Computing the Action of the Matrix Exponential

Nick Higham
School of Mathematics
The University of Manchester

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

Joint work with Awad H. Al-Mohy

16th ILAS Conference, Pisa, June 2010

The $f(A)b$ Problem

Given

- matrix function $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$,
- $A \in \mathbb{C}^{n\times n}$, $b \in \mathbb{C}^n$,

compute $f(A)b$ *without first computing $f(A)$*.

Most important cases

- $f(x) = x^{-1}$,
- $f(x) = e^x$.

Application:

$$y'(t) = Ay(t), \quad y(0) = b \quad \Rightarrow \quad y(t) = e^{At}b.$$

Second Order ODE

$$\frac{d^2y}{dt^2} + Ay = 0, \qquad y(0) = y_0, \quad y'(0) = y'_0$$

has solution

$$y(t) = \cos(\sqrt{A}\,t)y_0 + \left(\sqrt{A}\right)^{-1} \sin(\sqrt{A}\,t)y'_0.$$

Second Order ODE

$$\frac{d^2y}{dt^2} + Ay = 0, \qquad y(0) = y_0, \quad y'(0) = y'_0$$

has solution

$$y(t) = \cos(\sqrt{A}\,t)y_0 + \left(\sqrt{A}\right)^{-1}\sin(\sqrt{A}\,t)y'_0.$$

But

$$\begin{bmatrix} y' \\ y \end{bmatrix} = \exp\left(\begin{bmatrix} 0 & -tA \\ t\,I_n & 0 \end{bmatrix}\right)\begin{bmatrix} y'_0 \\ y_0 \end{bmatrix}.$$

Themes

- Simple, direct algorithm for computing $e^A B$ with arbitrary A using only matrix products.
- Many problems can be reduced to $e^A B$.
- Backward error viewpoint avoids consideration of conditioning in algorithm design.

Exponential Integrators

$$u'(t) = Au(t) + g(t, u(t)), \quad u(0) = u_0, \quad t \geq 0,$$

Solution can be written

$$u(t) = e^{tA}u_0 + \sum_{k=1}^{\infty} \varphi_k(tA)\, t^k\, u_k,$$

where $u_k = g^{(k-1)}(t, u(t))\,|_{t=0}$ and $\varphi_\ell(z) = \sum_{k=0}^{\infty} z^k/(k+\ell)!$.

Exponential Integrators

$$u'(t) = Au(t) + g(t, u(t)), \quad u(0) = u_0, \quad t \geq 0,$$

Solution can be written

$$u(t) = e^{tA} u_0 + \sum_{k=1}^{\infty} \varphi_k(tA)\, t^k\, u_k,$$

where $u_k = g^{(k-1)}(t, u(t))\,|_{t=0}$ and $\varphi_\ell(z) = \sum_{k=0}^{\infty} z^k/(k+\ell)!$.

$$u(t) \approx \widehat{u}(t) = e^{tA} u_0 + \sum_{k=1}^{p} \varphi_k(tA)\, t^k\, u_k.$$

Exponential time differencing (ETD) Euler ($p = 1$):

$$y_{n+1} = e^{hA} y_n + h\varphi_1(hA) g(t_n, y_n).$$

Saad's Trick (1992)

$$\varphi_1(z) = \frac{e^z - 1}{z}.$$

$$\exp\left(\begin{bmatrix} A & b \\ 0 & 0 \end{bmatrix}\right) = \begin{bmatrix} e^A & \varphi_1(A)b \\ 0 & 1 \end{bmatrix}$$

Theorem (Al-Mohy & H, 2010)

Let $A \in \mathbb{C}^{n\times n}$, $U = [u_1, u_2, \dots, u_p] \in \mathbb{C}^{n\times p}$, $\tau \in \mathbb{C}$, *and define*

$$B = \begin{bmatrix} A & U \\ 0 & J \end{bmatrix} \in \mathbb{C}^{(n+p)\times(n+p)}, \quad J = \begin{bmatrix} 0 & I_{p-1} \\ 0 & 0 \end{bmatrix} \in \mathbb{C}^{p\times p}.$$

