Design of optimal Runge-Kutta methods

David I. Ketcheson

King Abdullah University of Science & Technology
(KAUST)

King Abdullah University of Science and 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

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

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

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
- Strategy: keep Δx as large as possible by using high order methods

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
- Strategy: keep Δx as large as possible by using high order methods
- But: high order methods cost more and require more memory

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
- Strategy: keep Δx 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?

Time Integration

Using a better time integrator is usually simple but can be highly beneficial.

Time Integration

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

Time Integration

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 resrictions due to

Time Integration

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 resrictions due to
  - Linear stability

Time Integration

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 resrictions due to
  - Linear stability
  - Nonlinear stability

Time Integration

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 resrictions due to
  - Linear stability
  - Nonlinear stability
- Improve accuracy (truncation error, dispersion, dissipation)

Time Integration

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 resrictions due to
  - Linear stability
  - Nonlinear stability
- Improve accuracy (truncation error, dispersion, dissipation)
- Reduce storage requirements

Runge-Kutta Methods

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 $\implies$ 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 ϕ is called the *stability function* of the method:

$$\phi(z) = \frac{\det(\mathbf{I} - z(\mathbf{A} - \mathbf{e}\mathbf{b}^{\mathrm{T}}))}{\det(\mathbf{I} - z\mathbf{A})}$$

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 ϕ is called the *stability function* of the method:

$$\phi(z) = \frac{\det(\mathbf{I} - z(\mathbf{A} - \mathbf{e}\mathbf{b}^{\mathrm{T}}))}{\det(\mathbf{I} - z\mathbf{A})}$$

Example: **Euler's Method**

$$u^{n+1} = u^n + \Delta t F(u); \quad \phi(z) = 1 + z.$$

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 ϕ is called the *stability function* of the method:

$$\phi(z) = \frac{\det(\mathbf{I} - z(\mathbf{A} - \mathbf{e}\mathbf{b}^{\mathrm{T}})}{\det(\mathbf{I} - z\mathbf{A})}$$

Example: **Euler's Method**

$$u^{n+1} = u^n + \Delta t F(u); \quad \phi(z) = 1 + z.$$

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 $\lambda \in \sigma(L)$.

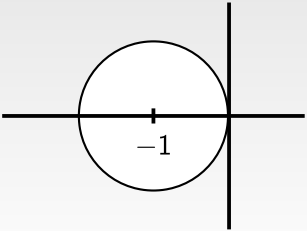
Absolute Stability

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**

$$u^{n+1} = u^n + \Delta t F(u); \quad \phi(z) = 1 + z.$$



Stability optimization

This leads naturally to the following problem.

Stability optimization

Given L, p, s,

$$\begin{aligned} \text{maximize} \quad & \Delta t \\ \text{subject to} \quad & |\phi(\Delta t \lambda)| - 1 \leq 0, \qquad \lambda \in \sigma(L), \\ \text{where} \quad & \phi(z) = \sum_{j=0}^{p} \frac{1}{j!} z^j + \sum_{j=p+1}^{s} \alpha_j z^j. \end{aligned}$$

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

Accuracy optimization

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

Accuracy optimization

Given L, p, s,

$$\begin{aligned} \text{maximize} \quad & \Delta t \\ \text{subject to} \quad & |\phi(\Delta t\lambda) - \exp(\Delta t\lambda| \le \epsilon, \qquad \lambda \in \sigma(L), \\ \text{where} \quad & \phi(z) = \sum_{j=0}^{p} \frac{1}{j!} z^j + \sum_{j=p+1}^{s} \alpha_j z^j. \end{aligned}$$

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

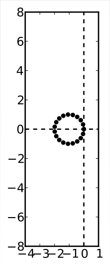
As an example, consider the advection equation

$$u_t + u_x = 0$$

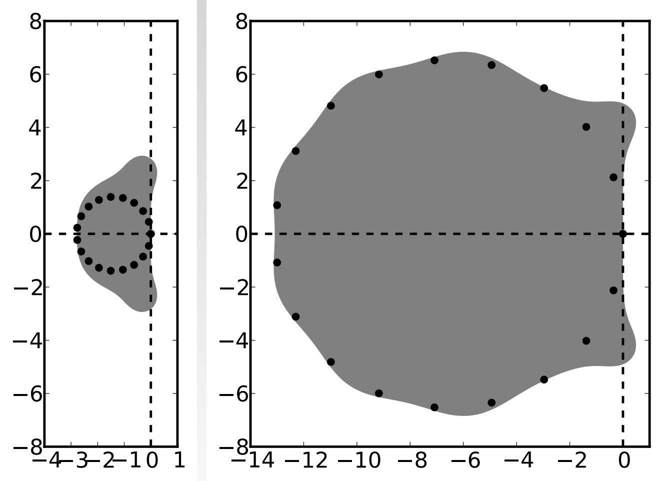
discretized in space by first-order upwind differencing with unit spatial mesh size

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

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?

