$S_{ij}^{(EI)}(A) := \text{trace}(L_{\exp}(A, \mathbf{e}_i \mathbf{e}_j^T)).$$

TABLE 3.1
Row and column indices used in the computation of $S_{ij}^{(TN)}$.

Network	Row index i	Column index j
Air500	256	123
Autobahn	218	605
Pajek/USPowerGrid	3579	2400
SNAP/as-735	3105	5000
SNAP/ca-HepTh	7200	6969

Applying the result (1.4), which is a consequence of Theorem 1.2 and [29, Corollary 4.5], we can reduce these to

$$S_{ij}^{(TN)}(A) = \mathbf{e}_i^T L_{\exp}(A^T, \mathbf{1}\mathbf{1}^T)\mathbf{e}_j,$$
$$S_{ij}^{(SC)}(A,\ell) = \mathbf{e}_i^T L_{\exp}(A^T, \mathbf{e}_\ell \mathbf{e}_\ell^T)\mathbf{e}_j,$$
$$S_{ij}^{(EI)}(A) = \mathbf{e}_i^T L_{\exp}(A^T, \text{id})\mathbf{e}_j.$$

We compute $S_{ij}^{(TN)}$ for the adjacency matrices from the following datasets:

Air500 [26]: The dataset from [22] contains as nodes the top 500 airports in the world based on passenger volume between July 2007 and June 2008. The 24009 edges in this network represent flight connections, resulting in a directed unweighted graph.

Autobahn [26]: The dataset from [18] represents the German highway network in 2002. The graph has 1168 vertices and represents the 1243 connections between highway segments as 2486 unweighted edges.

Pajek/USPowerGrid [11]: The dataset contains 4941 nodes and 13188 edges making up parts of the connections in the US power grid.

SNAP/as-735: The dataset contains 7716 nodes and 26467 edges representing the autonomous systems of internet router subgraphs from November 1997 to January 2000.

SNAP/ca-HepTh [11]: The dataset of arXiv collaborations in the High Energy Physics-Theory category from January 1993 to April 2003 contains 9877 vertices and 51971 edges.

To get a comparison between the methods of direct computation, the low-rank updating from [7], and our new method, we only compute $\mathbf{1}^T L_{\exp}(A^T, E_{ij})\mathbf{1}$ for a single pair of indices i and j. These indices are chosen according to Table 3.1. Because the direction of change in the Fréchet derivative always has rank 1, we can use the methods proposed in [12], [29], the Arnoldi method for $\mathcal{A}$, and our modified Arnoldi algorithm combining Algorithms 1 and 2. In Figures 3.1, 3.2, 3.3, 3.4, and 3.5 we display the convergence of these four methods for all networks.

In addition to the convergence plots above, we also compare the computational times for $k = 50$ steps of the four methods. These are only preliminary metrics, but they suggest that the additional costs from the computations are acceptable. In particular, the runtime of our algorithm matches that of the Arnoldi method for $\mathcal{A}$ closely. We list detailed runtimes in Table 3.2.

