A Block Krylov Method to Compute the Action of the Frechet Derivative of a Matrix Function on a Vector with Applications to Condition Number Estimation

Kandolf, Peter and Relton, Samuel D.

2016

MIMS EPrint: **2016.33**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

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*

PETER KANDOLF† AND SAMUEL D. RELTON‡

Abstract. We design a block Krylov method to compute the action of the Fréchet derivative of a matrix function on a vector using only matrix–vector products. The algorithm we derive is especially effective when the direction matrix in the derivative is of low rank. Our results and experiments are focused mainly on Fréchet derivatives with rank-1 direction matrices. Our analysis applies to all functions with a power series expansion convergent on a subdomain of the complex plane which, in particular, includes the matrix exponential. We perform an a priori error analysis of our algorithm to obtain rigorous stopping criteria. Furthermore, we show how our algorithm can be used to estimate the 2-norm condition number of $f(A)b$ efficiently. Our numerical experiments show that our new algorithm for computing the action of a Fréchet derivative typically requires a small number of iterations to converge and (particularly for single and half precision accuracy) is significantly faster than alternative algorithms. When applied to condition number estimation our experiments show that the resulting algorithm can detect ill-conditioned problems that are undetected by competing algorithms.

Key words. matrix function, matrix exponential, Fréchet derivative, condition number, Krylov subspace, block Krylov subspace

AMS subject classifications. 65F30, 65F60

1. Introduction.

Given a matrix function $f\colon \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ one commonly needs to compute $f(A)b$ for a large, sparse matrix A and a vector $b \in \mathbb{C}^n$. For example, when $f(A) = \mathrm{e}^A$ is the matrix exponential, this computation arises when solving evolution equations with exponential integrators (e.g. [11], [12], [13]) or in network analysis (e.g. [6], [7]). The latter paper also uses the Fréchet derivative of the matrix exponential, which we introduce shortly.

To approximate $f(A)b$ one might use a Krylov subspace method. Since $f(A) = p(A)$ for some scalar polynomial $p(z)$ we construct the space

$$\mathcal{K}_m(A,b) = \operatorname{span}\left\{b, Ab, A^2b, \ldots, A^{m-1}b\right\}, \tag{1.1}$$

and find a vector in this subspace which is close to $f(A)b$. Hopefully as the parameter m increases, the difference between the true value of $f(A)b$ and our approximation will decrease. More sophisticated methods that replace the polynomial subspace (1.1) by a rational Krylov subspace, utilizing the inverse of A, have also been developed. For the matrix exponential, there are a number of alternatives to using a Krylov subspace such as the Leja method [3] and methods based upon Taylor series approximation [1], which have been compared in a recent review paper [2].

In any numerical computation it is important to know the condition number of the problem: when the algorithm used is backward stable we know that the relative error

*Version of May 31, 2016.

†Institut für Mathematik, Universität Innsbruck, Austria (peter.kandolf@uibk.ac.at, https://numerical-analysis.uibk.ac.at/p.kandolf), Recipient of a DOC Fellowship of the Austrian Academy of Science at the Department of Mathematics, University of Innsbruck, Austria

‡School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK (samuel.relton@manchester.ac.uk, http://www.maths.manchester.ac.uk/~srelton), This work was supported by European Research Council Advanced Grant MATFUN (267526).

is approximately bounded above by the condition number times the unit roundoff. Therefore, being able to estimate the condition number efficiently is critical when determining how accurate a solution can be expected.

Before discussing the condition number of the $f(A)b$ problem, we must first define the Fréchet derivative of the matrix function f. The Fréchet derivative is an operator $L_f(A,\cdot)\colon \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$, which is linear in its second argument and for any matrix $E \in \mathbb{C}^{n\times n}$ satisfies

$$f(A+E) - f(A) = L_f(A,E) + o(\|E\|).$$

If we first consider the matrix function $f(A)$, without applying it to a vector, its relative condition number is closely related to the Fréchet derivative of the function. We have [9, Chap. 3]

$$\mathrm{cond}_{\mathrm{rel}}(f,A) := \lim_{\epsilon\to 0} \sup_{\|E\|\le\epsilon\|A\|} \frac{\|f(A+E)-f(A)\|}{\epsilon\|f(A)\|} = \max_{\|E\|=1} \frac{\|L_f(A,E)\|\|A\|}{\|f(A)\|}.$$

For the action of the matrix function, Deadman [5, p. 4] defines the relative condition number of $f(tA)b$. If one ignores the dependence on t by setting $t=1$, he defines

$$\mathrm{cond}(f,A,b) := \lim_{\epsilon\to 0} \sup_{\substack{\|E\|\le\epsilon\|A\| \\ \|\Delta b\|\le\epsilon\|b\|}} \frac{\|f(A+E)(b+\Delta b)-f(A)b\|}{\epsilon\|f(A)b\|}, \tag{1.2}$$

for any induced matrix norm and goes on to show that

$$\mathrm{cond}(f,A,b) := \lim_{\epsilon\to 0} \sup_{\substack{\|E\|\le\epsilon\|A\| \\ \|\Delta b\|\le\epsilon\|b\|}} \frac{\|L_f(A,E)b + f(A)\Delta b + o(\|E\|) + o(\|\Delta b\|)\|}{\epsilon\|f(A)b\|}, \tag{1.3}$$

where the $o(\|E\|)$ and $o(\|\Delta b\|)$ terms disappear as $\epsilon \to 0$. In order to estimate this condition number efficiently we need to compute $L_f(A,E)b$, for multiple matrices E, in a highly efficient manner. The details of how to estimate this condition number are discussed further in section 5.

The primary goal of this paper is to give an efficient means of evaluating $L_f(A,E)b$ for E of rank-1 based upon block Krylov subspace methods. This can easily be extended to E of rank-k since $L_f(A,E)$ is linear in E. We also provide a priori error analysis for our algorithm to derive rigorous stopping criteria. As a secondary goal we show how our algorithm can be applied to improve the computation of the condition number $\mathrm{cond}(f,A,b)$.

This work is organized as follows. In section 2 we give some background information on the block Krylov method that we use throughout the rest of our analysis. Following this in sections 3 and 4 we derive an algorithm to compute $L_f(A,E)b$ for a rank-1 matrix E and perform our a priori error analysis. In section 5 we show how our algorithm can be used to estimate $\mathrm{cond}(f,A,b)$ before performing a battery of numerical experiments in section 6. Finally in section 7 we give some concluding remarks.

2. Block Krylov method. The algorithm for computing $L_f(A,E)b$ we introduce in section 3 is based upon the block Krylov decomposition; this section introduces the notation required throughout the rest of this work.

We define a block Krylov subspace, with block size p, as the set

$$\mathcal{K}(A,[x_1,\dots,x_p]) := \mathrm{span}\{x_1,\dots,x_p,Ax_1,\dots,Ax_p,A^2x_1,\dots,A^2x_p,\dots\}. \tag{2.1}$$

In practice we only use a basis for the span of the first $M = mp$ vectors, which we call $\mathcal{K}_m(A, [x_1, \ldots, x_p])$. There are numerous variations of the block Arnoldi method available to compute a basis of this space, many of which can be found in [16, Sec. 6.12]. In this work we have chosen to use the Block Arnoldi–Ruhe algorithm introduced in [14] to compute a subspace with $p \geq 1$ starting vectors.

