The Scaling and Squaring Method for the Matrix Exponential Revisited

Nick Higham
Department of Mathematics
University of Manchester

higham@ma.man.ac.uk
http://www.ma.man.ac.uk/~higham/

THE UNIVERSITY of MANCHESTER

The Matrix Exponential

For $A \in \mathbb{C}^{n\times n}$,

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots.$$

- Difficulties in computing e^x noted by Stegun & Abramowitz (1956). They suggested $e^x = (e^{x/n})^n$, $|x/n| < 1$.
- Moler & Van Loan:
  Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, SIAM Rev., 45 (2003).
  - ► 355 citations on Science Citation Index.

Application: Control Theory

Convert **continuous-time system**

$$\frac{dx}{dt} = Fx(t) + Gu(t),$$
$$y = Hx(t) + Ju(t),$$

to **discrete-time state-space system**

$$x_{k+1} = Ax_k + Bu_k,$$
$$y_k = Hx_k + Ju_k.$$

Have

$$A = e^{F\tau}, \qquad B = \left(\int_0^\tau e^{Ft}dt\right)G,$$

where τ is the sampling period.
MATLAB Control System Toolbox: **`c2d`** and **`d2c`**.

Application: Differential Equations

Nuclear magnetic resonance: Solomon equations

$$dM/dt = -RM, \qquad M(0) = I,$$

where $M(t) =$ matrix of intensities and $R =$ symmetric relaxation matrix. NMR workers need to solve both *forward* and *inverse* problems.

Exponential time differencing for stiff systems (Cox & Matthews, 2002; Kassam & Trefethen, 2003)

$$y' = Ay + F(y,t).$$

Methods based on exact integration of linear part—require one *accurate* evaluation of e^{hA} and $e^{hA/2}$ per integration.

Quote

Whenever there is too much talk of applications,
one can rest assured that the theory
has very few of them.

— GIAN-CARLO ROTA, *Indiscrete Thoughts* (1997)

Scaling and Squaring Method

To compute $X \approx e^A$:

1. $A \leftarrow A/2^s$ so $\|A\|_\infty \approx 1$
2. $r_m(A) = [m/m]$ Padé approximant to e^A
3. $X = r_m(A)^{2^s}$

- Originates with Lawson (1967).
- Ward (1977): algorithm, with rounding error analysis and a posteriori error bound.
- Moler & Van Loan (1978): give backward error analysis covering truncation error in Padé approximations, allowing choice of s and m.

Padé Approximations r_m to e^x

$r_m(x) = p_m(x)/q_m(x)$ known explicitly:

$$p_m(x) = \sum_{j=0}^{m} \frac{(2m-j)!\,m!}{(2m)!\,(m-j)!}\frac{x^j}{j!}$$

and $q_m(x) = p_m(-x)$. The error satisfies

$$e^x - r_m(x) = (-1)^m \frac{(m!)^2}{(2m)!(2m+1)!} x^{2m+1} + O(x^{2m+2}).$$

Choice of Scaling and Padé Degree

Moler & Van Loan (1978) show that if $\|A/2^s\| \le 1/2$ then

$$r_m(A/2^s)^{2^s} = e^{A+E},$$

where $AE = EA$ and

$$\frac{\|E\|}{\|A\|} \le 2^{3-2m} \frac{(m!)^2}{(2m)!(2m+1)!}. \qquad (*)$$

- For $m = 6$, the bound is 3.4×10^{-16}.
- MATLAB's **`expm`** takes s so that $\|A/2^s\| \le 1/2$ and $m = 6$.

Choice of Scaling and Padé Degree

Moler & Van Loan (1978) show that if $\|A/2^s\| \le 1/2$ then

$$r_m(A/2^s)^{2^s} = e^{A+E},$$

where $AE = EA$ and

$$\frac{\|E\|}{\|A\|} \le 2^{3-2m}\frac{(m!)^2}{(2m)!(2m+1)!}. \qquad (*)$$

- For $m = 6$, the bound is 3.4×10^{-16}.
- MATLAB's `expm` takes s so that $\|A/2^s\| \le 1/2$ and $m = 6$.

- Why restrict to $\|A/2^s\| \le 1/2$?
- Bound $(*)$ is far from sharp.

