Design of optimal Runge-Kutta methods

David I. Ketcheson

King Abdullah University of Science & Technology (KAUST)
Acknowledgments

Some parts of this are joint work with:

- Aron Ahmadia
- Matteo Parsani
Outline

1. High order Runge-Kutta methods
2. Linear properties of Runge-Kutta methods
3. Nonlinear properties of Runge-Kutta methods
4. Putting it all together: some optimal methods and applications
1. High order Runge-Kutta methods

2. Linear properties of Runge-Kutta methods

3. Nonlinear properties of Runge-Kutta methods

4. Putting it all together: some optimal methods and applications
Solution of hyperbolic PDEs

The fundamental algorithmic barrier is the CFL condition:

$$\Delta t \leq a\Delta x$$

- Implicit methods don’t usually help (due to reduced accuracy)
Solution of hyperbolic PDEs

The fundamental algorithmic barrier is the CFL condition:

\[ \Delta t \leq a\Delta x \]

- Implicit methods don’t usually help (due to reduced accuracy)
- Strong scaling limits the effectiveness of spatial parallelism alone
The fundamental algorithmic barrier is the CFL condition:

\[ \Delta t \leq a\Delta x \]

- Implicit methods don’t usually help (due to reduced accuracy)
- Strong scaling limits the effectiveness of spatial parallelism alone
- Strategy: keep \( \Delta x \) as large as possible by using high order methods
The fundamental algorithmic barrier is the CFL condition:

\[ \Delta t \leq a \Delta x \]

- Implicit methods don’t usually help (due to reduced accuracy)
- Strong scaling limits the effectiveness of spatial parallelism alone
- Strategy: keep \( \Delta x \) as large as possible by using high order methods
- But: high order methods cost more and require more memory
The fundamental algorithmic barrier is the CFL condition:

\[ \Delta t \leq a\Delta x \]

- Implicit methods don’t usually help (due to reduced accuracy)
- Strong scaling limits the effectiveness of spatial parallelism alone
- Strategy: keep \( \Delta x \) as large as possible by using high order methods
- But: high order methods cost more and require more memory
- Can we develop high order methods that are as efficient as lower order methods?
Using a better time integrator is usually simple but can be highly beneficial.
Using a better time integrator is usually simple but can be highly beneficial.
Using a different time integrator can:

- Reduce the number of RHS evaluations required
Using a better time integrator is usually simple but can be highly beneficial.

Using a different time integrator can:

- Reduce the number of RHS evaluations required
- Alleviate timestep restrictions due to
Using a better time integrator is usually simple but can be highly beneficial.

Using a different time integrator can:
- Reduce the number of RHS evaluations required
- Alleviate timestep restrictions due to
  - Linear stability
Using a better time integrator is usually simple but can be highly beneficial.

Using a different time integrator can:
- Reduce the number of RHS evaluations required
- Alleviate timestep restrictions due to
  - Linear stability
  - Nonlinear stability
Using a better time integrator is usually simple but can be highly beneficial.

Using a different time integrator can:

- Reduce the number of RHS evaluations required
- Alleviate timestep restrictions due to
  - Linear stability
  - Nonlinear stability
- Improve accuracy (truncation error, dispersion, dissipation)
Using a better time integrator is usually simple but can be highly beneficial.

Using a different time integrator can:

- Reduce the number of RHS evaluations required
- Alleviate timestep restrictions due to
  - Linear stability
  - Nonlinear stability
- Improve accuracy (truncation error, dispersion, dissipation)
- Reduce storage requirements
To solve the initial value problem:

\[ u'(t) = F(u(t)), \quad u(0) = u^0 \]

a Runge-Kutta method computes approximations \( u^n \approx u(n\Delta t) \):

\[
y^i = u^n + \Delta t \sum_{j=1}^{i-1} a_{ij} F(y^j)
\]

\[
u^{n+1} = u^n + \Delta t \sum_{j=1}^{s-1} b_j F(y^j)
\]

The accuracy and stability of the method depend on the coefficient matrix \( A \) and vector \( b \).
Runge-Kutta Methods: a philosophical aside

- An RK method builds up information about the solution derivatives through the computation of intermediate stages.
- At the end of a step all of this information is thrown away!
- Use more stages $\Rightarrow$ keep information around longer.
Outline

1. High order Runge-Kutta methods
2. Linear properties of Runge-Kutta methods
3. Nonlinear properties of Runge-Kutta methods
4. Putting it all together: some optimal methods and applications
The Stability Function

