MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

CSC COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

A scaling and recovering algorithm for the matrix φ-functions

Xiaobo Liu
MPI for Dynamics of Complex Technical Systems, Germany

2026 GAMM Annual Meeting
Universität Stuttgart, Germany
March 18, 2026

Joint work with **Awad H. Al-Mohy**
King Khalid University, Saudi Arabia

One of the most studied matrix functions by far is the matrix exponential

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots = \sum_{k=0}^{\infty} \frac{A^k}{k!};$$

see (**Moler & Van Loan; '78, '03**)

Nineteen dubious ways to compute the exponential of a matrix (1978); —, twenty-five years later (2003).

Closely related are the matrix φ-functions

$$\varphi_0(A) = e^A, \quad \varphi_j(A) = \sum_{k=0}^{\infty} \frac{A^k}{(k+j)!}.$$

▷ φ_j satisfy the recurrence relation

$$\varphi_j(A) = A\varphi_{j+1}(A) + \frac{1}{j!}I.$$

For $A \in \mathbb{C}^{n\times n}$, $y \in \mathbb{C}^n$, and $y(0) = y_0$ (**Minchev & Wright, '05**),

$$\frac{\mathrm{d}y}{\mathrm{d}t} = Ay \quad \Rightarrow \quad y(t) = \mathrm{e}^{tA} y_0,$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = Ay + b + ct \quad \Rightarrow \quad y(t) = \mathrm{e}^{tA} y_0 + t\varphi_1(tA)b + t^2\varphi_2(tA)c,$$

$$\vdots$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = Ay + g(t,y) \quad \Rightarrow \quad y(t) = \mathrm{e}^{tA} y_0 + \sum_{k=1}^{\infty} \varphi_k(tA)\, t^k g^{(k-1)}(0, y_0).$$

- A suitable truncation forms the basis of many exponential integrators.

For $A \in \mathbb{C}^{n\times n}$, $y \in \mathbb{C}^n$, and $y(0) = y_0$ (**Minchev & Wright, '05**),

$$\frac{\mathrm{d}y}{\mathrm{d}t} = Ay \quad \Rightarrow \quad y(t) = \mathrm{e}^{tA}y_0,$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = Ay + b + ct \quad \Rightarrow \quad y(t) = \mathrm{e}^{tA}y_0 + t\varphi_1(tA)b + t^2\varphi_2(tA)c,$$

$$\vdots$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = Ay + g(t,y) \quad \Rightarrow \quad y(t) = \mathrm{e}^{tA}y_0 + \sum_{k=1}^{\infty} \varphi_k\,(tA)\,t^k g^{(k-1)}(0, y_0).$$

- A suitable truncation forms the basis of many exponential integrators.

E.g.: exponential Rosenbrock–Euler method (**Pope, '63**):

$$y_{n+1} = y_n + h\varphi_1(hJ_n)F(t_n, y_n) + h^2\varphi_2(hJ_n)\frac{\partial F}{\partial t}(t_n, y_n), \qquad J_n = \frac{\partial F}{\partial y}(t_n, y_n),$$

where $F(t,y) \equiv Ay + g(t,y)$, $y_n \approx y(t_n)$, $t_n = nh$, and $h > 0$ stepsize.

Krylov-Based Exponential Integrators

Different types of exponential integrator schemes can be expressed as

$$\varphi_0(A)w_0 + \varphi_1(A)w_1 + \cdots + \varphi_p(A)w_p,$$

where w_j some vector, p relates to the exponential integrator order.

[Compute $\varphi_j(A)w_j$ by (polynomial) Krylov method]:

1. Initialize $v_1 = w_j/\|w_j\|_2$.
2. Build orthonormal basis $V_m = [v_1, v_2, \ldots, v_m]$ of $\mathcal{K}_m(A, w_j) = \mathrm{span}\{w_j, Aw_j, \ldots, A^{m-1}w_j\}$ via the Arnoldi process:

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

3. Project $\varphi_j(A)w_j$ onto a lower-dim. subspace (**Hochbruck & Lubich, '97**):

$$\varphi_j(A)w_j \approx \frac{\|w_j\|_2}{2\pi \mathrm{i}} \int_\Gamma \varphi_j(\sigma) V_m (\sigma I - H_m)^{-1} e_1 \mathrm{d}\sigma = \|w_j\|_2 V_m \varphi_j(H_m) e_1.$$

▷ Accurately and stably computing $\varphi_j(H_m)$ a key and challenging step.

