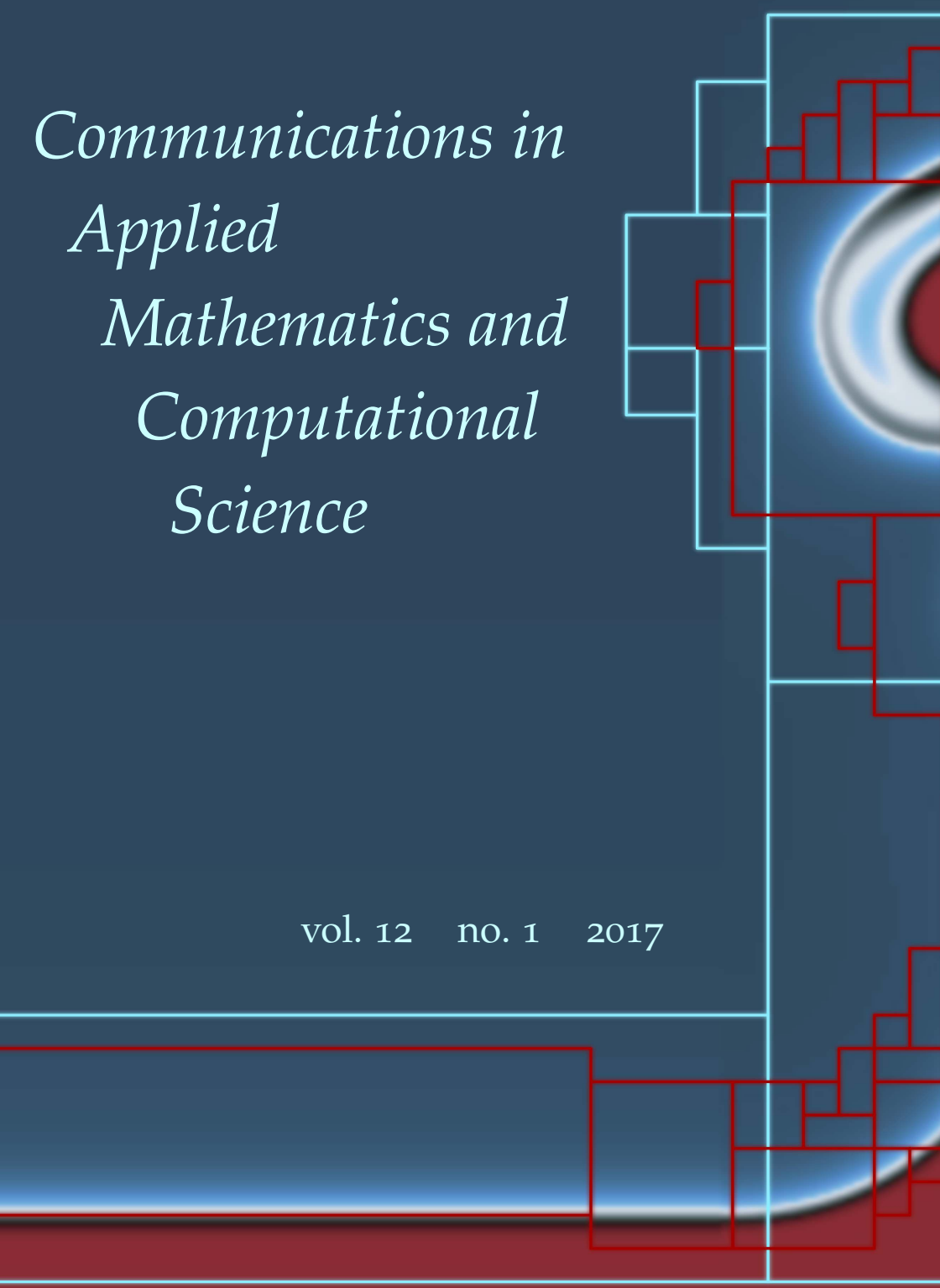
Communications in Applied Mathematics and Computational Science

vol. 12 no. 1 2017



msp

Communications in Applied Mathematics and Computational Science

msp.org/camcos

See inside back cover or msp.org/camcos for submission instructions.

The subscription price for 2017 is US $100/year for the electronic version, and $150/year (+$15, if shipping outside the US) for print and electronic. Subscriptions, requests for back issues from the last three years and changes of subscriber address should be sent to MSP.

Communications in Applied Mathematics and Computational Science (ISSN 2157-5452 electronic, 1559-3940 printed) at Mathematical Sciences Publishers, 798 Evans Hall #3840, c/o University of California, Berkeley, CA 94720-3840, is published continuously online. Periodical rate postage paid at Berkeley, CA 94704, and additional mailing offices.

CAMCoS peer review and production are managed by EditFLOW® from MSP.

PUBLISHED BY

mathematical sciences publishers

nonprofit scientific publishing

http://msp.org/

COMM. APP. MATH. AND COMP. SCI.
Vol. 12, No. 1, 2017
dx.doi.org/10.2140/camcos.2017.12.1

A SINGLE-STAGE FLUX-CORRECTED TRANSPORT ALGORITHM FOR HIGH-ORDER FINITE-VOLUME METHODS

CHRISTOPHER CHAPLIN AND PHILLIP COLELLA

We present a new limiter method for solving the advection equation using a high-order, finite-volume discretization. The limiter is based on the flux-corrected transport algorithm. We modify the classical algorithm by introducing a new computation for solution bounds at smooth extrema, as well as improving the preconstraint on the high-order fluxes. We compute the high-order fluxes via a method-of-lines approach with fourth-order Runge–Kutta as the time integrator. For computing low-order fluxes, we select the corner-transport upwind method due to its improved stability over donor-cell upwind. Several spatial differencing schemes are investigated for the high-order flux computation, including centered-difference and upwind schemes. We show that the upwind schemes perform well on account of the dissipation of high-wavenumber components. The new limiter method retains high-order accuracy for smooth solutions and accurately captures fronts in discontinuous solutions. Further, we need only apply the limiter once per complete time step.

1. Introduction

We wish to solve hyperbolic conservation laws of the form

$$\frac{\partial U}{\partial t} + \nabla \cdot (\vec{\boldsymbol{F}}(U)) = 0, \tag{1}$$

where U represents a vector of conserved values and $\vec{\boldsymbol{F}} = (\boldsymbol{F}^1 \ \cdots \ \boldsymbol{F}^D)$ the corresponding D-dimensional fluxes. The discrete solution of these equations at a given time t^{n+1} and spatial location $\boldsymbol{i}$ is given by

$$\langle U \rangle_{\boldsymbol{i}}^{n+1} = \langle U \rangle_{\boldsymbol{i}}^{n} - \frac{\Delta t}{h} \sum_{d=1}^{D} \left[(\boldsymbol{F}^d)_{\boldsymbol{i}+\boldsymbol{e}^d/2}^{n+1/2} - (\boldsymbol{F}^d)_{\boldsymbol{i}-\boldsymbol{e}^d/2}^{n+1/2} \right], \tag{2}$$

where $\langle U \rangle_{\boldsymbol{i}}^{n}$ approximates the average of U over a rectangular Cartesian control volume at time t^n and $(\boldsymbol{F}^d)^{n+1/2}$ approximates the average of $\vec{\boldsymbol{F}}(U)$ from time t^n to t^{n+1} over the faces of the same control volume. The parameters Δt and h represent

MSC2010: 65M08.
Keywords: finite-volume method, high order, advection, limiter.

the time-step size and grid spacing, respectively. Methods for accurately computing the fluxes, $(\boldsymbol{F}^d)^{n+1/2}$, to obtain high-order accuracy for smooth solutions are well understood. However, these high-order methods must be modified to selectively introduce dissipation in the presence of discontinuities or underresolved gradients. These modification methods are called limiter schemes. Modern limiter schemes seek to achieve high-order accuracy for smooth solutions regardless of complexity and to represent discontinuities well.

Many of the original second-order limiter schemes, such as monotonic upstream-centered schemes for conservation laws (MUSCL) [25], total variation diminishing (TVD) [11], piecewise parabolic method (PPM) [7], and flux-corrected transport (FCT) [2], are still used in some form today. But these original schemes struggled to achieve all of the aforementioned goals, particularly high-order accuracy for solutions that are both complicated and smooth. The standard and weighted essentially nonoscillatory (ENO/WENO) schemes were developed to address these issues for TVD [12; 17; 22]. An extremum-preserving limiter has been added to PPM [6]. For FCT, a nonclipping limiter was developed [28]. Finite-element methods, in particular discontinuous Galerkin (DG) methods, have also been gaining popularity for these problems. DG methods use a menagerie of technologies to handle discontinuities [3]. Recent efforts have been made to combine aspects of WENO limiting and the DG discretization [22]. FCT was also extended to finite-element discretizations [18; 19], and many of the recent improvements to the algorithm have been on these discretizations [16; 13; 15; 14]. Most of the limiter methods mentioned so far use a semidiscrete, method-of-lines formulation, wherein a standard ordinary differential equation (ODE) integrator is used to advance the solution after the spatial differencing scheme has been applied. There are fully discrete methods as well, such as arbitrary derivative in space and time (ADER) [24; 23]. They offer an alternative to the method-of-lines approaches [1; 9].

The starting point for our approach is a method-of-lines, finite-volume formulation. In all semidiscrete implementations mentioned above, the limiter algorithm is applied every time a high-order flux evaluation occurs. This requires the limiter to be applied several times during a single time-update procedure. For this study we chose to only apply limiting once per time update, after the total high-order flux was generated. This choice is motivated by both current and future performance considerations [10; 26]. Moving limiting to a postprocessing procedure allows for smaller stencils and less parallel communication. Limiters that are designed to preserve high-order accuracy at smooth extrema typically make use of second- or higher-order spatial derivative information to determine where the solution is smooth enough to not require limiting. Computing these derivatives at each stage in high-order ODE integration scheme requires a larger stencil than the standard high-order flux stencil, at least up until ninth order. The other half of this is that,

typically because the limiter procedure is complicated and has a wide stencil, ghost cells are synchronized at each stage computation. If one can fit all the data required to complete the entire time update in local memory, then synchronization must only occur once per time advance. These two hindrances, namely wider stencils and repeated synchronization barriers, motivated the departure from the stage-by-stage limiting approach.

We elected to use a version of flux-corrected transport (FCT) for our limiting scheme [2; 28; 15]. FCT introduces dissipation through a nonlinear hybridization of a high-order flux with a dissipative, low-order flux. To compute the high-order flux, we used a method-of-lines approach with fourth-order Runge–Kutta (RK4) as the time-integration scheme. For the spatial derivatives we looked at a family of methods based on high-order centered- and one-point-upwinded linear finite-volume interpolations. The low-order scheme was the corner-transport upwind (CTU) method [4; 21]. We elected to use CTU for two reasons: CTU permits the use of larger time steps than the standard donor-cell method, and CTU can be constructed in such a way as to preserve positivity in the solution. In addition to using these schemes for the high- and low-order fluxes, we modified the FCT algorithm in three important ways. First we included an extremum-preserving bound computation based on the approach used in [6; 20] for interpolation-based limiting. We also designed a more restrictive condition on applying the typical preconstraint for the high-order fluxes. Furthermore, we extended the product rule to sixth-order accuracy. All of these features were required in order to maintain high-order accuracy for complicated smooth solutions.

For this study, we restricted our attention to the scalar advection equation. This allowed us to explore design space for this novel single-stage limiter in a simple setting, but one that is still relatively unforgiving. These advective terms appear directly in real applications including transport of scalars in the atmosphere, Vlasov equations in phase space, and combustion.

Advection equation. We will consider the linear advection equation in the form

$$\frac{\partial q}{\partial t} + \nabla \cdot (q\vec{\boldsymbol{u}}) = 0, \tag{3}$$

$$\nabla \cdot \vec{\boldsymbol{u}} = 0, \tag{4}$$

on a D-dimensional square domain $\Omega = [0, 1]^D$. In this case $\vec{\boldsymbol{u}}$ is an advective velocity and q is a scalar field. The partial differential equation above can also be written as

$$\frac{dq}{dt} = 0, \qquad \frac{d\vec{\boldsymbol{x}}}{dt} = \vec{\boldsymbol{u}}. \tag{5}$$

Provided that an initial condition is specified ($q_0 = q(\vec{\boldsymbol{x}}(t_0), t_0)$), this system of ordinary differential equations yields a unique solution for any $q(\vec{\boldsymbol{x}}(t), t)$ and $\vec{\boldsymbol{x}}(t)$.

The solution arrived at by integrating the equations is that q is constant along characteristic curves defined by $\vec{\boldsymbol{x}}(t)$. Even though there is a simple solution to this equation, the analysis is still quite useful since there is no diffusion or entropy condition built into the equation: any numerical errors introduced are propagated through the domain.

Finite-volume discretization. Our approach is to use a finite-volume method to discretize the physical domain into a union of control volumes

$$V_{\boldsymbol{i}} = \left[\left(\boldsymbol{i} - \tfrac{1}{2}\right)h, \left(\boldsymbol{i} + \tfrac{1}{2}\right)h\right], \quad \boldsymbol{i} \in \mathbb{Z}^D, \tag{6}$$

where h is the grid spacing and $\boldsymbol{i}$ is a D-dimensional index denoting location. The origin in the physical domain occurs at the point $(\boldsymbol{i} - \frac{1}{2})h$ when $\boldsymbol{i} = \boldsymbol{0}$.