Analysis

Let

$$e^{-A} r_m(A) = I + G = e^H$$

and assume $\|G\| < 1$. Then

$$\|H\| = \|\log(I+G)\| \le \sum_{j=1}^{\infty} \|G\|^j / j = -\log(1 - \|G\|).$$

Hence

$$r_m(A) = e^A e^H = e^{A+H}.$$

Rewrite as

$$r_m(A/2^s)^{2^s} = e^{A+E},$$

where $E = 2^s H$ satisfies

$$\|E\| \le -2^s \log(1 - \|G\|).$$

Result

Theorem 1 *Let*

$$e^{-2^{-s}A} r_m(2^{-s}A) = I + G,$$

where $\|G\| < 1$*. Then the diagonal Padé approximant* r_m *satisfies*

$$r_m(2^{-s}A)^{2^s} = e^{A+E},$$

where

$$\frac{\|E\|}{\|A\|} \le \frac{-\log(1 - \|G\|)}{\|2^{-s}A\|}. \qquad \square$$

- Remains to bound $\|G\|$ in terms of $\|2^{-s}A\|$.

Bounding $\|G\|$

$$\rho(x) := e^{-x} r_m(x) - 1 = \sum_{i=2m+1}^{\infty} c_i x^i$$

converges absolutely for $|x| < \min\{\, |t| : q_m(t) = 0 \,\} =: \nu_m$.
Hence, with $\theta := \|2^{-s}A\| < \nu_m$,

$$\|G\| = \|\rho(2^{-s}A)\| \le \sum_{i=2m+1}^{\infty} |c_i| \theta^i =: f(\theta). \qquad (*)$$

Thus $\|E\|/\|A\| \le -\log(1 - f(\theta))/\theta$.

- If only $\|A\|$ known, $(*)$ is optimal bound on $\|G\|$.
- Moler & Van Loan (1978) bound less sharp; Dieci & Papini (2000) bound a different error.

Finding Largest θ

To obtain

$$f(\theta) = \sum_{i=2m+1}^{\infty} |c_i|\theta^i,$$

compute c_i symbolically, sum series in 250 digit arithmetic.

Use zero-finder to determine largest θ, denoted θ_m, such that b'err bound $\leq u = 2^{-53} \approx 1.1 \times 10^{-16}$ (IEEE double).

m	1	2	3	4	5	6	7	8	9	10
θ_m	3.7e-8	5.3e-4	1.5e-2	8.5e-2	2.5e-1	5.4e-1	9.5e-1	1.5e0	2.1e0	2.8e0

m	11	12	13	14	15	16	17	18	19	20
θ_m	3.6e0	4.5e0	5.4e0	6.3e0	7.3e0	8.4e0	9.4e0	1.1e1	1.2e1	1.3e1

Computational Cost

Efficient scheme for r_8:

$$\begin{aligned} p_8(A) &= b_8A^8 + b_6A^6 + b_4A^4 + b_2A^2 + b_0I \\ &\quad + A(b_7A^6 + b_5A^4 + b_3A^2 + b_1I) \\ &=: U + V. \end{aligned}$$

Then $q_8(A) = U - V$.
For $m \geq 12$ a different scheme is more efficient.

Number of mat mults π_m to evaluate r_m:

m	1	2	3	4	5	6	7	8	9	10
π_m	0	1	2	3	3	4	4	5	5	6

m	11	12	13	14	15	16	17	18	19	20
π_m	6	6	6	7	7	7	7	8	8	8

Optimal Algorithm

Recall $A \leftarrow 2^{-s}A$, $s = \lceil \log_2 \|A\|/\theta_m \rceil$ if $\|A\| \geq \theta_m$, else $s = 0$.
Hence cost of overall algorithm in mat mults is

$$\pi_m + s = \pi_m + \max\left(\lceil \log_2 \|A\| - \log_2 \theta_m \rceil, 0\right).$$

For $\|A\| \geq \theta_m$ simplify to $C_m = \pi_m - \log_2 \theta_m$.

