Journal of Computational and Applied Mathematics 319 (2017) 87–96

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Computing humps of the matrix exponential

Yu.M. Nechepurenko [a], M. Sadkane [b,*]

[a] Institute of Numerical Mathematics, Russian Academy of Sciences, ul. Gubkina 8, Moscow, 119333, Russia

[b] Université de Brest, CNRS-UMR 6205, Laboratoire de Mathématiques de Bretagne Atlantique, 6 avenue Victor Le Gorgeu, CS 93837, 29285 Brest Cedex 3, France

ARTICLE INFO

Article history:
Received 10 April 2016

Keywords:
Matrix exponential norm
Time integration method
Krylov subspace method
Truncated Taylor series method
Lanczos method
Alternating maximization

ABSTRACT

This work is devoted to finding maxima of the function $\varGamma(t) = \|\exp(tA)\|_2$ where $t \geq 0$ and A is a large sparse matrix whose eigenvalues have negative real parts but whose numerical range includes points with positive real parts. Four methods for computing $\varGamma(t)$ are considered which all use a special Lanczos method applied to the matrix $\exp(tA^*)\exp(tA)$ and exploit the sparseness of A through matrix–vector products. In any of these methods the function $\varGamma(t)$ is computed at points of a given coarse grid to localize its maxima, and then maximized by a standard maximization procedure or via an alternating maximization procedure. Results of such computations with some test matrices are reported and analyzed.

1. Introduction

This paper is concerned with the computation of maxima of the function

$$\varGamma(t) = \|\exp(tA)\|_2 \tag{1}$$

in a given nonnegative interval of time t, where A is an $n \times n$ matrix with negative spectral abscissa, i.e., the largest real part of eigenvalues of A, denoted by $\alpha(A)$, is negative but whose numerical range includes points with positive real parts, i.e., the largest eigenvalue of the Hermitian matrix $(A + A^*)/2$, which will be denoted by $\mu(A)$, is positive. Such maxima will be referred to as humps for the matrix exponential. The humps are needed, for example, for determining the transient growth in fluid mechanics, see, e.g., [1–5] and [6, chap. 4–5]. They may also be of interest in linear stability analysis of steady states of ordinary differential equations, see, e.g., [7–10].

It is known [11] that the eigenvalues of a Hermitian matrix whose entries depend analytically on a real parameter admit an analytic parametrization. Due to this fact and to the nonsingularity and analyticity of $\exp(tA)$ with respect to t, the function (1) can be represented as

$$\varGamma(t) = \max\{\varGamma_1(t), \ldots, \varGamma_n(t)\},$$

where $\varGamma_1(t), \ldots, \varGamma_n(t)$ are positive-valued analytic functions which form at each t the set of singular values (counting multiplicities) of $\exp(tA)$ or the set of positive square roots of eigenvalues of the Hermitian positive definite matrix

$$H(t) = \exp(tA^*)\exp(tA). \tag{2}$$

* Corresponding author.
E-mail addresses: yumn@inm.ras.ru (Yu.M. Nechepurenko), miloud.sadkane@univ-brest.fr (M. Sadkane).

http://dx.doi.org/10.1016/j.cam.2016.12.031
0377-0427/© 2016 Elsevier B.V. All rights reserved.

Therefore, the function (1) is continuous and differentiable on the left and on the right at each t. In addition, from formulas for derivatives of eigenvalues of Hermitian matrices (see, e.g., [12, chap. 2]), we have for any $t_* \geq 0$

$$2\Gamma(t_*)\,\frac{d\Gamma}{dt}(t_*-) = \min_{v\in\mathcal{V}_*}\left(\frac{dH}{dt}(t_*)v,\, v\right), \tag{3}$$

$$2\Gamma(t_*)\,\frac{d\Gamma}{dt}(t_*+) = \max_{v\in\mathcal{V}_*}\left(\frac{dH}{dt}(t_*)v,\, v\right), \tag{4}$$

where $\mathcal{V}_* \subset \mathbb{C}^n$ is the set of normalized eigenvectors corresponding to the largest eigenvalue $\Gamma(t_*)^2$ of $H(t_*)$.

Under the conditions on A stated above and the following widely-used equality (see, e.g., [6, chap. 4])

$$\frac{d\Gamma}{dt}(0+) = \mu(A)$$

which is a direct consequence of (4), we deduce that the global maximum $\Gamma_{\mathrm{opt}} = \Gamma(t_{\mathrm{opt}})$ is finite and larger than 1, where

$$t_{\mathrm{opt}} \in \arg\max_{t\geq 0} \Gamma(t). \tag{5}$$

Moreover, as is noticed in [13], $0 < t_{\mathrm{opt}} < t_{\min}$ where

$$t_{\min} = \inf\{t > 0 : \Gamma(t) < 1\}. \tag{6}$$

Indeed, if $t > t_{\min}$ then due to the continuity of $\Gamma(t)$ there exists $t' : t_{\min} < t' < t$ such that $\Gamma(t') < 1$ and therefore

$$\Gamma(t) \leq \Gamma(t-t')\Gamma(t') < \Gamma_{\mathrm{opt}}.$$

As to the local maxima, they can be larger, smaller, or equal to 1 and there is no a priori information about their location.

The algorithm proposed in [13] computes the function $\Gamma(t)$, and hence its maxima, with a given accuracy. The idea of the algorithm is to use the reordered Schur triangular form S instead of A and monitor the decay of the rows at the bottom of $\exp(tS)$ as t increases and to replace $\exp(tS)$ by a low-rank approximation obtained by removing those rows whose 2-norm is smaller than a given threshold. The resulting algorithm is significantly faster than the one using the scaling and squaring method [14]. However, due to the Schur decomposition this algorithm cannot be used for large sparse matrices. The aim of the present paper is to propose and compare different algorithms based essentially on the action of the matrix exponential on a vector which can be done in an economical way using the structure of A.

