On alternating maximization algorithm for computing the hump of matrix powers

Miloud Sadkane

On alternating maximization algorithm for computing the hump of matrix powers

Miloud Sadkane [a]

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

Abstract

Alternating maximization type algorithms for computing the maximal growth of the norm of matrix powers are discussed. Their convergence properties are established under the natural assumption that the matrix is discrete-stable. The implementation considers both the small and large problem sizes, where for the latter case, a variant of the Lanczos method is especially devised. The numerical tests confirm that the main advantages of the alternating maximization technique are its accuracy and speed of convergence.

Key words: hump of the matrix power, alternating maximization, Lanczos method

1. Introduction

The behavior of powers of matrices is well analyzed in the literature; see, for example, [5, chap 18] or [12, chap 4]. For a matrix $A \in \mathbb{C}^{n\times n}$ with spectral radius $\rho(A)$, we consider the sequence of 2-norms

$$\Gamma(k) = \|A^k\|_2, \ k = 0, 1, \ldots \tag{1}$$

It is known (see [6, p.322]) that $\lim_{k\to\infty} \Gamma(k)^{1/k} = \rho(A)$. Therefore there exists a sequence $(\delta_k)_{k\geq 1}$ such that

$$\Gamma(k) = (\rho(A) + \delta_k)^k, \quad \lim_{k\to\infty} \delta_k = 0. \tag{2}$$

Assuming that A is discrete-stable (i.e., $\rho(A) < 1$), then (2) shows that as $k \to \infty$

$$\Gamma(k) \approx (\rho(A)e^{\delta_k/\rho(A)})^k. \tag{3}$$

The above considerations are valid for any matrix norm but the 2-norm is of particular interest in many applications.

Email address: miloud.sadkane@univ-brest.fr (Miloud Sadkane).

The sequence $\Gamma(k)$ can grow rapidly with k before it decays with a rate governed by the spectral radius but the growth behavior is difficult to assess. In fact, as pointed out in [12, p. 165]: "In many cases $\|A^k\|_2$ grows in floating-point arithmetic approximately at the rate $(\rho_\varepsilon(A))^k$, where ε is on the order of machine epsilon". Here, $\rho_\varepsilon(A)$ denotes the ε-pseudospectral radius defined by $\rho_\varepsilon(A) = \sup_{\lambda\in\Lambda_\varepsilon} |\lambda|$ and Λ_ε is the set of $z \in \mathbb{C}$ such that $\|(zI - A)^{-1}\|_2 > 1/\varepsilon$.

In this note we are interested in computing the maximal growth, which, in the sequel, will be referred to as the hump (of the matrix power). More precisely, the aim is to compute k_h and $\Gamma_h = \Gamma(k_h) = \|A^{k_h}\|_2$ such that

$$\Gamma_h = \max_{k\in I} \Gamma(k), \tag{4}$$

where $I = [k_{\min}, k_{\max}]$ is a given interval of nonnegative integers. The opportunity to measure such a hump arises in various situations. For example, it is of interest in the stability analysis of initial value problems [13] and stiff ordinary differential equations [4], and in the calculations of optimal disturbances and transient growth in boundary layers [1,2,7].

Since $\|A^k\|_2 \le \|A\|_2^k$, it is obvious that $k_h = k_{\min}$ if $\|A\|_2 \le 1$. Therefore, in the sequel, we consider only the case where $\|A\|_2 > 1$ along with the assumption that A is discrete-stable. In this case, the function $k \to \Gamma(k)$ has at least one hump in the interval $[\hat{k}_{\min}, \hat{k}_{\max}]$ where

$$\hat{k}_{\min} \ge 1, \quad \hat{k}_{\max} \le \min\{k \ge 2;\ \Gamma(k) < 1\}.$$

In general, the graph of $\Gamma(k)$ may have many humps (see, e.g., Figures 1-right, 2, and 3). Therefore, unless further information is available, only a local maximum is a priori guaranteed to be found.

A simple and natural approach consists of computing the powers of A through the iterative process

$$E_0 = A^{k_{\min}}, \quad E_k = AE_{k-1} \tag{5}$$

and selecting the integer k with the largest norm $\|E_k\|_2$. The iterations continue until $k = k_{\max} - k_{\min}$. This actually leads to the global maximum. However, the main drawback of this approach is its cost since, besides forming E_0 which necessitates $\mathcal{O}\left(n^3 k_{\min}\right)$ operations, each iteration requires $\mathcal{O}\left(n^3\right)$ operations. This makes the method impractical for large n. As an alternative, we note that by definition of the 2-norm of a matrix (see [3])

$$\Gamma_h = \max_{k\in I} \max_{\|v\|_2=1} \|A^k v\|_2. \tag{6}$$

As we will see, this leads to a simple algorithm which consists of maximizing $\|A^k v\|_2$ alternatively with respect to k and v. The method needs to access the matrix A only in the form of matrix-vector operations, and can therefore be applied to large sparse matrices. The process requires very few iterations to converge. Very often, the computed hump corresponds to the global maximum in the considered interval I. This method has been used successfully for the matrix exponential [8,11]. We show that many ideas in these references can be adapted here but the fact that A^k is a function in the discrete variable k requires special treatments.

This note is structured as follows. In Section 2 we discuss some mathematical properties of the alternating maximization algorithm for the case under study and illustrate numerically its performance. Its extension to the large-scale case is considered in Section

3. Numerical illustrations carried out in MATLAB are given throughout sections 2 and 3. Finally, a conclusion is drawn in Section 4.

2. Alternating maximization

Note first that Γ_h is the largest singular value of A^{k_h}. Let v_h and u_h be the unit 2-norm right and left singular vectors corresponding to Γ_h. That is

$$A^{k_h} v_h = \Gamma_h u_h, \quad (A^{k_h})^* u_h = \Gamma_h v_h. \tag{7}$$

The following two propositions are consequences of (4) and (7). Proposition 2.1 will be useful when discussing the stopping criterion (see Proposition 2.5 and the discussion that follows). Proposition 2.2 suggests that Γ_h can be obtained by maximizing $\|A^k v\|_2$ with respect to k and v alternately.