Values of the conserved scalar quantity q are stored as cell averages $\langle q \rangle$ over each $V_{\boldsymbol{i}}$, and the fluxes $\boldsymbol{F}^d = q\boldsymbol{u}^d$ are stored as averages $\langle \boldsymbol{F}^d \rangle_{\boldsymbol{i} \pm \boldsymbol{e}^d/2}$ over the surface faces $A_d^{\pm}$ of each cell:

$$\langle q \rangle_{\boldsymbol{i}}(t) = \frac{1}{h^D} \int_{V_{\boldsymbol{i}}} q(\boldsymbol{x}, t)\, d\boldsymbol{x}, \tag{7}$$

$$\langle \boldsymbol{F}^d \rangle_{\boldsymbol{i} \pm \boldsymbol{e}^d/2}(t) = \frac{1}{h^{D-1}} \int_{A_d^{\pm}} \boldsymbol{F}^d(\boldsymbol{x}, t)\, d\boldsymbol{x}. \tag{8}$$

Applying the finite-volume discretization (6) to (3) yields a semidiscrete system of ordinary differential equations (ODEs) in time

$$\frac{d\langle q \rangle_{\boldsymbol{i}}}{dt} = -\frac{1}{h^D} \int_{V_{\boldsymbol{i}}} (\nabla \cdot (\vec{\boldsymbol{F}}))\, d\boldsymbol{x}. \tag{9}$$

The divergence theorem is then applied to (9):

$$\frac{d\langle q \rangle_{\boldsymbol{i}}}{dt} = -D \cdot \langle \vec{\boldsymbol{F}} \rangle(t), \tag{10}$$

$$= -\frac{1}{h} \sum_{d=1}^{D} \left[\langle \boldsymbol{F}^d \rangle_{\boldsymbol{i} + \boldsymbol{e}^d/2} - \langle \boldsymbol{F}^d \rangle_{\boldsymbol{i} - \boldsymbol{e}^d/2} \right]. \tag{11}$$

The integration of the above system with respect to time from t^n to t^{n+1} produces the solution

$$\langle q \rangle_{\boldsymbol{i}}^{n+1} = \langle q \rangle_{\boldsymbol{i}}^{n} - \frac{\Delta t}{h} \sum_{d=1}^{D} \left[\langle \boldsymbol{F}^d \rangle_{\boldsymbol{i} + \boldsymbol{e}^d/2}^{n+1/2} - \langle \boldsymbol{F}^d \rangle_{\boldsymbol{i} - \boldsymbol{e}^d/2}^{n+1/2} \right], \tag{12}$$

$$\langle \boldsymbol{F}^d \rangle_{\boldsymbol{i} \pm \boldsymbol{e}^d/2}^{n+1/2} = \frac{1}{\Delta t} \int_{t^n}^{t^n + \Delta t} \langle \boldsymbol{F}^d \rangle_{\boldsymbol{i} \pm \boldsymbol{e}^d/2}(t)\, dt. \tag{13}$$

The resulting challenge is to accurately compute $\langle \boldsymbol{F}^d \rangle_{\boldsymbol{i} \pm \boldsymbol{e}^d/2}^{n+1/2}$. It is important to note that no approximations have been made at this point: (12) and (13) are exact

relationships. However, to obtain a full discrete approximation, we need quadrature rules for the surface fluxes in (11) and for the time-averaged fluxes in (13). The quadrature rules for computing these fluxes are defined following ideas from [20]. In that work, the high-order quadratures were computed using a method-of-lines approach. The surface fluxes were computed using a high-order centered-difference method, and the temporal integration was computed using the classic RK4 method. We retained the use of RK4 in this study and investigated several high-order methods for computing the surface fluxes.

Hybridization. Returning to the flux description in (13), we may now define the hybridization

$$\langle \boldsymbol{F}^d \rangle^{n+1/2}_{\boldsymbol{i}+\boldsymbol{e}^d/2} = (\eta_{\boldsymbol{i}+\boldsymbol{e}^d/2})\langle \boldsymbol{F}^d_H \rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} + (1-\eta_{\boldsymbol{i}+\boldsymbol{e}^d/2})\langle \boldsymbol{F}^d_L \rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}, \tag{14}$$

where the subscripts H and L refer to the high-order and low-order fluxes and $\eta_{\boldsymbol{i}+\boldsymbol{e}^d/2}$ is the hybridization coefficient.

In the following sections of the paper we will describe the design choices and procedures for computing the high-order flux, the low-order flux, and the hybridization coefficient.

2. High-order flux computation

We compute the high-order fluxes using the method of lines. Two schemes must be chosen: a scheme for integrating the solution in time and a scheme for computing the spatial derivatives. High-order accuracy requires that both schemes be high-order accurate.

High-order temporal integration scheme. We use the RK4 scheme to advance the solution. Returning to the system of ODEs (10),

$$\frac{d\langle q \rangle}{dt} = -D \cdot \langle \vec{\boldsymbol{F}} \rangle(t),$$

we want to integrate $\langle q \rangle$ from t^n to t^{n+1}. RK4 is a fourth-order integration scheme that consists of computing a linear combination of stage-update variables k_s. The updates are defined as

$$\langle q \rangle^0 = \langle q \rangle(t^n), \qquad k_1 = -D \cdot \langle \vec{\boldsymbol{F}}(\langle q \rangle^0) \rangle \Delta t, \tag{15}$$

$$\langle q \rangle^1 = \langle q \rangle^0 + \tfrac{1}{2}k_1, \quad k_2 = -D \cdot \langle \vec{\boldsymbol{F}}(\langle q \rangle^1) \rangle \Delta t, \tag{16}$$

$$\langle q \rangle^2 = \langle q \rangle^0 + \tfrac{1}{2}k_2, \quad k_3 = -D \cdot \langle \vec{\boldsymbol{F}}(\langle q \rangle^2) \rangle \Delta t, \tag{17}$$

$$\langle q \rangle^3 = \langle q \rangle^0 + k_3, \qquad k_4 = -D \cdot \langle \vec{\boldsymbol{F}}(\langle q \rangle^3) \rangle \Delta t. \tag{18}$$

Each update variable k_s requires computing stage fluxes $\langle \boldsymbol{F}^d \rangle^s_{\boldsymbol{i}\pm\boldsymbol{e}^d/2} = \langle q\boldsymbol{u}^d \rangle^s_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}$. The stage fluxes are functions of the stage values $\langle q \rangle^s_{\boldsymbol{i}}$ and $\langle \boldsymbol{u}^d \rangle^s_{\boldsymbol{i}}$ exclusively, and the procedure for computing the fluxes will be described in the next section.

To perform the RK4 integration, we compute the appropriate linear combination of stage updates

$$\langle q \rangle(t^n + \Delta t) = \langle q \rangle(t^n) + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathbb{O}(h^5). \tag{19}$$

Using the conservation notation, this RK4 integration can also be described by

$$\langle q \rangle^{n+1}_{\boldsymbol{i}} = \langle q \rangle^n_{\boldsymbol{i}} - \frac{\Delta t}{h} \sum_{d=1}^{D} \big[\langle \boldsymbol{F}^d_H \rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} - \langle \boldsymbol{F}^d_H \rangle_{\boldsymbol{i}-\boldsymbol{e}^d/2} \big], \tag{20}$$

$$\langle \boldsymbol{F}^d_H \rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2} = \tfrac{1}{6}\big[\langle \boldsymbol{F}^d \rangle^{(0)}_{\boldsymbol{i}\pm\boldsymbol{e}^d/2} + 2\langle \boldsymbol{F}^d \rangle^{(1)}_{\boldsymbol{i}\pm\boldsymbol{e}^d/2} + 2\langle \boldsymbol{F}^d \rangle^{(2)}_{\boldsymbol{i}\pm\boldsymbol{e}^d/2} + \langle \boldsymbol{F}^d \rangle^{(3)}_{\boldsymbol{i}\pm\boldsymbol{e}^d/2} \big]. \tag{21}$$

High-order spatial difference schemes. We use high-order finite-difference methods to approximate the surface fluxes associated with the spatial derivatives. The fluxes $\langle q\boldsymbol{u}^d \rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}$ for the spatial derivatives are functions only of the cell-averaged $\langle q \rangle_{\boldsymbol{i}}$ and $\langle \boldsymbol{u}^d \rangle_{\boldsymbol{i}}$ at any time. Several methods were explored in this study for computing $\langle q \rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}$, including high-order centered-difference schemes and upwind schemes. The advantage of the upwind methods is that they have greater diffusion especially in regimes where the phase error begins to rise. The upwind methods only require a small additional computation, and the stability of similar-order centered-difference and upwind methods is almost identical. Although not investigated in this study, high-order centered-difference schemes with hyperdiffusive fluxes [28] offer a possible alternative to the upwind ones used here.

The interpolation formulae corresponding to the spatial differencing schemes used are presented below. For compactness, the following notation will be used:

$$\langle q \rangle^n_{\boldsymbol{i}+\boldsymbol{e}^d/2} = \sum_{s=-S}^{S} a_s \langle q \rangle^n_{\boldsymbol{i}+s\boldsymbol{e}^d}, \tag{22}$$

where S is the width of the stencil and a_s are the coefficients. The odd-ordered methods use the full range of coefficients, whereas the even-ordered methods have no coefficient at $s = -S$.

- Fourth-order centered difference ($S = 2$):

$$\{a_s : s = -S+1, \ldots, S\} = \tfrac{1}{12}\{-1, 7, 7, -1\}. \tag{23}$$

- Fifth-order upwind ($S = 2$):

$$\{a_s : s = -S, \ldots, S\} = \tfrac{1}{60}\{2, -13, 47, 27, -3\}. \tag{24}$$

- Sixth-order centered difference ($S = 3$):

$$\{a_s : s = -S+1, \ldots, S\} = \tfrac{1}{60}\{1, -8, 37, 37, -8, 1\}. \tag{25}$$

- Seventh-order upwind ($S = 3$):

$$\{a_s : s = -S, \ldots, S\} = \tfrac{1}{420}\{-3, 25, -101, 319, 214, -38, 4\}. \tag{26}$$

- Ninth-order upwind ($S = 4$):

$$\{a_s : s = -S, \ldots, S\} = \tfrac{1}{2520}\{4, -41, 199, -641, 1879, 1375, -305, 55, -5\}. \tag{27}$$

Product rule. To complete the flux computation, we must compute the average of the product of the scalar variable and the velocity ($\langle q\boldsymbol{u}^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}$). The 2D product rules for second-, fourth-, and sixth-order accuracy are

$$\langle q\boldsymbol{u}^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} = \langle q\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}\langle \boldsymbol{u}^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} + \mathbb{O}(h^2), \tag{28}$$

$$\langle q\boldsymbol{u}^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} = \langle q\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}\langle \boldsymbol{u}^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} + \tfrac{1}{12}h^2 \sum_{d'\neq d} \frac{\partial q}{\partial x^{d'}}\frac{\partial \boldsymbol{u}^d}{\partial x^{d'}} + \mathbb{O}(h^4), \tag{29}$$

$$\begin{aligned}\langle q\boldsymbol{u}^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} = {} & \langle q\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}\langle \boldsymbol{u}^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} + \tfrac{1}{12}h^2 \sum_{d'\neq d}\left(\frac{\partial q}{\partial x_{d'}}\frac{\partial \boldsymbol{u}^d}{\partial x_{d'}}\right) \\ & + \tfrac{1}{1440}h^4 \sum_{d'\neq d}\left(3\frac{\partial^3 q}{\partial x_{d'}^3}\frac{\partial \boldsymbol{u}^d}{\partial x_{d'}} + 3\frac{\partial^3 \boldsymbol{u}^d}{\partial x_{d'}^3}\frac{\partial q}{\partial x_{d'}} + 2\frac{\partial^2 \boldsymbol{u}^d}{\partial x_{d'}^2}\frac{\partial^2 q}{\partial x_{d'}^2}\right) + \mathbb{O}(h^6).\end{aligned} \tag{30}$$

The possible sources of error in the product formulae above are computing the averages $\langle q\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}$ and $\langle \boldsymbol{u}^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}$ and computing the partial-derivative sums. We have already discussed several methods and their accuracy for computing $\langle q\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}$. The velocity fields are analytic for advection, so $\langle \boldsymbol{u}^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}$ introduces no error. The derivative terms in the summations above were computed exclusively using centered-difference approximations of appropriate accuracy. For example, the derivatives in the fourth-order-accurate product formula were computed using a second-order centered difference. The derivatives in the sixth-order formula were computed to fourth order (for the term multiplied by h^2) and to second order (for the term multiplied by h^4).

We note that the product rule has no contribution in a 1D problem or in any multidimensional problem with constant velocity. To obtain an arbitrary $\mathbb{O}(h^N)$-accurate solution for a multidimensional problem with varying velocity, we need to ensure that the product rule along with the time integrator and the spatial differencing scheme are all at least $\mathbb{O}(h^N)$. In this study, our overall solution accuracy was constrained by the use of RK4 for integration. However, lower spatial discretization errors are produced using the sixth-order product rule, in place of the fourth-order rule, with fifth- and higher-order-accurate spatial differencing schemes.