Then for $X = e^{\tau B}$ *we have*

$$X(1\colon n, n+j) = \sum_{k=1}^{j} \tau^k \varphi_k(\tau A)\, u_{j-k+1}, \quad j = 1\colon p.$$

Theorem (Al-Mohy & H, 2010)

Let $A \in \mathbb{C}^{n\times n}$, $U = [u_1, u_2, \dots, u_p] \in \mathbb{C}^{n\times p}$, $\tau \in \mathbb{C}$, *and define*

$$B = \begin{bmatrix} A & U \\ 0 & J \end{bmatrix} \in \mathbb{C}^{(n+p)\times(n+p)}, \quad J = \begin{bmatrix} 0 & I_{p-1} \\ 0 & 0 \end{bmatrix} \in \mathbb{C}^{p\times p}.$$

Then for $X = e^{\tau B}$ *we have*

$$X(1\colon n, n+j) = \sum_{k=1}^{j} \tau^k \varphi_k(\tau A)\, u_{j-k+1}, \quad j = 1\colon p.$$

Completely removes the need to evaluate φ_k functions!

Implementation of Exponential Integrators

We compute

$$\widehat{u}(t) = e^{tA}u_0 + \sum_{k=1}^{p} \varphi_k(tA)\, t^k\, u_k$$

as, with $U = [u_p, \ldots, u_1]$,

$$\widehat{u}(t) = \begin{bmatrix} I_n & 0 \end{bmatrix} \exp\left(t \begin{bmatrix} A & \eta U \\ 0 & J \end{bmatrix} \right) \begin{bmatrix} u_0 \\ \eta^{-1} e_p \end{bmatrix}.$$

Choose

$$\eta = 2^{-\lceil \log_2(\|U\|_1) \rceil}$$

to avoid overscaling.

Formulae for e^A, $A \in \mathbb{C}^{n\times n}$

Power series $I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots$	**Limit** $\lim_{s\to\infty} (I + A/s)^s$	**Scaling and squaring** $(e^{A/2^s})^{2^s}$
Cauchy integral $\frac{1}{2\pi i}\int_\Gamma e^z (zI - A)^{-1}\, dz$	**Jordan form** $Z\mathrm{diag}(e^{J_k})Z^{-1}$	**Interpolation** $\sum_{i=1}^{n} f[\lambda_1, \ldots, \lambda_i] \prod_{j=1}^{i-1} (A - \lambda_j I)$
Differential system $Y'(t) = AY(t),\ Y(0) = I$	**Schur form** $Qe^T Q^*$	**Padé approximation** $p_{km}(A)q_{km}(A)^{-1}$

Krylov methods: Arnoldi fact. $AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k^T$ with Hessenberg H: $e^A b \approx Q_k e^{H_k} Q_k^* b$.

Scaling and Squaring Method

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

- MATLAB `expm` uses alg of H (2005).
- Improved algorithm: Al-Mohy & H (2009).

Can we adapt this approach for $e^A B$?

Computing $e^A B$

$\underbrace{A}_{n\times n}, \underbrace{B}_{n\times n_0}, n_0 \ll n.$ Exploit, for integer s,

$$e^A B = (e^{s^{-1}A})^s B = \underbrace{e^{s^{-1}A} e^{s^{-1}A} \cdots e^{s^{-1}A}}_{s \text{ times}} B.$$

Choose s so $T_m(s^{-1}A) = \sum_{j=0}^{m} \frac{(s^{-1}A)^j}{j!} \approx e^{s^{-1}A}$. Then

$$B_{i+1} = T_m(s^{-1}A)B_i, \quad i = 0\colon s-1, \qquad B_0 = B$$

yields $B_s \approx e^A B$.