φ-Functions from One Exponentiation

Theorem (Lu, '03; Higham, '08)

$$B = \begin{bmatrix} A & I \\ 0 & 0 \end{bmatrix} \in \mathbb{C}^{2n\times 2n} \quad \Rightarrow \quad \mathrm{e}^B = \begin{bmatrix} \mathrm{e}^A & \varphi_1(A) \\ 0 & I \end{bmatrix}.$$

φ-Functions from One Exponentiation

Theorem (Lu, '03; Higham, '08)

$$B = \begin{bmatrix} A & I \\ 0 & 0 \end{bmatrix} \in \mathbb{C}^{2n\times 2n} \quad \Rightarrow \quad e^B = \begin{bmatrix} e^A & \varphi_1(A) \\ 0 & I \end{bmatrix}.$$

Theorem (Al-Mohy & L., '25)

Let $E = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{n\times np}$ *and* $J = J_p(0) \otimes I$. *Then*

$$W = \begin{bmatrix} A & E \\ 0 & J \end{bmatrix} \in \mathbb{C}^{(p+1)n\times(p+1)n} \;\Rightarrow\; f(W) = \begin{bmatrix} f(A) & g_1(A) \;\ldots\; g_p(A) \\ 0 & f(J_p(0)) \otimes I \end{bmatrix},$$

where $g_i(A) = Ag_{i+1}(A) + f^{(i)}(0)I/i!, \quad g_0 = f, \quad i = 0: p-1.$

φ-Functions from One Exponentiation

Theorem (Lu, '03; Higham, '08)

$$B = \begin{bmatrix} A & I \\ 0 & 0 \end{bmatrix} \in \mathbb{C}^{2n\times 2n} \quad \Rightarrow \quad e^B = \begin{bmatrix} e^A & \varphi_1(A) \\ 0 & I \end{bmatrix}.$$

Theorem (Al-Mohy & L., '25)

Let $E = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{n\times np}$ *and* $J = J_p(0) \otimes I$. *Then*

$$W = \begin{bmatrix} A & E \\ 0 & J \end{bmatrix} \in \mathbb{C}^{(p+1)n\times(p+1)n} \Rightarrow f(W) = \begin{bmatrix} f(A) & g_1(A) \ \dots \ g_p(A) \\ 0 & f(J_p(0)) \otimes I \end{bmatrix},$$

where $g_i(A) = Ag_{i+1}(A) + f^{(i)}(0)I/i!, \quad g_0 = f, \quad i = 0\colon p-1.$

$\rightsquigarrow$ Corollary: setting $f = \exp \ \Rightarrow \ g_1(A) = \varphi_1(A), \dots, g_p(A) = \varphi_p(A)$.

- Evaluate the first block row of e^W *without* forming W?

Theorem (Al-Mohy & L., '25)

The $[m+p-j/m]$ *Padé approximants to* $\varphi_j(z)$, $\mathcal{R}_{m+p-j,m}(z) =: \mathcal{R}_m^{(j)}(z)$, $j = 1\colon p$, *satisfy the recurrence relation*

$$\mathcal{R}_m^{(j)}(z) = z\mathcal{R}_m^{(j+1)}(z) + \frac{1}{j!}, \quad j = p-1\colon -1\colon 0.$$

Idea of proof: set f as the $[m+p/m]$ Padé approximant, $\mathcal{R}_m^{(0)}(z)$, to e^z.

Theorem (Al-Mohy & L., '25)

The $[m+p-j/m]$ *Padé approximants to* $\varphi_j(z)$, $\mathcal{R}_{m+p-j,m}(z) =: \mathcal{R}_m^{(j)}(z)$, $j = 1\colon p$, *satisfy the recurrence relation*

$$\mathcal{R}_m^{(j)}(z) = z\mathcal{R}_m^{(j+1)}(z) + \frac{1}{j!}, \quad j = p-1 \colon -1 \colon 0.$$

Idea of proof: set f as the $[m+p/m]$ Padé approximant, $\mathcal{R}_m^{(0)}(z)$, to e^z.

• Adapt the Padé approximant degree per φ_j and fix the denominator in our new algorithm

$\rightsquigarrow$ Computational savings over evaluating fixed-degree Padé approximants to φ-functions independently, avoiding repeated rational approximations

Scaling and Squaring (Recovering) Method

For the matrix exponential: $e^A = (e^{2^{-s}A})^{2^s} \approx r_m(2^{-s}A)^{2^s}$

---

1 $B \leftarrow A/2^s$ so $\|B\|_1$ is close to the origin. ▷ scaling
2 Evaluate the $[m/m]$ Padé approximant $r_m(B)$ to e^B. ▷ evaluate
3 $e^A \approx r_m(B)^{2^s}$ ▷ squaring

- Proposed by Lawson (**Lawson, '67**).
- Algorithm, with rounding error analysis and a posteriori error bound (**Ward, '77**).
- Backward truncation error analysis allowing choice of s and m (**Moler & Van Loan, '78**).
- Sharper b'ward trunc. analysis giving optimal s and m minimizing computational cost in double precision (**Higham, '05**).
- B'ward trunc. bound based on $\|A^k\|^{1/k}$ rather than $\|A\|$, alleviating overscaling issues (**Al-Mohy & Higham, '09**), current MATLAB `expm`.