m	1	2	3	4	5	6	7	8	9	10
C_m	25	12	8.1	6.6	5.0	4.9	4.1	4.4	3.9	4.5

m	11	12	13	14	15	16	17	18	19	20
C_m	4.2	3.8	3.6	4.3	4.1	3.9	3.8	4.6	4.5	4.3

Optimal Algorithm

Recall $A \leftarrow 2^{-s}A$, $s = \lceil \log_2 \|A\|/\theta_m \rceil$ if $\|A\| \geq \theta_m$, else $s = 0$.
Hence cost of overall algorithm in mat mults is

$$\pi_m + s = \pi_m + \max\left(\lceil \log_2 \|A\| - \log_2 \theta_m \rceil, 0\right).$$

For $\|A\| \geq \theta_m$ simplify to $C_m = \pi_m - \log_2 \theta_m$.

m	1	2	3	4	5	6	7	8	9	10
C_m	25	12	8.1	6.6	5.0	4.9	4.1	4.4	3.9	4.5

m	11	12	13	14	15	16	17	18	19	20
C_m	4.2	3.8	3.6	4.3	4.1	3.9	3.8	4.6	4.5	4.3

- **For IEEE single, $m = 7$ is optimal.**
- **For quad prec., $m = 17$ is optimal.**

Rounding Errors in Evaluating r_m

Can show, improving Ward (1977) bounds,

$$\|p_m(A) - \widehat{p}_m(A)\|_1 \lesssim \widetilde{\gamma}_{mn} \|p_m(A)\|_1 \, e^{\theta_m} \qquad \text{(ditto for } q_m\text{)}$$

and

$$\|q_m(A)^{-1}\| \le \frac{e^{\theta_m/2}}{1 - \sum_{i=2}^{\infty} |d_i| \theta_m^i} =: \xi_{\mathbf{m}},$$

where $e^{x/2} q_m(x) - 1 = \sum_{i=2}^{\infty} d_i x^i$.

m	1	2	3	4	5	6	7	8	9	10
$\xi_{\mathbf{m}}$	1.0e0	1.0e0	1.0e0	1.0e0	1.1e0	1.3e0	1.6e0	2.1e0	3.0e0	4.3e0

m	11	12	13	14	15	16	17	18	19	20
$\xi_{\mathbf{m}}$	6.6e0	1.0e1	1.7e1	3.0e1	5.3e1	9.8e1	1.9e2	3.8e2	8.3e2	2.0e3

Algorithm

Algorithm 1 ***Evaluate e^A, for $A \in \mathbb{C}^{n\times n}$, using the scaling and squaring method.***

for $m = [3\ 5\ 7\ 9\ 13]$
    if $\|A\|_1 \le \theta_m$
        $X = r_m(A)$, return
    end
end
$A \leftarrow A/2^s$ with s min integer s.t. $\|A/2^s\|_1 \le \theta_{13} \approx 5.4$
    $(s = \lceil \log_2(\|A\|_1/\theta_{13}) \rceil)$

$X = r_{13}(A)$ [increasing ordering]

$X \leftarrow X^{2^s}$ by repeated squaring

- May want to add preprocessing to reduce the norm.

Comparison with Existing Algorithms

Method	m	$\max \|2^{-s}A\|$	
Alg 1	13	5.4	
Ward (1977)	8	1.0	$[\theta_8 = 1.5]$
MATLAB 7's `expm`	6	0.5	$[\theta_6 = 0.54]$
Sidje (1998)	6	0.5	

Comparison with Existing Algorithms

Method	m	$\max \|2^{-s}A\|$	
Alg 1	13	5.4	
Ward (1977)	8	1.0	$[\theta_8 = 1.5]$
MATLAB 7's **`expm`**	6	0.5	$[\theta_6 = 0.54]$
Sidje (1998)	6	0.5	

- $\|A\|_1 > 1$: Alg 1 requires 1–2 fewer mat mults than Ward, 2–3 fewer than **`expm`**.

$\|A\|_1 \in (2, 2.1)$:

	Alg 1	Ward	**`expm`**	Sidje
mults	5	7	8	10

- $\|A\|_1 \le 1$: Alg 1 requires up to 3 fewer, and no more, mat mults than **`expm`** and Ward.