This paper is organized as follows. In Section 2, we briefly describe three approaches to compute the action of the matrix exponential which will be used in numerical experiments throughout the paper: the first one, which we refer to as "time integration method" (though this term would also apply to the two other methods) uses the two-step backward differentiation scheme [15]; the second one is an iterative method based on Krylov subspace approximation to the exponential [16–19]; and the third one is a direct method based on truncated Taylor series approximation to the exponential [20]. In Section 3 we discuss the computation of $\Gamma(t)$ using a special Lanzcos process applied to the matrix $H(t)$. In Section 4 we discuss how to find a hump of $\Gamma(t)$. We consider first the most obvious way which consists in using the well-known minimization procedure `fmin` [21] which is implemented in MATLAB's `fminbnd` function and can be applied to the function $-\Gamma(t)$ to find maxima of $\Gamma(t)$. Then we consider the auxiliary function

$$\gamma(t, v) = \|\exp(tA)v\|_2/\|v\|_2, \tag{7}$$

describe its properties and use it within an alternating maximization procedure. For illustration purposes, we will use the following four matrices.

Matrix 1: A matrix of order $n = 1090$ with $\mu(A) \approx 9.11 \cdot 10^5$ and $\alpha(A) \approx -0.156$ which is obtained from the stability analysis of an aircraft structure and is taken from the NEP collection.[1]

Matrix 2: A matrix of order $n = 90\,000$ with $\mu(A) \approx 21.4$ and $\alpha(A) \approx -7.04 \cdot 10^{-3}$ which is taken from [22] and is obtained from a discretization of the elliptic operator

$$L = -\frac{\partial}{\partial x}u - \frac{\partial}{\partial y}v + \nu\Delta$$

in the square $(x, y) \in (0, 1) \times (0, 1)$ with Dirichlet boundary conditions, using a uniform grid with a number of inner nodes in each direction equal to 300 and the standard second order approximation. Here Δ denotes the 2D Laplace operator, $\nu = 5 \times 10^{-4}$ and

$$u = \frac{\partial\varphi}{\partial y}, \qquad v = -\frac{\partial\varphi}{\partial x}, \qquad \varphi(x, y) = \frac{1}{16\pi}\cos(8\pi x^2)\cos(8\pi y^2).$$

[1] See http://math.nist.gov/MatrixMarket/.

Matrix 3: An airfoil matrix of order $n = 23\,560$ with $\mu(A) \approx 246$ and $\alpha(A) \approx -0.273$ which is also taken from the NEP collection and is obtained from the transient stability analysis of Navier–Stokes equations.

Matrix 4: An artificial upper bi-diagonal matrix of order $n = 1000$ with $\mu(A) \approx 0.791$ and $\alpha(A) = -0.01$ whose nonzero entries are given by $A_{kk} = -0.01\,k^2,\ k = 1, \dots, n$ and $A_{k,k+1} = 1,\ k = 1, \dots, n-1$. This matrix is taken from [13].

All computations are carried out in MATLAB 7.14 on an Intel Core i7 CPU at 2.80 GHz.

2. Computing the action of the matrix exponential

At the heart of the methods considered in this paper is the action of matrix exponentials on vectors (more precisely, operations of the form $\exp(tA)v$ and $\exp(tA^*)v$ for given t, A and v). For this, we have chosen four methods which are briefly described and specified in this section.

2.1. Time integration methods

The action $\exp(tA)v$ can be viewed as the solution of the initial value problem

$$u(0) = v, \qquad \frac{du}{dt} = Au(t),$$

which can be computed using any time integration method, for example, the two-step backward differentiation method (BDF2). It is given by (see, e.g., [15]):

$$\frac{1.5u_j - 2u_{j-1} + 0.5u_{j-2}}{\tau} = Au_j, \quad j \ge 2, \tag{8}$$

where the vector u_j approximates $\exp(tA)v$ at $t = j\tau$, $u_0 = v$ and u_1 is obtained by one step of the implicit Euler method:

$$\frac{u_1 - u_0}{\tau} = Au_1. \tag{9}$$

For such a method we have an a priori error of the form

$$\|u_j - \exp(j\tau A)v\|_2 = \mathcal{O}(\tau^2).$$

The system of Eqs. (9) and (8) with $2 \le j \le N$ can be written in the matrix form

$$Mu = E_1 u_0, \tag{10}$$

where $u = (u_0^T, u_1^T, \dots, u_N^T)^T$ while

$$M = \begin{pmatrix} I & & & & \\ -I & I - \tau A & & & \\ \frac{1}{2}I & -2I & \frac{3}{2}I - \tau A & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{2}I & -2I & \frac{3}{2}I - \tau A \end{pmatrix} \quad \text{and} \quad E_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ I \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

are respectively square block lower tridiagonal matrix of order $(N+1)n$ and the rectangular matrix of size $(N+1)n \times n$ formed by the columns $(j-1)n+1, \dots, jn$ of the identity matrix of order $(N+1)n$, where I denotes the identity matrix of order n.

From (10) we obtain

$$u_N = E_N^T M^{-1} E_1 u_0, \tag{11}$$

and therefore the matrix $E_N^T M^{-1} E_1$ is an approximation of the matrix exponential $\exp(N\tau A)$. The conjugate transpose matrix $E_1^T M^{-*} E_N$ gives the corresponding approximation of $\exp(N\tau A^*)$. The matrices M^{-1} and M^{-*} are not computed. Instead, we solve linear systems with block-triangular matrices M and M^*. The systems with the diagonal blocks, i.e. $I - \tau A$ and $1.5I - \tau A$ and their conjugate transpose matrices, are solved either directly or iteratively.

2.2. Krylov subspace method

Starting with an arbitrary normalized vector v, the Arnoldi process [23] applied to A with initial vector $v_1 = v$ yields, after m steps, an $n \times m$ matrix $V_m = [v_1, v_2, \dots, v_m]$ whose columns span an orthonormal basis of the Krylov subspace

$$K_m(A, v) = \operatorname{span}\{v, Av, \dots, A^{m-1}v\}$$

and a Hessenberg $m \times m$ matrix $H_m = (h_{i,j})_{1\le i,j\le m} = V_m^* A V_m$ with positive subdiagonal elements such that

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^*,$$

where e_k denotes the kth column of the identity matrix of order m. Then for any polynomial p_j of degree $j \le m-1$ we have $p_j(A)v = V_m p_j(H_m)e_1$ which, for a given t, motivates the approximation (see [16])

$$\exp(tA)v \approx V_m \exp(tH_m)e_1.$$

The MATLAB function `expv` (see [17]) implements this procedure which we use in our numerical tests. The same function is applied to the matrix A^* instead of A for computing $\exp(tA^*)v$.

