Electronic Transactions on Numerical Analysis.
Volume yy, pp. 1–xx, 2024.

ISSN 1068–9613.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

CHALLENGES IN COMPUTING MATRIX FUNCTIONS*

MASSIMILIANO FASI*, STÉPHANE GAUDREAULT†, KATHRYN LUND‡, AND MARCEL SCHWEITZER§

This work is dedicated to the memory of Nicholas J. Higham (1961–2024).

Abstract. This manuscript summarizes the outcome of the focus groups at *The f(A)bulous workshop on matrix functions and exponential integrators*, held at the *Max Planck Institute for Dynamics of Complex Technical Systems* in Magdeburg, Germany, on 25–27 September 2023. There were three focus groups in total, each with a different theme: knowledge transfer, high-performance and energy-aware computing, and benchmarking. We collect insights, open issues, and perspectives from each focus group, as well as from general discussions throughout the workshop. Our primary aim is to highlight ripe research directions and continue to build on the momentum from a lively meeting.

Key words. exponential integrators, high-performance computing, matrix functions, matrix function times a vector, numerical linear algebra, research data management, workshop

AMS subject classifications. 15A16, 15-11, 65F60, 65Y05, 65Y20, 65-11

The "$f(A)\boldsymbol{b}$" community consists of researchers who develop, study, or use computational methods for computing the action of a matrix function on one or more vectors. These vectors can be considered either as a sequence of individual vectors or as a concatenation (*block vector*). A scalar function f can be evaluated at a square matrix A in a natural way that preserves many interesting properties of f. Formally, $f(A)$ can be defined by means of the Jordan canonical form of A, the Taylor series expansion of f, its Cauchy integral representation, or Hermite interpolation [51, Chapter 1]. If f is analytic in a region that contains the spectrum of A, then these definitions are all equivalent.

In numerical linear algebra, the expression "computing matrix functions" denotes two very different tasks:

T1. $f(A)$, i.e., the evaluation of the function f at the $m \times m$ matrix A, which will produce an $m \times m$ matrix; and

T2. $f(A)\boldsymbol{b}$, i.e., the computation of the action of $f(A)$ on the $m \times n$ matrix $\boldsymbol{b}$, where $n \ll m$, which will produce an $m \times n$ matrix.

In theory, any algorithm applicable to T1 can be used for T2 prior to a matrix–matrix product with $\boldsymbol{b}$. In many practical applications, however, the matrix A is large and sparse, and the dense matrix $f(A)$ becomes impossible to store explicitly. Some applications also require a linear combination of the action of multiple functions on different vectors, rendering this approach intractable.

A prime example of such types of problems is exponential time integrators [56, 63], a class of numerical methods for solving ordinary differential equations (ODEs) of the form

$$\frac{d}{dt}u(t) = F\big(u(t)\big), \quad u(t_0) = u_0. \tag{0.1}$$

Differential equations in this form appear in many areas of natural and social sciences. In the majority of applications, the variable $u(t)$ represents an unknown dynamic quantity, t is the independent variable, and F describes the dynamics of the system. Different types of

*Received .... Accepted .... Published online on .... Recommended by .... Work supported by ....

*School of Computing, University of Leeds, Woodhouse Lane, Leeds LS2 9JT (m.fasi@leeds.ac.uk).
†Recherche en prévision numérique atmosphérique, Environnement et Changement climatique Canada, 2121 Route Transcanadienne, Dorval, Québec, H9P 1J3, Canada (stephane.gaudreault@ec.gc.ca).
‡Max Planck Institute for Dynamics of Complex Technical Systems (lund@mpi-magdeburg.mpg.de).
§School of Mathematics and Natural Sciences, Bergische Universität Wuppertal, 42097 Wuppertal (marcel@uni-wuppertal.de).

exponential integrators can be derived by optimizing the coefficients of an "ansatz" of the form

$$\varphi_0(A)\boldsymbol{v}_0 + \varphi_1(A)\boldsymbol{v}_1 + \varphi_2(A)\boldsymbol{v}_2 + \ldots + \varphi_p(A)\boldsymbol{v}_p, \tag{0.2}$$

where the so-called φ-functions can be defined by the Taylor series

$$\varphi_k(A) = \sum_{i=0}^{\infty} \frac{A^i}{(i+k)!}.$$

In other words, the ansatz eq. (0.2) is just a linear combination of exponential-like functions evaluated at A that act on a set of vectors. Computationally, evaluating $\varphi_i(A)\boldsymbol{v}_i$ is the most expensive step in exponential time integrators, and it has therefore been the subject of a considerable amount research.

0. The workshop. In September 2023, members of the $f(A)\boldsymbol{b}$ community met at the Max Planck Institute for Dynamics of Complex Technical Systems in Magdeburg, Germany, to attend the first "f(A)bulous workshop on matrix functions and exponential integrators",[1] organized by Kathryn Lund, Stéphane Gaudreault, and Marcel Schweitzer. The event featured traditional-style scientific talks covering recent algorithmic advances as well as applications, but a significant portion of the two and a half days was reserved for moderated discussion sessions.

For these sessions, the workshop participants split into three focus groups (each led by one of the organizers of the workshop) in order to consider a broad challenge the community is facing, and they spent one afternoon looking at the issue and assembling potential solutions. For each group, a list of key questions was provided to foster and stimulate a lively—but also focused—discussion. For the guiding slides with questions, see [43].

The next day, each group reported the main discussion points to the other attendees, and the floor was then opened for further comments and questions. Several participants (in particular, Thomas Mach and Yannis Voet) took thorough notes on the discussions and shared them with other participants via the workshop website.