Proposition 2.1 *Assume that $k_{min} < k_h < k_{max}$. Then*

$$|v_h^* A v_h| = |u_h^* A u_h| \leq 1. \tag{8}$$

Proof Note that the equality $(A^{k_h} v_h)^* A (A^{k_h} v_h) = ((A^{k_h})^* A^{k_h} v_h)^* A v_h$ together with (7) shows that the equality $v_h^* A v_h = u_h^* A u_h$ is always true.
The first equality in (7) gives $\Gamma_h \|A u_h\|_2 = \|A^{1+k_h} v_h\|_2$. Since $k_h + 1 \in I$, we deduce from (4) that $\|A u_h\|_2 \leq 1$. Hence $|u_h^* A u_h| \leq \|A u_h\|_2 \leq 1$. □ □

Proposition 2.2 *We have*

$$\Gamma_h = \max_{k \in I} \max_{\|v\|_2 = 1} \|A^k v\|_2 = \max_{\|v\|_2 = 1} \max_{k \in I} \|A^k v\|_2 \tag{9}$$

$$= \max_{k \in I} \|A^k v_h\|_2 = \max_{\|v\|_2 = 1} \|A^{k_h} v\|_2. \tag{10}$$

Proof The first equality has already been used (see (6)). For the second one, let $k \in I$ and $v \in \mathbb{C}^n$ with $\|v\|_2 = 1$. Then

$$\|A^k v\|_2 \leq \max_{k \in I} \|A^k v\|_2 \leq \max_{\|v\|_2 = 1} (\max_{k \in I} \|A^k v\|_2).$$

Since this is true for arbitrary v with $\|v\|_2 = 1$, we have

$$\max_{\|v\|_2 = 1} \|A^k v\|_2 \leq \max_{\|v\|_2 = 1} (\max_{k \in I} \|A^k v\|_2).$$

Since this is true for arbitrary $k \in I$, we also have

$$\max_{k \in I} (\max_{\|v\|_2 = 1} \|A^k v\|_2) \leq \max_{\|v\|_2 = 1} (\max_{k \in I} \|A^k v\|_2).$$

Reversing the role of k and v in the above reasoning leads to (9). The equalities (10) follow simply from

$$\Gamma_h = \|A^{k_h} v_h\|_2 \leq \max_{k \in I} \|A^k v_h\|_2 \leq \max_{k \in I} \|A^k\|_2 = \Gamma_h,$$

$$\Gamma_h = \|A^{k_h} v_h\|_2 \leq \max_{\|v\|_2 = 1} \|A^{k_h} v\|_2 \leq \|A^{k_h}\|_2 = \Gamma_h. \qquad \square$$

□

Proposition 2.2 is actually the core foundation of the alternating maximization approach for computing k_h and Γ_h. A formal description is given in Algorithm 1.

Algorithm 1 Alternating maximization

Input: A, $I = [k_{\min}, k_{\max}]$, $k_0 \in I$.
Output: sequences $\{k_p\}$ and $\{v_p\}$.
1: **for** $p = 1, 2, \ldots$ **do**
2: Find $v_p : \|A^{k_{p-1}} v_p\|_2 = \max_{\|v\|_2=1} \|A^{k_{p-1}} v\|_2 = \Gamma(k_{p-1})$.
3: Find $k_p : \|A^{k_p} v_p\|_2 = \max_{k \in I} \|A^k v_p\|_2$.
4: **end for**

Starting with $k_0 \in I$, Algorithm 1 finds v_1 which maximizes $\|A^{k_0} v\|_2$ with respect to v and then finds k_1 which maximizes $\|A^k v_1\|_2$ with respect to k, and alternates steps 2 and 3. The hope is that the sequence (k_p, v_p) converges quickly to (k_h, v_h). If the user wants to start with $v_0 \in \mathbb{C}^n$, $\|v_0\|_2 = 1$, then steps 2 and 3 should be permuted, for example, as follows.

2: Find $k_p : \|A^{k_p} v_{p-1}\|_2 = \max_{k \in I} \|A^k v_{p-1}\|_2$.
3: Find $v_p : \|A^{k_p} v_p\|_2 = \max_{\|v\|_2=1} \|A^{k_p} v\|_2 = \Gamma(k_p)$.

No matter how the algorithm starts, the tests show that the convergence occurs within very few iterations, and it is the global maximum that is generally found, see Subsections 2.1 and 3.1.
The main convergence properties are given in the following two propositions.

Proposition 2.3 *The sequences* $(\Gamma(k_p))_{p \geq 1}$ *and* $\left(\|A^{k_p} v_p\|_2\right)_{p \geq 1}$ *constructed by Algorithm 1 are monotone nondecreasing, bounded above by* Γ_h*, and converge to the same limit.*

Proof The sequences are clearly bounded above by Γ_h. From steps 2 and 3 of Algorithm 1 we have

$$\Gamma(k_p) = \|A^{k_p} v_{p+1}\|_2 \geq \|A^{k_p} v_p\|_2 \geq \|A^{k_{p-1}} v_p\|_2 = \Gamma(k_{p-1}),$$

from which the proof follows. □ □

The following simple lemma will be used in the proof of Proposition 2.4.

Lemma 2.1 *For nonnegative integers* $q_1, q_2 \in I$ *and unit 2-norm vectors* w_1, w_2 *we have*

$$|\|A^{q_1} w_1\|_2 - \|A^{q_2} w_2\|_2| \leq C(|q_1 - q_2| + \|w_1 - w_2\|_2),$$

where $C = \Gamma_h \max(\|A - I\|_2, 1)$.

