Dubious no more: evaluation of the matrix exponential by solving differential equations

Raymond J. Spiteri

Department of Computer Science, University of Saskatchewan

Go20 Conference on Scientific Computing and Software
Marsalforn, Gozo

May 19–23, 2025

Acknowledgements

- Partners in crime

S. Wei

S. Gaudreault

- Support from

Environment and
Climate Change Canada

NSERC
CRSNG

Outline

Exponential time integrators

$$\frac{\mathrm{d}\mathcal{U}(t)}{\mathrm{d}t} = \mathcal{F}(\mathcal{U}(t)), \quad \mathcal{U}(0) = \mathcal{U}_0$$

Linearize right-hand side:

$$\frac{\mathrm{d}\mathcal{U}}{\mathrm{d}t} = \mathcal{F}_n + \mathcal{J}_n(\mathcal{U}(t) - \mathcal{U}_n) + \mathcal{R}_n(\mathcal{U}(t))$$

Integrating factor:

$$\mathcal{U}_{n+1} = \mathcal{U}_n + \mathcal{J}_n^{-1}(e^{h\mathcal{J}_n} - I)\mathcal{F}_n + \int_{t_n}^{t_n+h} e^{(t_n+h-t)\mathcal{J}_n}\mathcal{R}_n(\mathcal{U}(t))dt$$

Exponential time integrators

Defining $\varphi_1(z) = (e^z - 1)/z$,

$$\mathcal{U}_{n+1} = \mathcal{U}_n + \varphi_1(h\mathcal{J}_n)h\mathcal{F}_n + \int_{t_n}^{t_n+h} e^{(t_n+h-t)A_n}\mathcal{R}_n(\mathcal{U}(t))dt.$$

φ-functions

In general, for $z \in \mathbb{C}$,

$$\varphi_0(z) = e^z,$$
$$\varphi_k(z) = \int_0^1 \frac{e^{(1-\theta)z}\theta^{k-1}}{(k-1)!} d\theta, \ \forall k \in \mathbb{N},$$

and for $A \in \mathbb{C}^{d \times d}$,

$$\varphi_j(A) = \sum_{i=0}^{\infty} \frac{A^i}{(i+j)!}, \ \forall j \in \mathbb{N} \cup \{0\}.$$

EPI methods

Exponential propagation iterative (EPI) methods:

$$\mathcal{U}_{n+1} = \mathcal{U}_n + \varphi_1(h\mathcal{J}_n)h\mathcal{F}_n + \sum_{\ell=2}^{m} \varphi_\ell(h\mathcal{J}_n)\boldsymbol{q}_\ell,$$

$$\boldsymbol{q}_\ell = \sum_{i=1}^{P} \alpha_{\ell,i} h\mathcal{R}_n(\mathcal{U}_{n-i}),$$

where

- P is the number of previous points used
- m is maximum order of φ-functions
- $\mathcal{R}_n(\mathcal{U}_{n-i}) = \mathcal{F}(\mathcal{U}_{n-i}) - \mathcal{F}_n - \mathcal{J}_n(\mathcal{U}_{n-i} - \mathcal{U}_n)$.
- $\alpha_{\ell,i}$ satisfy order conditions

Exponential time integrators

EPI2:

$$\mathcal{U}_{n+1} = \mathcal{U}_n + \varphi_1(h\mathcal{J}_n)h\mathcal{F}_n.$$

EPI3:

$$\mathcal{U}_{n+1} = \mathcal{U}_n + \varphi_1(h\mathcal{J}_n)h\mathcal{F}_n + \frac{2}{3}\varphi_2(h\mathcal{J}_n)h\mathcal{R}_n(\mathcal{U}_{n-1}),$$

where $\mathcal{R}_n(\mathcal{U}_{n-1}) = \mathcal{F}(\mathcal{U}_{n-1}) - \mathcal{F}_n - \mathcal{J}_n(\mathcal{U}_{n-1} - \mathcal{U}_n).$

EPI4–EPI6 exist

Exponential time integrators

- powerful methods for solving ODEs:
  exceptional accuracy and stability properties
- computationally expensive:
  involve $e^{t\mathbf{A}}\mathbf{v}$ for large sparse $\mathbf{A}$

Background

- prevailing way:

  matrix-free / Krylov subspace methods

Background

- prevailing way:

  matrix-free / Krylov subspace methods

- methods/software:
  - KIOPS
    (Krylov with incomplete orthogonalization procedure solver)
  - PMEX (KIOPS with reduced communication)

Background

- dubious way:

  solve a linear non-autonomous ODE

Dubious way?