ALGORITHM 2.1 (Block Arnoldi–Ruhe algorithm [14], [16]). *Given a block size $p \geq 1$, the vectors $x_1, \ldots, x_p$, and the size of the desired subspace $M = mp$, the following algorithm outputs an orthonormal basis, $\{v_j\}$ for $j = 1\colon M$, that spans the block Krylov subspace $\mathcal{K}_m(A, [x_1, \ldots, x_p])$. When M is not a multiple of p the algorithm returns an orthonormal basis which spans the first M vectors from the block Krylov subspace.*

Compute p initial orthonormal vectors $\{v_i\}_{i=1,\ldots,p}$ that have the same span as $\{x_i\}_{i=1,\ldots,p}$ by extracting the columns of Q from $QR = [x_1, \ldots, x_p]$.
for $j = p\colon M + p - 1$
    $k = j - p + 1$
    $w = Av_k$
    for $i = 1\colon j$
        $h_{i,k} = w^* v_i$
        $w = w - h_{i,k} v_i$
    end for
    $h_{j+1,k} = \|w\|_2$
    $v_{j+1} = w / h_{j+1,k}$
end for

Note that, for $p = 1$, the algorithm reverts to the standard Arnoldi process. Furthermore, one potential advantage of this algorithm over other variants is that the dimension of the subspace approximation M is not necessarily forced to be a multiple of p. However, if the subspace dimension M is a multiple of p, then the algorithm can be rewritten to make use of level-3 BLAS routines to take advantage of the induced block parallelism. To increase the stability of the algorithm we can also add a reorthogonalization step at the end of each iteration.

For $M = mp$ we have analogues of the standard Arnoldi decomposition which are useful for our later analysis. Before giving these relationships we need to define our notation. First, let us define U_i to be the $n \times p$ matrix with columns $v_{p(i-1)+1}, \ldots, v_{pi}$ from Algorithm 2.1 and $V_m = [U_1, \ldots, U_m]$. Second, let $\bar{H}_m$ be the band Hessenberg matrix with nonzero entries h_{ij} from Algorithm 2.1 and take H_m to be the upper-left square matrix of size $mp \times mp$ in $\bar{H}_m$. Finally, let $H_{m,m+1}$ denote the $p \times p$ block in the lower-right of $\bar{H}_m$ and J_m be the final p columns of I_M. In this case we have the following analogues of the Arnoldi decomposition [16, p. 161]:

$$AV_m = V_m H_m + U_{m+1} H_{m+1,m} J_m^T \tag{2.2}$$

$$= V_{m+1} \bar{H}_m, \tag{2.3}$$

$$V_m^T A V_m = H_m. \tag{2.4}$$

3. Algorithm derivation. In this section we derive the basic algorithm upon which the rest of our work builds. For the analysis in the next two sections we take the direction matrix $E = \eta y z^*$ in the Fréchet derivative to be of rank-1 with $\eta \in \mathbb{R}$ and $\|y\|_2 = \|z\|_2 = 1$. Furthermore, without loss of generality, we will assume that $\|b\|_2 = 1$. We can easily generalize the development to rank-k matrices since the Fréchet derivative is linear in its second argument.

We start by proving the following result.

THEOREM 3.1. *Given A, $E \in \mathbb{C}^{n\times n}$ such that f is defined and Fréchet differentiable at A and such that $E = \eta yz^*$, then $L_f(A,E)b \in \mathcal{K}_m(A,y)$ for some $m \geq 0$, where $\mathcal{K}_m(A,y)$ is the Krylov subspace with starting vector y of dimension m.*

Proof. Assuming that the scalar function f is sufficiently differentiable [9, eq. (3.16)], we can form a 2×2 block matrix to get

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

Therefore, we can multiply both sides by a vector on the right-hand side to obtain

$$\begin{aligned} f\left(\begin{bmatrix} A & E \\ 0 & A \end{bmatrix}\right)\begin{bmatrix} 0 \\ b \end{bmatrix} &= \begin{bmatrix} f(A) & L_f(A,E) \\ 0 & f(A) \end{bmatrix}\begin{bmatrix} 0 \\ b \end{bmatrix} \\ &= \begin{bmatrix} L_f(A,E)b \\ f(A)b \end{bmatrix}, \end{aligned} \tag{3.1}$$

so that the quantity we desire is in the upper half of the resulting vector.

Now for $X \in \mathbb{C}^{l\times l}$ we know that $f(X) = p(X)$ for some polynomial p of degree at most l [9, Chap. 1] and therefore $L_f(A,E)b$ is in the span of the top half of the vectors formed by the matrix–vector products

$$\begin{bmatrix} A & E \\ 0 & A \end{bmatrix}^k \begin{bmatrix} 0 \\ b \end{bmatrix},$$

for $k = 1 : l$, where $l \leq 2n$. To complete the proof it remains to show that the top half of these vectors are in $\mathcal{K}_m(A,y)$, which we achieve via induction. The base case for $k = 0$ is obvious. For $k = 1$ and, recalling that $E = \eta yz^*$, we have

$$\begin{bmatrix} A & \eta yz^* \\ 0 & A \end{bmatrix}\begin{bmatrix} 0 \\ b \end{bmatrix} = \begin{bmatrix} \eta y(z^*b) \\ Ab \end{bmatrix},$$

where the top half of the result is in $\mathcal{K}_1(A,y)$ and the bottom half is in $\mathcal{K}_1(A,b)$. For induction, suppose we have a vector

$$c_k = \begin{bmatrix} y_k \\ b_k \end{bmatrix},$$

where $y_k \in \mathcal{K}_k(A,y)$ and $b_k \in \mathcal{K}_k(A,b)$. Applying our 2×2 block matrix we obtain

$$\begin{aligned} \begin{bmatrix} A & \eta yz^* \\ 0 & A \end{bmatrix} c_k &= \begin{bmatrix} A & \eta yz^* \\ 0 & A \end{bmatrix}\begin{bmatrix} y_k \\ b_k \end{bmatrix} \\ &= \begin{bmatrix} Ay_k + \eta y(z^*b_k) \\ Ab_k \end{bmatrix} \\ &=: \begin{bmatrix} y_{k+1} \\ b_{k+1} \end{bmatrix}, \end{aligned}$$

where $y_{k+1} \in \mathcal{K}_{k+1}(A,y)$ and $b_{k+1} \in \mathcal{K}_{k+1}(A,b)$, completing the proof. ∎

With this result we know that $L_f(A,yz^*)b \in \mathcal{K}_m(A,y)$ and our hope is that in practice a Krylov subspace with $m < n$ is required. Unfortunately, we have been unable to see how to approximate $L_f(A,yz^*)b$ solely from the space $\mathcal{K}_m(A,y)$. However, we will show that by using the block Krylov subspace $\mathcal{K}_m(A,[y,b])$, we can compute

an approximation to $L_f(A, yz^*)b$. Since $\mathcal{K}(A, b)$ is already required to compute $f(A)b$ with a Krylov method, this is not an unreasonable restriction.