Computing $e^A B$

$\underbrace{A}_{n\times n}, \underbrace{B}_{n\times n_0}, n_0 \ll n.$ Exploit, for integer s,

$$e^A B = (e^{s^{-1}A})^s B = \underbrace{e^{s^{-1}A} e^{s^{-1}A} \cdots e^{s^{-1}A}}_{s \text{ times}} B.$$

Choose s so $T_m(s^{-1}A) = \sum_{j=0}^{m} \frac{(s^{-1}A)^j}{j!} \approx e^{s^{-1}A}$. Then

$$B_{i+1} = T_m(s^{-1}A) B_i, \quad i = 0\colon s-1, \qquad B_0 = B$$

yields $B_s \approx e^A B$.

How to choose s and m?

Backward Error Analysis

Lemma (Al-Mohy & H, 2009)

$T_m(s^{-1}A)^s B = e^{A+\Delta A}B$, where $\Delta A = s h_{m+1}(s^{-1}A)$ and $h_{m+1}(x) = \log(e^{-x}T_m(x)) = \sum_{k=m+1}^{\infty} c_k x^k$. Moreover,

$$\|h_{m+1}(A)\| \le \sum_{k=m+1}^{\infty} |c_k| \, \alpha_p(A)^k$$

if $m+1 \ge p(p-1)$, where

$$\alpha_p(A) = \max(d_p, d_{p+1}), \qquad d_p = \|A^p\|^{1/p}.$$

Why Use $d_p = \|A^p\|^{1/p}$?

- $\|A^k\| \le \|A^{pk}\|^{1/p} \le \left(\|A^p\|^{1/p}\right)^k$.
- $\rho(A) \le \|A^p\|^{1/p} \le \|A\|$.
- With $A = \begin{bmatrix} 0.9 & 500 \\ 0 & -0.5 \end{bmatrix}$:

k	2	5	10
$\|A^k\|_1$	2.0e2	2.2e2	1.2e2
$\|A\|_1^k$	2.5e5	3.1e13	9.8e26
$d_2^k = \left(\|A^2\|_1^{1/2}\right)^k$	2.0e2	5.7e5	3.2e11
$d_3^k = \left(\|A^3\|_1^{1/3}\right)^k$	4.5e1	1.3e4	1.9e8

- Cheaply estimate $\|A^k\|$, for a few k (H & Tisseur, 2001).

Why Use $d_p = \|A^p\|^{1/p}$? — cont.

- $d_p = \|A^p\|^{1/p}$ provide information about the nonnormality of A.
- Their use helps avoid overscaling.
- What other uses do they have?

Choice of s and m

- $\alpha_p(A) := \max\big(d_p, d_{p+1}\big)$
- $\theta_m := \max\left\{ \theta : \sum_{k=m+1}^{\infty} |c_k|\theta^{k-1} \le \epsilon \right\}$

$\dfrac{\|\Delta A\|}{\|A\|} \le \epsilon$ if $m + 1 \ge p(p-1)$ and $s^{-1}\alpha_p(A) \le \theta_m$.

Computational cost for $B_s \approx e^A B$ is

$$C_m(A) = m \max\big(\lceil \alpha_p(A)/\theta_m \rceil, 1\big)$$

matrix products.

- Cost decreases with m.
- Restrict $2 \le p \le p_{\max}$, $p(p-1) - 1 \le m \le m_{\max}$.
- Minimize cost over p, m

Size of Taylor Series Argument

Constants θ_m (via symbolic, high precision):

m	10	20	30	40	55
single	1.0e0	3.6e0	6.3e0	9.1e0	1.3e1
double	1.4e-1	1.4e0	3.5e0	6.0e0	9.9e0

Preprocessing

Expand the Taylor series about $\mu \in \mathbb{C}$:

$$e^{\mu} \sum_{k=0}^{\infty} (A - \mu I)^k / k!$$

Choose μ so $\|A - \mu I\| \leq \|A\|$.

- Alg is based on 1-norm, but minimizing $\|A - \mu I\|_F$ does better empirically at minimizing $d_p(A - \mu I)$.
- Recover e^A from

$$e^{\mu} \left[T_m(s^{-1}(A - \mu I)) \right]^s, \qquad \left[e^{\mu/s} T_m(s^{-1}(A - \mu I)) \right]^s.$$

  First expression prone to overflow, so prefer second.
- Balancing is an option.