METHOD 5: GENERAL PURPOSE O.D.E. SOLVER. *Most computer center libraries contain programs for solving initial value problems in ordinary differential equations. Very few libraries contain programs that compute e^{tA}. ...undoubtedly the easiest and, from the programmer's point of view, the quickest way to compute a matrix exponential is to call upon a general purpose o.d.e. solver. This is obviously an expensive luxury since the o.d.e. routine does not take advantage of the linear, constant coefficient nature of our special problem.*

— Moler and Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later (SIREV 45(1), 2003).

Dubious way?

METHOD 5: GENERAL PURPOSE O.D.E. SOLVER. *Most computer center libraries contain programs for solving initial value problems in ordinary differential equations. Very few libraries contain programs that compute* e^{tA}*. . . . undoubtedly the easiest and, from the programmer's point of view, the quickest way to compute a matrix exponential is to call upon a general purpose o.d.e. solver. This is obviously an expensive luxury since the o.d.e. routine does not take advantage of the linear, constant coefficient nature of our special problem.*

— Moler and Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later (SIREV 45(1), 2003).

Dubious way?

METHOD 5: GENERAL PURPOSE O.D.E. SOLVER. *Most computer center libraries contain programs for solving initial value problems in ordinary differential equations. Very few libraries contain programs that compute e^{tA}. ...undoubtedly the easiest and, from the programmer's point of view, the quickest way to compute a matrix exponential is to call upon a general purpose o.d.e. solver. This is obviously an expensive luxury since the o.d.e. routine does not take advantage of the linear, constant coefficient nature of our special problem.*

— Moler and Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later (SIREV 45(1), 2003).

Dubious way?

METHOD 5: GENERAL PURPOSE O.D.E. SOLVER. *Most computer center libraries contain programs for solving initial value problems in ordinary differential equations. Very few libraries contain programs that compute e^{tA}. ...undoubtedly the easiest and, from the programmer's point of view, the quickest way to compute a matrix exponential is to call upon a general purpose o.d.e. solver. This is obviously an expensive luxury since the o.d.e. routine does not take advantage of the linear, constant coefficient nature of our special problem.*

— Moler and Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later (SIREV 45(1), 2003).

Dubious way?

The expression

$$\mathbf{y}(\tau) = \varphi_0(\tau A)\boldsymbol{q}_0 + \tau\varphi_1(\tau A)\boldsymbol{q}_1 + \tau^2\varphi_2(\tau A)\boldsymbol{q}_2 + \ldots + \tau^r\varphi_r(\tau A)\boldsymbol{q}_r$$

is a solution of the initial-value problem

$$\frac{d\mathbf{y}(\tau)}{d\tau} = A\mathbf{y}(\tau) + \boldsymbol{q}_1 + \tau\boldsymbol{q}_2 + \ldots + \frac{\tau^{r-1}}{(r-1)!}\boldsymbol{q}_r, \quad \mathbf{y}(0) = \boldsymbol{q}_0.$$

Then

$$\varphi_0(A)\boldsymbol{q}_0 + \varphi_1(A)\boldsymbol{q}_1 + \varphi_2(A)\boldsymbol{q}_2 + \ldots + \varphi_r(A)\boldsymbol{q}_r = \mathbf{y}(1).$$

Dubious way?

The ODE

$$\frac{d\mathbf{y}(\tau)}{d\tau} = A\mathbf{y}(\tau) + \boldsymbol{q}_1 + \tau\boldsymbol{q}_2 + \ldots + \frac{\tau^{r-1}}{(r-1)!}\boldsymbol{q}_r$$

is linear non-autonomous.

Dubious way?

The ODE

$$\frac{d\mathbf{y}(\tau)}{d\tau} = A\mathbf{y}(\tau) + \boldsymbol{q}_1 + \tau\boldsymbol{q}_2 + \ldots + \frac{\tau^{r-1}}{(r-1)!}\boldsymbol{q}_r$$

is linear non-autonomous.

