SIAM J. SCI. COMPUT.
Vol. 40, No. 5, pp. A3344–A3370

ROBUST LINEAR STABILITY ANALYSIS AND A NEW METHOD FOR COMPUTING THE ACTION OF THE MATRIX EXPONENTIAL[*]

MINGHAO W. ROSTAMI[†] AND FEI XUE[‡]

Abstract. Linear stability analysis of a large-scale dynamical system requires computing the rightmost eigenvalue of a large sparse matrix A. To enhance the convergence to this eigenvalue, an iterative eigensolver is usually applied to a transformation of A instead, which plays a similar role as a preconditioner for linear systems. Commonly used transformations such as shift-invert are unreliable and may cause convergence to a wrong eigenvalue. We propose using the exponential transformation since the rightmost eigenvalues of A correspond to the dominant ones of e^{hA} ($h > 0$), which are easily captured by iterative eigensolvers. Numerical experiments on several challenging eigenvalue problems arising from linear stability analysis and pseudospectral analysis demonstrate the robustness of the exponential transformation at "preconditioning" the rightmost eigenvalue. The key to the efficiency of this preconditioner is a fast algorithm for approximating the action of e^{hA} on a vector. We develop a new algorithm based on a rational approximation of e^x with only one (repeated) pole. Compared to polynomial approximations, it converges significantly faster when the spectrum of A has a wide horizontal span, which is common for matrices arising from PDEs; compared to the Krylov-type methods, our method requires considerably less memory.

Key words. linear stability analysis, rightmost eigenvalue, Arnoldi's method, matrix exponential, Leja points, rational approximation, pseudospectral analysis

AMS subject classifications. 65D05, 65F15, 65F50

DOI. 10.1137/17M1132537

1. Introduction.

This paper concerns how to *reliably* determine the stability of the steady state of a large-scale linear or linearized dynamical system

$$\dot{\mathbf{u}} = A\mathbf{u}, \tag{1}$$

typically arising from the semidiscretization of time-dependent partial differential equations (PDEs) in two-dimensional (2-D) or three-dimensional (3-D) domains. The vector $\mathbf{u} \in \mathbb{R}^n$ holds state variables such as velocity, pressure, and temperature, and the Jacobian matrix $A \in \mathbb{R}^{n\times n}$ is large, sparse, and generally nonsymmetric. It is well-known that the stability of the steady state of (1) is dictated by the *rightmost* eigenvalue (i.e., the eigenvalue with algebraically largest real part) of A [17, sect. 1.5]: If this eigenvalue has strictly negative real part, then the steady state is stable, meaning that small disturbances introduced to it will decay in time; otherwise, the steady state is unstable and disturbances will grow with time. Typically, the majority of the eigenvalues of A have negative real parts.

Finding the rightmost eigenvalues of large matrices can be quite difficult. Direct methods such as the QR and QZ algorithms [40, Chap. 2] produce the complete set of eigenvalues but are not feasible for large matrices. Iterative eigensolvers, such as

*Submitted to the journal's Methods and Algorithms for Scientific Computing section May 31, 2017; accepted for publication (in revised form) July 17, 2018; published electronically October 9, 2018.
http://www.siam.org/journals/sisc/40-5/M113253.html
Funding: The first author's work was supported in part by the Simons Foundation under award 527247 and the National Science Foundation under grant DMS-1818833. The second author's work was supported in part by the National Science Foundation under grants DMS-1719461 and DMS-1819097.

†Department of Mathematics, Syracuse University, Syracuse, NY 13244 (mwrostam@syr.edu).
‡Department of Mathematical Sciences, Clemson University, Clemson, SC 29634 (fxue@clemson.edu).

subspace iteration [34, Chap. 5] and the Arnoldi's method and variants (see [32, 39] and [34, Chaps. 6 and 7]) are both efficient and reliable at computing a few large (in modulus), exterior, and/or well-separated eigenvalues of a large sparse matrix. However, the rightmost eigenvalue of A is often neither large nor well-separated.

To achieve reliable and rapid convergence, we often apply iterative eigensolvers to a transformation $\mathcal{T}(A)$ of A to find the rightmost eigenvalue. An ideal transformation $\mathcal{T}$ should satisfy the following three criteria [24]: (i) the rightmost eigenvalue of A is mapped to a large and/or well-separated eigenvalue of $\mathcal{T}(A)$, (ii) the matrix $\mathcal{T}(A)$, which may be too costly to form, can still be applied to vectors efficiently, and (iii) it is easy to recover the eigenpairs of A from those of $\mathcal{T}(A)$. This transformation can therefore be viewed as a *preconditioner* for the rightmost eigenvalue.

Several preconditioners have been proposed in the literature to achieve this goal, namely, the shift-invert transformation [14] $\mathcal{S}_\sigma : A \to (A - \sigma I)^{-1}$, the Cayley transformation [16] $\mathcal{C}_{\sigma_1,\sigma_2} : A \to (A - \sigma_1 I)^{-1}(A - \sigma_2 I) = I + (\sigma_1 - \sigma_2)\mathcal{S}_{\sigma_1}(A)$, and the Kronecker sum transformation [13, 26] $\mathcal{K} : A \to \frac{1}{2}(A \otimes I + I \otimes A)$. The shift-invert transformation is reliable for finding the eigenvalues of A closest to σ; however, it is not suitable in our problem setting since an estimate for the rightmost eigenvalue of A is usually not available a priori. When $\sigma_1, \sigma_2 \in \mathbb{R}$ and $\sigma_1 > \sigma_2$, the Cayley transformation maps the eigenvalues of A on the right of the vertical line $x = \frac{\sigma_1+\sigma_2}{2}$ to those of $\mathcal{C}_{\sigma_1,\sigma_2}(A)$ outside the unit circle centered at the origin; its effectiveness at finding the rightmost eigenvalue of A relies heavily on the choice of σ_1 and σ_2, which are again difficult to configure beforehand. If A is real and its rightmost eigenvalue has negative real part, the most reliable existing preconditioner is the Kronecker sum transformation; it maps the rightmost eigenvalue of A to the smallest eigenvalue of $\mathcal{K}(A)$, which can be computed by the Lyapunov inverse iteration [12, 25]. However, this approach could fail if the rightmost eigenvalue has positive real part or if A is not real; moreover, it requires solving large-scale Lyapunov equations, and iterative methods such as [9, 10, 37, 29] are often storage-consuming.

In this paper, we propose using the exponential transformation

$$\mathcal{E} : A \to e^{hA} = I + hA + \frac{(hA)^2}{2} + \cdots \frac{(hA)^k}{k!} + \cdots \tag{2}$$

where $h > 0$ as a preconditioner for finding the rightmost eigenvalue of A. It is robust because for *any* square, complex, and diagonalizable matrix A and *any* $h > 0$, the rightmost eigenvalue of A is *guaranteed* to be mapped to the largest eigenvalue of e^{hA}. Applying an iterative eigensolver to e^{hA} entails a sequence of matrix-vector multiplications of the form $e^{hA}\mathbf{v}$.

Many algorithms, such as short-term recurrence polynomial approximations [1, 3, 6, 7, 8, 35] and polynomial/rational Krylov-type methods [20, 22, 27, 28, 33, 36], have been developed to approximate $e^{hA}\mathbf{v}$ without constructing e^{hA} (typically dense even if A is sparse). A comparison of some polynomial methods has been presented in [6], which shows the advantages of the polynomial Leja method [7, 8]. In this paper, we develop a new method for computing $e^{hA}\mathbf{v}$ based on the Newton interpolation of $e^{f(\xi)}$ at Leja points [30], where f is a rational function. The change of variable $x = f(\xi)$ is suggested in [42, Chap. 25] for approximating e^x over $(-\infty, 0]$. We refer to this new method as the rational Leja method. It outperforms the polynomial Leja method for most of the matrices considered in this study. When compared with the Restricted Denominator (RD)-rational method [27], both Leja methods usually take a longer time to converge; nevertheless, they require considerably less memory.

When A originates from real-world applications, its entries are often contaminated by small errors; in addition, eigenvalues do not determine the behavior of the transient solution to (1), which can be very different from the behavior of the steady state [43, sect. 14]. For instance, there are well-known examples in fluid mechanics that are predicted to be stable by eigenvalue analysis yet observed to be unstable in a lab environment [43, sect. 20]. A more robust measurement of stability is the rightmost point of the *ε-pseudospectrum* [43, sect. 2] of A:

$$\Lambda_\varepsilon(A) = \left\{ z \in \mathbb{C} \mid z \text{ is an eigenvalue of } A + \Delta, \text{ where } \Delta \in \mathbb{C}^{n\times n} \text{ and } \|\Delta\| \le \varepsilon \right\}, \tag{3}$$

where $\varepsilon > 0$ and $\|\cdot\|$ can be any matrix norm. In some applications, it is necessary to restrict the perturbation matrices Δ in (3) to be real, resulting in the *real-structured ε-pseudospectrum* [43, sect. 50] of A, denoted by $\Lambda_\varepsilon^{\mathbb{R}}(A)$. Several fast algorithms [18, 19, 31] have been developed to find the rightmost point of $\Lambda_\varepsilon(A)$ or $\Lambda_\varepsilon^{\mathbb{R}}(A)$ for large and sparse A, and they all require computing the rightmost eigenvalues for *a sequence* of low-rank updates of A. Our preconditioner $\mathcal{E}$ works particularly well for these eigenvalue problems.

The rest of this paper is organized as follows. In section 2, we present the main observation that relates the eigenpairs of A to those of e^{hA} and provide a motivational example to illustrate the effectiveness of the exponential transformation. In section 3, we review the polynomial Leja method and introduce the single-pole rational Leja method. In section 4, numerical results of the new preconditioner are presented and compared with those of some existing preconditioners. The performance of several methods for computing the action of the matrix exponential is also compared. Section 5 explores the applications of the exponential transformation in pseudospectral analysis. Concluding remarks are made in section 6.

Throughout the paper, we use $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ to denote the real and imaginary parts of a complex number or vector, respectively; and $\|\cdot\|$ denotes the Euclidean norm from now on.

2. Main theorems and a motivational example.

In this section, we show that the exponential transformation is a reliable preconditioner for computing a few rightmost eigenvalues. Specifically, the rightmost eigenvalues of A are mapped to the dominant eigenvalues of e^{hA} $(h > 0)$, which are also well-separated if h is reasonably large. It is easy to recover the rightmost eigenvalues of A via the Rayleigh–Ritz projection once the dominant eigenvectors of e^{hA} are found.

We investigate the one-to-one correspondence between the eigenpairs of A and those of e^{hA} in Theorem 2.1 and Corollary 2.2. The proof of the former is straightforward and omitted.

THEOREM 2.1. *Assume that $A \in \mathbb{C}^{n\times n}$ is diagonalizable. Then $(e^{h\mu}, \mathbf{x})$ is an eigenpair of e^{hA} if and only if $(\mu, \mathbf{x})$ is an eigenpair of A, where $\mu \in \mathbb{C}$ and $\mathbf{x} \in \mathbb{C}^n$.*

COROLLARY 2.2. *Assume that $A \in \mathbb{C}^{n\times n}$ is diagonalizable and $h > 0$. Let $\{(\mu_j, \mathbf{x}_j)\}_{j=1}^n$ be the eigenpairs of A ordered such that the real parts of $\{\mu_j\}_{j=1}^n$ are decreasing, i.e.,*

$$\mathrm{Re}(\mu_1) \ge \mathrm{Re}(\mu_2) \ge \cdots \ge \mathrm{Re}(\mu_n). \tag{4}$$

Then the following statements hold.

1. *Let $\lambda_j = e^{h\mu_j}$. Then $\{(\lambda_j, \mathbf{x}_j)\}_{j=1}^n$ are the eigenpairs of e^{hA} and the moduli of $\{\lambda_j\}_{j=1}^n$ are decreasing, i.e., $|\lambda_1| \ge |\lambda_2| \ge \cdots \ge |\lambda_n|$.*
2. *If $\mathrm{Re}(\mu_k) > \mathrm{Re}(\mu_{k+1})$, then the ratio $\frac{|\lambda_{k+1}|}{|\lambda_k|} = e^{h(\mathrm{Re}(\mu_{k+1}) - \mathrm{Re}(\mu_k))}$ is a decreasing function of h; in addition, $\lim_{h\to+\infty} \frac{|\lambda_{k+1}|}{|\lambda_k|} = 0$ and $\lim_{h\to 0} \frac{|\lambda_{k+1}|}{|\lambda_k|} = 1$.*

3. *On the complex plane, λ_j lies*
$$\begin{cases} \textit{inside the unit circle centered at } (0,0) \\ \qquad \textit{if } \mathrm{Re}(\mu_j) < 0, \\ \textit{on the unit circle centered at } (0,0) \\ \qquad \textit{if } \mathrm{Re}(\mu_j) = 0, \\ \textit{outside the unit circle centered at } (0,0) \\ \qquad \textit{if } \mathrm{Re}(\mu_j) > 0. \end{cases}$$

Proof. By Theorem 2.1, $\{(e^{h\mu_j}, \mathbf{x}_j)\}$ are the eigenpairs of e^{hA}. Statements 1 and 2 follow from

$$|\lambda_j| = \left|e^{h\mu_j}\right| = e^{\mathrm{Re}(h\mu_j)} = e^{h\mathrm{Re}(\mu_j)}, \tag{5}$$

$h > 0$, and (4). Statement 3 is obvious from (5). ∎

By statement 1 of Corollary 2.2, the k rightmost eigenvalues $\{\mu_j\}_{j=1}^k$ of A are mapped to the k dominant eigenvalues $\{\lambda_j\}_{j=1}^k$of e^{hA} for all $1 \le k \le n$. In addition, since μ_j and λ_j share eigenvectors, once we have obtained the eigenvectors $\{\mathbf{x}_j\}_{j=1}^k$ of e^{hA}, recovering the eigenvalues $\{\mu_j\}_{j=1}^k$ of A is easy by the Rayleigh–Ritz projection. It is well-known that the closer the ratio $\frac{|\lambda_{k+1}|}{|\lambda_k|}$ in statement 2 is to 1, the slower subspace iteration or Arnoldi's method and its variants converge to the dominant k eigenvalues of e^{hA}. Increasing h accelerates the convergence rate of the eigensolver, but it also makes $e^{hA}\mathbf{v}$ more difficult to compute (see later discussions). Statement 3 describes how the position of an eigenvalue λ_j of e^{hA} relative to the unit circle centered at $(0,0)$ is determined by the sign of the real part of the corresponding eigenvalue μ_j of A (see Figure 1). In fact, it is easy to see that each vertical line in the complex plane is mapped to a circle centered at the origin by the exponential transformation.