Termination Criterion

In evaluating

$$T_m(s^{-1}A)B_i = \sum_{j=0}^{m} \frac{(s^{-1}A)^j}{j!} B_i$$

we accept $T_k(A)B_i$ for the first k such that

$$\frac{\|A^{k-1}B_i\|}{(k-1)!} + \frac{\|A^k B_i\|}{k!} \le \epsilon \| T_k(A)B_i\|.$$

Algorithm for $F = e^{tA}B$

$\mu = \mathrm{trace}(A)/n$
$A = A - \mu I$
$[m_*, s] = \mathrm{parameters}(tA)$
$F = B$, $\eta = e^{t\mu/s}$
for $i = 1\colon s$
    $c_1 = \|B\|_\infty$
    for $j = 1\colon m_*$
        $B = tAB/(sj)$, $c_2 = \|B\|_\infty$
        $F = F + B$
        if $c_1 + c_2 \le \mathrm{tol}\|F\|_\infty$, quit, end
        $c_1 = c_2$
    end
    $F = \eta F$, $B = F$
end

George Forsythe "Pitfalls" (1970)

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots.$$

Since you learned mathematics because it is useful, you might expect to use the series to compute e^x. Suppose—just for illustration—that your floating-point number system F is characterized by $\beta = 10$ and $s = 5$. Let us use the series for $x = -5.5$, as proposed by Stegun and Abramowitz [13]. Here are the numbers we get:

$$\begin{aligned} e^{-5.5} \approx\ & 1.0000 \\ & -\ 5.5000 \\ & +15.125 \\ & -27.730 \\ & +38.129 \\ & -41.942 \\ & +38.446 \\ & -30.208 \\ & +20.768 \\ & -12.692 \\ & +\ 6.9803 \\ & -\ 3.4902 \\ & +\ 1.5997 \\ & \quad\vdots \\ \hline & +\ 0.0026363 \end{aligned}$$

Conditioning of $e^A B$

$$\kappa_{\exp}(A,B) \le \frac{\|e^A\|_F \|B\|_F}{\|e^A B\|_F}(1+\kappa_{\exp}(A)).$$

$$\|A\|_2 \le \kappa_{\exp}(A) \le \frac{e^{\|A\|_2}\|A\|_2}{\|e^A\|_2}$$

Relative forward error due to roundoff bounded by

$$u e^{\|A\|_2}\|B\|_2 / \|e^A B\|_F.$$

- A normal implies $\kappa_{\exp}(A) = \|A\|_2$. Then instability if $e^{\|A\|_2} \gg \|e^A\|_2$.
- A Hermitian implies spectrum of $A - n^{-1}\mathrm{trace}(A)I$ has $\lambda_{\max} = -\lambda_{\min} \Rightarrow$ (normwise) stability!

$e^{tA}B$ for Several t

Grid: $t_k = t_0 + kh$, $k = 0\colon q$, where $h = (t_q - t_0)/q$.

- Evaluate $B_k = e^{t_k A}B$, $k = 0\colon q$, directly.
- Form $B_0 = e^{t_0 A}B$ and then

$$B_k = e^{khA}B_0, \quad k = 1\colon q \qquad B_2 = e^{2hA}B_0 \tag{1}$$

- Form $B_0 = e^{t_0 A}B$ and then

$$B_k = e^{hA}B_{k-1}, \quad k = 1\colon q \qquad B_2 = e^{hA}(e^{hA}B_0) \tag{2}$$

Suffers from **overscaling** when h is small enough.

$e^{tA}B$ for Several t

Grid: $t_k = t_0 + kh$, $k = 0\colon q$, where $h = (t_q - t_0)/q$.