Dubious way?

The ODE

$$\frac{d\mathbf{y}(\tau)}{d\tau} = A\mathbf{y}(\tau) + \boldsymbol{q}_1 + \tau\boldsymbol{q}_2 + \ldots + \frac{\tau^{r-1}}{(r-1)!}\boldsymbol{q}_r$$

is linear non-autonomous.

Take advantage of structure to design efficient numerical method.

RK order conditions for linear non-autonomous ODEs

The RK order conditions for linear non-autonomous ODEs reduce to

$$\frac{1}{k!}\mathbf{b}\mathbf{A}^i\mathbf{c}^k = \frac{1}{(i+k+1)!}, \quad 0 \le i+k \le p-1,$$

where $\mathbf{A}^i$ denotes matrix exponentiation and $\mathbf{c}^k$ denotes the element-wise exponentiation.

RK order conditions for linear non-autonomous ODEs

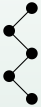
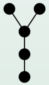
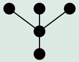
The RK order conditions for linear non-autonomous ODEs reduce to

$$\frac{1}{k!}\mathbf{b}\mathbf{A}^i\mathbf{c}^k = \frac{1}{(i+k+1)!}, \quad 0 \leq i+k \leq p-1,$$

where $\mathbf{A}^i$ denotes matrix exponentiation and $\mathbf{c}^k$ denotes the element-wise exponentiation.

- tall trees
- palm trees
- bushy trees

RK order conditions for linear non-autonomous ODEs

	Graph	Order Condition
"tall" tree		$\frac{1}{1!}\mathbf{bA}^3\mathbf{c}^1 = \frac{1}{5!}$
"palm" trees		$\frac{1}{2!}\mathbf{bA}^2\mathbf{c}^2 = \frac{1}{5!}$
		$\frac{1}{3!}\mathbf{bA}^1\mathbf{c}^3 = \frac{1}{5!}$
"bushy" tree		$\frac{1}{4!}\mathbf{bA}^0\mathbf{c}^4 = \frac{1}{5!}$

Explicit Runge–Kutta Methods: Error control

Embedded Runge–Kutta methods

- estimate local error at each step from difference between numerical solutions of order p and $q = p - 1$
- share stages to minimize additional work
- must match truncation errors / stability regions to be effective
- first-same-as-last (FSAL) for added efficiency

Explicit Runge–Kutta Methods: Error control

Embedded Runge–Kutta methods

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
\vdots & \vdots & \ddots & & & \\
c_s & a_{s1} & \dots & a_{s,s-1} & & \\
\hline
 & b_1 & \dots & b_{s-1} & b_s & \\
\hline
 & \hat{b}_1 & \dots & \hat{b}_{s-1} & \hat{b}_s & \hat{b}_{s+1}
\end{array}
\quad = \quad
\begin{array}{c|c}
\mathbf{c} & \mathbf{A} \\
\hline
 & \mathbf{b} \\
\hline
 & \hat{\mathbf{b}}
\end{array}
$$

Explicit Runge–Kutta Methods: Step-size control

- based on linear digital control theory
- generate smooth and regular step size sequences

$$h_{n+1} = g \left(\frac{\epsilon}{r_n} \right)^{\beta_1} \left(\frac{\epsilon}{r_{n-1}} \right)^{\beta_2} \left(\frac{h_n}{h_{n-1}} \right)^{-\alpha_2} h_n$$

- $\epsilon =$ fraction of the user-defined tolerance
- $r_n =$ local error estimate
- β_i and α_2: satisfy conditions for adaptivity, low-pass filtering
- arctangent limiter:

$$\hat{h}_{n+1} \leftarrow 1 + \kappa \arctan \left(\frac{h_{n+1} - 1}{\kappa} \right)$$

κ controls strength of limiter ($\kappa = 2$)

Explicit Runge–Kutta Methods: Step-size control

Controller	**Parameters** $(k\beta_1, k\beta_2, \alpha_2, g)$	**Description**
deadbeat	$(1, 0, 0, 0.9)$	Elementary controller
PI3040	$(0.7, -0.4, 0, 0.8)$	PI controller
PI4020	$(0.6, -0.2, 0, 0.8)$	PI controller non-stiff
H211PI	$\left(\frac{1}{6}, \frac{1}{6}, 0, 0.8\right)$	LP filter of PI structure
H110	$\left(\frac{1}{3}, 0, 0, 0.8\right)$	I controller (convolution filter)
H211D	$\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0.8\right)$	LP filter with gain $= 1/2$
H211B	$\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0.8\right)$	General purpose LP filter

$k = p + 1$, where p is order of auxiliary method.