Simultaneously for the φ-functions: $\varphi_0(A) = \mathrm{e}^A$, $\varphi_1(A)$, ..., $\varphi_p(A)$

Select a scaling parameter s and a degree m of Padé approximant.
$A \leftarrow A/2^s$
Evaluate the $[m/m]$ Padé approximant, $\mathcal{R}_m^{(p)}$, to $\varphi_p(A)$.
Invoke the recurrence $\mathcal{R}_m^{(j)} = A\mathcal{R}_m^{(j+1)} + \frac{1}{j!}I$, $j = p-1:-1:0$.
for $i = 1: s$ **do**
    for $j = p: -1 : 0$ **do**
        $\mathcal{R}_m^{(j)} \leftarrow 2^{-j}\left(\mathcal{R}_m^{(0)}\mathcal{R}_m^{(j)}(A) + \sum_{k=1}^{j} \mathcal{R}_m^{(k)}/(j-k)!\right)$
    end
end

- The double-argument formula $\varphi_j(2A) = \frac{1}{2^j}\left(\varphi_0(A)\varphi_j(A) + \sum_{k=1}^{j} \frac{\varphi_k(A)}{(j-k)!}\right)$ **(Berland, Skaflestad, Wright; '07)** for recov. $\varphi_j(A)$ from $\varphi_j(2^{-s}A)$, $j = 0: p$.

Scaling and Recovering Framework for φ-Functions

Simultaneously for the φ-functions: $\varphi_0(A) = \mathrm{e}^A, \varphi_1(A), \ldots, \varphi_p(A)$

Select a scaling parameter s and a degree m of Padé approximant.
$A \leftarrow A/2^s$
Evaluate the $[m/m]$ Padé approximant, $\mathcal{R}_m^{(p)}$, to $\varphi_p(A)$.
Invoke the recurrence $\mathcal{R}_m^{(j)} = A\mathcal{R}_m^{(j+1)} + \frac{1}{j!}I$, $j = p-1 : -1 : 0$.
for $i = 1 : s$ **do**
  for $j = p : -1 : 0$ **do**
    $\mathcal{R}_m^{(j)} \leftarrow 2^{-j} \left(\mathcal{R}_m^{(0)} \mathcal{R}_m^{(j)}(A) + \sum_{k=1}^{j} \mathcal{R}_m^{(k)}/(j-k)! \right)$
  end
end

- The double-argument formula $\varphi_j(2A) = \frac{1}{2^j}\left(\varphi_0(A)\varphi_j(A) + \sum_{k=1}^{j} \frac{\varphi_k(A)}{(j-k)!}\right)$ **(Berland, Skaflestad, Wright; '07)** for recov. $\varphi_j(A)$ from $\varphi_j(2^{-s}A)$, $j = 0 : p$.

$\rightsquigarrow$ Determine s and m to *guarantee accuracy* and *minimize comput'nal cost*.

Truncation Error Analysis: A Sketch

- We're computing the first block row of $\exp\left(\begin{bmatrix} A+\Delta A & E+\Delta E \\ 0 & J+\Delta J \end{bmatrix}\right)$.

Key: Set f as the $[m+p/m]$ Padé approximant, $\mathcal{R}_m^{(0)}(z)$, to e^z:

$$W = \begin{bmatrix} A & E \\ 0 & J \end{bmatrix} \quad \Rightarrow \quad \mathcal{R}_m^{(0)}(W) = \begin{bmatrix} \mathcal{R}_m^{(0)}(A) & \mathcal{R}_m^{(1)}(A) \ \ldots \ \mathcal{R}_m^{(p)}(A) \\ 0 & \mathcal{R}_m^{(0)}(J) \end{bmatrix}.$$

Truncation Error Analysis: A Sketch

- We're computing the first block row of $\exp\left(\begin{bmatrix} A+\Delta A & E+\Delta E \\ 0 & J+\Delta J \end{bmatrix}\right)$.

Key: Set f as the $[m+p/m]$ Padé approximant, $\mathcal{R}_m^{(0)}(z)$, to e^z:

$$W = \begin{bmatrix} A & E \\ 0 & J \end{bmatrix} \quad \Rightarrow \quad \mathcal{R}_m^{(0)}(W) = \begin{bmatrix} \mathcal{R}_m^{(0)}(A) & \mathcal{R}_m^{(1)}(A) \ \dots \ \mathcal{R}_m^{(p)}(A) \\ 0 & \mathcal{R}_m^{(0)}(J) \end{bmatrix}.$$

- Analysis uses $[m+p/m]$ Padé approximant, $\mathcal{R}_m^{(0)}(A)$ to e^A (**Higham, '08**):

$$\left[\mathcal{R}_m^{(0)}(2^{-s}A)\right]^{2^s} = \mathrm{e}^{A+2^s h_{m,p}(2^{-s}A)} =: \mathrm{e}^{A+\Delta A},$$

where $h_{m,p}(A) = \log(\mathrm{e}^{-A}\mathcal{R}_m^{(0)}(A)) = \sum_{k=2m+p+1}^{\infty} c_{m,p,k}A^k$. Then $\Delta A = 2^s h_{m,p}(2^{-s}A)$ represents the backward error for the exponential.