For the linear equation

\[ u' = \lambda u, \]

a Runge-Kutta method yields a solution

\[ u^{n+1} = \phi(\lambda \Delta t) u^n, \]

where \( \phi \) is called the \textit{stability function} of the method:

\[ \phi(z) = \frac{\det(I - z(A - eb^T))}{\det(I - zA)} \]
The Stability Function

For the linear equation
\[ u' = \lambda u, \]
a Runge-Kutta method yields a solution
\[ u^{n+1} = \phi(\lambda \Delta t)u^n, \]
where \( \phi \) is called the *stability function* of the method:
\[ \phi(z) = \frac{\det(I - z(A - eb^T))}{\det(I - zA)} \]

Example: **Euler’s Method**
\[ u^{n+1} = u^n + \Delta tF(u); \quad \phi(z) = 1 + z. \]
The Stability Function

For the linear equation

\[ u' = \lambda u, \]

a Runge-Kutta method yields a solution

\[ u^{n+1} = \phi(\lambda \Delta t) u^n, \]

where \( \phi \) is called the \textit{stability function} of the method:

\[ \phi(z) = \frac{\det(I - z(A - eb^T))}{\det(I - zA)} \]

Example: \textbf{Euler’s Method}

\[ u^{n+1} = u^n + \Delta tF(u); \quad \phi(z) = 1 + z. \]

For explicit methods of order \( p \):

\[ \phi(z) = \sum_{j=0}^{p} \frac{1}{j!} z^j + \sum_{j=p+1}^{s} \alpha_j z^j. \]
Absolute Stability

For the linear equation

\[ u'(t) = Lu \]

we say the solution is absolutely stable if \(|\phi(\lambda \Delta t)| \leq 1 \) for all \( \lambda \in \sigma(L) \).
Absolute Stability

For the linear equation

$$u'(t) = Lu$$

we say the solution is absolutely stable if $$|\phi(\lambda \Delta t)| \leq 1$$ for all $$\lambda \in \sigma(L)$$.

Example: Euler’s Method

$$u^{n+1} = u^n + \Delta t F(u); \quad \phi(z) = 1 + z.$$
This leads naturally to the following problem.

\[
\text{maximize } \Delta t \\
\text{subject to } |\phi(\Delta t \lambda)| - 1 \leq 0, \quad \lambda \in \sigma(L),
\]

where \( \phi(z) = \sum_{j=0}^{p} \frac{1}{j!} z^j + \sum_{j=p+1}^{s} \alpha_j z^j. \)

Here the decision variables are \( \Delta t \) and the coefficients \( \alpha_j, j = p + 1, \ldots, s. \) This problem is quite difficult; we approximate its solution by solving a sequence of convex problems (DK & A. Ahmadian, arXiv preprint).
Accuracy optimization

We could instead optimize accuracy over some region in $\mathbb{C}$:

Given $L$, $p$, $s$,

maximize $\Delta t$

subject to $|\phi(\Delta t \lambda) - \exp(\Delta t \lambda)| \leq \epsilon$, $\lambda \in \sigma(L)$,

where $\phi(z) = \sum_{j=0}^{p} \frac{1}{j!} z^j + \sum_{j=p+1}^{s} \alpha_j z^j$.

In the PDE case, we can replace $\exp(\Delta t \lambda)$ with the exact dispersion relation for each Fourier mode.
Stability Optimization: a toy example

As an example, consider the advection equation

$$u_t + u_x = 0$$

discretized in space by first-order upwind differencing with unit spatial mesh size

$$U'_i(t) = -(U_i(t) - U_{i-1}(t))$$

with periodic boundary condition $$U_0(t) = U_N(t)$$. 
Stability Optimization: a toy example

(a) RK(4,4)  
(b) Optimized 10-stage method
Stability Optimization: a toy example

What is the relative efficiency?

\[
\text{Stable step size} \quad \frac{1.4}{4} \approx 0.35 \\
\text{Cost per step} \\
RK(4,4): \quad \frac{6}{10} = 0.6
\]

By allowing even more stages, can asymptotically approach the efficiency of Euler’s method.
Second order discontinuous Galerkin discretization of advection:
Stability Optimization: one more example

\[ s = 20 \]
Stability Optimization: one more example

\[ s = 20 \]
Stability Optimization: one more example

\[ s = 20 \]
1. High order Runge-Kutta methods
2. Linear properties of Runge-Kutta methods
3. Nonlinear properties of Runge-Kutta methods
4. Putting it all together: some optimal methods and applications
Nonlinear accuracy

Besides the conditions on the stability polynomial coefficients, high order Runge-Kutta methods must satisfy additional nonlinear order conditions.

