École Polytechnique Fédérale de Lausanne
Chair of Numerical Algorithms and High-Performance Computing

On Krylov subspace methods for evaluating φ-functions appearing in exponential integrators

Master Semester Project
June 8, 2023

Student: Sepehr Mousavi

Supervisors: Prof. Daniel Kressner
Yannis Voet

Contents

1 Introduction

Efficient computation of certain matrix functions is one of the core topics in numerical linear algebra. Among other applications (see Chapter 2 of [5] for a comprehensive list of the applications), matrix functions often arise in solutions of differential equations:

- The exact solution to the initial value problem

  $$\dot{\mathbf{x}}(t) = A\mathbf{x}(t),\ \mathbf{x}(0) = \mathbf{x_0}, \quad \mathbf{x} \in \mathbb{C}^n, A \in \mathbb{C}^{n\times n}$$

  is given by $\mathbf{x}(t) = e^{tA}\mathbf{x_0}$. More generally, with some assumptions on the smoothness of a non-linear function g, the solution to the problem

  $$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{g}(t, \mathbf{x}(t)),\ \mathbf{x}(0) = \mathbf{x_0}, \quad \mathbf{x} \in \mathbb{C}^n, A \in \mathbb{C}^{n\times n} \tag{1.1}$$

  is given by

  $$\mathbf{x}(t) = e^{tA}\mathbf{x_0} + \int_0^t e^{(t-s)A}\mathbf{g}(t, x(s))ds. \tag{1.2}$$

- The exact solution to the initial value problem

  $$\dot{X}(t) = AX(t) + X(t)B,\ X(0) = C, \quad X, C \in \mathbb{C}^{m\times n}, A \in \mathbb{C}^{m\times m}, B \in \mathbb{C}^{n\times n}$$

  is given by $X(t) = e^{tA}Ce^{tB}$.

- The second order differential initial value problem

  $$\ddot{X}(t) + AX(t) = 0,\ X(0) = B, \dot{X}(0) = C, \quad A, X \in \mathbb{C}^{n\times n}$$

  has the solution

  $$X(t) = \cos\left(t\sqrt{A}\right)B + (\sqrt{A})^{-1}\sin\left(t\sqrt{A}\right)C = \cos\left(t\sqrt{A}\right)B + t\text{sinc}(t\sqrt{A})C,$$

  where the scalar sinc function is defined by $\text{sinc}(z) = \sin(z)/z$.

Other notable applications of matrix functions include the differential equations appearing in nuclear magnetic resonance, Markov models, and control theory.

Nonlinear ordinary differential equations of type (1.1) usually arise in spatial discretization of time dependant partial differential equations. Exponential integrators are a class of numerical methods for solving these problems, especially when they are stiff. These methods treat the linear term of (1.2) exactly and integrate the other term numerically. They are based on the premise that most of the stiffness of the problem originates from the matrix A, and not from the non-linear part. The matrix A that appears in such problems is typically large and sparse. In several applications, this matrix is symmetric and positive semi-definite, which is the case that is considered in this work. A presentation and review of these methods is given in [8]. A core step in these methods is the efficient computation of matrix-vector multiplications where the matrix is $\exp(tA)$ or $\varphi_p(tA)$ (to be defined later) for a suitable parameter t.

In this work, we focus on the computation of the action of matrix exponentials and φ-functions on given vectors using Krylov subspace methods. In section 2, basic definitions and properties of the

matrix functions regarded in this work are given. In section 3, we present the theoretical ideas and the numerical methods. The definitions of polynomial and rational Krylov subspaces, the Arnoldi process, the approximation methods, and the reference methods are presented. Furthermore, theoretical bounds for the error of the polynomial Krylov subspace approximation are derived in this section. In section 4, we present the details of our implementations of the methods of section 3 and the results of numerical experiments. In subsection 4.2, computing the action of three trigonometric matrix functions on certain vectors is motivated and the methods are applied and evaluated on them. Finally, the main conclusions of the project are summarized in the last section.

2 Matrix Functions

2.1 Definitions

This section reviews the basic definition and properties of matrix functions discussed in detail in [5]. Let $A \in \mathbb{C}^{n\times n}$ have s distinct eigenvalues $\sigma(A) = \{\lambda_k\}_{k=1}^s$. A can be expressed in the Jordan canonical form as $A = ZJZ^{-1} = Z\,\mathrm{diag}(J_k)_{k=1}^s Z^{-1}$. Let m_k denote the size of the largest Jordan block associated with λ_k. The matrix function associated with a scalar function f is defined by $f(A) := g(A)$, where $g(z)$ is the unique Hermite interpolating polynomial of degree less than $\sum_{k=1}^s m_k$, which satisfies

$$\frac{\partial^j}{\partial z^j} g(\lambda_k) = \frac{\partial^j}{\partial z^j} f(\lambda_k), \quad \forall j \in \{0, 1, \ldots, m_k - 1\}, \quad \forall k \in \{1, 2, \ldots, s\}, \tag{2.3}$$

assuming that all required derivatives of $f(z)$ exist.

Equivalently, $f(A)$ can also be defined as $f(A) := Z\,\mathrm{diag}(f(J_k))_{k=1}^s Z^{-1}$ where

$$f(J_k) = \begin{bmatrix} f(\lambda_k) & f'(\lambda_k) & \frac{1}{2}f^{(2)}(\lambda_k) & \cdots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & f'(\lambda_k) & \cdots & \frac{f^{(m_k-2)}(\lambda_k)}{(m_k-2)!} \\ & & f(\lambda_k) & \ddots & \vdots \\ & & & \ddots & f'(\lambda_k) \\ & & & & f(\lambda_k) \end{bmatrix}.$$

If $A = Z\,\mathrm{diag}(\lambda_k)_{k=1}^n Z^{-1}$ is diagonalizable, we have

$$f(A) = Z\,\mathrm{diag}(f(\lambda_k))_{k=1}^n Z^{-1}. \tag{2.4}$$

A third equivalent definition is via Cauchy's integral formula and it reads

$$f(A) := \frac{1}{2\pi i} \int_\Gamma f(z)(zI - A)^{-1} dz, \tag{2.5}$$

assuming that f is analytic on and inside a closed contour Γ that encloses $\sigma(A)$.

2.2 Matrix Exponential

The matrix function attributed to the scalar exponential function is called matrix exponential and is defined [5] as

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k. \tag{2.6}$$

Some properties of the matrix exponential follow:

- By taking the conjugate transpose of (2.6), it can be concluded that $\exp(A^*) = \exp(A)^*$.
- For all $A, B \in \mathbb{C}^{n\times n}$ that satisfy $AB = BA$, we have $\exp(A + B) = \exp(A)\exp(B)$.
- $\exp(A)$ is always invertible and its inverse is $\exp(A)^{-1} = \exp(-A)$.
- The derivative of $\exp(tA)$ with respect to a scalar t is given by $\frac{\mathrm{d}}{\mathrm{d}t}(\exp(tA)) = A\exp(tA)$.
- The determinant of $\exp(A)$ is the exponential of the trace of A: $\det(\exp(A)) = \exp(\mathrm{tr}(A))$.

2.3 φ-Functions

In the core of exponential integrators for solving stiff problems of type (1.1), lies the computation of the application of φ-functions on vectors from the previous iterations. Accurate and efficient computation of these vectors is vital for implementation of exponential integrators. For a scalar argument $z \in \mathbb{C}$, the φ-functions are defined [5] as

$$\begin{aligned} \varphi_0(z) &= e^z, \\ \varphi_p(z) &= \frac{1}{(p-1)!}\int_0^1 e^{(1-\theta)z}\theta^{p-1}d\theta, \quad p \in \mathbb{N}^*. \end{aligned} \tag{2.7}$$

The parameter p is related to the order of the exponential integrator and it typically takes values less than 5. The φ-functions satisfy [5] the recurrence relation

$$\begin{aligned} \varphi_p(z) &= z^{-1}\left[\varphi_{p-1}(z) - \frac{1}{(p-1)!}\right], && z \neq 0, \\ \varphi_p(z) &= \frac{1}{p!}, && z = 0, \end{aligned} \tag{2.8}$$

for all $p \in \mathbb{N}^*$.

The evaluation of the first three scalar φ-functions on the left half complex plane is illustrated in Figure 2.1. Note that the gradients are smaller for higher p's.

We can use (2.8) recursively as

$$\begin{aligned}
\varphi_p(z) &= z^{-1}\varphi_{p-1}(z) - \frac{1}{(p-1)!}z^{-1} \\
&= z^{-1}\left[z^{-1}\varphi_{p-2}(z) - \frac{1}{(p-2)!}z^{-1}\right] - \frac{1}{(p-1)!}z^{-1} \\
&= z^{-2}\varphi_{p-2}(z) - \frac{1}{(p-2)!}z^{-2} - \frac{1}{(p-1)!}z^{-1} \\
&= \cdots \\
&= z^{-p}\varphi_0(z) - \frac{1}{0!}z^{-p} - \frac{1}{1!}z^{-(p-1)} - \frac{1}{2!}z^{-(p-2)} - \cdots - \frac{1}{(p-1)!}z^{-1} \\
&= z^{-p}\exp(z) - \sum_{k=0}^{p-1}\frac{1}{k!}z^{-(p-k)}
\end{aligned}$$

to derive a closed form for φ_p:

$$\varphi_p(z) = z^{-p}\left(\exp(z) - \sum_{k=0}^{p-1}\frac{1}{k!}z^k\right), \quad z \neq 0. \tag{2.9}$$

For an invertible matrix $A \in \mathbb{C}^{n\times n}$, we can do the same steps and use (2.6) to get

$$\varphi_p(A) = A^{-p}\left(\exp(A) - \sum_{k=0}^{p-1}\frac{1}{k!}A^k\right) = \sum_{k=p}^{\infty}\frac{1}{k!}A^{k-p}. \tag{2.10}$$

Remark. *Using* (2.9) *and* (2.10) *for computing the φ-functions suffers from catastrophic cancellation and is numerically unstable. These relations are only used in the theoretical proofs.*

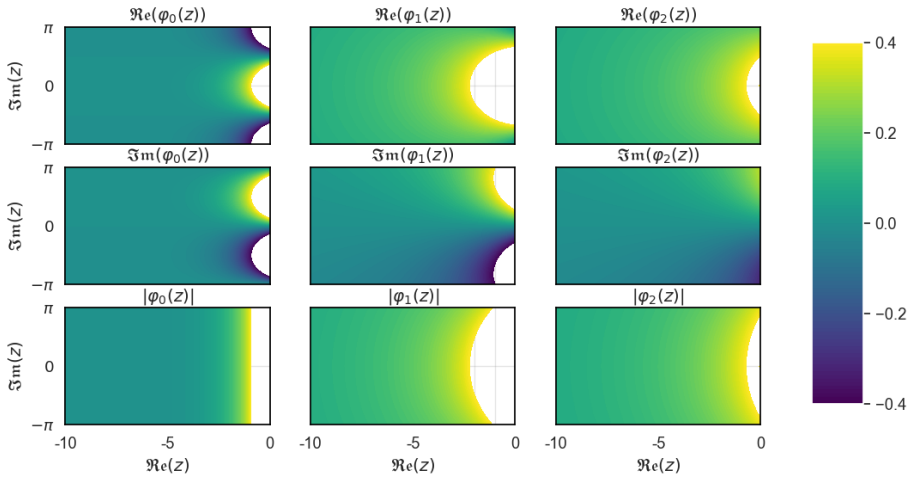


Figure 2.1: Evaluation of scalar φ-functions on the complex plane. The values that are out of the range of the color bar are not visualized (white areas).