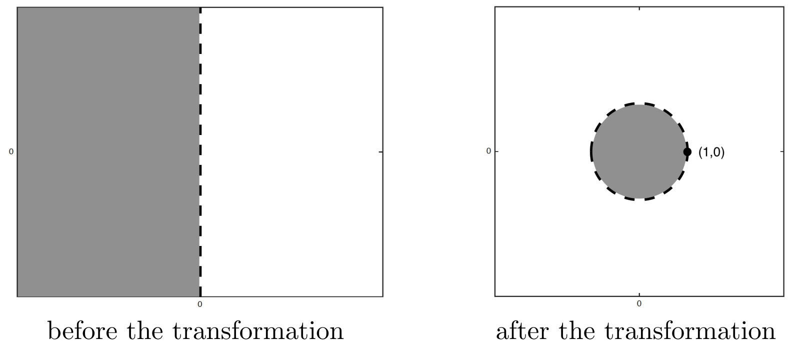

FIG. 1. *The effect of the exponential transformation e^{hA} with $h > 0$ on the complex plane.*

To illustrate the advantages of the exponential transformation, we consider a 4000×4000 nonsymmetric real matrix A that arises from aeroelasticity and is referred to as *Tolosa* in this paper. In Figure 2, the eigenvalues of both A and e^{hA} with $h = 1$ are displayed. The right plot indicates that except for a number of rightmost eigenvalues, most of the eigenvalues of A are mapped to a tight cluster around the origin. The rightmost eigenvalues $\mu_{1,2} = -0.156 \pm 156\mathrm{i}$ of A (crosses in the left plot) are mapped to the dominant eigenvalues $\lambda_{1,2} = e^{\mu_{1,2}} = 0.40347 \pm 0.75445\mathrm{i}$ of e^{hA} (crosses in the right plot). The second rightmost pair of eigenvalues of A are $\mu_{3,4} = -0.22394 \pm 162\mathrm{i}$, which get mapped to $\lambda_{3,4} = 0.16493 \pm 0.78216\mathrm{i}$. The ratio

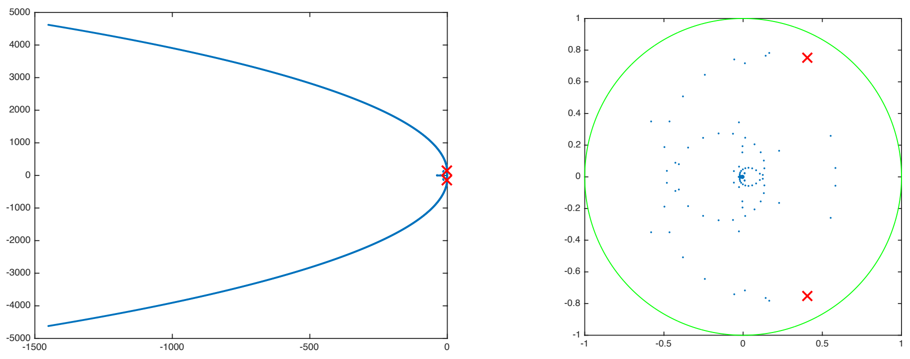

Fig. 2. *The complete sets of eigenvalues of A (left panel) and e^A (right panel) in the Tolosa case. The circle in the right plot is the unit circle centered at* $(0,0)$. *The crosses represent the rightmost eigenvalues of A or the largest eigenvalues of e^A.*

$\frac{|\lambda_{3,4}|}{|\lambda_{1,2}|} = e^{h(\mathrm{Re}(\mu_{3,4})-\mathrm{Re}(\mu_{1,2}))} = 0.93432$ is not very close to 1. Therefore, we expect an iterative eigensolver for e^{hA} to converge to $\lambda_{1,2}$ in a reasonable number of iterations, and this is indeed the case in our experiments in section 4.

3. The polynomial and single-pole rational Leja method.

Applying an iterative eigenvalue solver to compute dominant eigenvalues of e^{hA} entails multiple matrix-vector products with e^{hA}. We review the polynomial Leja method and describe the new rational Leja method for computing the action of the matrix exponential, both of which are based on the Newton interpolation at Leja points. A major advantage of both Leja methods is that they are very storage-efficient. In the following exposition, we assume that the majority of the eigenvalues of A have negative real parts. This is typical for matrices of interest in linear stability analysis.

3.1. Review of the polynomial Leja method.

This method is based on Newton's interpolation formula of e^x with divided differences [21, Chap. 2]:

$$e^x \approx p_\ell(x) = \textstyle\sum_{j=0}^{\ell} d_j r_j(x), \qquad x \in K, \tag{6}$$

where $\{d_j\}_{j=0}^{\ell}$ are the divided differences of e^x on distinct interpolation points $\{x_j\}_{j=0}^{\ell} \subset K$, i.e.,

$$d_0 = e[x_0] = e^{x_0},\ d_1 = \frac{e[x_1]-e[x_0]}{x_1-x_0},\ d_j = \frac{e[x_1,\dots,x_j]-e[x_0,\dots,x_{j-1}]}{x_j-x_0} \text{ for } j \ge 2,$$

and $r_j(x)$ is the following jth degree monomial:

$$r_0(x)=1, \quad r_j(x) = (x-x_0)\cdots(x-x_{j-1}) = \textstyle\prod_{i=0}^{j-1}(x-x_i) \text{ for } j \ge 1.$$

An attractive feature of the Newton form (6) is that given a new interpolation point $x_{\ell+1}$, it is easy to obtain $p_{\ell+1}(x)$ from $p_\ell(x)$ and other known quantities as follows:

$$r_{\ell+1}(x) = (x-x_\ell)r_\ell(x), \quad p_{\ell+1}(x) = p_\ell(x) + d_{\ell+1}r_{\ell+1}(x) \quad \text{for } \ell \ge 0.$$

A sequence of Leja points in a compact set $K \subset \mathbb{C}$ is defined recursively as

$$x_0 = \arg\max_{x\in K}|x|, \quad x_j = \arg\max_{x\in K}|x-x_0|\cdots|x-x_{j-1}| \text{ for } j \ge 1, \tag{7}$$

which are in general not unique. Using Leja points as the interpolation points has the advantages of a near-optimal, geometric rate of convergence and controlling the "condition number" of (6) as ℓ grows (see [30] and the references therein).

By replacing x with hA in (6) and right-multiplying both sides of it by $\mathbf{v}$, we could get a polynomial approximation of $e^{hA}\mathbf{v}$. When $h > 0$ is large, a polynomial of high degree is necessary in (6), which limits the accuracy of this approximation (see [6] for more details). Instead, $e^{hA}\mathbf{v}$ is computed in a number of substeps in [6, 7, 8], i.e.,

$$e^{hA}\mathbf{v} = \underbrace{e^{\tau A} \cdots e^{\tau A}}_{T \text{ applications}} \mathbf{v}, \tag{8}$$

where $T \geq 1$ is the number of substeps and $\tau = h/T$ is the size of each substep. The choice of Leja points is crucial in the computation of $e^{\tau A}\mathbf{v}$ and we briefly review the approach of [6, 8] in Appendix A. The polynomial Leja method with substepping is outlined in Algorithm 1.

Algorithm 1 The polynomial Leja method with substepping for computing $e^{hA}\mathbf{v}$.

```
Input: $h$, $A$, $\mathbf{v}$, $\alpha$, $\nu$, $\beta$, $\tau$, $L$, $reltol$, Leja points $\{\xi_j\}_{j=0}^{L}$ on $[-2,2]$ and $\{\zeta_j\}_{j=0}^{L}$ on $\mathfrak{i}[-2,2]$
Output: An approximation $\mathbf{w}$ of $e^{hA}\mathbf{v}$
1: Determine $\mathfrak{c}$, $\mathfrak{d}$, and $K$ from $\alpha$, $\nu$, and $\beta$ (see Appendix A).
2: $\mathbf{w} \leftarrow \mathbf{v}$, $T \leftarrow \lceil h/\tau \rceil$, $\tau \leftarrow h/T$
3: if $K$ is horizonal then
4: Compute the divided differences $\{\widehat{d}_j\}_{j=0}^{L}$ of $e^{\tau(\mathfrak{c}+\frac{\mathfrak{d}}{2}\xi)}$ on $\{\xi_j\}_{j=0}^{L}$;
5: $\{\omega_j\}_{j=0}^{L-1} \leftarrow \{\frac{2\mathfrak{c}}{\mathfrak{d}} + \xi_j\}_{j=0}^{L-1}$
6: else
7: Compute the divided differences $\{\widehat{d}_j\}_{j=0}^{L}$ of $e^{\tau(\mathfrak{c}+\frac{\mathfrak{d}}{2}\zeta)}$ on $\{\zeta_j\}_{j=0}^{L}$;
8: $\{\omega_j\}_{j=0}^{L-1} \leftarrow \{\frac{2\mathfrak{c}}{\mathfrak{d}} + \zeta_j\}_{j=0}^{L-1}$
9: endif
10: for $t = 1, \ldots, T$ do
11:     $\mathbf{r} \leftarrow \mathbf{w}$, $\mathbf{w} \leftarrow \widehat{d}_0\mathbf{r}$
12:     for $\ell = 1, \ldots, L$ do
13:         $\mathbf{r} \leftarrow \left(\frac{2}{\mathfrak{d}}A - \omega_{\ell-1}I\right)\mathbf{r}$
14:         $\Delta\mathbf{w} \leftarrow \widehat{d}_\ell\mathbf{r}$, $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$
15:         if $\|\Delta\mathbf{w}\|/\|\mathbf{w}\| < reltol$ then
16:             break from the inner for loop
17:         endif
18:     endfor
19: endfor
```

3.2. The single-pole rational Leja method. We observe from experiments that the polynomial Leja method (Algorithm 1) suffers from slow convergence when the spectrum of A has a wide horizontal span along $(-\infty, 0]$. This spectral property is shared by many matrices arising from spatial discretization of PDEs. To tackle this difficulty, we propose a rational Leja method.

First consider the scalar case of the problem: approximating e^x over $(-\infty, 0]$. It is well-known [42, Chap. 25] that the scalar case is difficult for polynomial interpolations since they diverge as $x \to -\infty$. A rational approximant is naturally much more suitable. In [42, Chap. 25], the following method is used: For a fixed $a > 0$, let $\xi \in (-2, 2]$ satisfy $x = a\frac{\xi-2}{\xi+2} \in (-\infty, 0]$ and define $f(\xi) = e^{a\frac{\xi-2}{\xi+2}}$;[1] then compute a polynomial approximation $q_\ell(\xi)$ of $f(\xi)$, which can in turn be written as a rational approximation of e^x in the variable x since $\xi = 2(a + x)/(a - x)$.

We propose using Newton interpolation polynomial of $f(\xi)$ on Leja points, that is,

$$e^x = f(\xi) \approx q_\ell(\xi) = \textstyle\sum_{j=0}^{\ell} \delta_j \widehat{r}_j(\xi),\ \xi \in (-2, 2], \tag{9}$$

where $\{\delta_j\}_{j=0}^{\ell}$ are the divided differences of $f(\xi)$ on Leja points $\{\xi_j\}_{j=0}^{\ell} \subset (-2, 2]$, and the monomials $\widehat{r}_j(\xi)$ are defined as

$$\widehat{r}_0(\xi) = 1, \quad \widehat{r}_j(\xi) = (\xi - \xi_0) \cdots (\xi - \xi_{j-1}) = \textstyle\prod_{i=0}^{j-1}(\xi - \xi_i) \text{ for } j \geq 1.$$

Since $\xi = 2\frac{a+x}{a-x}$, (9) gives a rational approximation

$$e^x \approx \Upsilon_{\ell,\ell}(x) = \textstyle\sum_{j=0}^{\ell} \delta_j \widehat{r}_j\big(2(a + x)/(a - x)\big) = \sum_{j=0}^{\ell} \delta_j \rho_{j,j}(x),\ x \in (-\infty, 0], \tag{10}$$

where

$$\rho_{0,0}(x) = 1, \quad \rho_{j,j}(x) = \textstyle\prod_{i=0}^{j-1} \big(2(a + x)/(a - x) - \xi_i\big) \text{ for } j \geq 1. \tag{11}$$

Note that $\rho_{j,j}(x)$ is a rational function of type (j, j), which can be written as the quotient of two polynomials of degree j (see [42, Chap. 23]). It is not difficult to see that $\Upsilon_{\ell,\ell}(x)$ is of type (ℓ, ℓ). In addition, updating $\Upsilon_{\ell,\ell}(x)$ to $\Upsilon_{\ell+1,\ell+1}(x)$ is easy:

$$\begin{aligned} \rho_{\ell+1,\ell+1}(x) &= \big(2(a + x)/(a - x) - \xi_\ell\big)\rho_{\ell,\ell}(x), \\ \Upsilon_{\ell+1,\ell+1}(x) &= \Upsilon_{\ell,\ell}(x) + \delta_{\ell+1}\rho_{\ell+1,\ell+1}(x) \quad \text{for } \ell \geq 0. \end{aligned} \tag{12}$$

We now return to the problem of approximating $e^{hA}\mathbf{v}$. By replacing x with hA in (10)–(12) and right-multiplying both sides of them by $\mathbf{v}$, we obtain the approximation

$$e^{hA}\mathbf{v} \approx \Upsilon_{\ell,\ell}(hA)\mathbf{v} = \textstyle\sum_{j=0}^{\ell} \delta_j \rho_{j,j}(hA)\mathbf{v} \tag{13}$$

as well as means of updating it, i.e.,

$$\begin{aligned} \rho_{\ell+1,\ell+1}(hA)\mathbf{v} &= \big(2(aI - hA)^{-1}(aI + hA) - \xi_\ell I\big)\, \rho_{\ell,\ell}(hA)\mathbf{v} \quad \text{for } \ell \geq 0, \\ \Upsilon_{\ell+1,\ell+1}(hA)\mathbf{v} &= \Upsilon_{\ell,\ell}(hA)\mathbf{v} + \delta_{\ell+1}\rho_{\ell+1,\ell+1}(hA)\mathbf{v}. \end{aligned} \tag{14}$$

The choice of the scaling factor a will be discussed in section 3.3.2.

The single-pole rational Leja method with substepping (8) is summarized in Algorithm 2. It can be viewed both as a generalization of the method in [42, Chap. 25] to the matrix case and as a generalization of the polynomial Leja method [6, 7, 8] to $e^{f(B)}\mathbf{v}$ where f is a rational function. A particularly appealing property of the rational approximation $\Upsilon_{\ell,\ell}$ is that it only has one pole (with multiplicity ℓ). Consequently, although every update (14) requires solving a linear system, its coefficient matrix $aI - hA$ stays the same. We may prefactorize it or construct a preconditioner

[1]To be more precise, $\xi \in (-1, 1]$ such that $x = a\frac{\xi-1}{\xi+1} \in (-\infty, 0]$ and $f(\xi) = e^{a\frac{\xi-1}{\xi+1}}$ are used instead in [42, Chap. 25]. Here, we rescale ξ so that its domain has unit capacity, as in Appendix A.