- $p = 1$: $\sum_i b_i = 1$
- $p = 2$: $\sum_{i,j} b_i a_{ij} = 1/2$
- $p = 3$: $\sum_{i,j,k} b_i a_{ij} a_{jk} = 1/6$
  \[ \sum_{i,j,k} b_i a_{ij} a_{ik} = 1/3 \]

Number of conditions grows factorially (719 conditions for order 10).
Beyond linear stability

Classical stability theory and its extensions focus on
- weak bounds: $\|u^n\| \leq C(t)$
- linear problems
- inner product norms

For hyperbolic PDEs, we are often interested in
- strict bounds $\|u^n\| \leq C$
- nonlinear problems
- $L_1, L_\infty, TV$, or positivity

We refer to bounds of the latter types as strong stability properties. For example:

$$\|u^n\|_{TV} \leq \|u^{n-1}\|_{TV}$$
Strong stability preservation

Designing fully-discrete schemes with strong stability properties is notoriously difficult!
Designing fully-discrete schemes with strong stability properties is notoriously difficult!
Instead, one often takes a method-of-lines approach and assumes explicit Euler time integration.

Strong stability preserving methods provide higher order accuracy while maintaining any convex functional bound satisfied by Euler time stepping.
Designing fully-discrete schemes with strong stability properties is notoriously difficult! Instead, one often takes a method-of-lines approach and assumes explicit Euler time integration.

But in practice, we need to use higher order methods, for reasons of both accuracy and linear stability.
Designing fully-discrete schemes with strong stability properties is notoriously difficult!
Instead, one often takes a method-of-lines approach and assumes explicit Euler time integration.

But in practice, we need to use higher order methods, for reasons of both accuracy and linear stability. **Strong stability preserving** methods provide higher order accuracy while maintaining any convex functional bound satisfied by Euler timestepping.
The Forward Euler condition

Recall our ODE system (typically from a PDE)

\[ u_t = F(u), \]

where the spatial discretization \( F(u) \) is carefully chosen\(^1\) so that the solution from the forward Euler method

\[ u^{n+1} = u^n + \Delta t F(u^n), \]

satisfies the monotonicity requirement

\[ ||u^{n+1}|| \leq ||u^n||, \]

in some norm, semi-norm or convex functional \( || \cdot || \), for a suitably restricted timestep

\[ \Delta t \leq \Delta t_{FE}. \]

\(^1\) e.g. TVD, TVB
Consider the two-stage method:

\[
\begin{align*}
y^1 &= u^n + \Delta t F(u^n) \\
u^{n+1} &= u^n + \frac{1}{2} \Delta t \left( F(u^n) + F(y^1) \right)
\end{align*}
\]

Is \( ||u^{n+1}|| \leq ||u^n|| \)?
Consider the two-stage method:

\[
\begin{align*}
y^1 &= u^n + \Delta t F(u^n) \\
u^{n+1} &= \frac{1}{2} u^n + \frac{1}{2} (y^1 + \Delta t F(y^1)).
\end{align*}
\]

Take $\Delta t \leq \Delta t_{FE}$. Then $\|y^1\| \leq \|u^n\|$, so

\[
\|u^{n+1}\| \leq \frac{1}{2} \|u^n\| + \frac{1}{2} \|y^1 + \Delta t F(y^1)\| \leq \|u^n\|.
\]
In general, an SSP method preserves strong stability properties satisfied by Euler’s method, under a modified step size restriction:

$$\Delta t \leq C \Delta t_{FE}.$$  

A fair metric for comparison is the effective SSP coefficient:

$$C_{eff} = \frac{C}{\# \text{ of stages}}.$$  

By designing high order methods with many stages, we can achieve $C_{eff} \to 1$. 