2.3. Truncated Taylor series method

The method proposed in [20] uses a truncated Taylor series approximation to the exponential and a scaling analogous to the one of the scaling and squaring method [14]. The method exploits the structure of A through matrix–vector products and its cost is dominated by these products. It is shown in [20] that the computed value of $\exp(tA)v$ is given by $\exp(t(A + \Delta))v$, where Δ satisfies $\|\Delta\|_2 \le \|A\|_2\delta$ where δ is the required accuracy. This method is implemented by the author of [20] in MATLAB function `expmv`.

2.4. Specification of the methods

In all computations discussed in the next sections, the functions `expv` and `expmv` are used with their default setting, both invoked with three parameters: t, A and v. For time integration methods, the time integration step τ is chosen equal to $2\cdot 10^{-5}$, $1.5\cdot 10^{-3}$, 10^{-2} and 10^{-1} for Matrices 1–4, respectively. The linear systems with matrices $I - \tau A$ and $1.5I - \tau A$ and their conjugate transpose matrices are solved either by a direct solver using LU decompositions of $I - \tau A$ and $1.5I - \tau A$ with pivoting or by GMRES [23] using right `ILU` preconditioning with pivoting. In the latter case the drop tolerance is fixed at 10^{-4} and a default value of 30 is used for the dimension of the Krylov basis. Since in our computations the time integration step τ is fixed, the LU or `ILU` decompositions need only be performed once. Note that the dimension of the Krylov basis in the default setting of `expv` is also equal to 30. In the sequel, these methods for computing the matrix exponential action on a vector will be referred to as `expv`, `expmv`, `TI-LU` and `TI-GMRES`. Since Matrix 4 is upper triangular, its LU decomposition is not computed and `TI-GMRES` is not used.

3. Computing $\Gamma(t)$

For a fixed t one can apply the Lanczos algorithm [24] to the matrix $H(t)$ given in (2), and compute its largest eigenvalue whose square root gives an approximate value of $\Gamma(t)$. Starting with a normalized vector q_1, this algorithm constructs at step l a matrix $Q_l = [q_1, q_2, \dots, q_l]$ whose columns form an orthonormal basis of the Krylov subspace $K_l(H(t), q_1)$.

Let $T_l = \operatorname{tridiag}(\beta_{i-1}, \alpha_i, \beta_i)$ be the real symmetric tridiagonal matrix whose diagonal and sub-(upper) diagonal entries are given respectively by $\alpha_i = q_i^* H(t) q_i$, $1 \le i \le l$, and $\beta_i = q_i^* H(t) q_{i+1}$, $1 \le i \le l-1$. Then the interlacing properties of Hermitian matrices (see [24]) ensure that the sequence $\lambda_{\max}(T_l)$, where $\lambda_{\max}$ denotes the largest eigenvalue, is nondecreasing. As l increases, this value tends to the largest eigenvalue of $H(t)$. However, increasing l leads to an increase of storage and computational requirements. To avoid this drawback, we stop the algorithm when the increase of $\lambda_{\max}(T_l)$ becomes too slow or when l reaches a maximum allowable number of iterations. As our experiments have shown, in practice, only a few iterations are generally required to obtain a reasonable estimate of the largest singular value. Then the Ritz vector $v = Q_l y_l$, where y_l is the eigenvector associated with $\lambda_{\max}(T_l)$, constitutes an approximate eigenvector associated with the largest eigenvalue of $H(t)$.

The main steps are sketched in Algorithm 1. Each iteration of this algorithm requires a matrix–vector multiplication of the form $v = H(t)q$ for a given vector q, performed as the actions $w = \exp(tA)q$ and $v = \exp(tA^*)w$ by one of the methods described in Section 2. Since the Lanczos orthogonalization is not carried out exactly, errors may arise in the Lanczos vectors and hence in the approximate eigenvalue and eigenvector. For this reason the Lanczos algorithm is followed by one iteration of the power method. In all computations with Algorithm 1, the initial vector is taken at random, the maximum number of iterations $l_{\max}$ and the parameter tol in the `while` loop are respectively equal to 40 and 10^{-6}. These parameters were fixed after several trials. No significant improvement was made by increasing $l_{\max}$ or decreasing tol.

Note that more sophisticated methods such as Lanczos bidiagonalization [25] applied to $\exp(tA)$ could be used but our practice has shown that Algorithm 1 is quite efficient for our aim.

Algorithm 1 Lanczos method for computing $\Gamma(t)$.

```
Input: $t, A$, initial vector $v$, tolerance parameter tol, maximum number of Lanczos vectors $l_{\max}$.
Output: approximation of the largest singular value $s$ and corresponding right singular vector $v$.
  if $l_{\max} > 1$ then
    $s_{-1} = s_0 = 0,\ q_0 = 0,\ \beta_0 = \|v\|_2,\ l = 0$
    while $l < l_{\max} - 1$ and $\beta_l > 0$ and $s_l \geq (1 + \text{tol})s_{l-1}$ do
      $l := l + 1,\ q_l = v/\beta_{l-1}$
      $w = H(t)q_l,\ \alpha_l = q_l^* w$
      $v = w - \beta_{l-1}q_{l-1},\ \beta_l = \|v\|_2$
      Form the $l \times l$ tridiagonal matrix $T_l = \text{tridiag}\,(\beta_{i-1}, \alpha_i, \beta_i)$
      $s_l = \sqrt{\lambda_{\max}(T_l)}$
    end while
    Compute the eigenvector $y$ associated with the largest eigenvalue of $T_l$
    $v = [q_1, \ldots, q_l]y$
  end if
  Add one step of the power method:
  $w = \exp(tA)v,\ v = w/\|w\|_2$
  $w = \exp(tA^*)v,\ s = \|w\|_2,\ v = w/s$
```

