Randomized Krylov Methods for Matrix Functions

Vilhelm Peterson Lithell

KTH Royal Institute of Technology
<vipl@kth.se>

Abstract

Let $A \in \mathbb{C}^{N \times N}$, $\boldsymbol{b} \in \mathbb{C}^N$. We consider the problem of computing the action of a matrix function, i.e. $f(A)\boldsymbol{b}$, by means of a randomized Krylov method. Initially, we introduce some basic concepts needed for the derivation of the method. We then derive the so-called *sFOM* method, a randomized analog of the classical FOM method for matrix functions. We also make explicit a stopping condition for this algorithm, incorporating the most important qualities of the main algorithm with regards to memory access. Finally, we provide a suite of numerical experiments, validating the algorithm and implementation, and also highlighting some potential pit-falls. We conclude by an illustrative example suggesting a possible application of the method to exponential integrators. The source code for these experiments is also made publicly available.

Contents

1 Introduction **1**

2 Derivation of sFOM **2**
2.1 Briefly on Random Matrices 2
2.2 The sFOM Approximant 3
2.3 A Closed Form Representation 4
2.4 Stopping Criterion 6

3 Numerical Experiments **8**
3.1 Basic Verification 8
3.1.1 Convergence 8
3.1.2 Stopping Criterion 10
3.1.3 Effect of Sketching Parameter 11
3.2 Example Application 12
3.2.1 Exponential Integrators 12
3.2.2 A Quasi-Linear One-Dimensional Parabolic PDE 13

4 Conclusion and Outlook **15**

References **15**

1 Introduction

In the field of numerical linear algebra, randomized methods are increasingly becoming an indispensable tool for performance-hungry computation. In many applications, these methods have unlocked solutions to previously intractable problems. Familiar examples of such methods include randomized methods for matrix decomposition and low-rank approximation that have in many instances become standard techniques during the last decade. See for instance the review paper [5]. Such techniques utilize randomized reduction to extract the most important actions of matrices or operators, and in this work we will consider similar ideas applied to iterative Krylov methods. More precisely, our application of these randomization techniques will be concerned with computing the action of a matrix function. The task of computing this action on a vector is a basic task arising in many areas of computational science. In this work, we follow the approach suggested by Güttel and Schweitzer [4] and employ accelerated computations by means of a randomized Krylov subspace method for the $f(A)\boldsymbol{b}$ problem.

In many applications, the matrix A is sparse and very large, for instance as in discretization-matrices of PDEs, or Laplacians in network problems. In classical Krylov methods for the $f(A)\boldsymbol{b}$ problem, the cost of orthogonalization can pose a significant bottleneck for computation, especially on modern architectures where parallelization is ubiquitous. For these applications, the communication time is often limiting to the application, and orthogonalization here imposes a global synchronization step, making any decrease in the orthogonalization time required very welcome. In addition, if the matrix in question is very large, frequently accessing the full Krylov basis may be a limiting factor to the speed of the approach. In the current method, these problems are alleviated by means of a random embedding matrix, which significantly reduces the cost of orthogonalization as a result of the formulation of the method. Additionally, the method requires full access to the Krylov basis only in the final iteration, and in some special cases; never.

We will focus on developing a randomized analog of the *full orthogonalization method* for matrix functions. Moreover, [4] provides additional derivations of a randomized analog of the classical GMRES method, in the context of matrix functions. The reader is referred to this resource for additional information on the randomized GMRES method.

The report is structured as follows. In Section 2 we present the derivation of the main algorithm, along with references to additional resources and a short discussion on random matrices for applications in Krylov methods. Section 3 presents some numerical experiments performed with the method, and also provides additional details on the implementation. We also explicitly implement a stopping condition suggested but not used in [4]. Finally, in Section 4 we conclude by summarizing our results, and provide some outlook for future continuation of the current work.

2 Derivation of sFOM

Consider a function $f : \mathbb{C} \mapsto \mathbb{C}$, and let $A \in \mathbb{C}^{N\times N}$. Let $\Lambda(A)$ denote the spectrum of A, and suppose Γ is a Jordan curve enclosing the negated spectrum of A, $-\Lambda(A)$. Finally, suppose f is holomorphic in and on Γ. Then f permits the integral representation

$$f(A)\boldsymbol{b} = \int_\Gamma f(t)(tI + A)^{-1}\boldsymbol{b}d\mu(t)\,. \tag{2.1}$$

This integral form is the basis of the method considered here. As in classical Krylov methods for matrix functions, we will focus our attention on the shifted linear system arising in (2.1). The approximation of the solution to this system will be done by means of randomized Krylov methods, and we will focus on the *full orthogonalization method*, henceforth *FOM*, coupled with a random *sketching matrix* to accelerate the computations.

2.1 Briefly on Random Matrices

A very important element in our method is a random matrix, called a *sketching matrix*, that will be used to facilitate accelerated computations, at the cost of some accuracy. The sketching matrix will be a (possibly) complex matrix $S \in \mathbb{C}^{s\times N}$, where importantly $s \ll N$. This matrix can be viewed as an embedding from $\mathbb{R}^N$ to $\mathbb{R}^s$ [1], and we can estimate inner products on $\mathbb{R}^N$ by

$$\langle x, y\rangle \approx \langle Sx, Sy\rangle\,,$$

for $x, y \in \mathbb{R}^N$. Presently, we relate some definitions concerning a certain class of subspace embedding that will be of particular importance to us.