To demonstrate the capability of our algorithm in a setting where the direction E of the Fréchet derivative $L_{\exp}(A, E)$ is not low-rank, we take the quantity $\mathbf{1}^T L_{\exp}(A, \mathbf{e}_i \mathbf{e}_j^T)\mathbf{1}$ and instead compute $S_{ij}^{(FR-TN)} = \mathbf{1}^T L_{\exp}(A, E)\mathbf{1}$. The matrix E is chosen such that $E_{ij} = 1$ if
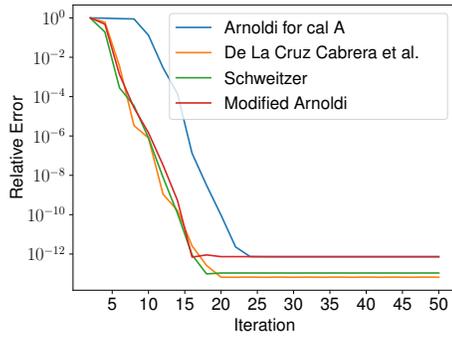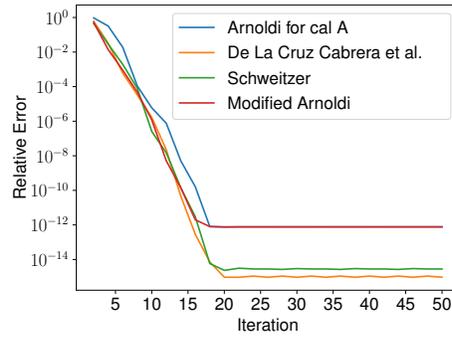
FIG. 3.1. *Convergence of $S_{ij}^{(TN)}$ for* `Air500`.
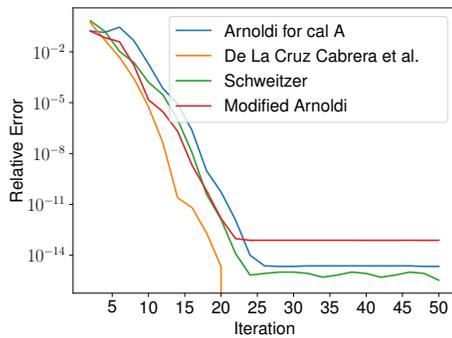
FIG. 3.2. *Convergence of $S_{ij}^{(TN)}$ for* `Autobahn`.

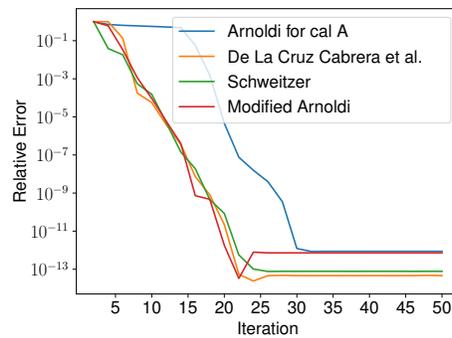FIG. 3.3. *Convergence of $S_{ij}^{(TN)}$ for* `Pajek/USPowerGrid`.

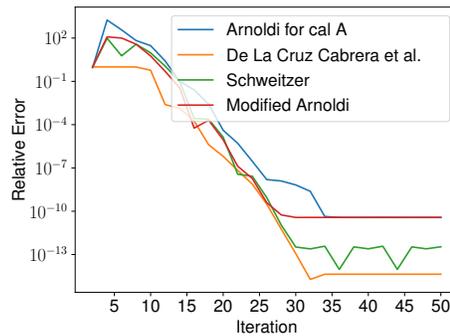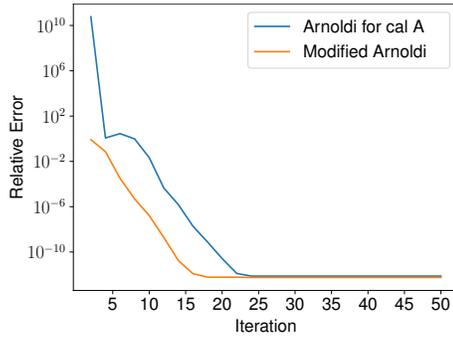FIG. 3.4. *Convergence of $S_{ij}^{(TN)}$ for* `SNAP/as-735`.

FIG. 3.5. *Convergence of $S_{ij}^{(TN)}$ for* `SNAP/ca-HepTh`.

$A_{ij} \neq 0$, which generally results in E not being low-rank. In this case the methods of [12] and [29] do not apply, and thus we only present the convergence of the Arnoldi method for $\mathcal{A}$ and our modified Arnoldi algorithm, combining Algorithms 1 and 2, in Figures 3.6, 3.7, 3.8, 3.9, and 3.10.

3.2. Parameter fitting for the 2D parametric heat equation. We demonstrate the application of Fréchet derivatives in the setting of parameter fitting. The experiment we conduct here serves as a simplified version of the more advanced examples used in machine

Runtimes in seconds of the four methods computing $S_{ij}^{(TN)}$ for the different networks with $k = 50$ steps.