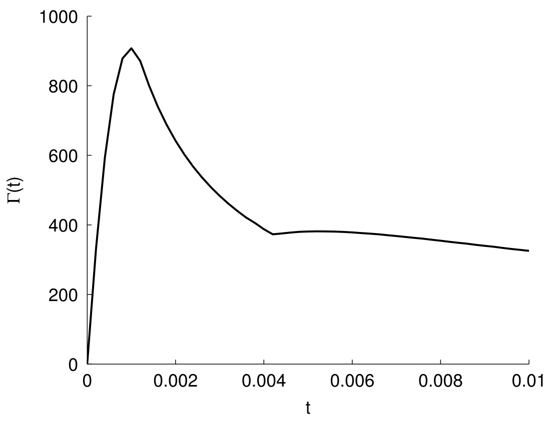
Table 1
Run time (sec) and number of calls to Algorithm 1.

A	Action	Computing $\Gamma(t)$	Maximizing	$\Gamma(t)$	Maximizing	$\gamma(t, v)$
		Run time	Run time	k	Run time	k
1	`expv`	34.56	4.47	8	0.26	2
	`expmv`	30.14	4.45	8	0.27	2
	`TI-LU`	59.70	2.22	10	0.08	1
	`TI-GMRES`	625.80	23.37	9	0.50	1
2	`expv`	297.88	192.54	7	116.66	7
	`expmv`	93.77	58.60	7	35.28	7
	`TI-LU`	1201.63	640.20	6	176.92	4
	`TI-GMRES`	1349.60	679.02	6	215.06	4
3	`expv`	2364.63	329.12	7	249.14	4
	`expmv`	2923.35	406.26	7	311.29	4
	`TI-LU`	21 619.34	3397.31	7	1028.50	3
	`TI-GMRES`	18 558.76	2079.06	7	720.28	3
4	`expv`	730.26	676.71	8	461.48	3
	`expmv`	6323.60	3670.83	8	2913.26	3
	`TI-LU`	2.90	3.24	8	1.23	2

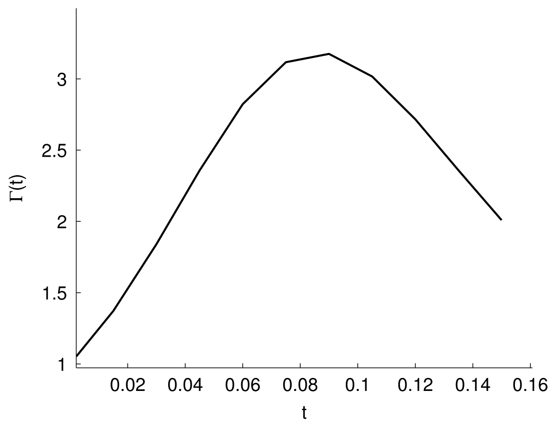
Using Algorithm 1 we compute $\Gamma(t)$ on a coarse grid in an interval of the form $0 \leq t \leq t_{\max}$. This preliminary computation helps understand the behavior of $\Gamma(t)$ and provides rough locations of humps to be refined later. There are several ways of choosing $t_{\max}$. For example, it can be chosen a priori as an upper bound of $t_{\min}$ in (6) which can be obtained numerically or theoretically using any known upper bound of $\Gamma(t)$ of the form $c\exp(-\kappa t)$, where c and κ are positive constants depending on A (see, e.g., [8,26,27,6] and references therein), or a posteriori by carrying out computations of $\Gamma(t)$ until $\Gamma(t) < 1$. In our experiments we compute $\Gamma(t)$ at a few points t and monitor the variation of $\Gamma(t)$. The computation is stopped when an increase of $\Gamma(t)$ followed by a marked decrease is observed. Typical results for the test matrices are shown in Fig. 1 and the third column of Table 1. These results depend a little on the initial random vectors used in Algorithm 1. The figures display the function $\Gamma(t)$ computed at equidistant points $t = 0, h, \ldots, Mh = t_{\max}$. For each matrix the value of h was chosen so that h/τ is integer, where τ is the time integration step defined in Section 2.4. To keep the computational cost low, we used some relatively large values of h leading to $M = 50$ for Matrices 1 and 3, and $M = 10$ for Matrices 2 and 4.

There are four indistinguishable curves in each subfigure (a), (b) and (c) corresponding to the use of `expv`, `expmv`, `TI-LU` and `TI-GMRES` and three indistinguishable curves in subfigure (d) corresponding to the use of `expv`, `expmv` and `TI-LU`. Additional computations have shown that the curves obtained by `TI-LU` and `TI-GMRES` with larger τ may differ from the ones obtained by `expv` or `expmv`. This, combined with the third column of Table 1 shows, with the exception of Matrix 4, the advantage in using `expv` and `expmv`. The advantage in using `TI-LU` for Matrix 4 is explained by the ability to solve systems with this matrix very cheaply.

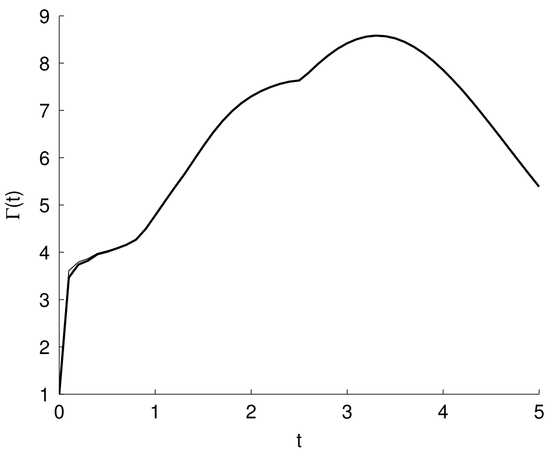
Table 2 presents the number of nonzero entries of $I - \tau A$, $1.5I - \tau A$ and their LU and ILU decompositions. This table shows that the numbers of nonzero entries in the LU and ILU decompositions of Matrix 1 are almost the same while in the case of Matrix 2 or 3 the ILU decomposition has significantly smaller number of nonzero entries. This explains a significant advantage of `TI-LU` in run time in comparison with `TI-GMRES` for Matrix 1.