2.4 Trigonometric Matrix Functions

The two most important trigonometric functions, sine and cosine, are defined [5] for all $A \in \mathbb{C}^{n\times n}$ by

$$\cos(A) = I - \frac{A^2}{2!} + \frac{A^4}{4!} - \frac{A^6}{6!} + \cdots = I + \sum_{k=1}^{\infty} \frac{(-1)^k}{(2k)!} A^{2k}; \tag{2.11}$$

and

$$\sin(A) = A - \frac{A^3}{3!} + \frac{A^5}{5!} - \frac{A^7}{7!} + \cdots = A + \sum_{k=1}^{\infty} \frac{(-1)^k}{(2k+1)!} A^{2k+1}. \tag{2.12}$$

The matrix sinc function associated with the scalar sinc function $\operatorname{sinc}(z) = \sin(z)/z$ is defined by

$$\begin{aligned} \operatorname{sinc}(A) &= A^{-1}\sin(A) = I - \frac{A^2}{3!} + \frac{A^4}{5!} - \frac{A^6}{7!} + \cdots \\ &= I + \sum_{k=1}^{\infty} \frac{(-1)^k}{(2k+1)!} A^{2k}. \end{aligned} \tag{2.13}$$

Some properties of the matrix sine and cosine follow:

- The matrix form of Euler's formula reads $e^{iA} = \cos(A) + i\sin(A)$.
- For all $A \in \mathbb{C}^{n\times n}$ we have $\cos(A) = \frac{1}{2}(e^{iA} + e^{-iA})$.
- For all $A \in \mathbb{C}^{n\times n}$ we have $\sin(A) = \frac{1}{2i}(e^{iA} - e^{-iA})$.
- For all $A \in \mathbb{C}^{n\times n}$ we have $\cos^2(A) + \sin^2(A) = I$.
- For real $A \in \mathbb{R}^{n\times n}$ we have $\cos(A) = \mathfrak{Re}(e^{iA})$.
- For real $A \in \mathbb{R}^{n\times n}$ we have $\sin(A) = \mathfrak{Im}(e^{iA})$.

3 Methodology

3.1 Krylov Subspaces and the Arnoldi Process

Given a matrix $A \in \mathbb{C}^{n\times n}$ and a vector $\mathbf{v} \neq 0 \in \mathbb{C}^n$, the Krylov subspace of dimension $m \leq n$ for them is defined [2] as

$$\begin{aligned} \mathcal{K}_m(A, \mathbf{v}) &:= \operatorname{span}\{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \ldots, A^{m-1}\mathbf{v}\} \\ &= \{g(A)\mathbf{v} \mid g \in \Pi_{m-1}\}, \end{aligned} \tag{3.14}$$

where Π_{m-1} is the set of all polynomials of order less than or equal to $m-1$. Krylov subspaces are essential in many methods in numerical linear algebra for solving eigenvalue problems and linear systems. However, the vectors $\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \ldots, A^{m-1}\mathbf{v}$ are not a good basis for $\mathcal{K}_m(A, \mathbf{v})$. We know from the power method that these vectors have almost the same direction and converge to the eigenvector paired with the largest eigenvalue of A. From the Arnoldi process [11] described in Algorithm 1, we can get an orthonormal basis for $\mathcal{K}_m(A, \mathbf{v})$ and an upper Hessenberg matrix $H_m = V_m^* A V_m$ which is the projection of the action of A on $\mathcal{K}_m(A, \mathbf{v})$. The Arnoldi factorization reads

$$AV_m = V_m H_m + h_{m+1,m}\mathbf{v_{m+1}}\mathbf{e_m}^\top, \tag{3.15}$$

where $\mathbf{e_m}$ is the last canonical basis vector of $\mathbb{R}^m$.

Algorithm 1: Standard Arnoldi process

Input: Matrix $A \in \mathbb{C}^{n\times n}$
Vector $\mathbf{v} \in \mathbb{C}^n \backslash \{0\}$ with $n \geq m$
Output: Orthonormal basis $V_m = (\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_m})$ of $\mathcal{K}_m(A, \mathbf{v})$

1 Set $\mathbf{v_1} = \mathbf{v}/\|\mathbf{v}\|_2$;
2 Set $V_1 = \mathbf{v_1}$;
3 **for** $j = 1, 2, \ldots, m$ **do**
4 Compute $\mathbf{w} = A\mathbf{v_j}$;
5 Compute $\mathbf{h_j} = V_j^*\mathbf{w}$;
6 Compute $\tilde{\mathbf{v}}_{\mathbf{j+1}} = \mathbf{w} - V_j\mathbf{h_j}$;
7 **if** $\|\tilde{\mathbf{v}}_{\mathbf{j+1}}\|_2 < 0.7\|\mathbf{w}\|_2$ **then**
8 Set $\hat{\mathbf{h}}_{\mathbf{j}} = V_j^*\tilde{\mathbf{v}}_{\mathbf{j+1}}$;
9 Set $\mathbf{h_j} = \mathbf{h_j} + \hat{\mathbf{h}}_{\mathbf{j}}$;
10 Set $\tilde{\mathbf{v}}_{\mathbf{j+1}} = \tilde{\mathbf{v}}_{\mathbf{j+1}} - V_j\hat{\mathbf{h}}_{\mathbf{j}}$;
11 **end**
12 Set $h_{j+1,j} = \|\tilde{\mathbf{v}}_{\mathbf{j+1}}\|_2$;
13 Set $\mathbf{v_{j+1}} = \tilde{\mathbf{v}}_{\mathbf{j+1}}/h_{j+1,j}$;
14 Set $V_{j+1} = (V_j, \mathbf{v_{j+1}})$;
15 **end**

Remark. *Lines 7 to 11 of Algorithm 1 are necessary in order to avoid a numerical instability that is well-known for the Arnoldi process. If the vector $\mathbf{w} = A\mathbf{v_j}$ is nearly contained in the column space of V_j, the vectors $\mathbf{w}$ and $V_j\mathbf{h_j} = V_jV_j^*\mathbf{w}$ will be very close to each other. Consequently, the computation of the vector $\tilde{\mathbf{v}}_{\mathbf{j+1}}$ in line 6 will be inaccurate because of catastrophic cancellation. This phenomenon is known as "loss of orthogonality" and it is signaled by small norms of the vector $\tilde{\mathbf{v}}_{\mathbf{j+1}}$. It can be cured by orthogonalizing the vector $\tilde{\mathbf{v}}_{\mathbf{j+1}}$ one more time as in lines 7 to 11.*

The rational Krylov subspace [4] associated with the matrix A and the vector $\mathbf{v}$ with non-zero poles $\xi_1, \xi_2, \ldots, \xi_m \in \overline{\mathbb{C}} := \mathbb{C} \cup \{\infty\}$ different from all eigenvalues of A, is defined as

$$\begin{aligned} \mathcal{Q}_m(A, \mathbf{v}) &:= q_{m-1}(A)^{-1}\,\mathrm{span}\{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \ldots, A^{m-1}\mathbf{v}\} \\ &= \{q_{m-1}(A)^{-1}g(A)\mathbf{v} \mid g \in \Pi_{m-1}\}, \end{aligned} \tag{3.16}$$

where g/q_{m-1} is a rational function with a prescribed denominator

$$\Pi_{m-1} \ni q_{m-1}(z) = \prod_{j=1}^{m-1}(1 - z/\xi_j).$$

Algorithm 1 can be slightly modified to get an orthonormal basis for $\mathcal{Q}_m(A, \mathbf{v})$. In line 4, instead of $\mathbf{w} = A\mathbf{v_j}$ we set $\mathbf{w} = (I - A/\xi_j)^{-1}A\mathbf{v_j}$. When the poles are repeated, the algorithm can benefit a lot from pre-computing the LU decomposition or, if applicable, the Cholesky factorization of $(I - A/\xi_j)$ outside of the main loop. These ideas are summarized in Algorithm 2.

Algorithm 2: Rational Arnoldi process

Input: Matrix $A \in \mathbb{C}^{n\times n}$
Vector $\mathbf{v} \in \mathbb{C}^n \backslash \{0\}$ with $n \geq m$,
Poles $\xi_1, \xi_2, \ldots, \xi_m$

Output: Orthonormal basis $V_m = (\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_m})$ of $\mathcal{Q}_m(A, \mathbf{v})$

for $j = 1, 2, \ldots, m$ **do**
  if ξ_j *has been seen before* **then**
    Retrieve P_j, L_j, and U_j from the previous iterations;
  else
    Compute $P_j L_j U_j = (I - A/\xi_j)$;
  end
end
Set $\mathbf{v_1} = \mathbf{v}/\|\mathbf{v}\|_2$;
Set $V_1 = \mathbf{v_1}$;
for $j = 1, 2, \ldots, m$ **do**
  Compute $\mathbf{w} = (I - A/\xi_j)^{-1} A\mathbf{v_j}$ using P_j, L_j, and U_j;
  Compute $\mathbf{h_j} = V_j^* \mathbf{w}$;
  Compute $\tilde{\mathbf{v}}_{\mathbf{j+1}} = \mathbf{w} - V_j \mathbf{h_j}$;
  if $\|\tilde{\mathbf{v}}_{\mathbf{j+1}}\|_2 < 0.7 \|\mathbf{w}\|_2$ **then**
    Set $\hat{\mathbf{h}}_\mathbf{j} = V_j^* \tilde{\mathbf{v}}_{\mathbf{j+1}}$;
    Set $\mathbf{h_j} = \mathbf{h_j} + \hat{\mathbf{h}}_\mathbf{j}$;
    Set $\tilde{\mathbf{v}}_{\mathbf{j+1}} = \tilde{\mathbf{v}}_{\mathbf{j+1}} - V_j \hat{\mathbf{h}}_\mathbf{j}$;
  end
  Set $h_{j+1,j} = \|\tilde{\mathbf{v}}_{\mathbf{j+1}}\|_2$;
  Set $\mathbf{v_{j+1}} = \tilde{\mathbf{v}}_{\mathbf{j+1}}/h_{j+1,j}$;
  Set $V_{j+1} = (V_j, \mathbf{v_{j+1}})$;
end

The rational Arnoldi factorization reads

$$AV_m(I_m + H_m D_m) + Ah_{m+1,m}\mathbf{v_{m+1}}\xi_m^{-1}\mathbf{e_m}^\top = V_m H_m + h_{m+1,m}\mathbf{v_{m+1}}\mathbf{e_m}^\top, \tag{3.17}$$

where $D_m = \mathrm{diag}(\xi_1^{-1}, \ldots, \xi_m^{-1})$. Note that the rational Krylov subspace with a repeated infinity pole is identical to the standard (polynomial) Krylov subspace.

3.2 Polynomial Krylov Subspace Approximation

In order to compute $f(A)\mathbf{v}$ for a matrix $A \in \mathbb{C}^{n\times n}$, a vector $\mathbf{v} \in \mathbb{C}^n$, and a function f that is analytic on a compact domain which contains the eigenvalues of A, we apply Algorithm 1 to get an orthonormal basis $V_m \in \mathbb{C}^{n\times m}$ for $\mathcal{K}_m(A, \mathbf{v})$ and the projection of the action of A on this Krylov subspace $H_m = V_m^* A V_m$.