Proof Assume that $q_1 > q_2$ (the proof is similar if $q_1 \leq q_2$). Then

$$\begin{aligned} |\|A^{q_1} w_1\|_2 - \|A^{q_2} w_2\|_2| &\leq \|A^{q_1} w_1 - A^{q_2} w_2\|_2 \\ &= \|(A^{q_1} - A^{q_2}) w_1 + A^{q_2}(w_1 - w_2)\|_2 \\ &\leq \|A^{q_1} - A^{q_2}\|_2 + \Gamma_h \|w_1 - w_2\|_2 \end{aligned}$$

and

$$\begin{aligned} \|A^{q_1} - A^{q_2}\|_2 &= \|\Sigma_{i=q_2}^{q_1 - 1}(A^{i+1} - A^i)\|_2 \\ &\leq \Sigma_{i=q_2}^{q_1 - 1} \|A^i\|_2 \|I - A\|_2 \leq (q_1 - q_2)\Gamma_h\|_2 \|I - A\|_2. \qquad \square \end{aligned}$$

□

Remark 2.1 *Lemma 2.1 shows that the function* $(q, w) \to \|A^q w\|_2$ *is uniformly continuous on* $I \times \{w \in \mathbb{R}^n : \|w\|_2 = 1\}$.

Proposition 2.4 *Let* (k', v') *be an accumulation point of the sequence* $\{(k_p, v_p)\}_{p\geq 1}$. *Then*

$$\Gamma(k') = \|A^{k'} v'\|_2 = \max_{k\in I} \|A^k v'\|_2 = \max_{\|v\|_2=1} \|A^{k'} v\|_2. \tag{11}$$

Proof Since the sequences $\{k_p\}_{p\geq 1}$ and $\{v_p\}_{p\geq 1}$ are bounded, they admit convergent subsequences

$$\lim_{p'\to\infty} k_{p'} = k', \quad \lim_{p'\to\infty} v_{p'} = v'.$$

Since $\|A^{k_{p'}} v_{p'+1}\|_2 = \Gamma(k_{p'})$ and the subsequences $\{v_{p'}\}$ and $\{k_{p'}\}$ are convergent, Proposition 2.3 and the continuity of the function $(k, v) \to \|A^k v\|_2$, $k \in I$, $\|v\|_2 = 1$ (see Remark 2.1) show that

$$\|A^{k'} v'\|_2 = \lim_{p'\to\infty} \|A^{k_{p'}} v_{p'}\|_2 = \lim_{p'\to\infty} \Gamma(k_{p'+1}) = \lim_{p'\to\infty} \Gamma(k_{p'}) = \Gamma(k'). \tag{12}$$

For $k \in I$, we have by the triangle inequality and Lemma 2.1

$$\begin{aligned} \|A^k v'\|_2 &\leq |\|A^k v'\|_2 - \|A^k v_{p'}\|_2| + \|A^k v_{p'}\|_2 \\ &\leq C\|v' - v_{p'}\|_2 + \|A^k v_{p'}\|_2. \end{aligned} \tag{13}$$

Step 3 of Algorithm 1 ensures that $\|A^k v_{p'}\|_2 \leq \|A^{k_{p'}} v_{p'}\|_2$. Hence by the triangle inequality and Lemma 2.1

$$\begin{aligned} \|A^{k_{p'}} v_{p'}\|_2 &\leq |\|A^{k_{p'}} v_{p'}\|_2 - \|A^{k'} v'\|_2| + \|A^{k'} v'\|_2 \\ &\leq C(|k_{p'} - k'| + \|v_{p'} - v'\|_2) + \|A^{k'} v'\|_2. \end{aligned} \tag{14}$$

Now from (13) and (14) we obtain

$$\|A^k v'\|_2 \leq C(2\|v_{p'} - v'\|_2 + |k_{p'} - k'|) + \|A^{k'} v'\|_2. \tag{15}$$

It follows from (15) by letting $p' \to \infty$ that $\|A^k v'\|_2$ has a maximum at $\|A^{k'} v'\|_2$.

We proceed similarly to show that $\|A^{k'} v'\|_2 = \max_{\|v\|_2=1} \|A^{k'} v\|_2$. Let v be a unit 2-norm vector. Then by the triangle inequality and Lemma 2.1

$$\begin{aligned} \|A^{k'} v\|_2 &\leq |\|A^{k'} v\|_2 - \|A^{k_{p'}} v\|_2| + \|A^{k_{p'}} v\|_2 \\ &\leq C|k' - k_{p'}| + \|A^{k_{p'}} v\|_2. \end{aligned} \tag{16}$$

Step 2 of Algorithm 1 ensures that $\|A^{k_{p'}} v\|_2 \leq \|A^{k_{p'}} v_{p'+1}\|_2$ and hence, again by the triangle inequality and Lemma 2.1

$$\begin{aligned} \|A^{k_{p'}} v_{p'+1}\|_2 &\leq |\|A^{k_{p'}} v_{p'+1}\|_2 - \|A^{k'} v'\|_2| + \|A^{k'} v'\|_2 \\ &\leq C(|k_{p'} - k'| + \|v_{p'+1} - v_{p'}\|_2 + \|A^{k'} v'\|_2. \end{aligned} \tag{17}$$

Now from (16) and (17) we obtain

$$\|A^{k'} v\|_2 \leq C(2|k' - k_{p'}| + \|v_{p'+1} - v_{p'}\|_2) + \|A^{k'} v'\|_2,$$

and the proof follows by letting $p' \to \infty$. □ □

A few comments are in order. From (10) and (11) we conclude that, in theory, Algorithm 1 will converge to a hump or, in the worst case, stagnate at accumulation points where $\|A^{k'}v'\|_2$ achieves the maximum with respect to k and v without being a maximum of $\|A^k v\|_2$. However, in the many tests we have done, the latter case has never occurred.
It can be shown (as in Proposition 2.1) that $|(v')^* A v'| \leq 1$. This characterization suggests that Algorithm 1 may be stopped when $|v_p^* A v_p| \leq 1$. However, the condition $k_p = k_{p-1}$ has proved to be more useful in our tests. It means that the sequence $\Gamma(k_p)$ ceases to increase at step p. Besides, it is available without cost and implies that $|v_p^* A v_p| \leq 1$. A clarification is given in the following proposition (note the similarity with Proposition 2.1).