When the scalar function f is analytic on and inside a contour Γ enclosing $\Lambda(A)$, the spectrum of A, we can represent $f(A)$ using the Cauchy integral [9, Def. 1.11]

$$f(A) := \frac{1}{2\pi i}\int_\Gamma f(\omega)(\omega I - A)^{-1}\mathrm{d}\omega. \tag{3.2}$$

Taking the Fréchet derivative of this representation using the chain rule [9, Prob. 3.8] we obtain

$$L_f(A, \eta yz^*) = \frac{\eta}{2\pi i}\int_\Gamma f(\omega)(\omega I - A)^{-1}yz^*(\omega I - A)^{-1}\mathrm{d}\omega. \tag{3.3}$$

For the next step, assume that we have a block Krylov subspace $\mathcal{K}_m(A, [y, b])$ so that we can form the approximation $(zI - A)^{-1}y \approx V_m(zI - H_m)^{-1}e_1$, and similarly $(zI - A)^{-1}b \approx V_m(zI - H_m)^{-1}\tilde{e}$, where e_1 is the first unit vector and $\tilde{e} = V_m^*b$. By employing these two approximations we see that

$$\begin{aligned} L_f(A, \eta yz^*)b &\approx \frac{\eta}{2\pi i}\int_\Gamma f(\omega)V_m(\omega I - H_m)^{-1}e_1z^*V_m(\omega I - H_m)^{-1}\tilde{e}\ \mathrm{d}\omega \\ &= \eta V_m L_f(H_m, e_1(z^*V_m))\tilde{e} \qquad (3.4)\\ &= \eta V_m L_f(H_m, e_1\tilde{z}^*)\tilde{e}, \qquad (3.5)\\ &=: L^{(m)} \end{aligned}$$

where $\tilde{z} = V_m^*z$ and $L^{(m)}$ is our approximation to $L_f(A, \eta yz^*)$ from the Krylov subspace $\mathcal{K}_m(A, [y, b])$.

Overall, we have shown that if the block Krylov subspace $\mathcal{K}_m(A, [y, b])$ can be used to approximate both $(zI - A)^{-1}y$ and $(zI - A)^{-1}b$, then we can approximate $L^{(m)} \approx L_f(A, \eta yz^*)b$ by computing the Fréchet derivative at the (potentially much smaller matrix) H_m.

To make this into an iterative algorithm we must now introduce a stopping criteria. We note that, in exact arithmetic, the algorithm will converge in at most $n/2$ iterations since we add two vectors in each iteration. If we want to compute the answer to some relative tolerance, tol, we can continue adding vectors into our Krylov subspace by increasing m until either $\|L^{(m)} - L^{(m-1)}\|/\|L^{(m)}\| \leq \text{tol}$ or $m = \text{floor}(n/2)$.

This results in the following basic algorithm.

ALGORITHM 3.2. *Let A be an $n \times n$ matrix and f a matrix function that is defined and Fréchet differentiable at A, whilst being analytic on and inside a contour enclosing $\Lambda(A)$. Furthermore, suppose y, z, and b are three vectors satisfying $\|y\|_2 = \|z\|_2 = \|b\|_2 = 1$. Finally, let $\eta \in \mathbb{R}$ and a tolerance $\text{tol} > 0$ be given. Then the following algorithm approximates $L_f(A, \eta yz^*)b$.*

```
for m = 1:floor(n/2)
    Compute K_m(A, [y, b]) with the corresponding H_m and V_m.
    Compute L^(m) = η V_m L_f(H_m, e_1(z* V_m))ẽ, where ẽ = V_m* b.
    if m > 1 and ||L^(m) − L^(m−1)||/||L^(m)|| ≤ tol
        break out of loop
    end if
end for
return L^(m)
```

4. A priori error analysis. The goal of this section is to improve the reliability of the stopping criteria in Algorithm 3.2 by performing an a priori error analysis. Our methodology is based upon that by Saad [15] for the matrix exponential, though we need to make a number of modifications to account for the Fréchet derivative.

During our analysis we will repeatedly use the fact that if a power series $f(x) = \sum_{i=0}^{\infty} a_i x^i$ has radius of convergence r then for $A, E \in \mathbb{C}^{n\times n}$ with $\|A\| < r$ we have [9, Prob. 3.6]

$$L_f(A,E) = \sum_{i=1}^{\infty} a_i \sum_{j=1}^{i} A^{j-1} E A^{i-j}. \tag{4.1}$$

Our first step towards an a priori error estimate is to relate polynomials of the above form with corresponding polynomials in H_m, the block-Hessenberg matrix, which is the subject of the following lemma.

LEMMA 4.1. *Let A be any matrix and V_m, H_m the results of m steps of the block Arnoldi method applied to A with initial vectors $[y, b]$ and let $E = \eta y z^*$. Then for integers $0 \le i, j \le m-1$ we have*

$$A^i E A^j b = \eta V_m H_m^i e_1 z^* V_m H_m^j \tilde{e}, \quad \text{where} \quad \tilde{e} = V_m^* b, \quad \text{and} \tag{4.2}$$

$$\sum_{s=0}^{j} \alpha_s \sum_{k=0}^{s} A^k E A^{s-k} b = \eta V_m \left(\sum_{s=0}^{j} \alpha_s \sum_{k=0}^{s} H_m^k e_1 z^* V_m H_m^{s-k} \right) \tilde{e}. \tag{4.3}$$

Proof. It suffices to prove (a), from which (b) follows easily. Writing out E in full we have

$$A^i E A^j b = \eta (A^i y) z^* (A^j b).$$

Now, upon applying a result by Saad [15, Lem. 3.1], we have $A^i y = V_m H_m^i e_1$. Furthermore, since we know that $Ab = V_m H_m \tilde{e}$ where $\tilde{e} = V_m^* b$ we can apply a trivial modification of Saad's result to get $A^j b = V_m H_m^j \tilde{e}$. Making these substitutions we obtain

$$A^i E A^j b = \eta V_m H_m^i e_1 z^* V_m H_m^j \tilde{e},$$

which completes the proof. ∎

Next, let us consider the series expansion of $L_f(H_m, e_1 z^* V_m)$ from equation (4.1). By the Cayley–Hamilton theorem, H_m^k for $k \ge M$ can be expressed as a linear combination of lower order powers of H_m and therefore we must have

$$p_{M-1}(H_m, e_1 z^* V_m) = L_f(H_m, e_1 z^* V_m), \tag{4.4}$$

for some polynomial $p_{M-1}(x, e) = \sum_{i=1}^{M} c_i \sum_{j=1}^{i} x^{i-1} e x^{j-1}$ and constants c_i.

Using these polynomials we are ready to show that our approximation to $L_f(A, E)b$ is equivalent to forming $p_{M-1}(A, E)b$.

THEOREM 4.2. *The approximation $L_f(A, \eta y z^*)b \approx \eta V_m L_f(H_m, e_1 z^* V_m)\tilde{e}$ is mathematically equivalent to utilizing the approximation $p_{M-1}(A, E)b$ where $p_{M-1}(A, E)$ is the polynomial defined above.*

Proof. Starting from our approximation we have

$$\begin{aligned}\eta V_m L_f(H_m, e_1 z^* V_m)\tilde{e} &= \eta V_m p_{M-1}(H_m, e_1 z^* V_m)\tilde{e} \\ &= p_{M-1}(A, E)b,\end{aligned}$$

where the final equivalence is an application of Lemma 4.1. ∎

Using this result we can now write an expression for the error which we will then bound.