Method	Stability constraint
4th center	$\sigma \lesssim 2.06/D$
5th upwind	$\sigma \lesssim 1.73/D$
6th center	$\sigma \lesssim 1.78/D$
7th upwind	$\sigma \lesssim 1.69/D$
9th upwind	$\sigma \lesssim 1.60/D$

Table 1. Stability of methods for varying spatial difference operators and dimensionality D.

Stability. We compute the stability for each high-order scheme to determine the allowable time-step size following the procedure in [5]. Stability for the method of lines requires the eigenvalues of the right-hand side to lie within the stability region of the time integrator. These eigenvalues are computed by diagonalizing the semidiscrete system (10). For advection the eigenvalues are defined as the product of the velocity and the spatial derivative operator:

$$\frac{d\langle q\rangle}{dt} = \lambda\langle q\rangle, \tag{31}$$

$$\lambda\langle q\rangle = -\vec{\boldsymbol{u}}\frac{\partial}{\partial \boldsymbol{x}}\langle q\rangle. \tag{32}$$

The particular eigenvalues for each spatial differencing scheme will be presented later.

These eigenvalues must lie within the stability region of the time integrator. The stability region for RK4 is well known and can be described by its characteristic polynomial

$$P(z) = 1 + z + \tfrac{1}{2}z^2 + \tfrac{1}{6}z^3 + \tfrac{1}{24}z^4, \tag{33}$$

where $z = \Delta t\,\lambda$. Stability for this problem requires that $|P(z)| \le 1$. The resulting stability constraints for each spatial differencing scheme are presented in Table 1, where $\sigma = |u|\Delta t/h$.

Along with stability, the phase error and dissipation were computed (Figure 1). The dissipation was defined as $(1 - |g|)$, where

$$|g| = \sqrt{\mathrm{Re}(g)^2 + \mathrm{Im}(g)^2}, \tag{34}$$

$$\mathrm{Re}(g) = (1 + x + \tfrac{1}{2}x^2 + \tfrac{1}{6}x^3 + \tfrac{1}{24}x^4) - \tfrac{1}{2}y^2(1 + x + \tfrac{1}{2}x^2) + \tfrac{1}{24}y^4, \tag{35}$$

$$\mathrm{Im}(g) = y(1 + x + \tfrac{1}{2}x^2 + \tfrac{1}{6}x^3) - \tfrac{1}{6}y^3(1 + x), \tag{36}$$

and $z = x + iy$. The normalized phase error, $|1 - \alpha|$, is defined using

$$\alpha = \frac{\alpha(\beta)}{|u|} = -\frac{1}{\sigma\beta}\frac{\mathrm{Im}(g)}{\mathrm{Re}(g)}, \tag{37}$$

where $\beta = 2\pi kh$ for $k = 0, \pm 1, \pm 2, \ldots, \pm N/2$.

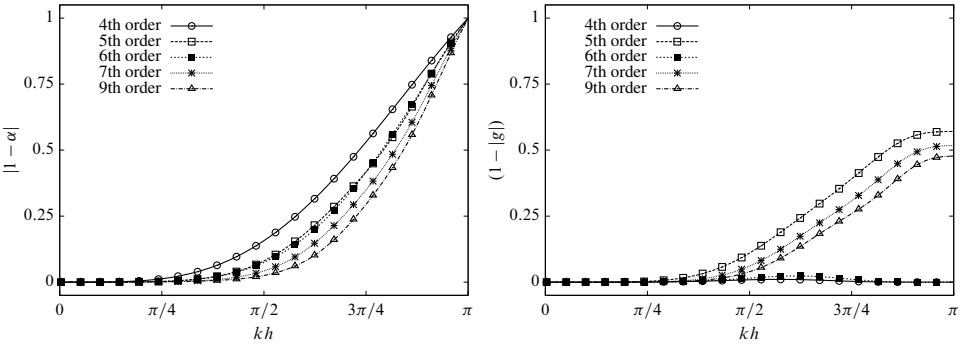


Figure 1. Normalized phase error (left) and dissipation (right) for the high-order methods ($\sigma = 0.8$).

Spatial differencing eigenvalues. The eigenvalues for each of the different high-order spatial differencing schemes are presented below. In each of the eigenvalue descriptions, β_d may range from $-\pi$ to π and is defined as $2\pi k_d h$ with $k_d = 0, \pm 1, \pm 2, \ldots, \pm N/2$.

- Fourth-order centered difference:

$$\lambda_4 = \frac{i}{12h} \sum_{d=1}^{D} u^d [16 \sin(\beta_d) - 2 \sin(2\beta_d)]. \tag{38}$$

- Fifth-order upwind:

$$\lambda_5 = \frac{1}{60h} \sum_{d=1}^{D} u^d \big[\big(-2 \cos(3\beta_d) + 12 \cos(2\beta_d) - 30 \cos(\beta_d) + 20\big) + i \big(2 \sin(3\beta_d) - 18 \sin(2\beta_d) + 90 \sin(\beta_d)\big)\big]. \tag{39}$$

- Sixth-order centered difference:

$$\lambda_6 = \frac{i}{60h} \sum_{d=1}^{D} u^d [2 \sin(3\beta_d) - 18 \sin(2\beta_d) + 90 \sin(\beta_d)]. \tag{40}$$

- Seventh-order upwind:

$$\lambda_7 = \frac{1}{420h} \sum_{d=1}^{D} u^d \big[\big(3 \cos(4\beta_d) - 24 \cos(3\beta_d) + 84 \cos(2\beta_d) - 168 \cos(\beta_d) + 105\big) + i \big(-3 \sin(4\beta_d) + 32 \sin(3\beta_d) - 168 \sin(2\beta_d) + 672 \sin(\beta_d)\big)\big]. \tag{41}$$

- Ninth-order upwind:

$$\lambda_9 = \frac{1}{2520h}\sum_{d=1}^{D} u^d\big[\big(-4\cos(5\beta_d) + 40\cos(4\beta_d) - 180\cos(3\beta_d) + 480\cos(2\beta_d) - 840\cos(\beta_d) + 504\big) + i\big(4\sin(5\beta_d) - 50\sin(4\beta_d) + 300\sin(3\beta_d) - 1200\sin(2\beta_d) + 4200\sin(\beta_d)\big)\big]. \tag{42}$$

3. Low-order flux computation

The low-order fluxes are computed using the CTU method [4; 21]. CTU is a first-order time-advancement scheme. The method is desirable over the simpler donor-cell upwind method because its stability is independent of dimensionality. However, this increased stability comes with a price. Instead of a single flux being defined by a single upwind value, the CTU flux is dependent upon a set of upwinded values. These values are determined by tracing the characteristic paths from the nodes that define the flux surface. This process involves an increasing number of Riemann solves as the dimensionality of the problem increases. In the 1D case, CTU is identical to donor-cell upwind.

4. Computing the hybridization coefficient

We compute the hybridization coefficient η using a modified multidimensional flux-corrected transport (FCT) algorithm. Note that the time superscript notation (n) for fluxes is dropped for the remainder of the paper, but it is implied. Our algorithm is based upon the method described first in [28]. Here is the generic FCT procedure:

(1) Compute the high-order fluxes $\langle \boldsymbol{F}_H^d \rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}$ over the cell volume $V_{\boldsymbol{i}}$.

(2) Compute the low-order fluxes $\langle \boldsymbol{F}_L^d \rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}$ and the corresponding low-order update

$$\langle q \rangle_{\boldsymbol{i}}^{td} = \langle q \rangle_{\boldsymbol{i}}^{n} - \frac{\Delta t}{h}\sum_{d=1}^{D}\big[\langle \boldsymbol{F}_L^d \rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} - \langle \boldsymbol{F}_L^d \rangle_{\boldsymbol{i}-\boldsymbol{e}^d/2}\big]. \tag{43}$$

(3) Compute the antidiffusive fluxes

$$\langle A^d \rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2} = \langle \boldsymbol{F}_H^d \rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2} - \langle \boldsymbol{F}_L^d \rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}. \tag{44}$$

(4) Limit the antidiffusive fluxes:

$$\langle A_\eta^d \rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2} = \eta^d_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}\langle A^d \rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}, \quad 0 \le \eta^d_{\boldsymbol{i}\pm\boldsymbol{e}^d/2} \le 1. \tag{45}$$

(5) Update the solution with the limited antidiffusive fluxes:

$$\langle q\rangle_{\boldsymbol{i}}^{n+1} = \langle q\rangle_{\boldsymbol{i}}^{td} - \frac{\Delta t}{h}\sum_{d=1}^{D}\left[\langle A_{\eta}^{d}\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} - \langle A_{\eta}^{d}\rangle_{\boldsymbol{i}-\boldsymbol{e}^d/2}\right]. \tag{46}$$

Limiting the antidiffusive flux. The primary challenge in the above formulation is computing the hybridization coefficients ($\eta_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}$). Following the procedure in [28], we compute the coefficients in the following manner.

Preconstrain the high-order fluxes $\langle \boldsymbol{F}_H\rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}$. This is a prelimiting step that in effect sets $\langle A^d\rangle_{\boldsymbol{i}\pm\boldsymbol{e}^d/2}$ to zero when it would otherwise admit diffusion and flatten the solution profile.

Compute the sum ($P_{\boldsymbol{i}}^{\pm}$) of all the antidiffusive fluxes into and out of the cell and a measure of the allowable flux into or out of the cell ($Q_{\boldsymbol{i}}^{\pm}$):

$$P_{\boldsymbol{i}}^{+} = \sum_{d=1}^{D}[\max(\langle A^d\rangle_{\boldsymbol{i}-\boldsymbol{e}^d/2}, 0) - \min(\langle A^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}, 0)], \tag{47}$$

$$Q_{\boldsymbol{i}}^{+} = ((q_{\max})_{\boldsymbol{i}} - \langle q\rangle_{\boldsymbol{i}}^{td})\frac{h}{\Delta t}, \tag{48}$$

$$P_{\boldsymbol{i}}^{-} = \sum_{d=1}^{D}[\max(\langle A^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}, 0) - \min(\langle A^d\rangle_{\boldsymbol{i}-\boldsymbol{e}^d/2}, 0)], \tag{49}$$

$$Q_{\boldsymbol{i}}^{-} = (\langle q\rangle_{\boldsymbol{i}}^{td} - (q_{\min})_{\boldsymbol{i}})\frac{h}{\Delta t}. \tag{50}$$

Compute the least upper bounds

$$R_{\boldsymbol{i}}^{+} = \begin{cases}\min(1.0,\, Q_{\boldsymbol{i}}^{+}/P_{\boldsymbol{i}}^{+}) & \text{if } P_{\boldsymbol{i}}^{+} > 0.0,\\ 0.0 & \text{otherwise,}\end{cases} \tag{51}$$

$$R_{\boldsymbol{i}}^{-} = \begin{cases}\min(1.0,\, Q_{\boldsymbol{i}}^{-}/P_{\boldsymbol{i}}^{-}) & \text{if } P_{\boldsymbol{i}}^{-} > 0.0,\\ 0.0 & \text{otherwise.}\end{cases} \tag{52}$$

Select the hybridization coefficient with the most restrictive upper bound:

$$\eta_{\boldsymbol{i}+\boldsymbol{e}^d/2} = \begin{cases}\min(R_{\boldsymbol{i}+\boldsymbol{e}^d}^{+},\, R_{\boldsymbol{i}}^{-}) & \text{if } \langle A^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} > 0.0,\\ \min(R_{\boldsymbol{i}}^{+},\, R_{\boldsymbol{i}+\boldsymbol{e}^d}^{-}) & \text{if } \langle A^d\rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} \le 0.0.\end{cases} \tag{53}$$

In the above description the user is provided with two design choices: preconstraint for the high-order flux and method of computing the solution bounds $(q_{\max})_{\boldsymbol{i}}$ and $(q_{\min})_{\boldsymbol{i}}$.

Computing the solution bounds. Compute initial estimates of the solution bounds, $(q_{\max})_{\boldsymbol{i}}$ and $(q_{\min})_{\boldsymbol{i}}$. First, compute the bounded solutions in a rectangular stencil ($B_{\boldsymbol{i}}$) that is $[2s_{\boldsymbol{i}}+1]^D$ cells in size, where $s_{\boldsymbol{i}}$ is the stencil size. Following convention the stencil size was fixed to be one cell.