Lemma 3.1. *Considering the settings described above, we can write*

$$A^k\mathbf{v} = V_m H_m^k V_m^* \mathbf{v}, \quad \forall k \in \{0, 1, \ldots, m-1\}. \tag{3.18}$$

Proof. For $k = 0$, it reads $V_m V_m^* \mathbf{v} = \mathbf{v}$ which is true because $\mathbf{v} \in \mathcal{K}_m$ and $V_m V_m^*$ is the orthogonal projection matrix into $\mathcal{K}_m$. For $k \geq 1$, we prove it by induction. Assuming that the lemma holds true for $k-1$, we can write

$$A^k \mathbf{v} = A A^{k-1} \mathbf{v} = A V_m H_m^{k-1} V_m^* \mathbf{v}.$$

Since $A^k \mathbf{v} \in \mathcal{K}_m$ for all $1 \leq k \leq m-1$, projecting it on $\mathcal{K}_m$ will result in the same vector, thus

$$A^k \mathbf{v} = V_m V_m^* A^k \mathbf{v} = V_m \underbrace{V_m^* A V_m}_{H_m} H_m^{k-1} V_m^* \mathbf{v},$$

which completes the proof. □

We approximate $f(A)\mathbf{v}$ by $f(V_m H_m V_m^*)\mathbf{v}$. Because of the orthonormality of the columns of V_m, it can be easily shown that for any $k \in \mathbb{N}^*$,

$$(V_m H_m V_m^*)^k = V_m H_m^k V_m^*,$$

which can be used alongside with the definition of matrix functions to conclude that $f(V_m H_m V_m^*)\mathbf{v} = V_m f(H_m) V_m^* \mathbf{v}$. Because all columns of V_m except the first one are orthogonal to $\mathbf{v}$, we have $V_m^* \mathbf{v} = \|\mathbf{v}\|_2 \mathbf{e_1}$, where $\mathbf{e_1}$ is the first vector in the canonical basis of $\mathbb{R}^m$. Putting all the pieces together, the approximation can be written as

$$f(A)\mathbf{v} \simeq \|\mathbf{v}\|_2 V_m f(H_m)\mathbf{e_1} =: \mathbf{f_m^{PA}}. \tag{3.19}$$

With this approximation, instead of evaluating the matrix function for a $n \times n$ matrix, we just need to evaluate it for the $m \times m$ matrix H_m which is much smaller than A.

Corollary 3.1.1. *For all $\Pi_{m-1} \ni g(z) = \sum_{k=0}^{m-1} \alpha_k z^k$, the approximation in (3.19) is exact; where Π_{m-1} is the set of all polynomials of degree up to $m-1$. This can be shown using the definition of g and Lemma 3.1:*

$$g(A)\mathbf{v} = \sum_{k=0}^{m-1} \alpha_k A^k \mathbf{v} = \sum_{k=0}^{m-1} \alpha_k V_m H_m^k V_m^* \mathbf{v} = V_m \underbrace{\left(\sum_{k=0}^{m-1} \alpha_k H_m^k \right)}_{g(H_m)} \underbrace{V_m^* \mathbf{v}}_{\|\mathbf{v}\|_2 \mathbf{e_1}}$$

$$= \|\mathbf{v}\|_2 V_m g(H_m) \mathbf{e_1}.$$

The following lemma gives rise to efficient methods for computing the application of φ-functions and the trigonometric matrix functions on a vector.

Lemma 3.2. *Consider a matrix-vector pair $(M, \mathbf{u})$ with $M \in \mathbb{C}^{n \times n}$ and $\mathbf{u} \in \mathbb{C}^n$. If we construct a matrix $\hat{M} \in \mathbb{C}^{(n+p)\times(n+p)}$ as*

$$\hat{M} = \begin{bmatrix} M & \mathbf{u} & 0 \\ 0 & 0 & I_{p-1} \\ 0 & 0 & 0 \end{bmatrix} \begin{matrix} n \\ p-1 \\ 1 \end{matrix} \quad \begin{matrix} rows \\ row(s) \\ row \end{matrix}, \tag{3.20}$$

with $p \geq 1$, the last column of $\hat{M}^k$, denoted by $(\hat{M}^k)_{[:,n+p]} \in \mathbb{C}^{n+p}$ takes the form

$$(\hat{M}^k)_{[:,n+p]} = \begin{cases} \begin{bmatrix} 0 \\ \mathbf{e_{p-k}} \\ 0 \end{bmatrix} \begin{matrix} n & rows \\ p-1 & row(s), \\ 1 & row \end{matrix} & k \in \{1, 2, \ldots, p-1\} \\ \begin{bmatrix} M^{k-p}\mathbf{u} \\ 0 \\ 0 \end{bmatrix} \begin{matrix} n & rows \\ p-1 & rows, \\ 1 & rows \end{matrix} & k \geq p \end{cases},$$

where $\mathbf{e_{p-k}}$ is the $p-k^{th}$ canonical basis vector of $\mathbb{R}^{p-1}$.

Proof. For $k = 1$, the lemma holds by definition. For $k \geq 2$, we now show that assuming that it holds for $k-1$, it is also true for k. Since $\hat{M}^k = \hat{M}\hat{M}^{k-1}$, we can get the $n+p^{\text{th}}$ column of $\hat{M}^k$, denoted by $(\hat{M}^k)_{[:,n+p]}$, by multiplying only the $n+p^{\text{th}}$ column of $\hat{M}^{k-1}$. For $2 \leq k \leq p-1$ we have

$$\begin{aligned}(\hat{M}^k)_{[:,n+p]} &= \hat{M}(\hat{M}^{k-1})_{[:,n+p]} \\ &= \begin{bmatrix} M & \mathbf{u} & 0 \\ 0 & 0 & I_{p-1} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{e_{p-k+1}} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{e_{p-k}} \\ 0 \end{bmatrix}.\end{aligned}$$

After each multiplication, we get the previous column of $\hat{M}$ until we reach the $n+1^{\text{th}}$ column for $k = p$, $(\hat{M}^p)_{[:,n+p]} = \begin{bmatrix} \mathbf{u}^\top & 0 & 0 \end{bmatrix}^\top$, which is consistent with the lemma. With this, for $k \geq p+1$ we have

$$\begin{aligned}(\hat{M}^k)_{[:,n+p]} &= \hat{M}(\hat{M}^{k-1})_{[:,n+p]} \\ &= \begin{bmatrix} M & \mathbf{u} & 0 \\ 0 & 0 & I_{p-1} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} M^{k-1-p}\mathbf{u} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} M^{k-p}\mathbf{u} \\ 0 \\ 0 \end{bmatrix}.\end{aligned}$$

□

Corollary 3.2.1. *For a matrix-vector pair $(A, \mathbf{v})$ with $A \in \mathbb{C}^{n\times n}$ and $\mathbf{v} \in \mathbb{C}^n$, the action of the p^{th} φ-function of A on the vector $\mathbf{v}$ can be read from the first n entries of the last column of $\exp\left(\hat{A}\right)$:*

$$\exp\left(\hat{A}\right)_{[1:n,n+p]} = \varphi_p(A)\mathbf{v},$$

where $\hat{A}$ is defined in (3.20) for the matrix-vector pair $(A, \mathbf{v})$.

Proof. Using the definition of matrix exponential, Lemma 3.2, and (2.10), we can write

$$\exp\left(\hat{A}\right)_{[1:n,n+p]} = \sum_{k=0}^{\infty} \frac{1}{k!}(\hat{A}^k)_{[1:n,n+p]} = \sum_{k=p}^{\infty} \frac{1}{k!}A^{k-p}\mathbf{v} = \varphi_p(A)\mathbf{v}.$$

□

Using Corollary 3.2.1, to compute the approximation of the φ-functions in (3.19), we follow [10] and read $\varphi_p(H_m)\mathbf{e_1}$ from the first m entries of the last column of $\exp\left(\hat{H}_m\right)$, with $\hat{H}_m$ defined in (3.20) for the matrix-vector pair $(H_m, \mathbf{e_1})$.

Corollary 3.2.2. *For a matrix-vector pair $(A, \mathbf{v})$ with $A \in \mathbb{C}^{n\times n}$ and $\mathbf{v} \in \mathbb{C}^n$, the action of the matrix sinc function of A on the vector $\mathbf{v}$ can be computed by computing only the last column of $\sin\left(\hat{A}\right)$:*

$$\sin\left(\hat{A}\right)_{[1:n,n+1]} = \hat{A}_{[1:n,n+1]} + \sum_{k=1}^{\infty} \frac{(-1)^k}{(2k+1)!} \hat{A}^{2k+1}_{[1:n,n+1]} = \mathbf{v} + \sum_{k=1}^{\infty} \frac{(-1)^k}{(2k+1)!} A^{2k}\mathbf{v} = \operatorname{sinc}(A)\mathbf{v},$$

where $\hat{A}$ is defined in (3.20) for the matrix-vector pair $(A, \mathbf{v})$ with $p = 1$.

3.2.1 Error Estimation

The following theorem gives an upper bound for the approximation error of (3.19).

Theorem 3.3. *Let $A \in \mathbb{C}^{n\times n}$ be a Hermitian matrix. For a general scalar function $f : \mathbb{C} \to \mathbb{C}$ that is analytic on a domain $\Omega \subset \mathbb{C}$ containing the spectrum of A, $\sigma(A) \subset [\alpha, \beta] \subset \mathbb{R}$, the error of the approximation $\mathbf{f_m} := \|\mathbf{v}\|_2 V_m f(H_m)\mathbf{e_1}$ defined in (3.19) is bounded by the maximum error of the best scalar polynomial approximation of f in $[\alpha, \beta]$:*

$$\|f(A)\mathbf{v} - \mathbf{f_m}\|_2 \leq 2\|\mathbf{v}\|_2 \min_{g\in\Pi_{m-1}} \max_{z\in[\alpha,\beta]} |f(z) - g(z)| \tag{3.21}$$

Proof. Considering an arbitrary polynomial $g \in \Pi_{m-1}$, we define the error $\varepsilon = f - g$, we use Corollary 3.1.1, the triangle inequality, and the orthonormality of the columns of V_m to get

$$\begin{aligned}
\|f(A)\mathbf{v} - \mathbf{f_m}\|_2 &= \left\| f(A)\mathbf{v} \overbrace{- g(A)\mathbf{v} + \|\mathbf{v}\|_2 V_m g(H_m)\mathbf{e_1}}^{\text{zero by Corollary 3.1.1}} - \|\mathbf{v}\|_2 V_m f(H_m)\mathbf{e_1} \right\|_2 \\
&\leq \|f(A)\mathbf{v} - g(A)\mathbf{v}\|_2 + \|\mathbf{v}\|_2 \|V_m[g(H_m)\mathbf{e_1} - f(H_m)\mathbf{e_1}]\|_2 \\
&\leq \|\mathbf{v}\|_2 (\|\varepsilon(A)\|_2 + \|\varepsilon(H_m)\|_2) \\
&\leq 2\|\mathbf{v}\|_2 \max_{z\in[\alpha,\beta]} |\varepsilon(z)| .
\end{aligned}$$

In the last step, we used that $\sigma(H_m) \subset [\alpha, \beta]$ holds due to eigenvalue interlacing. □

To specialize Theorem 3.3 to φ-functions, we follow two approaches. The first one is using truncated Taylor series to get an upper bound for the right-hand-side of (3.21), which results in the following lemma.