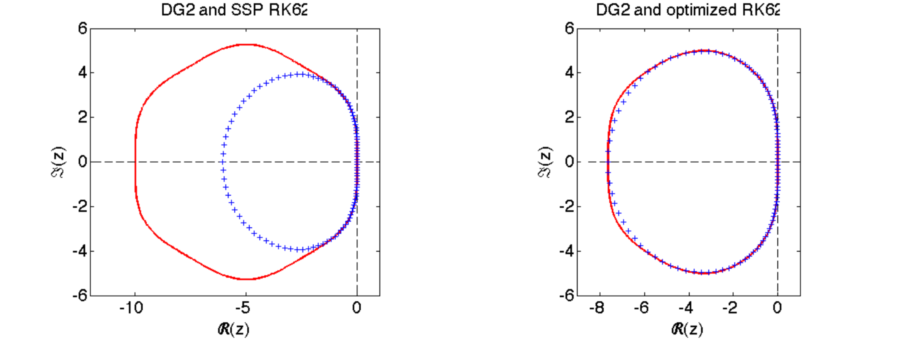
$$\frac{\text{Stable step size}}{\text{Cost per step}}$$

$$\text{RK(4,4):}\ \frac{1.4}{4} \approx 0.35$$

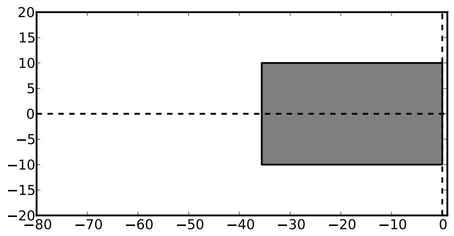
$$\text{RK(10,4):}\ \frac{6}{10} = 0.6$$

By allowing even more stages, can asymptotically approach the efficiency of Euler's method.

Second order discontinuous Galerkin discretization of advection:

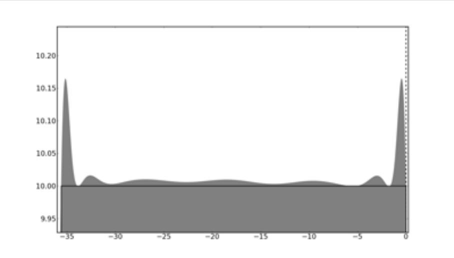


Stability Optimization: one more example



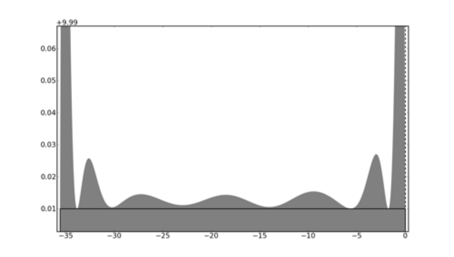
$s = 20$

Stability Optimization: one more example



$s = 20$

Stability Optimization: one more example



$$s = 20$$

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

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_∞, TV, or positivity

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

$$\|u^n\|_{\mathrm{TV}} \leq \|u^{n-1}\|_{\mathrm{TV}}$$

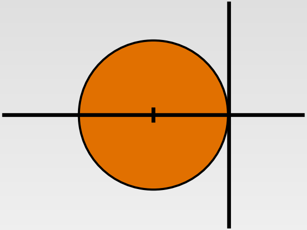
Strong stability preservation

Designing fully-discrete schemes with strong stability properties is notoriously difficult!

Strong stability preservation

Designing fully-discrete schemes with strong stability properties is notoriously difficult!

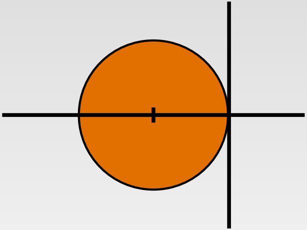
Instead, one often takes a method-of-lines approach and assumes **explicit Euler** time integration.



Strong stability preservation

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.

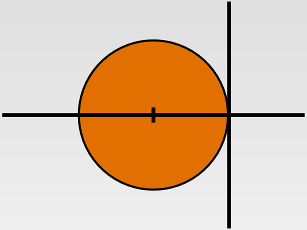


But in practice, we need to use higher order methods, for reasons of both accuracy and linear stability.