Algorithm 2 The single-pole rational Leja method with substepping for computing $e^{hA}\mathbf{v}$.

```
Input: $h$, $A$, $\mathbf{v}$, $\tau$, $L$, $a$, reltol, Leja points $\{\xi_j\}_{j=0}^{L}$ on $(-2,2]$
Output: An approximation $\mathbf{w}$ of $e^{hA}\mathbf{v}$
1:  Compute the $L$ divided differences $\{\delta_j\}_{j=0}^{L-1}$ of $e^{a\frac{\xi-2}{\xi+2}}$ on $\{\xi_j\}_{j=0}^{L-1}$.
2:  $\mathbf{w} \leftarrow \mathbf{v}$, $T \leftarrow \lceil h/\tau \rceil$, $\tau \leftarrow h/T$
3:  for $t = 1, \dots, T$ do
4:      $\mathbf{r} \leftarrow \mathbf{w}$, $\mathbf{w} \leftarrow \delta_0 \mathbf{r}$
5:      for $\ell = 1, \dots, L$ do
6:          Solve $(aI - \tau A)\mathbf{y} = (aI + \tau A)\mathbf{r}$ for $\mathbf{y}$.
7:          $\mathbf{r} \leftarrow 2\mathbf{y} - \xi_{\ell-1}\mathbf{r}$
8:          $\Delta\mathbf{w} \leftarrow \delta_\ell \mathbf{r}$, $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$
9:          if $\|\Delta\mathbf{w}\|/\|\mathbf{w}\| < reltol$ then
10:             break from the inner for loop
11:         endif
12:     endfor
13: endfor
```

for it "once and for all," which is usually much more efficient than a sequence of linear solves with different coefficient matrices. This is a major advantage over the rational approximations that have multiple distinct poles for large matrices. Moreover, unlike in the polynomial Leja method (see Appendix A), there is no need to find a rectangle $[\alpha, \nu] \times [-\beta, \beta]$ containing the eigenvalues of A.

We end this section with a minor technical comment. If the set K is not compact, such as $(-2, 2]$, the optimization problem (7) may not have a solution. The sequence of Leja points used in Algorithm 2 is obtained as follows: We first find a sequence of Leja points in $[-2, 2]$; since -2 is guaranteed to be among them, we replace it with $-2 + \vartheta$ where $\vartheta > 0$ is small.

3.3. Parameter estimation. In this section, we discuss how to determine the parameter values for of the polynomial Leja method (Algorithm 1) and the single-pole rational Leja method (Algorithm 2). These parameters are listed in Table 1.

TABLE 1
The parameters of Algorithms 1 *and* 2.

Parameter	Algorithm	Description
α, ν, β	1	spectral bounds of A (see Appendix A)
τ	1 and 2	substep size (8)
L	1 and 2	maximum degree of Newton polynomials (9) and (20)
reltol	1 and 2	tolerance used in the stopping criterion (see line 15 of Algorithm 1 and line 9 of Algorithm 2)
a	2	scaling factor used in the change of variable $x = a\frac{\xi-2}{\xi+2}$

3.3.1. Estimate the spectral bounds for polynomial Leja. Algorithm 1 requires finding a rectangle $[\alpha, \nu] \times [-\beta, \beta]$ enclosing the spectrum of A; see Appendix A. Estimating the bound ν requires approximating the rightmost eigenvalue of A, which is challenging and, in fact, the main goal of this paper. Based on the assumption at the beginning of section 3, it is reasonable to simply take ν to be 0. We focus on estimating the bounds α and β.

It is obvious that for all $1 \le j \le n$,

$$\mathrm{Re}(\mu_j) \ge -\max\{|\mu_i|\}_{i=1}^n \quad \text{and} \quad |\mathrm{Im}(\mu_j)| \le \max\{|\mu_i|\}_{i=1}^n.$$

Therefore, we can choose

$$\alpha = -\max\{|\mu_i|\}_{i=1}^n \quad \text{and} \quad \beta = \max\{|\mu_i|\}_{i=1}^n. \tag{15}$$

Alternatively, the bounds α and β can also be determined as follows. Let $\{\theta_j\}_{j=1}^n$ and $\{\eta_j\}_{j=1}^n$ be the eigenvalues of $\frac{1}{2}(A + A^T)$ and $\frac{1}{2\mathrm{i}}(A - A^T)$, respectively. By [15, Theorem 2],

$$\mathrm{Re}(\mu_j) \ge \min\{\theta_i\}_{i=1}^n \quad \text{and} \quad \mathrm{Im}(\mu_j) \le \max\{\eta_i\}_{i=1}^n. \tag{16}$$

Since

$$\min\{\theta_i\}_{i=1}^n \ge -\max\{|\theta_i|\}_{i=1}^n \quad \text{and} \quad \max\{\eta_i\}_{i=1}^n \le \max\{|\eta_i|\}_{i=1}^n,$$

we can also choose

$$\alpha = -\max\{|\theta_i|\}_{i=1}^n \quad \text{and} \quad \beta = \max\{|\eta_i|\}_{i=1}^n. \tag{17}$$

In order to find α, β defined in (15) or (17), we apply the MATLAB function `eigs` with the option `lm` (largest magnitude) to A or to $\frac{1}{2}(A + A^T)$ and $\frac{1}{2\mathrm{i}}(A - A^T)$. If (1) arises from a finite element discretization, then A is in the form of $M^{-1}J$, where both M, J are sparse and $M^{-1}J$ is often too costly to compute. In this case, we apply `eigs` to the *generalized* eigenvalue problem $J\mathbf{x} = \mu M\mathbf{x}$ or to $\frac{1}{2}(J + J^T)\mathbf{x} = \mu M\mathbf{x}$ and $\frac{1}{2\mathrm{i}}(J - J^T)\mathbf{x} = \mu M\mathbf{x}$. In [6, 8], α and β are chosen to be the lower bound of $\{\theta_i\}_{i=1}^n$ and the upper bound of $\{\eta_i\}_{i=1}^n$ obtained by applying the Gershgorin circle theorem to $\frac{1}{2}(A + A^T)$ and $\frac{1}{2\mathrm{i}}(A - A^T)$. However, unlike `eigs`, the Gershgorin theorem is *not* directly applicable to generalized eigenvalue problems.

Our strategy is the following: Compute both sets of α, β given by (15) and (17), and choose the algebraically larger α and the smaller β. The main cost of this strategy is finding the largest eigenvalues for A, $\frac{1}{2}(A+A^T)$, and $\frac{1}{2\mathrm{i}}(A-A^T)$. Although it is more costly than applying the Gershgorin theorem, it is applicable to A arising from a finite element discretization; its runtime is negligible compared to the runtime saved by using tighter spectral bounds in Algorithm 1, especially in cases where many matrix-vector products $e^{hA}\mathbf{v}$ need to be evaluated.

There is no need to estimate the spectral bounds of A for the rational Leja method. Given a matrix A, we need only find the largest substep size τ such that $e^{\tau A}\mathbf{v}$ can be approximated by $\Upsilon_{L,L}(\tau A)\mathbf{v}$ accurately. This will be discussed in section 3.3.3.

3.3.2. The scaling factor a and the degree L of Newton polynomials. Assume that A is diagonalizable and that $A = P\Lambda P^{-1}$ is its eigenvalue decomposition. Then

$$\begin{aligned} e^{hA}\mathbf{v} &= e^{hP\Lambda P^{-1}}\mathbf{v} = Pe^{h\Lambda}P^{-1}\mathbf{v}, \\ \Upsilon_{L,L}(hA)\mathbf{v} &= \Upsilon_{L,L}(hP\Lambda P^{-1})\mathbf{v} = P\Upsilon_{L,L}(h\Lambda)P^{-1}\mathbf{v}, \end{aligned} \tag{18}$$

where $\Upsilon_{L,L}(x)$ is defined in (10). Equation (18) implies that

$$\|e^{hA}\mathbf{v} - \Upsilon_{L,L}(hA)\mathbf{v}\| \le \|P\| \cdot \|e^{h\Lambda} - \Upsilon_{L,L}(h\Lambda)\| \cdot \|P^{-1}\| \cdot \|\mathbf{v}\|. \tag{19}$$

Ideally, to minimize the upper bound in (19), we should choose a and L such that $\Upsilon_{L,L}(x)$ approximates e^x as accurately as possible at the eigenvalues of hA. Since the eigenvalues of hA can be anywhere on the complex plane, we aim to perform the following parameter optimization: Find a and L such that the region on the complex plane where $\Upsilon_{L,L}(x)$ approximates e^x well is as large as possible. To obtain a crude solution to this problem, we discretize the region $[-100, 5] \times [-25, 25]$ on the complex plane using a Cartesian grid and compute the approximation errors $|e^x - \Upsilon_{L,L}(x)|$ at the grid points; we also restrict the choices of a, L to integers between 1 and 100. The optimal values of a and L produced by this heuristic are 50 and 45, respectively.[2] To show the effectiveness of the choice $L = 45$, we fix $a = 50$ and display the contour plots of $\log_{10} |e^x - \Upsilon_{L,L}(x)|$ corresponding to $L = 35$, 45, and 55 in Figure 3. It can be seen that the region where $|e^x - \Upsilon_{L,L}(x)| \leq 10^{-10}$ is the largest when $L = 45$. We emphasize that our approach does *not* require any spectral information about A. The choices $a = 50$ and $L = 45$ will be used for all matrices.

Note that we cannot apply the same heuristic to find a problem-independent L for the *polynomial* Leja method since its choice of Newton polynomial depends on A (see Appendix A). We simply use $L = 100$ in the polynomial Leja method.

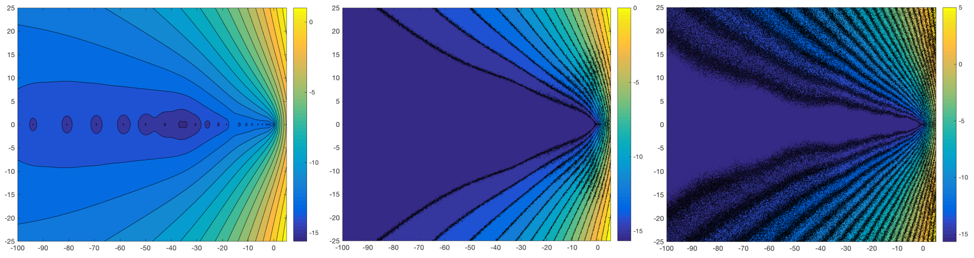

FIG. 3. *The contour plots of* $\log_{10} |e^x - \Upsilon_{L,L}(x)|$ *corresponding to* $L = 35$ *(left),* 45 *(middle), and* 55 *(right)* $(a = 50$ *is fixed).*

3.3.3. The substep size τ. For fixed L and *reltol*, the larger τ is, the fewer substeps T are required by Algorithm 1 or 2. We aim to find the largest τ such that a Newton polynomial of degree L (for Algorithm 1) or a rational function of type (L, L) (for Algorithm 2) is accurate enough for approximating $e^{\tau A}\mathbf{v}$ without substepping.

Our approach outlined in Algorithm 3 is essentially a bisection method. The purpose of lines 1 to 10 is to first identify a "coarse" interval $[t_l, t_u]$ such that Algorithm 1 or 2 *without substepping* converges for $e^{\tau_l A}\mathbf{v}$ but not for $e^{\tau_u A}\mathbf{v}$, where $\tau_l = 2^{t_l}$ and $\tau_u = 2^{t_u}$. From lines 11 to 19, we keep bisecting the interval $[t_l, t_u]$ until a fine enough new interval $[t_l, t_u]$ is found such that Algorithm 1 or 2 *without substepping* again converges for $e^{\tau_l A}\mathbf{v}$ but not for $e^{\tau_u A}\mathbf{v}$. Note that since no estimate for the optimal τ is available in general, to quickly determine its order of magnitude, we bisect the interval $[t_l, t_u]$ of the exponent instead of the interval $[\tau_l, \tau_u]$ of τ directly. In the numerical experiments, $t_l = -5$, $t_u = 5$, $\Delta t = 10$, and $tol_t = 0.01$ are used in Algorithm 3. This algorithm can be modified easily to compute τ for methods other than Algorithms 1 and 2.

[2]We note that the following is stated in [42, Chap. 25] with no further explanation: "*A good choice of the parameter is $a = 9$, which has a big effect for numerical computation in improving the conditioning for the approximation problem.*" In our experiments, $a = 50$ leads to obviously faster approximation to $e^{hA}\mathbf{v}$ than $a = 9$.

Algorithm 3 Procedure for determining τ.

```
Input: $A$, $\mathbf{v}$, $L$, $a$, $reltol$, Leja points $\{\xi_j\}_{j=0}^{L}$ on $(-2,2]$, $t_l$, $t_u$ ($t_l < t_u$), $\Delta t > 0$, and $tol_t$
Output: $\tau$ for Algorithm 1 or 2 for which $e^{\tau A}\mathbf{v}$ can be approximated without substepping
1: $\tau_l \leftarrow 2^{t_l}$, $\tau_u \leftarrow 2^{t_u}$
2: Apply Algorithm 1 or 2 *without substepping* to compute $e^{\tau_l A}\mathbf{v}$ and $e^{\tau_u A}$
3: while Algorithm 1 or 2 fails to converge for $e^{\tau_l A}\mathbf{v}$ do
4:     $t_u \leftarrow t_l$, $\tau_u \leftarrow \tau_l$, $t_l \leftarrow t_l - \Delta t$, $\tau_l \leftarrow 2^{t_l}$
5:     Apply Algorithm 1 or 2 *without substepping* to compute $e^{\tau_l A}\mathbf{v}$
6: endwhile
7: while Algorithm 1 or 2 successfully converges for $e^{\tau_u A}\mathbf{v}$ do
8:     $t_l \leftarrow t_u$, $\tau_l \leftarrow \tau_u$, $t_u \leftarrow t_u + \Delta t$, $\tau_u \leftarrow 2^{t_u}$
9:     Apply Algorithm 1 or 2 *without substepping* to compute $e^{\tau_u A}\mathbf{v}$
10: endwhile
11: while $t_u - t_l \geq tol_t$ do
12:     $t_m = \frac{t_u + t_l}{2}$, $\tau_m = 2^{t_m}$
13:     Apply Algorithm 1 or 2 *without substepping* to compute $e^{\tau_m A}\mathbf{v}$
14:     if Algorithm 1 or 2 successfully converges for $e^{\tau_m A}\mathbf{v}$ then
15:         $t_l \leftarrow t_m$, $\tau_l \leftarrow 2^{t_l}$
16:     else
17:         $t_u \leftarrow t_m$, $\tau_u \leftarrow 2^{t_u}$
18:     endif
19: endwhile
20: $\tau \leftarrow \tau_l$
```

In [6], τ is chosen so that a Newton polynomial of degree L is sufficient for approximating $e^{\tau S}\mathbf{v}$, where S is a 2×2 matrix whose spectrum is also enclosed in $[\alpha, 0] \times [-\beta, \beta]$. In our experience, this approach tends to underestimate τ, causing the use of an unnecessarily large number of substeps. Algorithm 3 is more expensive, but it produces a larger τ, which in turn reduces the number of substeps and runtime for both the polynomial and the rational Leja methods. When many matrix-vector products with e^{hA} need to be computed, the extra cost of Algorithm 3 is negligible.