Proposition 2.5 *If at iteration p of Algorithm 1, $k_{p-1} = k_p \in (k_{min}, k_{max})$, then $|v_p^* A v_p| = |u_p^* A u_p| \leq 1$, where u_p is the unit 2-norm left singular vector corresponding to the largest singular value $\Gamma(k_p)$.*

Proof Note first that $A^{k_{p-1}} v_p = \Gamma(k_{p-1}) u_p$ and, as in the proof of Proposition 2.1, $v_p^* A v_p = u_p^* A u_p$.
Proposition 2.3 and steps 2 and 3 of Algorithm 1 ensure that

$$\begin{aligned}
\Gamma(k_{p-1}) &= \|A^{k_{p-1}} v_p\|_2 \\
&= \|A^{k_p} v_p\|_2 \quad (\text{since } k_{p-1} = k_p) \\
&\geq \|A^{k_{p-1}+1} v_p\|_2 \quad (\text{since } k_{p-1} + 1 \in I) \\
&= \Gamma(k_{p-1}) \, \|A u_p\|_2.
\end{aligned}$$

Hence $\|A u_p\|_2 \leq 1$ and therefore $|u_p^* A u_p| \leq 1$. □ □

Algorithm 2 summarizes the discussion. In our experiments, we start with

$$k_0 = \left\lfloor \sqrt{\cos^2(\theta) k_{\min}^2 + \sin^2(\theta) k_{\max}^2} \right\rfloor, \tag{18}$$

where $\lfloor \ \rfloor$ denotes the integer part and θ is chosen randomly in the interval $(0, 2\pi)$. By varying θ, formula (18) provides several starting points that can be used to assess the reliability of the method.
The vector v_p in step 2 is the right singular vector of $A^{k_{p-1}}$ corresponding to the largest singular value $\sigma_{\max}(A^{k_{p-1}}) = \Gamma(k_{p-1})$ which we compute using the svd function provided by MATLAB. This step requires $\mathcal{O}\left(n^3 k_{p-1}\right)$ operations. In principle, step 3 can be carried out using any appropriate optimization procedure. However, our preferred approach consists of computing $\|A^k v_p\|_2$ for $k_{\min} \leq k \leq k_{\max}$ and then finding the integer k_p which realizes the maximum. Doing so provides the expected value of k_p and, compared to standard optimization procedures, this approach has the least number of matrix-vector products. Using the fact that $\|A^k v_p\|_2 = \|A(A^{k-1} v_p)\|_2$, this step requires $\mathcal{O}\left(n^2 k_{\max}\right)$ operations. In step 4, the algorithm stops when the number of iterations exceeds a given upper bound $p_{\max}$ or when $k_p = k_{p-1}$. In our experiments we set $p_{\max} = 10$ but this value has never been attained. Thus, the computational cost of Algorithm 2 is essentially that of steps 2 and 3, which is suitable for small size matrices.

Algorithm 2 Alternating maximization with stopping criterion

Input: A, $I = [k_{\min}, k_{\max}]$, $k_0 \in I$, $p_{\max}$.
Output: sequences $\{k_p\}$ and $\{v_p\} \in \mathbb{C}^n$.

1: **for** $p = 1, \ldots, p_{\max}$ **do**
2: Find $v_p : \|A^{k_{p-1}} v_p\|_2 = \max_{\|v\|_2=1} \|A^{k_{p-1}} v\|_2 = \Gamma(k_{p-1})$.
3: Find $k_p : \|A^{k_p} v_p\|_2 = \max_{k \in I} \|A^k v_p\|_2$.
4: Stop if $p > p_{\max}$ or $k_p = k_{p-1}$.
5: **end for**

2.1. *Numerical Examples*

We illustrate the performance of Algorithm 2 with the following two examples. In the first example, the graph of $\Gamma(k)$ has one hump. In the second one, it has many humps, two of which are almost identical.

Example 1: The matrix A is upper triangular with $1s$ on the strict upper part and $A_{kk} = 1/(k+1),\ k = 1, \ldots, n$. Figure 1 (left) shows the graph of the function $k \to \Gamma(k)$, $0 \le k \le 100$.
Algorithm 2 takes three iterations (including the initialization step) to converge. Table 1 displays the values of $\Gamma(k_{p-1})$ and k_p (as computed in steps 2 and 3 of the algorithm) for different values of the interval $[k_{\min}, k_{\max}]$. Note that the condition $p > p_{\max}$ has never been satisfied.

Table 1
Results of Algorithm 1 (Example 1)

	$k_{\min} = 20,\ k_{\max} = 40$		$k_{\min} = 40,\ k_{\max} = 60$		$k_{\min} = 60,\ k_{\max} = 80$	
p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$	k_p
0	-	30	-	50	-	70
1	$2.2431 \cdot 10^{25}$	40	$3.3398 \cdot 10^{29}$	54	$8.1813 \cdot 10^{27}$	60
2	$1.7945 \cdot 10^{28}$	40	$4.1603 \cdot 10^{29}$	54	$2.2507 \cdot 10^{29}$	60

Example 2: In this example the random matrices have mean zero and standard deviation one. Let A_{11} and A_{12} be random matrices of size $m \times m$ and $m \times p$ respectively. Let $\rho(A_{11})$ be the spectral radius of A_{11} and $r = 1/(\rho(A_{11}) + 0.01)$. Let A_{22} be a Jordan block of size $p \times p$ with eigenvalue r. Then we consider the block upper triangular matrix

$$A = \begin{pmatrix} rA_{11} & npA_{12} \\ & A_{22} \end{pmatrix}.$$

It is known that $\rho(A_{11}) \approx \sqrt{m}$ for relatively large m (see [12, chap. VII]) and so $r \approx 1/(\sqrt{m} + 0.01)$.
Taking $m = 100$ and $p = 20$, we obtain $n = 120$, $\rho(A_{11}) = 10.63$, r = 0.094. Figure 1 (right) shows the graph of the function $k \to \Gamma(k)$, $0 \le k \le 100$.
Tables 2 displays, for different intervals $[k_{\min}, k_{\max}]$, the values of the sequences $\{k_p\}$ and $\{\Gamma(k_{p-1})\}$. The convergence occurs within 4 iterations (including the initialization step).