Embedded Runge–Kutta Methods: Design

Order condition errors:

$$\epsilon_{i,k}^{(p)} = \frac{1}{k!}\mathbf{b}\mathbf{A}^i\mathbf{c}^k - \frac{1}{p!}, \qquad i+k = p-1, i = 0, 1, \ldots, p-2$$

$$\hat{\epsilon}_{i,k}^{(q)} = \frac{1}{k!}\hat{\mathbf{b}}\hat{\mathbf{A}}^i\hat{\mathbf{c}}^k - \frac{1}{q!} \qquad i+k = q-1, i = 0, 1, \ldots, q-2,$$

where $\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{b} & 0 \end{bmatrix}$ and $\hat{\mathbf{c}} = \begin{bmatrix} \mathbf{c} \\ 1 \end{bmatrix}$.

Principal errors:

$$\mathcal{A}^{(p+1)} = ||\boldsymbol{\epsilon}^{(p+1)}||_2 := \sqrt{\sum_{i=0}^{p-1}(\epsilon_{i,k}^{(p+1)})^2},$$

$$\hat{\mathcal{A}}^{(q+1)} = ||\hat{\boldsymbol{\epsilon}}^{(q+1)}||_2 := \sqrt{\sum_{i=0}^{q-1}(\hat{\epsilon}_{i,k}^{(q+1)})^2}.$$

Embedded Runge–Kutta Methods: Design

Other useful characteristics:

$$\mathcal{B}^{(q+2)} = \frac{\hat{\mathcal{A}}^{(q+2)}}{\hat{\mathcal{A}}^{(q+1)}},$$

$$\mathcal{C}^{(q+2)} = \frac{||\hat{\epsilon}^{(q+2)} - \epsilon^{(q+2)}||_2}{\hat{\mathcal{A}}^{(q+1)}},$$

$$\mathcal{D} = \max\{|a_{ij}|, |b_i|, |\hat{b}_i|, |c_i|\},$$

$$\mathcal{E}^{(q+2)} = \frac{\mathcal{A}^{(q+2)}}{\hat{\mathcal{A}}^{(q+2)}}.$$

Embedded Runge–Kutta Methods: Design

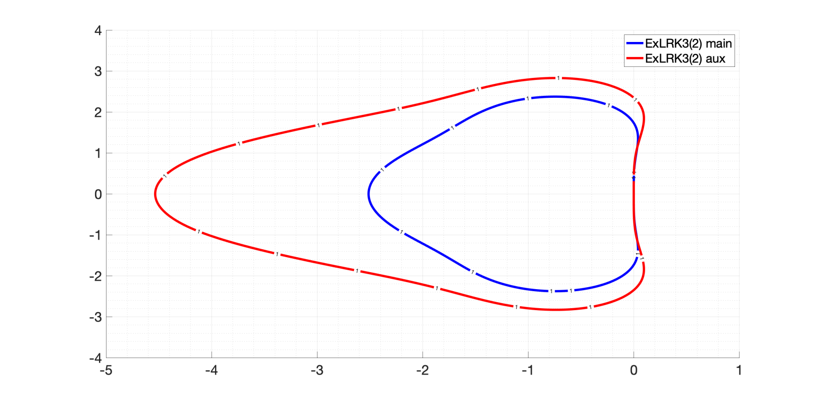
Strategy:

1) For a given order p, construct a s-stage explicit Runge–Kutta method by minimizing the principal error $\mathcal{A}^{(p+1)}$.
2) Construct an $(s+1)$-stage auxiliary method of order $q = p - 1$ such that its stability region (marginally) covers that of the main method.
3) Ensure the other characteristics $\mathcal{B}^{(q+2)}$, $\mathcal{C}^{(q+2)}$, $\mathcal{E}^{(q+2)}$ are order unity, and $\mathcal{D}$ is less than 20.