4. Numerical results.

We compare the robustness and efficiency of several eigensolver and preconditioner combinations at computing a few rightmost eigenvalues for a representative set of test problems, many of which arise from PDEs. Several methods for computing the action of the matrix exponential are also compared.

4.1. Test problems.

We consider twelve test problems divided into two groups. The first group of six, namely *Tolosa* [4], *Obstacle* (see [11, Chap. 10] and [13]), *Cavity* (see [11, Chap. 8] and [13]), *Plate* [11, Chap. 8], *Crystal* [4], and *Aerofoil* [41], are summarized as follows.

- They stem from a variety of applications: physical chemistry (*Crystal*), fluid mechanics (*Obstacle*, *Cavity*, and *Plate*), and aerodynamics (*Tolosa* and *Aerofoil*).
- All six matrices are real and nonsymmetric with $n = 4000$, 9512, 9539, 9539, 10000, and 16388, respectively. The *Tolosa*, *Crystal*, and *Aerofoil* matrices are sparse.

- The rightmost eigenvalues of the *Tolosa*, *Obstacle*, *Cavity*, and *Aerofoil* matrices have negative real parts, and those of the *Plate* and *Crystal* matrices have positive real parts. The *Crystal* matrix is the only matrix whose rightmost eigenvalue is real.
- The *Crystal* matrix arises from a finite difference discretization, and the *Aerofoil* matrix arises from a finite volume discretization.
- The *Obstacle*, *Cavity*, and *Plate* matrices arise from finite element discretizations of three models of the incompressible Navier–Stokes equations whose respective Reynolds numbers are 350, 7800, and 7850. Here, $A = M^{-1}J$, where J is the Jacobian matrix, M is the mass matrix, and both J and M are sparse. Note that A is never explicitly formed.
- The *Tolosa* and *Crystal* matrices are available from the Matrix Market [4], whereas the *Obstacle*, *Cavity*, *Plate*, and *Aerofoil* matrices are made available by the authors.

The second group of problems are created "artificially" based on the problems from the first group (hence the extra "(a)" in their names). Let A be any $n \times n$ matrix in the first group and continue to denote its eigenvalues by $\{\mu_i\}_{i=1}^n$, where $\mathrm{Re}(\mu_1) \geq \mathrm{Re}(\mu_2) \geq \cdots \geq \mathrm{Re}(\mu_n)$. Let $MaxIm = \max\{\{|\mathrm{Im}(\mu_i)|\}_{i=1}^n, 200\}$. The artificial counterpart of A is of size $(n+4) \times (n+4)$ and has the block-diagonal structure $\mathrm{diag}\{A, G\}$ where $G \in \mathbb{R}^{4\times 4}$ and its eigenvalues are

$$\mu_{1,2}^G = \begin{cases} \pm \mathrm{i} MaxIm \text{ if } \mathrm{Re}(\mu_1) < 0, \\ \frac{1}{2}\left(\mathrm{Re}(\mu_1) + \mathrm{Re}(\mu_2)\right) \pm \mathrm{i} MaxIm \;\text{ otherwise,} \end{cases}$$

$$\mu_{3,4}^G = \begin{cases} \pm \mathrm{i}\frac{1}{2} MaxIm \text{ if } \mathrm{Re}(\mu_1) < 0, \\ \frac{1}{2}\left(\mathrm{Re}(\mu_1) + \mathrm{Re}(\mu_2)\right) \pm \mathrm{i}\frac{1}{2} MaxIm \;\text{ otherwise.} \end{cases}$$

By construction, the eigenvalues of the artificial matrix corresponding to A are $\{\mu_i\}_{i=1}^n \bigcup \{\mu_i^G\}_{i=1}^4$, among which the rightmost or second rightmost pair have the largest (in modulus) imaginary part and the four added eigenvalues have the same real part. When $A = M^{-1}J$ such as for the *Obstacle*, *Cavity*, and *Plate* problems, we augment M and J instead so that the same four eigenvalues $\{\mu_i^G\}_{i=1}^4$ are added to the spectrum of A. We introduce the second group of matrices to increase the moduli of the imaginary parts of the target eigenvalues. As can be seen from sections 4.3 and 4.4, this modification creates rather challenging test problems.

4.2. Description of the numerical experiments. An iterative eigensolver for computing k ($k \ll n$) target eigenvalues of A often has the following *inner-outer* structure:

Outer iteration: iteration of the eigensolver applied to $\mathcal{T}(A)$,
Inner iteration: iteration of an algorithm for approximating $\mathcal{T}(A)\mathbf{v}$,

where $\mathcal{T}$ is a suitable transformation (preconditioner). We compare the performance of five preconditioners: $\mathcal{E}:\ A \to e^{hA}$ (exponential transformation), $\mathcal{I}:\ A \to A$ (no preconditioner), $\mathcal{S}_0:\ A \to A^{-1}$ (shift-invert transformation with zero shift), $\mathcal{C}_{\sigma_1,\sigma_2}:\ A \to (A - \sigma_1 I)^{-1}(A - \sigma_2 I)$ (Cayley transformation), and $\mathcal{K}:\ A \to \frac{1}{2}(A \otimes I + I \otimes A)$ (Kronecker sum transformation). The eigensolver for the first four transformations is the implicitly restarted Arnoldi's (IRA) method [39] implemented in the MATLAB `eigs`, and the eigensolver for the Kronecker sum transformation is the Lyapunov inverse iteration [12, 13]. A brief description of the Lyapunov inverse iteration can be found in Appendix B; roughly speaking, it is a version of inverse iteration designed specifically for $\mathcal{K}(A)$ by exploiting the structure of its eigenvectors. Our targets are the five rightmost eigenvalues of A.

When $\mathcal{T} = \mathcal{E}$, we first apply `eigs` with the option `lm` to find the five largest eigenvalues $\{\lambda_j\}_{j=1}^5$ of e^{hA} and their eigenvectors $\{\mathbf{x}_j\}_{j=1}^5$;[3] and then we recover the five rightmost eigenvalues $\{\mu_j\}_{j=1}^5$ of A according to Corollary 2.2. The inner iteration is computing a matrix-vector product $e^{hA}\mathbf{v}$.

When $\mathcal{T} = \mathcal{I}$, i.e., no preconditioner is used, we apply `eigs` with the option `lr` (largest real part) to A. The inner iteration is multiplying A to a vector.

When $\mathcal{T} = \mathcal{S}_0$, we apply `eigs` with the option `sm` (smallest magnitude) to find the $\widehat{k}$ ($5 \leq \widehat{k} \ll n$) eigenvalues of A closest to the origin. The five rightmost eigenvalues among these $\widehat{k}$ *computed* ones are then selected. However, it is impossible to know a priori how large $\widehat{k}$ needs to be to guarantee finding the five desired eigenvalues. Our heuristic is to choose $\widehat{k} = 100$. The inner iteration of this approach is solving a linear system whose coefficient matrix is A.

When $\mathcal{T} = \mathcal{C}_{\sigma_1,\sigma_2}$, we first apply `eigs` with the option `lm` to find the $\widehat{k}$ largest eigenvalues of $\mathcal{C}_{\sigma_1,\sigma_2}$ and their eigenvectors; since A and $\mathcal{C}_{\sigma_1,\sigma_2}$ have the same eigenvectors, we can then recover $\widehat{k}$ eigenvalues of A using the Rayleigh–Ritz projection. Finally, the five rightmost ones among the *retrieved* eigenvalues are chosen. It is again not clear how large $\widehat{k}$ should be to ensure that the five rightmost eigenvalues of A will be found, and $\widehat{k} = 100$ is chosen again. In addition, we fix $\sigma_1 = 0$ and try a sequence of values for σ_2, namely $\{1,2\}\bigcup\{10^{-\ell}\}_{\ell=1}^3 \bigcup\{2\times 10^{-\ell}\}_{\ell=1}^3 \bigcup\{5\times 10^{-\ell}\}_{\ell=1}^3$. The inner iteration is also solving a linear system with coefficient matrix A.

We choose the parameters of `eigs` as follows. The dimension of the Krylov subspace (or the parameter `p`) is set to 25 for $\mathcal{T} = \mathcal{E}$ and 200 for $\mathcal{T} = \mathcal{I}$, $\mathcal{S}_0$ or $\mathcal{C}_{\sigma_1,\sigma_2}$ The tolerance `tol` is set to 10^{-8}, the starting vector `v0` is the vector of all 1's, the maximal iteration count `maxit` is 50000, and default values are used for all other parameters of `eigs`.

We use the single-step Lyapunov inverse iteration proposed in [13] (see Appendix B for a brief review) to compute the smallest eigenvalue of $\mathcal{K}(A)$. The five rightmost eigenvalues of A can be recovered subsequently at little additional cost. This method mainly requires the solution to a large-scale Lyapunov equation, which is approximated by a modified version [12] of the *rational Krylov subspace method* (RKSM) [10].

The implementation of all five eigensolver and preconditioner combinations needs to be slightly modified for the *Obstacle*, *Cavity*, and *Plate* problems since $A = M^{-1}J$ is not explicitly formed. For instance, the matrix-vector product $A\mathbf{v}$ should be computed by solving $M\mathbf{y} = J\mathbf{v}$ for $\mathbf{y}$, and the solution $\mathbf{y}$ to $(A - \sigma I)\mathbf{y} = \mathbf{x}$ should be found by solving $(J - \sigma M)\mathbf{y} = M\mathbf{x}$.

All of the linear systems in our experiments are solved using the MATLAB backslash (a sparse Gaussian elimination). If multiple linear systems share a coefficient matrix, we prefactorize the matrix by the `lu` function. All experiments are done in MATLAB version R2016b on an iMac desktop equipped with a 3.2GHz Intel Core i5 processor, 32GB of 1867 MHz DDR3 RAM, and Mac OS X 10.11.6. The runtimes are measured by the `tic` and `toc` commands in MATLAB.

[3]The MATLAB command is `[V,D] = eigs(afun,length(A),5,'lm',opts)`, where `afun` is a routine that approximates $e^{hA}\mathbf{v}$ for a given vector $\mathbf{v}$. The input `opts` is a structure that specifies the values of some parameters.

4.3. Comparison of four methods for computing $e^{hA}\mathbf{v}$. We compare the following four methods: the standard Krylov method [36], the RD-rational method [27], the polynomial Leja method (Algorithm 1) [6, 8], and the rational Leja method (Algorithm 2) proposed in section 3.2. We fix $\mathbf{v}$ to be the vector of all ones in this subsection.

The maximal degree of Newton polynomials (L in section 3) is 100 for Algorithm 1 and 45 for Algorithm 2, whereas the maximal number of Arnoldi steps is 100 for both Krylov methods. The size of the substep (τ in section 3) of each method is found by Algorithm 3. For the computation of $e^{\tau A}\mathbf{v}$ in all four methods, we use the stopping criterion $\frac{\|\Delta\mathbf{w}\|}{\|\mathbf{w}\|} \leq 10^{-9}$ (*reltol* in section 3), where $\mathbf{w}$ is the current estimate to $e^{\tau A}\mathbf{v}$ and $\Delta\mathbf{w}$ is the difference between $\mathbf{w}$ and the previous estimate to $e^{\tau A}\mathbf{v}$. However, the accuracy of the final estimate to $e^{hA}\mathbf{v}$ produced by each method remains unknown. Consequently, although the same stopping criterion is used for the computation of $e^{\tau A}\mathbf{v}$, the estimates to $e^{hA}\mathbf{v}$ obtained by the four methods may *not* have the same order of accuracy. In addition, since both rational methods use a single pole, the coefficient matrix of the linear systems arising from either method is prefactorized.

The numbers of substeps and the runtimes required by all methods are reported in Table 2. In terms of runtime, the RD-rational method is clearly the winner: Out of the twelve test problems, it converges the fastest in nine problems and the second fastest in two problems; its advantage is especially pronounced for the second group of test problems whose rightmost few eigenvalues have large imaginary parts. Except when applied to the two *Tolosa* matrices, this method does not need substepping. The rational Leja method (Algorithm 2) is the second most efficient, converging the fastest in two problems and the second fastest in seven problems. In terms of storage, the two Leja methods are superior since they only require storing four vectors whereas both Krylov methods need to store as many as 100 vectors. Therefore, we recommend using the RD-rational method if storage is not a concern and the rational Leja method otherwise.

Knowing the rough shape of the spectrum of A also helps us make a decision as to which method should be used. As shown in Table 2, when the spectrum of A is "tall and skinny," such as for the two *Tolosa* matrices (see the left panel of Figure 2), the polynomial Leja method converges considerably faster than other methods. On the other hand, when the spectrum of A is "short and wide," such as for the *Crystal*, *Obstacle*, *Plate*, and *Cavity* matrices and their artificial counterparts, the two rational methods are significantly more efficient. We note that the matrices arising from spatial discretization of PDEs typically fall into this category. The width of the spectra of both *Aerofoil* matrices is only slightly larger than their height and accordingly, the RD-rational method is only slightly faster than the polynomial Leja method. By comparing the performance of the rational Leja method on the two groups of test problems, we also observe that this method deteriorates as the imaginary parts of a few rightmost eigenvalues grow. It can again be explained by Figure 3: the rational function $\Upsilon_{L,L}(x)$ approximates e^x poorly if both $\mathrm{Re}(x)$ and $|\mathrm{Im}(x)|$ are large; for such an x, a small substep size τ is necessary to make $\Upsilon_{L,L}(\tau x)$ a good approximation to $e^{\tau x}$. By contrast, the performance of the RD-rational method is largely insensitive to the growth in the imaginary parts of the few rightmost eigenvalues.