LEMMA 4.3. *Let A be any matrix and V_m, H_m the results of M steps of the block Arnoldi method applied to A with starting block $[y, b]$, let $E = \eta y z^*$, and let $p(A, E)$ be any polynomial of the form $p(A, E) = \sum_{i=1}^{M-1} c_i \sum_{j=1}^{i} A^{i-1} E A^{i-j}$ which approximates $L_f(A, E)$. Then*

$$L_f(A, E)b - \eta V_m L_f(H_m, e_1 z^* V_m)\tilde{e} = \eta \left[r_m(A, yz^*)b - V_m r_m(H_m, e_1 z^* V_m)\tilde{e} \right] \tag{4.5}$$

where we define $r_m(A, E) := L_f(A, E) - p(A, E)$ to be the remainder.

Proof. By rearranging the definition of the remainder polynomial we have

$$\begin{aligned}L_f(A, E)b &= p(A, E)b + r_m(A, E)b, \\ &= \eta V_m p(H_m, e_1 z^* V_m)\tilde{e} + r_m(A, E)b, \quad \text{by Lemma 4.1}, \\ &= \eta V_m \left[L_f(H_m, e_1 z^* V_m)\tilde{e} - r_m(H_m, e_1 z^* V_m)\tilde{e} \right] + r_m(A, E)b.\end{aligned}$$

Now as $r_m(A, E)b = \eta\, r_m(A, yz^*)b$ we have

$$L_f(A, E) - \eta V_m L_f(H_m, e_1 z^* V_m)\tilde{e} = \eta \left[r_m(A, yz^*)b - V_m r_m(H_m, e_1 z^* V_m)\tilde{e} \right],$$

as required. ∎

Finally, by taking the norm of both sides, we have

$$\begin{aligned}\|L_f(A, E)b - \eta V_m L_f(H_m, e_1 z^* V_m)\tilde{e}\|_2 &= \|\eta[r_m(A, yz^*)b - V_m r_m(H_m, e_1 z^* V_m)\tilde{e}]\|_2 \\ &\leq \eta \left(\|r_m(A, yz^*)b\|_2 + \|V_m r_m(H_m, e_1 z^* V_m)\tilde{e}\|_2 \right) \\ &\leq \eta \left(\|r_m(A, yz^*)\|_2 + \|r_m(H_m, e_1 z^* V_m)\|_2 \right), \qquad (4.6)\end{aligned}$$

where the final inequality is justified by the fact that $\|X\|_2 \leq \|Xq\|_2$ for any X and q where $\|q\|_2 = 1$, and the unitary invariance of the the 2-norm.

Using the previous result we can now obtain a priori error bounds on our approximation which depend upon the size of the Krylov subspace. Given some polynomial $p(A, E)$ we can find the remainder and then obtain an upper bound in equation (4.6). The remaining question is how to choose the polynomials: the polynomials p_{M-1} defined by equation (4.4) are not very useful for generating upper bounds since their coefficients depend upon both the function $f(z)$ and the matrix H_m, making them problem-dependent. Instead we consider the polynomials

$$s_{m-1}(A, E) = \sum_{i=1}^{m-1} a_i \sum_{j=1}^{i} A^{j-1} E A^{i-j}, \tag{4.7}$$

a truncation of the power series expansion of the Fréchet derivative, see (4.1). For such polynomials we have the remainder function

$$\begin{aligned}r_m(A, E) &= L_f(A, E) - s_{m-1}(A, E) \\ &= \sum_{i=m}^{\infty} a_i \sum_{j=1}^{i} A^{j-1} E A^{i-j}. \qquad (4.8)\end{aligned}$$

Our next lemma bounds this remainder from above.

LEMMA 4.4. *Let f have a Taylor series convergent in a disc of radius r, let $\|A\|_2 < r$, and let f be Fréchet differentiable at A. Furthermore, let $E \in \mathbb{C}^{n\times n}$ be given. Then the remainder function (4.8) is bounded above by*

$$\|r_m(A,E)\|_2 \le \|E\|_2\, t_m(\|A\|_2),$$

where $t_m(x) = \sum_{i=m}^{\infty} |a_i| i x^i$.

Proof.

$$\begin{aligned}
\|r_m(A,E)\|_2 &= \|\sum_{i=m}^{\infty}\sum_{j=1}^{i} A^{j-1}EA^{i-j}\|_2 \\
&\le \sum_{i=m}^{\infty}|a_i|\sum_{j=1}^{i}\|A\|_2^{j-1}\|E\|_2\|A\|_2^{i-j} \\
&\le \|E\|_2\sum_{i=m}^{\infty}|a_i| i \|A\|_2^{i-1} \\
&= \|E\|_2\, t_m(\|A\|_2). \qquad \square
\end{aligned}$$

This leads to the following corollary.

COROLLARY 4.5. *For functions f and matrices A satisfying the criteria of Lemma 4.4, with $E = \eta yz^*$ and b given such that $\|b\|_2 = \|y\|_2 = \|z\|_2 = 1$, the error in approximating $L_f(A,E)b$ by $\eta V_m L_f(H_m, e_1 z^* V_m)\tilde{e}$ is bounded above by*

$$\|L_f(A,E)b - \eta V_m L_f(H_m, e_1 z^* V_m)\tilde{e}\|_2 \le 2\eta t_m(\|A\|_2),$$

where t_m is defined in Lemma 4.4.

Proof. Combine the upper bound (4.6) with Lemma 4.4 noting that $\|H_m\|_2 \le \|A\|_2$ and that t_m is a monotonically increasing function. $\square$

To obtain more concrete bounds it is helpful to look at a specific function. For example, let us consider the matrix exponential. In this case we have the coefficients $a_i = 1/i!$ and remainder $\sum_{i=m}^{\infty} x^{i-1}/(i-1)!$ which is the same as that considered by Saad [15, Lem. 4.2, Thm. 4.3]. Using Saad's result we have the upper bound $t_m(x) \le (x^{m-1}\mathrm{e}^x)/(m-1)!$ and therefore we have proven the following corollary.

COROLLARY 4.6. *When considering the matrix exponential in Corollary 4.5 we obtain the upper bound*

$$\|L_f(A,E)b - \eta V_m L_f(H_m, e_1 z^* V_m)\tilde{e}\|_2 \le \frac{2\eta\|A\|_2^{m-1}\mathrm{e}^{\|A\|_2}}{(m-1)!}.$$

Proof. See above. $\square$

The analysis in this section has allowed us to use a relative a priori error bound to terminate our iterative algorithm as opposed to simply taking the relative difference between iterates in Algorithm 3.2. The resulting algorithm is as follows.