Lemma 3.4. *For a φ-function $\varphi_p(z)$ defined in (2.7) and a Hermitian matrix $A \in \mathbb{C}^{n\times n}$ with the spectrum $\sigma(A) \subset [-\lambda, 0] \subset \mathbb{R}$ with $\lambda > 0$, the error of the approximation $\varphi_{\mathbf{p},\mathbf{m}} := \|\mathbf{v}\|_2 V_m \varphi_p(H_m)\mathbf{e_1}$ is upper bounded as:*

$$\|\varphi_p(A)\mathbf{v} - \varphi_{\mathbf{p},\mathbf{m}}\|_2 \leq 2\|\mathbf{v}\|_2 \frac{\lambda^m}{(m+p)!} \tag{3.22}$$

Proof. We replace the exponential in (2.9) by its truncated Taylor expansion around zero and

we keep the first $m+p-1$ terms to get

$$\begin{aligned}\varphi_p(z) &= z^{-p}\left(\sum_{k=0}^{m+p-1}\frac{1}{k!}z^k + \frac{1}{(m+p)!}|z|^{m+p}\exp(z') - \sum_{k=0}^{p-1}\frac{1}{k!}z^k\right)\\ &= z^{-p}\left(\sum_{k=p}^{m+p-1}\frac{1}{k!}z^k + \frac{1}{(m+p)!}|z|^{m+p}\exp(z')\right)\\ &\leq \sum_{k=0}^{m-1}\frac{1}{(k+p)!}z^k + \frac{1}{(m+p)!}|z|^m, \quad \forall z\in[-\lambda,0]\end{aligned}$$

where $z'\in[z,0]\subseteq[-\lambda,0]$. In the last step, we used $\exp(z')\leq 1$ for all non-positive z'. By choosing $g(z)=\sum_{k=0}^{m-1}\frac{1}{(k+p)!}z^k$ and taking the maximum of the absolute value of both sides for $z\in[-\lambda,0]$, we can write

$$\max_{z\in[-\lambda,0]}|\varphi_p(z)-g(z)| \leq \frac{1}{(m+p)!}\max_{z\in[-\lambda,0]}|z|^m = \frac{1}{(m+p)!}\lambda^m.$$

With this, we have found a polynomial $g\in\Pi_{m-1}$ that has this error bound, so we can be sure that the minimum of the error bound over all polynomials in Π_{m-1} is less than this bound. Combining this result with Theorem 3.3 completes the proof. □

However, this bound turns out to be very pessimistic for small m when λ is large. Using the polynomial approximation given in [7, Lemma A.1], we can get a better error bound for approximating $\varphi_1(A)\mathbf{v}$. The polynomial approximation states that

$$\min_{g\in\Pi_{m-1}}\max_{z\in[-\lambda,0]}|\varphi_1(z)-g(z)| \leq \begin{cases}\frac{5\lambda^2}{2m^3}\exp\left(-\frac{4m^2}{5\lambda}\right) & \sqrt{\lambda}\leq m\leq\frac{\lambda}{2}\\ \frac{32}{12m-5\lambda}\left(\frac{e\lambda}{4m+2\lambda}\right)^m & m\geq\frac{\lambda}{2}\end{cases}. \tag{3.23}$$

Using this polynomial approximation error bound and combining it with (3.21) gives the following theorem for the Arnoldi method approximation of $\varphi_1(A)v$ for Hermitian matrices.

Theorem 3.5. *For the φ-function $\varphi_1(z)$ defined in (2.7) and a Hermitian matrix $A\in\mathbb{C}^{n\times n}$ with the spectrum $\sigma(A)\subset[-\lambda,0]\subset\mathbb{R}$ for $\lambda>0$, the error of the approximation $\varphi_{\mathbf{1},\mathbf{m}} := \|\mathbf{v}\|_2 V_m\varphi_1(H_m)\mathbf{e_1}$ scales with λ as*

$$\|\varphi_1(A)\mathbf{v}-\varphi_{\mathbf{1},\mathbf{m}}\|_2 \leq \begin{cases}\|\mathbf{v}\|_2\frac{5\lambda^2}{m^3}\exp\left(-\frac{4m^2}{5\lambda}\right) & \sqrt{\lambda}\leq m\leq\frac{\lambda}{2},\\ \|\mathbf{v}\|_2\frac{64}{12m-5\lambda}\left(\frac{e\lambda}{4m+2\lambda}\right)^m & m\geq\frac{\lambda}{2}.\end{cases} \tag{3.24}$$

3.3 Rational Krylov Subspace Approximation

Similarly to subsection 3.2, we apply Algorithm 2 to get an orthonormal basis for $\mathcal{Q}_m(A,\mathbf{v})$ and an upper Hessenberg matrix H_m. We then approximate $f(A)\mathbf{v}$ by $f(V_mA_mV_m^*)\mathbf{v}$ where $A_m = V_m^*AV_m$. If $\xi_m=\infty$, it can be easily shown from (3.17) that $A_m = H_mK_m^{-1}$ where $K_m = I_m + H_mD_m$. With similar arguments as in the previous section, we can write the approximation as

$$f(A)\mathbf{v}\simeq\|\mathbf{v}\|_2 V_m f(A_m)\mathbf{e_1} =: \mathbf{f_m^{RA}}. \tag{3.25}$$

For the φ-functions, we can exploit Corollary 3.2.1, and compute $\varphi_p(A_m)\mathbf{e_1}$ by reading the first m entries of the last column of $\exp\left(\hat{A}_m\right)$, with $\hat{A}_m$ defined in (3.20) for the matrix-vector pair $(A_m,\mathbf{e_1})$.

3.3.1 Pole Selection

Before building a rational Krylov subspace and do the steps in Algorithm 2, we need to specify the poles of the rational functions in this subspace. The choice of the poles is very important since it restricts the denominator of the rational functions in the subspace. It can be shown (see Corollary 3.4 in [4]) that the rational Krylov subspace approximation satisfies

$$\left\| f(A)\mathbf{v} - \mathbf{f_m^{RA}} \right\|_2 \leq 2C \left\|\mathbf{v}\right\|_2 \min_{r_m \in \mathcal{Q}_m} \max_{z \in \Sigma} |f(z) - r_m(z)|$$

with a constant $C \leq 11.08$ ($C = 1$ for Hermitian A), assuming that f is analytic in the neighbourhood of a compact set $\Sigma \supseteq \mathcal{W}(A)$, where $\mathcal{W}(A)$ is the numerical range of the matrix A and is defined as

$$\mathcal{W}(A) = \{\mathbf{x}^* A\mathbf{x} \mid \mathbf{x} \in \mathbb{C}^n, \|\mathbf{x}\|_2 = 1\}. \tag{3.26}$$

The goal is to choose the poles such that the rational Krylov subspace contains a good approximant of $f(z)$ in the set Σ. The AAA algorithm, introduced in [9], takes advantage of the barycentric representation of rational functions and approximates a function by interpolating it at a number of nodes that are chosen among the given nodes on a domain in the complex plane. At each iteration, it chooses the next interpolation point by a greedy algorithm which picks the point that has the maximum absolute residual on that iteration. The poles of the rational function can then be easily computed. One limitation of this algorithm is that since it works based on the interpolation points, it does not allow for putting constrains on the poles directly. However, for some functions, it might be possible to constrain the poles indirectly by finding a mapping between the domain of the interpolation points and the domain of the resulting poles.

3.4 Reference Evaluations

In this subsection, we describe the methods that are used for computing the reference evaluations of the action of matrix functions on vectors for large sparse matrices. We take these evaluations as reference for calculating the error of the approximation methods.

Consider a large sparse matrix $A \in \mathbb{C}^{n\times n}$ and a vector $\mathbf{v} \in \mathbb{C}^n$. In order to compute the action of $\varphi_p(A)$ on $\mathbf{v}$, we use Corollary 3.2.1 for the matrix-vector pair $(A, \mathbf{v})$ and read $\varphi_p(A)\mathbf{v}$ from the first n entries of the last column of $\exp\left(\hat{A}\right)$. Contrary to the approximation techniques described in the previous subsections, here, the embedded matrix will be large and sparse. Thus, computing its matrix exponential directly might introduce memory issues since the matrix exponential does not inherit the sparsity of A. However, since we are only interested in the last column of the matrix exponential, we can exploit the sparsity of the embedded matrix $\hat{A}$ and directly compute the action of the matrix exponential on the last canonical basis vector of $\mathbb{R}^{n+p}$, $\mathbf{e_{n+p}} = [0\ 0\ \cdots\ 1]^\top$, which gives its last column:

$$\varphi_p(A)\mathbf{v} = \exp\left(\hat{A}\right)_{[1:n,n+p]} = \exp\left(\hat{A}\right)\mathbf{e_{n+p}} = \phi_{\mathbf{p}}^{\mathbf{EX}}.$$

We use the `scipy.sparse.linalg.expm_multiply` function from the SciPy library [12] for this purpose.

For computing the action of $\cos(A)$ and $\sin(A)$ on $\mathbf{v}$, we take advantage of the properties of

trigonometric matrix functions described in subsection 2.4 to get

$$\cos(A)\mathbf{v} = \frac{1}{2}(\exp(iA)\mathbf{v} + \exp(-iA)\mathbf{v}),$$
$$\sin(A)\mathbf{v} = \frac{1}{2i}(\exp(iA)\mathbf{v} - \exp(-iA)\mathbf{v}).$$

For real A and $\mathbf{v}$, the above equations can be simplified to the real and the imaginary part of $\exp(iA)\mathbf{v}$ for $\cos(A)\mathbf{v}$ and $\sin(A)\mathbf{v}$, respectively. The advantage of this technique for large sparse matrices is that instead of computing the whole $\cos(A)$ and $\sin(A)$ matrices, we only need to compute the last column of two matrix exponentials. This is again achieved with the `scipy.sparse.linalg.expm_multiply` function.

For computing the action of $\operatorname{sinc}(A)$ on $\mathbf{v}$, we use Corollary 3.2.2 and compute the last column of $\sin\left(\hat{A}\right)$ with $\hat{A}$ defined in (3.20) for the matrix-vector pair $(A, \mathbf{v})$ with $p = 1$. For computing $\operatorname{sinc}^2(A)\mathbf{v}$, we first compute $\mathbf{u} := \operatorname{sinc}(A)\mathbf{v}$ with the method described above, and then compute $\operatorname{sinc}^2(A)\mathbf{v} = \operatorname{sinc}(A)\mathbf{u}$ by reading the last column of $\sin\left(\hat{A}\right)$ with $\hat{A}$ being defined by (3.20) for the matrix-vector pair $(A, \mathbf{u})$.

4 Implementations and Experiments

The implementations[1] of the methods described in section 3 are presented and evaluated in this section.

4.1 φ-Functions

Test Matrices

In order to test the methods, we consider the matrices that appear after the finite difference discretization of 1D and 2D Laplace operator using a uniform grid. These matrices are sparse, real, symmetric, and definite. They are obtained by

$$\mathbb{R}^{n\times n} \ni \hat{A}_{1D} = (n+1)^2 K_n,$$
$$\mathbb{R}^{n^2\times n^2} \ni \hat{A}_{2D} = (n+1)^2 (I_n \otimes K_n + K_n \otimes I_n),$$

where $\otimes$ denotes the Kronecker product, n is the number of interior grid points, and K_n is a tridiagonal matrix defined as

$$K_n = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{bmatrix}.$$

[1] The codes are available in Github.