4.4. Comparison of eigensolver and preconditioner combinations. In Tables 3 to 7, we present the results of the following five pairs of eigensolver and preconditioner: `eigs` with no preconditioner, `eigs` with the shift-invert transformation, `eigs` with the Cayley transformation, Lyapunov inverse iteration with the Kronecker

TABLE 2
Performance comparison among four methods for approximating $e^{hA}\mathbf{v}$ *with* $h = 1$, $\mathbf{v} = [1, \ldots, 1]^T$, *and reltol* $= 10^{-9}$. *(The runtimes are measured in seconds.)*

	P-Leja(100)		Arnoldi(100)		R-Leja(45)		RD(100)	
Problem	substep count T	run time	substep count T	run time	substep count T	run time	substep count T	run rime
Tolosa	96	**0.315**	71	9.391	494	2.360	41	5.965
Aerofoil	20	1.223	10	3.607	36	9.265	1	**0.923**
Crystal	62	0.547	42	3.899	1	**0.030**	1	0.108
Obstacle	63	11.675	24	7.554	3	**0.397**	1	**0.397**
Plate	48	10.651	24	8.944	7	1.761	1	**0.659**
Cavity	5061	868.723	831	155.664	14	1.687	1	**0.648**
Tolosa (*a*)	119	**0.385**	72	9.609	494	3.278	41	6.024
Aerofoil (*a*)	24	1.530	11	3.742	77	29.438	1	**1.256**
Crystal (*a*)	94	0.891	49	7.347	21	1.015	1	**0.116**
Obstacle (*a*)	64	14.173	26	9.195	18	2.948	1	**0.409**
Plate (*a*)	64	14.451	27	9.921	20	4.779	1	**0.665**
Cavity (*a*)	5026	960.136	913	267.926	30	5.629	1	**0.691**

TABLE 3
The performance of eigs(lm) *applied to* e^{hA} *(Krylov subspace dim.* 25, tol $= 10^{-8}$*).*

Problem	Method	h	Substep count T	`eigs` iter	Run time	eig-values found	min eig-residual	max eig-residual
Tolosa	P-Leja(100)	0.5	48	5	28.14	all 6	$2.66\ 10^{-9}$	$5.15\ 10^{-7}$
Aerofoil	RD(100)	10	6	3	286.77	all 6	$2.71\ 10^{-7}$	$3.04\ 10^{-6}$
Crystal	R-Leja(45)	0.5	1	2	1.66	all 5	$1.46\ 10^{-11}$	$3.80\ 10^{-11}$
Obstacle	R-Leja(45)	2	5	1	13.75	all 5	$7.02\ 10^{-11}$	$8.38\ 10^{-9}$
Plate	RD(100)	1	1	4	49.90	all 6	$6.22\ 10^{-9}$	$1.97\ 10^{-8}$
Cavity	RD(100)	5	4	4	166.30	all 5	$1.03\ 10^{-7}$	$5.84\ 10^{-6}$
Tolosa (*a*)	P-Leja(100)	0.5	36	5	33.98	all 6	$1.63\ 10^{-9}$	$3.78\ 10^{-9}$
Aerofoil (*a*)	RD(100)	10	9	2	309.21	all 6	$2.59\ 10^{-9}$	$3.85\ 10^{-7}$
Crystal (*a*)	RD(100)	0.5	1	1	1.79	all 5	$3.23\ 10^{-9}$	$4.33\ 10^{-8}$
Obstacle (*a*)	RD(100)	2	2	2	19.28	all 6	$6.52\ 10^{-13}$	$3.10\ 10^{-9}$
Plate (*a*)	RD(100)	1	1	4	55.47	all 6	$1.75\ 10^{-11}$	$4.43\ 10^{-9}$
Cavity (*a*)	RD(100)	5	4	4	157.23	all 6	$7.92\ 10^{-13}$	$1.62\ 10^{-7}$

sum transformation, and `eigs` with the exponential transformation. The accuracy of a computed eigenpair $(\widehat{\mu}_j, \widehat{\mathbf{x}}_j)$ is measured by the relative residual norm $\frac{\|A\widehat{\mathbf{x}}_j - \widehat{\mu}_j \widehat{\mathbf{x}}_j\|_2}{\|A\widehat{\mathbf{x}}_j\|_2}$ or $\frac{\|J\widehat{\mathbf{x}}_j - \widehat{\mu}_j M\widehat{\mathbf{x}}_j\|_2}{\|J\widehat{\mathbf{x}}_j\|_2}$. The runtimes are again measured in seconds.

As we see from Tables 3–7, applying `eigs(lm)` to e^{hA} is clearly the most robust approach since it is the *only* one that can find all the target eigenvalues for every test problem. For each test problem, the method that computes $e^{A}\mathbf{v}$ the fastest (see section 4.3) is used to compute $e^{hA}\mathbf{v}$ in `eigs`. As h increases, fewer outer iterations (`eigs`) are necessary because of statement 2 of Corollary 2.2, but the inner iteration (computing $e^{hA}\mathbf{v}$) becomes more expensive. The values of h in Table 3 are chosen to strike a balance between the inner and outer iteration counts. More specifically, for each problem, we choose the smallest h from $\{0.5, 1, 2, 5, 10\}$ such that `eigs(lm)` for e^{hA} with `tol` $= 0.01$ converges in 25 Arnoldi steps before the first restart.

When `eigs(lr)` is applied to A (see Table 4), it looks for the target eigenvectors in a Krylov subspace of A. Therefore, this approach has difficulties finding the rightmost eigenvalues whose moduli are small. It explains why `eigs(lr)` tends to work better on the artificial problems: The moduli of the target eigenvalues of an artificial problem are always larger than those of the corresponding original problem. Overall, the convergence of `eigs(lr)` is very slow, even with a large Krylov subspace of dimension 200. Moreover, this method is unable to find the complete set of target eigenvalues for five problems.

TABLE 4
The performance of eigs(lr) *applied to A (Krylov subspace dim.* 200, tol $= 10^{-8}$*).*

Problem	`eigs` iter	Run time	Eigenvalues found	min eigen-residual	max eigen-residual
Tolosa	50000	6804.25	$\varnothing$	–	–
Aerofoil	2820	2722.50	all 6	$7.83\ 10^{-11}$	$9.11\ 10^{-9}$
Crystal	579	170.09	all 5	$6.83\ 10^{-12}$	$9.73\ 10^{-9}$
Obstacle	1123	691.68	all 5	$1.22\ 10^{-11}$	$7.11\ 10^{-10}$
Plate	6629	5707.81	$\{\mu_{7,8,13,14}\}$	–	–
Cavity	9020	7517.61	$\varnothing$	–	–
Tolosa (*a*)	50000	6595.70	$\{\mu_{\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{4}}\}$	$2.19\ 10^{-12}$	$4.85\ 10^{-12}$
Aerofoil (*a*)	712	752.61	all 6	$3.42\ 10^{-15}$	$9.87\ 10^{-9}$
Crystal (*a*)	1594	497.21	all 5	$1.27\ 10^{-13}$	$9.89\ 10^{-9}$
Obstacle (*a*)	269	217.88	all 6	$2.54\ 10^{-15}$	$1.08\ 10^{-9}$
Plate (*a*)	342	440.04	all 6	$2.15\ 10^{-14}$	$1.10\ 10^{-8}$
Cavity (*a*)	1487	1386.43	$\{\mu_{\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{4}}\}$	$1.56\ 10^{-12}$	$2.11\ 10^{-12}$

TABLE 5
The performance of eigs(sm) *applied to A (Krylov subspace dim.* 200, tol $= 10^{-8}$*).*

Problem	`eigs` iter	Run time	Eigenvalues found	min eigen-residual	max eigen-residual
Tolosa	4	0.48	$\{\mu_{5,6,9,10,13,14}\}$	$1.93\ 10^{-13}$	$1.93\ 10^{-13}$
Aerofoil	19	20.44	all 6	$2.01\ 10^{-6}$	$6.03\ 10^{-6}$
Crystal	1	0.76	$\{\mu_{97,98,99,100,101}\}$	–	–
Obstacle	3	2.04	all 5	$1.46\ 10^{-11}$	$1.72\ 10^{-11}$
Plate	2	1.77	$\{\mu_{47}\}$	–	–
Cavity	3	2.12	$\{\mu_{\mathbf{3},\mathbf{4},\mathbf{5},8,11,12}\}$	$2.17\ 10^{-11}$	$3.36\ 10^{-10}$
Tolosa (*a*)	4	0.48	$\{\mu_{9,10,13,14,17,18}\}$	–	–
Aerofoil (*a*)	19	21.53	$\{\mu_{\mathbf{5},\mathbf{6},7,8,9,10}\}$	$6.51\ 10^{-6}$	$6.51\ 10^{-6}$
Crystal (*a*)	1	0.75	$\{\mu_{101,102,103,104,105}\}$	–	–
Obstacle (*a*)	3	2.01	$\{\mu_{\mathbf{5},\mathbf{6},7,8,9,10}\}$	$2.73\ 10^{-11}$	$2.73\ 10^{-11}$
Plate (*a*)	2	1.75	$\{\mu_{51}\}$	–	–
Cavity (*a*)	3	2.11	$\{\mu_{7,8,9,12,15,16}\}$	–	–

When `eigs(sm)` is applied to A (see Table 5), it searches for the target eigenvectors in a Krylov subspace of A^{-1} instead and may fail to find the rightmost eigenvalues whose moduli are large. The performance of this approach is worse when it is applied to the artificial matrices for the same reason that `eigs(lr)` works better on them; in fact, it is not able to find the complete set of target eigenvalues for any artificial matrix. Applying `eigs(lm)` to $\mathcal{C}_{0,\sigma_2}(A)$ produces similar results (see Table 6). Recall from section 4.2 that various values of σ_2 are considered. In Table 6, we only report the best results and the values of σ_2 used to produce them. Both `eigs(sm)` applied to A and `eigs(lm)` applied to $\mathcal{C}_{0,\sigma_2}(A)$ converge rapidly but often to the wrong eigenvalues.

The numerical results of Lyapunov inverse iteration (see Algorithm 4) are shown in Table 7. This method is rather restrictive in that it may not converge to the rightmost eigenvalue of A if A is not real or if the rightmost eigenvalue of A has positive real part, i.e., if one of the assumptions of Theorem B.1 is violated. The *Crystal* matrix, *Plate* matrix, and their artificial counterparts all have eigenvalues with positive real parts; indeed, when A is one of them, the real part of the rightmost eigenvalue of

TABLE 6
The performance of eigs(lm) *applied to* $\mathcal{C}_{0,\sigma_2}(A)$ *(Krylov subspace dim.* 200, tol $= 10^{-8}$*).*

Problem	σ_2	`eigs` iter	Run time	Eigenvalues found	min eigen-residual	max eigen-residual
Tolosa	$[0.002, 0.01]$	1	0.24	$\{\mu_{\mathbf{5},\mathbf{6},9,10,13,14}\}$	$7.31\ 10^{-8}$	$7.31\ 10^{-8}$
Aerofoil	$[1.7, 2]$	24	25.09	all 6	$4.50\ 10^{-6}$	$5.34\ 10^{-6}$
Crystal	$[0.001, 2]$	1	0.85	$\varnothing$	–	–
Obstacle	$[0.001, 2]$	3	2.23	$\{\mu_{\mathbf{3},\mathbf{4},\mathbf{5},\mathbf{6},9,10}\}$	$1.83\ 10^{-11}$	$2.91\ 10^{-11}$
Plate	$[0.001, 2]$	3	2.30	$\{\mu_{47}\}$	–	–
Cavity	$[1.4, 2]$	4	2.71	$\{\mu_{\mathbf{3},\mathbf{4},\mathbf{5},8,11,12}\}$	$2.96\ 10^{-11}$	$3.09\ 10^{-10}$
Tolosa (*a*)	$[0.002, 0.01]$	1	0.25	$\{\mu_{9,10,13,14,17,18}\}$	–	–
Aerofoil (*a*)	$[1.7, 2]$	23	25.02	$\{\mu_{\mathbf{5},\mathbf{6},7,8,9,10}\}$	$1.27\ 10^{-6}$	$1.27\ 10^{-6}$
Crystal (*a*)	$[0.001, 2]$	1	0.85	$\varnothing$	–	–
Obstacle (*a*)	$[0.001, 2]$	7	3.57	$\{\mu_{7,8,9,10,13,14}\}$	–	–
Plate (*a*)	$[0.001, 2]$	3	2.25	$\{\mu_{51}\}$	–	–
Cavity (*a*)	$[1.4, 2]$	4	2.67	$\{\mu_{7,8,9,12,15,16}\}$	–	–

TABLE 7
The performance of Lyapunov inverse iteration (Algorithm 4).

Problem	*lyaptol*	*eigtol*	Run time	Subspace dim.	Eigenvalues found	min eigen-residual	max eigen-residual
Tolosa	10^{-3}	10^{-2}	9.40	180	all 6	$5.06\ 10^{-11}$	$1.26\ 10^{-9}$
Aerofoil	10^{-8}	10^{-7}	41.40	120	all 6	$1.02\ 10^{-11}$	$2.55\ 10^{-9}$
Crystal	10^{-6}	10^{-5}	> 10000	–	–	–	–
Obstacle	10^{-7}	10^{-6}	8.39	60	all 5	$4.51\ 10^{-12}$	$1.99\ 10^{-8}$
Plate	10^{-6}	10^{-5}	23.40	100	$\{\mu_{43,44,47}\}$	$1.07\ 10^{-9}$	$7.28\ 10^{-9}$
Cavity	10^{-8}	10^{-7}	132.24	280	all 6	$7.84\ 10^{-11}$	$6.60\ 10^{-7}$
Tolosa (*a*)	10^{-3}	10^{-2}	9.45	180	$\{\mu_{5:10}\}$	$5.95\ 10^{-11}$	$9.93\ 10^{-10}$
Aerofoil (*a*)	10^{-8}	10^{-7}	40.84	100	$\{\mu_{5:10}\}$	$1.61\ 10^{-9}$	$2.04\ 10^{-5}$
Crystal (*a*)	10^{-6}	10^{-5}	> 10000	–	–	–	–
Obstacle (*a*)	10^{-7}	10^{-6}	512.79	790	all 6	$2.49\ 10^{-14}$	$1.37\ 10^{-10}$
Plate (*a*)	10^{-7}	10^{-6}	231.47	420	$\{\mu_{47,48}\}$	$2.54\ 10^{-11}$	$2.95\ 10^{-3}$
Cavity (*a*)	10^{-8}	10^{-7}	1217.84	1000	all 6	$1.81\ 10^{-13}$	$1.17\ 10^{-10}$

A is not the smallest eigenvalue of $\mathcal{K}(A)$. In addition, Lyapunov solver encounters difficulties when solving (25) if A is the *Crystal* matrix or its artificial counterpart because both matrices are ill-conditioned. Even when all the assumptions of Theorem B.1 are satisfied, Lyapunov inverse iteration may converge to the wrong eigenvalues or suffer from slow convergence if the target eigenvalues have very large imaginary parts. This can be seen by comparing its performance on the two groups of test problems. For instance, in theory, it should work for the *Tolosa* matrix, *Aerofoil* matrix, and their artificial counterparts; in the numerical experiments, however, it fails to find most of the target eigenvalues for the two artificial matrices, indicating that the Lyapunov equation (25) is not solved accurately enough. Moreover, although this approach is able to find all of the target eigenvalues for the artificial *Obstacle* and *Cavity* matrices, it requires significantly longer runtimes and larger Krylov subspaces than it does when applied to the original matrices. As can be seen in Table 3, thanks to the RD-rational method, the runtime required by applying `eigs(lm)` to e^{hA} is more or less the same whether A is an original matrix or its artificial counterpart.