After the stencil is determined, four bounds are computed: max based on $\langle q \rangle^n$, min based on $\langle q \rangle^n$, max based on $\langle q \rangle^{td}$, and min based on $\langle q \rangle^{td}$:

$$(q_{\max})_{\boldsymbol{i}}^n = \max(B_{\boldsymbol{i}}(\langle q \rangle^n)), \tag{54}$$

$$(q_{\min})_{\boldsymbol{i}}^n = \min(B_{\boldsymbol{i}}(\langle q \rangle^n)), \tag{55}$$

$$(q_{\max})_{\boldsymbol{i}}^{td} = \max(B_{\boldsymbol{i}}(\langle q \rangle^{td})), \tag{56}$$

$$(q_{\min})_{\boldsymbol{i}}^{td} = \min(B_{\boldsymbol{i}}(\langle q \rangle^{td})). \tag{57}$$

Then select the upper and lower bounds of the two estimates:

$$(q_{\max})_{\boldsymbol{i}} = \max((q_{\max})_{\boldsymbol{i}}^n, (q_{\max})_{\boldsymbol{i}}^{td}), \tag{58}$$

$$(q_{\min})_{\boldsymbol{i}} = \min((q_{\min})_{\boldsymbol{i}}^n, (q_{\min})_{\boldsymbol{i}}^{td}). \tag{59}$$

Accurate solution bounds at smooth extrema. For the vast majority of cells within the domain, the previous bound computation is sufficiently accurate. However, computing bounds at extrema is more complicated. Ideally the bounds need to keep the solution monotonic and positive, but the bounds should also not "clip" the solution. There are a few different methods for avoiding clipping, and we use a geometric construction that is only applied at smoothly varying extrema. It is based on the ideas in [6].

The first task is to detect a smooth extremum. The smooth-extremum criterion in 1D is

$$(\text{ext}^d)_{\boldsymbol{i}} = \min[(dq)_{\boldsymbol{i}} \cdot (dq)_{\boldsymbol{i}+\boldsymbol{e}^d}, (dq)_{\boldsymbol{i}-\boldsymbol{e}^d} \cdot (dq)_{\boldsymbol{i}+2\boldsymbol{e}^d}] \le 0.0, \quad 1.25 \cdot (dq_{\text{tot}})_{\boldsymbol{i}} < (tv)_{\boldsymbol{i}}, \tag{60}$$

where

$$(dq)_{\boldsymbol{i}} = \langle q \rangle_{\boldsymbol{i}}^{td} - \langle q \rangle_{\boldsymbol{i}-\boldsymbol{e}^d}^{td}, \tag{61}$$

$$(dq_{\text{tot}})_{\boldsymbol{i}} = |\langle q \rangle_{\boldsymbol{i}+2\boldsymbol{e}^d}^{td} - \langle q \rangle_{\boldsymbol{i}-2\boldsymbol{e}^d}^{td}|, \tag{62}$$

$$(tv)_{\boldsymbol{i}} = |(dq)_{\boldsymbol{i}+2\boldsymbol{e}^d}| + |(dq)_{\boldsymbol{i}+\boldsymbol{e}^d}| + |(dq)_{\boldsymbol{i}}| + |(dq)_{\boldsymbol{i}-\boldsymbol{e}^d}|. \tag{63}$$

This criterion has two parts. First, check for a sign change in the first derivative. The sign change will indicate either an extremum or a discontinuity in the solution. Second, ensure that the solution locally is not a perturbation of a discontinuity.

For a smooth multidimensional extremum, either $(\text{ext}^d)_{\boldsymbol{i}}$ must be true in all dimensions or it must be true for some d and the solution must remain constant along the dimensions in which $(\text{ext}^d)_{\boldsymbol{i}}$ is not true. We use this criterion to determine if the solution is constant:

$$\max(|(q_{\max}^d)_{\boldsymbol{i}}^{td} - \langle q \rangle_{\boldsymbol{i}}^{td}|, |(q_{\min}^d)_{\boldsymbol{i}}^{td} - \langle q \rangle_{\boldsymbol{i}}^{td}|) \le 10^{-14}, \tag{64}$$

where

$$(q^d_{\max})^{td}_{\boldsymbol{i}} = \max\big(\langle q\rangle^{td}_{\boldsymbol{i}-\boldsymbol{e}^d}, \langle q\rangle^{td}_{\boldsymbol{i}}, \langle q\rangle^{td}_{\boldsymbol{i}+\boldsymbol{e}^d}\big), \tag{65}$$

$$(q^d_{\min})^{td}_{\boldsymbol{i}} = \min\big(\langle q\rangle^{td}_{\boldsymbol{i}-\boldsymbol{e}^d}, \langle q\rangle^{td}_{\boldsymbol{i}}, \langle q\rangle^{td}_{\boldsymbol{i}+\boldsymbol{e}^d}\big). \tag{66}$$

Once we have determined that the solution at $V_{\boldsymbol{i}}$ is at a smooth extremum, we compute new values of $(q_{\max})_{\boldsymbol{i}}$ and $(q_{\min})_{\boldsymbol{i}}$. The first step is to construct a parabolic function from the local values of $\langle q\rangle^n$:

$$q^d(x) = \tfrac{1}{2}(d2q)^n_{\boldsymbol{i}}x^2 + \tfrac{1}{2}(\langle q\rangle^n_{\boldsymbol{i}+\boldsymbol{e}^d} - \langle q\rangle^n_{\boldsymbol{i}-\boldsymbol{e}^d})x + \langle q\rangle^n_{\boldsymbol{i}}, \tag{67}$$

where

$$(d2q)^n_{\boldsymbol{i}} = \langle q\rangle^n_{\boldsymbol{i}+\boldsymbol{e}^d} + \langle q\rangle^n_{\boldsymbol{i}-\boldsymbol{e}^d} - 2\langle q\rangle^n_{\boldsymbol{i}}. \tag{68}$$

The location of the vertex (x_c) is given by the ratio $-b/2a$, where a and b are the quadratic and linear coefficients from (67):

$$x_c = -\frac{\langle q\rangle^n_{\boldsymbol{i}+\boldsymbol{e}^d} - \langle q\rangle^n_{\boldsymbol{i}-\boldsymbol{e}^d}}{2(d2q)^n_{\boldsymbol{i}}} \tag{69}$$

and $-0.5 \le x_c \le 0.5$. Then, we evaluate the quadratic at the vertex to find the extremum value as well as deconvolve to get an estimate of the point value:

$$(q^d_{\text{ext}})_{\boldsymbol{i}} = \tfrac{1}{2}(d2q)^n_{\boldsymbol{i}}x_c^2 + \tfrac{1}{2}(\langle q\rangle^n_{\boldsymbol{i}+\boldsymbol{e}^d} - \langle q\rangle^n_{\boldsymbol{i}-\boldsymbol{e}^d})x_c + \langle q\rangle^n_{\boldsymbol{i}} - \tfrac{1}{24}(d2q)^n_{\boldsymbol{i}}. \tag{70}$$

We select the largest $(q^d_{\text{ext}})_{\boldsymbol{i}}$ or smallest $(q^d_{\text{ext}})_{\boldsymbol{i}}$ depending on the sign of the second derivative:

$$(q_{\text{ext}})_{\boldsymbol{i}} = \begin{cases} \max_d((q^d_{\text{ext}})_{\boldsymbol{i}}, (q_{\max})_{\boldsymbol{i}}) & \text{if } \operatorname{sgn}((d2q)^n_{\boldsymbol{i}}) \le 0.0, \\ \min_d((q^d_{\text{ext}})_{\boldsymbol{i}}, (q_{\min})_{\boldsymbol{i}}) & \text{otherwise.} \end{cases} \tag{71}$$

Finally, we compute the appropriate extremum bound by augmenting the solution value at the previous time by a scaled difference between the extremum value and the solution value

$$(q_{\max})_{\boldsymbol{i}} = \begin{cases} \langle q\rangle^n_{\boldsymbol{i}} + 2.0[(q_{\text{ext}})_{\boldsymbol{i}} - \langle q\rangle^n_{\boldsymbol{i}}] & \text{if } \operatorname{sgn}((d2q)^n_{\boldsymbol{i}}) \le 0.0, \\ (q_{\max})_{\boldsymbol{i}} & \text{otherwise,} \end{cases} \tag{72}$$

$$(q_{\min})_{\boldsymbol{i}} = \begin{cases} \langle q\rangle^n_{\boldsymbol{i}} + 2.0[(q_{\text{ext}})_{\boldsymbol{i}} - \langle q\rangle^n_{\boldsymbol{i}}] & \text{if } \operatorname{sgn}((d2q)^n_{\boldsymbol{i}}) > 0.0, \\ (q_{\min})_{\boldsymbol{i}} & \text{otherwise.} \end{cases} \tag{73}$$

Updating $R^{\pm}_{\boldsymbol{i}}$ at extrema. We flag the extrema at which the Laplacian is changing sign. In most cases this flag should not be activated. However, if the Laplacian does change sign at a smooth extremum, we turn the limiter on so that the low-order flux is chosen. This is a protective measure we have included in the algorithm.

We compute the D-dimensional approximation to the Laplacian over a three-point stencil

$$\Delta q_{\boldsymbol{i}} = \sum_{d=1}^{D} \frac{\partial^2 q_{\boldsymbol{i}}}{\partial x_d^2} \approx \sum_{d=1}^{D} \frac{(d2q)_{\boldsymbol{i}}^n}{h^2}. \tag{74}$$

If Δq changes sign anywhere in the three-point vicinity of $\boldsymbol{i}$, then we flag that cell $\boldsymbol{i}$. We then update the least upper bound multiplier at the flagged cells:

$$R_{\boldsymbol{i}}^{\pm} = 0 \quad \text{if } \boldsymbol{i} \text{ flagged.} \tag{75}$$

Preconstraining the high-order flux. We preconstrain the high-order flux where the corresponding antidiffusive flux would admit diffusion and flatten the solution profile. In practice, the value of the antidiffusive flux is edited instead of the high-order flux directly. Following [28] we set the antidiffusive flux to zero in these regions.

The baseline condition for applying the preconstraint is

$$\langle A^d \rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2} (\langle q \rangle_{\boldsymbol{i}+\boldsymbol{e}^d}^{td} - \langle q \rangle_{\boldsymbol{i}}^{td}) \leq 0.0. \tag{76}$$

However, this condition was not sufficient for our algorithm. The condition was occasionally satisfied at smooth areas in the solution. This manifested itself as a drop in convergence rate. We noticed that the preconstraint was mainly being applied near steep gradients and discontinuities. Moving forward, we only want to apply the preconstraint at discontinuities.

We added the following requirements to make sure we only apply this condition away from smooth areas:

$$\min\left[(d2q)_{\boldsymbol{i}+\boldsymbol{e}^d}^n \cdot (d2q)_{\boldsymbol{i}}^n, (d2q)_{\boldsymbol{i}}^n \cdot (d2q)_{\boldsymbol{i}-\boldsymbol{e}^d}^n, (d2q)_{\boldsymbol{i}+\boldsymbol{e}^d}^n \cdot (d2q)_{\boldsymbol{i}+2\boldsymbol{e}^d}^n\right] < 0.0, \tag{77}$$

$$|\langle A^d \rangle_{\boldsymbol{i}+\boldsymbol{e}^d/2}| \leq \frac{|(\boldsymbol{u}^d)_{\boldsymbol{i}+\boldsymbol{e}^d/2}|h}{2}(1-\sigma_{\boldsymbol{i}+\boldsymbol{e}^d/2})\frac{|(d2q)_{\boldsymbol{i}} + (d2q)_{\boldsymbol{i}+\boldsymbol{e}^d}|}{2}, \tag{78}$$

where $\sigma_{\boldsymbol{i}+\boldsymbol{e}^d/2} = |(\boldsymbol{u}^d)_{\boldsymbol{i}+\boldsymbol{e}^d/2}|\Delta t/h$.

The first constraint above (77) attempts to detect a discontinuity in the solution. However, there are smooth multidimensional solutions in which the second derivative naturally changes sign. The second constraint (78) seeks to preclude this case. The term on the right-hand side of the inequality (78) is the d-directional dissipation term, scaled by the cell size, in the modified equation analysis of CTU applied to the advection equation:

$$\frac{\partial q}{\partial t} + \sum_{d=1}^{D}\left(\boldsymbol{u}^d \frac{\partial q}{\partial x_d}\right) = \sum_{d=1}^{D}\left(\frac{\boldsymbol{u}^d h}{2}(1-\sigma_d)\frac{\partial^2 q}{\partial x_d^2}\right) + \mathbb{O}(h^2). \tag{79}$$

We are interested in comparing the magnitude of this dissipative term to the antidiffusive flux. The magnitude of the dissipative term is large in the neighborhood

of discontinuities. If this magnitude is large relative to the antidiffusive term, then we assume we are near a discontinuity and allow the preconstraint.

Steepening. In the previous subsection, we stated that the antidiffusive flux is set to zero in the regions where it would flatten the profile. As recognized in [8; 15], we may alternatively reverse the sign of and scale the antidiffusive flux. This process, known as steepening, seems to produce even sharper solution profiles at fronts. However, steepening has led to robustness issues in the past, including producing overshoots in the solution if the scaling factor is too large. For this reason we elected to keep the scaling factor at zero for the vast majority of the tests in this study.

If steepening is deemed necessary, we make the following changes to the algorithm: make the smoothness check more stringent and scale the antidiffusive flux instead of zeroing it. We found that scaling the right-hand side of (78) by 0.5 worked well to ensure the solution was discontinuous. In addition, we found that the best scaling factor for the tests in this study was 2.0.