Note that the last values correspond well to the global maxima in the considered interval.

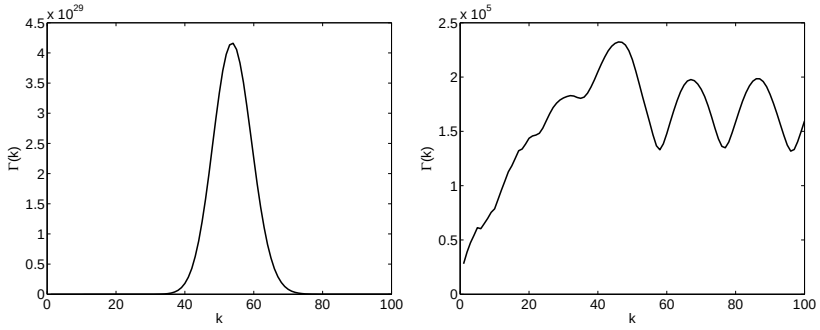

Fig. 1. $\Gamma(k)$, $1 \leq k \leq 100$, for Example 1 (left) and Example 2 (right)

Table 2
Results of Algorithm 2 (Example 2)

	$k_{\min}=1$, $k_{\max}=100$		$k_{\min}=60$, $k_{\max}=100$		$k_{\min}=60$, $k_{\max}=80$	
p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$	k_p
0	-	50	-	80	-	70
1	$2.1608 \cdot 10^5$	47	$1.5868 \cdot 10^5$	85	$1.8878 \cdot 10^5$	68
2	$2.3201 \cdot 10^5$	46	$1.9623 \cdot 10^5$	86	$1.9667 \cdot 10^5$	67
3	$2.3235 \cdot 10^5$	46	$1.9846 \cdot 10^5$	86	$1.9768 \cdot 10^5$	67

3. The case when A is large and sparse

When A is large but sparse so that matrix-vector products are cheap to perform, then Algorithm 2 can be applied with certain modifications. This is the potential advantage of the alternating maximization algorithm.
The vector v_p in step 2 of Algorithm 2 can be computed as the eigenvector of the positive semidefinite matrix

$$\mathcal{A} = (A^{k_{p-1}})^* A^{k_{p-1}} \tag{19}$$

corresponding to its largest eigenvalue $\lambda_{\max}(\mathcal{A}) = \sigma^2_{\max}(A^{k_{p-1}}) = (\Gamma(k_{p-1}))^2$. This can be achieved through the Lanczos process which we briefly include for completeness.

Algorithm 3 Lanczos process for computing v_p and $\Gamma(k_{p-1})$

Input: k_{p-1}, A, $l_{\max}$, ε, unit 2-norm vector q_1.
Output: approximations of v_p and $\Gamma(k_{p-1})$.
1: $l = 0$, $q_0 = 0$, $\beta_0 = 1$, $v = q_1$, $\sigma_{-1} = \sigma_0 = 0$
2: **while** $l \leq l_{\max}$ and $\beta_l > 0$ and $\sigma_l \geq (1+\varepsilon)\sigma_{l-1}$ **do**
3: $q_{l+1} = v/\beta_l$
4: $l = l + 1$
5: $w = \mathcal{A}q_l$
6: $\alpha_l = q_l^* w$, $v = w - \alpha_l q_l - \beta_{l-1} q_{l-1}$
7: $\beta_l = \|v\|_2$
8: $\sigma_l = \sqrt{\lambda_{\max}(T_l)}$ where T_l is given by (20)
9: **end while**
10: Compute the eigenvector y_l associated with the largest eigenvalue of T_l
11: $v = Q_l y_l$ where $Q_l = [q_1, \ldots, q_l]$

Starting with a unit 2-norm vector $q_1 \in \mathbb{C}^n$, Algorithm 3 generates via steps 3 - 7 an $n \times l$ matrix $Q_l = [q_1, q_2, \ldots, q_l]$ whose columns span an orthonormal basis of the Krylov subspace $K_l(\mathcal{A}, q_1) = \text{span}\{\mathrm{q}_1, \mathcal{A}\mathrm{q}_1, \ldots, \mathcal{A}^{\mathrm{l}-1}\mathrm{q}_1\}$ and a real symmetric, tridiagonal, positive semidefinite matrix

$$T_l = Q_l^* \mathcal{A} Q_l = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & \alpha_{l-1} & \beta_{l-1} \\ & & & \beta_{l-1} & \alpha_l \end{pmatrix} \tag{20}$$

such that

$$\mathcal{A}Q_j = Q_j T_j + \beta_j q_{j+1} e_j^*, \; j = 1, \ldots, l, \tag{21}$$

where e_j denotes the j-th column of the identity matrix of order j.