4.5. Relaxing the substep size for the rational Leja method. Our main goal is to reliably compute the rightmost eigenvalues of A reasonably accurately, which can be guaranteed as long as $e^{hA}\mathbf{v}$ is approximated to a commensurate level of accuracy. In this section, we show that using a larger substep size τ in the computation of $e^{hA}\mathbf{v}$ can reduce the runtimes for a few problems recorded in Table 3 without significantly compromising the accuracies of the target eigenpairs.

In fact, as shown in [38, sect. 11], the difference between the true and computable eigenresiduals of the Ritz pairs extracted from an inexact Krylov subspace is proportional to the errors of the matrix-vector multiplications performed while constructing the Krylov subspace. In our new approach, as long as the accuracy of $e^{hA}\mathbf{v}$ is *comparable* to the computable residuals (`tol` $= 10^{-8}$ used to terminate `eigs`), `eigs` with inexact $e^{hA}\mathbf{v}$ should deliver eigenpairs that are as accurate as those found by `eigs` with exact $e^{hA}\mathbf{v}$.

Table 8
The performance of eigs(lm) *applied to* e^{hA} *(Krylov subspace dim.* 25, tol $= 10^{-8}$*, and* $e^{hA}\mathbf{v}$ *is computed using aggressive substepping).*

Problem	Method	h	T_{agrs}	T_0/T_{agrs}	`eigs` iter	Run time	eig-vals found	min eig-residual	max eig-residual
Tolosa	P-Leja(100)	0.5	30	1.6	4	15.90	all 6	$3.20\ 10^{-9}$	$3.70\ 10^{-7}$
Aerofoil	R-Leja(45)	10	2	180	4	46.47	all 6	$3.61\ 10^{-10}$	$1.67\ 10^{-7}$
Obstacle	R-Leja(45)	2	3	1.6667	1	9.30	all 5	$7.62\ 10^{-10}$	$1.95\ 10^{-8}$
Cavity	R-Leja(45)	5	9	7.4444	4	107.35	all 5	$3.30\ 10^{-9}$	$9.98\ 10^{-7}$

In light of this convergence result, we propose adopting the following aggressive substepping strategy for computing $e^{hA}\mathbf{v}$. As in sections 4.3 and 4.4, we first find a substep size τ_0 by applying Algorithm 3 with $reltol = 10^{-9}$ and compute an approximation $\mathbf{u}_0 \approx e^{hA}\mathbf{v}$ using $T_0 = \lceil h/\tau_0 \rceil$ substeps. Then we keep decreasing the number of substeps and recomputing $e^{hA}\mathbf{v}$ until we identify the smallest number of substeps, T_{agrs}, such that $\frac{\|\mathbf{u}_{\text{agrs}}-\mathbf{u}_0\|}{\|\mathbf{u}_0\|} \leq 10^{-6}$ is still satisfied, where $\mathbf{u}_{\text{agrs}} \approx e^{hA}\mathbf{v}$ is computed using T_{agrs} substeps. Once such a new substep size has been determined, it is used in all of the subsequent approximations to $e^{hA}\mathbf{v}$, with the original stopping criterion disabled.

In Table 8, we show the results of applying `eigs(lm)` with aggressive substepping to e^{hA} for some original test problems. For the *Tolosa* matrix, the polynomial Leja method with aggressive substepping is used to compute $e^{hA}\mathbf{v}$; for the other problems, the rational Leja method with substepping is chosen. (Note that the RD-rational method was used in Table 3 for the *Aerofoil* and *Cavity* matrices.) The *Crystal* and *Plate* matrices are not considered because the substep count $T_0 = 1$ in Table 3 cannot be further reduced. By comparing the results in Tables 3 and 8, we see that aggressive substepping reduces the runtimes without degrading the accuracy of the computed eigenpairs; for instance, it is able to reduce the runtime required by the *Aerofoil* matrix from 289 to 46 seconds. However, even a mild reduction in the number of substeps of the RD-rational method *does* produce much less accurate eigenpairs.

The rational Leja method with aggressive substepping does not work on the artificial problems. This can be explained by examining how well $\Upsilon_{L,L}(x)$ approximates e^x at the eigenvalues of a matrix. Consider the two *Aerofoil* matrices. In Figure 4, we plot the rightmost part of the spectrum of τA, where $\tau = 2$ and A is the original *Aerofoil* matrix (left panel) or its artificial counterpart (right panel); the contour plot of $\log_{10}|e^x - \Upsilon_{L,L}(x)|$ is shown in both panels. We see that the eigenvalues of τA in the left panel lie in a region where the error $|e^x - \Upsilon_{L,L}(x)|$ is small; however, the four rightmost eigenvalues of τA (marked by hexagons) in the right panel are in a region where the approximation error is quite large. Therefore, although a substep size $\tau = 2$ can be used for the original *Aerofoil* matrix, a much smaller τ must be used for its artificial counterpart so that all of the eigenvalues of τA lie in the region of accurate approximation. For all of the artificial matrices except the artificial *Tolosa* matrix, the RD-rational method is still the most efficient.

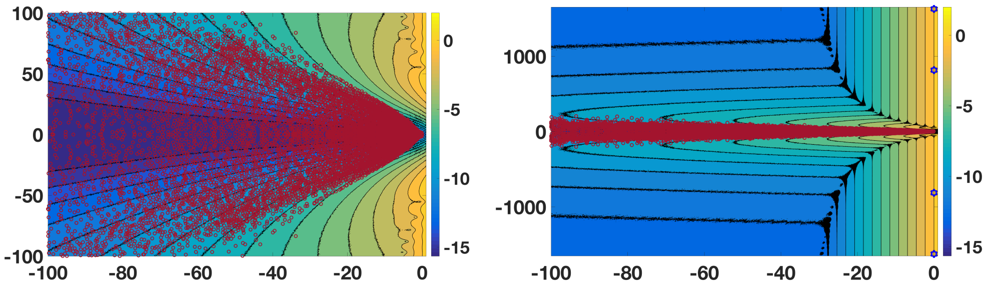

FIG. 4. *Left: The eigenvalues (red circles) of τA where $\tau = 2$ and A is the original Aerofoil. Right: The eigenvalues (red circles and blue hexagons) of τA where A is the artificial Aerofoil. The contour plot of $\log_{10}|e^x - \Upsilon_{L,L}(x)|$ is shown in both panels ($a = 50$, $L = 45$). (Figure is in color online.)*

Finally, we compare the runtimes and storage of the three *most robust* approaches considered in section 4.4, namely, `eigs(lm)` applied to e^{hA}, Lyapunov inverse iteration applied to $\mathcal{K}(A)$, and `eigs(lr)` applied to A. As can be seen from Tables 4 and 7, the latter two approaches are not always successful at finding all of the target eigenvalues. Since robustness is our primary concern, we only consider the problems for which all three methods succeed. The runtimes and storage (measured by the number of vectors in $\mathbb{R}^n$) of the three methods are reported in Table 9. The rational Leja method with aggressive substepping is used for the *Aerofoil* and *Obstacle* matrices, and the RD-rational method is used for the artificial *Obstacle* matrix. The storage required by `eig(lm)` with $\mathcal{T} = e^{hA}$ is 29 or 125 vectors: 25 Arnoldi vectors built by `eigs`, and 4 (Leja) or 100 (RD-rational) additional vectors for the computation of $e^{hA}\mathbf{v}$. In Lyapunov inverse iteration, it is necessary to store the basis vectors of a Krylov subspace built for solving (25), and the dimension of this subspace varies drastically from problem to problem. When applying `eigs(lr)` with $\mathcal{T} = I$, we have to store 200 basis vectors of a Krylov subspace constructed by `eigs`. Overall, applying `eigs(lm)` with e^{hA} requires the least storage and its runtime also compares favorably with that of Lyapunov inverse iteration; its advantage is especially pronounced in the artificial *Obstacle* problem.

TABLE 9
Performance comparison among eigs(lr) *with $\mathcal{T} = I$, Lyapunov inverse iteration, and* eigs(lm) *with $\mathcal{T} = e^{hA}$ on problems that can be solved successfully by all three methods.*

	`eigs(lr)` $\mathcal{T} = I$		Lyapunov		`eigs(lm)` $\mathcal{T} = e^{hA}$	
Problem	storage	runtime	storage	runtime	storage	runtime
Aerofoil	200	2723	120	41	$4 + 25$	46
Obstacle	200	692	60	8	$4 + 25$	9
Obstacle (*a*)	200	218	790	513	$100 + 25$	16

5. Applications in pseudospectral analysis. Compared to the rightmost eigenvalue, the rightmost point of the ε-pseudospectrum or that of the real structured ε-pseudospectrum is a more robust measurement of stability. We continue to denote the two pseudospectra of A by $\Lambda_\varepsilon(A)$ and $\Lambda_\varepsilon^{\mathbb{R}}(A)$, respectively. A number of fast algorithms [18, 19, 31] have been developed to find the rightmost points of $\Lambda_\varepsilon(A)$ and $\Lambda_\varepsilon^{\mathbb{R}}(A)$ for a large, sparse A. They iteratively produce a sequence of perturbation matrices $\{E_\ell\}_{\ell=0}^m$ of unit norm and low rank such that the rightmost eigenvalues of $\{A+\varepsilon E_\ell\}_{\ell=0}^m$ often converge to the rightmost point of $\Lambda_\varepsilon(A)$ or $\Lambda_\varepsilon^{\mathbb{R}}(A)$. In all three algorithms, the construction of the next perturbation matrix $E_{\ell+1}$ relies on finding the rightmost eigenvalue of $A+\varepsilon E_\ell$ and its corresponding eigenvector. As will be shown in this section, the exponential transformation is a robust preconditioner for $A+\varepsilon E_\ell$ as well.

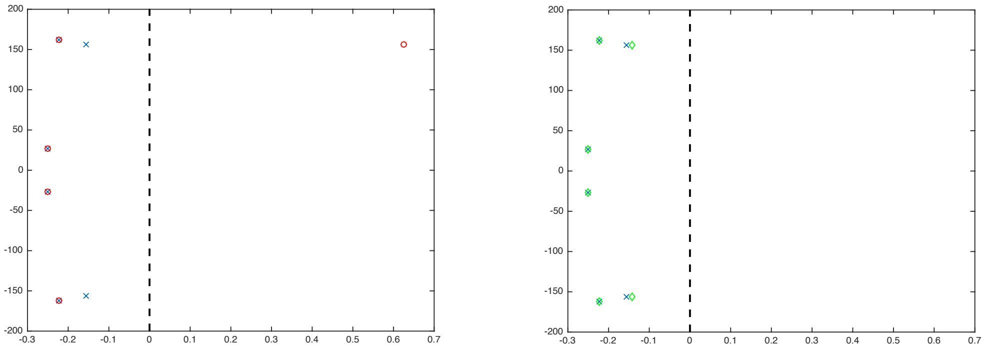

Fig. 5. *Eigenvalues of A and $A^\dagger$ in the Tolosa case. The circles in the left plot are the five rightmost eigenvalues of $A+0.01\cdot\mathbf{y}_1\mathbf{x}_1^*$, and the diamonds in the right plot are the six rightmost eigenvalues of $A+0.01\cdot U_1V_1^T$. In both plots, the crosses represent the six rightmost eigenvalues of A, and the dashed line is the imaginary axis.*

We will focus on the eigenvalue problems arising from the methods of [19, 31], both of which employ a "greedy" approach of selecting the perturbation matrices. The method of [19] is for finding the rightmost point of $\Lambda_\varepsilon(A)$. It chooses the initial perturbation E_0 to be the rank-1, norm-1 matrix $\mathbf{y}_1\mathbf{x}_1^*$, where $\mathbf{x}_1,\mathbf{y}_1$ are the right and left eigenvectors associated with a rightmost eigenvalue μ_1 of A and are scaled such that $\|\mathbf{x}_1\|=\|\mathbf{y}_1\|=1$, $\mathbf{y}_1^*\mathbf{x}_1>0$. The method of [31] aims at computing the rightmost point of $\Lambda_\varepsilon^{\mathbb{R}}(A)$ and therefore has to restrict all of the perturbation matrices to be real. It chooses the initial perturbation matrix E_0 to be $U_1V_1^T$ with $U_1\Sigma V_1^T$ being the truncated singular value decomposition of $\mathrm{Re}(\mathbf{y}_1)\mathrm{Re}(\mathbf{x}_1^T)+\mathrm{Im}(\mathbf{y}_1)\mathrm{Im}(\mathbf{x}_1^T)$, which is real and has rank at most two. Obviously, if μ_1 is real, $\mathbf{y}_1\mathbf{x}_1^*$ and $U_1V_1^T$ coincide. All subsequent perturbation matrices in both algorithms are chosen in a similar fashion. Let $A^\dagger$ denote $A+\varepsilon\mathbf{y}_1\mathbf{x}_1^*$ or $A+\varepsilon U_1V_1^T$. In Figure 5, we display a few rightmost eigenvalues of both $A^\dagger$ (with $\varepsilon=0.01$) associated with the *Tolosa* matrix. As can be seen in Figure 5, both perturbation matrices advance μ_1 (the rightmost eigenvalue of A with positive imaginary part) to the right; in particular, the perturbation $0.01\mathbf{y}_1\mathbf{x}_1^*$ moves μ_1 to the right of the imaginary axis. Interestingly, the same matrix pushes $\overline{\mu_1}$ to the left instead (the perturbed $\overline{\mu_1}$ is not shown in the left panel). Other eigenvalues of A are hardly affected by either perturbation. The eigenvalues of $A+0.01\mathbf{y}_1\mathbf{x}_1^*$ shown in the left panel do not appear in conjugate pairs since this matrix is not real.