The three challenges that were selected for these focus groups were as follows:

1. knowledge transfer between the "$f(A)\boldsymbol{b}$ community" and researchers from other areas (moderated by Marcel Schweitzer),
2. high-performance and energy-aware computing (moderated by Stéphane Gaudreault), and
3. benchmark problems and FAIR comparisons (moderated by Kathryn Lund).

The next three sections of this report summarize the main conclusions of the three focus groups and the main points that were raised in the discussion that ensued.

1. Knowledge transfer. The first group looked at ways to ensure that the algorithms developed within the community can reach the end users who need them.

As discussed before, items T1 and T2 are fundamentally different from a computational perspective, and they generally require different techniques. In particular, when only $f(A)\boldsymbol{b}$ is sought, computing $f(A)$ is typically an unduly expensive and totally unnecessary step.

Anecdotal evidence suggests that the distinction between the two problems is not as clear outside the $f(A)\boldsymbol{b}$ community, and that researchers that wish to compute $f(A)\boldsymbol{b}$ in an application domain often rely on the simple (but extremely inefficient) approach of multiplying $\boldsymbol{b}$ by $f(A)$ after having computed the latter explicitly.

There are several reasons behind this phenomenon. For one, there is no recent, authoritative source summarizing the existing methods for evaluating $f(A)\boldsymbol{b}$. The only comprehensive

[1] https://indico3.mpi-magdeburg.mpg.de/event/30/

survey of the literature [40] is now over 16 years old and does not reflect the breadth of methods currently available; more recent surveys and a thesis [46, 47, 69] only deal with Krylov subspace methods and focus especially on limited-memory scenarios. The situation is similar for exponential integrators, where the last comprehensive survey [56] is 14 years old. This is in stark contrast to the literature on $f(A)$, which boasts a book [51],[2] a survey paper dedicated to computational aspects [52], and even a survey of existing software, which is periodically updated (and also includes software for $f(A)\boldsymbol{b}$) [28, 53, 54].

An additional obstacle in knowledge transfer is the absence of $f(A)\boldsymbol{b}$ in the standard academic curriculum. In effect, most practitioners learn about the topic either during their graduate studies or through self-study, and their learning is hindered by the lack of a comprehensive review or textbook.

For most functions f of interest, it is relatively easy to find robust and efficient software toolboxes to evaluate $f(A)$. These are available for most programming environments and work out-of-the-box for a large selection of test problems. The situation is very different in the $f(A)\boldsymbol{b}$ case, as the performance of the algorithm depends on a number of factors, and, with the software currently available, a certain level of experience is needed to find the right combination of parameters for a given computation.

In most cases, the missing cornerstone is a reliable stopping criterion. A stopping criterion requires a way of measuring or bounding the approximation error, but at present there is a knowledge gap in this area: available results typically only apply to normal matrices and are restricted to specific classes of matrices (e.g., Kronecker sums [10, 11]) or specific functions (e.g., Cauchy–Stieltjes functions [35, 39, 48, 49, 62], Laplace transforms [31, 37] or functions which can be related to an underlying ODE initial-value problem [13, 14, 15, 16]). The error bounds available for more general cases may be very pessimistic, and they are typically not fit for use as stopping criteria in practice. Furthermore, the derivation of error bounds quickly becomes interdisciplinary, as often function-specific, analytical results are necessary for deriving error expressions. This is in stark contrast to, say, linear systems of the form $A\boldsymbol{x} = \boldsymbol{b}$, where a notion of residual $\boldsymbol{r} := \boldsymbol{b} - A\widetilde{\boldsymbol{x}}$ for an approximation $\widetilde{\boldsymbol{x}}$ is readily available from the data, can be cheaply approximated in Krylov subspace methods like GMRES, and is widely used as a reliable stopping criterion [68].

The situation is further complicated by the lack of comprehensive comparisons of the performance of different algorithms, which is made arduous by the large number of implementation details and parameters that have to be selected. We address this point further in Section 3.

This focus group also identified a few timely challenges for the community, which represent, at the same time, opportunities for research advances.

- Current methods may not exploit the full breadth of available techniques, a case in point being randomized methods, which have just started to be considered [18, 27, 50, 66, 67].
- Existing implementations are not able to fully leverage the variety of hardware on modern computers and supercomputers, which makes them potentially less attractive. See also Section 2.
- Often, unlocking the full potential of methods might require "intermingling" the algorithm used for approximating $f(A)\boldsymbol{b}$ with the surrounding ecosystem, taking into account specifics of the application at hand, instead of just treating it as a black-box

[2]Incidentally, in the preface of [51], the author states "The problem of computing a function of a matrix times a vector, $f(A)\boldsymbol{b}$, is of growing importance, though as yet numerical methods are relatively undeveloped". Due to this growing importance, a lot of developments have taken place since then, which are of course not covered by the last "$f(A)\boldsymbol{b}$ survey" [40], which was published in the same year as the book [51].

that returns an approximate solution (see, e.g., [26] for an example of a "Krylov-aware" approach in trace estimation, or [44] for an algorithm exploiting the intimate connection to exponential integrators when approximating linear combinations of φ-functions).

Next steps. Although a clear path to solve all the challenges affecting knowledge transfer is hard to pin down, some steps the community can take to make some progress on these issues are relatively easy to trace.

First and foremost, there is a need for effective benchmarking standards. All methods to be compared should be implemented from the same building blocks, and the same implementation choices should be applied consistently. The performance of the methods should be measured in a uniform way, and this is not necessarily a simple task: estimating the execution time and memory usage of an algorithm is simple, but assessing its accuracy is not. Accuracy is usually measured in terms of the forward error of the computed result, but since the exact solution is often not available, a reference solution computed using a different algorithm—potentially run in higher-than-working precision—is typically employed. Attention should be paid to how the reference solution is computed, and how the error is estimated from it.

