The Journal of Open Source Software

# SummationByPartsOperators.jl: A Julia library of provably stable discretization techniques with mimetic properties

**Hendrik Ranocha**[1]

**1** Applied Mathematics Münster, University of Münster, Germany

**DOI:** 10.21105/joss.03454

**Software**

- Review
- Repository
- Archive

**Editor:** Viviane Pons

**Reviewers:**

- @dawbarton
- @kellertuer

**Submitted:** 25 May 2021
**Published:** 25 August 2021



## Summary

SummationByPartsOperators.jl is a Julia library of summation-by-parts (SBP) operators, which are discrete derivative operators developed to get provably stable (semi-) discretizations, paying special attention to boundary conditions. Discretizations included in this framework are finite difference, Fourier pseudospectral, continuous Galerkin, and discontinuous Galerkin methods. The main aim of SummationByPartsOperators.jl is to be useful for both students learning the basic concepts and researchers developing new numerical algorithms based on SBP operators. Therefore, SummationByPartsOperators.jl provides a unified framework of all of these seemingly different discretizations. At the same time, the implementation is reasonably optimized to achieve good performance without sacrificing flexibility.

## Statement of need

Partial differential equations (PDEs) are widely used in science and engineering to create mathematical models of real-world processes. Since PDEs often need to be solved numerically, a vast amount of numerical methods has been developed. Since it is impossible to keep up with all recent research, sub-communities focussing on specific applications or numerical methods emerged. Sometimes, these communities develop different vocabulary and notations, making it hard for newcomers (or even experienced researchers) to see similarities and connections between seemingly unrelated approaches. To transfer new ideas and developments and knowledge from one community to another, common abstractions can be helpful. The concept of SBP operators is such an abstraction. In recent years, SBP operators have attracted a lot of attention, in particular for PDEs modeling advection-dominated problems, where they enabled the construction of energy- or entropy-stable numerical methods, including finite difference, discontinuous Galerkin, continuous Galerkin, and (pseudo-) spectral methods. Their success is based on mimetic properties which enable the transfer of results obtained for differential equations at the continuous level to the discrete level. In particular, SBP operators are designed to mimic integration-by-parts discretely as summation-by-parts, enabling discrete analogues of energy/entropy methods for PDEs.

SummationByPartsOperators.jl is written entirely in Julia (Bezanson et al., 2017). Making use of multiple dispatch and generic types, SummationByPartsOperators.jl provides a unified interface for different SBP operators. At the same time, the implementations are reasonably fast (again, due to multiple dispatch and specialized implementations for each operator class). Together, this facilitates the development of new algorithms and research in numerical analysis, which is the primary goal of this package. In addition, SummationByPartsOperators.jl has been used in a number of graduate-level numerical analysis courses, allowing students to

understand the connections between different SBP methods by presenting them in a unified framework. In addition, some of the operators were not available in open source software previously (to the best of the author's knowledge).

## Features

SummationByPartsOperators.jl implements numerical methods based on SBP operators of the following classes:

- finite difference methods
- Fourier collocation methods
- continuous Galerkin methods
- discontinuous Galerkin methods

Since a discrete derivative operator is a linear operator, all of these SBP operators implement the basic interface of such linear operators (`AbstractMatrix` in Julia) such as multiplication by vectors and addition of operators. Finite difference and Fourier operators on periodic domains also allow the construction of rational functions of operators and their efficient implementation using the fast Fourier transform (Frigo & Johnson, 2005).

In addition to basic SBP derivative operators, SummationByPartsOperators.jl contains a number of related operators, such as

- SBP artificial dissipation operators
- spectral viscosity operators for Fourier methods
- modal filter operators for Fourier methods and Legendre pseudospectral methods

Using Julia's rich type system, all of these operators are implemented as their own types. This enables several optimizations such as a memory requirement independent of the number of grid points. In contrast, implementations based on sparse/banded matrices have a memory requirement growing linearly with the number of grid points. In addition, the structure of the operators can be taken into account for operator-vector multiplications, usually resulting in speed-ups of an order of magnitude or more on consumer hardware. For example, the application an optimized the sixth-order (in the interior) finite difference SBP operator (Almquist, 2017) on a grid with 1000 nodes takes roughly 330 ns on a consumer CPU from 2017 (Intel® Core™ i7-8700K) using version v0.5.5 of SummationByPartsOperators.jl. In contrast, the same operation takes roughly 3.9 microseconds using a sparse matrix format used in other implementations of this operator (Almquist, 2017). This benchmark is based on the following code, which also provides a very basic example of SummationByPartsOperators.jl.

```
# install the package if necessary
using Pkg; Pkg.add("SummationByPartsOperators")

# load the package
using SummationByPartsOperators

# create a finite difference SBP operator
D = derivative_operator(MattssonAlmquistVanDerWeide2018Accurate(),
  derivative_order=1, accuracy_order=6,
  xmin=0.0, xmax=1.0, N=10^3
)
```

```
# evaluate the function `sinpi` on the discrete grid
x = grid(D); u = sinpi.(x)

# use `D` to approximate the derivative of `u`
du = D * u

# compute the discrete L² error of the approximation
integrate(u -> u^2, du - pi * cospi.(x), D) |> sqrt
```

The output of the last command will be a relatively small number on the order of `4.2e-13`.