We briefly discuss some implementation details of Algorithms 1 (polynomial Leja) and 2 (rational Leja) when applied to compute $e^{hA^\dagger}\mathbf{v}$. It is not recommended to explicitly form the dense matrix $A^\dagger$; instead, we should take full advantage of its special structure, i.e., $A^\dagger$ is the sum of a sparse matrix A and a low-rank matrix $\varepsilon\mathbf{y}_1\mathbf{x}_1^*$ or $\varepsilon U_1V_1^T$. As the perturbations only seem to affect a few eigenvalues of A and leave other eigenvalues unchanged, we use the spectral bounds α and β of A in

Algorithm 1. If $A^\dagger = A + \varepsilon \mathbf{y}_1 \mathbf{x}_1^*$, then step 13 of Algorithm 1 needs to be changed to

$$\mathbf{r} \leftarrow \left(\frac{2}{\mathfrak{d}} A - \omega_{\ell-1} I \right) \mathbf{r} + \frac{2\varepsilon}{\mathfrak{d}} \mathbf{y}_1 (\mathbf{x}_1^* \mathbf{r});$$

step 6 of Algorithm 2 can be replaced by the following lines:

$$\begin{aligned} &\text{solve } (aI - \tau A)\mathbf{y}' = (aI + \tau A)\mathbf{r} + \tau\varepsilon \mathbf{y}_1(\mathbf{x}_1^* \mathbf{r}) \text{ for } \mathbf{y}', \\ &\text{solve } (aI - \tau A)\mathbf{y}'' = -\tau\varepsilon \mathbf{y}_1 \text{ for } \mathbf{y}'', \\ &\mathbf{y} \leftarrow \mathbf{y}' - \mathbf{y}''(1 + \mathbf{x}_1^* \mathbf{y}'')^{-1}(\mathbf{x}_1^* \mathbf{y}'), \end{aligned}$$

which amount to solving $(aI - \tau A^\dagger)\mathbf{y} = (aI + \tau A^\dagger)\mathbf{r}$ for $\mathbf{y}$ by the Sherman–Morrison–Woodbury formula. Note that the solve for $\mathbf{y}''$ above only needs to be done once throughout Algorithm 2. When $A^\dagger = A + \varepsilon U_1 V_1^T$ or the RD-rational method is used to compute $e^{hA^\dagger}\mathbf{v}$, a similar set of revisions can be made without difficulty.

TABLE 10
The performance of eigs(lm) *applied to* $e^{hA^\dagger}$ *where* $A^\dagger = A + 0.01 \cdot \mathbf{y}_1 \mathbf{x}_1^*$ *(aggressive substepping is used when a Leja method is applied to compute* $e^{hA^\dagger}\mathbf{v}$*).*

Problem	Method	`eigs` iter	Run time	Eigenvalues found	min eigen-residual	max eigen-residual
Tolosa	P-Leja(100)	4	20.94	all 5	$1.16\ 10^{-13}$	$8.30\ 10^{-7}$
Obstacle	R-Leja(45)	1	12.69	all 5	$7.77\ 10^{-10}$	$1.59\ 10^{-8}$
Cavity	R-Leja(45)	4	119.36	all 5	$2.50\ 10^{-9}$	$9.06\ 10^{-7}$
Plate	RD(100)	3	58.03	all 5	$6.20\ 10^{-9}$	$1.79\ 10^{-7}$
Crystal	R-Leja(45)	2	2.91	all 5	$6.88\ 10^{-14}$	$6.90\ 10^{-8}$
Aerofoil	R-Leja(45)	2	53.54	all 5	$1.33\ 10^{-12}$	$2.67\ 10^{-7}$

TABLE 11
The performance of eigs(lm) *applied to* $e^{hA^\dagger}$ *where* $A^\dagger = A + 0.01 \cdot U_1 V_1^T$ *(aggressive substepping is used when a Leja method is applied to compute* $e^{hA^\dagger}\mathbf{v}$*).*

Problem	Method	`eigs` iter	Run time	Eigenvalues found	min eigen-residual	max eigen-residual
Tolosa	P-Leja(100)	4	17.28	all 6	$3.08\ 10^{-10}$	$7.49\ 10^{-8}$
Obstacle	R-Leja(45)	1	10.22	all 5	$7.11\ 10^{-10}$	$1.46\ 10^{-8}$
Cavity	R-Leja(45)	4	112.29	all 5	$2.87\ 10^{-9}$	$9.39\ 10^{-6}$
Plate	RD(100)	5	80.83	all 6	$1.85\ 10^{-9}$	$3.98\ 10^{-8}$
Aerofoil	R-Leja(45)	2	49.40	all 5	$7.23\ 10^{-13}$	$1.19\ 10^{-7}$

The performance of `eigs(lm)` applied to $e^{hA^\dagger}$ is shown in Tables 10 and 11. This method again finds all the target eigenvalues of $A^\dagger$ for every test problem; for brevity, we only show the numerical results for the first group of test problems. The *Crystal* problem appears in Table 10 but not in Table 11 since its rightmost eigenvalue is real which implies that $\mathbf{y}_1 \mathbf{x}^* = U_1 V_1^T$. The same values of h listed in Table 3 are used here.

We emphasize that the exponential transformation is robust for the *entire* sequence of matrices $\{A\} \bigcup \{A + \varepsilon E_\ell\}_{\ell=0}^m$ arising from the methods of [18, 19, 31], whereas other preconditioners considered in section 4 may only work for a subset of them. For example, in the *Tolosa* case, Lyapunov inverse iteration can be applied to reliably calculate the rightmost eigenvalue of A (see Table 7). However, it is not applicable to $A^\dagger = A + 0.01 \cdot \mathbf{y}_1 \mathbf{x}_1^*$ since the real part of the rightmost eigenvalue of

$A^\dagger$ is not the smallest eigenvalue of $\mathcal{K}(A^\dagger) = \frac{1}{2}(A^\dagger \otimes I + I \otimes A^\dagger)$. The *Aerofoil* problem demonstrates how unreliable the shift-invert transformation is in pseudospectral analysis. It has been shown in Table 5 that the rightmost six eigenvalues of A can be found by computing the 50 eigenvalues of A with smallest moduli. However, as shown in Figure 6, even if we compute the 500 eigenvalues of $A^\dagger = A + 0.01 \cdot U_1 V_1^T$ with smallest moduli, we still miss its rightmost eigenvalue.

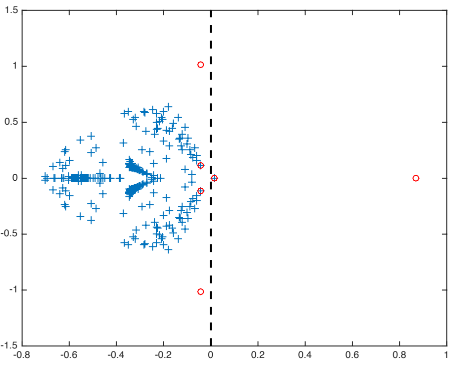

FIG. 6. *Eigenvalues of* $A^\dagger = A + 0.01 \cdot U_1 V_1^T$ *in the Aerofoil case. The circles are the six rightmost eigenvalues of* $A^\dagger$*, and the plus signs are the* 500 *eigenvalues of* $A^\dagger$ *with smallest moduli. The dashed line is the imaginary axis.*

6. Conclusions.

Our main contributions include a robust preconditioner for computing a few rightmost eigenvalues of a large matrix and an efficient method for approximating the action of the matrix exponential. Existing preconditioners for the rightmost eigenvalues are either very sensitive to the choice of parameters or only applicable to a limited subset of matrices. We propose using the exponential transformation since the target eigenvalues of A correspond to the largest ones of e^{hA} $(h > 0)$, which are easy to find by an iterative eigensolver. The preconditioner itself does not require selecting any parameter and can be applied to any square, complex matrix.

Computing the largest eigenvalues of e^{hA} entails computing a sequence of matrix-vector products $e^{hA}\mathbf{v}$, which is quite challenging for matrices with a spectrum of large capacity. Based on an existing rational approximation of e^x over $(-\infty, 0]$ [42, Chap. 25], we develop a new rational Leja method for computing $e^{hA}\mathbf{v}$. Compared to the RD-rational method [27], it converges slower in most cases but requires considerably less memory. Both rational methods have only one (repeated) pole, allowing us to prefactorize the coefficient matrix or compute a preconditioner for it once and for all. Compared to polynomial Leja [6, 8], our method converges faster overall and requires the same memory. We also demonstrate the robustness of the proposed preconditioner at solving the eigenvalue problems arising from pseudospectral analysis [18, 19, 31].

The MATLAB codes and matrices that we use in the numerical experiments are available from https://www.mathworks.com/matlabcentral/fileexchange/67267-a-solver-for-the-rightmost-eigenvalues.

Appendix A. Computing $e^{\tau A}\mathbf{v}$ by the polynomial Leja method.

Suppose that the real and imaginary parts of the eigenvalues of A are contained in the intervals $[\alpha, \nu]$ and $[-\beta, \beta]$, respectively, where $\alpha < \nu$ and $\beta \geq 0$. We then find the ellipse with smallest capacity,[4] denoted by $\mathscr{E}_{\rm sc}$, among all the ones enclosing the rectangle $[\alpha,\nu]\times[-\beta,\beta]$ on the complex plane. The domain K of Leja points is chosen to be the interval between the two foci of $\mathscr{E}_{\rm sc}$. The rationale behind this choice can be found in [8] and the references therein. To avoid overflow and underflow, [30] suggests scaling K to have capacity 1. In [6, 8], this is achieved by a change of variable. Assume that $(\mathfrak{c},0)$ is the center of $\mathscr{E}_{\rm sc}$ and $\mathfrak{d}>0$ is half of the distance between its foci. Then K is either the horizontal interval $\{(s,0)|\mathfrak{c}-\mathfrak{d}\le s\le \mathfrak{c}+\mathfrak{d}\}$ or the vertical interval $\{(\mathfrak{c},t)|-\mathfrak{d}\le t\le\mathfrak{d}\}$. Assume K is the former and introduce $\xi\in[-2,2]$ that satisfies $x=\mathfrak{c}+\frac{\mathfrak{d}}{2}\xi\in K$. Then $e^{\tau x}$ can be approximated by the Newton interpolation formula of $g(\xi)=e^{\tau\left(\mathfrak{c}+\frac{\mathfrak{d}}{2}\xi\right)}$ on Leja points $\{\xi_j\}_{j=0}^{\ell}\subset[-2,2]$, i.e.,

$$e^{\tau x}=g(\xi)\approx\widehat{p}_\ell(\xi)=\textstyle\sum_{j=0}^{\ell}\widehat{d}_j\widehat{r}_j(\xi),\qquad \xi\in[-2,2], \tag{20}$$

where

$$\widehat{d}_0=g[\xi_0]=g(\xi_0),\quad \widehat{d}_1=\frac{g[\xi_1]-g[\xi_0]}{\xi_1-\xi_0},\quad \widehat{d}_j=\frac{g[\xi_1,\ldots,\xi_j]-g[\xi_0,\ldots,\xi_{j-1}]}{\xi_j-\xi_0}\text{ for } j\ge 2, \tag{21}$$

$$\widehat{r}_0(\xi)=1,\quad \widehat{r}_j(\xi)=(\xi-\xi_0)\cdots(\xi-\xi_{j-1})=\textstyle\prod_{i=0}^{j-1}(\xi-\xi_i)\text{ for } j\ge 1. \tag{22}$$

The formula (20) is over the domain $[-2,2]$, which has capacity 1 as desired.

Since $\xi=\frac{2}{\mathfrak{d}}(x-\mathfrak{c})$, by replacing ξ with $\frac{2}{\mathfrak{d}}(A-\mathfrak{c}I)$ in (20)–(22) and right-multiplying both sides of them by $\mathbf{v}$, we obtain the approximant

$$e^{\tau A}\mathbf{v}\approx\widehat{p}_\ell\left(\frac{2}{\mathfrak{d}}(A-\mathfrak{c}I)\right)\mathbf{v}=\textstyle\sum_{j=0}^{\ell}\widehat{d}_j\widehat{r}_j\left(\frac{2}{\mathfrak{d}}(A-\mathfrak{c}I)\right)\mathbf{v}$$

which can be updated via

$$\begin{aligned}\widehat{r}_{\ell+1}\left(\frac{2}{\mathfrak{d}}(A-\mathfrak{c}I)\right)\mathbf{v}&=\left(\frac{2}{\mathfrak{d}}A-\left(\frac{2\mathfrak{c}}{\mathfrak{d}}+\xi_\ell\right)I\right)\widehat{r}_\ell\left(\frac{2}{\mathfrak{d}}(A-\mathfrak{c}I)\right)\mathbf{v}\text{ for }\ell\ge 0,\\ \widehat{p}_{\ell+1}\left(\frac{2}{\mathfrak{d}}(A-\mathfrak{c}I)\right)\mathbf{v}&=\widehat{p}_\ell\left(\frac{2}{\mathfrak{d}}(A-\mathfrak{c}I)\right)\mathbf{v}+\widehat{d}_{\ell+1}\widehat{r}_{\ell+1}\left(\frac{2}{\mathfrak{d}}(A-\mathfrak{c}I)\right)\mathbf{v}.\end{aligned} \tag{23}$$

Equation (23) indicates that each update simply entails one matrix-vector product $\left(\frac{2}{\mathfrak{d}}A-\left(\frac{2\mathfrak{c}}{\mathfrak{d}}+\xi_\ell\right)I\right)\mathbf{r}$. In the case where K is the vertical interval $\{(\mathfrak{c},t)|-\mathfrak{d}\le t\le\mathfrak{d}\}$, let $\zeta\in\mathfrak{i}[-2,2]$ be such that $x=\mathfrak{c}+\frac{\mathfrak{d}}{2}\zeta\in K$ and compute a Newton polynomial in ζ similar to (20). The domain $\mathfrak{i}[-2,2]$ of this polynomial also has capacity 1. The rest of the derivation is very similar.

The Leja points $\{\xi_j\}_{j=0}^{L}$ and $\{\zeta_j\}_{j=0}^{L}$ can be precomputed because they are for problem-independent reference domains. In [6, 8], whenever an imaginary Leja point ζ is chosen, its conjugate $\overline{\zeta}$ will be included in the sequence $\{\zeta_j\}_{j=0}^{L}$ as well. Thus, Algorithm 1 can be implemented in real arithmetic even when K is vertical (see [6]). How to compute Leja points and divided differences is beyond the scope of this paper, and we refer interested readers to [2] and [5].

Appendix B. Review of the Lyapunov inverse iteration.

This method is developed based on the following theorem (see also Theorems 2.1 and 2.2 of [13]).

[4]The capacity of an ellipse is the sum of its two semiaxes; see Remark 2.1 and (16) in [3] for how to find the ellipse with smallest capacity given α, ν, and β. Equation (1.1) in [30] gives the definition of capacity for a general compact set in $\mathbb{C}$. Also see [42, pp. 92–94].