The starting vector q_1 can be chosen randomly, but in our context (computation of v_p) we have found it useful to start with $q_1 = v_{p-1}$.
The sequence of the largest eigenvalue of T_l is nondecreasing and bounded above by the last eigenvalue of $\mathcal{A}$ (see [9, Theorem 10.1.1]), that is,

$$\sigma_{l-1} \leq \sigma_l \leq \Gamma(k_{p-1}), \tag{22}$$

where $\sigma_j = \sqrt{\sigma_{\max}(T_j)}$.
In practice, the inequalities in (22) get close to equalities after only a few iterations.
The while loop terminates when at least one of the following conditions is satisfied: $l = l_{\max} + 1$ where $l_{\max} \ll n$; $\beta_l = 0$; $\sigma_l < (1+\varepsilon)\sigma_{l-1}$, where $\varepsilon > 0$ is a small given threshold. The first condition aims to minimize the storage requirements and computational costs while the second one (which is unlikely to occur in practice) means that the subspace $\text{span}\{\mathrm{q}_1, \ldots, \mathrm{q}_\mathrm{l}\}$ is A-invariant and therefore that the eigenvalues of T_l are eigenvalues of A, see (21). The third condition means that the sequence $\{\sigma_l\}$ practically ceases to increase and that σ_{l-1} approximates $\Gamma(k_{p-1})$. It is this latter condition that is most frequently encountered in our experiments. The following proposition justifies its use.

Proposition 3.1 *If $\sigma_l < (1+\varepsilon)\sigma_{l-1}$, then*

$$\|\mathcal{A}x_{l-1} - \sigma_{l-1}^2 x_{l-1}\|_2 \leq \sigma_{l-1}\sqrt{\sigma_l^2 - \alpha_l}\sqrt{2\varepsilon+\varepsilon^2},$$

where x_{l-1} is the Ritz vector given by $x_{l-1} = Q_{l-1}y_{l-1}$ and y_{l-1} is the unit 2-norm eigenvector of T_{l-1} corresponding to the eigenvalue σ_{l-1}^2.

Proof Since σ_l^2 is the largest eigenvalue of T_l, it satisfies the maximization property (see [9, Theorem 10.2.1]) $\sigma_l^2 = \max_{u\neq 0} \frac{u^* T_l u}{\|u\|_2^2}$. Therefore

$$\sigma_l^2 = \max_{u\neq 0} \frac{(Q_l u)^* \mathcal{A} Q_l u}{\|Q_l u\|_2^2} = \max_{v\in \mathrm{span}\{Q_l\}} \frac{v^*\mathcal{A}v}{\|v\|_2^2}.$$

Choosing $v = x_{l-1} + \xi q_l$ where $\xi \in \mathbb{R}$ is arbitrary, we obtain

$$\sigma_l^2 \geq \frac{\sigma_{l-1}^2 + 2\xi q_l^* \mathcal{A} x_{l-1} + \xi^2 \alpha_l}{1+\xi^2}.$$

That is, for all $\xi \in \mathbb{R}$,

$$(\sigma_l^2 - \alpha_l)\xi^2 - 2q_l^*\mathcal{A}x_{l-1}\,\xi + \sigma_l^2 - \sigma_{l-1}^2 \geq 0.$$

Hence

$$(q_l^*\mathcal{A}x_{l-1})^2 - (\sigma_l^2 - \alpha_l)(\sigma_l^2 - \sigma_{l-1}^2) \leq 0.$$

The proof follows by noting that $\sigma_l^2 - \sigma_{l-1}^2 < (2\varepsilon+\varepsilon^2)\sigma_{l-1}^2$ and that

$$\begin{aligned}
\mathcal{A}x_{l-1} - \sigma_{l-1}^2 x_{l-1} &= (\mathcal{A}Q_{l-1} - Q_{l-1}T_{l-1})y_{l-1} \\
&= \beta_{l-1} q_l e_{l-1}^* y_{l-1} \quad \text{(using (21))} \\
q_l^*\mathcal{A}x_{l-1} &= q_l^*(\mathcal{A}x_{l-1} - \sigma_{l-1}^2 x_{l-1}) \quad \text{(since } q_l^* x_{l-1} = 0\text{)} \\
&= \beta_{l-1} e_{l-1}^* y_{l-1}.
\end{aligned}$$

□ □

Remark 3.1 *Proposition 3.1 shows that the condition $\sigma_l < (1+\varepsilon)\sigma_{l-1}$ implies that the pair (σ_{l-1}, x_{l-1}) may be taken as an approximation of $(\Gamma(k_{p-1}), v_p)$.*

The last computed value of σ_l at step 8 provides the desired approximation of $\Gamma(k_{p-1})$ while v_p is approximated by the Ritz vector v computed at step 11. More details on the convergence theory of the Lanczos process can be found in [9,10].

At each step of Algorithm 3, the main operation is a matrix-vector product of the form $w = (A^{k_{p-1}})^* A^{k_{p-1}} q$ for a given vector q (see step 5), which is performed as

$$\tilde{w} = A(A(\ldots(Aq)\ldots)), \quad w = A^*(A^*(\ldots(A^*\tilde{w})\ldots)). \tag{23}$$

Each of these $2k_{p-1}$ matrix-vector products exploit the structures of A and A^*.

Step 3 of Algorithm 2 requires essentially matrix-vector products with A. This can be done by computing

$$x_0 = A^{k_{\min}} v_p, \quad x_k = Ax_{k-1} \tag{24}$$

and then selecting the integer k_p which realizes

$$\max_{0\leq k\leq k_{\max}-k_{\min}} \|x_k\|_2. \tag{25}$$

The operation $x_0 = A^{k_{\min}} v_p = A(A(\ldots(Av_p)\ldots))$ is performed as in (23).

Algorithm 4 is identical to Algorithm 3 but now steps 2 and 3 are suited for large sparse matrices.

Algorithm 4 Alternating maximization in the large sparse case

Input: A, $k_{\min}$, $k_{\max}$, k_0, $l_{\max}$, ε, $p_{\max}$, unit 2-norm vector v_0
Output: sequences $\{k_p\}$ and $\{v_p\} \in \mathbb{C}^n$.