ExLRK3(2)

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
1 & -1 & 2 & & \\
\hline
 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & \\
\hline
 & \frac{1}{4} & \frac{1}{2} & \frac{1}{10} & \frac{3}{20}
\end{array}
$$

ExLRK3(2)

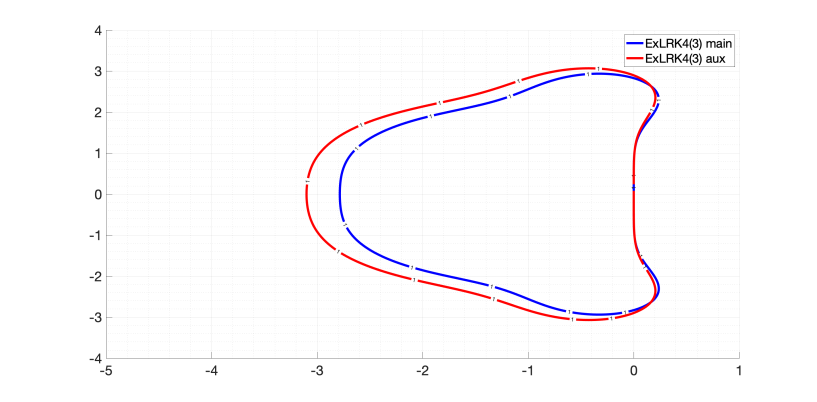


Stability regions of ExLRK3(2).

ExLRK4(3)

$$
\begin{array}{c|ccccc}
0 & & & & & \\
\frac{1}{2} & \frac{1}{2} & & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & & \\
1 & 0 & 0 & 1 & & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{41}{300} & \frac{3}{100}
\end{array}
$$

ExLRK4(3)



Stability regions of ExLRK4(3).

Advection-diffusion-reaction problem

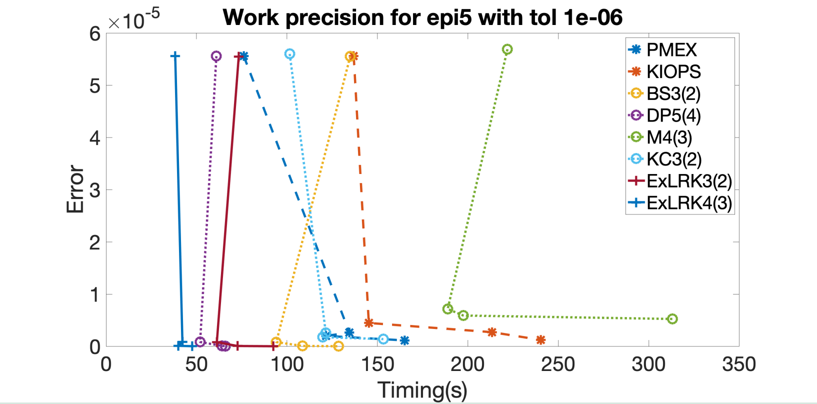
$$\frac{\partial u}{\partial t} = -\alpha \nabla \cdot u + \epsilon \nabla^2 u + \gamma u \left(u - \frac{1}{2} \right) (1 - u),$$

- $t \in [0, 0.1]$, $(x, y) \in [0, 1]^2$
- homogeneous Neumann boundary conditions
- initial condition