5. Results

Results in one and two dimensions are presented. A total of four initial conditions were investigated. Of the four, one initial condition was smooth and the others contained a discontinuity. For the two-dimensional tests, we used a constant diagonal velocity field and a solid-body-rotation velocity field:

$$\boldsymbol{u} = [1, 1], \tag{80}$$

$$\boldsymbol{u} = 2\pi[y-0.5, 0.5-x]. \tag{81}$$

The center for the constant-velocity initial condition was in the middle of the domain, whereas it was offset by 0.25 of the grid height for the solid-body-rotation examples:

$$\boldsymbol{x}_c^{\text{const}} = (0.5, 0.5),$$
$$\boldsymbol{x}_c^{\text{solid}} = (0.5, 0.75).$$

Initial conditions. The smooth initial condition was constructed as a power of cosines

$$q_i(t_0) = \begin{cases} \cos^8\left(\frac{\pi}{2}(R/R_0)\right) & \text{if } R \le R_0, \\ 0 & \text{otherwise,} \end{cases} \tag{82}$$

with

$$R = \sqrt{(\boldsymbol{x_i} - \boldsymbol{x}_c)^2}, \quad \boldsymbol{x_i} \in [0, 1], \qquad R_0 = 0.15.$$

Three different discontinuous initial conditions were investigated. The first was a square and is described as

$$q_i(t_0) = \begin{cases} 1 & \text{if } |x_i^D - x_c^D| \le 0.15 \text{ for each } D, \\ 0 & \text{otherwise.} \end{cases} \tag{83}$$

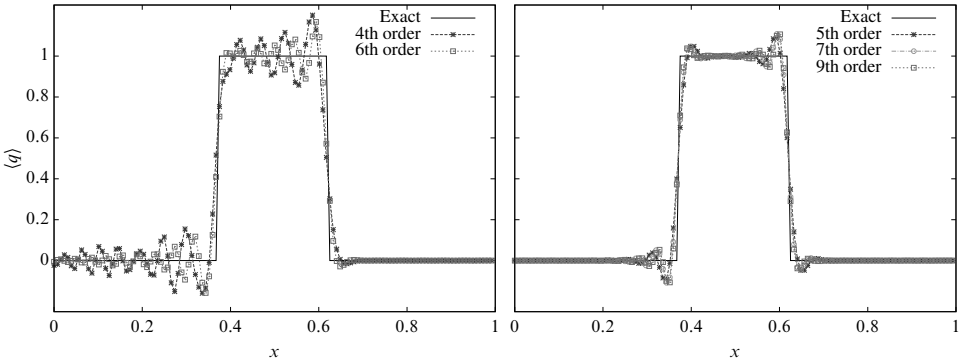


Figure 2. Centered (left) versus upwind (right) difference methods without limiting ($\sigma = 0.8$, $t = 1.0$, and $N = 128$).

The next is a semiellipse

$$q_i(t_0) = \begin{cases} \sqrt{1.0 - (R/R_0)^2} & \text{if } R \le R_0, \\ 0 & \text{otherwise,} \end{cases} \quad \text{with } R_0 = 0.25. \tag{84}$$

The last test case is the classic slotted cylinder in two dimensions

$$q_i(t_0) = \begin{cases} 1 & \text{if } |x_i^0 - x_c^0| \ge 0.025 \text{ or } x_i^1 \ge 0.85 \text{ (provided } R \le 0.15), \\ 0 & \text{otherwise,} \end{cases} \tag{85}$$

with $R = \sqrt{(\boldsymbol{x_i} - \boldsymbol{x}_c^{\text{solid}})^2}$.

Effectiveness of design features. We seek to demonstrate the need for each of the design choices made in the algorithm. The first feature is upwind methods for high-order fluxes. To show the effectiveness of the upwind methods, we examined the performance of the high-order fluxes with no limiting (Figure 2) on the square initial condition (83) in 1D. In the presence of a discontinuity, upwind methods produced much smaller magnitude oscillations than centered-difference methods. This outcome is consistent with the amplitude and phase error analysis presented earlier. When the limiter is turned on, the oscillations are clipped for both types of fluxes but the dispersive errors remain in the centered-difference solutions. Figure 3 shows this remaining dispersive error on the semiellipse initial condition (84).

The second design feature is the extremum-preserving limiter. We simulated advection with the smooth cosine initial condition (82) in 2D. This initial condition has a smoothly varying extremum in the middle of the domain. Figure 4, left, shows the excessive diffusion at this extremum that results from not using the extremum-preserving limiter. Not only is there excess diffusion, but also the convergence rate of the method suffers.

The preconstraint is another important feature of this algorithm. Figure 4, right, shows the effectiveness of the preconstraint on the square initial condition running

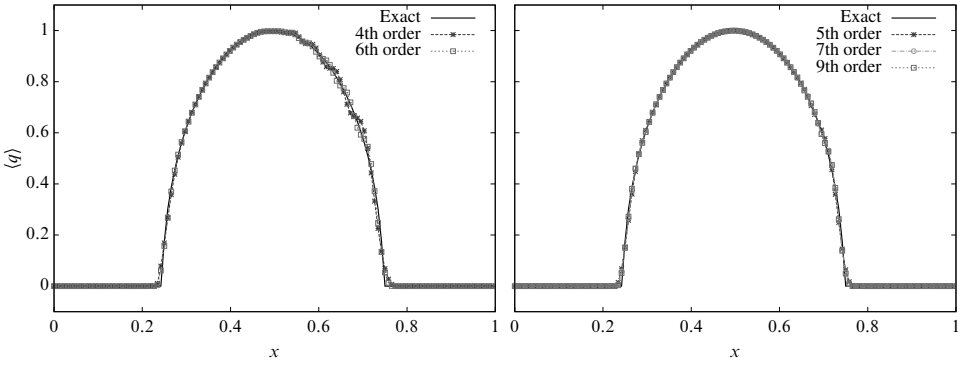


Figure 3. Centered (left) versus upwind (right) difference methods with limiting ($\sigma = 0.8$, $t = 1.0$, and $N = 128$).

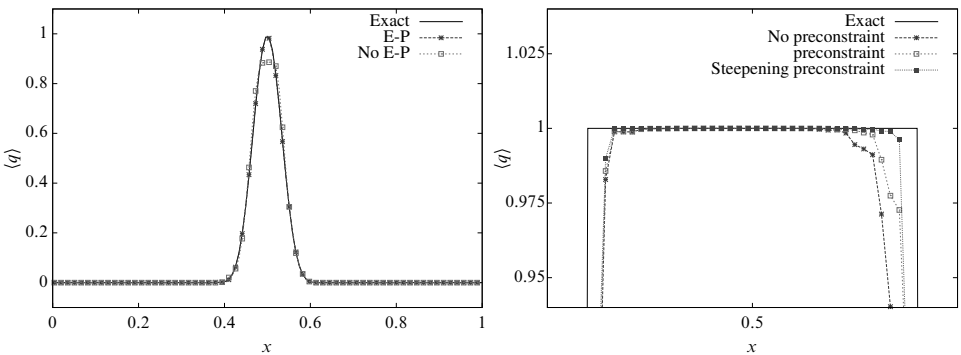


Figure 4. Effectiveness of extremum-preserving limiter at $t = 5.0$ (left) and preconstraint and steepener in 2D at $t = 1.0$ (right) ($\sigma = 0.8$ and $N = 128$).

right at the stability limit of the method in 2D. The preconstraint sharpens the solution profile near the front. The steepening preconstraint improves the solution profile even further. However, for most of the tests in this study, our preconstraint produced identical results with and without steepening. The remaining results do not include steepening.

One-dimensional tests. The first requirement for the limiter method is that it reduces to the high-order scheme away from discontinuities. All of the high-order schemes running at a large CFL number ($\sigma = 0.8$) achieved similar errors for smooth solutions in 1D (Figure 5). At this CFL number, the rate of convergence for each method was 4.0. We also computed the errors running at a lower CFL number ($\sigma = 0.2$). The error-reduction rate for each scheme at the low CFL number roughly matched the order of the spatial differencing scheme for the grid sizes displayed here. These results demonstrate that the limiter is not being activated in smooth regions, and hence, the first requirement is met.

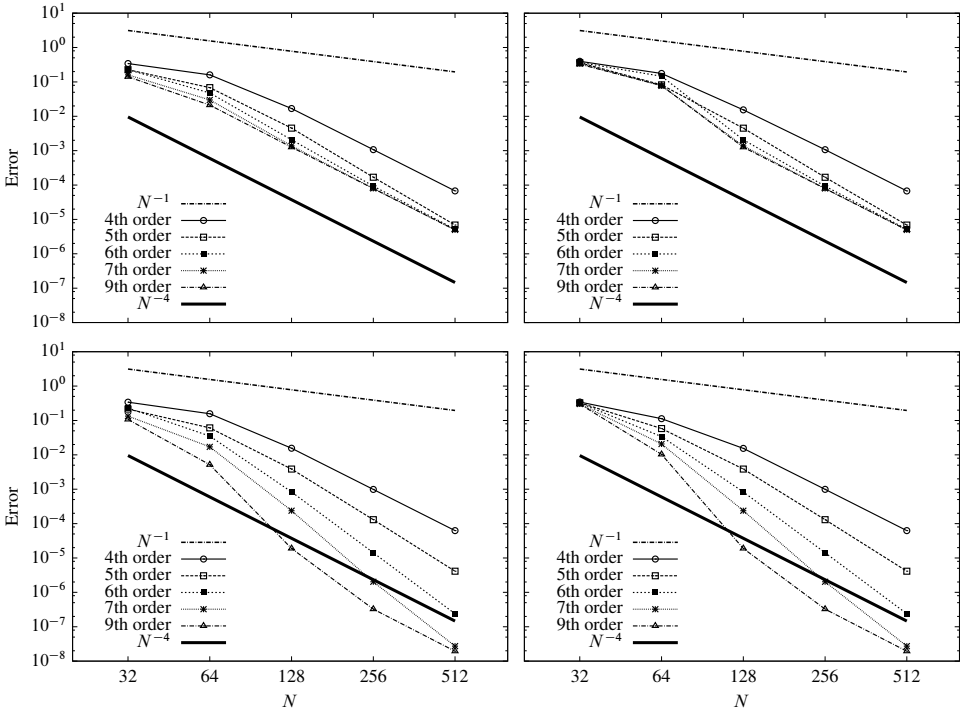


Figure 5. L_∞ errors in 1D. Left column: error without limiter. Right column: error with limiter. Top row: $\sigma = 0.8$. Bottom row: $\sigma = 0.2$.

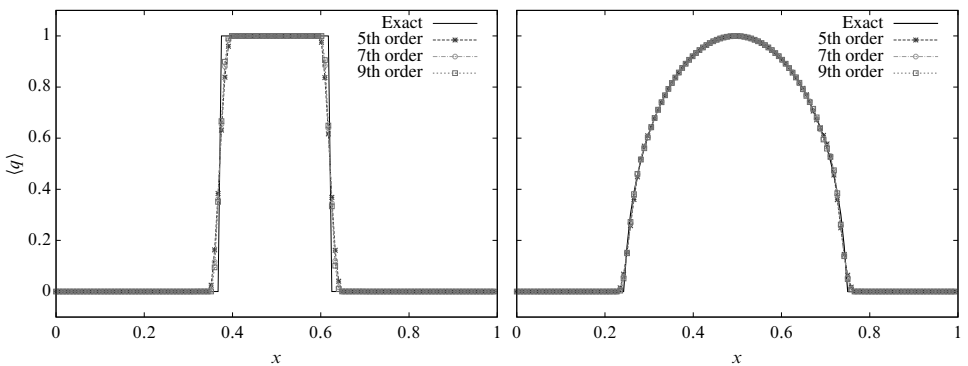


Figure 6. Discontinuous solutions with limiting and square (left) and semiellipse (right) initial conditions ($\sigma = 0.8$, $t = 1.0$, and $N = 128$).

The second requirement is that the limiter method accurately represent discontinuities. Figure 6 shows how the limiter performs on the square and semiellipse initial conditions in 1D. For both cases, the limiter method accurately captures the front and keeps the solution bounded.

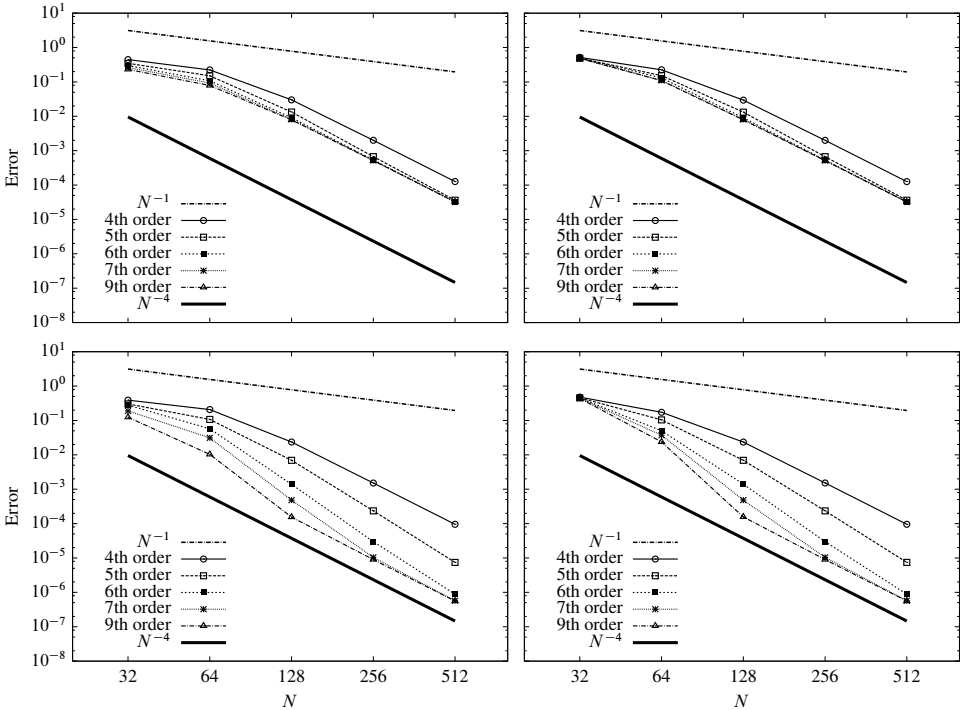


Figure 7. L_∞ errors in 2D. Left column: without limiter. Right column: with limiter. Top row: constant velocity. Bottom row: rotation velocity.