Strong stability preservation

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)

$$\mathbf{u}_t = F(\mathbf{u}),$$

where the spatial discretization $F(\mathbf{u})$ is carefully chosen[1] so that the solution from the forward Euler method

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t F(\mathbf{u}^n),$$

satisfies the monotonicity requirement

$$||\mathbf{u}^{n+1}|| \le ||\mathbf{u}^n||,$$

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

$$\Delta t \le \Delta t_{\mathsf{FE}}.$$

[1] e.g. TVD, TVB

Runge–Kutta methods as a convex combination of Euler

Consider the two-stage method:

$$\begin{aligned}
\mathbf{y}^1 &= \mathbf{u}^n + \Delta t F(\mathbf{u}^n) \\
\mathbf{u}^{n+1} &= \mathbf{u}^n + \frac{1}{2}\Delta t \left(F(\mathbf{u}^n) + F(\mathbf{y}^1)\right)
\end{aligned}$$

Is $||\mathbf{u}^{n+1}|| \leq ||\mathbf{u}^n||$?

Runge–Kutta methods as a convex combination of Euler

Consider the two-stage method:

$$\begin{aligned}
\mathbf{y}^1 &= \mathbf{u}^n + \Delta t F(\mathbf{u}^n) \\
\mathbf{u}^{n+1} &= \frac{1}{2}\mathbf{u}^n + \frac{1}{2}\left(\mathbf{y}^1 + \Delta t F(\mathbf{y}^1)\right).
\end{aligned}$$

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

$$||\mathbf{u}^{n+1}|| \le \frac{1}{2}||\mathbf{u}^n|| + \frac{1}{2}||\mathbf{y}^1 + \Delta t F(\mathbf{y}^1)|| \le ||\mathbf{u}^n||.$$

$$||\mathbf{u}^{n+1}|| \le ||\mathbf{u}^n||$$

Optimized SSP methods

In general, an SSP method preserves strong stability properties satisfied by Euler's method, under a modified step size restriction:

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

A fair metric for comparison is the *effective SSP coefficient*:

$$\mathcal{C}_{\text{eff}} = \frac{\mathcal{C}}{\# \text{ of stages}}$$

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

Example: A highly oscillatory flow field

$$u_t + \left(\cos^2(20x + 45t)u\right)_x = 0 \qquad\qquad u(0,t) = 0$$

Method	c_{eff}	Monotone effective timestep
NSSP(3,2)	0	0.037
SSP(50,2)	0.980	0.980
NSSP(3,3)	0	0.004
NSSP(5,3)	0	0.017
SSP(64,3)	0.875	0.875
RK(4,4)	0	0.287
SSP(5,4)	0.302	0.416
SSP(10,4)	0.600	0.602

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

3S Algorithm

$$
\begin{array}{ll}
 & \mathtt{S}_3 := \mathtt{u}^{\mathtt{n}} \\
(y_1) & \mathtt{S}_1 := \mathtt{u}^{\mathtt{n}} \\
 & \mathtt{for\ i} = 2 : \mathtt{m} + 1\ \mathtt{do} \\
 & \quad \mathtt{S}_2 := \mathtt{S}_2 + \delta_{\mathtt{i}-1}\mathtt{S}_1 \\
(y_i) & \quad \mathtt{S}_1 := \gamma_{\mathtt{i}1}\mathtt{S}_1 + \gamma_{\mathtt{i}2}\mathtt{S}_2 + \gamma_{\mathtt{i}3}\mathtt{S}_3 + \beta_{\mathtt{i},\mathtt{i}-1}\Delta\mathtt{tF}(\mathtt{S}_1) \\
 & \mathtt{end} \\
(\hat{u}^{n+1}) & \quad \mathtt{S}_2 := \frac{1}{\sum_{\mathtt{j}=1}^{\mathtt{m}+2}\delta_{\mathtt{j}}}\left(\mathtt{S}_2 + \delta_{\mathtt{m}+1}\mathtt{S}_1 + \delta_{\mathtt{m}+2}\mathtt{S}_3\right) \\
 & u^{n+1} = \mathtt{S}_1
\end{array}
$$

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

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 α_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 α_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 α_j
2. Optimize the nonlinear stability/accuracy and storage requirements by choosing the Butcher coefficients a_{ij}, b_j.