Truncation Error Analysis: A Sketch

- We're computing the first block row of $\exp\left(\begin{bmatrix} A+\Delta A & E+\Delta E \\ 0 & J+\Delta J \end{bmatrix}\right)$.

Key: Set f as the $[m+p/m]$ Padé approximant, $\mathcal{R}_m^{(0)}(z)$, to e^z:

$$W = \begin{bmatrix} A & E \\ 0 & J \end{bmatrix} \quad \Rightarrow \quad \mathcal{R}_m^{(0)}(W) = \begin{bmatrix} \mathcal{R}_m^{(0)}(A) & \mathcal{R}_m^{(1)}(A) \ \dots \ \mathcal{R}_m^{(p)}(A) \\ 0 & \mathcal{R}_m^{(0)}(J) \end{bmatrix}.$$

- Analysis uses $[m+p/m]$ Padé approximant, $\mathcal{R}_m^{(0)}(A)$ to e^A **(Higham, '08)**:

$$\left[\mathcal{R}_m^{(0)}(2^{-s}A)\right]^{2^s} = \mathrm{e}^{A+2^s h_{m,p}(2^{-s}A)} =: \mathrm{e}^{A+\Delta A},$$

where $h_{m,p}(A) = \log(\mathrm{e}^{-A}\mathcal{R}_m^{(0)}(A)) = \sum_{k=2m+p+1}^{\infty} c_{m,p,k}A^k$. Then $\Delta A = 2^s h_{m,p}(2^{-s}A)$ represents the backward error for the exponential.

$\rightsquigarrow$ Need bound $\|\sum_{k=2m+p+1}^{\infty} c_{m,p,k}(2^{-s}A)^k\|$ (not shown for ΔE)

Table of the Scaling Threshold $\theta_{m,p}$

Table: Scaling threshold $\theta_{m_i,p}$, at optimal Paterson–Stockmeyer degrees m_i.

p	$m_0 = 1$	$m_1 = 2$	$m_2 = 3$	$m_3 = 4$	$m_4 = 6$	$m_5 = 8$	$m_6 = 10$	$m_7 = 12$
1	2.00e-5	3.81e-3	3.97e-2	1.54e-1	7.26e-1	1.76	3.17	4.87
2	3.76e-5	6.09e-3	5.81e-2	2.13e-1	9.28e-1	2.06	3.54	5.28
3	7.37e-5	9.87e-3	8.53e-2	2.94e-1	1.16	2.37	3.91	5.69
4	1.50e-4	1.62e-2	1.26e-1	4.06e-1	1.40	2.69	4.28	6.09
5	3.15e-4	2.70e-2	1.87e-1	5.62e-1	1.66	3.01	4.65	6.50
6	6.86e-4	4.55e-2	2.80e-1	7.79e-1	1.92	3.34	5.02	6.90
7	1.54e-3	7.75e-2	4.18e-1	1.05	2.20	3.68	5.40	7.30
8	3.54e-3	1.33e-1	6.26e-1	1.26	2.48	4.01	5.77	7.69
9	8.35e-3	2.30e-1	9.34e-1	1.48	2.77	4.35	6.14	8.08
10	2.01e-2	3.99e-1	1.16	1.71	3.07	4.69	6.51	8.47

- Scale A down by 2^s such that $2^{-s}\alpha_r(A) \le \theta_{m,p}$, $\rho(A) \le \alpha_r(A) \le \|A\|$.

Next: choose *feasible* s and m to minimize the computational cost.

Minimizing the Computational Cost

Select a scaling parameter s and a degree m of Padé approximant.
$A \leftarrow A/2^s$
Evaluate the $[m/m]$ Padé approximant, $\mathcal{R}_m^{(p)}$, to $\varphi_p(A)$.
Invoke the recurrence $\mathcal{R}_m^{(j)} = A\mathcal{R}_m^{(j+1)} + \frac{1}{j!}I$, $j = p-1 : -1 : 0$.
for $i = 1 : s$ **do**
    for $j = p : -1 : 0$ **do**
        $\mathcal{R}_m^{(j)} \leftarrow 2^{-j}\left(\mathcal{R}_m^{(0)}\mathcal{R}_m^{(j)} + \sum_{k=1}^{j}\mathcal{R}_m^{(k)}/(j-k)!\right)$
    end
end

Total cost is

$$C_{m_i,s} = i + p + \frac{4}{3} + s(p+1),$$

where $s = \max\left(\lceil \log_2\left(\alpha_r(A)/\theta_{m,p}\right)\rceil, 0\right)$ s.t. $2^{-s}\alpha_r(A) \le \theta_{m,p}$.