Squaring Phase

- The bound

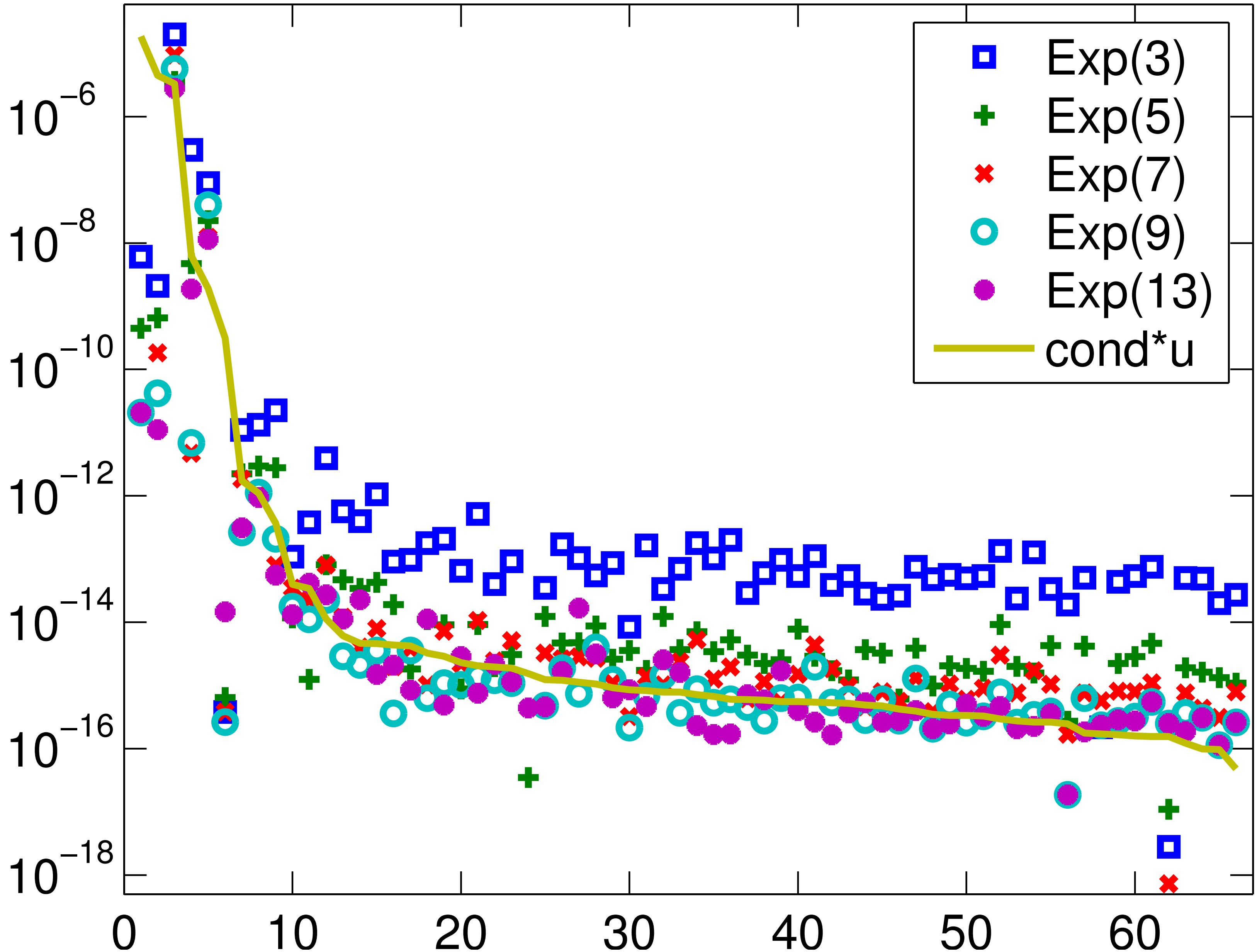
$$\|A^2 - fl(A^2)\| \le \gamma_n \|A\|^2, \qquad \gamma_n = \frac{nu}{1-nu}.$$

  shows the dangers in matrix squaring.
- **Open question**: are errors in squaring phase consistent with conditioning of the problem?
- Our choice of parameters uses 1–5 fewer matrix squarings than existing implementations, hence has **potential accuracy advantages**.