Two-dimensional tests. The errors for the smooth initial condition test in two dimensions are reported with and without the limiter for both velocity fields (Figure 7). As in the one-dimensional case, the high-order solution accuracy requirement is met. Interestingly, for the solid-body-rotation solution, the error reduction is greater than fourth order for many of the spatial differencing schemes. Also the ninth-order scheme is still convergent right at its theoretical stability limit ($\sigma \approx 0.8$).

The limiter also performs quite well at representing discontinuities in two dimensions. Various solution plots for discontinuous initial conditions are presented (Figures 8–10). All of the two-dimensional plots were generated using the ninth-order scheme in space, running at $\sigma = 0.8$. The square solution under constant velocity has few, if any, ripples and is nicely bounded (Figure 8). There is some distortion of the corners, particularly at the top-left and bottom-right. The square solution under solid-body rotation looks similar to the constant-velocity solution, and the corner issue is mitigated.

The semiellipse solution is likewise well resolved (Figure 9). As with the one-dimensional case there are some dispersive errors on the leading edge, but they are small. The solutions under both velocity fields are accurately bounded. The

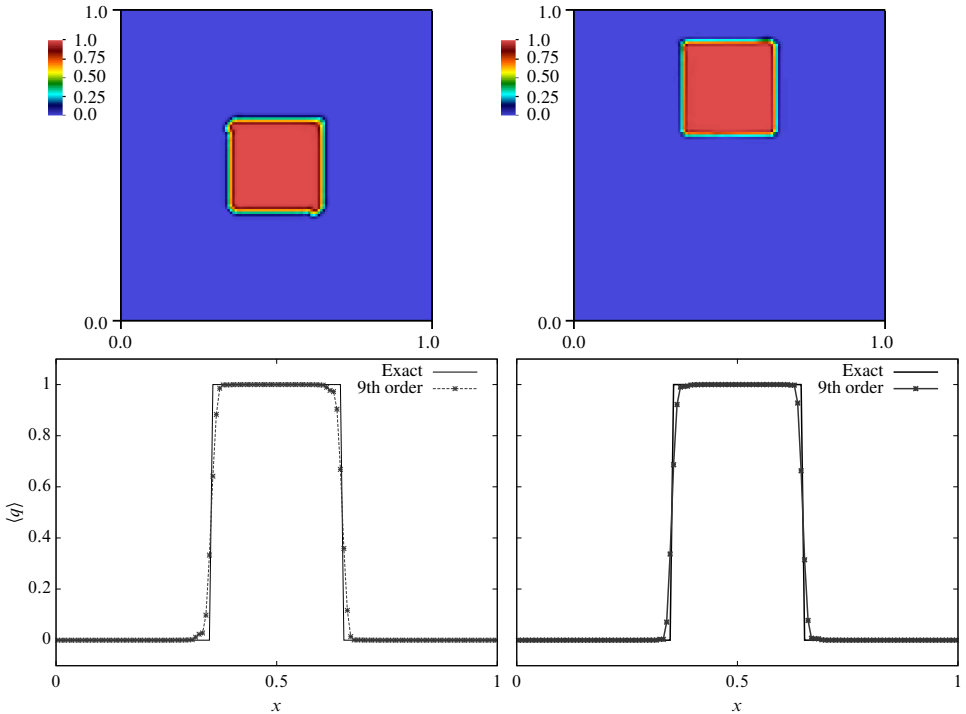


Figure 8. Square solutions ($\sigma = 0.8$, $t = 1.0$, and $N = 128$). Top left: constant velocity. Top right: solid-body rotation. Bottom row: centerline comparison.

semiellipse solution under solid-body rotation was centered at $\boldsymbol{x}_c^{\text{solid}} = (1.0, 1.5)$ to keep the edge of the condition away from the domain boundary. The domain was also expanded to $\boldsymbol{x_i} \in [0, 2]$.

The final test was the slotted cylinder (Figure 10). The limiter method keeps the solution bounded and resolves the fronts quite nicely. At lower grid resolutions, the slot can fill in and the bounds may not be enforced. But as the grid is refined, both of these issues are resolved.

6. Conclusions

We presented a new flux limiter based upon FCT that retains high-order accuracy for smooth solutions and captures fronts well. Our algorithm presented here uses CTU for low-order fluxes, upwind schemes for high-order fluxes, and RK4 for time integration. Our additions to the previous FCT method included a new computation for the extrema, an expanded preconstraint on the high-order fluxes, and a sixth-order-accurate finite-volume product rule. Furthermore, the limiter was only applied once per each time advance.

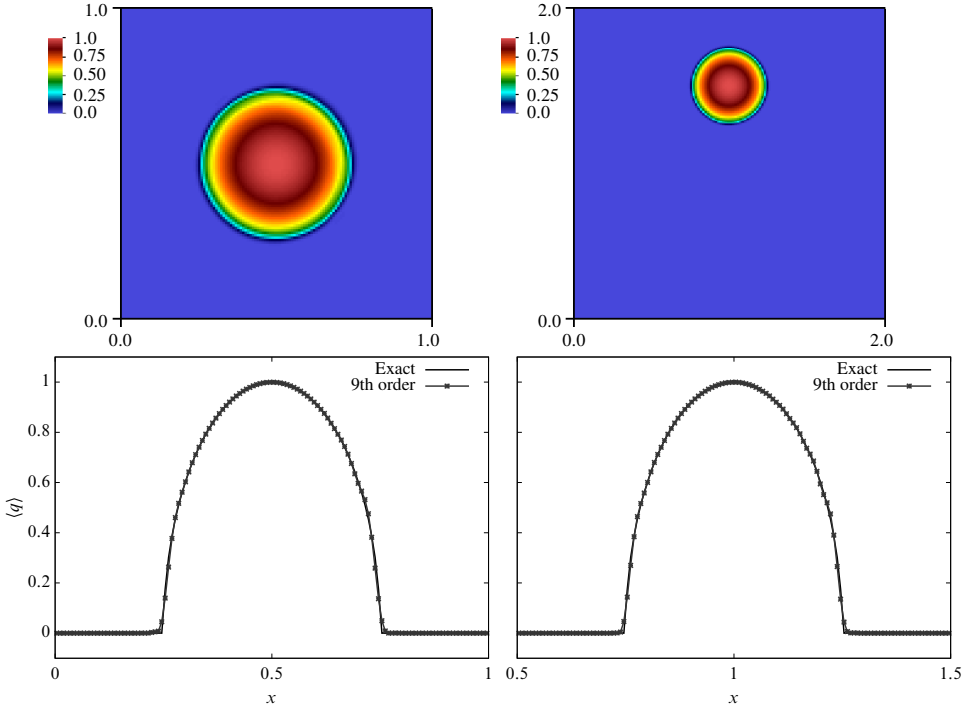


Figure 9. Semiellipse solutions ($\sigma = 0.8$, $t = 1.0$, and $N = 128$). Top left: constant velocity. Top right: solid-body rotation. Bottom row: centerline comparison.

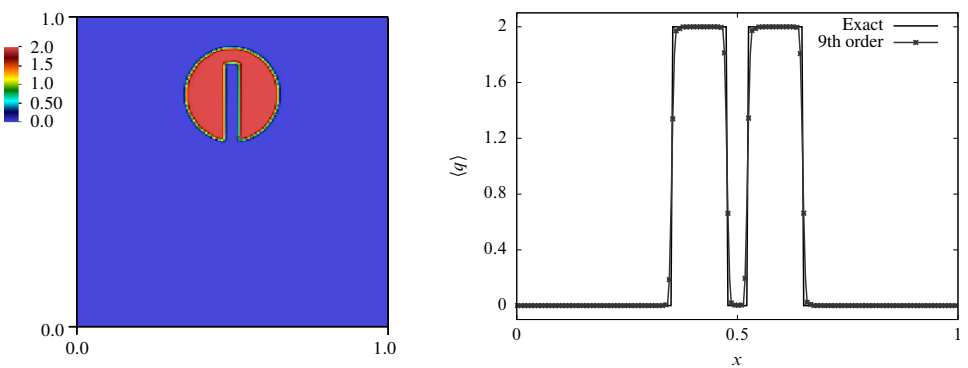


Figure 10. Slotted cylinder under solid-body rotation ($\sigma = 0.8$, $t = 1.0$, and $N = 256$). Left: solution. Right: centerline comparison.

The convergence rates for the smooth initial-condition tests were extremely similar in all three of the standard norms: L_1, L_2, and L_∞. On account of this only the L_∞ errors were displayed. Each high-order spatial discretization achieved fourth-order accuracy, at a minimum, for the smooth initial condition. In theory, we could also have achieved a higher overall order of accuracy with a more accurate

temporal integration scheme. Fourth-order accuracy was expected since we used a fourth-order-accurate time integrator (RK4) and a relatively large CFL number (0.8) for the majority of the test cases presented. At this CFL number, the error from the temporal discretization is much larger than the spatial discretization. To confirm this we also ran simulations at a low CFL number (0.2). The algorithm achieved higher convergence rates at this lower CFL number. We concluded that the errors from the spatial differencing schemes dominate the solution when running at low CFL numbers. The solid-body-rotation test also produced higher-order convergence rates than expected. This is likely explained by the fact that the solution is being advected at a range of CFL numbers tending toward zero as the solution approaches the center of the domain. Another interesting feature is the large difference in errors between the various spatial differencing schemes for the solid-body-rotation example. At the higher CFL number there are nearly two orders of magnitude difference in the max-norm error between the fourth- and ninth-order schemes. For the same initial condition running at the lower CFL number, there are over three orders of magnitude difference between these max-norm errors.

Potential extensions for this work are applying the limiter to systems of hyperbolic conservation laws, developing new high-order upwind methods with corner coupling, and further improving the preconstraint on the high-order fluxes. Applying the limiter to hyperbolic systems is the most direct extension of this work. Implementing this limiter for a compressible gas dynamics solver would be a good starting point. Following [15] we would not apply the limiter directly to the conserved variables but rather to the characteristic variables. The equations that govern the characteristic variables in 1D look like a system of decoupled advection equations, and the ideas presented here have clear applicability. Characteristic decomposition and limiting become more difficult in multiple dimensions, but there is a road map to follow. There are two general approaches available: either compute the fluxes and limiter along each direction in a split manner, or compute the fluxes in a multidimensional manner and limit the directional fluxes independently or sequentially. In previous studies [15] there was no discernible difference between the two approaches for the test cases analyzed, but it is important to consider both in general.

High-order, corner-coupled upwind methods for use with general multistage time integrators could remove the dimensional dependence of the stability. However, no upwind method of this nature currently exists. The preconstraint on the high-order fluxes is another area where additional study could pay off. In this work we found that the preconstraint affected a delicate balance between effectively representing discontinuities and retaining high-order accuracy in smooth yet complex areas. It was relatively simple to achieve one or the other. Ensuring both required testing many versions of the preconstraint. Introducing a steepening coefficient with the preconstraint improved discontinuity representation, but additional work must be

done to ensure robustness. The primary robustness concern with steepening is avoiding overshoot in the solution; however, our restrictions of only applying the preconstraint near fronts as well as limiting smooth extrema at which the Laplacian changes sign should reduce the risk of overshoot. We could also use other antidiffusive approaches for handling these types of contact discontinuities [27]. On the other hand, several canonical hyperbolic systems of equations have steepening mechanisms built into the physics, so steepening may only be useful in certain limited contexts.

Acknowledgements

This research was supported at the Lawrence Berkeley National Laboratory by the Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract Number DE-AC02-05CH11231.

References

[1] D. S. Balsara, C. Meyer, M. Dumbser, H. Du, and Z. Xu, *Efficient implementation of ADER schemes for Euler and magnetohydrodynamical flows on structured meshes — speed comparisons with Runge–Kutta methods*, J. Comput. Phys. **235** (2013), 934–969.

[2] J. P. Boris and D. L. Book, *Flux-corrected transport, I: SHASTA, a fluid transport algorithm that works*, J. Comput. Phys. **11** (1973), no. 1, 38–69.