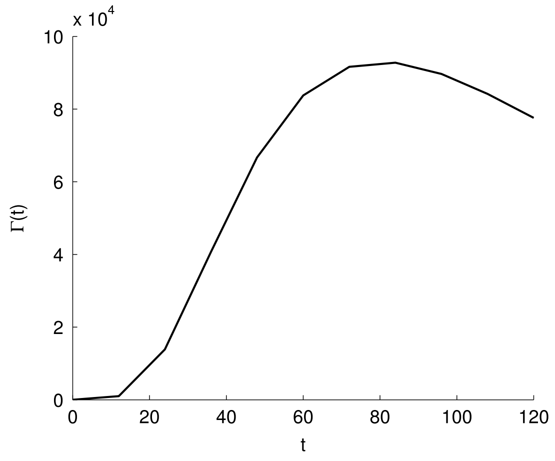
(a) $\Gamma(t), 0 \le t \le 0.01$, for Matrix 1.



(b) $\Gamma(t), 0 \le t \le 0.15$, for Matrix 2.



(c) $\Gamma(t), 0 \le t \le 5$, for Matrix 3.



(d) $\Gamma(t), 0 \le t \le 120$, for Matrix 4.

Fig. 1. Test matrices.

Table 2
Number of nonzero elements.

			LU		ILU	
1	$I - \tau A$	3 764	2 433	2 829	1 492	1 923
	$1.5I - \tau A$	3 764	2 433	2 829	1 492	1 923
2	$I - \tau A$	448 800	2 814 460	2 812 491	362 191	360 425
	$1.5I - \tau A$	448 800	2 788 784	2 785 953	338 908	337 971
3	$I - \tau A$	460 598	4 170 211	4 171 263	443 466	454 635
	$1.5I - \tau A$	460 598	4 170 211	4 171 263	392 348	399 180

4. Computing humps with maximization procedures

From Fig. 1 we choose for each matrix A an interval of t which contains t_{opt}. For example, we take the intervals $0 \le t \le 3.8 \cdot 10^{-3}$, $0 \le t \le 0.15$, $3 \le t \le 4.6$ and $0 \le t \le 120$ for Matrices 1–4, respectively. Then a further localization of t_{opt} is carried out using the function `fminbnd` and computing values of $\Gamma(t)$ by Algorithm 1 to find a local minimum of the function $-\Gamma(t)$ in the chosen interval. When the time integration method is used with time step τ, `fminbnd` is applied to the function $-\Gamma(\tau[t/\tau])$ instead of $-\Gamma(t)$, where $[\xi]$ denotes the integer part of ξ. Doing this we avoid the computation of complete or incomplete LU decomposition at each new value of t.

The main results are given in the fourth and fifth columns of Table 1, where in the fifth column, k denotes the number of evaluations of $-\Gamma(t)$ required for the minimization of this function by `fminbnd`, and the third and fourth columns of Table 3, where $\hat{t}_{\text{opt}}$ and $\hat{\Gamma}_{\text{opt}}$ denote the computed approximate values of t_{opt} and Γ_{opt} respectively. For Matrices 2 and 3 these results are consistent with those of Section 3 on the comparative efficiency of `expv`, `expmv`, `TI-LU` and `TI-GMRES`. In particular, we see a clear advantage in using `expmv` for Matrix 2 and an advantage in using `expv` for Matrix 3. On the other

Table 3
Results of maximization procedures.

A	Action	Maximizing $\hat{t}_{\text{opt}}$	$\Gamma(t)$ $\hat{\Gamma}_{\text{opt}}$	Maximizing $\hat{t}_{\text{opt}}$	$\gamma(t,v)$ $\hat{\Gamma}_{\text{opt}}$
1	`expv`	$9.8247 \cdot 10^{-4}$	$9.0811 \cdot 10^{2}$	$9.8252 \cdot 10^{-4}$	$9.0812 \cdot 10^{2}$
	`expmv`	$9.8199 \cdot 10^{-4}$	$9.0811 \cdot 10^{2}$	$9.8252 \cdot 10^{-4}$	$9.0812 \cdot 10^{2}$
	`TI-LU`	$9.8443 \cdot 10^{-4}$	$9.0777 \cdot 10^{2}$	$9.8000 \cdot 10^{-4}$	$9.0778 \cdot 10^{2}$
	`TI-GMRES`	$9.8450 \cdot 10^{-4}$	$9.0777 \cdot 10^{2}$	$9.8000 \cdot 10^{-4}$	$9.0778 \cdot 10^{2}$
2	`expv`	$8.6069 \cdot 10^{-2}$	3.1832	$8.6059 \cdot 10^{-2}$	3.1832
	`expmv`	$8.6069 \cdot 10^{-2}$	3.1832	$8.6059 \cdot 10^{-2}$	3.1832
	`TI-LU`	$8.8151 \cdot 10^{-2}$	3.1819	$8.5500 \cdot 10^{-2}$	3.1818
	`TI-GMRES`	$8.7029 \cdot 10^{-2}$	3.1819	$8.5500 \cdot 10^{-2}$	3.1818
3	`expv`	3.3120	8.5810	3.3100	8.5810
	`expmv`	3.3158	8.5811	3.3100	8.5810
	`TI-LU`	3.3108	8.5811	3.3100	8.5811
	`TI-GMRES`	3.3158	8.5811	3.3100	8.5811
4	`expv`	$8.0395 \cdot 10^{1}$	$9.2992 \cdot 10^{4}$	$8.0395 \cdot 10^{1}$	$9.2992 \cdot 10^{4}$
	`expmv`	$8.0395 \cdot 10^{1}$	$9.2992 \cdot 10^{4}$	$8.0395 \cdot 10^{1}$	$9.2992 \cdot 10^{4}$
	`TI-LU`	$8.0439 \cdot 10^{1}$	$9.2992 \cdot 10^{4}$	$8.0400 \cdot 10^{1}$	$9.2992 \cdot 10^{4}$

Table 4
Ratios of run time for computing $\Gamma(t)$ to that of maximizing $\Gamma(t)$ and their estimates.