Minimizing the Computational Cost

Select a scaling parameter s and a degree m of Padé approximant.
$A \leftarrow A/2^s$
Evaluate the $[m/m]$ Padé approximant, $\mathcal{R}_m^{(p)}$, to $\varphi_p(A)$.
Invoke the recurrence $\mathcal{R}_m^{(j)} = A\mathcal{R}_m^{(j+1)} + \frac{1}{j!}I$, $j = p-1:-1:0$.
for $i = 1: s$ **do**
  for $j = p: -1 : 0$ **do**
    $\mathcal{R}_m^{(j)} \leftarrow 2^{-j}\left(\mathcal{R}_m^{(0)}\mathcal{R}_m^{(j)} + \sum_{k=1}^{j} \mathcal{R}_m^{(k)}/(j-k)!\right)$
  end
end

Total cost is

$$C_{m_i,s} = i + p + \frac{4}{3} + s(p+1),$$

where $s = \max\left(\lceil \log_2(\alpha_r(A)/\theta_{m,p}) \rceil, 0\right)$ s.t. $2^{-s}\alpha_r(A) \leq \theta_{m,p}$.

$\rightsquigarrow$ Cost minimized over the index pair (m_i, r), $m_i \leq m_{\max} = 12$ s.t. $\kappa(D_{m_i}(A))$ modest, $2 \leq r \leq r_{\max}$, $r_{\max}$ determined when $m_{\max}$ fixed.

Experimental setup

Algorithms:

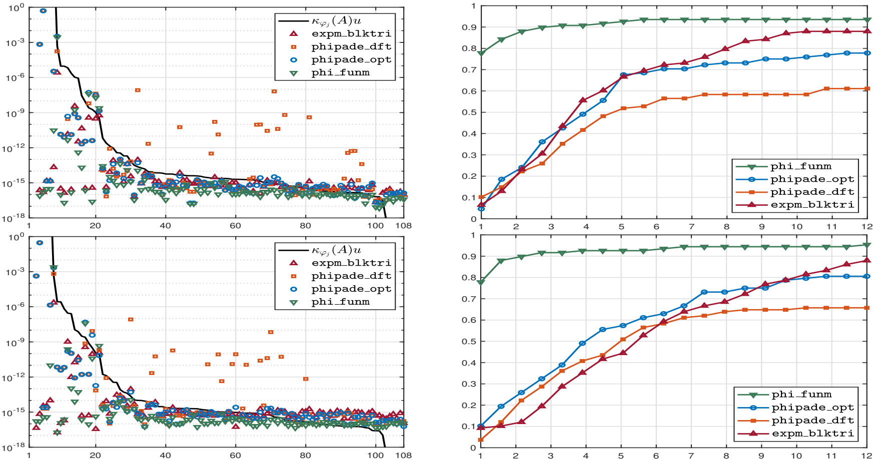
- `phi_funm`: the proposed algorithm;
- `phipade`: from EXPINT (**Berland, Skaflestad, Wright; '07**), realizing the algorithm (**Skaflestad & Wright, '09**) executed in two configurations:
  - `phipade_dft`: the default setting: uses the $[7/7]$ Padé approximant;
  - `phipade_opt`: the adaptive setting proposed in (**Skaflestad & Wright, '09**), seeking the optimal Padé degree $3 \le m \le 13$ to minimize the cost;
- `expm_blktri`: the algorithm of (**Al-Mohy, '25**), for computing the exp of block triangular matrices. (Comput. $(1, 2)$ block of e^W implicitly, not exploiting the structure within W.)

Test set:

- 108 *nonnormal* matrices from matrix function literature, $2 \le n \le 41$.
- 5 Hessenberg matrices ($m = 30$, 80) from Arnoldi within Krylov methods for comput. $\varphi_j(A)e$, $e =$ `ones(n,1)`, $900 \le n \le 392,257$.

Working precision: binary64 (IEEE double) in MATLAB

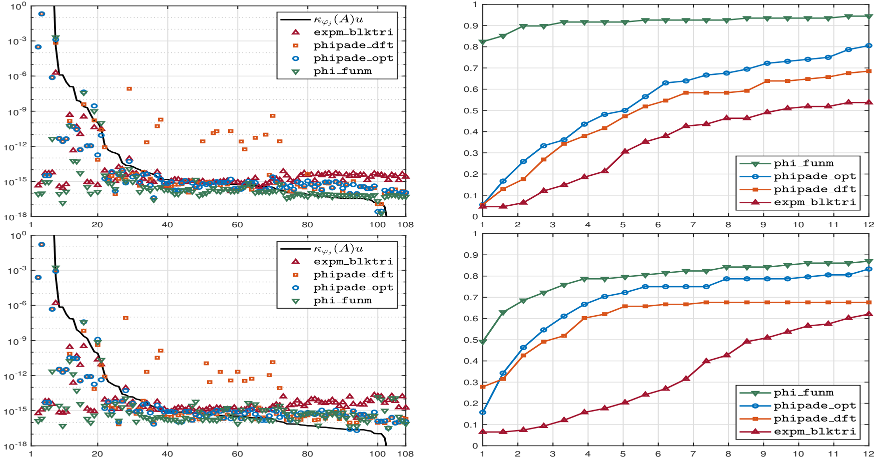
Accuracy and Stability: I



Top: $j = 1$; bottom: $j = 4$.