[3] E. Casoni, J. Peraire, and A. Huerta, *One-dimensional shock-capturing for high-order discontinuous Galerkin methods*, Internat. J. Numer. Methods Fluids **71** (2013), no. 6, 737–755.

[4] P. Colella, *Multidimensional upwind methods for hyperbolic conservation laws*, J. Comput. Phys. **87** (1990), no. 1, 171–200.

[5] P. Colella, M. R. Dorr, J. A. F. Hittinger, and D. F. Martin, *High-order, finite-volume methods in mapped coordinates*, J. Comput. Phys. **230** (2011), no. 8, 2952–2976.

[6] P. Colella and M. D. Sekora, *A limiter for PPM that preserves accuracy at smooth extrema*, J. Comput. Phys. **227** (2008), no. 15, 7069–7076.

[7] P. Colella and P. R. Woodward, *The Piecewise Parabolic Method (PPM) for gas-dynamical simulations*, J. Comput. Phys. **54** (1984), no. 1, 174–201.

[8] C. R. DeVore, *An improved limiter for multidimensional flux-corrected transport*, technical report NRL/MR/6440-98-8330, Naval Research Laboratory, Washington, DC, 1998.

[9] M. Dumbser, O. Zanotti, A. Hidalgo, and D. S. Balsara, *ADER-WENO finite volume schemes with space-time adaptive mesh refinement*, J. Comput. Phys. **248** (2013), 257–286.

[10] S. H. Fuller and L. I. Millett (eds.), *The future of computing performance: game over or next level?*, National Academies, Washington, DC, 2011.

[11] A. Harten, *High resolution schemes for hyperbolic conservation laws*, J. Comput. Phys. **49** (1983), no. 3, 357–393.

[12] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy, *Uniformly high-order accurate essentially nonoscillatory schemes, III*, J. Comput. Phys. **71** (1987), no. 2, 231–303.

[13] D. Kuzmin, *A vertex-based hierarchical slope limiter for p-adaptive discontinuous Galerkin methods*, J. Comput. Appl. Math. **233** (2010), no. 12, 3077–3085.

[14] _____, *Hierarchical slope limiting in explicit and implicit discontinuous Galerkin methods*, J. Comput. Phys. **257**B (2014), 1140–1162.

[15] D. Kuzmin, R. Löhner, and S. Turek (eds.), *Flux-corrected transport: principles, algorithms, and applications*, 2nd ed., Springer, Dordrecht, 2012.

[16] D. Kuzmin, M. Möller, and S. Turek, *High-resolution FEM-FCT schemes for multidimensional conservation laws*, Comput. Methods Appl. Mech. Engrg. **193** (2004), no. 45–47, 4915–4946.

[17] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys. **115** (1994), no. 1, 200–212.

[18] R. Löhner, K. Morgan, J. Peraire, and M. Vahdati, *Finite element flux-corrected transport (FEM-FCT) for the Euler and Navier–Stokes equations*, Internat. J. Numer. Methods Fluids **7** (1987), no. 10, 1093–1109.

[19] R. Löhner, K. Morgan, M. Vahdati, J. P. Boris, and D. L. Book, *FEM-FCT: combining unstructured grids with high resolution*, Comm. Appl. Numer. Methods **4** (1988), no. 6, 717–729.

[20] P. McCorquodale and P. Colella, *A high-order finite-volume method for conservation laws on locally refined grids*, Commun. Appl. Math. Comput. Sci. **6** (2011), no. 1, 1–25.

[21] J. Saltzman, *An unsplit 3D upwind method for hyperbolic conservation laws*, J. Comput. Phys. **115** (1994), no. 1, 153–168.

[22] C.-W. Shu, *High order weighted essentially nonoscillatory schemes for convection dominated problems*, SIAM Rev. **51** (2009), no. 1, 82–126.

[23] V. A. Titarev and E. F. Toro, *ADER schemes for three-dimensional non-linear hyperbolic systems*, J. Comput. Phys. **204** (2005), no. 2, 715–736.

[24] E. F. Toro and V. A. Titarev, *Solution of the generalized Riemann problem for advection-reaction equations*, R. Soc. Lond. Proc. A **458** (2002), no. 2018, 271–281.

[25] B. van Leer, *Towards the ultimate conservative difference scheme, V: A second-order sequel to Godunov's method*, J. Comput. Phys. **32** (1979), no. 1, 101–136.

[26] S. Williams, A. Waterman, and D. Patterson, *Roofline: an insightful visual performance model for multicore architectures*, Commun. ACM **52** (2009), no. 4, 65–76.

[27] Z. Xu and C.-W. Shu, *Anti-diffusive flux corrections for high order finite difference WENO schemes*, J. Comput. Phys. **205** (2005), no. 2, 458–485.

[28] S. T. Zalesak, *Fully multidimensional flux-corrected transport algorithms for fluids*, J. Comput. Phys. **31** (1979), no. 3, 335–362.

Received May 4, 2015. Revised January 13, 2017.

CHRISTOPHER CHAPLIN: CChaplin@lbl.gov
Applied Numerical Algorithms Group, Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, MS 50A-3111, Berkeley, CA 94720, United States

PHILLIP COLELLA: PColella@lbl.gov
Applied Numerical Algorithms Group, Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, MS 50A-3111, Berkeley, CA 94720, United States

COMM. APP. MATH. AND COMP. SCI.
Vol. 12, No. 1, 2017
dx.doi.org/10.2140/camcos.2017.12.25

ACHIEVING ALGORITHMIC RESILIENCE FOR TEMPORAL INTEGRATION THROUGH SPECTRAL DEFERRED CORRECTIONS

RAY W. GROUT, HEMANTH KOLLA,
MICHAEL L. MINION AND JOHN B. BELL

Spectral deferred corrections (SDC) is an iterative approach for constructing higher-order-accurate numerical approximations of ordinary differential equations. SDC starts with an initial approximation of the solution defined at a set of Gaussian or spectral collocation nodes over a time interval and uses an iterative application of lower-order time discretizations applied to a correction equation to improve the solution at these nodes. Each deferred correction sweep increases the formal order of accuracy of the method up to the limit inherent in the accuracy defined by the collocation points. In this paper, we demonstrate that SDC is well suited to recovering from soft (transient) hardware faults in the data. A strategy where extra correction iterations are used to recover from soft errors and provide algorithmic resilience is proposed. Specifically, in this approach the iteration is continued until the residual (a measure of the error in the approximation) is small relative to the residual of the first correction iteration and changes slowly between successive iterations. We demonstrate the effectiveness of this strategy for both canonical test problems and a comprehensive situation involving a mature scientific application code that solves the reacting Navier–Stokes equations for combustion research.

1. Introduction

Since its introduction by Dutt et al. [12], the iterative nature of spectral deferred corrections (SDC) has been leveraged extensively to create efficient, high-accuracy methods for temporal integration tailored to specific types of problems. For example, in multi-implicit spectral deferred correction methods [3; 27], the terms in an advection-diffusion-reaction system are integrated separately with different time steps but coupled together using the SDC approach to achieve higher-order temporal accuracy than is achievable with traditional operator-splitting schemes. A similar approach is used to reduce splitting errors in a low-Mach combustion code by Nonaka et al. [32], where the SDC iterates are used to couple together interacting

MSC2010: primary 65D30, 65M12, 65M22, 80A25, 94B99; secondary 65M20.
Keywords: SDC, deferred correction, resilience, time integration, combustion.

physical processes. In this case, a significant advantage is realized in that the reduction in splitting error reduces nonphysical excursions into a chemical state space that artificially excites the intrinsic stiffness in the system. SDC has also been used to construct efficient time-parallel methods for partial differential equations (PDEs) [15]. Such desirable features that are not readily available in classical methods such as linear multistep or Runge–Kutta (RK) methods can make SDC an attractive choice for time integration despite the fact that SDC often requires relatively more function evaluations per time step.

There is growing concern about the impact of hardware errors — especially those that can lead to successful completion with erroneous results known as *silent data corruption*. This concern is driven by trends towards increasing concurrency as well as operation near design limits. Reducing voltage to improve energy efficiency has long been known to increase susceptibility to soft errors (e.g., [2; 10]). Further, modern designs tend to have elevated operating temperatures, which also increases the soft error rate [38; 9]. Wei et al. [41] define error resilience eloquently as *the ability of a program to prevent an error from becoming a silent data corruption*. We will look to leverage the iterative nature of SDC to provide algorithmic error resilience for temporal integration in the face of soft errors in the arithmetic operations and scratch variables used to update the solution. We expect that the iterative nature will be well suited to recover from transient errors. Chen et al. [7] note that an adaptive RK scheme, where the solution update is computed for two different time steps, should be able to detect soft faults as the two evaluations will be dramatically different if a soft fault has occurred. Benson et al. [1] constructed an error detector by evaluating an alternative time integration scheme (tailored for speed rather than stability, such as an embedded RK scheme of lower order, an explicit counterpart to a linear multistep method, or simple extrapolation) and examining the norm of the difference between the base and alternative schemes for anomalies in the context of a window of time steps. Benson et al. used the window of time steps because they noted that a hard threshold on the difference norm is meaningless because the expected norm of the difference changes with the solution. We use similar logic when inspecting the convergence rate between successive correction iterations to determine if the solution is acceptable.

The primary contributions of this paper are, firstly, to show that monitoring the residual in SDC correction sweeps can be used to detect soft (transient) errors resulting from hardware faults that could lead to silent data corruption using a reference integration algorithm and, secondly, to demonstrate the feasibility of recovering from soft errors by continuing SDC correction iterations. The intent of this paper is not to look at the details of low-level fault injection but rather at how a time integration algorithm can recover from those faults that migrate up the call tree through the return values of kernels. Here we use the term *kernel* to refer to

routines at the application level that compute terms in the governing differential equations being integrated. For example, the kernels from the application discussed in Section 2.4 are operations that compute advective or diffusive terms for the method-of-lines formulation or evaluate transport coefficients.

The remainder of this paper is organized as follows. In the next section, we present a brief outline of the SDC algorithm, relevant aspects of the state of research on fault injection and algorithmic resilience, and an overview of the combustion code used as an application benchmark later in the paper. We then turn in Section 3 to the behavior of the application in the context of single-occurrence synthetic errors, using the explicit Runge–Kutta integrator traditionally employed in the application code as a baseline to assess susceptibility to silent data corruption. We also examine the ability of the SDC algorithm to recover from such errors in an application test case and in a linear problem to demonstrate how the damping proceeds in a controlled setting. Finally, in Section 3.4, we look at a comprehensive error injection test case where we inject errors at an elevated rate into many runs of the application test case to see how our SDC iteration strategy narrows the distribution of the simulation output in a challenging scenario.

2. Preliminaries and related work

2.1. ***SDC formulation.*** Spectral deferred correction schemes were proposed by Dutt et al. [12] and subsequently developed significantly by Minion and colleagues (e.g., [31; 3; 28]). The basic approach is briefly recapped in this subsection before we consider additional aspects of its performance relevant to use in practical applications.

SDC schemes are based on recasting the ordinary differential equation (ODE)

$$\psi' = F(\psi, t), \qquad \psi(t_n) = \psi_n \tag{1}$$

over the time interval $t^n \le t \le t^{n+1}$ in integral form as

$$\psi(t) = \psi_n + \int_{t_n}^{t} F(\psi, \tau)\, d\tau. \tag{2}$$

Subdividing the interval $[t_n, t_{n+1}]$ by choosing $M+1$ Gauss–Lobatto quadrature nodes t_m ($t^n = t_0$ and $t^{n+1} = t_M$), for each node we can write the approximation

$$\phi_m = \psi_n + \Delta t \sum_{j=0}^{M} q_{m,j} F(\phi_j, t_j). \tag{3}$$

This integral provides an approximation to the solution $\psi_{n+1} \approx \phi_M$ at t^{n+1}; however, it effectively couples the solution at all of the quadrature nodes in the interval. Equation (3) is referred to as the collocation formulation (see, e.g., [20]) and

is equivalent to a fully implicit Runge–Kutta method with stages given by the quadrature nodes and coefficients in the Butcher tableau corresponding to the $q_{m,j}$. SDC can be thought of as providing an efficient iterative approach for computing the solution to this coupled system by iterative substepping over the nodes.