A	Action	Run time ratio	Its estimate
1	`expv`	7.73	6.25
	`expmv`	6.77	6.25
	`TI-LU`	26.89	25.40
	`TI-GMRES`	26.77	28.21
2	`expv`	1.54	1.43
	`expmv`	1.60	1.43
	`TI-LU`	1.87	1.42
	`TI-GMRES`	1.98	1.44
3	`expv`	7.18	7.14
	`expmv`	7.19	7.14
	`TI-LU`	6.36	5.39
	`TI-GMRES`	8.92	5.38
4	`expv`	1.07	1.25
	`expmv`	1.72	1.25
	`TI-LU`	0.89	0.93

hand, the comparative efficiency for Matrix 1 has changed. The advantage in run time of the first two methods is now not so clear as for computing $\Gamma(t)$. Furthermore, the smallest run time is achieved with `TI-LU`. Since this method yields a bit lower accuracy than `expv` or `expmv` ($\hat{\Gamma}_{\text{opt}}$ computed with `TI-LU` is smaller) we can conclude that the first three methods have almost equal efficiency for computations with Matrix 1.

The difference between the results for Matrix 1 and Matrices 2 and 3 is explained by a difference in location of the hump in the search interval and a difference in the ratio M/k. Indeed, the run time of a time integration method when computing the action $H(t)v$ is proportional to t. Therefore, the run times of computing $\Gamma(t)$ at points $h, 2\,h, \ldots, Mh$ and maximizing $\Gamma(t)$ with `TI-LU` or `TI-GMRES` can be estimated respectively as $c\,(t_{\max}/M)(M^2/2) = c\,t_{\max}M/2$ and $c\,\hat{t}_{\text{opt}}k$ with some constant c that depends on the used method. Assuming the run time of computing the action $H(t)v$ using `expv` or `expmv` weakly depends on t, we see that the run times of computing $\Gamma(t)$ and maximizing $\Gamma(t)$ with `expv` or `expmv` can be estimated respectively as cM and ck with some constant c that also depends on the used method. So, the run time of computing $\Gamma(t)$ is larger than that of maximizing $\Gamma(t)$ in about M/k times when `expv` or `expmv` is used, and in about $(M/k)t_{\max}/(2\hat{t}_{\text{opt}})$ times when `TI-LU` or `TI-GMRES` is used. This is confirmed by Table 4 which shows the ratios of the run time for computing $\Gamma(t)$ to that of maximizing $\Gamma(t)$ and the estimate M/k when `expv` or `expmv` is used and the estimate $(M/k)t_{\max}/(2\hat{t}_{\text{opt}})$ when `TI-LU` or `TI-GMRES` is used.

Note that `fminbnd` has an optional termination tolerance parameter `TolX`. We have chosen `TolX` $= 1.5\tau$ where τ is defined in Section 2.4 for all computations with this function. This choice gives a local minimum with the absolute accuracy τ if we compute the function under minimization accurately enough.

In the following subsections we consider the auxiliary function $\gamma(t, v)$ defined in (7), where v is a nonzero vector. We will study some of its properties and propose an alternating maximization method for computing humps.

4.1. Properties of $\gamma(t,v)$

The function $\gamma(t,v)$ is positive-valued and analytic with respect to t. A direct calculation of its partial derivative yields

$$\frac{\partial\gamma}{\partial t}(t,v)=\frac{1}{\gamma(t,v)}\frac{\mathrm{Re}(Av,H(t)v)}{(v,v)}. \tag{12}$$

In particular, if $t=0$ and v is an eigenvector corresponding to the largest eigenvalue $\mu(A)$ of $(A+A^*)/2$, then (12) gives

$$\frac{\partial\gamma}{\partial t}(0,v)=\frac{\mathrm{Re}\,(Av,v)}{(v,v)}=\mu(A). \tag{13}$$

As it follows immediately from (1), $\gamma(t,v)\le\Gamma(t)$ for all $t\ge0$, and $\gamma(t,v)=\Gamma(t)$ if and only if v is a right singular vector of the matrix $\exp(tA)$ corresponding to its largest singular value $\Gamma(t)$ or, equivalently, an eigenvector of the matrix $H(t)$ corresponding to its largest eigenvalue $\Gamma(t)^2$.

If $\gamma(t_*,v_*)=\Gamma(t_*)$ then from (2)–(4) we have

$$\frac{d\Gamma}{dt}(t_*-)\le\frac{\partial\gamma}{\partial t}(t_*,v_*)\le\frac{d\Gamma}{dt}(t_*+). \tag{14}$$

The function $\gamma(t,v)$ can be used to compute the maxima of $\Gamma(t)$ but for this purpose we need an additional criterion. Indeed, if $\Gamma(t)$ has a local maximum at t_* and $\gamma(t_*,v_*)=\Gamma(t_*)$, then $\gamma(t,v_*)$ clearly has a local maximum at t_*. Conversely, if $\gamma(t,v_*)$ has a local maximum at t_* and $\gamma(t_*,v_*)=\Gamma(t_*)$, we cannot conclude that $\Gamma(t)$ has a local maximum at t_*. We only have

$$\frac{d\Gamma}{dt}(t_*-)\le0\le\frac{d\Gamma}{dt}(t_*+).$$

Thus, the equality $\gamma(t_*,v_*)=\Gamma(t_*)$ and the existence of a local maximum of $\gamma(t,v_*)$ at t_* is only a necessary condition for $\Gamma(t)$ to have a local maximum at t_*. To see if it is a true local maximum, we can choose a sufficiently small positive δ and check if

$$\Gamma(t_*\pm\delta)<\Gamma(t_*). \tag{15}$$

4.2. Alternating maximization

Algorithm 2 maximizes the function $\gamma(t,v)$ alternatively with respect to $t\ge0$ and v.