- Evaluate $B_k = e^{t_k A}B$, $k = 0\colon q$, directly.
- Form $B_0 = e^{t_0 A}B$ and then

$$B_k = e^{khA}B_0, \quad k = 1\colon q \qquad B_2 = e^{2hA}B_0 \tag{1}$$

- Form $B_0 = e^{t_0 A}B$ and then

$$B_k = e^{hA}B_{k-1}, \quad k = 1\colon q \qquad B_2 = e^{hA}(e^{hA}B_0) \tag{2}$$

Suffers from **overscaling** when h is small enough.

x	$e^x - (1+x)$	$e^x - (1+x/2)^2$
9.9e-9	2.2e-16	6.7e-16
8.9e-9	0	6.7e-16

$e^{tA}B$ for Several t

Grid: $t_k = t_0 + kh$, $k = 0\colon q$, where $h = (t_q - t_0)/q$.

- Evaluate $B_k = e^{t_k A}B$, $k = 0\colon q$, directly.
- Form $B_0 = e^{t_0 A}B$ and then

$$B_k = e^{khA}B_0, \quad k = 1\colon q \qquad B_2 = e^{2hA}B_0 \qquad (1)$$

- Form $B_0 = e^{t_0 A}B$ and then

$$B_k = e^{hA}B_{k-1}, \quad k = 1\colon q \qquad B_2 = e^{hA}(e^{hA}B_0) \qquad (2)$$

Suffers from **overscaling** when h is small enough.

- Use (1) when no cost penalty (no scaling), else (2).
- Opportunity to save and re-use some matrix products.

The Sixth Dubious Way (1)

Moler & Van Loan (1978, 2003)

METHOD 6. SINGLE STEP O.D.E. METHODS. Two of the classical techniques for the solution of differential equations are the fourth order Taylor and Runge–Kutta methods with fixed step size. For our particular equation they become

$$x_{j+1} = \left(I + hA + \cdots + \frac{h^4}{4!}A^4\right)x_j = T_4(hA)x_j$$

and

$$x_{j+1} = x_j + \tfrac{1}{6}k_1 + \tfrac{1}{3}k_2 + \tfrac{1}{3}k_3 + \tfrac{1}{6}k_4,$$

where $k_1 = hAx_j$, $k_2 = hA(x_j + \frac{1}{2}k_1)$, $k_3 = hA(x_j + \frac{1}{2}k_2)$, and $k_4 = hA(x_j + k_3)$. A little manipulation reveals that in this case, the two methods would produce identical results were it not for roundoff error. As long as the step size is fixed, the matrix $T_4(hA)$ need be computed just once and then x_{j+1} can be obtained from x_j with just one matrix-vector multiplication. The standard Runge–Kutta method would require 4 such multiplications per step.

Let us consider $x(t)$ for one particular value of t, say $t = 1$. If $h = 1/m$, then

$$x(1) = x(mh) \simeq x_m = [T_4(hA)]^m x_0.$$

The Sixth Dubious Way (2)

Advantages of our method over the one-step ODE integrator:

- Fully exploits the **linearity** of the ODE.
- **Backward error** based; ODE integrator control local (forward) errors.
- **Overscaling** avoided.

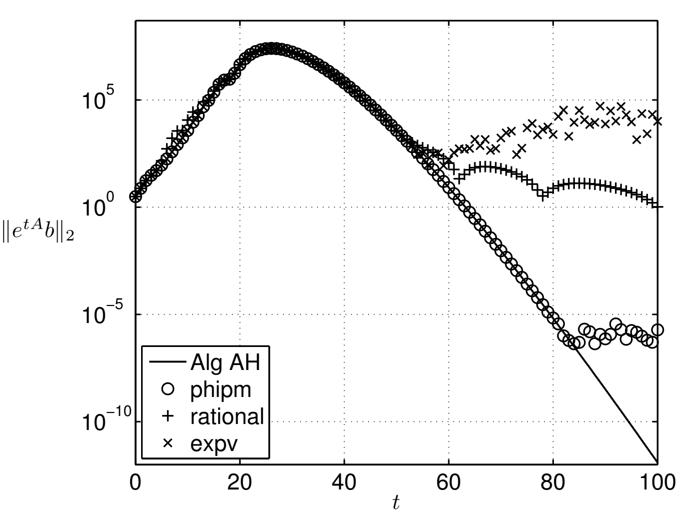
Experiment 1

Trefethen, Weideman & Schmelzer (2006):
$A \in \mathbb{R}^{9801\times 9801}$, 2D Laplacian,
`-2500*gallery('poisson',99)`.