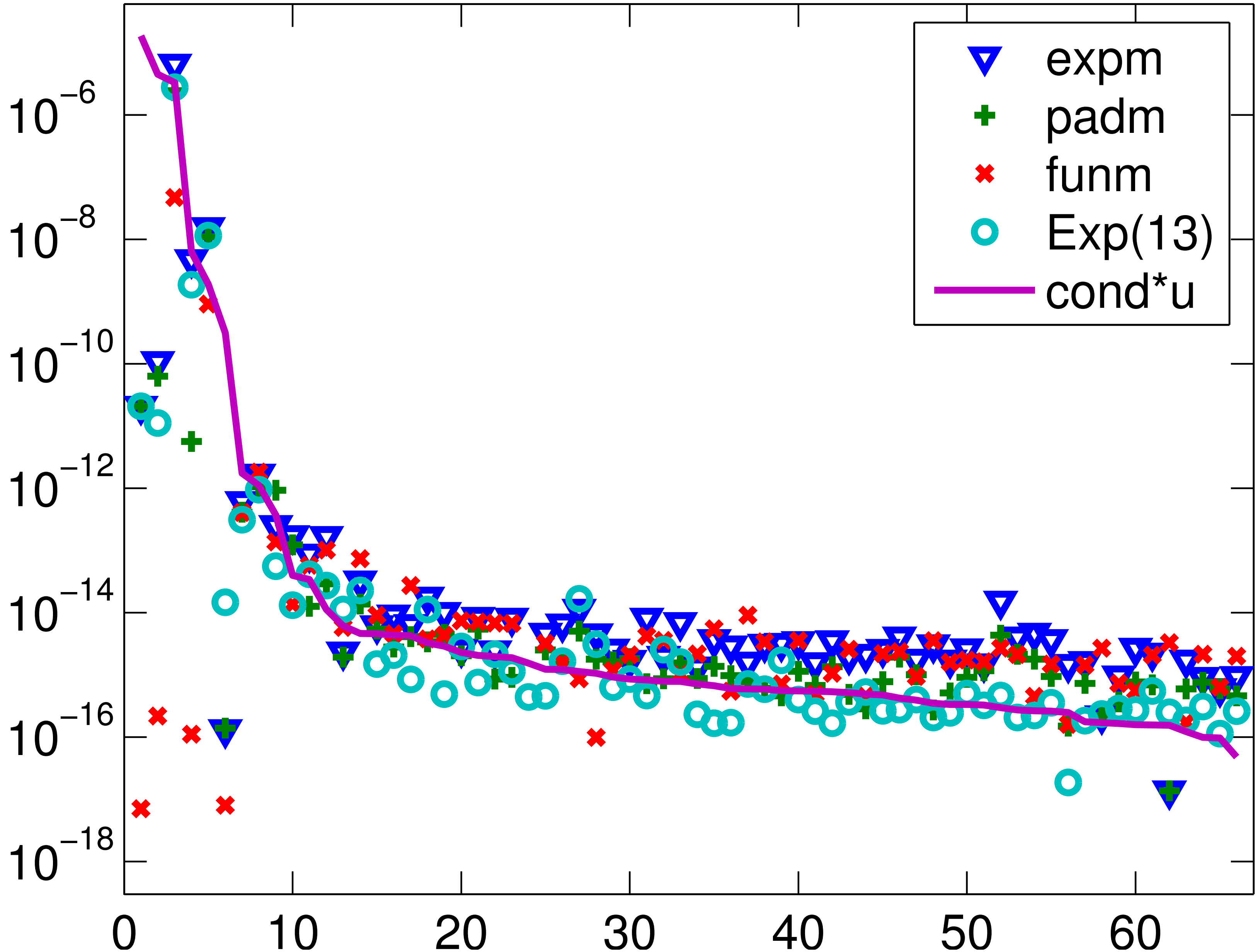
Numerical Experiment

- 66 8×8 test matrices: 53 from the function `matrix` in Matrix Computation Toolbox and 13 of dimension 2–10 from e^A literature.
- Evaluated 1-norm relative error.
- Used Alg 1 and modified version with max Padé degree a parameter, $m_{\max}$, denoted `Exp`($m_{\max}$) .
- Notation:
  - `expm`: MATLAB 7 scaling & squaring $(m = 6)$.
  - `funm`: MATLAB 7 Schur–Parlett function.
  - `padm`: Sidje $(m = 6)$.
- $\operatorname{cond}(A) = \lim_{\epsilon \to 0} \max_{\|E\|_2 \le \epsilon \|A\|_2} \dfrac{\|e^{A+E} - e^A\|_2}{\epsilon \|e^A\|_2}.$

Different $m_{\max}$



Different S&S Codes and funm



Performance Profiles

Dolan & Moré (2002) propose a new type of performance profile.