Algorithm 2 Alternating maximization applied to $\gamma(t,v)$.

```
Input: $A$
Output: sequences $\{t_k\}$ and $\{v_k\}$.
  1: Compute a normalized eigenvector $v_0$ corresponding to the largest eigenvalue $\mu(A)$ of $(A+A^*)/2$ and set $k=1$.
  2: Find $t_k=\min\arg\max\gamma(t,v_{k-1})$ with respect to $t$ where $t\ge0$.
  3: Find $v_k$ by maximizing $\gamma(t_k,v)$ with respect to $v\in\mathbb{C}^n$, $\|v\|_2=1$.
  4: Set $k:=k+1$ and go to 2.
```

We discuss some important properties of the sequences $\{t_k\}$ and $\{v_k\}$ generated by this algorithm. Note first that due to (13) and our assumption on the numerical range of A,

$$\frac{\partial\gamma}{\partial t}(0,v_0)=\mu(A)>0,$$

and, since $\gamma(0,v)=1$ for all v, we have $t_1>0$ and $\gamma(t_1,v_0)>1$. Moreover, since

$$\gamma(t_k,v_{k-1})\le\gamma(t_k,v_k)\le\gamma(t_{k+1},v_k), \tag{16}$$

we have $t_k>0$ and $\gamma(t_k,v_{k-1})>1$ for all $k>1$ as well. On the other hand $t_k<t_{\min}$, where $t_{\min}$ is defined in (6), since $\gamma(t,v)\le\Gamma(t)$.

Theorem 4.1. *If $t_k\to t_*$ as $k\to\infty$ then there exists $v_*\in\mathbb{C}^n$, $\|v_*\|_2=1$ such that*

$$\gamma(t_*,v_*)=\Gamma(t_*)=\max_{t\ge0}\gamma(t,v_*). \tag{17}$$

Proof. The vector v_k which is found in the third step is a singular vector of $\exp(t_kA)$ corresponding to the largest singular value $\Gamma(t_k)$. Therefore $\gamma(t_k,v_k)=\Gamma(t_k)$. From (16) we see that the sequences $\{\gamma(t_k,v_k)\}$ and $\{\gamma(t_{k+1},v_k)\}$ are nondecreasing and since they are bounded they converge and we have

$$\lim_{k\to\infty} \gamma(t_k, v_k) = \lim_{k\to\infty} \gamma(t_{k+1}, v_k) = \lim_{k\to\infty} \Gamma(t_k) = \Gamma_* \tag{18}$$

with some Γ_*: $1 < \Gamma_* \leq \Gamma_{\text{opt}}$.

Since the sequence $\{v_k\}$ is bounded, it has at least one accumulation point v_* and there exists a subsequence $\{v_{k_l}\}$ which converges to v_* as $l \to \infty$. Letting $l \to \infty$ in

$$\gamma(t_{k_l}, v_{k_l}) = \Gamma(t_{k_l}), \qquad \gamma(t_{k_l+1}, v_{k_l}) = \max_{t\geq 0} \gamma(t, v_{k_l})$$

and using (18) we obtain (17). □

Thus, if $t_k \to t_*$ as $k \to \infty$ and the criterion (15) is satisfied then $\Gamma(t)$ has a local maximum at t_*.

To be of practical use, the operations in Algorithm 2 should be done inexactly using iterative methods. The initial vector can be obtained with a few iterations of the Lanczos algorithm applied to $(A + A^*)/2$. When the time integration method is used, step 2 has to be carried out at the time integration nodes. Otherwise we use the function `fminbnd` to find a local minimum of $-\gamma(t, v_{k-1})$ in the chosen interval of t. Algorithm 1 applied to $H(t_k)$ can be used to find the vector v_k at step 3. These modifications are reflected in Algorithm 3 which ensures that the inequalities (16) are satisfied even under approximate computations (they are consequences of the maximization over v of $\gamma(t_k, v)$ and over t of $\gamma(t, v_{k-1})$). The algorithm is stopped if $t_k = t_{k-1}$, which means that the sequence $\{t_k\}$ has converged, or when the increase of $\gamma(t_k, v_k)$ is too slow (this is similar to the third condition used in the while loop of Algorithm 1).

Algorithm 3 Practical alternating maximization applied to $\gamma(t, v)$.

```
Input: $A$, $t_{\max}$, tol and $\tau$ such that $J = t_{\max}/\tau$ is integer if TI is used.
Output: an approximate local maximum.
  • Compute an approximate normalized eigenvector $v_0$ corresponding to the largest eigenvalue $\mu(A)$ of $(A + A^*)/2$ with a
  few iterations of the Lanczos algorithm starting with a random vector.
  • $t_{-1} = -1$, $s_{-1} = t_0 = 0$, $s_0 = 1$, $k = 0$
  while $t_k \neq t_{k-1}$ and $s_k \geq (1 + \text{tol})s_{k-1}$ do
    • $k = k + 1$
    • Find $t_k = \min \arg\max \gamma(t, v_{k-1})$ for $0 \leq t \leq t_{\max}$ using fminbnd
    or, if TI is used, for $t = 0, \tau, \ldots, t_{\max}$, computing $\gamma(t, v_{k-1})$ at each of these points.
    • Compute an approximate normalized eigenvector $v_k$ corresponding to the largest eigenvalue of $H(t_k)$ by Algorithm 1
    starting with $v_{k-1}$.
    • $s_k = \gamma(t_k, v_k)$
  end while
```

Note that, from the inequality (see, e.g., [6])

$$\Gamma(\xi) \leq \exp(\xi\mu(A)), \quad \xi \geq 0,$$

we deduce that

$$\gamma(t, v) \leq \max_{t\leq s\leq t+\tau} \gamma(s, v) \leq \max_{t\leq s\leq t+\tau} \Gamma(s - t)\gamma(t, v) \leq \exp(\tau\mu(A))\gamma(t, v).$$

Thus, using the fact that $\mu(A) \approx \text{Re}(Av_0, v_0)$, we can find a local maximum of $\gamma(t, v_{k-1})$ with a given accuracy by choosing $\tau = \varepsilon/\text{Re}(Av_0, v_0)$ with a sufficiently small $\varepsilon > 0$.