Definition 2.1.1. Let $0 < \varepsilon < 1$. The sketching matrix $S \in \mathbb{C}^{s\times N}$ is said to be an ε-subspace embedding of $V \subset \mathbb{R}^N$ if

$$\langle \boldsymbol{u}, \boldsymbol{v}\rangle - \varepsilon\|\boldsymbol{u}\|\|\boldsymbol{v}\| \le \langle S\boldsymbol{u}, S\boldsymbol{v}\rangle \le \langle \boldsymbol{u}, \boldsymbol{v}\rangle + \varepsilon\|\boldsymbol{u}\|\|\boldsymbol{v}\|\,, \tag{2.2}$$

for all $\boldsymbol{u}, \boldsymbol{v} \in V$.

Notice that this definition essentially establishes a class of embedding matrices with a certain quality in the inner-product approximation mentioned above. Definition 2.1.1 will be important later, for when we need to perform estimates on the size of the iterates in our method.

In he considered approach, a crucial observation is that the embedding matrix S can be drawn at random. In our case however, V in Definition 2.1.1 will correspond to a Krylov basis, and we will not have a-priori knowledge of the quality of the embedding, since we do not have access to the full basis. Therefore, we would like to draw a random matrix that satisfies (2.2) with high probability, say $1 - \delta$, where $\delta \ll 1$.

Definition 2.1.2. The random matrix S, taking values in $\mathbb{C}^{s\times N}$, is called a (ε,δ) *oblivious* subspace embedding for any fixed subspace $V\subset\mathbb{R}^N$ if it is an ε-embedding for this space with probability at least $1-\delta$, for $\delta\in[0,1]$.

There are a number of distributions that are known to satisfy this embedding property, among them; Gaussians, Rademacher-distributions, and randomly sub-sampled Fourier matrices [1]. In the numerical experiments performed in this work, S will be a randomly sub-sampled discrete cosine transform, following its usage to good effect in [4]. See also [8]. Related embedding matrices are reviewed in some detail in [11].

To construct our embedding matrix, we let $S\in\mathbb{C}^{s\times N}$ be defined by

$$S=\sqrt{\frac{n}{s}}DFE\,, \tag{2.3}$$

with $D\in\mathbb{C}^{s\times N}$, $F\in\mathbb{C}^{N\times N}$, and $E\in\mathbb{C}^{N\times N}$. Here D is a so-called diagonal projector onto our m-dimensional space. In our case, this matrix will be selecting s rows of the product FE at random. Furthermore, E is a diagonal matrix with independent Rademacher distributed entries. Finally, F is the discrete cosine transform matrix.

Now, following [8], to obtain empirical subspace embedding quality ε (in the sense of expectation), we take the embedding dimension to be $s=d/\varepsilon^2$, where d will eventually correspond to the dimension of our approximation space. Hence, the so-called *sketching parameter*, the dimension s, will be taken to be twice the maximum number of iterations in our method, cf. Section 2. This choice yields an empirical subspace embedding quality of $1/\sqrt{2}$. For our purposes, this will be more than sufficient, cf. Section 3.1.

It is noted in [8] that for "worst-case problems", a more careful choice of subspace embedding might be necessary. However, for the analysis and experiments contained in this work, the choice (2.3) is satisfactory. We also note that (2.3) will, for performance reasons, naturally make use of the fast cosine transform.

2.2 The sFOM Approximant

Returning to our main goal of developing an approximation to $f(A)\boldsymbol{b}$, we turn our attention to the shifted linear system in (2.1), and recall the standard tool of all Krylov methods; the Arnoldi relation. Applying m iterations of Arnoldi's method to the pair A, $\boldsymbol{b}$, yields the Arnoldi factorization

$$AV_m=V_mH_m+(h_{m+1})_m\boldsymbol{v_{m+1}}\boldsymbol{e}_m^T\,, \tag{2.4}$$

where $V_m=[\boldsymbol{v_1},\ldots,\boldsymbol{v_m}]\in\mathbb{C}^{N\times m}$ is a matrix containing an orthonormal basis for the m:th Krylov space, $\mathcal{K}_m(A,\boldsymbol{b})$. Here H_m is an unreduced upper Hessenberg matrix. Proceeding, we follow the derivation in [4], and consider the classical FOM approximant, defined for a possibly nonorthonormal Krylov basis V_m by

$$\boldsymbol{f}_m=V_mf(H_m)V_m^\dagger\boldsymbol{b}\,, \tag{2.5}$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose inverse, and $H_m=V_m^\dagger AV_m$. It is vital to note that H_m is small with respect to the size of the problem, and so its evaluation under f is cheap. Continuing, we recall the integral form (2.1), and write (2.5) as

$$\begin{aligned} \boldsymbol{f}_m = V_m \int_\Gamma f(t)(tI + H_m)^{-1} d\mu(t) V_m^\dagger \boldsymbol{b} &= \int_\Gamma f(t) \|\boldsymbol{b}\| V_m (tI + H_m)^{-1} \boldsymbol{e}_1 d\mu(t) \\ &:= \int_\Gamma f(t) \boldsymbol{x}(t) d\mu(t) \,. \end{aligned} \tag{2.6}$$

We find that the integrand contains the approximants for the shifted linear system in (2.1):

$$\boldsymbol{x}_m(t) := \|\boldsymbol{b}\| V_m (tI + H_m)^{-1} \boldsymbol{e}_1 \,, \tag{2.7}$$

and we also write $\boldsymbol{x}_m(t) := V_m \boldsymbol{y}_m(t)$, for some $\boldsymbol{y}_m(t)$.

