

RK-Opt: A package for the design of numerical ODE solvers

David I. Ketcheson^{*1}, Matteo Parsani¹, Zachary J. Grant², Aron J. Ahmadi³, and Hendrik Ranocha¹

¹ King Abdullah University of Science and Technology, Saudi Arabia ² Oak Ridge National Laboratory, USA ³ Capital One, USA

DOI: [10.21105/joss.02514](https://doi.org/10.21105/joss.02514)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Patrick Diehl](#) ↗

Reviewers:

- [@gardner48](#)
- [@debdeepbh](#)
- [@emconsta](#)

Submitted: 21 July 2020

Published: 30 October 2020

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Ordinary and partial differential equations (ODEs and PDEs) are used to model many important phenomena. In most cases, solutions of these models must be approximated by numerical methods. Most of the relevant algorithms fall within a few classes of methods, with the properties of individual methods determined by their coefficients. The choice of appropriate coefficients in the design of methods for specific applications is an important area of research. RK-Opt is a software package for designing numerical ODE solvers with coefficients optimally chosen to provide desired properties. It is available from <https://github.com/ketch/RK-Opt>, with documentation at <http://numerics.kaust.edu.sa/RK-Opt/>. The primary focus of the package is on the design of Runge-Kutta methods, but some routines for designing other classes of methods such as multistep Runge-Kutta and general linear methods are also included.

Statement of need

Over the last several decades, a great deal of work has gone into the design of numerical ODE solvers. Initially this work was aimed at developing general purpose solvers, but over time the emphasis shifted increasingly toward development of optimized methods for specific applications. Different accuracy, stability, performance, and other properties may be relevant or essential depending on the nature of the equations to be solved.

An s -stage Runge-Kutta method has roughly s^2 coefficients (roughly $s^2/2$ for explicit methods), which can be chosen so as to provide high accuracy, stability, or other properties. Historically, most interest in Runge-Kutta methods has focused on methods using the minimum number of stages for a given order of accuracy. However, in the past few decades there has been increasing recognition that using extra stages can be worthwhile in order to improve other method properties. Some areas where this is particularly useful are in the enhancement of linear and nonlinear stability properties, the reduction of storage requirements, and the design of embedded pairs. Methods with dozens or even hundreds of stages are not unheard of.

At the same time, most existing Runge-Kutta methods have been designed by hand, by researchers laboriously solving the order conditions. When using extra stages, the number of available parameters makes the selection of a near-optimal choice by hand impossible, and one resorts to computational optimization. This leads to a different paradigm of numerical method design, in which we use sophisticated numerical (optimization) algorithms to design

*Corresponding author.

sophisticated numerical (integration) algorithms. It can be expected that this trend will accelerate in the future, and perhaps one day simple manually-constructed algorithms will be the exception.

RK-Opt contains a set of tools for designing Runge-Kutta methods in this paradigm. It provides code that can enforce desired properties and/or objective functions. The constraints and objective are then used within an optimization framework, to determine coefficients of methods that best achieve the desired goal. Thus, RK-Opt is a sort of meta-software, consisting of algorithms whose purpose is to create other algorithms.

Typically, the most obvious formulation of the corresponding optimization problem is intractable. Therefore, these problems are reformulated in ways that make them amenable to available techniques. These reformulations include, for instance, turning a nonconvex problem into a sequence of convex problems or even linear programs. The resulting algorithms can often guarantee optimality of their output. However, for the general problem of determining Runge-Kutta coefficients, the nonconvex problem must be attacked directly and optimality cannot be guaranteed.

RK-Opt is written entirely in MATLAB, and leverages the MATLAB Optimization Toolbox as well as the Global Optimization Toolbox. Its development has been motivated largely by research needs and it has been used in a number of papers (see below).

Features

RK-Opt includes the following subpackages.