Following good software development practices, SummationByPartsOperators.jl makes use of continuous integration and automated tests required before merging pull requests. Documentation is provided in form of docstrings, a general introduction, and tutorials. In addition, SummationByPartsOperators.jl is a registered Julia package and can be installed using the built-in package manager, handling dependencies and version requirements automatically.

## Related research and software

There are of course many open-source software packages providing discretizations of differential equations. However, many of them focus on a single class of numerical methods or a specific application, e.g.,

- finite difference methods (DiffEqOperators.jl, a part of DifferentialEquations.jl (Rackauckas & Nie, 2017))
- finite volume methods (Oceananigans.jl (Ramadhan et al., 2020), Kinetic.jl (Xiao, 2021))
- spectral methods (ApproxFun.jl (Olver & Townsend, 2014), FourierFlows.jl (Constantinou & Wagner, 2021))
- finite element methods (Gridap.jl (Badia & Verdugo, 2020))
- discontinuous spectral element methods (Trixi.jl (Ranocha, Schlottke-Lakemper, et al., 2021; Schlottke-Lakemper et al., 2021, 2020))

We are not aware of any open-source software library implementing all of the SBP classes using a unified interface or even several finite difference SBP operators on finite domains, which are usually heavily optimized (Mattsson et al., 2014, 2018) and not available in other open source packages. Sometimes, restricted sets of coefficients are available online (Almquist, 2017; O'Reilly, 2019), but there is no other extensive collection of these methods.

Of course, there is a plethora of additional open source software implementing numerical methods for PDEs and each package has its own design criteria and goals. SummationByPartsOperators.jl provides a unified interface of different SBP operators. Thus, there is a partial overlap with some of the packages mentioned above such as finite difference operators on periodic domains (DiffEqOperators.jl, but with a different handling of bounded domains) or Fourier methods on periodic domains (ApproxFun.jl, but with a different interface and extensions). In addition, many packages focus on a specific application such as some specific fluid models (Oceananigans.jl, FourierFlows.jl) or hyperbolic PDEs (Trixi.jl). In contrast, SummationByPartsOperators.jl focuses on the numerical methods and provides them in a form usable for rather general PDEs. For example, there is ongoing work to use the basic operators provided by SummationByPartsOperators.jl in Trixi.jl.

Some of the research projects that have made use of SummationByPartsOperators.jl (most of which have led to its further development) include numerical analysis of and algorithms for

- nonlinear dispersive wave equations (Ranocha, Mitsotakis, et al., 2021; Ranocha, Quezada de Luna, et al., 2021)
- hyperbolic conservation laws (LeFloch & Ranocha, 2021; Öffner & Ranocha, 2019; Ranocha & Gassner, 2021)
- ordinary differential equations (Ranocha, 2021; Ranocha & Ketcheson, 2020; Ranocha & Nordström, 2021)
- Helmholtz Hodge decomposition and analysis of plasma waves (Ranocha et al., 2020)

## Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grant SO~363/14-1 and Germany's Excellence Strategy EXC 2044-390685587, Mathematics Münster: Dynamics-Geometry-Structure as well as King Abdullah University of Science and Technology (KAUST).

## References

Almquist, M. (2017). *Optimized SBP operators*. https://bitbucket.org/martinalmquist/optimized_sbp_operators.

Badia, S., & Verdugo, F. (2020). Gridap: An extensible finite element toolbox in Julia. *Journal of Open Source Software*, *5*(52), 2520. https://doi.org/10.21105/joss.02520

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Constantinou, N. C., & Wagner, G. L. (2021). *FourierFlows/FourierFlows.jl: FourierFlows v0.6.18 (version v0.6.18)*. https://github.com/FourierFlows/FourierFlows.jl. https://doi.org/10.5281/zenodo.1161724

Frigo, M., & Johnson, S. G. (2005). The design and implementation of FFTW3. *Proceedings of the IEEE*, *93*(2), 216–231. https://doi.org/10.1109/JPROC.2004.840301