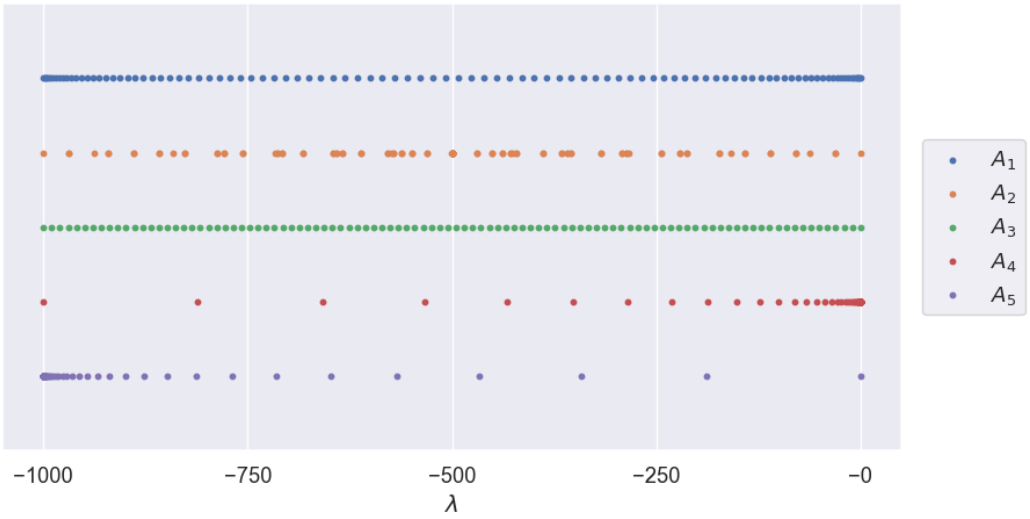


Figure 4.2: Distribution of the eigenvalues of the test matrices with $n = 100$ and $\lambda = 1000$.

We are interested in computing $\varphi_p(tA)\mathbf{v}$ with $t \in \mathbb{R}$ being a time step. We take $\hat{A}_{1D} \in \mathbb{R}^{n\times n}$ and $\hat{A}_{2D} \in \mathbb{R}^{n\times n}$ (n must have an integer square root), and we spectrally scale them so that they both have the same spectral interval $[-\lambda, -\epsilon]$ with a parameter $\lambda > 0$ and $\epsilon = 0$. We denote the resulting matrices by A_1 and A_2, respectively. The spectral interval of a complex Hermitian or a real symmetric matrix is the smallest interval in $\mathbb{R}$ that contains all the eigenvalues of the matrix, and it coincides with the numerical range of the matrix. Assuming that the spectral interval of the original matrix is $[\lambda_1, \lambda_n] \subset \mathbb{R}^-$, the spectral scaling is done by $A_i = a\hat{A}_{iD} + bI$ for $i = 1, 2$; where the coefficients are $a = (\lambda - \epsilon)/(\lambda_n - \lambda_1)$ and $b = (-\lambda\lambda_n + \epsilon\lambda_1)/(\lambda_n - \lambda_1)$. By scaling the matrices this way, we can focus on studying the effect of different eigenvalue distributions on the same interval. The value of $\varphi_p(A_i)\mathbf{v}$ can then be associated to $\varphi_p(t\hat{A}_{iD})\mathbf{v}$ with $t \simeq \lambda/|\lambda_1|$ assuming that ϵ is sufficiently small and $\lim_{n\to\infty} \lambda_n = 0$, which happens to be the case for $\hat{A}_{1D}$ and $\hat{A}_{2D}$. For $\hat{A}_{1D}$ with $n = 10^4$, for instance, picking $\lambda = 10^5$ corresponds to $t \simeq 2.5\text{e}{-}04$.

Beside these two matrices, we consider three diagonal matrices of size $n \times n$, A_3, A_4, and A_5, with eigenvalues distributed on the same interval, $[-\lambda, -\epsilon]$. The eigenvalues of A_3 are distributed uniformly. The eigenvalues of A_4 and A_5 are distributed geometrically so that they are concentrated on the right and the left ends, respectively. We test with matrices with different λ's and different sizes. Figure 4.2 shows the eigenvalues of these matrices with $n = 100$ and $\lambda = 1000$.

Pole Selection for Rational Krylov Subspace

We use the `baryrat` library introduced in [6] which implements the AAA algorithm for rational approximations. One downside of using this algorithm is the additional computational cost that it brings for computing the poles. The AAA algorithm computes the residual of the approximation on all the given interpolation points at each iteration. Hence, for approximating a rational function of degree m, its computation cost scales as $O(mq)$ where q is the number of interpolation points given as input. Furthermore, since this algorithm computes the poles based on the interpolation points, the selected poles by this algorithm are very sensitive to the discretization of the interpolation domain. Considering the $[-\lambda, 0] \subset \mathbb{R}$ interval, the number of

interpolation points for the AAA algorithm to work accurately scales with λ, which becomes a real burden when $\lambda \to \infty$. However, as depicted in Figure 4.3, numerical results show that for this interval, the computed poles converge quite early when λ is increased. We take advantage of this behaviour and pre-compute the poles by running the AAA algorithm on the interval $[-10^4, -10^{-16}]$ with 30000 logarithmically spaced interpolation points. The poles for φ_1, φ_3, φ_5, and φ_{10} are pre-computed and stored. Spacing the interpolation points logarithmically results in more interpolation points close to the right end of the interval. This is better than spacing them uniformly because for the exponential and the φ-functions, the gradients become very small as we go further away from the origin to the left side of the real axis (i.e., as $\mathfrak{Re}(z) \to -\infty$ in Figure 2.1).

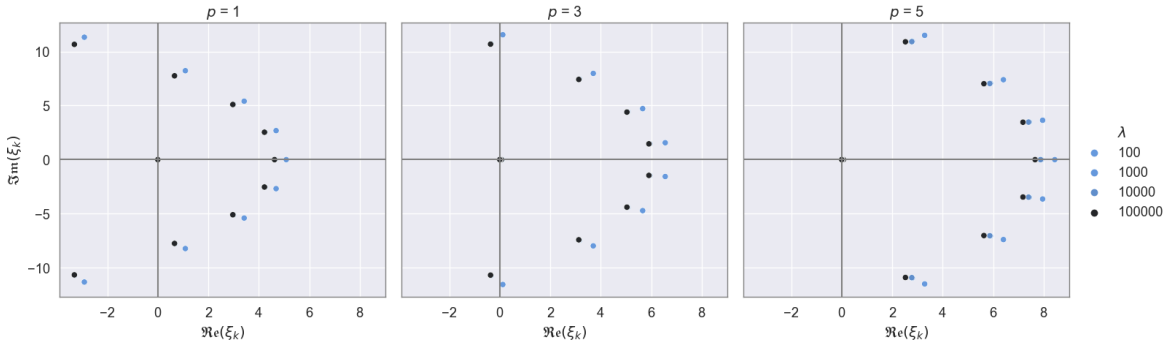


Figure 4.3: The first 10 poles computed by the AAA algorithm for φ-functions with 20000 logarithmically spaced interpolation points on the interval $[-\lambda, -10^{-16}]$.

Using the poles from this strategy, the maximum errors of the scalar rational approximations on different intervals have been computed and are presented in Figure 4.4. These plots show that it is possible to reach a maximum error of 10^{-12} with a degree less than 15. On the other hand, it has been observed that when the degree is increased, the computed poles start taking values close to or on the negative real axis. Figure 4.5 illustrates this behavior for φ_3. With $m = 15$, we can see that some poles are getting close to the real negative axis. With $m = 20$, 3 poles lie on the negative real axis. With $m = 25$, we can see that the situation is already very bad with many poles lying on the negative real axis. This can significantly impact the performance of the rational Krylov subspace approximation method if the input matrix has real negative eigenvalues, which happens to be the case for our test matrices. If a pole ξ_j falls close to one of the eigenvalues of the matrix A, the matrix $(I - A/\xi_j)$ in line 11 of Algorithm 2 will have an eigenvalue close to zero and thus, it will have a high condition number. Comparing the computed poles with different number of interpolation points (20000 and 30000) in Figure 4.5, it can be concluded that when m gets larger, more interpolation points are needed in order to compute the poles accurately. This justifies the unexpected increases in the errors in Figure 4.4 for φ_1 and φ_5.

For a rational Krylov subspace of dimension m, we take k pre-computed poles $\eta_1, \ldots, \eta_k$ by the AAA algorithm and we repeat them enough times to fill the m poles $\xi_1, \ldots, \xi_m$. When $m < k$, we just take the first m poles. With this scheme for the pole selection, the rational Krylov subspace method is denoted by `RA-AAAk` where `k` is replaced by its value. We also consider the case with a single repeated pole $\xi_1 = \xi_2 = \cdots = \xi_m = 1$ and we denote it by `RA-ONES`.

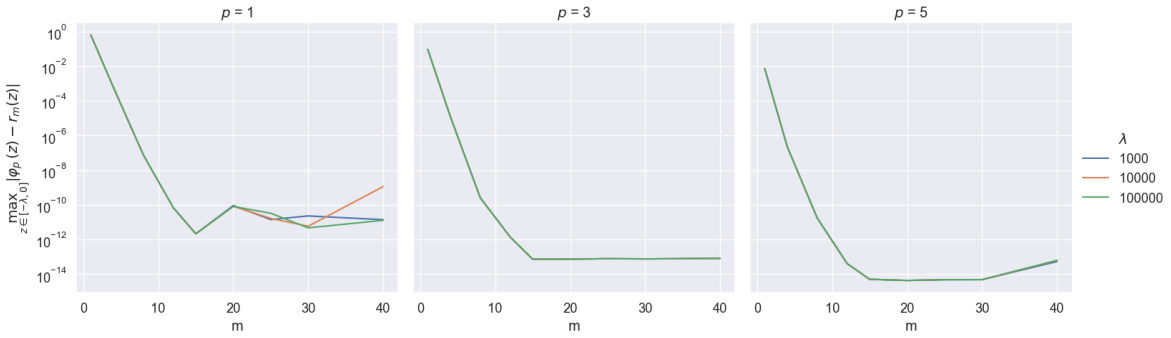


Figure 4.4: Maximum error of the rational approximation of scalar φ-functions on the interval $[-\lambda, 0]$. The approximations are done on the interval $[-10^4, -10^{-16}]$ with 30000 logarithmically spaced interpolation points.

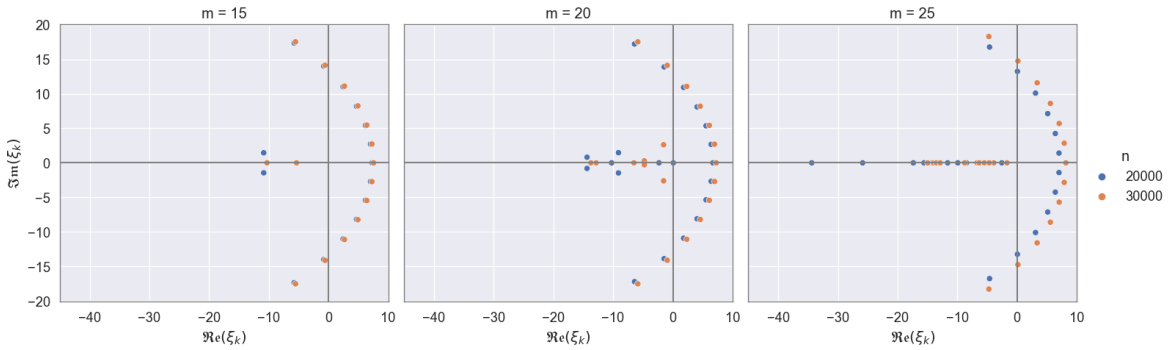


Figure 4.5: Computed poles by the AAA algorithm for φ_3 with 20000 and 30000 logarithmically spaced interpolation points on the interval $[-10^4, -10^{-16}]$.