$$u(x, y, 0) = 256 \left(xy(1 - x)(1 - y) \right)^2 + 0.3,$$

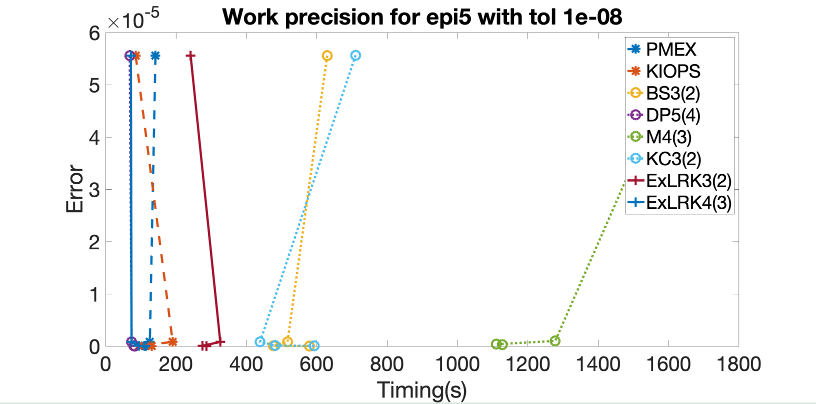
- central finite differences on uniform grid $\Delta x = \Delta y = 1/400$
- $\alpha = -10$, $\epsilon = 1/100$, $\gamma = 100$

Advection-diffusion-reaction problem



2D ADR problem: EPI5 with tolerance $1e{-}6$

Advection-diffusion-reaction problem



2D ADR problem: EPI5 with tolerance $1e{-}8$

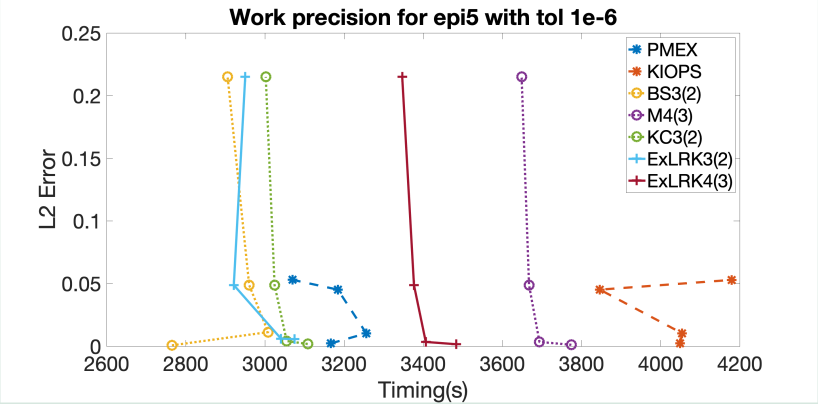
Rossby–Haurwitz wave

- Rossby–Haurwitz wave number 4
- analytical solution to non-linear barotropic vorticity equation
- well-known susceptibility to instabilities due to truncation in initial conditions; numerical solution eventually loses structure
- nonetheless, solution expected to remain stable over 14 days
- wave number 4 expected to propagate steadily and largely retain structure

Rossby–Haurwitz wave

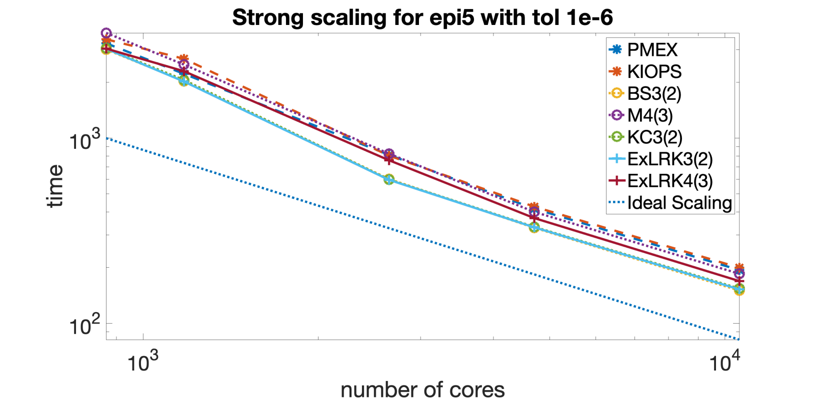
- $\Delta t = 900$ s
- tolerance $= 1\text{e}{-6}$
- $N_{\text{proc}} = [864, 1176, 2646, 4704, 10584]$

Rossby–Haurwitz wave



Rossby–Haurwitz wave: EPI5 with tolerance $1e{-}6$

Rossby–Haurwitz wave



Rossby–Haurwitz wave: EPI5 with tolerance $1e{-6}$

Conclusions

- evaluate φ-functions by solving a linear ODE
- design customized embedded explicit Runge–Kutta methods
- methods scale well, can outperform competitors