Ideally, one could identify classes of problems where certain methods perform well, so that precise and easy-to-follow recommendations—based on the structure of A, the behavior of f, or a combination of both—can be made.

In order for the comparison to be useful, it is also crucial that a representative set of benchmark problems be established. Difficult questions that should be addressed regard:

- what information should be collected, in addition to the obvious f, A, and $\boldsymbol{b}$, and
- what format should be used to store this information.

These points were discussed in more detail by another focus group—see Section 3.

These efforts would put the community in a better position to summarize the literature and produce easy-to-follow guidance for all those interested in computing $f(A)\boldsymbol{b}$ without delving into algorithmic and theoretical details: a general consensus is that drafting a modern survey of numerical methods for computing $f(A)\boldsymbol{b}$ should be a priority.[3] The lack of a comprehensive literature review for more specific problems, such as exponential integrators, is also a shared concern which should be addressed in coming years.

Finally, a question that was raised is whether understanding better the sensitivity of the proposed algorithms, as well as offering ways of estimating the conditioning of a given problem, could help practitioners feel more confident about the use of a chosen algorithm for $f(A)\boldsymbol{b}$—after all, this is one of the aspects that make the BLAS and LAPACK stand out.

There does indeed exist a lot of work on estimating the condition number of the computation of $f(A)$ and $f(A)\boldsymbol{b}$; see, e.g., [51, Chapter 3] for a general overview of the topic. However, these condition number estimates are intimately related to the *Fréchet derivative* $L_f(A, \cdot)$ of $f(A)$, an object that is typically several times more costly to compute than $f(A)$ itself. For the $f(A)$ problem, there exist several algorithms (each for a specific function f) that allow for the computation of $f(A)$ and its Fréchet derivative simultaneously and reuse certain computations in the process [3, 4]. Building on this, [3, Algorithm 7.4] computes the matrix exponential e^A together with a (quite reliable) condition number estimate at a cost of roughly 17 times that of computing e^A alone.[4] Thus, even when using cleverly designed algorithms hand-tailored to a specific function, the overhead induced by the condition number estimator is quite substantial.

Additionally, it is currently unclear how to extend such approaches to, e.g., Krylov subspace algorithms for $f(A)\boldsymbol{b}$, as computing $f(A)\boldsymbol{b}$ and $L_f(A, \cdot)$ has much less in common

[3] Indeed, efforts in this direction are already underway.
[4] The factor 17 can be reduced to 9 if a slightly reduced reliability is acceptable.

than computing $f(A)$ and $L_f(A, \cdot)$. Recently, there has been some progress in Krylov subspace algorithms for low-rank approximations of the Fréchet derivative [57, 58, 61], which might facilitate making a first step in this direction.

2. High-performance and energy-aware computing. The second focus group looked at the challenges surrounding the applications of $f(A)\boldsymbol{b}$ in high-performance computing (HPC). In many domains within the natural and social sciences and engineering disciplines, there is a need to compute $f(A)\boldsymbol{b}$ where A is extremely large and sparse. Such large problems are typically solved using supercomputers, which are machines composed of many nodes with distributed memory, sometimes employing heterogeneous computing hardware. Each node is typically equipped with a number of CPUs and accelerators, such as GPUs, and to achieve peak performance, a routine must make the best use of all available resources.

Aside from those for $A^{-1}\boldsymbol{b}$, numerical methods to compute $f(A)\boldsymbol{b}$ have seldom been used in large-scale parallel applications. One of the most active fields of research in this area is the solution of differential equations using exponential time integrators with Krylov and Leja point methods. Various factors impede the application of certain algorithms developed by the $f(A)\boldsymbol{b}$ community, and we will provide a brief overview in this section.

When the A matrix is large and sparse, it is often impossible to store it explicitly in memory. Fortunately, in many cases one can use "matrix-free" algorithms, which converge without the cost of forming or storing the matrix. These are frequently used in HPC applications because they allow the solution of problems that would otherwise be intractable. In the context of $f(A)\boldsymbol{b}$, most matrix-free algorithms require only the action of the matrix (or an approximation to it) in the form of matrix–vector products. For example, instead of storing the sparse Jacobian J of a vector-valued function $F(\boldsymbol{u})$, its action on a vector can be approximated using the finite difference $J\boldsymbol{b} \approx \left[F(\boldsymbol{u} + \epsilon) - F(\boldsymbol{u})\right]/\epsilon$, where ϵ is a small perturbation [17]. Other matrix-free approaches, such as the complex-step approximation [70] or automatic differentiation [45], are often also used.

Requirements in terms of parallelism and memory storage considerably restrict the choice of possible algorithms. The finite difference approximation of the Jacobian action illustrated above, for example, does not allow operations such as transposition, slicing, or pivoting without a prohibitive computational cost. For the most part, it is not problematic to use a matrix–vector product routine instead of matrices for methods based on the Krylov subspace, the Taylor series, or Leja points. However, difficulties arise when information about the norm or the spectrum of A is needed to 1) compute the parameters that make these methods efficient, or 2) determine the stopping criterion. Without the matrix representation, it can be expensive to compute the operator norm or to estimate eigenvalues. While the spectral radius $\rho(A)$ can be cheaply approximated using the power method on a single CPU, the communication cost in parallel implementations renders this idea inefficient in many HPC applications. Further research will be necessary to develop numerical algorithms more suitable to this kind of problems.