- $p = 10$. Left: 1-norm forward error. Right: performance profiles: fraction of tests whose relative error is within factor (x-axis) of the smallest.

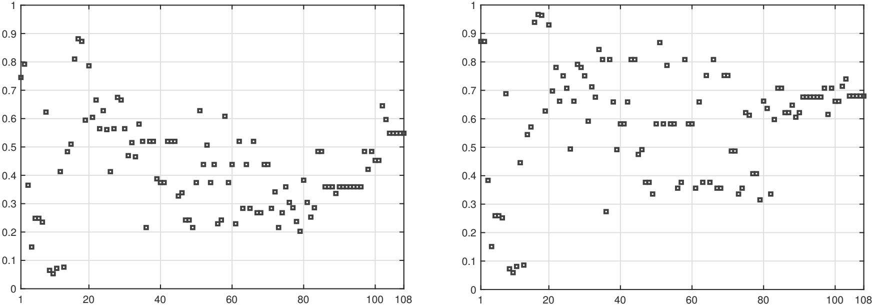
Accuracy and Stability: II



Top: $j = 7$; bottom: $j = 10$.

- phipade_dft unstable (overscaling), expm_blktri worsen as j grows.

Asymptotic Computational Cost

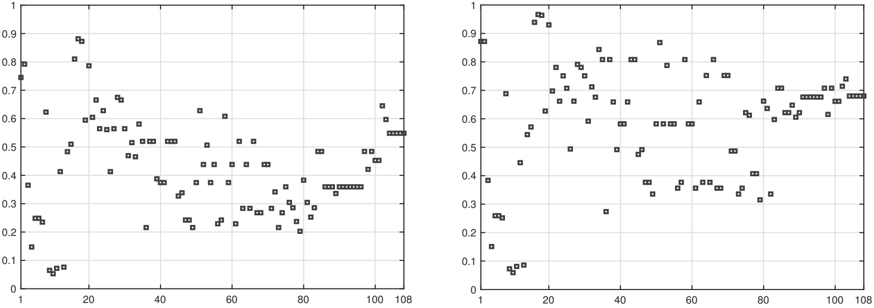


(a) phi_funm over phipade_dft. (b) phi_funm over phipade_opt.

Figure: Ratios of computational cost: phi_funm to phipade.

- phi_funm *always* more efficient, can be $10\times$ to $20\times$ cheaper:
  - based on $\alpha_r(A)$ rather than $\|A\|_1 \rightsquigarrow$ less prone to overscaling.
  - adapts the Padé degree per $\varphi_j \rightsquigarrow$ saves at least p multiple linear-system solvers.

Asymptotic Computational Cost

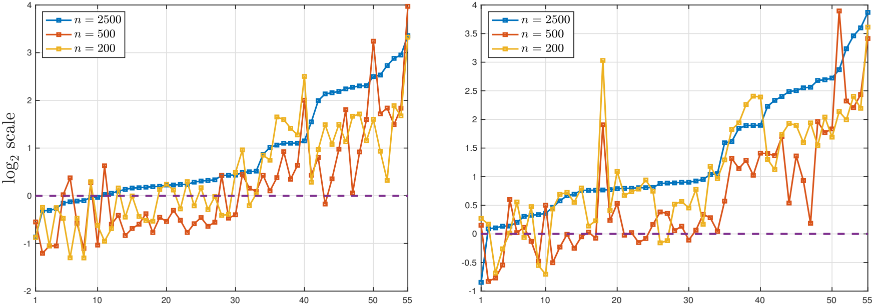


(a) `phi_funm` over `phipade_dft`. (b) `phi_funm` over `phipade_opt`.

Figure: Ratios of computational cost: `phi_funm` to `phipade`.

- `phi_funm` *always* more efficient, can be $10\times$ to $20\times$ cheaper:
  - based on $\alpha_r(A)$ rather than $\|A\|_1 \rightsquigarrow$ less prone to overscaling.
  - adapts the Padé degree per $\varphi_j \rightsquigarrow$ saves at least p multiple linear-system solvers.

Runtime Comparison

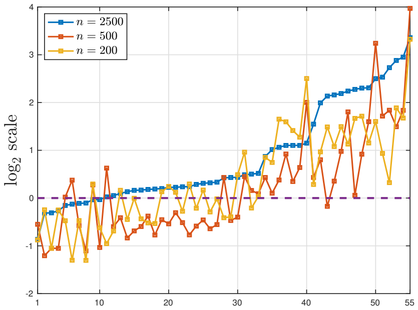


(a) phi_funm vs phipade_dft. (b) phi_funm vs phipade_opt.