`polyopt`

This package computes optimal stability functions for Runge-Kutta methods. Here *optimal* means that the stable step size is maximized for a given ODE spectrum. The corresponding optimization problem is intractable under a direct implementation. The package uses the algorithm developed in (Ketcheson & Ahmadi, 2012), which relaxes the global optimization problem by solving a sequence of convex subproblems. Under certain technical assumptions, the result is guaranteed to be the optimal solution of the original problem. `polyopt` relies on CVX (Grant & Boyd, 2008, 2014) to solve the convex subproblems. This package is usually used as the first step in designing a Runge-Kutta method.

`RK-Coeff-Opt`

This package computes optimal Runge-Kutta coefficients based on a desired set of constraints and an objective. Available constraints include:

- The number of stages and order of accuracy
- The class of method (explicit, implicit, diagonally implicit, low-storage)
- The coefficients of the stability polynomial (usually determined using `polyopt`)

Two objective functions are provided; methods can be optimized for the strong stability preserving (SSP) coefficient or the principal error norm (a measure of the leading-order truncation error coefficients). In addition to standard Runge-Kutta methods, various classes of multistep Runge-Kutta methods can also be optimized.

The optimization problem in question is highly nonconvex and the available solvers may fail to find a solution, or may converge to a non-optimal solution. For this reason, the implementation

is based on solving many local optimization problems in parallel from different random initial points, using MATLAB's Global Optimization Toolbox.

The packages `dwrk-opt` and `low-storage` are specialized but less full-featured versions of `RK-Coeff-Opt` that were developed for specific research projects involving downwind Runge-Kutta methods and low-storage Runge-Kutta methods, respectively.

`am_radius-opt`

Whereas the previous two subpackages are fairly general-purpose tools, this package solves a very specific and discrete set of problems described in (Ketcheson, 2009). Specifically, the provided routines determine the coefficients of multistep methods (including classes of general linear methods) with the largest possible SSP coefficient (also known as radius of absolute monotonicity). The corresponding optimization problem had previously been attacked using brute force search, but this limited its solvability to methods with very few steps. In this package the problem is reformulated as a sequence of linear programming problems, enabling its efficient solution for methods with many steps.

Related research and software

`RK-Opt` development has proceeded in close connection to the `NodePy` package (<https://github.com/ketch/NodePy>). Whereas `RK-Opt` is focused on the design of numerical methods, `NodePy` is focused more on their analysis. A common workflow involves generating new methods with `RK-Opt` and then studying their properties in more detail using `NodePy`.

Some of the research projects that have made use of `RK-Opt` include development of:

- SSP Runge-Kutta methods (Gottlieb, Grant, & Higgs, 2015; Higuera & Roldán, 2019a; Ketcheson, 2008; Ketcheson et al., 2009)
- SSP linear multistep methods (Ketcheson, 2009)
- SSP general linear methods (Bresten et al., 2017; Ketcheson et al., 2011)
- SSP IMEX Runge-Kutta methods (Conde, Gottlieb, Grant, & Shadid, 2017)
- Low-storage Runge-Kutta methods (Higuera & Roldán, 2019b; Ketcheson, 2010)
- Optimal Runge-Kutta stability polynomials (Ketcheson & Ahmadi, 2012)
- Additive and downwind SSP Runge-Kutta methods (Higuera, Ketcheson, & Kocsis, 2018; Ketcheson, 2011)
- Optimal Runge-Kutta methods for specific PDE semi-discretizations (Kubatko, Yeager, & Ketcheson, 2014; Parsani & Ketcheson, 2012; Parsani et al., 2012; Parsani, Ketcheson, & Deconinck, 2013)
- Optimal Runge-Kutta methods for pseudo-time stepping (Vermeire, Loppi, & Vincent, 2019, 2020)
- Embedded pairs for Runge-Kutta methods (Conde, Fekete, & Shadid, 2018)
- Runge-Kutta methods with high weak stage order (Ketcheson, Seibold, Shirokoff, & Zhou, 2018)
- SSP multistage, multiderivative methods (Christlieb, Gottlieb, Grant, & Seal, 2016; Grant, Gottlieb, & Seal, 2019; Reynoso, Gottlieb, & Grant, 2017)

As can be seen from this list, applications have mostly stemmed from the work of the main developer's research group, but have since expanded beyond that.