Another important consideration is the capability of an algorithm to scale and optimize energy efficiency as the amount of computing resources increases. Clearly, existing implementations are not yet ready for the future exascale machines, i.e., supercomputers capable of performing at least 10^{18} binary64[5] floating-point operations per second. The problem of implementing Krylov subspace methods efficiently on a GPU has been considered from a theoretical point of view [34], but studies focusing on high-performance implementations suggest that the use of GPUs is most beneficial when A is dense [5], which is not often the case for $f(A)\boldsymbol{b}$ problems, or when A has a very specific sparsity pattern that can be mapped

[5] Previously known as "double precision".

efficiently to GPU architectures [33]. Therefore, software for this problem primarily targets CPUs and can only rely on GPUs in a limited number of cases or for a subset of the relevant operations.

One obstacle is the fact that most implementations target binary64 accuracy, but binary64 arithmetic is not very efficient on GPUs. When using the tensor cores on the latest NVIDIA H100 SXM5 GPUs [64, Table 1], for example, the theoretical peak performance of the 19-bit TensorFloat-32 arithmetic is 494.7 trillion floating-point operations per second (TFLOPS), which improves for BFLOAT16 and binary16 arithmetic (989.4 TFLOPS) and breaks the PFLOPS barrier for the fp8 formats (1978.9 TFLOPS). The peak performance of binary64 arithmetic is almost 30 times slower, with just 66.0 TFLOPS when tensor cores are used for matrix–matrix multiplications. Using low-precision arithmetic is key to harnessing the full potential of GPUs, but only binary64 accuracy is typically sufficient for a range of applications. In other areas of numerical linear algebra, this challenge has been addressed effectively by developing mixed-precision algorithms [1, 55]. Efforts have been made recently in the context of exponential integrators to design such schemes [9], but further research will be necessary.

The issue is not solely with implementations: the algorithms themselves do not seem ready to address large-scale problems either. Many methods rely on matrix operations that do not scale well in a distributed environment. For example, full-basis orthogonalization, central to many Krylov subspace methods, necessitates numerous communication operations (i.e., message passing and synchronization) throughout the computation. This can result in unacceptable latencies, with most processes idly waiting for the slowest process to complete [8]. There are a number of new developments on this front, including but not limited to low-synchronization orthogonalization [12, 21, 22, 23, 24, 41, 42, 65, 71, 76, 78] and s-step methods [19, 20, 75, 76], which can also be combined with one another. Furthermore, a number of well established techniques are being rediscovered as communication-reducing, such as the natural short-term recurrences of Lanczos or batching vectors into tall-skinny matrices (block vectors) to take better advantage of BLAS Level 3 [29, 30]. Sketching, randomization, and low precision can also be leveraged to reduce memory movement by shrinking the size of vectors to be stored and manipulated [6, 7, 27, 50, 66, 77]. The performance of these techniques has been and is being thoroughly explored for linear systems solvers, but their transfer to matrix functions requires a better understanding of their backward stability, how they can be integrated into extended and rational Krylov subspace methods, as well as the conditioning of $f(A)\boldsymbol{b}$ itself (cf. Section 1).

All these issues remain true for the computation of the full matrix $f(A)$ as well.

A significant push in this direction could arise from an increased interest in exponential integrators. However, there are many factors that hinder their use in practical applications. Firstly, they are seldom featured in textbooks and are often absent from university curricula, resulting in many practitioners being unfamiliar with them. In addition, despite their better stability properties, they are generally more complicated to implement than explicit methods. Even when stability is important, practitioners tend to be more attracted by implicit or implicit-explicit schemes, because of the availability of techniques that deal with the stiffness of their particular problems. Furthermore, highly optimized libraries that implement algorithms for solving linear or nonlinear problems are readily available. This is not the case for exponential integrators, and the necessity to implement a parallel solver for the φ-functions often discourages the use of these methods in HPC applications. In recent years, there has been a renewed interest in exponential integrators, and this can largely be attributed to advances in numerical algorithms for the computation of $f(A)\boldsymbol{b}$. While this is an encouraging trend, more work is needed to make these time integration schemes easier to use in applications.

Next steps. Readying current $f(A)\boldsymbol{b}$ work for exascale presents a number of significant challenges, but the community is well equipped to make some progress towards this goal. It is clear that the focus should be on two distinct fronts, since not only the implementations, but also the algorithms, will require significant work in order to leverage the full computational power of next-generation supercomputers.

In terms of rethinking existing algorithms, there is a clear need for reducing the number and frequency of communication operations. In particular, parallel inner products are a known communication bottleneck on distributed systems. Numerical algorithms with high arithmetic intensity should be favored over those requiring a high degree of data movements. Some work has already been done in this area (see, for example, block methods [38], truncated orthogonalization [50, 60, 66] or restarts [2, 16, 32, 36]), but a completely different solution may be needed.

In terms of implementations, a significant challenge is to leverage the untapped potential of GPUs, which, as they become faster and more prevalent, represent an increasingly large share of the overall performance of a supercomputer.

Writing high-performance numerical linear algebra code that can target GPUs presents various difficulties. First and foremost, the variation in capabilities between different models of GPUs, especially those from different vendors, is dramatically larger than the variation between CPUs. As a consequence, there is no unified implementation of the BLAS and LAPACK for GPUs. Vendors provide highly-optimized libraries for their own hardware, but these require very different frameworks, which means that porting an implementation from one GPU to another requires a significant human effort. For example, NVIDIA provides cuBLAS,[6] which is part of the CUDA Toolkit,[7] while AMD provides support through rocBLAS,[8] which is part of the ROCm Platform.[9]

Potential solutions, which include the C++ runtime API HIP,[10] also part of the ROCm Platform, the programming model SYCL[11] [59], and libraries such as MAGMA[12] [73, 74], are not yet mature enough to be used in production code.

At present, the community should attempt to rewrite existing algorithms to ensure optimal performance on HPC architectures. They should seek to minimize communications and use low precision (binary32, binary16, or lower) for the bulk of the computation, switching to higher precision (typically binary64) only when strictly necessary.