ALGORITHM 4.7. *Let A be an $n \times n$ matrix and f a matrix function that is defined and Fréchet differentiable at A, whilst being analytic on and inside a contour enclosing $\Lambda(A)$. Furthermore, suppose y, z, and b are three vectors satisfying $\|y\|_2 = \|z\|_2 = \|b\|_2 = 1$. Finally, let $\eta \in \mathbb{R}$ and* tol > 0 *be given. Then the following algorithm attempts to approximate $L_f(A, \eta yz^*)b$ within a relative tolerance* tol.

```
for m = 1: floor(n/2)
    Compute K_m(A, [y, b]) with the corresponding H_m and V_m.
    Compute L^(m) = η V_m L_f(H_m, e_1(z* V_m)) ẽ, where ẽ = [I_M 0] V_m* b.
    Set α equal to the a priori error estimate given by Corollary 4.5 or 4.6.
    α = α/‖L^(m)‖_2        % Make the bound relative
    if m > 1 and α < tol
        break out of loop
    end if
end for
return L^(m)
```

5. Application to condition number estimation. In this section we use our new algorithm for computing $L_f(A,E)b$ to design an algorithm for estimating the condition number of $f(A)b$, building upon work by Deadman [5]. In particular Deadman proves the following bounds on $\mathrm{cond}(f,A,b)$, which we defined in equation (1.3).

LEMMA 5.1. *Given $A \in \mathbb{C}^{n\times n}$, $b \in \mathbb{C}^n$, an induced matrix norm $\|\cdot\|$, and a matrix function f which is defined and Fréchet differentiable at A; the condition number* $\mathrm{cond}(f,A,b)$ *satisfies the following bounds.*

$$\frac{1}{\|f(A)b\|}\max\left(\|A\|\max_{\|E\|=1}\|L_f(A,E)b\|, \|f(A)\|\|b\|\right) \le \mathrm{cond}(f,A,b)$$
$$\le \frac{1}{\|f(A)b\|}\left(\|A\|\max_{\|E\|=1}\|L_f(A,E)b\| + \|f(A)\|\|b\|\right)$$

Proof. See [5, p. 5]. □

Deadman then proceeds to work in the 1-norm and shows that

$$\mathrm{cond}(f,A,b) \approx \left(2\sqrt{n}\gamma\|A\|_1 + \|f(A)\|_1\|b\|_1\right)/\|f(A)b\|_1, \tag{5.1}$$

where $\|A\|_1$ and $\|f(A)\|_1$ can be estimated using the 1-norm estimation algorithm by Higham and Tisseur [10]. The quantity γ is an 2-norm estimate of the matrix $K_f(A,b) \in \mathbb{C}^{n\times n^2}$ (not to be confused with the Krylov subspace) which is a linear operator such that $K_f(A,b)\,\mathrm{vec}(E) = \mathrm{vec}(L_f(A,E)b)$ for any matrix $E \in \mathbb{C}^{n\times n}$ [5, p. 7]. It is simple to show that $K_f(A,b) = (b\otimes I_n)K_f(A)$ where $\otimes$ represents the Kronecker product and $K_f(A)$ is the Kronecker form of the Fréchet derivative [9, Chap. 3].

Instead of following Deadman by approximating the 1-norm condition number we choose to estimate the 2-norm condition number, for which this approximation becomes

$$\mathrm{cond}(f,A,b) \approx (2\gamma\|A\|_2 + \|f(A)\|_2\|b\|_2)/\|f(A)b\|_2. \tag{5.2}$$

Note that this avoids introducing the $\sqrt{n}$ factor in the first term which, for large sparse matrices, could potentially dominate the approximation. We can then estimate $\|f(A)\|_2$ and $\|A\|_2$ using a power method. This leaves only the estimation of γ.

To estimate γ we first need to introduce the adjoint of a Fréchet derivative, $L_f^\star(A,E)$. For simplicity we will focus on matrix functions with real power series (such as the matrix exponential) for which we know that $L_f^\star(A,E) = L_f(A^*,E)$. For further details on the adjoint see Higham [9, p. 66], for example.

Using the adjoint, the value $\gamma \approx \|K_f(A,b)\|_2$ is estimated from the following algorithm given by Deadman [5].

ALGORITHM 5.2 (Condition number estimate - cf. [5, Alg. 3.3]). *For the function f, $A \in \mathbb{C}^{n\times n}$, and $b \in \mathbb{C}^n$ this algorithm computes an estimate $\gamma \le \|K_f(A,b)\|_2$.*

1 Choose a nonzero starting vector $y_0 \in \mathbb{C}^n$.
2 it_max $= 10$ % Choose the maximum number of iterations.
3 for $k = 0 : \infty$
4 $\quad y_{k+1} = L_f(A, L_f^\star(A, y_k b^*))b$
5 $\quad \gamma_{k+1} = \sqrt{\|y_{k+1}\|_2}$
6 $\quad$ if $|\gamma_{k+1} - \gamma_k| < 0.1\gamma_{k+1}$ or $k >$ it_max, $\gamma = \gamma_{k+1}$, quit, end
7 $\quad y_{k+1} = y_{k+1}/\|y_{k+1}\|_2$
8 end

The major cost of this algorithm is computing the derivative on line 4, which can be computed by extracting the top n elements of the vector

$$f\left(\begin{bmatrix} A & L_f^\star(A, y_k b^*) \\ 0 & A \end{bmatrix}\right)\begin{bmatrix} 0 \\ b \end{bmatrix}. \tag{5.3}$$

Let us denote the 2×2 block matrix in the above equation by X. If we intend to compute $f(X)v$ with a Krylov method, or any other method requiring only matrix–vector products, this leaves us with the inner subproblem of computing matrix–vector products with $L_f^\star(A, y_k b^*)$. Following the approach taken by Deadman, since $L_f^\star(A, y_k b^*) = L_f(A^*, y_k b^*)$ for functions with real power series expansions, we could compute $L_f^\star(A, y_k b^*)v$ as the top n elements of

$$f\left(\begin{bmatrix} A^* & y_k b^* \\ 0 & A^* \end{bmatrix}\right)\begin{bmatrix} 0 \\ v \end{bmatrix}.$$

However, since the direction term in the Fréchet derivative is of rank-1, our new algorithm can compute this inner subproblem in an extremely efficient manner. Note that, since the condition number does not generally need to be known to high precision, our new algorithm can be run with a low tolerance for even greater efficiency. Before we give the overall algorithm, we first define the following code fragment.

CODE FRAGMENT 5.3 ($w = g(z)$). *This code computes the matrix-vector product*

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} A & L_f^\star(A, y_k b^*) \\ 0 & A \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix},$$

where the direction term in the Fréchet derivative is of rank-1, and computed by Algorithm 3.2 or 4.7.

1 $w_1 = Az_1 + L_f(A^*, y_k b^*)z_2$
2 $w_2 = Az_2$
3 return $w = [w_1^T, w_2^T]^T$

Altogether, this leads to the following algorithm for computing $L_f(A, L_f^\star(A, y_k b^*))b$ on line 4 of Algorithm 5.2.

ALGORITHM 5.4. *Given $A \in \mathbb{C}^{n\times n}$, $y_k \in \mathbb{C}^n$ and $b \in \mathbb{C}^n$ with $\|b\|_2 = \|y_k\|_2 = 1$. The user-supplied routine* `fAb` *for which* `fAb`$(g, b) = f(A)b$, *using only matrix–vector products $w = g(z)$ evaluated by Code Fragment 5.3. The algorithm computes $x = L_f(A, L_f^\star(A, yb^*))b$.*

1 $v = [0, b^T]^T$
2 $x =$ `fAb`(g, v) g computes matrix–vector products using Code Fragment 5.3.
3 $x = x(1\colon n)$
4 return x

By combining the bound on the condition number (5.2) with the estimate of γ in Algorithm 5.2, where we replace line 4 with Algorithm 5.4, we obtain the following algorithm to estimate the 2-norm condition number.