Remark. *In order to approximate the scalar function, the AAA algorithm needs to evaluate the scalar $\varphi_p(z)$ many times in the interval. For φ_0 the scalar exponential is used. For φ_1, the* `scipy.special.expm1` *function divided by z gives accurate evaluations for large and small z. For φ_p, $p > 1$, two methods are used depending on the absolute value of z. For $|z| > 1$, (2.8) is used in a recursive algorithm. However, for small $|z|$, this method lacks numerical stability because of the numerical cancellation that happens in subtracting the limit value $\lim_{z\to 0} \varphi_{p-1}(z) = \frac{1}{(p-1)!}$ from $\varphi_{p-1}(z)$. Hence, for non-zero $|z| < 1$, we use Corollary 3.2.1 and read $\varphi_p(z)$ from the first entry of the last column of $\exp\left(\hat{A}\right)$ with $\hat{A}$ being defined in (3.20) for the matrix-vector pair $(z \in \mathbb{C}^{1\times 1}, 1 \in \mathbb{C}^1)$.*

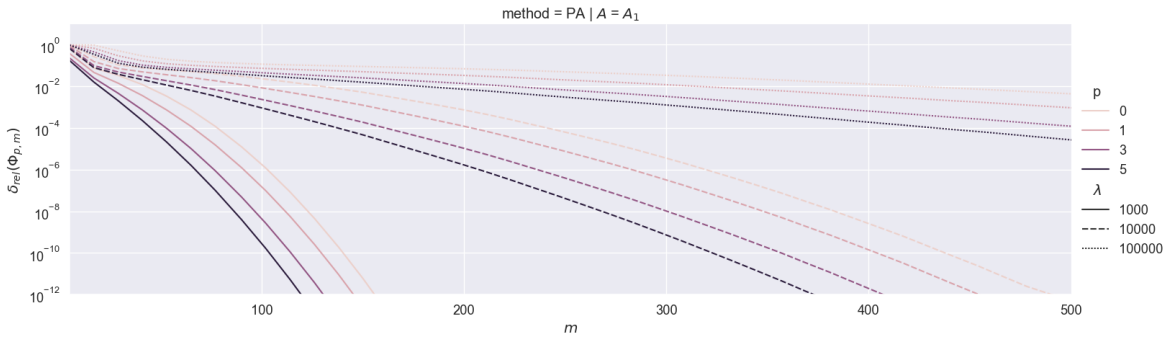
Numerical Results

The methods described in subsection 3.2 (PA) and subsection 3.3 (RA) are implemented and their approximations have been evaluated for the test matrices. For all the test matrices, the approximations are carried out for a unit random vector $\mathbf{v}$. For computing $\exp\left(\hat{H}_m\right)$ and $\exp\left(\hat{A}_m\right)$, the `scipy.linalg.expm` function from the the SciPy library [12] is used which implements a scaling and squaring algorithm with a Padé approximant for which the order is decided based on the matrix [1]. In order to evaluate the approximation methods, we take the vectors computed by the methods described in subsection 3.4 as reference and we report the relative error of the re-

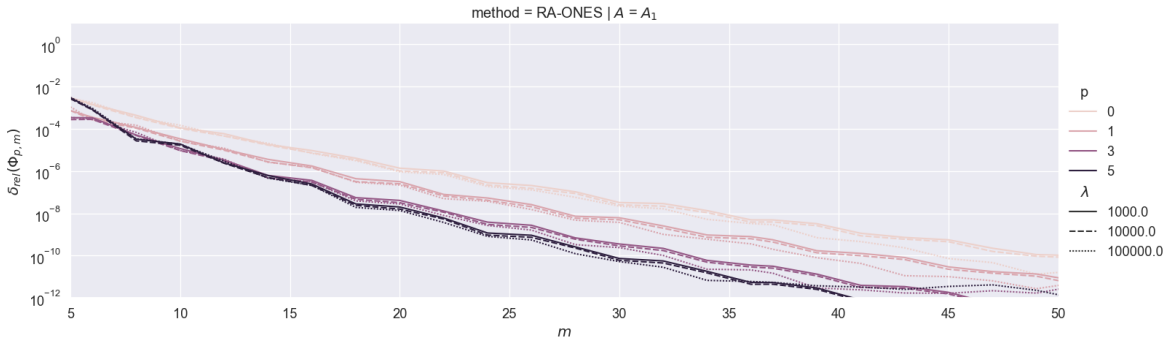
sults from the Krylov subspace methods of dimension m, denoted by $\mathbf{\Phi_{p,m}}$, against the reference vectors, denoted by $\mathbf{\Phi_p^{EX}}$. The relative error is computed as:

$$\delta_{rel}(\mathbf{\Phi_{p,m}}) = \frac{\left\| \mathbf{\Phi_{p,m}} - \mathbf{\Phi_p^{EX}} \right\|_2}{\left\| \mathbf{\Phi_p^{EX}} \right\|_2}.$$

The convergence plots for the polynomial (top) and the rational (bottom) Krylov subspace approximations are illustrated in Figure 4.6 for A_1 with size $n = 10^4$. Comparing the top and the bottom plots, it could be concluded that the RA method needs fewer iterations to converge. Another major difference is the impact of the parameter λ (the smallest eigenvalue). With the polynomial Krylov subspace method, increasing λ drastically slows down the convergence rate; whereas the rational Krylov subspace method converges more or less with the same rate for $\lambda = 10^3$ and $\lambda = 10^5$. Furthermore, it can be observed that both methods always converge slightly faster for larger p's. This observation is consistent with the error bound provided in Lemma 3.4, where we can see that increasing p slightly improves the bound. This slight improvement can also be justified by looking at the φ-functions on the complex plane. As depicted in Figure 2.1, when p is increased, the function φ_p grows more slowly on the real axis and thus, its changes can be captured better by a polynomial of a certain degree. Therefore, the polynomial Krylov subspace approximation needs fewer iterations to reach a certain tolerance according to Theorem 3.3.



(a) Polynomial Krylov subspace approximation.



(b) Rational Krylov subspace approximation.

Figure 4.6: Convergence of the approximations (3.19) and (3.25) for matrices of size $n = 10^4$.

Figure 4.7 compares the convergence of the test matrices with the same numerical range (same λ). With $\lambda = 10^3$, the method converges with almost the same rate for all the test matrices, which is

very close to the rate estimated in (3.24). As λ gets larger, we observe that the convergence for the matrices A_2 and A_5 gets considerably faster than for the other matrices. What these two matrices have in common is that the distributions of their eigenvalues (see Figure 4.2) are less dense on the right end of the interval $[-\lambda, 0]$, where the scalar function has high gradients. Therefore, for most of their eigenvalues, the polynomial approximant has a smaller error compared to the other matrices.

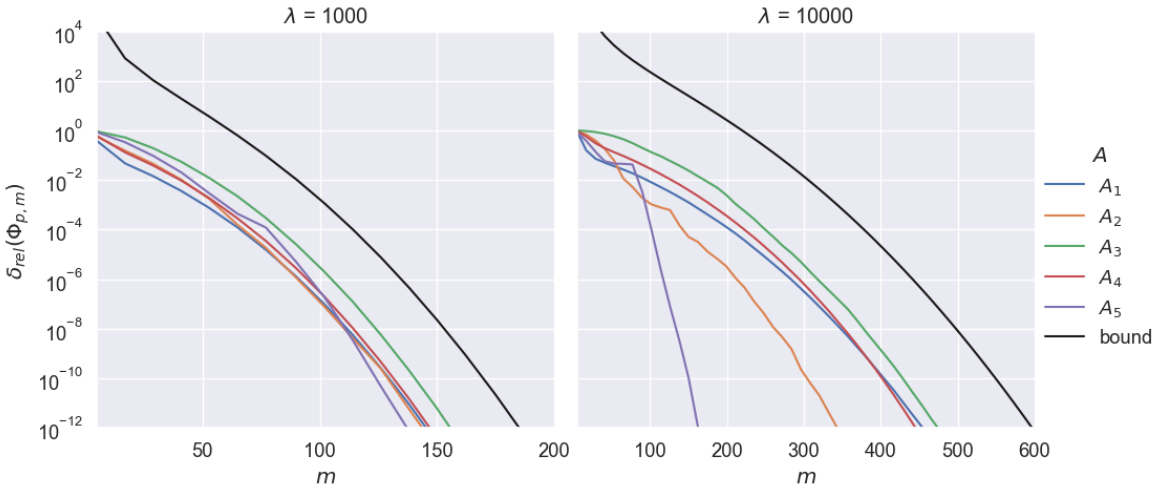


Figure 4.7: Convergence of the approximation (3.19) for φ_1 with different test matrices of size $n = 10^4$ and the same spectral interval. The error estimation in (3.24) is plotted in black.

Figure 4.8 compares the convergence of the rational Krylov subspace method with different number of poles for $\lambda = 10^3$ (top) and $\lambda = 10^5$ (bottom). As expected, the method converges in fewer iterations when it is provided with more poles. However, there is no improvement with more than 15 poles as we can see that the plots of `RA-AAA30` and `RA-AAA15` overlap. This observation is consistent with what has been observed in Figure 4.4 for the scalar approximation errors. From the scalar approximation errors, we also know that rational functions of degrees below 10 cannot approximate the φ-functions well enough. This is a compelling reason for the failure of the methods `RA-AAA3`, `RA-AAA5`, and `RA-AAA7`. With these methods, the relative error of the RA method stagnates after reaching a certain value. However, this is not the case with only one or two repeated poles. The reasoning behind this behaviour needs more consideration but it can originate from the fact that the poles in `RA-AAA1` and `RA-AAA2` are real-valued. Another interesting observation is that for `RA-AAA7` and `RA-AAA10`, the convergence is much faster in the first "turn" of the poles (before they get repeated). It looks like in the second turn and afterwards, the poles do not add as much information to the subspace as they do in the first turn; and therefore, we see a major drop in the convergence rate. `RA-AAA15` and `RA-AAA30` converge before reaching the second turn, so this behaviour cannot be seen for these approximations.

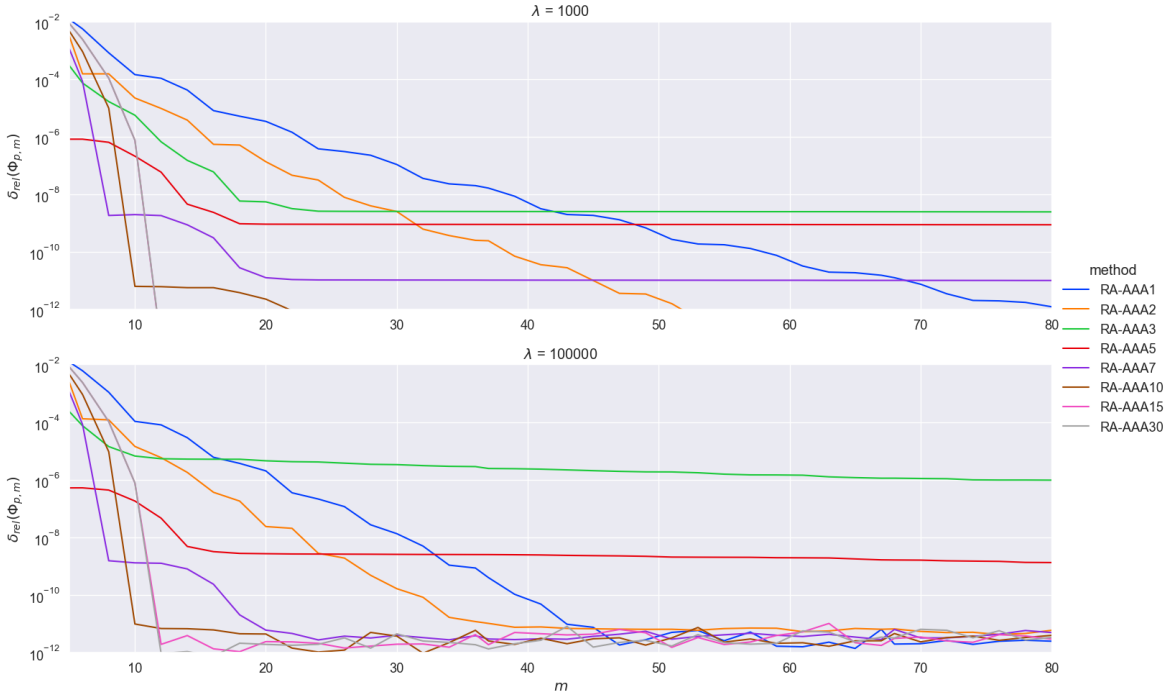


Figure 4.8: Convergence of approximation (3.25) for $\varphi_3(A_1^{n\times n})\mathbf{v}$ with different numbers of repeated poles and $n = 10^4$.