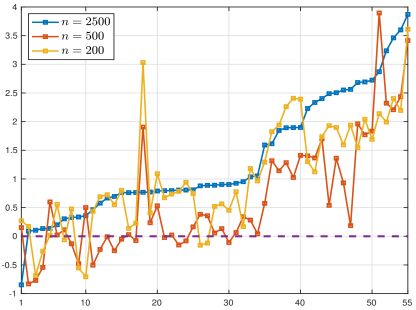
Figure: $\log_2$-speedup of phi_funm to phipade. $y \geq 0$ indicate phi_funm is faster.

- For $n = O(100)$, phi_funm and phipade comparable in speed.
- Lower asymptotic cost of phi_funm reflected in runtime as n grows.
- phi_funm more *reliable*: runtimes $\lesssim 2\times$ the fastest, $16\times$ for phipade.

Runtime Comparison



(a) `phi_funm` vs `phipade_dft`.



(b) `phi_funm` vs `phipade_opt`.

Figure: $\log_2$-speedup of `phi_funm` to `phipade`. $y \geq 0$ indicate `phi_funm` is faster.

- For $n = O(100)$, `phi_funm` and `phipade` comparable in speed.
- Lower asymptotic cost of `phi_funm` reflected in runtime as n grows.
- `phi_funm` more *reliable*: runtimes $\lesssim 2\times$ the fastest, $16\times$ for `phipade`.

Hessenberg Matrices from Krylov Methods

		bcspwr10		gr_30_30		helm2d03		orani678		poisson99	
$p = 1$		Error	Cost	Error	Cost	Error	Cost	Error	Cost	Error	Cost
$m = 30$	phi_funm	3.2e-15	11.3	1.0e-15	12.3	1.4e-15	12.3	3.7e-16	8.3	7.5e-14	34.3
	phipade_dft	5.3e-9	14.7	1.1e-9	16.7	1.9e-10	16.7	5.7e-16	14.7	8.2e-14	38.7
	phipade_opt	2.0e-15	13.7	3.3e-15	15.7	2.1e-15	15.7	7.4e-16	13.7	7.8e-14	35.7
$m = 80$	phi_funm	3.2e-15	11.3	1.0e-15	12.3	1.5e-15	12.3	4.9e-16	8.3	9.1e-14	34.3
	phipade_dft	5.3e-9	14.7	3.4e-14	18.7	1.9e-10	16.7	6.4e-16	18.7	1.1e-13	38.7
	phipade_opt	2.0e-15	13.7	1.8e-15	15.7	2.1e-15	15.7	9.8e-16	15.7	9.9e-14	35.7
$p = 4$		Error	Cost	Error	Cost	Error	Cost	Error	Cost	Error	Cost
$m = 30$	phi_funm	1.1e-15	16.3	8.2e-15	17.3	4.0e-15	17.3	6.8e-16	10.3	1.5e-14	72.3
	phipade_dft	5.0e-10	27.7	1.1e-9	32.7	1.8e-10	32.7	3.6e-16	27.7	5.4e-14	87.7
	phipade_opt	1.3e-15	23.7	3.1e-15	28.7	1.9e-15	28.7	4.5e-16	23.7	5.2e-14	78.7
$m = 80$	phi_funm	1.1e-15	16.3	8.8e-15	17.3	4.1e-15	17.3	8.6e-16	10.3	2.0e-14	72.3
	phipade_dft	5.0e-10	27.7	3.3e-14	37.7	1.8e-10	32.7	1.3e-15	37.7	6.4e-14	87.7
	phipade_opt	1.3e-15	23.7	1.6e-15	28.7	1.9e-15	28.7	1.1e-15	28.7	5.9e-14	78.7

- phi_funm always has lower *computational cost*. Execution times comparable between 10^{-3} and 10^{-2} seconds (not shown).
- phi_funm and phipade_opt deliver good and comparable accuracy, but phipade_dft again shows *instability*.