Because of the nature of `RK-Opt`, applications often involve writing some additional code to impose special constraints, or simply using the existing code as a template. A number of related optimization routines written for similar purposes in this vein can be found at <https://github.com/SSPmethods>.

Acknowledgements

Much of the initial RK-Opt development was performed by D. Ketcheson while he was supported by a DOE Computational Science Graduate Fellowship and by AFOSR grant number FA9550-06-1-0255. Development has also been supported by funding from King Abdullah University of Science and Technology.

References

- Bresten, C., Gottlieb, S., Grant, Z., Higgs, D., Ketcheson, D. I., & Németh, A. (2017). Strong stability preserving multistep Runge-Kutta methods. *Mathematics of Computation*, *86*, 747–769. doi:[10.1090/mcom/3115](https://doi.org/10.1090/mcom/3115)
- Christlieb, A. J., Gottlieb, S., Grant, Z., & Seal, D. C. (2016). Explicit strong stability preserving multistage two-derivative time-stepping schemes. *Journal of Scientific Computing*, *68*(3), 914–942. doi:[10.1007/s10915-016-0195-8](https://doi.org/10.1007/s10915-016-0195-8)
- Conde, S., Fekete, I., & Shadid, J. N. (2018). Embedded error estimation and adaptive step-size control for optimal explicit strong stability preserving Runge-Kutta methods. *arXiv preprint arXiv:1806.08693*.
- Conde, S., Gottlieb, S., Grant, Z. J., & Shadid, J. N. (2017). Implicit and implicit-explicit strong stability preserving Runge-Kutta methods with high linear order. *Journal of Scientific Computing*, *73*(2-3), 667–690. doi:[10.1007/s10915-017-0560-2](https://doi.org/10.1007/s10915-017-0560-2)
- Gottlieb, S., Grant, Z., & Higgs, D. (2015). Optimal explicit strong stability preserving Runge-Kutta methods with high linear order and optimal nonlinear order. *Mathematics of Computation*, *84*(296), 2743–2761. doi:[10.1007/s10915-016-0195-8](https://doi.org/10.1007/s10915-016-0195-8)
- Grant, M., & Boyd, S. (2008). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, & H. Kimura (Eds.), *Recent advances in learning and control*, Lecture notes in control and information sciences (pp. 95–110). Springer-Verlag Limited. doi:[10.1007/978-1-84800-155-8_7](https://doi.org/10.1007/978-1-84800-155-8_7)
- Grant, M., & Boyd, S. (2014, March). CVX: Matlab software for disciplined convex programming, version 2.1. <http://cvxr.com/cvx>.
- Grant, Z., Gottlieb, S., & Seal, D. C. (2019). A strong stability preserving analysis for explicit multistage two-derivative time-stepping schemes based on Taylor series conditions. *Communications on Applied Mathematics and Computation*, *1*(1), 21–59. doi:[10.1007/s10915-016-0195-8](https://doi.org/10.1007/s10915-016-0195-8)
- Higuera, I., Ketcheson, D. I., & Kocsis, T. A. (2018). Optimal monotonicity-preserving perturbations of a given Runge-Kutta method. *Journal of Scientific Computing*, *76*(3), 1337–1369. doi:[10.1007/s10915-018-0664-3](https://doi.org/10.1007/s10915-018-0664-3)
- Higuera, I., & Roldán, T. (2019a). Strong stability preserving properties of composition Runge-Kutta schemes. *J. Sci. Comput.*, *80*(2), 784–807. doi:[10.1007/s10915-019-00916-3](https://doi.org/10.1007/s10915-019-00916-3)
- Higuera, I., & Roldán, T. (2019b). New third order low-storage SSP explicit Runge-Kutta methods. *Journal of Scientific Computing*, *79*(3), 1882–1906.
- Ketcheson, D. I. (2008). Highly efficient strong stability preserving Runge-Kutta methods with low-storage implementations. *SIAM Journal on Scientific Computing*, *30*(4), 2113–2136. doi:[10.1137/07070485X](https://doi.org/10.1137/07070485X)