THEOREM B.1. *Assume that all of the eigenvalues of $A \in \mathbb{R}^{n\times n}$ have nonpositive real parts and A also has a complete set of eigenvectors. Let μ_1 denote a rightmost eigenvalue of A. Then the eigenvalue of $\mathcal{K}(A) = \frac{1}{2}(A \otimes I + I \otimes A)$ with smallest modulus is $\lambda_1 = \mathrm{Re}(\mu_1)$. Furthermore, assume that μ_1 is a simple eigenvalue of A, and let $\mathbf{x}_1$ be an eigenvector associated with μ_1.*

1. *If μ_1 is real and the only rightmost eigenvalue of A, then the eigenvector of $\mathcal{K}(A)$ associated with λ_1 is $z_1 = c\mathbf{x}_1 \otimes \mathbf{x}_1$ with $c \neq 0$.*
2. *If μ_1 is not real and μ_1, $\overline{\mu_1}$ are the only rightmost eigenvalues of A, then the eigenvector of $\mathcal{K}(A)$ associated with λ_1 is $z_1 = c_1\overline{\mathbf{x}_1} \otimes \mathbf{x}_1 + c_2\mathbf{x}_1 \otimes \overline{\mathbf{x}_1}$, where c_1, c_2 are not both zero.*

Because of the correspondence between Kronecker products and Lyapunov equations [23, pp. 254–255], the eigenvalue problem $\mathcal{K}(A)z = \lambda z$ can be written as

$$AZ + ZA^T = 2\lambda Z, \tag{24}$$

where $Z \in \mathbb{C}^{n\times n}$ and $z = vec(Z)$. Under the same assumptions as in Theorem B.1, $\lambda_1 = \mathrm{Re}(\mu_1)$ is the eigenvalue of (24) with smallest modulus and $Z_1 = c_1\mathbf{x}_1\overline{\mathbf{x}_1}^T + c_2\overline{\mathbf{x}_1}\mathbf{x}_1^T$ (c_1, c_2 not both zero), which has rank 1 or 2, is the eigenvector associated with λ_1. When $c_1, c_2 \in \mathbb{R}$ and $c_1 = c_2 \neq 0$, Z_1 is real and symmetric. Let VDV^T be the truncated eigenvalue decomposition of such a Z_1, where V has one or two orthonormal columns. Since the column space of Z_1 is the same as span$\{\mathbf{x}_1, \overline{\mathbf{x}_1}\}$, we can recover the target eigenpair $(\mu_1, \mathbf{x}_1)$ easily from that of V^TAV.

Algorithm 4 A single-step Lyapunov inverse iteration for (27).

```
Input: $A \in \mathbb{R}^{n\times n}$, $V_0 \in \mathbb{R}^n$, eigtol, lyaptol
Output: An approximation $(\widehat{\lambda}, \widehat{Z} = \widehat{V}\widehat{D}\widehat{V}^T)$ to $(\lambda_1, Z_1)$
1: Let $\widehat{Z} = V_0V_0^T$ and compute the truncated eigenvalue decomposition $PCP^T$ of $2S\widehat{Z}S^T$.
2: Compute a low-rank approximate solution $\widehat{Y} = WXW^T$ to
       (25)    $SY + YS^T = PCP^T$
   that satisfies $\|S\widehat{Y} + \widehat{Y}S^T - PCP^T\|_F < lyaptol \cdot \|C\|_F$. $W$ has $d \ll n$ orthonormal columns.
3: Compute the eigenpair $(\widetilde{\lambda}_1, \widetilde{Z}_1)$ of the small eigenvalue problem
       (26)    $\widetilde{S}\widetilde{Z} + \widetilde{Z}\widetilde{S}^T = 2\widetilde{\lambda}\widetilde{S}\widetilde{Z}\widetilde{S}^T$,
   where $\widetilde{S} = W^TSW$. $\widetilde{\lambda}_1$ is the eigenvalue of (26) with smallest modulus. $\widetilde{Z}_1$ is real, symmetric, of rank 1 or 2, and $\|\widetilde{Z}_1\|_F = 1$. Compute its truncated eigenvalue decomposition $\widetilde{V}\widetilde{D}\widetilde{V}^T$.
4: Let $\widehat{\lambda} = \widetilde{\lambda}_1$ and $\widehat{Z} = \widehat{V}\widehat{D}\widehat{V}^T$, where $\widehat{V} = W\widetilde{V}$ and $\widehat{D} = \widetilde{D}$. Compute the residual norm $\|S\widehat{Z} + \widehat{Z}S^T - 2\widehat{\lambda}S\widehat{Z}S^T\|_F$ associated with (27).
5: while $\|S\widehat{Z} + \widehat{Z}S^T - 2\widehat{\lambda}S\widehat{Z}S^T\|_F > eigtol$ do
6: Increase the rank of $\widehat{Y}$ such that $\|S\widehat{Y} + \widehat{Y}S^T - PCP^T\|_F < lyaptol \cdot \|C\|_F$ still holds.
7:     Repeat steps 3 and 4.
8: endwhile
```

Lyapunov inverse iteration refers to a version of inverse iteration developed for eigenvalue problems with a similar structure as (24). It exploits the real, symmetric, and low-rank structure of the target eigenvector Z_1. Whereas each step of the standard inverse iteration requires a linear solve, each step of Lyapunov inverse iteration entails solving a Lyapunov equation whose right-hand side is of rank 1 or 2. Its solution is real, symmetric, and can be approximated by a low-rank matrix, which allows for efficient computation and storage. For an invertible A, let $S = A^{-1}$. In our experience, it is more efficient to apply Lyapunov inverse iteration to

$$SZ + ZS^T = 2\lambda SZS^T \tag{27}$$

instead, which is mathematically equivalent to (24).

Lyapunov inverse iteration often converges rapidly, requiring the solution of only a few Lyapunov equations. In particular, it has been observed in [13] to converge after only one step if the first Lyapunov equation is solved "accurately enough" (see Theorem 4.1 and Corollary 4.2 of [13]). A single-step Lyapunov inverse iteration is outlined in Algorithm 4.

In Algorithm 4, the matrices $\widehat{Y}$ and $\widehat{Z}$ are never explicitly formed; instead, we only store their factors W, X, $\widehat{V}$, and $\widehat{D}$. In step 2, we use the rational Krylov subspace method (RKSM) with adaptively chosen shifts [10] and a modification proposed in [12] to solve the Lyapunov equation (25). The main cost of the modified RKSM [12] for computing a rank-d approximate solution $\widehat{Y}$ is the construction of a d-dimensional Krylov subspace, which, in turn, entails d linear solves in the form of $(I - sA)\mathbf{x} = \mathbf{b}$ (or $(M - sJ)\mathbf{x} = \mathbf{b}$ if $A = M^{-1}J$). The shift $s > 0$ is, in general, *different* for each system. In step 3, the small eigenvalue problem (26) is solved using Lyapunov inverse iteration as well, and the matrix $\widetilde{S}$ can be obtained from RKSM as a "by-product." When the approximate eigenpair $(\widehat{\lambda}, \widehat{Z})$ is not accurate enough, instead of performing another step of inverse iteration, in step 6, we expand the existing Krylov subspace built in step 2 to produce a new $\widehat{Y}$ whose rank is higher. A new estimate $(\widehat{\lambda}, \widehat{Z})$ can then be computed from this $\widehat{Y}$.

Not only can we apply Algorithm 4 to find the rightmost eigenvalue of A, as shown in [44, sects. 4.3 and 4.4], we can also reuse certain intermediate computational results of this algorithm such as $\widehat{Y}$ and $\widetilde{S}$ to compute a few other rightmost eigenvalues of A.

Acknowledgments. We thank Dr. Sebastian Timme at the University of Liverpool for providing the *Aerofoil* matrix. We are also grateful to the referees for their many suggestions and comments that helped improve our manuscript.

REFERENCES

[1] A. H. Al-Mohy and N. J. Higham, *Computing the action of the matrix exponential, with an application to exponential integrators*, SIAM J. Sci. Comput., 33 (2011), pp. 488–511, https://doi.org/10.1137/100788860.

[2] J. Baglama, D. Calvetti, and L. Reichel, *Fast Leja points*, Electron. Trans. Numer. Anal., 7 (1998), pp. 124–140.

[3] L. Bergamaschi, M. Caliari, and M. Vianello, *Efficient approximation of the exponential operator for discrete 2D advection-diffusion problems*, Numer. Linear Algebra Appl., 10 (2003), pp. 271–289.

[4] R. Boisvert, R. Pozo, K. Remington, B. Miller, and R. Lipman, *Matrix Market*, available online from http://math.nist.gov/MatrixMarket/.

[5] M. Caliari, *Accurate evaluation of divided differences for polynomial interpolation of exponential propagators*, Computing, 80 (2007), pp. 189–201.

[6] M. Caliari, P. Kandolf, A. Ostermann, and S. Rainer, *Comparison of software for computing the action of the matrix exponential*, BIT, 54 (2014), pp. 113–128.

[7] M. Caliari, P. Kandolf, A. Ostermann, and S. Rainer, *The Leja method revisited: Backward error analysis for the matrix exponential*, SIAM J. Sci. Comput., 38 (2016), pp. A1639–A1661, https://doi.org/10.1137/15M1027620.

[8] M. Caliari, M. Vianello, and L. Bergamaschi, *Interpolating discrete advection-diffusion propagators at Leja sequences*, J. Comput. Appl. Math., 172 (2004), pp. 79–99.

[9] V. Druskin, L. Knizhnerman, and V. Simoncini, *Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation*, SIAM J. Numer. Anal., 49 (2011), pp. 1875–1898, https://doi.org/10.1137/100813257.

[10] V. Druskin and V. Simoncini, *Adaptive rational Krylov subspaces for large-scale dynamical systems*, Systems Control Lett., 60 (2011), pp. 546–560.

[11] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics*, Oxford University Press, Oxford, 2014.

[12] H. C. Elman and M. W. Rostami, *Efficient iterative algorithms for linear stability analysis of incompressible flows*, IMA J. Numer. Anal., 36 (2016), pp. 296–316.

[13] H. C. Elman and M. Wu, *Lyapunov inverse iteration for computing a few rightmost eigenvalues of large generalized eigenvalue problems*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1685–1707, https://doi.org/10.1137/120897468.

[14] T. Ericsson and A. Ruhe, *The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems*, Math. Comp., 35 (1980), pp. 1251–1268.

[15] K. Fan, *On a theorem of Weyl concerning eigenvalues of linear transformations.* II, Proc. Nat. Acad. Sci. U.S.A., 36 (1950), pp. 31–35.

[16] T. J. Garratt, G. Moore, and A. Spence, *A generalised Cayley transforms for the numerical detection of Hopf bifurcations in large systems*, in Contributions in Numerical Mathematics, R. P. Agarwal, ed., World Scientific, Singapore, 1993, pp. 177–195.

[17] W. J. F. Govaerts, *Numerical Methods for Bifurcations of Dynamical Equilibria*, SIAM, Philadelphia, 2000, https://doi.org/10.1137/1.9780898719543.

[18] N. Guglielmi and C. Lubich, *Low-rank dynamics for computing extremal points of real pseudospectra*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 40–66, https://doi.org/10.1137/120862399.

[19] N. Guglielmi and M. L. Overton, *Fast algorithms for the approximation of the pseudospectral abscissa and pseudospectral radius of a matrix*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1166–1192, https://doi.org/10.1137/100817048.

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

[21] F. B. Hildebrand, *Introduction to Numerical Analysis*, 2nd ed., McGraw-Hill, New York, 1974.

[22] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925, https://doi.org/10.1137/S0036142995280572.

[23] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1991.

[24] K. Meerbergen and D. Roose, *Matrix transformations for computing rightmost eigenvalues of large sparse non-symmetric eigenvalue problems*, IMA J. Numer. Anal., 16 (1996), pp. 297–346.

[25] K. Meerbergen and A. Spence, *Inverse iteration for purely imaginary eigenvalues with application to the detection of Hopf bifurcation in large scale problems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1982–1999, https://doi.org/10.1137/080742890.

[26] K. Meerbergen and R. Vandebril, *A reflection on the implicitly restarted Arnoldi method for computing eigenvalues near a vertical line*, Linear Algebra Appl., 436 (2012), pp. 2828–2844.

[27] I. Moret and P. Novati, *RD-rational approximations of the matrix exponential*, BIT, 44 (2004), pp. 595–615.

[28] J. Niesen and W. M. Wright, *Algorithm* 919: *A Krylov subspace algorithm for evaluating the φ-functions appearing in exponential integrators*, ACM Trans. Math. Software, 38 (2012), 22.

[29] R. Nong and D. C. Sorensen, *A Parameter Free ADI-like Method for the Numerical Solution of Large Scale Lyapunov Equations*, Tech. Report 09-16, Computational and Applied Mathematics Department, Rice University, Houston, TX, 2009, available online from http://www.caam.rice.edu/~sorensen/Tech_Reports.html.

[30] L. Reichel, *Newton interpolation at Leja points*, BIT, 30 (1990), pp. 332–346.

[31] M. W. Rostami, *New algorithms for computing the real structured pseudospectral abscissa and the real stability radius of large and sparse matrices*, SIAM J. Sci. Comput., 37 (2015), pp. S447–S471, https://doi.org/10.1137/140975413.

[32] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.

[33] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228, https://doi.org/10.1137/0729014.

[34] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, SIAM, Philadelphia, 2011, https://doi.org/10.1137/1.9781611970739.

[35] B. H. Sheehan, Y. Saad, and R. B. Sidje, *Computing* $\exp(-\tau A)b$ *with Laguerre polynomials*, Electron. Trans. Numer. Anal., 37 (2010), pp. 147–165.

[36] R. B. Sidje, *Expokit: A software package for computing matrix exponentials*, ACM Trans. Math. Software, 24 (1998), pp. 130–156.

[37] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288, https://doi.org/10.1137/06066120X.

[38] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477, https://doi.org/10.1137/S1064827502406415.

[39] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385, https://doi.org/10.1137/0613025.

[40] G. W. Stewart, *Matrix Algorithms Volume* II*: Eigensystems*, SIAM, Philadelphia, 2001, https://doi.org/10.1137/1.9780898718058.

[41] S. Timme, K. J. Badcock, M. Wu, and A. Spence, *Lyapunov inverse iteration for stability analysis using computational fluid dynamics*, in Proceedings of the 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA Paper 2012-1563, 2012.

[42] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, Philadelphia, 2013.

[43] L. N. Trefethen and M. Embree, *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*, Princeton University Press, Princeton, NJ, 2005.

[44] M. Wu, *Linear Stability Analysis using Lyapunov Inverse Iteration*, Ph.D. thesis, University of Maryland, College Park, MD, 2012.