ALGORITHM 5.5. *Given f, A, and b satisfying the criteria of Algorithms 3.2 and 4.7, and assuming that an algorithm for computing $f(A)v$ is available, the following algorithm computes $f(A)b$ and estimates its condition number $\kappa \approx \mathrm{cond}(f, A, b)$ in the 2-norm.*

1 Compute $f(A)b$ using any algorithm.
2 Estimate $\|A\|_2$ and $\|f(A)\|_2$ using the power method.
3 Compute γ using Algorithm 5.2, replacing line 4 with Algorithm 5.4.
4 $\kappa = (2\gamma\|A\|_1 + \|f(A)\|_2\|b\|_2)/\|f(A)b\|_2$.

6. Numerical experiments. In this section we will perform a number of numerical experiments to test the accuracy and efficiency of our new algorithms. All experiments were performed using MATLAB 2015a on a Linux machine (specifically the 64-bit variant – glnxa64) and all timings use only one processor. The Python code is run using the Anaconda distribution of Python 2.7, with the libraries scipy 0.17.0 and numpy 1.10.4, linked to Intel MKL BLAS.

We use the matrix exponential exclusively throughout our tests, since this allows us to test the performance of our algorithm using `expmv` [1] which has implementations in both MATLAB and Python. We can use `expmv` to compute $L_{\exp}(A, E)b$ using the 2×2 block formula from equation (3.1). We will refer to `expmv` used within the block formula for the Fréchet derivative as the "block algorithm".

Throughout our experiments the direction matrix E will be rank-1 such that $E = yz^*$, where y and z have elements sampled from a random normal $N(0, 1)$ distribution.

We will also need to compute $L_{\exp}(A, E)b$ in a highly accurate manner, so that we can compute the relative error achieved by our algorithm. We compute this by forming the 2×2 block formula (3.1) in variable-precision arithmetic from the MATLAB Symbolic Toolbox. We compute the function $f(X)$ in high-precision by taking an eigendecomposition $VDV^{-1} = X + \Delta X$ using 200 digits, where $\|\Delta X\|_2 = 10^{-100}$ is used to ensure that, with probability 1, the matrix is diagonalizable. We can then form $Vf(D)V^{-1}$ and round back to double precision floating point.

Our first two experiments are designed to investigate the behaviour of Algorithms 3.2 and 4.7. We compare the relative errors achieved by `expmv` and Algorithm 3.2 when aiming for half, single, or double precision accuracy over a range of test matrices. We also compare the amount of work performed by each algorithm.

Our second experiment shows the convergence of our new algorithm as m, the number of block Krylov iterations, increases. This allows us to compare the effectiveness of the two stopping criteria in Algorithms 3.2 and 4.7.

The third experiments illustrates the performance of our new algorithm on large, sparse matrices. We also illustrate the storage advantages of our algorithm in comparison to the block algorithm.

Finally, in our fourth experiment, we investigate the application of our algorithm in computing a bound of the condition number of $\mathrm{e}^A b$ for several different matrices.

EXPERIMENT 1 (Behaviour in floating point arithmetic). This experiment is designed to test the behaviour of the method in floating point arithmetic. To do this, we will check that the algorithms attain the prescribed relative error that they aim for. In particular we use the tolerances *half, single* and *double* which in IEEE floating point arithmetic correspond to 2^{-11}, 2^{-24} and 2^{-53}, respectively.

The test matrices used in this experiment are taken from The Matrix Computation Toolbox [8], where we set their dimension to $n = 100$ and compute $L_{\exp}(A, yz^*)b$ for y, z and b with elements sampled from a normal $N(0, 1)$ distribution. For reference

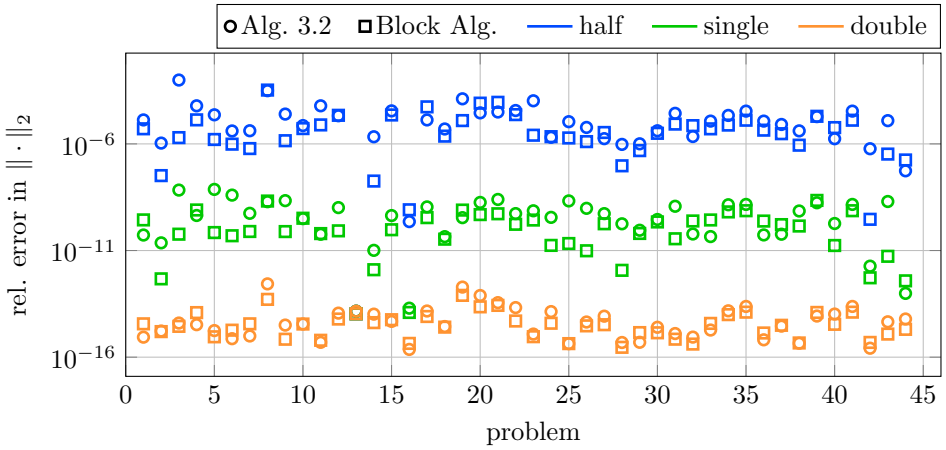

FIG. 6.1. *Relative errors obtained for the computation of $L_{\exp}(A,E)b$ by the new algorithm and the 2×2 block form when aiming for half, single, and double precision accuracy.*

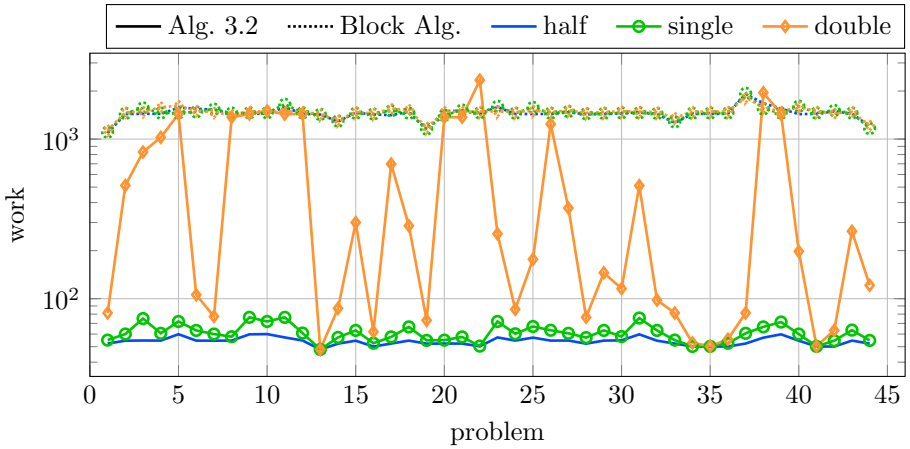

FIG. 6.2. *Units of work required to compute $L_f(A,E)b$ for the examples in Figure 6.1. Each unit of work is equivalent to one matrix–vector product with an $n \times n$ matrix.*

we also compute $L_{\exp}(A, yz^*)b$ by applying `expmv` to the 2×2 block matrix shown in equation (3.1), i.e. the block algorithm. We compute the exact solution, from which we calculate the relative error obtained, using variable precision arithmetic as described previously.