- Ketcheson, D. I. (2009). Computation of optimal monotonicity preserving general linear methods. *Mathematics of Computation*, 78(267), 1497–1513. doi:[10.1090/S0025-5718-09-02209-1](https://doi.org/10.1090/S0025-5718-09-02209-1)
- Ketcheson, D. I. (2010). Runge–Kutta methods with minimum storage implementations. *Journal of Computational Physics*, 229(5), 1763–1773. doi:[10.1016/j.jcp.2009.11.006](https://doi.org/10.1016/j.jcp.2009.11.006)
- Ketcheson, D. I. (2011). Step sizes for strong stability preservation with downwind-biased operators. *SIAM Journal on Numerical Analysis*, 49(4), 1649. doi:[10.1137/100818674](https://doi.org/10.1137/100818674)
- Ketcheson, D. I., & Ahmadi, A. J. (2012). Optimal stability polynomials for numerical integration of initial value problems. *Communications in Applied Mathematics and Computational Science*, 7(2), 247–271. doi:[10.2140/camcos.2012.7.247](https://doi.org/10.2140/camcos.2012.7.247)
- Ketcheson, D. I., Gottlieb, S., & Macdonald, C. B. (2011). Strong stability preserving two-step Runge–Kutta methods. *SIAM Journal on Numerical Analysis*, 49(6), 2618–2639. doi:[10.1137/10080960X](https://doi.org/10.1137/10080960X)
- Ketcheson, D. I., Macdonald, C. B., & Gottlieb, S. (2009). Optimal implicit strong stability preserving Runge–Kutta methods. *Applied Numerical Mathematics*, 59(2), 373–392. doi:[10.1016/j.apnum.2008.03.034](https://doi.org/10.1016/j.apnum.2008.03.034)
- Ketcheson, D. I., Seibold, B., Shirokoff, D., & Zhou, D. (2018). DIRK schemes with high weak stage order. In S. J. Sherwin, D. Moxey, J. Peiró, P. E. Vincent, & C. Schwab (Eds.), *Spectral and High Order Methods for Partial Differential Equations (ICOSAHOM) 2018*.
- Kubatko, E. J., Yeager, B. A., & Ketcheson, D. I. (2014). Optimal strong-stability-preserving Runge–Kutta time discretizations for discontinuous Galerkin methods. *Journal of Scientific Computing*, 60(2), 313–344. doi:[10.1007/s10915-013-9796-7](https://doi.org/10.1007/s10915-013-9796-7)
- Parsani, M., & Ketcheson, D. I. (2012). Optimized low-order explicit Runge–Kutta schemes for high-order spectral difference method. In *Proceedings of the 11th Finnish Mechanics Days* (pp. 1–5).
- Parsani, M., Ketcheson, D. I., & Deconinck, W. (2012). Explicit Runge–Kutta schemes for the spectral difference method applied to wave propagation problems. In *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering*.
- Parsani, M., Ketcheson, D. I., & Deconinck, W. (2013). Optimized explicit Runge–Kutta schemes for the spectral difference method applied to wave propagation problems. *SIAM Journal on Scientific Computing*, 35(2), A957–A986. doi:[10.1137/120885899](https://doi.org/10.1137/120885899)
- Reynoso, G. F., Gottlieb, S., & Grant, Z. J. (2017). Strong stability preserving sixth order two-derivative Runge–Kutta methods. In *AIP Conference Proceedings* (Vol. 1863, p. 560068). AIP Publishing LLC. doi:[10.1063/1.4992751](https://doi.org/10.1063/1.4992751)
- Vermeire, B. C., Loppi, N. A., & Vincent, P. E. (2019). Optimal Runge–Kutta schemes for pseudo time-stepping with high-order unstructured methods. *Journal of Computational Physics*, 383, 55–71. doi:[10.1016/j.jcp.2019.01.003](https://doi.org/10.1016/j.jcp.2019.01.003)
- Vermeire, B. C., Loppi, N. A., & Vincent, P. E. (2020). Optimal embedded pair Runge–Kutta schemes for pseudo-time stepping. *Journal of Computational Physics*, 109499. doi:[10.1016/j.jcp.2020.109499](https://doi.org/10.1016/j.jcp.2020.109499)