Compute $e^{\alpha tA}b$ for 100 equally spaced $t \in [0,1]$. tol $= u_d$.

	$\alpha = 0.02$			$\alpha = 1$		
	speed	cost	diff	speed	cost	diff
Alg AH	1	1119		1	49544	
`expv`	46.6	25575	4.5e-15	66.0	516429	6.2e-14
`phipm`	10.5	10744	5.5e-15	9.3	150081	6.3e-14
rational	107.8	700	9.1e-14	7.9	700	1.0e-12

$A =$ `-gallery('triw',20,4.1)`, $b_i = \cos i$, tol $= u_d$.



Experiment 3

Harwell–Boeing matrices:

- **orani678**, $n = 2529$, $t = 100$, $b = [1, 1, \ldots, 1]^T$;
- **bcspwr10**, $n = 5300$, $t = 10$, $b = [1, 0, \ldots, 0, 1]^T$.

2D Laplacian matrix, **poisson**. tol $= u_s$.

	Alg AH			ode15s		
	time	cost	error	time	cost	error
orani678	0.13	878	4e-8	136	$7780 + \cdots$	2e-6
bcspwr10	0.021	215	7e-7	2.92	$1890 + \cdots$	5e-5
poisson	3.76	29255	2e-6	2.48	$402 + \cdots$	8e-6
4poisson	15	116849	9e-6	3.24	$49 + \cdots$	**1e-1**

Comparison with Krylov Methods

Alg AH	**Krylov Methods**
Most time spent in matrix–vector products.	Krylov recurrence and e^H can be significant.
“Direct method”, cost predictable.	Iterative method; needs stopping test.
No parameters to estimate.	Select Krylov subspace size.
Storage: 2 vectors	Storage: Krylov basis
Evaluation of e^{At} at multiple points on interval.	
Can handle mult col B.	Need block Krylov method.
Cost tends to $\Uparrow$ with $\|A\|$.	Some $\|A\|$ dependence.

Conclusions

$$e^A B = (e^{s^{-1}A})^s B \approx \underbrace{T_m(s^{-1}A) \dots T_m(s^{-1}A)}_{s \text{ times}} B.$$

Key ideas: s and m to achieve b'err bound; terminate Taylor series prematurely; shifting and balancing; compute $e^{t_k A}B$ on grid, avoiding overscaling.

- Alg AH is best method in our experiments.
- Very easy to implement.
- Exploits optimized matrix products.
- Attractive for exponential integrators—using theorem on avoidance of φ functions.

A. H. Al-Mohy and N. J. Higham, MIMS EPrint 2010.30, March 2010. Paper and MATLAB code.

References I

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

G. E. Forsythe.
Pitfalls in computation, or why a math book isn't enough.
Amer. Math. Monthly, 77(9):931–956, 1970.

N. J. Higham.
The Matrix Function Toolbox.
`http://www.ma.man.ac.uk/~higham/mftoolbox.`

References II

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

- N. J. Higham.
  Functions of Matrices: Theory and Computation.
  Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.
  ISBN 978-0-898716-46-7.
  xx+425 pp.

- N. J. Higham and A. H. Al-Mohy.
  Computing matrix functions.
  Acta Numerica, 19:159–208, 2010.

References III

N. J. Higham and F. Tisseur.
A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra.
SIAM J. Matrix Anal. Appl., 21(4):1185–1201, 2000.

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

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

References IV

L. N. Trefethen, J. A. C. Weideman, and T. Schmelzer.
Talbot quadratures and rational approximations.
BIT, 46(3):653–670, 2006.