Algorithm 3 finds a hump which is close to 0 but it can easily be modified to find a hump right to a given point. Particularly, if the time integration is used, then to find a hump right to a given point $t_o = J_o\tau$, where J_o is a positive integer smaller than J, one needs to set $t_0 = t_o$ instead of 0 and find each t_k by maximizing $\gamma(t, v_{k-1})$ for $t = t_o, t_o + \tau, \ldots, t_{\max}$. The initial vector v_0 can be chosen as an approximate normalized eigenvector corresponding to the largest eigenvalue of $H(t_o)$ computed with Algorithm 1. In this case the value of s_0 must be set equal to $\gamma(t_0, v_0)$.

The results of Algorithm 3 for Matrices 1, 2 and 4 and with the above modification for Matrix 3 are given in the last two columns of Tables 1 and 3. These results are consistent with those obtained by the maximization of $\Gamma(t)$.

5. Conclusions

Taking into account the accuracy and execution time, the tests clearly show that the alternating maximization procedure (Algorithm 3) exhibits superior performance. Further work is, however, needed to better understand its convergence. The execution times for this algorithm with `expv` and `expmv` are comparable for Matrix 1 but are in favor of `expmv` for Matrix 2 and of `expv` for Matrix 3. Thus, in the context of this algorithm, there is no a priori reason to favor one computation of the action of the matrix exponential over the other. They can both be used efficiently with their defaults parameters and generally perform much better than TI-LU and TI-GMRES. The methods TI-LU and TI-GMRES can be very useful in situations where a good preconditioner for GMRES can be found or when the LU decompositions of $I - \tau A$ and $1.5I - \tau A$ can be computed with small fill-in.

Acknowledgments

We thank Roger Sidje for a fruitful discussion on an earlier version of the paper. The development of numerical algorithms was supported by the Russian Science Foundation (Project No. 14-21-00025) and the analysis of numerical experiments was supported by the Russian Foundation for Basic Research (Project No. 16-01-00572).

References

[1] L.N. Trefethen, A.E. Trefethen, S.C. Reddy, T.A. Driscoll, Hydrodynamic stability without eigenvalues, Science 261 (1993) 578–584.
[2] P. Schmid, D.S. Henningson, Stability and Transition in Shear Flows, Springer–Verlag, Berlin, 2000.
[3] P. Yecko, S. Zaleski, Transient growth in two-phase mixing layers, J. Fluid Mech. 528 (2005) 43–52.
[4] X. Mao, S.J. Sherwin, Transient growth associated with continuous spectra of the Batchelor vortex, J. Fluid Mech. 697 (2012) 35–59.
[5] A. Monokrousos, E. åkervik, L. Brandt, S. Henningson, Global three-dimensional optimal disturbances in the Blasius boundary-layer flow using time-steppers, J. Fluid Mech. 650 (2010) 181–214.
[6] L.N. Trefethen, M. Embree, Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators, Princeton University Press, Princeton, 2005.
[7] J.L.M. van Dorsselaer, J.F.B.M. Kraaijevanger, M.N. Spijker, Linear stability analysis in the numerical solution of initial value problems, Acta Numer. 2 (1993) 199–237.
[8] S.K. Godunov, Ordinanry Differential Equations with Constant Coefficients, in: Translations of Mathematical Monographs, vol. 169, Amer. Math. Soc., Providence, RI, 1997.
[9] C. Moler, C. van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, SIAM Rev. 45 (2003) 3–49.
[10] C. Moler, C. van Loan, Nineteen dubious ways to compute the exponential of a matrix, SIAM Rev. 20 (1978) 801–836.
[11] I. Gohberg, P. Lancaster, L. Rodman, Matrix Polinomials, Academic Press, N.-Y, 1982.
[12] T. Kato, Perturbation Theory for Linear Operators, second ed., Springer-Verlag, Berlin, 1976.
[13] Yu.M. Nechepurenko, M. Sadkane, A low-rank approximation for computing the matrix exponential norm, SIAM J. Matrix Anal. Appl. 32 (2011) 349–363.
[14] N.J. Higham, The scaling and squaring method for the matrix exponential revisited, SIAM J. Matrix Anal. Appl. 26 (2005) 1179–1193.
[15] E. Hairer, S.P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems, second ed., Springer-Verlag, Berlin, 1993.
[16] Y. Saad, Analysis of some Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 29 (1992) 209–228.
[17] R.B. Sidje, Expokit: A software package for computing matrix exponentials, ACM Trans. Math. Software 24 (1998) 130–156.
[18] M. Hochbruck, C. Lubich, On Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 34 (1997) 1911–1925.
[19] A. Frommer, V. Simoncini, Matrix functions, in: W. Schilders, H. Van der Vorst, J. Rommes (Eds.), Model Order Reduction: Theory, Research Aspects and Applications, Mathematics in Industry, Springer-Verlag, Berlin, 2008, pp. 275–304.
[20] A.H. Al-Mohy, N.J. Higham, Computing the action of the matrix exponential, with an application to exponential integrators, SIAM J. Sci. Comput. 33 (2011) 488–511.
[21] G.E. Forsythe, M.A. Malcolm, M.A. Moler, Computer Methods for Mathematical Computations, Prentice Hall, N.-Y, 1976.
[22] G. El Khoury, Yu.N. Nechepurenko, M. Sadkane, Acceleration of inverse subspace iteration with Newtons method, J. Comput. Appl. Math. 259 (2014) 205–215.
[23] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing, Boston, 1996.
[24] B.N. Parlett, The Symmetric Eigenvalue Problem, SIAM, 1998.
[25] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., The John Hopkins University Press, Baltimore, 1996.
[26] K. Veselic, Bounds for exponentially stable semigroups, Linear Algebra Appl. 358 (2003) 309–333.
[27] Yu.M. Nechepurenko, Bounds for the matrix exponential based on the Lyapunov equation and limits of the Hausdorff set, Comput. Math. Math. Phys. 42 (2002) 125–134.