For research reproducibility, the $f(A)\boldsymbol{b}$ community should adopt and promote open science best practices. This entails authors sharing the code and data that would allow to replicate the results presented in their publications.

3. Benchmarking.

The last focus group discussed best practices for sharing code and data sets so that they can be easily reused, in accordance with FAIR guidelines.[13] The main goal is to simplify two important steps of the algorithm development process:

- evaluating new implementations on established test problems, and
- comparing their performance with that of existing algorithms in the literature.

A welcome side effect, which comes at no additional cost, is the reproducibility of experimental results. This idea promises to solve a number of problems that commonly arise when new algorithms are proposed in the literature.

[6] https://docs.nvidia.com/cuda/cublas/
[7] https://developer.nvidia.com/cuda-toolkit/
[8] https://rocm.docs.amd.com/projects/rocBLAS/
[9] https://rocm.docs.amd.com
[10] https://rocm.docs.amd.com/projects/HIP/
[11] https://www.khronos.org/sycl/
[12] https://icl.utk.edu/magma/
[13] https://www.go-fair.org/

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Unless the authors decide to compare their proposed new method with all state-of-the-art algorithms for the same problem, it is impossible for the reader to understand how the new method compares with existing alternatives. A new implementation could easily perform worse than a much simpler and well established one, but the reader would have to spend a significant amount of effort to check whether this is the case, especially if the code used in the original publication is not available.

The peer review process can help with this, but there are limitations. Reviewers can recommend that new approaches be compared with the most relevant existing alternatives, but it is difficult to ensure that the comparison is fair, and most journals in numerical analysis and numerical linear algebra do not yet require submission of software or reproducibility of the experimental results. Moreover, as a test set of representative $f(A)\boldsymbol{b}$ problems is not currently available, it is difficult for a reviewer—and for the reader, later on—to make sure that the numerical experiments reported in a publication provide an impartial representation of the merits and drawbacks of new algorithms. Not all methods are suitable for a given choice of f, A, and $\boldsymbol{b}$, and having a battery of tests with clear classes of functions and matrices can help identify what types of problems a certain algorithm can deal with effectively. This can help corroborate theoretical results, in addition to providing a quick and standardized way of comparing all relevant algorithms for a specific choice of f, A, and $\boldsymbol{b}$.

The metrics against which these algorithms should be compared are also not uniquely determined, and authors are free, within reason, to choose the ones that suit them best. In some cases, the metric itself is poorly defined and can depend on a range of factors that are not within the control of who is performing the test. A case in point is runtime, which is commonly used to assess the performance of different implementations on a same test set. Runtime is very sensitive to the hardware configuration, as well as some low-level details of the software libraries being used, so that algorithm α_1 can easily be faster than algorithm α_2 on a machine and slower on another for the same test problem.

When an algorithm cannot be implemented in the most efficient way possible, for example, because of limitations of existing hardware, an appealing alternative is to rely on the number of floating-point operations being performed. This metric is only meaningful for large matrices, and it can be very inaccurate on modern hardware and especially in distributed-computing settings, as it focuses on arithmetic intensity when, in practice, the performance of most algorithms is bounded by the memory bandwidth.

A final difficulty is represented by the lack of clear licensing for code and test problems alike. This prevents reuse and, in many cases, hinders reproducibility of existing results. For more information on research data management in mathematics, see a recent white paper by the Germany-based Mathematical Research Data Initiative (MaRDI) [72].

Next steps. It is a priority for the community to produce a set of representative test cases whereon new and old algorithms can be compared. Ideally, one would want access to a remote facility that is capable of testing submitted implementations against known and unknown benchmark problems, providing overall scores for a number of metrics including accuracy, stability, and runtime performance. Similar services exist for machine learning research [25], where the unknown problems are used to prevent authors from overfitting their models to the test set.

The main difficulty to address in order to deliver this golden standard is to ensure a fair comparison among implementations written using different languages, as Julia, MATLAB/GNU Octave, and Python are all well established in this community, and being able to compare code in these different languages is likely to pose significant challenges in terms of software engineering.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

A more modest but attainable result would be the development of a curated reference collection of test problems. Authors testing their code could then simply choose which parts of the collection to include, and they could justify their choices by pointing out which classes of problems are not suitable in their context. Reviewers could equally rely on such a collection to ensure that authors are providing a fair picture of the merits of their algorithms.

Building and maintaining this infrastructure would come with some logistical challenges. A sufficient number of examples should be included, so that the collection represents the main applications in which $f(A)\boldsymbol{b}$ appears. As test examples can be quite large, the collection might require a hosting service with sufficient storage space and bandwidth, or standardized protocol to point to resources like Zenodo, from which data could be downloaded. A possible remedy for the latter issue is to promote the use of so-called procedural examples, whereby a problem is specified mathematically and the matrix A and vector $\boldsymbol{b}$ can be generated with a desired size or other properties via a script.

It is necessary to ensure that the test cases remain relevant, and that the collection grows and remains representative despite hardware and algorithm improvements that may make problems that are difficult today trivial in the near future. The test cases will likely come from a number of researchers in various research domains, and will have to be collected and added to the collection by a number of volunteers. A standard license—or set of licenses—should be adopted that ensure reproducibility and that fair credit is given to test problem creators and curators.

Although it will not be possible for the community to enforce such a requirement, publishing and advertising a reasonable set of recommendations should be one of the priorities of the group working on this.