LeFloch, P. G., & Ranocha, H. (2021). Kinetic functions for nonclassical shocks, entropy stability, and discrete summation by parts. *Journal of Scientific Computing*, *87*. https://doi.org/10.1007/s10915-021-01463-6

Mattsson, K., Almquist, M., & Carpenter, M. H. (2014). Optimal diagonal-norm SBP operators. *Journal of Computational Physics*, *264*, 91–111. https://doi.org/10.1016/j.jcp.2013.12.041

Mattsson, K., Almquist, M., & Weide, E. van der. (2018). Boundary optimized diagonal-norm SBP operators. *Journal of Computational Physics*, *374*, 1261–1266. https://doi.org/10.1016/j.jcp.2018.06.010

O'Reilly, O. (2019). *Sbp*. https://github.com/ooreilly/sbp.

Olver, S., & Townsend, A. (2014). A practical framework for infinite-dimensional linear algebra. *2014 First Workshop for High Performance Technical Computing in Dynamic Languages*, 57–62. https://doi.org/10.1109/HPTCDL.2014.10

Öffner, P., & Ranocha, H. (2019). Error boundedness of discontinuous Galerkin methods with variable coefficients. *Journal of Scientific Computing*, *79*(3), 1572–1607. https://doi.org/10.1007/s10915-018-00902-1

Rackauckas, C., & Nie, Q. (2017). DifferentialEquations.jl – A performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, *5*(1), 15. https://doi.org/10.5334/jors.151

Ramadhan, A., Wagner, G. L., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza, A., Edelman, A., Ferrari, R., & Marshall, J. (2020). Oceananigans.jl: Fast and friendly geophysical fluid dynamics on GPUs. *Journal of Open Source Software*, *5*(53), 2018. https://doi.org/10.21105/joss.02018

Ranocha, H. (2021). On strong stability of explicit Runge-Kutta methods for nonlinear semibounded operators. *IMA Journal of Numerical Analysis*, *41*(1), 654–682. https://doi.org/10.1093/imanum/drz070

Ranocha, H., & Gassner, G. J. (2021). Preventing pressure oscillations does not fix local linear stability issues of entropy-based split-form high-order schemes. *Communications on Applied Mathematics and Computation*. https://doi.org/10.1007/s42967-021-00148-z

Ranocha, H., & Ketcheson, D. I. (2020). Energy stability of explicit Runge-Kutta methods for nonautonomous or nonlinear problems. *SIAM Journal on Numerical Analysis*, *58*(6), 3382–3405. https://doi.org/10.1137/19M1290346

Ranocha, H., Mitsotakis, D., & Ketcheson, D. I. (2021). A broad class of conservative numerical methods for dispersive wave equations. *Communications in Computational Physics*, *29*(4), 979–1029. https://doi.org/10.4208/cicp.OA-2020-0119

Ranocha, H., & Nordström, J. (2021). A new class of $A$ stable summation by parts time integration schemes with strong initial conditions. *Journal of Scientific Computing*, *87*. https://doi.org/10.1007/s10915-021-01454-7

Ranocha, H., Ostaszewski, K., & Heinisch, P. (2020). Discrete vector calculus and Helmholtz Hodge decomposition for classical finite difference summation by parts operators. *Communications on Applied Mathematics and Computation*, *2*, 581–611. https://doi.org/10.1007/s42967-019-00057-2

Ranocha, H., Quezada de Luna, M., & Ketcheson, D. I. (2021, February). *On the rate of error growth in time for numerical solutions of nonlinear dispersive wave equations*. http://arxiv.org/abs/2102.07376

Ranocha, H., Schlottke-Lakemper, M., Winters, A. R., Faulhaber, E., Chan, J., & Gassner, G. (2021, August). *Adaptive numerical simulations with Trixi.jl: A case study of Julia for scientific computing*. http://arxiv.org/abs/2108.06476

Schlottke-Lakemper, M., Gassner, G. J., Ranocha, H., & Winters, A. R. (2020). *Trixi.jl: A tree-based numerical simulation framework for hyperbolic PDEs written in Julia*. https://github.com/trixi-framework/Trixi.jl. https://doi.org/10.5281/zenodo.3996439

Schlottke-Lakemper, M., Winters, A. R., Ranocha, H., & Gassner, G. J. (2021). A purely hyperbolic discontinuous Galerkin approach for self-gravitating gas dynamics. *Journal of Computational Physics*, 110467. https://doi.org/10.1016/j.jcp.2021.110467

Xiao, T. (2021). Kinetic.jl: A portable finite volume toolbox for scientific and neural computing. *Journal of Open Source Software*, *6*(62), 3060. https://doi.org/10.21105/joss.03060