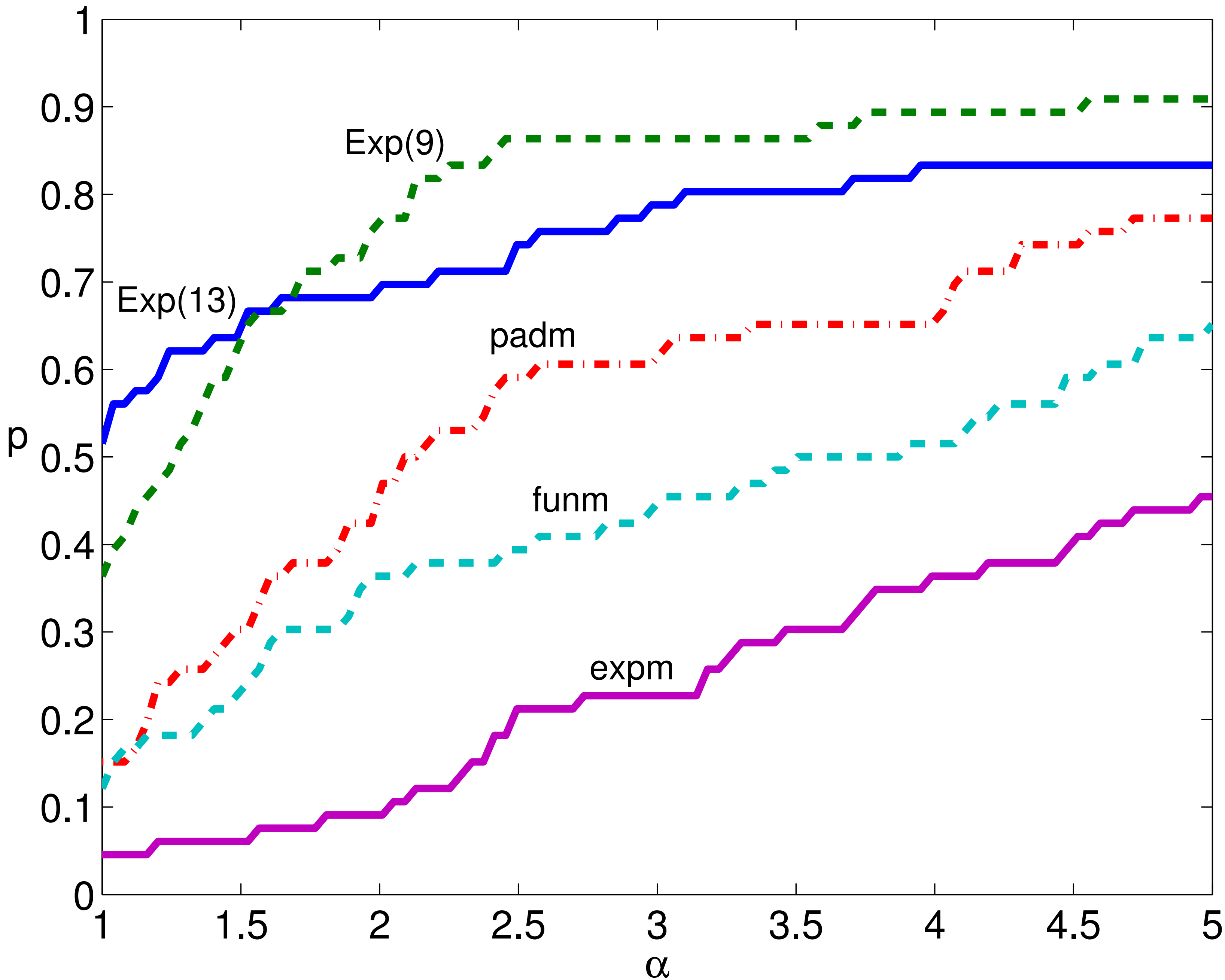
- Let $t_s(p)$ measure cost or accuracy of solver $s \in S$ on problem $p \in P$.
- Performance ratio