Again, following [4], we relax the problem by imposing a weak orthogonality condition on the residual. More precisely, the residual of the approximant is explicitly given by

$$\boldsymbol{r}_m(t) = \boldsymbol{b} - (tI + A)\boldsymbol{x}_m(t) \,, \tag{2.8}$$

and we classically require $V_m^H \boldsymbol{r}_m(t) = 0$, recognized as a Galerkin orthogonality condition. Instead of imposing this condition, we instead only require that the sketched residual be orthogonal to the sketched span of V_m, i.e. we take the following condition on the residual

$$(SV_m)^H \left[S\boldsymbol{b} - S(tI + A)\widehat{\boldsymbol{x}}_m(t) \right] = \boldsymbol{0} \,, \tag{2.9}$$

where $\widehat{\boldsymbol{x}} = V_m \widehat{\boldsymbol{y}}$ for some $\widehat{\boldsymbol{y}}$. Definition (2.1.1) provides some motivation for this condition. Since sketching in a sense corresponds to approximating inner product via the embedding S, the use of sketching can be interpreted as providing an approximate orthogonality condition on the residual. We will prefer working with $\widehat{\boldsymbol{y}}_m(t)$ for reasons that will be evident later. Hence we can write equivalently, assuming the inverse is well-defined,

$$\widehat{\boldsymbol{y}}_m(t) = \left[(SV_m)^H (tSV_m + SAV_m) \right]^{-1} (SV_m)^H S\boldsymbol{b} \,. \tag{2.10}$$

Now, comparing with (2.6), we are motivated to use (2.10) to make the following definition.

Definition 2.2.1. The sketched FOM (*sFOM*) approximant is given by

$$\widehat{\boldsymbol{f}}_m := \int_\Gamma f(t) \widehat{\boldsymbol{x}} d\mu(t) = V_m \int_\Gamma f(t) \left[(SV_m)^H (tSV_m + SAV_m) \right]^{-1} d\mu(t) (SV_m)^H S\boldsymbol{b} \,. \tag{2.11}$$

We note that the approximant (2.11) is equivalent to the regular FOM approximant if $S = I$.

We stress that the derivation of (2.11) assumed no orthogonality of the Krylov basis, and can therefore be constructed using a truncated Arnoldi iteration, significantly reducing the time that needs to be spent in orthogonalization, as compared to the generation of an orthonormal Krylov basis. Instead, the orthogonality of the sketched residual with respect to the sketched span of the Krylov basis is imposed explicitly in (2.9).

2.3 A Closed Form Representation

In its current form, (2.11) is defined through a complex path integral, and so would require quadrature for its evaluation. However, it is possible to derive a closed form expression for $\widehat{\boldsymbol{f}}_m$ using only basic manipulation of the integrand. Notice that so long as (2.11) is well defined, with respect to the inverse, we must have that (SV_m) is non-singular, and hence $(SV_m)^H(SV_m)$ must

also be non-singular. We can then rewrite the expression in square brackets in the integrand of (2.11) as

$$\begin{aligned}\left[(SV_m)^H(tSV_m + SAV_m)\right]^{-1} &= \left[t(SV_m)^H SV_m + (SV_m)^H SAV_m\right]^{-1} \\ &= \left[tV_m^H S^H SV_m + V_m^H S^H SAV_m\right]^{-1} \\ &= \left[V_m^H S^H SV_m\right]^{-1}\left[tI + V_m^H S^H SAV_m (V_m^H S^H SV_m)^{-1}\right]^{-1} .\end{aligned} \tag{2.12}$$

Inserting this into the definition of sFOM, we get

$$\begin{aligned}\widehat{\boldsymbol{f}}_m &= V_m \int_\Gamma f(t)\left[(SV_m)^H(tSV_m + SAV_m)\right]^{-1} d\mu(t)(SV_m)^H S\boldsymbol{b} \\ &= V_m\left[V_m^H S^H SV_m\right]^{-1}\int_\Gamma f(t)\left[tI + V_m^H S^H SAV_m(V_m^H S^H SV_m)^{-1}\right]^{-1} d\mu(t)(SV_m)^H S\boldsymbol{b} \\ &= V_m\left[V_m^H S^H SV_m\right]^{-1} f\left(V_m^H S^H SAV_m(V_m^H S^H SV_m)^{-1}\right)(SV_m)^H S\boldsymbol{b} ,\end{aligned} \tag{2.13}$$

and we arrive at a closed form expression for the sFOM approximant. However, we can further simplify this expression under some additional assumption on the structure of the sketched Krylov basis. In particular, the approximant is, like mentioned previously, completely independent of the choice of basis, as long as it satisfies $\text{span}\{V_m\} = \mathcal{K}_m(A, \boldsymbol{b})$. Following the particular choice of SV_m suggested in [1][4][11], we require that our sketched basis be orthonormal, i.e.

$$(SV_m)^H(SV_m) = I , \tag{2.14}$$

and applying this condition to (2.13) yields the much simplified form

$$\widehat{\boldsymbol{f}}_m = V_m f(V_m^H S^H SAV_m) V_m^H S^H S\boldsymbol{b} . \tag{2.15}$$

We need however still find a way to impose the condition (2.14). In [1] this is done during the orthogonalization process of Arnoldi's iteration, but instead we will proceed by way of [4], where this condition is imposed at a lower cost. The idea is to satisfy this condition retrospectively, by letting $SV_m = Q_m R_m$ be a thin QR-decomposition of the m:th sketched Krylov basis SV_m. Then, the following substitutions will ensure the orthonormality of the sketched basis:

$$SV_m \leftarrow Q_m, \quad SAV_m \leftarrow (SAV_m)R_m^{-1}, \quad V_m \leftarrow V_m R_m^{-1} . \tag{2.16}$$

It is stressed that these substitutions should be done only implicitly where possible. These substitutions ensure our sketched basis satisfies (2.14), and so completes our derivation of the sFOM method. The full algorithm can be found in Algorithm 1. In this work, we do not make explicit any error- or convergence analysis, and instead the reader is referred to the main reference [4].

The framework presented in this derivation is rather general, and similar approaches can be taken to derive randomized versions of other Krylov methods. For instance, in [4], a derivation similar to the one presented above yields a randomized version of GMRES, although no closed form is obtained. Another similar approach to Krylov subspace methods for matrix functions has recently been suggested in [3]. Yet another, specialized, approach can also be found in [10].