Two-step optimization process

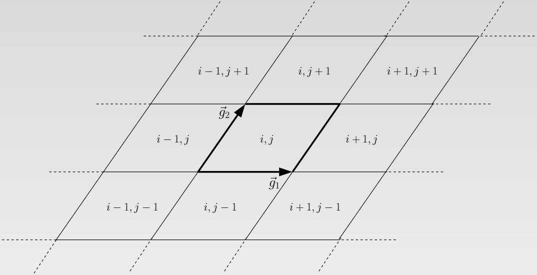
Our optimization approach proceeds in two steps:

1. Optimize the linear stability or accuracy of the scheme by choosing the stability polynomial coefficients α_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.

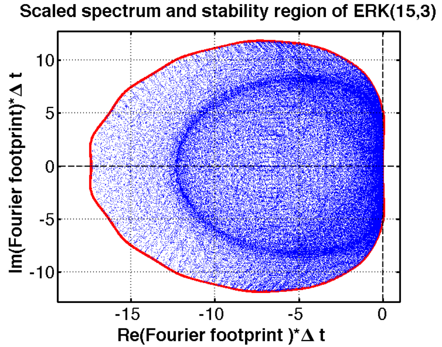
Optimizing for the SD spectrum

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



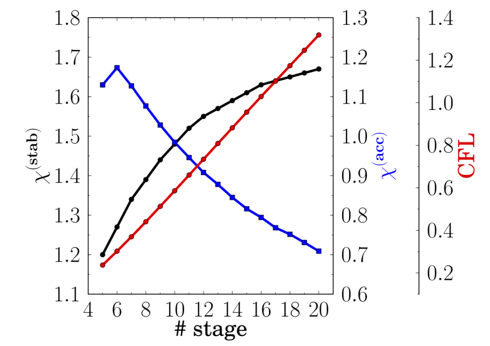
$$\frac{d\mathbf{W}_{i,j}}{dt} + \frac{a}{\Delta g}\left(\mathbf{T}^{0,0}\,\mathbf{W}_{i,j} + \mathbf{T}^{-1,0}\,\mathbf{W}_{i-1,j} + \mathbf{T}^{0,-1}\,\mathbf{W}_{i,j-1}\right.$$
$$\left. + \mathbf{T}^{+1,0}\,\mathbf{W}_{i+1,j} + \mathbf{T}^{0,+1}\,\mathbf{W}_{i,j+1}\right) = 0$$

Optimizing for the SD spectrum



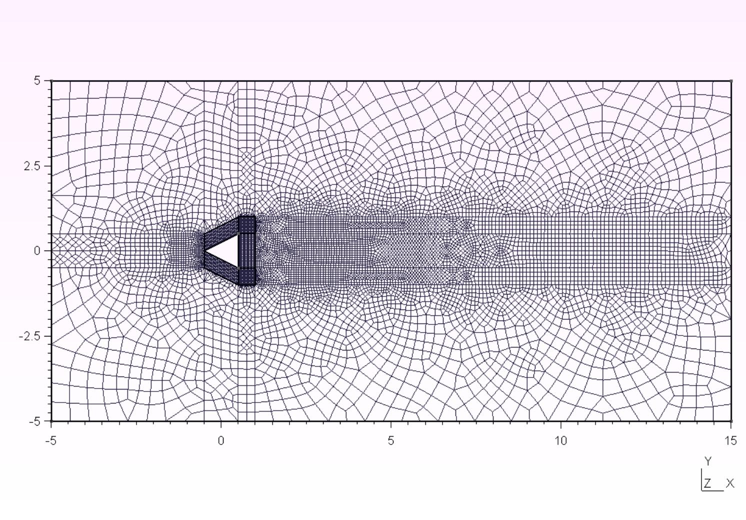
- Blue: eigenvalues; Red: RK stability boundary
- The convex hull of the generated spectrum is used as a proxy to accelerate the optimization process

Optimizing for the SD spectrum



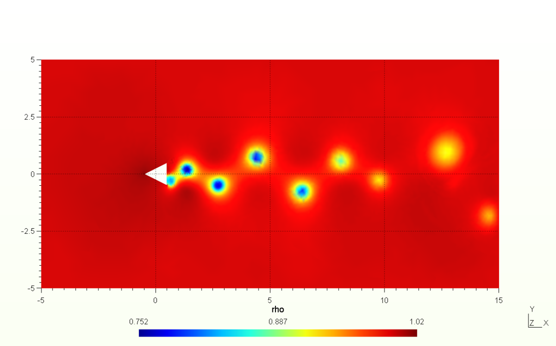
- Primarily optimized for stable step size
- Secondary optimization for nonlinear accuracy and low-storage (3 memory locations per unknown)

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)