D. Ketcheson (KAUST)
Example: A highly oscillatory flow field

\[ u_t + (\cos^2(20x + 45t)u)_x = 0 \quad u(0, t) = 0 \]

<table>
<thead>
<tr>
<th>Method</th>
<th>( c_{\text{eff}} )</th>
<th>Monotone effective timestep</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSSP(3,2)</td>
<td>0</td>
<td>0.037</td>
</tr>
<tr>
<td>SSP(50,2)</td>
<td>0.980</td>
<td>0.980</td>
</tr>
<tr>
<td>NSSP(3,3)</td>
<td>0</td>
<td>0.004</td>
</tr>
<tr>
<td>NSSP(5,3)</td>
<td>0</td>
<td>0.017</td>
</tr>
<tr>
<td>SSP(64,3)</td>
<td>0.875</td>
<td>0.875</td>
</tr>
<tr>
<td>RK(4,4)</td>
<td>0</td>
<td>0.287</td>
</tr>
<tr>
<td>SSP(5,4)</td>
<td>0.302</td>
<td>0.416</td>
</tr>
<tr>
<td>SSP(10,4)</td>
<td>0.600</td>
<td>0.602</td>
</tr>
</tbody>
</table>
Low storage methods

- Straightforward implementation of an $s$-stage RK method requires $s + 1$ memory locations per unknown.

- Special low-storage methods are designed so that each stage only depends on one or two most recent previous stages.

- Thus older stages can be discarded as the new ones are computed.

- It is often desirable to:
  - Keep the previous solution around to be able to restart a step.
  - Compute an error estimate.

- This requires a minimum of three storage locations per unknown.
Low storage methods

3S Algorithm

\[ S_3 := u^n \]

\((y_1)\) \[ S_1 := u^n \]

\text{for } i = 2 : m + 1 \text{ do}

\[ S_2 := S_2 + \delta_{i-1}S_1 \]

\((y_i)\) \[ S_1 := \gamma_{i1}S_1 + \gamma_{i2}S_2 + \gamma_{i3}S_3 + \beta_{i,i-1}\Delta tF(S_1) \]

\text{end}

\((\hat{u}^{n+1})\) \[ S_2 := \frac{1}{\sum_{j=1}^{m+2} \delta_j} (S_2 + \delta_{m+1}S_1 + \delta_{m+2}S_3) \]

\[ u^{n+1} = S_1 \]
Outline

1. High order Runge-Kutta methods
2. Linear properties of Runge-Kutta methods
3. Nonlinear properties of Runge-Kutta methods
4. Putting it all together: some optimal methods and applications
Our optimization approach proceeds in two steps:

1. Optimize the linear stability or accuracy of the scheme by choosing the stability polynomial coefficients $\alpha_j$.
Our optimization approach proceeds in two steps:

1. Optimize the linear stability or accuracy of the scheme by choosing the stability polynomial coefficients $\alpha_j$.

2. Optimize the nonlinear stability/accuracy and storage requirements by choosing the Butcher coefficients $a_{ij}, b_j$. 
Our optimization approach proceeds in two steps:

1. Optimize the linear stability or accuracy of the scheme by choosing the stability polynomial coefficients $\alpha_j$.
2. Optimize the nonlinear stability/accuracy and storage requirements by choosing the Butcher coefficients $a_{ij}, b_j$. 
Two-step optimization process

Our optimization approach proceeds in two steps:

1. Optimize the linear stability or accuracy of the scheme by choosing the stability polynomial coefficients $\alpha_j$.
2. Optimize the nonlinear stability/accuracy and storage requirements by choosing the Butcher coefficients $a_{ij}, b_j$.

Each of these steps is a complex numerical problem in itself, involving nonconvex optimization in dozens to hundreds of variables, with nonlinear equality and inequality constraints.
Optimizing for the SD spectrum

- On regular grids, SD leads to a block-Toeplitz operator
- We perform a von Neumann-like analysis using a "generating pattern"

\[
\frac{dW_{i,j}}{dt} + \frac{a}{\Delta g} \left( T_{0,0}^{0,0} W_{i,j} + T_{-1,0}^{-1,0} W_{i-1,j} + T_{0,-1}^{0,-1} W_{i,j-1} \\
+ T_{1,0}^{1,0} W_{i+1,j} + T_{0,1}^{0,+1} W_{i,j+1} \right) = 0
\]
Optimizing for the SD spectrum

- Blue: eigenvalues; Red: RK stability boundary
- The convex hull of the generated spectrum is used as a proxy to accelerate the optimization process
Primarily optimized for stable step size
Secondary optimization for nonlinear accuracy and low-storage (3 memory locations per unknown)
Application: flow past a wedge

fully unstructured mesh
Application: flow past a wedge

Density at $t = 100$

- 62% speedup using optimized method
Conclusions

- Numerical optimization allows for flexible, targeted design of time integrators
- Stability optimization based on spectra from a model (linear) problem on a uniform grid seems to work well even for nonlinear problems on fully unstructured grids
- Significant speedup can be achieved in practice (greater for higher order methods)