Algorithm 1 sFOM for $f(A)\boldsymbol{b}$

Input: $A \in \mathbb{C}^{N\times N}$, $\boldsymbol{b} \in \mathbb{C}^N$, $f : \mathbb{C}^{N\times N} \mapsto \mathbb{C}^{N\times N}$, $N \gg s \in \mathbb{Z}^+$, `tol`, `max_iter`
Output: $\widehat{\boldsymbol{f}} \approx f(A)\boldsymbol{b}$
for $m = 1$ to `max_iter` **do**
  Draw sketching matrix $S \in \mathbb{C}^{s\times N}$
  Generate non-orthogonal Krylov basis V_m, and SV_m, SAV_m
  Compute thin QR-decomposition of SV_m
  Compute $\widehat{\boldsymbol{q}}_m = R_m^{-1} f(Q_m^H SAV_m R^{-1}) Q_m^H S\boldsymbol{b}$
  Use $\widehat{\boldsymbol{q}}_m$ to evaluate `stop_crit`
  if `stop_crit` $<$ `tol` **then**
    Compute $\widehat{\boldsymbol{f}}_m = V_m \widehat{\boldsymbol{q}}_m$
    Return $\widehat{\boldsymbol{f}}_m$
  end if
end for

Notice that (1) only requires the evaluation of the full approximant when the method has converged, and so we need full access to the Krylov basis only at the final iteration. This is one of the main benefits of the method with respect to the memory requirements of the compute. Furthermore, for some applications, it is reasonable to assume that only some components of the resulting vector need to be accessed. In this case, we do not even need access to the full Krylov basis, but only parts of it. This case would of course lower the memory requirements even more.

2.4 Stopping Criterion

At the moment, we have no way of knowing if our method has converged our not. As with all iterative methods, we need a stopping criterion to be able to judge when to stop iterating the method. For the current method, this is particularly important, since it appears it displays somewhat irregular convergence. This is especially pronounced if the method is further iterated after convergence, where it has a tendency to inflate errors induced by rounding. In some cases, this leads to completely erroneous approximations (see Section 3.1).

To develop a robust stopping criterion, we will use an approach that is very common for iterative methods. This particular method is due to the main reference [4], where it was suggested, but never implemented in their tests. As an indicator of convergence, we usually study the difference between iterates, $\|\widehat{\boldsymbol{f}}_{m+d} - \widehat{\boldsymbol{f}}_m\|$, for some small integer d. If this difference at any point is found to be smaller than some tolerance, `tol`, the method is deemed to have converged, and we stop iterating. Keeping in mind that one of the main aims of the current method is to alleviate the excessive memory access associated with classical Krylov methods, especially for very large problems, we would like to not explicitly access $\widehat{\boldsymbol{f}}_m$, since this requires full access to $\mathcal{K}_m(A, \boldsymbol{b})$.

Following [4], we use the fact that $\widehat{\boldsymbol{f}}_m, \widehat{\boldsymbol{f}}_{m+d} \in \mathcal{K}_{m+d}(A, \boldsymbol{b})$, together with the fact that our sketching matrix S is an ε-embedding of this space (cf. Definition 2.1.1). Then we can write

$$\left\|\widehat{\boldsymbol{f}}_{m+d} - \widehat{\boldsymbol{f}}_m\right\| \leq \frac{1}{\sqrt{1-\varepsilon}} \left\|S\left(\widehat{\boldsymbol{f}}_{m+d} - \widehat{\boldsymbol{f}}_m\right)\right\| . \tag{2.17}$$

Now, writing $\widehat{\boldsymbol{f}}_m = V_m \widehat{\boldsymbol{q}}_m$ (cf. Algorithm 1), and substituting this above, yields

$$\left\|\widehat{\boldsymbol{f}}_{m+d} - \widehat{\boldsymbol{f}}_m\right\| \leq \frac{1}{\sqrt{1-\varepsilon}} \left\| SV_{m+d} \left(\widehat{\boldsymbol{q}}_{m+d} - \begin{bmatrix} \widehat{\boldsymbol{q}}_m \\ \boldsymbol{0}_d \end{bmatrix} \right) \right\| . \tag{2.18}$$

Since SV_m is significantly smaller, in the sense of the dimension of the columns, than the full basis V_m, this expression can be evaluated cheaply, operating only on small matrices and vectors. However, since the subspace embedding-quality, ε, is unknown, we need to estimate it. A simple estimate of this parameter can be obtained by noticing that (2.2) implies that

$$(1-\varepsilon) \leq \frac{\|S\boldsymbol{v}_{m+d}\|^2}{\|\boldsymbol{v}_{m+d}\|^2} \leq (1+\varepsilon) . \tag{2.19}$$

Using $\|\boldsymbol{v}_{m+d}\| = 1$, we have

$$\frac{1}{\sqrt{1+\varepsilon}} \leq \frac{1}{\|S\boldsymbol{v}_{m+d}\|} \leq \frac{1}{\sqrt{1-\varepsilon}} , \tag{2.20}$$

and we can use the quantity $1/\|S\boldsymbol{v}_{m+d}\|$ as an estimate on the unknown quantity $1/\sqrt{1-\varepsilon}$. For reasonable embedding qualities ε, this interval is relatively tight, yielding a good estimate. However, it deteriorates for embeddings of lesser quality. To see this, notice that for an embedding quality approaching $\varepsilon = 1$, i.e. a very low quality embedding, the interval containing the approximating quantity becomes unbounded. We then no longer have any guarantee that our approximation is reasonably close to the quantity we are trying to estimate, and so the strategy presented here has the potential to fail.