4. Conclusions. It is easy to feel overwhelmed looking at the long to-do lists we have outlined. A change of perspective may lessen the anxiety: these are exciting opportunities, some of them even so-called “low-hanging fruit”, and the impact of addressing them is huge, even for such a small field. Matrix functions continue to surface in diverse applications, and many of the techniques developed for $f(A)\boldsymbol{b}$ can cross-pollinate work in linear systems, matrix equations, Fréchet derivatives, and other problems we are not yet aware of. Furthermore, the development of comprehensive surveys and language-agnostic benchmarking workflows for $f(A)\boldsymbol{b}$ can set an example for other mathematical fields that are struggling to modernize and keep up with an ever-increasing publication load. Our primary aim is that this manuscript builds on the momentum of a successful workshop and inspires new, meaningful projects in $f(A)\boldsymbol{b}$ and beyond.

Acknowledgments. The workshop itself was funded in part by DFG Project Number 529315380. We thank all of the workshop participants (excluding the present authors) for their contributions to the substance of this manuscript: Francesca Arrigo, Michele Benzi, Kai Bergermann, Philipp Birken, Liam Burke, Marco Caliari, Benjamin Carrel, Fabio Cassini, Ranjan Kumar Das, Vladimir Druskin, Andreas Frommer, Oswald Knoth, Patrick Kürschner, Thomas Mach, David Persson, Helmut Podhaisky, Michele Rinelli, Jonas Schulze, Roger Sidje, Igor Simunec, Martin Stoll, Mayya Tokman, Manuel Tsolakis, Paul Van Dooren, and Yannis Voet.

REFERENCES

[1] A. ABDELFATTAH, H. ANZT, E. G. BOMAN, E. CARSON, T. COJEAN, J. DONGARRA, A. FOX, M. GATES, N. J. HIGHAM, X. S. LI, J. LOE, P. LUSZCZEK, S. PRANESH, S. RAJAMANICKAM, T. RIBIZEL, B. F. SMITH, K. SWIRYDOWICZ, S. THOMAS, S. TOMOV, Y. M. TSAI, AND U. M. YANG, *A survey*

of numerical linear algebra methods utilizing mixed-precision arithmetic, Int. J. High Performance Computing Applications, 35 (2021), pp. 344–369.

[2] M. AFANASJEW, M. EIERMANN, O. G. ERNST, AND S. GÜTTEL, *Implementation of a restarted Krylov subspace method for the evaluation of matrix functions*, Linear Algebra Appl., 429 (2008), pp. 2293–2314.

[3] 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.

[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] N. AUER, L. EINKEMMER, P. KANDOLF, AND A. OSTERMANN, *Magnus integrators on multicore CPUs and GPUs*, Comput. Phys. Comm., 228 (2018), p. 115–122.

[6] O. BALABANOV AND L. GRIGORI, *Randomized Gram–Schmidt Process with Application to GMRES*, SIAM J. Sci. Comput., 44 (2022), pp. A1450–A1474.

[7] ———, *Randomized block Gram-Schmidt process for solution of linear systems and eigenvalue problems*, e-print 2111.14641, arXiv, 2023.

[8] G. BALLARD, E. C. CARSON, J. W. DEMMEL, M. HOEMMEN, N. KNIGHT, AND O. SCHWARTZ, *Communication lower bounds and optimal algorithms for numerical linear algebra*, Acta Numer., 23 (2014), pp. 1–155.

[9] C. J. BALOS, S. ROBERTS, AND D. J. GARDNER, *Leveraging mixed precision in exponential time integration methods*, in 2023 IEEE High Performance Extreme Computing Conference (HPEC), 2023, pp. 1–8.

[10] M. BENZI AND V. SIMONCINI, *Decay bounds for functions of Hermitian matrices with banded or Kronecker structure*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 1263–1282.

[11] ———, *Approximation of functions of large matrices with Kronecker structure*, Numer. Math., 135 (2016), pp. 1–26.

[12] D. BIELICH, J. LANGOU, S. THOMAS, K. ŚWIRYDOWICZ, I. YAMAZAKI, AND E. G. BOMAN, *Low-synch Gram–Schmidt with delayed reorthogonalization for Krylov solvers*, Parallel Computing, 112 (2022), p. 102940.

[13] M. A. BOTCHEV, V. GRIMM, AND M. HOCHBRUCK, *Residual, restarting, and Richardson iteration for the matrix exponential*, SIAM J. Sci. Comput., 35 (2013), pp. A1376–A1397.

[14] M. A. BOTCHEV, L. KNIZHNERMAN, AND M. SCHWEITZER, *Krylov subspace residual and restarting for certain second order differential equations*, SIAM J. Sci. Comput., (2023), pp. S223–S253.

[15] M. A. BOTCHEV, L. KNIZHNERMAN, AND E. E. TYRTYSHNIKOV, *Residual and restarting in Krylov subspace evaluation of the φ function*, SIAM J. Sci. Comput., 43 (2021), pp. A3733–A3759.

[16] M. A. BOTCHEV AND L. A. KNIZHNERMAN, *ART: Adaptive residual-time restarting for Krylov subspace matrix exponential evaluations*, J. Comput. Appl. Math., 364 (2020), p. 112311.

[17] P. N. BROWN, H. F. WALKER, R. WASYK, AND C. S. WOODWARD, *On using approximate finite differences in matrix-free Newton–Krylov methods*, SIAM Journal on Numerical Analysis, 46 (2008), pp. 1892–1911.

[18] L. BURKE AND S. GÜTTEL, *Krylov subspace recycling with randomized sketching for matrix functions*, arxiv:2308.02290 [math.NA], Aug. 2023.

[19] E. CARSON, T. GERGELITS, AND I. YAMAZAKI, *Mixed precision s-step Lanczos and conjugate gradient algorithms*, Numer. Linear Algebra Appl., 29 (2022), p. e2425.

[20] E. C. CARSON, *An adaptive s-step conjugate gradient algorithm with dynamic basis updating*, Appl. Math., 65 (2020), pp. 123–151.