In an attempt to control the, sometimes extremely large, condition number of the matrices we rescaled them all to have unit 2-norm. Nevertheless, several of the matrices were still too ill-conditioned to obtain reasonable results with either method. The excluded matrices were those with indices $[4, 7, 12, 17, 18, 29, 35, 40]$ from the Matrix Computation Toolbox.

In Figure 6.1 we see the relative error obtained by each algorithm. The results show that, for each test matrix, both algorithms obtain a relative error close to the desired accuracy. In particular this shows that our new algorithm appears to behave in a forwards stable manner.

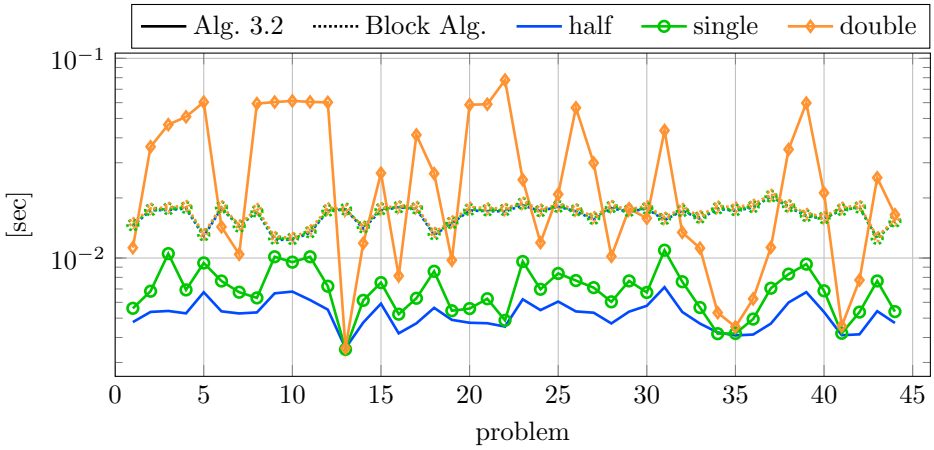

FIG. 6.3. *Wall time (in seconds) required for the computation of the examples in Figure 6.1.*

In Figure 6.2 we show the amount of work required by each algorithm. We define one unit of work to be the effort (in flops) required to compute one matrix–vector product with an $n \times n$ matrix. Therefore, if our new algorithm is using an $m \times m$ matrix on the current iteration then each matrix–vector product contributes $2m^2/2n^2$ units of work. We see that the block algorithm performs a similar amount of work regardless of the accuracy desired: they are barely distinguishable on the graph since all three lines overlap one another. Meanwhile our new algorithm is often cheaper when aiming for double precision accuracy and over a factor of 10 times cheaper for single and half precision accuracy.

In Figure 6.3 we see how the savings in work described above translates to savings in time. In order to obtain reliable timings we ran each code 500 times and display the mean time. Furthermore, we used the MATLAB options `-nojvm` and `-singleCompThread` to increase the reliability of the results.

We see that Algorithm 3.2 is faster when the desired accuracy is half or single precision. However, with a few exceptions, it is slower when we desire double precision accuracy. This is partially due to the fact that inside our implementation we evaluate the stopping criteria, equation 3.5, at the end of every iteration. We may be able to improve the runtime by, for example, checking the stopping criteria less frequently.

EXPERIMENT 2 (Convergence profile). In our second experiment we illustrate the convergence behaviour of Algorithm 3.2 and 4.7 with respect to the degree m of the block Arnoldi process, see Figure 6.4. This will allow us to compare the two stopping critera used in Algorithms 3.2 and 4.7, respectively.

For the convergence profile we have chosen the matrix A to be the `poisson2D` matrix from the University of Florida Sparse Matrix Collection [4]. The vectors b, y and z are chosen randomly with elements sampled from a normal $N(0, 1)$ distribution. As before, we are aiming to compute $L_{\exp}(A, yz^*)b$.

For the computation shown in Figure 6.4 the stopping criteria are ignored, we continue iterating until $m = n/2$. The exact solution is computed in variable precision arithmetic with 200 digits in the manner described at the beginning of this section. The graphs are cropped at $m = 25$ as, by this point, full double precision accuracy has already been achieved.

As we might expect, we see that the a priori estimate overestimates the number

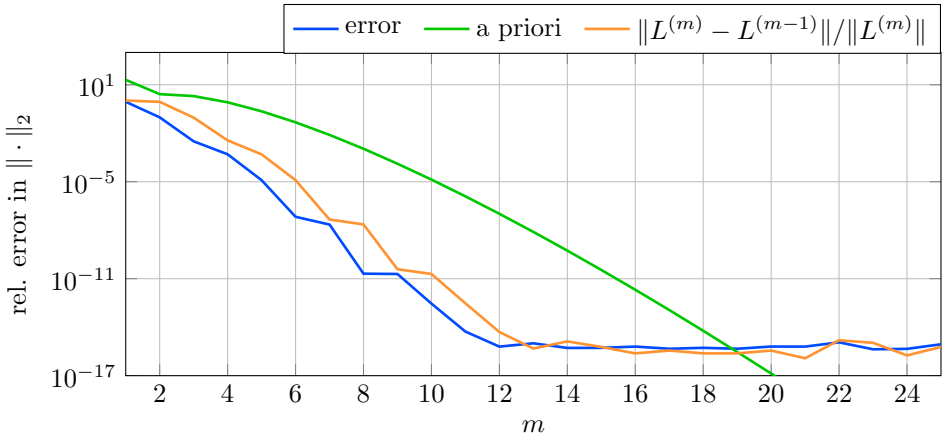

FIG. 6.4. *Convergence behaviour of Algorithm 3.2 and the a priori error estimate as well as the heuristic early termination criterion. The error is measured in the 2-norm.*

TABLE 6.1
Relative error in the 2-norm and the time (in seconds) for various test matrices when aiming for single precision accuracy.

	Alg. 3.2		block alg.	
Matrix	time	2-norm err.	time	2-norm err.
Gleich/minnesota	6.00e-02	3.90e-09	2.46e+00	3.19e-10
FEMLAB/poisson2D	8.32e-03	4.93e-10	5.75e-02	9.33e-11
Pajek/USAir97	1.32e-02	2.73e-09	5.17e-02	1.64e-09
Gset/G45	5.31e-02	8.46e-09	4.32e-01	2.51e-10
HB/gre_1107	1.36e-02	1.24e-09	5.08e-01	7.02e-10
vanHeukelum/cage8	2.20e-02	4.83e-09	3.98e-01	1.42e-10
MUFC Twitter	4.39e+00	1.16e-08	-	-

of iterations required to compute the Fréchet derivative at the desired accuracy but certainly provides an upper bound on the number of iterations required. On the other hand, we can see that the heuristic stopping criteria, $\|L^{(m)} - L^{(m-1)}\|/\|L^{(m)}\|$, captures the rate of convergence well.

EXPERIMENT 3 (Large scale matrices). In this experiment we compare Algorithm 3.2 and block algorithm for large scale matrices. We use examples from the University of Florida Sparse Matrix Collection [4] and the `MUFC Twitter` matrix[1].