So far, we have seen that using rational Krylov subspaces not only improves the convergence rate, but it also allows for larger λ's, which could be translated to larger time steps in the context of exponential integrators. However, these advantages come at a cost. In Algorithm 2, we need to solve a linear system m times. In the case of repeated poles, the algorithm benefits from pre-computing the LU factorization of the corresponding matrices but it still needs to do the factorization once for each pole. Therefore, for some cases, it might be computationally more efficient to do more iterations of the polynomial Krylov subspace method. To address this issue, the computation time of each method is presented in Figure 4.9 for the matrices A_1 (top row) and A_2 (bottom row) with different smallest eigenvalues and different sizes n. The reported computation time for each method corresponds to the minimum number of iterations (m, dimension of the Krylov subspace) required by the method to reach a relative error of 10^{-10} or smaller. First of all, we can see that the computation time of the reference method (exponential of the embedded matrix), denoted by `EX`, increases when the matrix becomes larger and when its spectral interval becomes wider. Since the reference method implements an scaling and squaring algorithm, it takes longer when the input matrix has a larger norm (larger λ). With the polynomial Krylov subspace method, denoted by `PA`, although the performance is acceptable for some cases, when the input matrix becomes larger, it becomes even slower than the reference method especially for wider spectra. Note that in the right top plot, this method is not present because it does not converge even with 1000 iterations. With the rational Krylov subspace method (denoted by `RA`), on the contrary, the approximations can be computed very fast even for large matrices with wide spectra. However, we can see that when more poles are provided, the method becomes slower even with fewer iterations. The `RA-AAA1` method is always the fastest one despite needing more iterations to converge. It should be noted that since the poles are

pre-computed, the computation time of the poles by the AAA algorithm is not included in the reported CPU times.

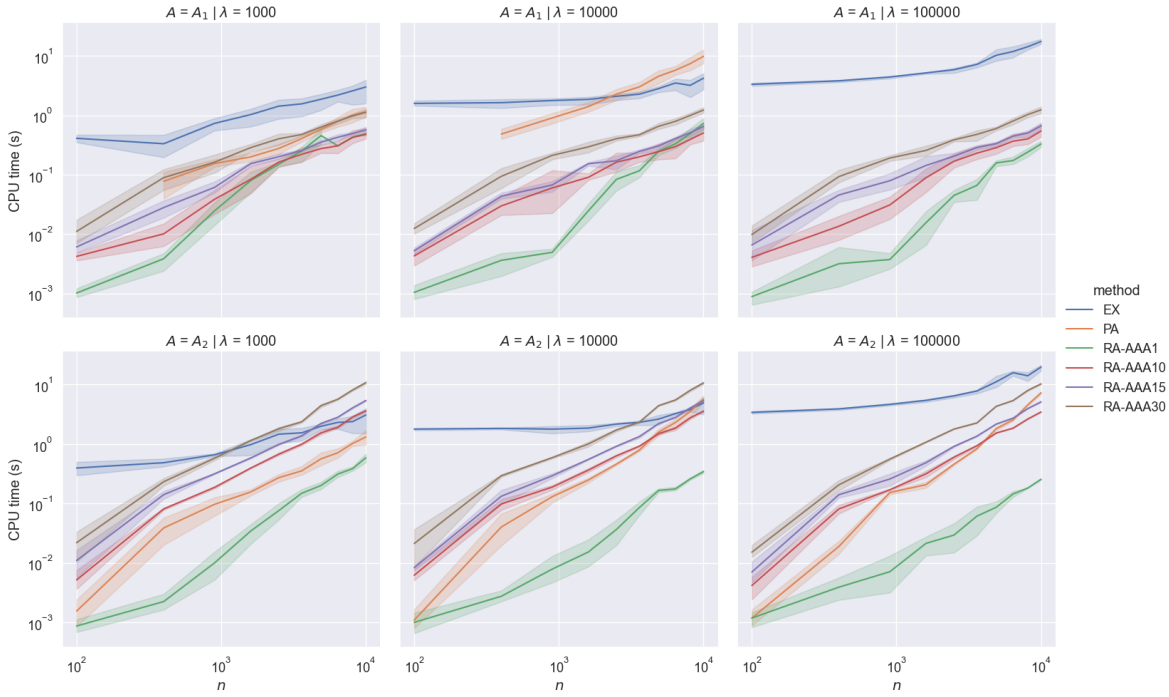


Figure 4.9: CPU time for different evaluation methods. For the approximation methods, the number of iterations required to reach a relative error less than 10^{-10} is considered. Shadows of each line represent the range of the CPU time for φ_0, φ_1, φ_3, and φ_5. `EX` represents the reference evaluation.

4.2 Trigonometric Functions

Using finite elements for solving the PDE corresponding to a 2D linear elasticity problem on a unit square Ω, the following initial value problem is derived in [13]:

$$M\ddot{\mathbf{x}}(t) + K\mathbf{x}(t) = \mathbf{f}(t), \quad \mathbf{x}(0) = \mathbf{u_0'}, \ \dot{\mathbf{x}}(0) = \mathbf{v_1'}, \tag{4.27}$$

where M and K are both symmetric positive definite matrices. Computing the Cholesky factorization of $M = LL^\top$, pre-multiplying the equation by L^{-1}, and a change of variables results in

$$\ddot{\mathbf{u}}(t) + A\mathbf{u}(t) = \mathbf{g}(t), \quad \mathbf{u}(0) = \mathbf{u_0}, \ \dot{\mathbf{u}}(0) = \mathbf{v_0}, \tag{4.28}$$

where

$$\mathbf{u}(t) = L^\top \mathbf{x}(t), \qquad A = L^{-1}KL^{-\top}, \quad \mathbf{g}(t) = L^{-1}\mathbf{f}(t),$$
$$\mathbf{u_0} = \mathbf{u_0'}, \qquad\qquad \mathbf{v_0} = \mathbf{v_0'}.$$

Assuming that $\mathbf{f}(t) = \mathbf{f_0}$ is constant in time, the solution to the problem satisfies

$$\mathbf{u}(t) = \cos\left(t\sqrt{A}\right)\mathbf{u_0} + t\,\mathrm{sinc}(t\sqrt{A})\mathbf{v_0} + \frac{t^2}{2}\mathrm{sinc}^2(\frac{t}{2}\sqrt{A})\mathbf{g_0}, \tag{4.29}$$

where $\mathbf{g_0} = L^{-1}\mathbf{f_0}$.

In this subsection, we investigate the performance of the methods presented in section 3 for approximating the three terms in (4.29) with suitable time steps.

Test Matrices

Since we are studying specific matrix functions for an specific problem, we will stick to the matrices that appear for this problem. The matrices M and K in (4.27) are calculated using a finite element space of degree 1 on a triangulation of the unit square Ω. The matrix $A_{LE}^{0.04} \in \mathbb{R}^{888\times 888}$ is then formed by computing the Cholesky factorization of $M = LL^\top$ and multiplying K by L^{-1} and $L^{-\top}$ from left and right. This matrix inherits the symmetry and the positive definiteness of K but unlike the test matrices in the previous subsection, it will be dense. However, it is possible to modify the Arnoldi process in order to avoid forming this matrix explicitly. The square root of this matrix (computed by `scipy.linalg.sqrtm`) is also symmetric positive definite, and its numerical range is:

$$\mathcal{W}\left(\sqrt{A_{LE}^{0.04}}\right) = [2.37 \times 10^6, 6.01 \times 10^6].$$

Pole Selection for Rational Krylov Subspace Approximation

The pole selection strategy is the same as for φ-functions in subsection 4.1, using the AAA algorithm and repeating the poles. The only difference is that the interpolation points are 2000 uniformly spaced points in the numerical range of the input matrix. The arguments that justified the choice of a logarithmic distribution of the interpolation points are no longer valid for the trigonometric functions.

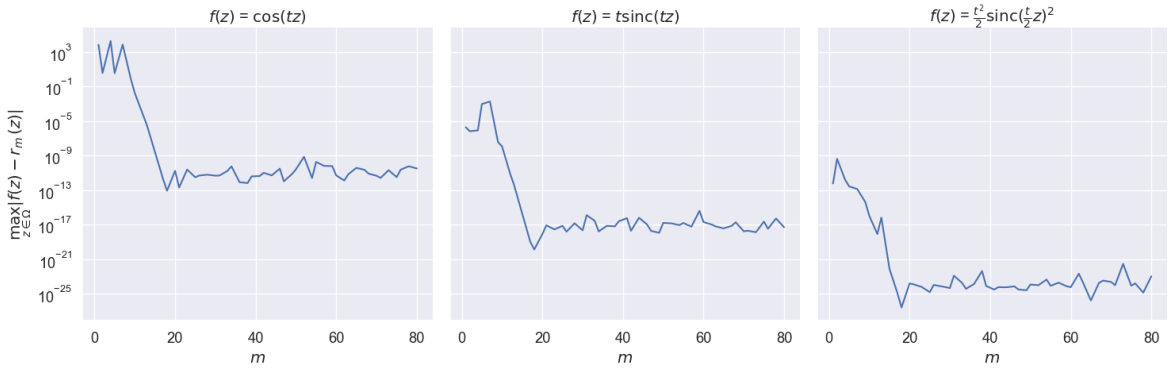


Figure 4.10: Maximum error of the rational approximation of the trigonometric functions with $t = 5 \times 10^{-6}$ on the interval $\Omega = [1 \times 10^6, 6 \times 10^6]$. The approximations are done by the AAA algorithm with 2000 uniformly spaced interpolation points.

The absolute errors of the rational approximations of the scalar functions are presented in Figure 4.10. For all the three functions, the errors do not decrease when the degree of the rational function is too low. This behaviour lies on the oscillatory nature of the trigonometric functions. When the degree is increased, at some point, the rational functions become powerful enough to capture the oscillations and the errors start decreasing. With a degree around 20, the functions

reach their full potential and there is no improvement afterwards. This *minimum sufficient degree* is higher for bigger time steps. For $t = 2 \times 10^{-5}$, for instance, a rational function of degree at least 40 is needed. The reason that the starting absolute errors are small for the functions $f(z) = t\text{sinc}(tz)$ and $f(z) = \frac{t^2}{2}\text{sinc}(\frac{t}{2}z)^2$ is that, because of their leading coefficients, their values are small.