$$r_{p,s} := \frac{t_s(p)}{\min\{\, t_\sigma(p) : \sigma \in S \,\}} \geq 1.$$

- Plot α against

$$P(r_{p,s} \leq \alpha \text{ for all } s).$$

Performance Profile



Indirect Padé Approximation

Najfeld & Havel (1995) suggest using Padé approx. to

$$\tau(x) = x\coth(x) = x(e^{2x}+1)(e^{2x}-1)^{-1}$$
$$= 1 + \cfrac{x^2}{3 + \cfrac{x^2}{5 + \cfrac{x^2}{7 + \cdots}}},$$

for which

$$e^{2x} = \frac{\tau(x)+x}{\tau(x)-x}.$$

For example, $[2m/2m]$ Padé approximant to τ is

$$\widetilde{r}_8(x) = \frac{\frac{1}{765765}x^8 + \frac{4}{9945}x^6 + \frac{7}{255}x^4 + \frac{8}{17}x^2 + 1}{\frac{1}{34459425}x^8 + \frac{2}{69615}x^6 + \frac{1}{255}x^4 + \frac{7}{51}x^2 + 1}.$$

Najfeld & Havel Algorithm

Error in r_{2m} has form

$$\tau(x) - \widetilde{r}_{2m}(x) = \sum_{k=1}^{\infty} d_k x^{4m+2k} = \sum_{k=1}^{\infty} d_k (x^2)^{2m+k}$$

$$\Rightarrow \quad \|\tau(A) - \widetilde{r}_{2m}(A)\| \le \sum_{k=1}^{\infty} d_k \|A^2\|^{2m+k} =: \omega_{2m}(\|A^2\|).$$

Let θ_{2m} be largest θ such that $\omega_{2m}(\theta) \le u$.

- $\widetilde{A} \leftarrow A/2^{s+1}$ with $s \ge 0$ s.t. $\|\widetilde{A}^2\| = \|A^2\|/2^{2s+2} \le \theta_{2m}$.
- Evaluate $\widetilde{r}_{2m}(\widetilde{A})$ then $(\widetilde{r}_{2m} + \widetilde{A})(\widetilde{r}_{2m} - \widetilde{A})^{-1}$.
- Square result s times.
- $m = 8$ leads to most efficient algorithm.

Equivalence

Theorem 2 *The* $[2m/2m]$ *Padé approximant* $\widetilde{r}_{2m}(x)$ *to* $x\coth(x)$ *is related to the* $[2m+1/2m+1]$ *Padé approximant* $r_{2m+1}(x)$ *to* e^x *by*

$$r_{2m+1}(x) = \frac{\widetilde{r}_{2m}(x/2) + x/2}{\widetilde{r}_{2m}(x/2) - x/2}.$$

- N & H alg ($m = 8$) implicitly uses same Padé approximant to e^x as Alg 1 with $m = 9$.
- N & H derivation bounds error $\|\tau(A) - \widetilde{r}_{2m}(A)\|$ for scaled A. What does this imply about $\|e^{2A} - (\widetilde{r}_{2m} + A)(\widetilde{r}_{2m} - A)^{-1}\|$?
- $\widetilde{r}_{2m} - A$ can be arbitrarily ill conditioned.
- No backward error bound analogous to that for Alg 1.

Conclusions

- ★ New scaling & squaring implementation **up to 1.6 times faster** than `expm` and **significantly more accurate**.
- ★ Improvement comes by replacing mathematically elegant error bound by sharper bound, which is evaluated symbolically/numerically.
- ★ High degree Padé approximants are **numerically viable**. (Error analysis guarantees stable evaluation.)
- ★ Another example where **faster $\Rightarrow$ more accurate**!
- ★ No example of instability of new alg seen in the tests. Open question: **Is S&S method stable?**
- ★ **Performance profiles**—a useful tool in numerical linear algebra, not just optimization.