Then, we will use the following estimate for the difference between iterates:

$$\left\|\widehat{\boldsymbol{f}}_{m+d} - \widehat{\boldsymbol{f}}_m\right\| \approx \frac{1}{\|S\boldsymbol{v}_{m+d}\|} \left\| SV_{m+d} \left(\widehat{\boldsymbol{q}}_{m+d} - \begin{bmatrix} \widehat{\boldsymbol{q}}_m \\ \boldsymbol{0}_d \end{bmatrix} \right) \right\| . \tag{2.21}$$

This estimate is evaluated at every iteration, and if we find it to be smaller than some user specified tolerance `tol`, we terminate the algorithm. Numerical experiments suggest this stopping criterion to be robust and very accurate, for the embedding strategy used in this work, as well as in [4]. See Section 3.1.

3 Numerical Experiments

This section provides some numerical experiments on the sFOM method. The sFOM algorithm is implemented in the Julia programming language [2]. We provide a basic implementation of the main algorithm that can be of general use, as well as some examples where sFOM is employed in the context of simple exponential integrators. Unless explicitly stated, we use a truncated Arnoldi iteration with truncation length 4, and subspace embedding dimension $s = 2 \cdot$ `max_iter`. The source code is freely available at `https://github.com/lithell/RandomKrylovfAb`.

3.1 Basic Verification

We begin with some basic verification, to ensure that the method is producing good approximations to the problem. Initially, we verify that the method is converging as expected, and we then move on to investigating the behaviour of the stopping criterion developed in Section 2.4, as well as the effect of varying the embedding quality ε.

3.1.1 Convergence

Wathen Matrix

For our first convergence-verification, we will consider the problem of approximating $\exp(-A)\boldsymbol{b}$, where A is the "Wathen"-matrix available through the Julia package `MatrixDepot.jl` [13]. To use this matrix, simply call `A = matrixdepot("wathen", nn, nn)`. A is then a sparse, symmetric and positive definite matrix, arising as the consistent mass matrix in the finite element method [12]. Here, nn is the grid size in this method. We take $nn = 25$, yielding a 1976×1976 matrix. We run sFOM ten times on this problem, recording the error decay. Each time we run the algorithm with a random $\boldsymbol{b}$, scaled to be on the same order as the elements in A. The results of can be seen in Figure 3.1, where we also show the sparsity structure of A.

For this problem, sFOM converges as expected, and the error bottoms out in around 100 iterations, much less than the size of the problem. We also note that depending on $\boldsymbol{b}$, the error stagnates at different levels. The final error obtained can differ by as much as an order of magnitude for this problem. Noting the error curves in Figure 3.1, we also see that the method exhibits somewhat sporadic convergence, consistent with the findings in [4]. However, for this problem the convergence is robust as a whole, and the method yields good approximations.

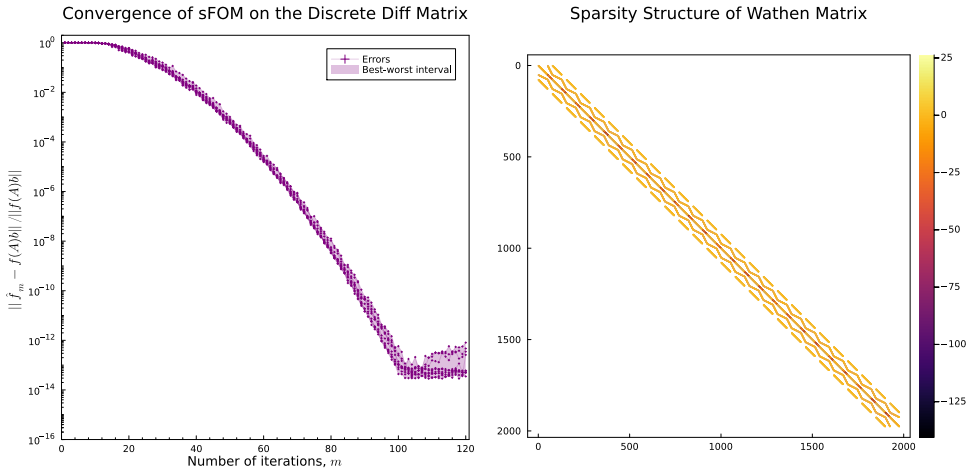

Figure 3.1: *Convergence of sFOM for the Wathen-matrix. We approximate* $\exp(-A)\boldsymbol{b}$*. The left figure shows the errors decaying for 10 runs of sFOM on the Wathen matrix, with random choices of* $\boldsymbol{b}$*. Here we use a fixed number of iterations,* $m = 120$*. The right figure shows the sparsity structure of the matrix.*

Discrete Diffusion Operator

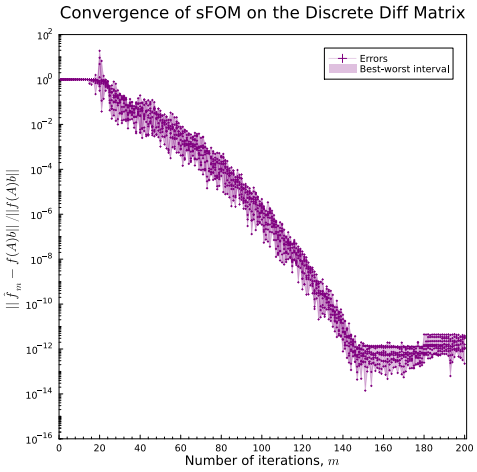

Figure 3.2: *Convergence of sFOM on the 2D discrete diffusion operator. We approximate* $\exp(A)\boldsymbol{b}$*. The figure shows the relative error decaying as a function of iteration, for 10 runs of the method. We use a fixed number of iterations,* $m = 200$*.*

The 2D discrete diffusion operator, arising in finite difference methods, is defined as

$$A = T \otimes I + I \otimes T\,,$$

where $T = \frac{1}{h^2}\text{tridiag}(1, -2, 1)$ is the $N \times N$ standard discrete 1D Laplacian. Here h is the discretization size over the unit square, assuming a uniform discretization in x and y. We denote

by $\otimes$ the Kronecker product of two matrices. Then T is $N^2 \times N^2$ symmetric. We consider the problem of approximating $\exp(A)\boldsymbol{b}$, with a random $\boldsymbol{b}$ scaled to have norm on the same order as the elements in A. We take $N = 35$, yielding $A \in \mathbb{R}^{1225\times 1225}$. The results can be seen in Figure 3.2. We see that the method has a much more sporadic convergence for this problem compared with the Wathen-matrix. This convergence pattern is also representative of the general convergence characteristic of the method. However, as a whole, the method still converges consistently.