Hessenberg Matrices from Krylov Methods

		bcspwr10		gr_30_30		helm2d03		orani678		poisson99	
$p = 1$		Error	Cost	Error	Cost	Error	Cost	Error	Cost	Error	Cost
$m = 30$	phi_funm	3.2e-15	11.3	1.0e-15	12.3	1.4e-15	12.3	3.7e-16	8.3	7.5e-14	34.3
	phipade_dft	5.3e-9	14.7	1.1e-9	16.7	1.9e-10	16.7	5.7e-16	14.7	8.2e-14	38.7
	phipade_opt	2.0e-15	13.7	3.3e-15	15.7	2.1e-15	15.7	7.4e-16	13.7	7.8e-14	35.7
$m = 80$	phi_funm	3.2e-15	11.3	1.0e-15	12.3	1.5e-15	12.3	4.9e-16	8.3	9.1e-14	34.3
	phipade_dft	5.3e-9	14.7	3.4e-14	18.7	1.9e-10	16.7	6.4e-16	18.7	1.1e-13	38.7
	phipade_opt	2.0e-15	13.7	1.8e-15	15.7	2.1e-15	15.7	9.8e-16	15.7	9.9e-14	35.7
$p = 4$		Error	Cost	Error	Cost	Error	Cost	Error	Cost	Error	Cost
$m = 30$	phi_funm	1.1e-15	16.3	8.2e-15	17.3	4.0e-15	17.3	6.8e-16	10.3	1.5e-14	72.3
	phipade_dft	5.0e-10	27.7	1.1e-9	32.7	1.8e-10	32.7	3.6e-16	27.7	5.4e-14	87.7
	phipade_opt	1.3e-15	23.7	3.1e-15	28.7	1.9e-15	28.7	4.5e-16	23.7	5.2e-14	78.7
$m = 80$	phi_funm	1.1e-15	16.3	8.8e-15	17.3	4.1e-15	17.3	8.6e-16	10.3	2.0e-14	72.3
	phipade_dft	5.0e-10	27.7	3.3e-14	37.7	1.8e-10	32.7	1.3e-15	37.7	6.4e-14	87.7
	phipade_opt	1.3e-15	23.7	1.6e-15	28.7	1.9e-15	28.7	1.1e-15	28.7	5.9e-14	78.7

- phi_funm always has lower *computational cost*. Execution times comparable between 10^{-3} and 10^{-2} seconds (not shown).
- phi_funm and phipade_opt deliver good and comparable accuracy, but phipade_dft again shows *instability*.

Conclusions

Faster, **more accurate**, **b'ward stable** algorithm for **matrix φ-functions**:

- New recurrence between non-diagonal Padé approximants to $\varphi_j \rightsquigarrow$ Avoid repeated rational approximations $\rightsquigarrow$ Computational savings.
- Algorithmic parameters selected on the fly in $O(n^2)$ to minimize the overall computational cost.
- Backward error analysis $\rightsquigarrow$ Sharp relative error bounds & Proposed algorithm b'ward stable in *exact* arithmetic.
- Matrix triangularity exploitation (not shown) by recomput. diagonals of $\varphi_0(A) = \mathrm{e}^A$ via explicit formulae $\rightsquigarrow$ Controlled error propagation.

▶ A. H. Al-Mohy and X. Liu. A scaling and recovering algorithm for the matrix φ-functions. SIAM J. Sci. Comput., To appear. Currently available at https://arxiv.org/abs/2506.01193.

Code available at https://github.com/xiaobo-liu/phi_funm

Next?

- Excellent *numerical* f'ward stability observed. Rounding error analysis?

Bibliography I

Awad H. Al-Mohy.
A new algorithm for computing the exponential of a block triangular matrix.
SIAM J. Sci. Comput., 47(5):A3064–A3081, 2025.

Awad H. Al-Mohy and Nicholas J. Higham.
A new scaling and squaring algorithm for the matrix exponential.
SIAM J. Matrix Anal. Appl., 31(3):970–989, 2009.

Håvard Berland, Bård Skaflestad, and Will M. Wright.
EXPINT—A MATLAB package for exponential integrators.
ACM Trans. Math. Softw., 33(1):4–es, 2007.

Nicholas J. Higham.
The scaling and squaring method for the matrix exponential revisited.
SIAM J. Matrix Anal. Appl., 26(4):1179–1193, 2005.

Nicholas J. Higham.
Functions of Matrices: Theory and Computation.
SIAM, Philadelphia, PA, USA, 2008. xx+425 pp. ISBN 978-0-898716-46-7.

Marlis Hochbruck and Christian Lubich.
On Krylov subspace approximations to the matrix exponential operator.
SIAM J. Numer. Anal., 34(5):1911–1925, 1997.

Bibliography II

J. Douglas Lawson.
Generalized Runge–Kutta processes for stable systems with large Lipschitz constants.
SIAM J. Numer. Anal., 4(3):372–380, 1967.

Ya Yan Lu.
Computing a matrix function for exponential integrators.
J. Comput. Appl. Math., 161(1):203–216, 2003.

Borislav V. Minchev and Will M. Wright.
A review of exponential integrators for first order semi-linear problems.
Tech. Report 2/05, Norwegian University of Science and Technology, Trondheim, Norway, 2005.

Cleve Moler and Charles Van Loan.
Nineteen dubious ways to compute the exponential of a matrix.
SIAM Rev., 20(4):801–836, 1978.

Cleve Moler and Charles Van Loan.
Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later.
SIAM Rev., 45(1):3–49, 2003.

Bibliography III

David A. Pope.
An exponential method of numerical integration of ordinary differential equations.
Comm. Assoc. Comput. Mach., 6(8):491–493, 1963.

Bård Skaflestad and Will M. Wright.
The scaling and modified squaring method for matrix functions related to the exponential.
Appl. Numer. Math., 59(3–4):783–799, 2009.

Robert C. Ward.
Numerical computation of the matrix exponential with accuracy estimate.
SIAM J. Numer. Anal., 14(4):600–610, 1977.