[21] E. C. CARSON, K. LUND, Y. MA, AND E. OKTAY, *On the loss of orthogonality of low-synchronization variants of reorthogonalized block Gram-Schmidt*, tech. rep., In preparation, 2024.

[22] E. C. CARSON, K. LUND, AND E. OKTAY, *Reorthogonalized Pythagorean variants of block classical Gram Schmidt*, tech. rep., In preparation, 2024.

[23] E. C. CARSON, K. LUND, AND M. ROZLOŽNÍK, *The stability of block variants of classical Gram-Schmidt*, SIAM J. Matrix Anal. Appl., 42 (2021), pp. 1365–1380.

[24] E. C. CARSON, K. LUND, M. ROZLOŽNÍK, AND S. THOMAS, *Block Gram-Schmidt algorithms and their stability properties*, Linear Algebra Appl., 638 (2022), pp. 150–195.

[25] R. CHAN, K. LIS, S. UHLEMEYER, H. BLUM, S. HONARI, R. SIEGWART, P. FUA, M. SALZMANN, AND M. ROTTMANN, *SegmentMeIfYouCan: A benchmark for anomaly segmentation*, in Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, J. Vanschoren and S. Yeung, eds., vol. 1, 2021, p. 13.

[26] T. CHEN AND E. HALLMAN, *Krylov-aware stochastic trace estimation*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 1218–1244.

[27] A. CORTINOVIS, D. KRESSNER, AND Y. NAKATSUKASA, *Speeding up Krylov subspace methods for computing $f(a)b$ via randomization*, arXiv:2212.12758 [math.NA], Dec. 2022. Revised June 2023.

[28] E. DEADMAN AND N. J. HIGHAM, *Testing matrix function algorithms using identities*, ACM Trans. Math. Software, 42 (2016), pp. 4:1–4:15.

[29] N.-A. DREIER AND C. ENGWER, *Strategies for the Vectorized Block Conjugate Gradients method*, in Numer. Math. Adv. Appl. ENUMATH 2019, F. J. Vermolen and C. Vuik, eds., vol. 139 of Lect. Notes Comput.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Sci. Eng., Springer, Cham, 2020, pp. 381–388.

[30] N.-A. Dreier and C. Engwer, *A Hardware-aware and Stable Orthogonalization Framework*, e-print 2204.13393, arXiv, 2022.

[31] V. Druskin, *On monotonicity of the Lanczos approximation to the matrix exponential*, Linear Algebra and its Applications, 429 (2008), pp. 1679–1683.

[32] M. Eiermann and O. G. Ernst, *A restarted Krylov subspace method for the evaluation of matrix functions*, SIAM J. Numer. Anal., 44 (2006), pp. 2481–2504.

[33] L. Einkemmer and A. Ostermann, *Exponential integrators on graphic processing units*, in Proceedings of the 2013 International Conference on High Performance Computing & Simulation (HPCS), Institute of Electrical and Electronics Engineers, July 2013.

[34] M. E. Farquhar, T. J. Moroney, Q. Yang, and I. W. Turner, *GPU accelerated algorithms for computing matrix function vector products with applications to exponential integrators and fractional diffusion*, SIAM J. Sci. Comput., 38 (2016), p. C127–C149.

[35] A. Frommer, S. Güttel, and M. Schweitzer, *Convergence of restarted Krylov subspace methods for Stieltjes functions of matrices*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 1602–1624.

[36] A. Frommer, S. Güttel, and M. Schweitzer, *Efficient and stable Arnoldi restarts for matrix functions based on quadrature*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 661–683.

[37] A. Frommer, K. Kahl, M. Schweitzer, and M. Tsolakis, *Krylov subspace restarting for matrix Laplace transforms*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 693–717.

[38] A. Frommer, K. Lund, and D. B. Szyld, *Block Krylov subspace methods for functions of matrices*, Electron. Trans. Numer. Anal., 47 (2017), pp. 100–126.

[39] A. Frommer and M. Schweitzer, *Error bounds and estimates for Krylov subspace approximations of Stieltjes matrix functions*, BIT Numerical Mathematics, 56 (2015), pp. 865–892.

[40] A. Frommer and V. Simoncini, *Matrix functions*, in Model Order Reduction: Theory, Research Aspects and Applications, vol. 13 of Math. Ind., Springer-Verlag, Aug. 2008, pp. 275–303.

[41] T. Fukaya, R. Kannan, Y. Nakatsukasa, Y. Yamamoto, and Y. Yanagisawa, *Shifted Cholesky QR for computing the QR factorization of ill-conditioned matrices*, SIAM J. Sci. Comput., 42 (2020), pp. A477–A503.

[42] T. Fukaya, Y. Nakatsukasa, Y. Yanagisawa, and Y. Yamamoto, *CholeskyQR2: A Simple and Communication-Avoiding Algorithm for Computing a Tall-Skinny QR Factorization on a Large-Scale Parallel System*, in 2014 5th Workshop Latest Adv. Scalable Algorithms Large-Scale Syst., 2014, pp. 31–38.

[43] S. Gaudreault, K. Lund, and M. Schweitzer, *Introduction to focus groups topics*, 2023. https://indico3.mpi-magdeburg.mpg.de/event/30/attachments/182/265/focus_group_slides.pdf.

[44] S. Gaudreault, G. Rainwater, and M. Tokman, *KIOPS: A fast adaptive Krylov subspace solver for exponential integrators*, J. Comput. Phys., 372 (2018), pp. 236–255.

[45] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM, 2008.

[46] S. Güttel, *Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection*, GAMM Mitteilungen, 36 (2013), pp. 8–31.

[47] S. Güttel, D. Kressner, and K. Lund, *Limited-memory polynomial methods for large-scale matrix functions*, GAMM Mitteilungen, 43 (2020), p. e202000019.