To conclude this section, our experiments seem to indicate that the method converges consistently, and generates good approximations. However, when it has converged we see that the behaviour of the error can be troublesome. Note for instance that the general trend in Figure 3.1, and Figure 3.2 seem to be that the error increases after the method has converged. We have also found that for certain problems this increase can quickly get out of hand, and generate completely erroneous approximations if the results are trusted blindly. This is of course assuming that we do not have knowledge of the behaviour of the error, as might be the case in practical applications. It is therefore imperative to employ the stopping criterion developed in Section 2.4 if we ever hope to use the method effectively in application.

3.1.2 Stopping Criterion

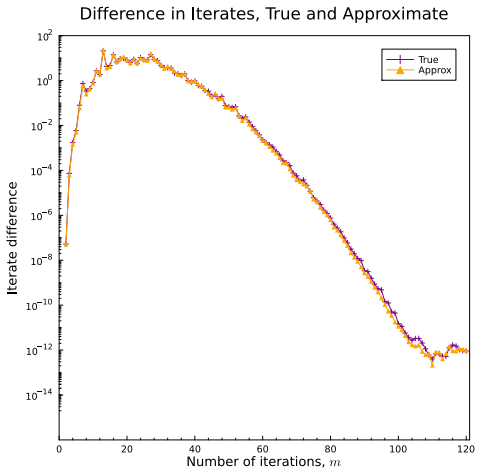

Figure 3.3: *True and approximate difference between iterates.*

In this section we verify that the stopping criterion developed in 2.4 is satisfactory for our application. We compare the iteration difference as predicted by (2.21) with the actual difference between iterates. For this verification we will reuse the problem with the Wathen-matrix presented above. The results can be seen in Figure 3.3.

We see that the estimate (2.21) follows the true difference between iterates very closely. For our formulation of the method, this estimate is good enough to reliably use it as a stopping criterion. However, it is possible that it would becomes less reliable for embedding matrices of lesser quality.

It is also noteworthy that the iteration difference is very small for small values of m. This "startup" period can be rather long for larger problems, cf. [4], and it might be advised to use some additional condition on the number of iterations performed, before we actually start comparing our estimate to some tolerance. For instance, in this problem the iteration difference

becomes as small as 10^{-7}, and for generous tolerances, a naive approach may terminate the iterations prematurely. It is not unreasonable to assume that some problems will produce a very long startup periods, and it is probably wise to use such a condition on the iteration number if this method is to be deployed in practice. However, for the problems considered in this work, we have not found it necessary to employ such a strategy. Choosing an appropriate value of this iteration threshold might be hard for some problems. Numerical experiments suggest that sFOM can converge very quickly for small to medium scale problems, and choosing a threshold that significantly over-estimates the startup period will be crippling in terms of performance. It would therefore be nice to develop a strategy for estimating this startup period, see Section 4.

3.1.3 Effect of Sketching Parameter

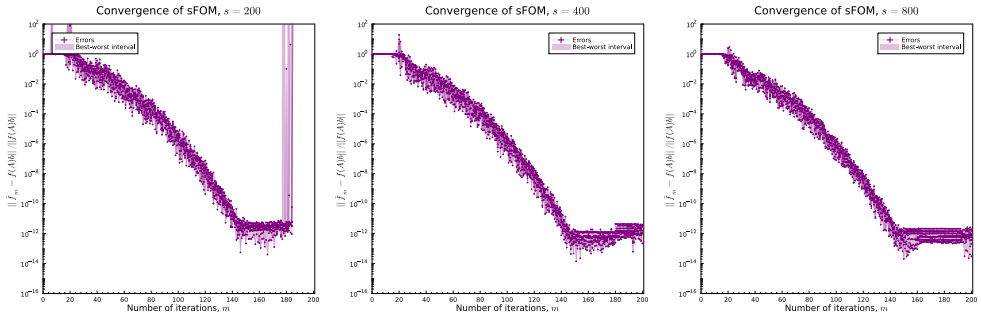

Figure 3.4: *Effect of sketching parameter on the convergence of sFOM on the discrete diffusion operator. The problem size is $N = 1225$, and we run the method three times respectively for sketching parameters $s = 200, 400, 800$. Numerical instabilities can clearly be seen in the convergence of sFOM for $s = 200$.*

Continuing, we investigate experimentally the effect of the sketching parameter s. That is, we vary the subspace embedding dimension. Intuitively, a very small subspace embedding dimension yields a faster computation, with the trade-off being slower convergence per iteration (if the method converges at all). Conversely, if s is taken to be closer to the dimension of the full problem, N, the computations would be more expensive, but yield more accurate results. Of course, in the extreme case, $s = N$, there is no dimensional reduction at all, and we might as well compute the full FOM approximation. We reuse the discrete diffusion operator problem, with the same dimensions, i.e. $N = 1225$. We run the method ten times, for $s = 200, 400, 800$ respectively. The maximum Krylov dimension we encounter is still taken to be 200. The choices $s = 200, 400, 800$ then correspond to one, two and four times the maximum iteration. The results from these tests are related in Figure 3.4. Again, the sporadic convergence of sFOM is evident from these figures. The effect is especially pronounced for $s = 200$, and for sketching parameters significantly smaller than this do not generate usable approximations. For these very small sketching parameters, the method does not seem to converge at all, and numerical instabilities becomes completely dominant in the compute. This also highlight the importance of an adaptive stopping condition.