Numerical Results

The methods described in subsection 3.2 (PA) and subsection 3.3 (RA) have been evaluated for the presented test matrices. For all the test matrices, the approximations are carried out for a unit random vector v. For computing the cosine and the sine of the matrices $\hat{H}_m$ and $\hat{A}_m$ in the PA and the RA methods, respectively, the `scipy.linalg.cosm` and `scipy.linalg.sinm` function from the the SciPy library [12] are used which use the `scipy.linalg.expm` function internally. Similarly to subsection 4.1, we take the vectors computed by the methods described in subsection 3.4 as reference and we report the relative error of the results from the Krylov subspace methods of dimension m, denoted by $\mathbf{f_m}$, against the reference vectors, denoted by $\mathbf{f^{EX}}$. The relative error is computed as:

$$\delta_{rel}(\mathbf{f_m}) = \frac{\left\|\mathbf{f_m} - \mathbf{f^{EX}}\right\|_2}{\left\|\mathbf{f^{EX}}\right\|_2}.$$

The relative errors of the approximations with different dimensions of Krylov subspaces are presented in Figure 4.11 with the time steps $t = 5 \times 10^{-6}$ (top) and $t = 2 \times 10^{-5}$ (bottom). With the polynomial Krylov subspace approximation (PA), the error does not decrease when the dimension of the Krylov subspace is too low. This is similar to what has been observed in Figure 4.10 and it is again because of the oscillations of the trigonometric functions. With the rational Krylov subspace approximations (RA), the performance is not satisfactory in general. The main reason is that when only a portion of the poles are provided (i.e., when $m < k$), the rational functions are not able to approximate the functions well. We can see that the relative error of the `RA-AAAk` method suddenly drops when the dimension of the subspace reaches k and all the poles have been used. For this reason, the convergence of this method is delayed by the number of the poles. On the other hand, if k is not large enough, the rational functions are not powerful enough to approximate the scalar functions (see Figure 4.10). This is evident in the bottom row of Figure 4.11 where the method does not converge with 20 poles and it can only reach an error of 10^{-8} with 30 poles.

The observations in the bottom row of Figure 4.11 gives the idea that there might be some potential advantages in the RA method for bigger time steps, if the pole selection strategy is improved. In an attempt for improving the pole selection strategy, we did the same experiments by using the poles from the AAA algorithm only once and using $\xi_j = \infty$ for $j > k$. Since the rational Krylov subspace with infinity poles is identical to the polynomial Krylov subspace, the idea here is to benefit from the poles in the first k iterations and do the same as in polynomial Krylov subspace in the rest of the iterations. The resulting subspace will contain rational functions of the type (m, k) when $m \geq k$. Figure 4.12 shows the relative errors for the time step $t = 2 \times 10^{-5}$. Although there are some improvements when $m > k$ especially for `RA-AAA20`, this strategy did not solve the main problem of the RA method. The convergence still loses momentum when the dimension of the subspace reaches k.

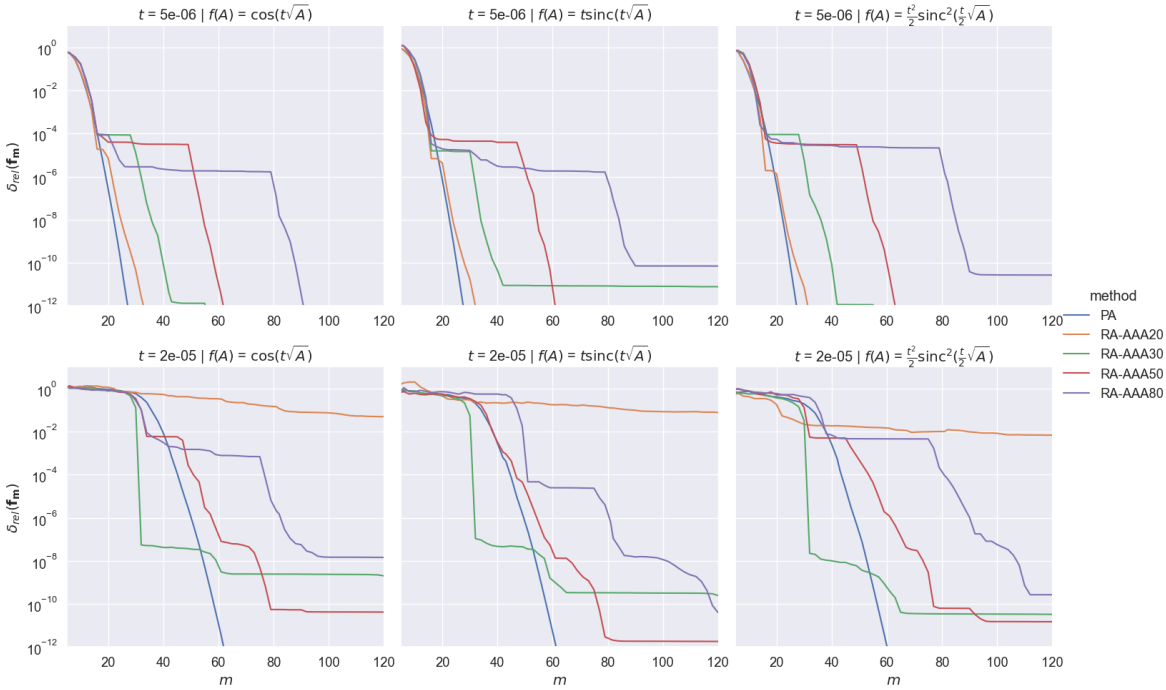


Figure 4.11: Convergence of the approximations (3.19) and (3.25) with rational functions of type (m, m) for $A_{LE}^{0.04}$. Each column shows plots for one function. Each row shows plots for one time step.

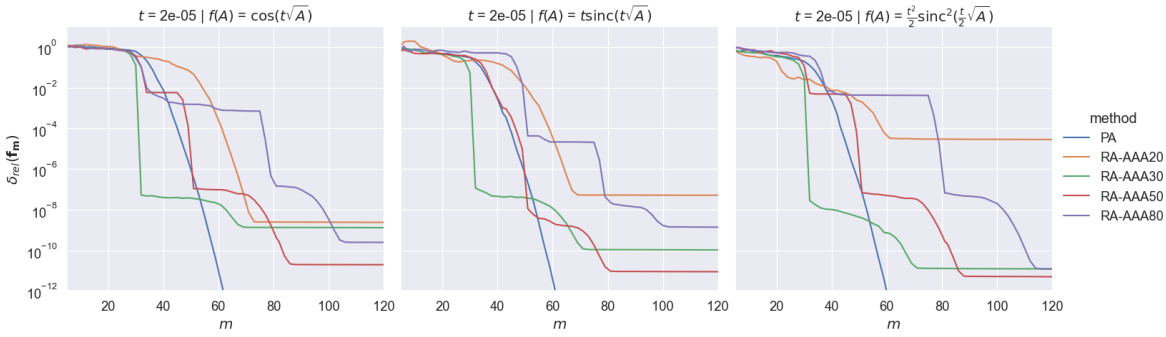


Figure 4.12: Convergence of the approximations (3.19) and (3.25) with rational functions of type (m, k) for $A_{LE}^{0.04}$. Each column shows plots for one function.

5 Conclusion

In this work, we investigated the performance of polynomial and rational Krylov subspace methods for approximating the action of φ-functions and trigonometric matrix functions on a given vector for real symmetric matrices.

For the φ-functions, we focused on the interval $(-\infty, 0]$ and parameterized the numerical range of the input matrix as $[-\lambda, 0]$ with $\lambda > 0$. We studied the convergence for different λ's. In the context of exponential integrators, the spectrum of the input matrix usually depends on parameters (e.g., mesh size) that cannot be changed and it is scaled by the time step of the exponential integrator. Therefore, we can translate our findings about the effect of the parameter λ to the time step in exponential integrators. Here is a summary of the main findings of this work for the φ-functions:

- The approximation of higher order φ-functions with polynomials and rational functions is slightly easier because they change more slowly on the left half complex plane (see Figure 2.1).
- The poles of the rational functions can be pre-computed on a sufficiently large interval and be used for all time steps.
- The rational Krylov subspace method needs fewer iterations to converge. Furthermore, contrary to the polynomial Krylov subspace method, it allows for larger time steps without increasing the number of the iterations (i.e., the dimension of the Krylov subspace).
- The rational Krylov subspace method benefits from poles that are returned by the AAA algorithm. However, one needs to be careful to provide it with a sufficient number of poles since it might not converge with only a few poles, unless they are all real-valued. The performance of the method depends on the error of the scalar rational approximation.
- Regarding the computation time for large sparse matrices, the polynomial approximation performs even slower than the baseline methods (scaling and squaring methods for e^A) when the time step is increased. The rational Krylov subspace method with a few repeated poles should be the method of choice. When more poles are used, the method converges in fewer iterations but it will need to do more LU factorizations in the beginning. The number of the poles can be fine tuned depending on the settings of the problem.

For the trigonometric functions, we focused on the 2D linear elasticity problem. Here is a summary of the main findings for the trigonometric functions:

- Because of the oscillatory nature of the functions that have been studied, the approximation errors do not change much before a minimum dimension of Krylov subspace is reached.
- The polynomial Krylov subspace method needs more iterations to converge when the time step is increased.
- The convergence of the rational Krylov subspace method is delayed at least until the subspace has been fed with all the poles from the AAA algorithm. Also, repeating the poles or using infinity poles afterwards does not help the convergence.

For the rational approximations, we focused on the AAA algorithm for choosing the poles. Since

this algorithm is based on interpolating the scalar function on a set of given points, it has the disadvantage that it does not allow for putting constrains on the poles. For example, in the case of the φ-functions, restricting the poles to real positive values or being far from the negative real axis might be of interest. We saw in Figure 4.8 that the method converges even with only a few repeated poles (`RA-AAA1` and `RA-AAA2`), provided that the poles are real. However, it might not converge if some of the poles are complex. It remains for a future work to investigate the theoretical reason behind this behaviour. In the case of one real repeated pole, the method has been linked to the shift-and-invert method (see Section 7.4 of [3]).

References

[1] A. H. Al-Mohy and N. J. Higham, *A new scaling and squaring algorithm for the matrix exponential*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 970–989.

[2] G. H. Golub and C. F. Van Loan, *Matrix computations*, JHU press, 2013.

[3] S. Güttel, *Rational Krylov methods for operator functions*, PhD thesis, Technische Universität Bergakademie Freiberg, 2010.

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

[5] N. J. Higham, *Functions of matrices: theory and computation*, SIAM, 2008.

[6] C. Hofreither, *An algorithm for best rational approximation based on barycentric rational interpolation*, Numerical Algorithms, 88 (2021), pp. 365–388.

[7] D. Kressner, *A Krylov subspace method for the approximation of bivariate matrix functions*, Structured Matrices in Numerical Linear Algebra: Analysis, Algorithms and Applications, (2019), pp. 197–214.

[8] B. V. Minchev and W. Wright, *A review of exponential integrators for first order semi-linear problems*, Preprint, NTNU Trondheim, (2005).

[9] Y. Nakatsukasa, O. Sète, and L. N. Trefethen, *The AAA algorithm for rational approximation*, SIAM Journal on Scientific Computing, 40 (2018), pp. A1494–A1522.

[10] J. Niesen and W. M. Wright, *Algorithm 919: A Krylov subspace algorithm for evaluating the φ-functions appearing in exponential integrators*, ACM Transactions on Mathematical Software, 38 (2012), pp. 1–19.

[11] L. N. Trefethen and D. Bau III, *Numerical linear algebra, vol. 50*, 1997.

[12] P. Virtanen, R. Gommers, T. E. Oliphant, et al., *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods, 17 (2020), pp. 261–272.

[13] Y. Voet, *On the computation of matrix functions for the finite element solution of linear elasticity problems in dynamics*, tech. rep., École Polytechnique Fédérale de Lausanne, 2020.