1: **for** $p := 1, \ldots, p_{\max}$ **do**
2: Starting Algorithm 3 with $q_1 = v_{p-1}$, compute v_p and $\Gamma(k_{p-1})$ such that $\|A^{k_{p-1}} v_p\|_2 = \Gamma(k_{p-1})$.
3: Using (24)-(25), compute k_p such that $\|A^{k_p} v_p\|_2 = \max_{k \in I} \|A^k v_p\|_2$.
4: Stop if $p > p_{\max}$ or $k_p = k_{p-1}$
5: **end for**

Step 2 of Algorithm 4 requires $2k_{r-1}l_r$ matrix-vector products using l_r Lanczos iterations while step 3 requires $k_{\max}$ matrix-vector products. Therefore, p iterations require a total of

$$(2\Sigma_{r=1}^{p} k_{r-1} l_r) + k_{\max} p \tag{26}$$

matrix-vector products. Since $k_{r-1} \leq k_{\max}$, $l_r \leq l_{\max}$ and $p \leq p_{\max}$, an upper bound for the required matrix-vector products is $k_{\max} p_{\max}(1 + 2l_{\max})$.

3.1. *Numerical Examples*

To illustrate the behavior of Algorithm 4 we consider the four matrices described in Table 3. For each matrix the table indicates the spectral radius $\rho(A)$, the order n, the number of nonzero entries nnz, and a short description of the origin. More details can be found at http : //math.nist.gov/MatrixMarket/.

Table 3
Characteristics of test matrices

A	$\rho(A)$	n	nnz	Description
BP1000	15.4409	822	4661	optimization
NNC1374	$7.7980 \cdot 10^2$	1374	8606	Nuclear reactor models
PDE2961	9.9194	2961	14585	Partial Differential Equation
CRY10000	$4.2158 \cdot 10^4$	10^4	49699	Crystal Growth Simulation

In the experiments we use the following scaling

$$A/(\rho(A) + \eta), \ \eta = 0.01 \tag{27}$$

to make the matrices discrete-stable and still refer to them by their original name. Figures 2 – 5 show the norms $\Gamma(k)$. A zoom on the interval of interest is shown on the right.

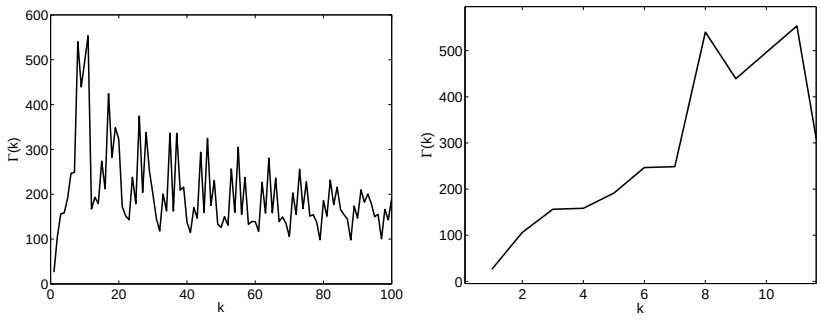

Fig. 2. $\Gamma(k)$ for BP1000, $k \in [1, 100]$ (left), $k \in [1, 11]$ (right)

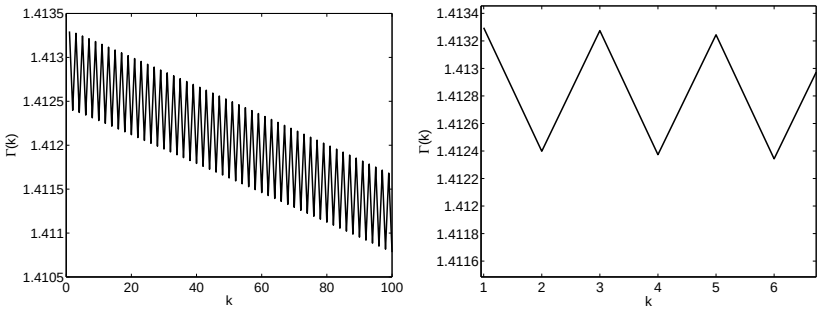

Fig. 3. $\Gamma(k)$ for NNC1374, $k \in [1, 100]$ (left), $k \in [1, 7]$ (right)

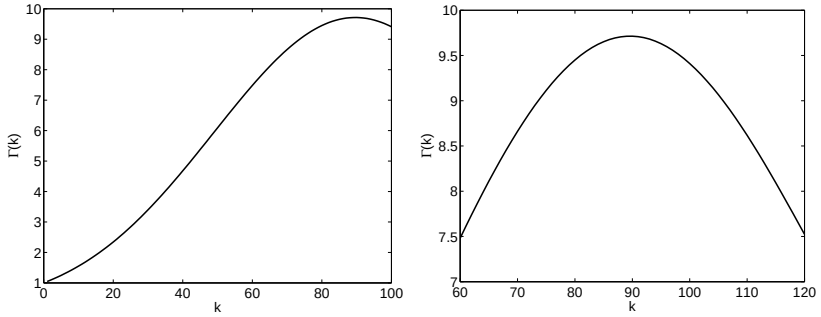

Fig. 4. $\Gamma(k)$ for PDE2961, $k \in [1, 100]$ (left), $k \in [1, 120]$ (right)

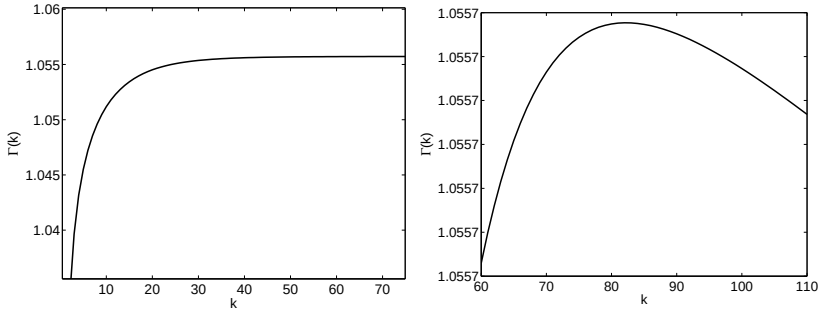

Fig. 5. $\Gamma(k)$ for CRY10000, $k \in [1, 80]$ (left), $k \in [1, 110]$ (right)