That said, when the method does converge, the speed of convergence with respect to the number of iterations do not seem to be significantly affected by the sketching parameter. The convergence behaviour is almost identical for all three choices of sketching parameter. Hence, there exists no incentive to take the sketching dimension to be excessively large. However, since

the method exhibits strong indications of numerical instabilities, it would not be wise to take s to be too small either. There is a need to balance the expense of computation with potential instabilities of the compute. To do this effectively, there is a need for a better understanding of convergence properties of the method, cf. Section 4.

3.2 Example Application

3.2.1 Exponential Integrators

Consider the ordinary differential equation

$$\frac{\partial}{\partial t}u(t) = g(y(t)), \quad y(t_0) = y_0\,, \tag{3.1}$$

where $g : \mathbb{C}^N \mapsto \mathbb{C}^N$ is some general nonlinear function, and $y(t) : \mathbb{R} \mapsto \mathbb{C}^N$. In the special case where g is given by $g(y(t)) = Ay(t) + b$, (3.1) can be solved exactly by means of a so-called *φ-function*. We provide the following definition.

Definition 3.2.1. The φ-functions are defined recursively via the relation

$$\varphi_0(z) = \exp(z), \quad \varphi_\ell(z) = \frac{1}{(\ell-1)!}\int_0^1 \exp((1-s)z)s^{\ell-1}ds, \quad \ell \geq 1\,. \tag{3.2}$$

For our purposes it will be sufficient to consider $\varphi_1(z)$, given explicitly as

$$\varphi(z) := \varphi_1(z) = \frac{\exp(z)-1}{z}\,. \tag{3.3}$$

Returning to our linear ODE, the solution of (3.1) with right hand side $Ay(t) + b$ is given explicitly by $y(t) = y_0 + t\varphi(tA)(Ay_0 + b)$. Here we implicitly assume some extension of φ to matrices. This can be achieved through a number of standard definitions of matrix functions, such as the Cauchy integral form. This simple form of the solution of (3.1), while convenient, does not hold for a general non-linear g. However, it will serve as the basis for our simple explicit Euler exponential integrator.

Exponential integrators have proved to be highly effective for certain classes of ODEs. For a thorough review of exponential integrators, and φ-functions, the reader is referred to the review paper by Hochbruck and Ostermann [6].

We proceed with the definition of our exponential integrator.

Definition 3.2.2. Let $0 = t_0 < t_1 < \cdots < t_N$. The explicit Euler exponential integrator for (3.1) generates the approximations $y_k \approx y(t_k)$, defined as

$$y_{k+1} = y_k + h_k\varphi(h_kJ_k)g(y_k) \tag{3.4}$$

where $h_k = t_{k+1} - t_k$, and $J_k := g'(y_k)$ is the Jacobian of g.

This is the approximation scheme we will use as the basis for our test-case. Notice that (3.4) necessitates evaluation of $\varphi(h_kJ_k)$. For moderate to large J_k, this evaluation will be the computationally dominant part of the iteration, and hence any improvement in the speed of this compute is very welcome. It is for this purpose we will use sFOM.

In our test-cases, the ODE (3.1) will correspond to a semi-discretization of a parabolic PDE.

3.2.2 A Quasi-Linear One-Dimensional Parabolic PDE

As an indication of possible applications of the sFOM algorithm, we use the exponential integrator from above on a simple one-dimensional problem. This application is meant to be indicative of how sFOM could be used in practice, and we do not aim to get highly accurate results, which is why we employ our simple explicit Euler scheme, and not some more intricate scheme. We will study a problem suggested by Ostermann and Hochbruck in [7]. Consider the quasi-linear parabolic problem

$$\frac{\partial U(x,t)}{\partial t} = \frac{\partial^2 U(x,t)}{\partial x^2} + \frac{1}{1+U(x,t)^2} \tag{3.5}$$

with $x \in [0,1]$, $t \in [0, 0.02]$. We also impose homogeneous Dirichlet boundary conditions. We discretize this problem by the standard method of lines process, yielding the semi-discrete problem

$$\frac{\partial W(t)}{\partial t} = TW(t) + F(W(t))\,, \quad \text{with} \quad F(W(t)) = \begin{bmatrix} \frac{1}{1+W_1(t)^2} \\ \vdots \\ \frac{1}{1+W_{N-1}(t)^2} \end{bmatrix}. \tag{3.6}$$

Here, T is the standard finite difference Laplacian. Then, comparing with our method (3.4), we also write the Jacobian as:

$$J(W(t)) = T - 2W(t)F(W(t))^2\,. \tag{3.7}$$

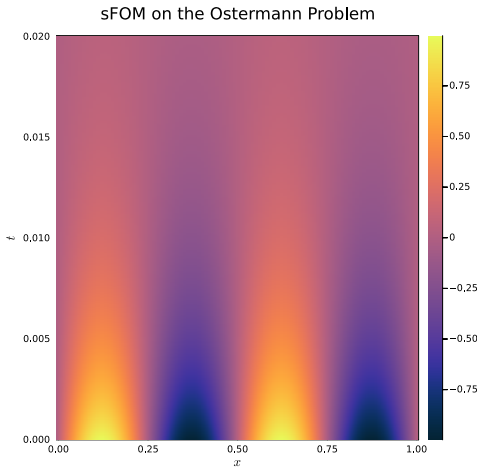

Figure 3.5: *sFOM on a quasi-linear parabolic problem with homogeneous Dirichlet boundary conditions. We use a spatial discretization $N = 100$, and a temporal discretization $M = 350$. We approximate the solution using the explicit Euler exponential integrator, where sFOM is used for the evaluation of the matrix function φ. We also impose a sinusoidal initial condition.*

We approximate the solution to this semi-discrete problem by our exponential integrator, where sFOM is employed for the computation of the φ-function. We compute the solution with a spatial discretization $N = 100$, and a temporal discretization $M = 350$, with a sinusoidal initial condition. The results can be seen in Figure 3.5.