From Table 6.1 we can see that we get similar results for large scale matrices to those seen in Experiment 1. We choose *single* precision as the desired accuracy, which is achieved in all of the examples. In order to get an estimate for the error we compute a reference solution by calling the block algorithm with *double* precision accuracy.

Furthermore, the `MUFC Twitter` matrix highlights the storage advantages of Algorithm 3.2 over the block algorithm. This is a 148918×148918 matrix with 193032 nonzero values. Due to the size of the matrix the rank-1 direction term in the Fréchet derivative needs approximately 165.2GB of storage since it is a dense matrix. As a result, a straight forward computation on a standard workstation is not possible. This

[1] Data provided under a Creative Commons licence by The University of Strathclyde and Bloom Agency. http://www.mathstat.strath.ac.uk/outreach/twitter/mufc

TABLE 6.2

Relative errors and condition numbers errors when computing the exponential of large sparse matrices using Algorithm 5.5 for the 2-norm and [5, Alg. 4.1] for the 1-norm.

Matrix	2-norm err.	2-norm cond u	1-norm err.	1-norm cond u
Gleich/minnesota	9.17e-11	1.11e-7	3.62e-11	3.18e-7
FEMLAB/poisson2D	4.34e-11	2.42e-8	4.01e-11	4.26e-8
Pajek/USAir97	9.14e-10	3.24e-7	7.68e-10	1.97e-6
Gset/G45	1.0	4.88e+4	1.0	5.48e-6
HB/gre_1107	1.46e-10	3.65e-8	1.09e-10	5.84e-8
vanHeukelum/cage8	3.29e-10	2.59e-7	1.25e-10	7.41e-7

is why there are no results for the block algorithm in this case. Algorithm 3.2, on the other hand, can compute with this matrix without running out of memory and obtains a result in around 4 seconds for *single* precision accuracy.

EXPERIMENT 4 (Condition numbers). Our final experiment shows how our algorithm can be used to bound the condition number of computing $\mathrm{e}^A b$ by combining Algorithms 4.7 and 5.5. For each of the matrices in Table 6.1, minus the MUFC Twitter which was too large for the intermediate calculations to fit in memory, we report the relative error obtained when comparing our new algorithm to the exact solution and the estimated condition number (multiplied by the machine epsilon $u = 2^{-53}$) in Table 6.2. The exact solution is computed using variable precision arithmetic as explained at the beginning of this section. In order to avoid overflow the matrices were scaled such that $\|A\|_2 = 1$ and the vectors b were chosen to have elements sampled from a normal $N(0, 1)$ distribution. In all cases we can see that the relative error is bounded above by the condition number times u, meaning that our method behaves in a forward stable manner.

For comparison we have also included the 1-norm relative error and 1-norm condition number, where the latter is computed via the Python code corresponding to [5, Alg. 4.1][2]. It is interesting to note that for the extremely ill-conditioned matrix "Gset/G45" estimating the condition number with Algorithm 5.5 would inform the user that the computed solution is untrustworthy whilst the 1-norm algorithm returns a condition number estimate that is orders of magnitude smaller than the observed relative error. This discrepancy is likely due to the design choice mentioned by Deadman [5, p. 13] to only choose a single set of parameters for computing all the matrix exponentials required by the algorithm: clearly this can lead to poor condition number estimates in some cases.

7. Conclusions. We have derived a new method to compute the action of the Fréchet derivative of a matrix function on a vector for low rank direction matrices E, making use of block Krylov subspace approximations. After introducing the method we showed how one can obtain rigorous a priori error bounds to ensure that the action of the Fréchet derivative is computed to sufficient accuracy. Our numerical experiments confirm that our error bounds are effective in practice. Furthermore, we observe that typically the size of the required Krylov subspace is much smaller than the size of the original matrix. This allows the action of the Fréchet derivative to be computed much more efficiently than by other currently used methods.

[2]Download available at https://github.com/edvindeadman/fAbcond. Accessed on 14.12.2015.

We have also shown how our algorithm can be used to efficiently estimate the condition number of computing $f(A)b$ in the 2-norm. The growing use of matrix functions in a variety of areas within applied mathematics makes it vital to have efficient condition number estimates available. Our experiments also show that our condition number estimator can detect ill-conditioned problems, where the returned solution is likely to be suspect, that are not detected by competing algorithms.

Looking towards the future, the block Krylov scheme used by our algorithms is relatively basic at this stage. By incorporating data reuse, implicit restarting, rational Krylov approximations, and other state-of-the-art techniques, it is likely that the action of the Fréchet derivative could be computed even more efficiently. This would have a dramatic impact on the computation of the condition number, for which multiple Fréchet derivative computations are required.

REFERENCES

[1] Awad H. Al-Mohy and Nicholas J. Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM J. Sci. Comput.*, 33(2):488–511, 2011.

[2] Marco Caliari, Peter Kandolf, Alexander Ostermann, and Stefan Rainer. Comparison of software for computing the action of the matrix exponential. *BIT Numerical Mathematics*, 54 (1):113–128, 2014.

[3] Marco Caliari, Marco Vianello, and Luca Bergamaschi. Interpolating discrete advection-diffusion propergators at Leja sequences. *J. Comput. Appl. Math.*, 172(1):79–99, 2004.

[4] Tim Davis and Yifan Hu. The university of florida sparse matrix collection, 2011. 1–25 pp.

[5] Edvin Deadman. Estimating the condition number of $f(A)b$. *Numerical Algorithms*, 70(2): 287–308, 2014.

[6] Ernesto Estrada and Desmond J. Higham. Network properties revealed through matrix functions. *SIAM Review*, 52(4):696–714, 2010.

[7] Ernesto Estrada, Desmond J. Higham, and Naomichi Hatano. Communicability betweeness in complex networks. *Physica A: Statistical Mechanics and its Applications*, 388(5):764–774, 2009.

[8] Nicholas J. Higham. The Matrix Computation Toolbox, September 2002. `http://www.maths.manchester.ac.uk/~higham/mctoolbox`.

[9] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008. xx+425 pp. ISBN 978-0-898716-46-7.

[10] Nicholas J. Higham and Françoise Tisseur. A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. *SIAM J. Matrix Anal. Appl.*, 21(4):1185–1201, 2000.

[11] David Hipp, Marlis Hochbruck, and Alexander Ostermann. An exponential integrator for non-autonomous parabolic problems. *Electronic Transactions on Numerical Analysis*, 41: 497–511, 2014.

[12] Marlis Hochbruck, Christian Lubich, and Hubert Selhofer. Exponential integrators for large systems of differential equations. *SIAM J. Sci. Comput.*, 19(5):1552–1574, 1998.

[13] Dominik L. Michels, Gerrit A. Sobottka, and Andreas G. Weber. Exponential integrators for stiff elastodynamic problems. *ACM Trans. Graph.*, 33(1):7:1–7:20, 2014.

[14] Axel Ruhe. Implentation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Math. Comp.*, 33(146):680–687, 1979.

[15] Yousef Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29(1):209–228, 1992.

[16] Yousef Saad. *Itervative Methods for Sparse Linear Systems*. Second edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003. xvii+520 pp. ISBN 978-0-89871-534-7.