We apply Algorithm 4 with the parameters $l_{\max} = 10$, $\varepsilon = 10^{-14}$ (for Algorithm 3) and $p_{\max} = 10$. The spectral radius $\rho(A)$ is computed in MATLAB using the eigs function. Tables 4, 5, 6 and 7 summarize the convergence behavior of Algorithm 4 on the four test matrices using different intervals $[k_{\min}, k_{\max}]$. At each iteration $p \geq 1$, the tables show the values of $\Gamma(k_{p-1})$ and k_p as computed in Steps 2 and 3 of Algorithm 4. The total number of matrix-vector products required for convergence is also indicated. Note that in all tests, the iterations terminate due to the condition $k_p = k_{p-1}$. In other words, the condition $p > p_{\max}$ (see step 4 of Algorithm 4) was never used.
From Figures 2 - 5, we see that the last values of $\Gamma(k_{p-1})$ and k_p correspond to global maxima in the interval $[k_{\min}, k_{\max}]$.

Table 4
Results of Algorithm 4 (BP1000)

	$k_{\min} = 1$, $k_{\max} = 100$		$k_{\min} = 1$, $k_{\max} = 10$		$k_{\min} = 6$, $k_{\max} = 9$	
	mat-vec: 846		mat-vec: 132		mat-vec: 211	
p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$	k_p
0	-	16	-	8	-	6
1	$2.1208 \cdot 10^2$	9	$5.4018 \cdot 10^2$	8	$2.4665 \cdot 10^2$	8
2	$4.3933 \cdot 10^2$	11			$5.4018 \cdot 10^2$	8
3	$5.5352 \cdot 10^2$	11				

Table 5
Results of Algorithm 4 (NNC1374)

	$k_{\min} = 1$, $k_{\max} = 100$		$k_{\min} = 2$, $k_{\max} = 6$		$k_{\min} = 4$, $k_{\max} = 7$	
	mat-vec: 946		mat-vec: 198		mat-vec: 164	
p	$\Gamma(k_p)$	k_p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$	k_p
0	-	39	-	5	-	5
1	1.4127	1	1.4132	3	1.4132	5
2	1.4133	1	1.4133	3		

Table 6
Results of Algorithm 4 (PDE2961)

	$k_{\min}=1,\ k_{\max}=100$		$k_{\min}=1,\ k_{\max}=60$		$k_{\min}=60,\ k_{\max}=90$	
	mat-vec: 3168		mat-vec: 1344		mat-vec: 2240	
p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$	k_p
0	-	96	-	37	-	89
1	9.5975	91	4.2759	60	9.7124	90
2	9.7080	90	7.4819	60	9.7131	90
3	9.7131	90				

Table 7
Results of Algorithm 4 (CRY10000)

	$k_{\min}=1,\ k_{\max}=100$		$k_{\min}=1,\ k_{\max}=60$		$k_{\min}=60,\ k_{\max}=90$	
	mat-vec: 1440		mat-vec: 920		mat-vec: 1450	
p	k_p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$	k_p	$\Gamma(k_{p-1})$
0	40	-	14	-	60	-
1	82	1.055609	60	1.053107	82	1.055712
2	82	1.055723	60	1.055712	82	1.055723

4. Conclusion

This work has shown that alternating maximization type algorithms can be used to efficiently compute the size of the hump $\max_{k\in I}\|A^k\|_2$ of a discrete-stable matrix A in a given interval I. We presented algorithms suitable for both small and large problem sizes, though it is this latter case that actually makes the approach attractive. The main advantages are the flexibility (the algorithm can start with k_0 or with v_0), the simplicity (the algorithm requires few vector updates and the matrix A is accessed only through matrix-vector multiplications), and the ability to efficiently terminate at no cost. The theoretical analysis partly explains the convergence behavior of the method, but still does not explain its good numerical performance. This suggests the need for additional convergence theory that takes into account the discrete-time nature of the problem.

Acknowledgements. The author is grateful to the reviewers for their constructive comments and suggestions.

References

[1] P. Andersson, M. Berggren, and D. S. Henningson, Optimal disturbances and bypass transition in boundary layers, Physics of Fluids, 11(1999), 134–150.
[2] P. Corbett, A. Bottaro, Optimal perturbations for boundary layers subject to stream-wise pressure gradient, Physics of Fluids, 12 (2000), 120–130.
[3] G.H. Golub and C.F. Van Loan, Matrix Computations, 3rd ed., The John Hopkins University Press, Baltimore, 1996.
[4] D.J. Higham and L.N. Trefethen, Stiffness of ODEs, BIT, 33 (1993), 285–303.

[5] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia,1996.
[6] R.A. Horn and C.R. Johnson, Matrix Analysis, Cambridge U.P., 1985.
[7] P. Luchini, Reynolds-number-independent instability of the boundary layer over a flat surface: optimal perturbations, J. Fluid Mech. 404 (2000), pp. 289–309.
[8] Y. Nechepurenko and M. Sadkane, Computing humps of the matrix exponential, J. Comput. Appl. Math., 319 (2017), 87-96.
[9] B.N. Parlett, The Symmetric Eigenvalue Problem, SIAM, 1998.
[10] Y. Saad, Numerical methods for large eigenvalue problems, Manchester University Press, Manchester, 1992.
[11] M. Sadkane and R.B. Sidje, An alternating maximization method for approximating the hump of the matrix exponential, Bit Numer Math 57 (2017), pp. 609–628.
[12] L.N. Trefethen, M. Embree, Spectra and pseudospectra: The Behavior of Nonnormal Matrices and Operators, Princeton University Press, Princeton, 2005.
[13] J.L.M. van Dorsselaer, J.F.B.M. Kraaijevanger and M.N. Spijker, Linear stability analysis in the numerical solution of initial value problems, Acta Numerica, 2 (1993), 199–237.