[48] S. Güttel and L. Knizhnerman, *A black-box rational Arnoldi variant for Cauchy–Stieltjes matrix functions*, BIT, 53 (2013), pp. 595–616.

[49] S. Güttel and M. Schweitzer, *A comparison of limited-memory Krylov methods for Stieltjes functions of Hermitian matrices*, SIAM J. Matrix Anal. Appl., 42 (2021), p. 83–107.

[50] S. Güttel and M. Schweitzer, *Randomized sketching for Krylov approximations of large-scale matrix functions*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 1073–1095.

[51] N. J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[52] N. J. Higham and A. H. Al-Mohy, *Computing matrix functions*, Acta Numerica, 19 (2010), pp. 159–208.

[53] N. J. Higham and E. Deadman, *A catalogue of software for matrix functions. Version 1.0*, MIMS EPrint 2014.8, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, Feb. 2014.

[54] N. J. Higham and E. Hopkins, *A catalogue of software for matrix functions. Version 3.0*, MIMS EPrint 2020.7, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, Mar. 2020.

[55] N. J. Higham and T. Mary, *Mixed precision algorithms in numerical linear algebra*, Acta Numerica, 31 (2022), pp. 347–414.

[56] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numerica, 19 (2010), pp. 209–286.

[57] 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), p. e2401.

[58] P. Kandolf and S. D. Relton, *A block Krylov method to compute the action of the fréchet derivative of a matrix function on a vector with applications to condition number estimation*, SIAM J. Sci. Comput., 39 (2017), pp. A1416–A1434.

[59] Khronos© SYCL™ Working Group, *SYCL™ Specification*, no. Git revision: tags/SYCL-1.2.1/final-rev7-0-g7145c7006, The Khronos© Group, Apr. 2020. Version 1.2.1.

[60] A. Koskela, *Approximating the matrix exponential of an advection-diffusion operator using the incomplete orthogonalization method*, in Numerical Mathematics and Advanced Applications-ENUMATH 2013: Proceedings of ENUMATH 2013, the 10th European Conference on Numerical Mathematics and Advanced Applications, Lausanne, August 2013, Springer, 2014, pp. 345–353.

[61] D. Kressner, *A Krylov subspace method for the approximation of bivariate matrix functions*, Structured Matrices in Numerical Linear Algebra: Analysis, Algorithms and Applications, (2019), pp. 197–214.

[62] S. Massei and L. Robol, *Rational Krylov for Stieltjes matrix functions: Convergence and pole selection*, BIT, 61 (2020), pp. 237–273.

[63] B. V. Minchev and W. Wright, *A review of exponential integrators for first order semi-linear problems*, (2005).

[64] NVIDIA Corporation, *NVIDIA H100 tensor core GPU architecture*, tech. rep., 2022.

[65] E. Oktay and E. Carson, *Using Mixed Precision in Low-Synchronization Reorthogonalized Block Classical Gram-Schmidt*, PAMM, 23 (2023), p. e202200060.

[66] D. Palitta, M. Schweitzer, and V. Simoncini, *Sketched and truncated polynomial Krylov subspace methods: Evaluation of matrix functions*, arXiv:2306.06481 [math.NA], 2023.

[67] D. Persson and D. Kressner, *Randomized low-rank approximation of monotone matrix functions*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 894–918.

[68] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, Philadelphia, 2nd ed ed., 2003.

[69] M. Schweitzer, *Restarting and error estimation in polynomial and extended Krylov subspace methods for the approximation of matrix functions*, PhD thesis, Fakultät für Mathematik und Naturwissenschaften, Bergische Universität Wuppertal, 2015.

[70] W. Squire and G. Trapp, *Using complex variables to estimate derivatives of real functions*, SIAM review, 40 (1998), pp. 110–112.

[71] K. Świrydowicz, J. Langou, S. Ananthan, U. Yang, and S. Thomas, *Low synchronization Gram–Schmidt and generalized minimal residual algorithms*, Numer Linear Algebra Appl, 28 (2021), p. e2343.

[72] The MaRDI Consortium, *Research data management planning in mathematics*, e-print, 2023.

[73] S. Tomov, J. Dongarra, and M. Baboulin, *Towards dense linear algebra for hybrid GPU accelerated manycore systems*, Parallel Comput., 36 (2010), p. 232–240.

[74] S. Tomov, R. Nath, H. Ltaief, and J. Dongarra, *Dense linear algebra solvers for multicore with GPU accelerators*, in Proceedings of the 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), Institute of Electrical and Electronics Engineers, Apr. 2010.

[75] I. Yamazaki, E. Carson, and B. Kelley, *Mixed Precision s-step Conjugate Gradient with Residual Replacement on GPUs*, in 2022 IEEE Int. Parallel Distrib. Process. Symp. IPDPS, 2022, pp. 886–896.

[76] I. Yamazaki, S. Thomas, M. Hoemmen, E. G. Boman, K. Świrydowicz, and J. J. Eilliot, *Low-synchronization orthogonalization schemes for s-step and pipelined Krylov solvers in Trilinos*, in Proc. 2020 SIAM Conf. Parallel Process. Sci. Comput. PP, 2020, pp. 118–128.

[77] I. Yamazaki, S. Tomov, J. Kurzak, J. J. Dongarra, and J. L. Barlow, *Mixed-precision block Gram Schmidt orthogonalization*, Proc. ScalA 2015 6th Workshop Latest Adv. Scalable Algorithms Large-Scale Syst. - Held Conjunction SC 2015 Int. Conf. High Perform. Comput. Netw. Storage Anal., (2015).

[78] Q. Zou, *A flexible block classical Gram–Schmidt skeleton with reorthogonalization*, Numerical Linear Algebra with Applications, 30 (2023), p. e2491.