Network	Arnoldi for $\mathcal{A}$	De La Cruz Cabrera et al.	Schweitzer	Modified Arnoldi
`Air500`	0.01511	0.03423	0.02542	0.00735
`Autobahn`	0.02713	0.0396	0.03605	0.01091
`USPowerGrid`	0.09601	0.1645	0.133	0.03422
`as-735`	0.1486	0.27532	0.2154	0.06692
`HepTh`	0.18254	0.40463	0.28057	0.06598

FIG. 3.6. *Convergence of $S_{ij}^{(FR-TN)}$ for* `Air500`.
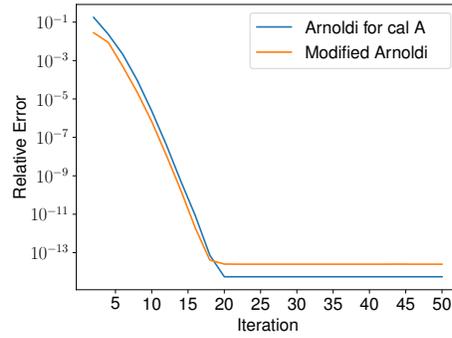

FIG. 3.7. *Convergence of $S_{ij}^{(FR-TN)}$ for* `Autobahn`.
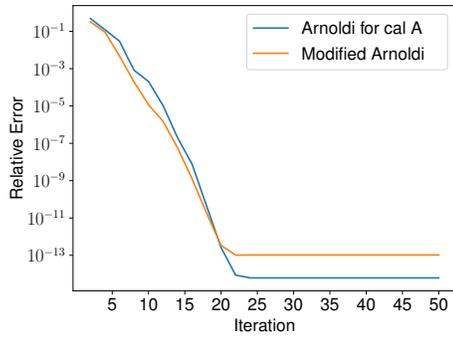

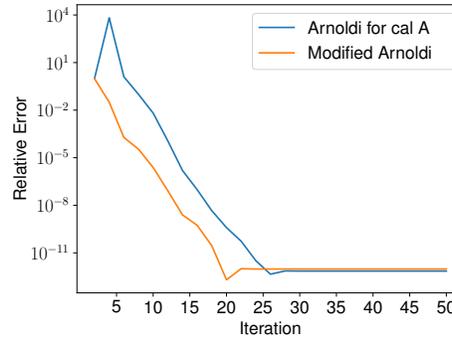FIG. 3.8. *Convergence of $S_{ij}^{(FR-TN)}$ for* `Pajek/USPowerGrid`.

FIG. 3.9. *Convergence of $S_{ij}^{(FR-TN)}$ for* `SNAP/as-735`.

learning, a topic which would go beyond the scope of this paper.

Let $\Omega = [-1, 1]^2$ be a 2D domain and $T = 1$ the final time. The parametric heat equation with homogeneous Dirichlet boundary conditions is given by

$$
\begin{aligned}
\partial_t u(t, x, y) &= \sigma \Delta u(t, x, y), & \forall t \in [0, T], x, y \in \Omega, \\
u(t, x, y) &= 0, & \forall t \in [0, T], x, y \in \partial\Omega, \\
u(0, x, y) &= u_0(x, y), & \forall x, y \in \Omega,
\end{aligned}
$$
(3.1)

where σ is a scalar parameter determining the thermal conductivity of the domain Ω.

We discretize (3.1) on an equispaced 75×75 grid using second-order central finite differences, resulting in a system matrix $A(\sigma)$ of size 5625×5625. The solution of the
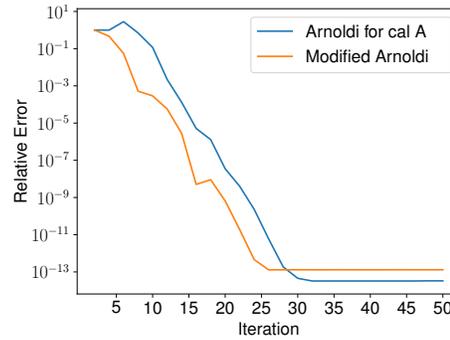
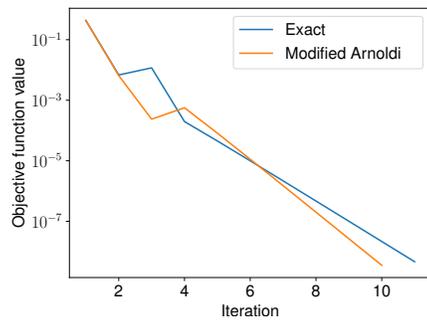FIG. 3.10. *Convergence of $S_{ij}^{(FR-TN)}$ for* `SNAP/ca-HepTh`.