Problems such as this one are a promising area of applications for methods similar to sFOM. In this particular case, the problem is not huge, and so the exponential integrator approach might not be as competitive as other classes of methods. However, if a very fine discretization is required, the system matrix rapidly becomes very large. In this case, approaches making use of exponential-integrator-type schemes become much more viable. In particular, these cases are an ideal situation for the developing randomized methods in NLA.

4 Conclusion and Outlook

The results from Section 3 seem to indicate that sFOM is both robust and highly effective, especially in combination with the stopping criterion developed in Section 2.4. Furthermore, the results from the example with the exponential integrator indicate that this is a class of problems that could benefit greatly use of randomized techniques, especially for large, sparse, and nonlinear problems, where computing the action of matrix functions are dominating in terms of computing power.

There do however remain some questions concerning the convergence of sFOM. First, and foremost, it is essential to be able to quantitatively characterize the, at times, very sporadic convergence of sFOM. Recently, Palitta, Schweitzer, and Simoncini [9] have developed new tools to analyze sketched Krylov methods, via a generalization of the Arnoldi relation (2.4), so-called *Arnoldi-like* decompositions. Perhaps with these tools, the theoretical results from [9] could be translated into practical tools for managing the convergence properties of implementations of randomized Krylov methods.

Additionally, it would be beneficial to develop a more generally robust estimation of the embedding quality, for use in the stopping criterion. As mentioned previously, (2.20) is sufficiently accurate for high-quality embeddings, say $\varepsilon < 1/2$. After this point, the estimate may start to become unreliable, and for very low-quality embeddings, it becomes practically useless. Perhaps with some more insight into the properties of oblivious subspace embeddings, a more quantifiable estimate could be developed. Of course, a lower embedding dimension would allow for even faster computation, and so there is legitimate interest in developing methods for dealing with these cases.

Finally, since the method of this work is rather general, with the pivotal step being the sketching of the orthogonality condition in the FOM method, it would be interesting to investigate whether a unified framework for randomization of classical Krylov methods could be developed.

In conclusion, the randomization of Krylov methods is a highly active research area with much potential, where it is essential to advance the theory and methods. Methods in this class are very promising experimentally, and with the emerging theory for these methods, the class of randomized Krylov solvers promise to become a powerful feature in the toolkit of any researcher in numerical linear algebra.

References

[1] Oleg Balabanov and Laura Grigori. "Randomized Gram–Schmidt Process with Application to GMRES". *SIAM Journal on Scientific Computing* 44.3 (June 2022), A1450–A1474. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/20M138870X.

[2] Jeff Bezanson et al. "Julia: A Fresh Approach to Numerical Computing". *SIAM Review* 59.1 (Jan. 2017), pp. 65–98. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/141000671.

[3] Alice Cortinovis, Daniel Kressner, and Yuji Nakatsukasa. *Speeding up Krylov subspace methods for computing f(A)b via randomization*. June 5, 2023. arXiv: 2212.12758[cs, math].

[4] Stefan Güttel and Marcel Schweitzer. *Randomized sketching for Krylov approximations of large-scale matrix functions*. Mar. 17, 2023. arXiv: 2208.11447[cs,math].

[5] N. Halko, P. G. Martinsson, and J. A. Tropp. "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions". *SIAM Review* 53.2 (Jan. 2011), pp. 217–288. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/090771806.

[6] Marlis Hochbruck and Alexander Ostermann. "Exponential integrators". *Acta Numerica* 19 (May 2010), pp. 209–286. ISSN: 0962-4929, 1474-0508. DOI: 10.1017/S0962492910000048.

[7] Marlis Hochbruck and Alexander Ostermann. "Exponential Runge–Kutta methods for parabolic problems". *Applied Numerical Mathematics* 53.2 (May 2005), pp. 323–339. ISSN: 01689274. DOI: 10.1016/j.apnum.2004.08.005.

[8] Yuji Nakatsukasa and Joel A. Tropp. "Fast & Accurate Randomized Algorithms for Linear Systems and Eigenvalue Problems" (2021). Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2111.00113.

[9] Davide Palitta, Marcel Schweitzer, and Valeria Simoncini. *Sketched and truncated polynomial Krylov methods: Evaluation of matrix functions*. June 10, 2023. arXiv: 2306.06481[cs,math].

[10] Geoff Pleiss et al. "Fast Matrix Square Roots with Applications to Gaussian Processes and Bayesian Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 22268–22281.

[11] Vladimir Rokhlin and Mark Tygert. "A fast randomized algorithm for overdetermined linear least-squares regression". *Proceedings of the National Academy of Sciences of the United States of America* 105.36 (Sept. 9, 2008), pp. 13212–13217. ISSN: 1091-6490. DOI: 10.1073/pnas.0804869105.

[12] A. J. Wathen. "Realistic Eigenvalue Bounds for the Galerkin Mass Matrix". *IMA Journal of Numerical Analysis* 7.4 (Oct. 1, 1987), pp. 449–457. ISSN: 0272-4979, 1464-3642. DOI: 10.1093/imanum/7.4.449.

[13] Weijian Zhang and Nicholas J. Higham. "Matrix Depot: an extensible test matrix collection for Julia". *PeerJ Computer Science* 2 (Apr. 6, 2016), e58. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.58.