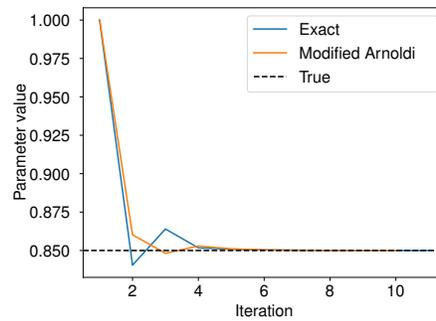FIG. 3.11. *Convergence of the function value $f(\sigma)$ during the gradient descent.*

FIG. 3.12. *Convergence of the parameter value σ during the gradient descent.*

discretized equation (3.1) is given by the solution operator $\mathbf{s}(\sigma) \coloneqq \exp\left(TA(\sigma)\right)\mathbf{u}_0$. This system serves as a simple model in parameter fitting where the objective is to find a parameter value σ^* which, given a reference solution $\mathbf{s}_{\mathrm{ref}}$, minimizes

$$f(\sigma) = \|\mathbf{s}(\sigma) - \mathbf{s}_{\mathrm{ref}}\|_2^2.$$

To solve this problem we employ a gradient descent method, requiring the evaluation of $f(\sigma_k)$ and $f'(\sigma_k)$ for multiple parameter values $\sigma_1, \sigma_2, \ldots$ This is expensive because every evaluation of f requires the computation of the matrix function $\exp\left(TA(\sigma)\mathbf{u}_0\right)$, and approximations of f' via techniques such as finite differences require additional evaluations of f. Instead, we use Algorithm 1 to compute the matrix function action $\exp\left(TA(\sigma_k)\right)\mathbf{u}_0$ and its Fréchet derivative action $L_{\exp}(TA(\sigma_k), \partial_\sigma A(\sigma_k)\mathbf{u}_0$ at the same time for the given right-hand side.

In Figure 3.11 we display the value of $f(\sigma)$ over the trajectory of σ computed in the gradient descent algorithm. We used an initial step size of $1/2$ and an absolute stopping tolerance of 10^{-8}. In Figure 3.12 we additionally present the trajectory of the parameter σ from the initial guess $\sigma = 1$ to the optimal value of $\sigma = 0.85$.

REFERENCES

[1] A. H. AL-MOHY AND B. ARSLAN, *The complex step approximation to the higher order Fréchet derivatives of a matrix function*, Numer. Algorithms, 87 (2021), pp. 1061–1074.

[2] 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 (2008), pp. 1639–1657.

[3] ———, *A new scaling and squaring algorithm for the matrix exponential*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 970–989.

[4] 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.

[5] Z. BAI AND Y. SU, *SOAR: a second-order Arnoldi method for the solution of the quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 640–659.

[6] B. BECKERMANN, A. CORTINOVIS, D. KRESSNER, AND M. SCHWEITZER, *Low-rank updates of matrix functions II: rational Krylov methods*, SIAM J. Numer. Anal., 59 (2021), pp. 1325–1347.

[7] B. BECKERMANN, D. KRESSNER, AND M. SCHWEITZER, *Low-rank updates of matrix functions*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 539–565.

[8] B. BECKERMANN AND L. REICHEL, *Error estimates and evaluation of matrix functions via the Faber transform*, SIAM J. Numer. Anal., 47 (2009), pp. 3849–3883.

[9] M. BENZI AND P. BOITO, *Matrix functions in network analysis*, GAMM-Mitt., 43 (2020), Paper No. e202000012, 36 pages.

[10] M. CROUZEIX AND D. KRESSNER, *A bivariate extension of the Crouzeix–Palencia result with an application to Fréchet derivatives of matrix functions*, Linear Multilinear Algebra, 73 (2025), pp. 2493–2500.

[11] T. A. DAVIS AND Y. HU, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), Art. 1, 25 pages.

[12] O. DE LA CRUZ CABRERA, J. JIN, S. NOSCHESE, AND L. REICHEL, *Communication in complex networks*, Appl. Numer. Math., 172 (2022), pp. 186–205.

[13] G. DIDIER, N. E. GLATT-HOLTZ, A. J. HOLBROOK, A. F. MAGEE, AND M. A. SUCHARD, *On the surprising effectiveness of a simple matrix exponential derivative approximation, with application to global SARS-CoV-2*, Proc. Natl. Acad. Sci. USA, 121 (2024), Paper No. e2318989121, 10 pages.

[14] T. ERICSSON, *Computing functions of matrices using Krylov subspace methods*, Tech. Report, Chalmers University of Technology, Department of Computer Science, Göteborg, 1990.

[15] N. J. HIGHAM, *Functions of Matrices*, SIAM, Philadelphia, 2008.

[16] N. J. HIGHAM AND L. LIN, *An improved Schur-Padé algorithm for fractional powers of a matrix and their Fréchet derivatives*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1341–1360.

[17] M. HOCHBRUCK AND C. LUBICH, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.

[18] M. KAISER AND C. C. HILGETAG, *Spatial growth of real-world networks*, Phys. Rev. E, 69 (2004), Paper No. 036103, 5 pages.

[19] P. KANDOLF, A. KOSKELA, S. D. RELTON, AND M. SCHWEITZER, *Computing low-rank approximations of the Fréchet derivative of a matrix function using Krylov subspace methods.*, Numer. Linear Algebra Appl., 28 (2021), Paper No. e2401, 31 pages.

[20] C. S. KENNEY AND A. J. LAUB, *A Schur-Fréchet algorithm for computing the logarithm and exponential of a matrix*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 640–663.

[21] D. KRESSNER, *A Krylov subspace method for the approximation of bivariate matrix functions*, in Structured Matrices in Numerical Linear Algebra, D. A. Bini, F. Di Benedetto, E. Tyrtyshnikov, and M. Van Barel, eds., Springer INdAM Ser., 30, Springer, Cham, 2019, pp. 197–214.

[22] J. MARCELINO AND M. KAISER, *Critical paths in a metapopulation model of H1N1: efficiently delaying influenza spreading through flight cancellation*, PLoS Curr., 4 (2012), Paper No. e4f8c9a2e1fca8.

[23] F. MONTI, X. JI, AND M. A. SUCHARD, *Nonparametric modeling of continuous-time Markov chains*, Preprint on arXiv, 2025. https://arxiv.org/abs/2511.03954

[24] I. NAJFELD AND T. F. HAVEL, *Derivatives of the matrix exponential and their computation*, Adv. in Appl. Math., 16 (1995), pp. 321–375.

[25] S. NOSCHESE AND L. REICHEL, *Edge importance in complex networks*, Numer. Algorithms, 99 (2025), pp. 377–410.

[26] PRECISION IMAGING, *Dynamic Connectome Lab - Resources*, website. https://sites.google.com/view/dynamicconnectomelab/resources

[27] S. POZZA AND F. TUDISCO, *On the stability of network indices defined by means of matrix functions*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 1521–1546.

[28] Y. SAAD, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.

[29] M. SCHWEITZER, *Sensitivity of matrix function based network communicability measures: computational methods and a priori bounds*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 1321–1348.

[30] D. THANOU, X. DONG, D. KRESSNER, AND P. FROSSARD, *Learning heat diffusion graphs*, IEEE Trans. Signal Inform. Process. Netw., 3 (2017), pp. 484–499.

[31] A. ZEILMANN, S. PETRA, AND C. SCHNÖRR, *Learning linearized assignment flows for image labeling*, J. Math. Imaging Vis., 65 (2023), pp. 164–184.