The basic idea is, given an approximate continuous solution $\phi^k(t)$, one can define a residual that measures the error in the approximation ϕ^k as

$$R(\phi^k, t) = \phi_n + \int_{t_n}^{t} F(\phi^k(\tau), \tau)\, d\tau - \phi^k(t). \tag{4}$$

If we define $c^k(t) = \phi(t) - \phi^k(t)$, then by substituting the definition of the residual into the integral form of the original equation, we obtain

$$c^k(t) = \int_{t^n}^{t} [F(\phi^k(\tau) + c^k(\tau), \tau) - F(\phi^k(\tau), \tau)]\, d\tau + R(\phi^k(t), t). \tag{5}$$

We then discretize this equation using the approximate residual

$$R_m(\phi^k) = \phi_n + \Delta t \sum_{j=0}^{M} q_{m,j} F(\phi_j^k, t_j) - \phi_m^k. \tag{6}$$

An explicit Euler-type method to discretize (5) gives the resulting update formula for the k-th iterate

$$c_{m+1}^k = c_m^k + \Delta t_m [F(\phi_m^{k+1}, t_m) - F(\phi_m^k, t_m)] + R_{m+1}(\phi^k) - R_m(\phi^k) \tag{7}$$

or, in a direct update form for $\phi_m^{k+1} = \phi_m^k + c_m^k$,

$$\phi_{m+1}^{k+1} = \phi_m^{k+1} + \Delta t_m [F(\phi_m^{k+1}, t_m) - F(\phi_m^k, t_m)] + I_m^{m+1}(\phi^k), \tag{8}$$

where

$$I_m^{m+1}(\phi^k) = \Delta t \sum_{j=1}^{M} (q_{m+1,j} - q_{m,j}) F(\phi^k(t_l), t_l) \approx \int_{t_m}^{t_{m+1}} F(\phi^k(\tau), \tau)\, d\tau. \tag{9}$$

I_m^{m+1} is the equivalent to the integral of the polynomial interpolant of ϕ^k over the interval $[t_m, t_{m+1}]$. Each such iteration can improve by one the formal order of accuracy of the approximate solution up to the order of the underlying quadrature. In the case of $M+1$ Lobatto nodes, the method achieves order $2M$ of convergence.

In the numerical results, we focus on SDC methods using three Lobatto nodes. If the initial value at all three nodes is taken to be the value at t_n and four correction iterations as described by (8) are performed, then the resulting method is formally fourth-order accurate. This is the same order of accuracy as the collocation or fully implicit Runge–Kutta method using the same three nodes, but the two schemes are not identical. In fact, the fourth-order method with four iterations can be considered

an explicit Runge–Kutta method with eight stages (see, e.g., [8]). Additional SDC iterations will not raise the formal order of accuracy but will drive the numerical solution to that from the collocation formulation with linear convergence with a rate proportional to the time step.

2.2. ***Soft error fault injection.*** For the purpose of this study, we follow the taxonomy of Bridges et al. [4], wherein hard faults are those that cause program interruptions and clearly denote an incomplete program execution while soft faults are typically observed as random bit flips, where one or more bits of memory are reversed. These faults are transient and do not indicate hardware damage, as opposed to persistent faults such as bits that are immutable due to a physical defect ("stuck bit" errors). Depending on where in the memory hierarchy they occur and the robustness of the algorithm, soft faults may not always lead to a solution failure but might result in an erroneous solution despite completely evading detection [14]. It might be acceptably inexpensive to provide soft fault detection and correction mechanisms for some, but not all, memory levels. For instance, error correction codes have been shown to correct a majority of soft faults in main memory [38] while processor registers are difficult to protect from soft faults [24]. Many factors such as altitude, age, temperature, and utilization are thought to affect error rates in real machines with a significant variability observed across various DRAM vendors. Recent studies have attempted to characterize and quantify error rates by surveying hardware logs from real machines, although a consensus is far from apparent. Schroeder et al. [35] study error rates from commodity clusters in Google's server fleet and observe that a majority of the errors are hard errors and soft errors are far less probable (a soft error probability of $\sim 2\%$ for every hard error). On the other hand, Sridharan et al. [39] find the opposite to be the case from a survey of data from two high-performance computing systems: Cielo at Los Alamos National Laboratory and Jaguar at Oak Ridge National Laboratory. Nonetheless, the most dominant mode seems to be single-bit errors (60%) with hard and soft errors being approximately equiprobable.

Considering the various enmeshed layers of software and hardware, the propagation of soft faults from one layer to another can be complicated to model. Strictly speaking, a bit flip at the level of hardware instructions is unlikely to migrate up to the application level as a single bit flip after several operations have been performed on the data. Even near the hardware level, a single bit flip in an instruction input might result in multiple bit flips in the destination register [14]. Despite this, there is some evidence that injecting single bit flips at higher levels produces similar effects from an application perspective as injecting errors near the hardware level. We choose this approach because it allows us to reason about the algorithmic sensitivity to the errors while eliminating the potentially confounding effects of interaction of the errors before reaching the application level.

Wei et al. compare the behavior of high-level fault injection (implemented at the LLVM intermediate representation (IR) level) to low-level fault injection (using Intel Pin tools) and find that, while there were significant differences in the number of program crashes between the two techniques, the IR-level fault injection is effective for assessing the impact of soft faults that result in silent data corruption. Wei et al. [41] also note that it is an established de facto standard that single bit flips [16] are an appropriate approach. In a related issue, Fang et al. [16] look at the effect of fault injection on multithreaded programs implemented using OpenMP and consider the sensitivity of the thread where the faults are injected due to the emphasis of the master thread on problem setup/teardown (phases of their chosen benchmarks that are particularly prone to resulting in ultimate silent data corruption in the output from fault injection). In our present application of interest, the setup/teardown phases are a very small portion of the overall run time, and otherwise the application follows a bulk-synchronous model.

Since our focus here is on the algorithmic robustness of SDC, we adopt a simple fault injection model. Considering that processor registers and arithmetic lookup units (ALUs) and floating point units (FPUs) are the most vulnerable to soft faults [41], we model soft faults as single bit flips in processor registers. However, we inject errors at the level of the application rather than at or very near the hardware level. We adopt an approach similar to, but even closer to the application level than, that of Wei et al. [41] and inject faults as if they manifest as single bit flips in register work arrays of the application level kernels that evaluate the terms contributing to the time derivative (F) of our system of ODEs.

2.3. ***Algorithmic approaches to resilience.*** Since a large number of scientific applications employ linear system solvers, methods to incorporate resilience in iterative linear solver algorithms have received wide attention. For example, Heroux and Bridges et al. [23; 4] propose a *fault-tolerant* version of the generalized minimal residual method (FT-GMRES) whereby the inner iteration that corresponds to the preconditioning step for the outer iteration is allowed to be unreliable. Rank deficiency of the subsequent upper-Hessenberg matrix could signal a potentially faulty execution of the inner iteration that would require some recovery strategy. The decision about whether a fault has occurred, and the subsequent recovery, is a global operation and involves agreement and hence global communication. Sloan et al. [36] suggest that error detection and recovery should instead be localized near the fault occurrence. The most expensive computational kernel in linear solver algorithms such as GMRES, the quasiminimal residual method (QMR), and conjugate gradient (CG) is usually a matrix-vector multiplication. Sloan et al. [36] contend that a soft error is most probable in this kernel and suggest an identity check that involves projecting the result of the matrix-vector multiplication onto a test vector. The

projection can be computed two different ways, so the results should agree if there were no faults in the original matrix-vector multiplication. By choosing the test vector to initially have all elements set to unity, they suggest a recursive hierarchical algorithm to hone in on the exact locations of faulty execution. Stoyanov and Webster [40] consider Jacobi and Gauss–Seidel fixed point iteration algorithms and leverage the identity that the norm of the difference between successive iterates should reduce at the same rate as the convergence of the algorithm. They suggest that checking this identity can be used as a method to identify errors due to soft faults and propose rejecting iterations that fail this test as a means to incorporate resilience.

Alternative approaches have also been proposed for explicit PDE schemes that are not iterative in nature. Mayo et al. [30] suggest combining two extremes in the tradeoff space for resilient explicit PDE algorithms: *artificial viscosity*, the physical mechanism that damps perturbations, and *triple modular redundancy*, the strategy of performing computations three times and accepting a result that was reproducible at least twice. They propose using multiple finite difference schemes over stencils of different widths at each grid point of the same formal order of accuracy to identify and discard outliers that might have been corrupted due to soft faults. Donzis and Aditya [11] propose asynchronous explicit finite difference schemes for PDEs that could be viewed as a potential resilience strategy. Typically, the explicit scheme for spatial derivatives requires the solution from neighboring grid points, which involves communication of ghost regions across processing elements (PEs). In the conventional implementation of such schemes, the communication and the calculation of spatial derivatives are completed by all PEs before the next time step is begun; i.e., all portions of the domain advance the solution in a time-step-synchronized fashion. However, one might envision that soft faults cause some portions of the domain to take longer to execute an iteration, introducing an asynchrony between PEs. Donzis and Aditya [11] propose asynchronous schemes whereby neighboring PEs could be at different time steps but still perform spatial derivatives to an intended order of accuracy. They model the asynchrony between neighboring PEs as a random process and show that while such schemes can be stable the accuracy in both time and space might be degraded. However, the desired order of accuracy severely limits the maximum asynchrony allowable between any pair of PEs.

2.4. ***S3D reacting flow solver and ignition benchmark problem.*** S3D is a solver for compressible reacting flows developed by Chen et al. [6]. S3D uses eighth-order finite-difference approximations of spatial derivatives with a method-of-lines discretization integrated temporally using a six-stage, fourth-order compact Runge–Kutta integrator from the family developed by Kennedy et al. [26]. Second

derivatives are obtained by repeated application of the discrete first-derivative operator. The code has been used to produce direct numerical simulations (e.g., sufficiently resolved to capture all relevant continuum scales for turbulence, chemical reaction, and turbulence-chemistry interaction) of a variety of turbulent combustion problems. Past problems include premixed flames [21; 33; 19], nonpremixed flames [22; 42; 18; 17], and autoignition problems [13; 34]. The code solves the compressible Navier–Stokes equations along with transport of the mass fractions of K chemically reacting species using a mixture-averaged transport model. The species density, momentum, and energy equations of hydrodynamics are given by

$$\frac{\partial}{\partial t}(\rho_k) + \nabla \cdot (\rho_k \boldsymbol{v}) + \nabla \cdot \boldsymbol{\mathcal{F}}_k = \dot{S}_k, \tag{10}$$

$$\frac{\partial}{\partial t}(\rho \boldsymbol{v}) + \nabla \cdot [\rho \boldsymbol{v}\boldsymbol{v}^T + p\boldsymbol{I}] + \nabla \cdot \tau = 0, \tag{11}$$

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot [(\rho E + p)\boldsymbol{v}] + \nabla \cdot [\boldsymbol{\mathcal{Q}} + \tau \cdot \boldsymbol{v}] = 0, \tag{12}$$

where ρ_k, $\boldsymbol{v}$, p, E, and $\dot{S}_k$ denote, respectively, the mass density for species k, fluid velocity, pressure, total specific energy, and chemical source term for species k for a mixture with K species ($k = 1, \ldots, K$). We note that $\sum_k \boldsymbol{\mathcal{F}}_k = 0$ and $\sum_k \dot{S}_k = 0$ so that summing the species equations gives conservation of mass with $\sum_k \rho_k = \rho$, the total fluid density. Note that $\boldsymbol{v}\boldsymbol{v}^T$ is a (tensor) outer product with T indicating transpose and $\boldsymbol{I}$ is the identity tensor (i.e., $\nabla \cdot p\boldsymbol{I} = \nabla p$). Transport properties are given in terms of the species diffusion flux $\boldsymbol{\mathcal{F}}$, viscous stress tensor τ, and heat flux $\boldsymbol{\mathcal{Q}}$. The viscous stress tensor is

$$\tau = -\eta(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T) + \tfrac{2}{3}\eta(\nabla \cdot \boldsymbol{v})\boldsymbol{I}, \tag{13}$$

where η is the shear viscosity. The heat flux is

$$\boldsymbol{\mathcal{Q}} = \sum_k h_k \boldsymbol{\mathcal{F}}_k - \lambda \nabla T, \tag{14}$$

where h_k is the enthalpy of the k-th species and λ is the thermal conductivity.

The diffusion velocity of the k-th species is modeled with a mixture-average formulation for $k-1$ species:

$$\boldsymbol{\mathcal{F}}_k = -\bar{D}_k \left[\nabla Y_k + Y_k \overline{W} \nabla W_k + (1 - M_k \overline{W}) \frac{1}{p} \nabla p \right], \tag{15}$$

where $Y_k = \rho_k/\rho$ is the mass fraction of species k, $\bar{D}_k$ is the mixture-averaged diffusion of species k, W_k is molecular weight of species k, and $\overline{W}$ is the mean molecular weight. The final species diffusion velocity is computed so as to enforce

conservation of mass:

$$\mathcal{F}_K = \sum_{k=1}^{K-1} -\mathcal{F}_k, \tag{16}$$

where K is the dominant species, typically N_2. Thermodynamic properties are temperature-dependent; the temperature is related to the energy by

$$E = e_s + \tfrac{1}{2} u_k u_k, \quad e_s = \int_{T_0}^{T} C_v \, dT - \frac{RT_0}{\overline{W}}, \tag{17}$$

where C_v is the mixture constant volume specific heat and R is the ideal gas constant.

The chemical source terms appearing in the species equations are computed by evaluating a chemical reaction network

$$\dot{S}_k = W_k \sum_{j=1}^{N_r} \nu_{kj} R_j, \tag{18}$$

where ν_{kj} are the stoichiometric coefficients for reaction j and the rates of the N_r reactions are given by expressions of the Arrhenius form used by [25]. For example, for a reaction where reactants A and B are converted into products C and D,

$$A + B \Leftrightarrow C + D, \tag{19}$$

the forward rate is given by

$$R_f = [A][B] k_f, \quad k_f = A_{fj} T^{\beta_j} \exp\left(\frac{-T_{aj}}{T}\right), \tag{20}$$

where A_{fj}, β_j, and T_{aj} are coefficients describing the j-th reaction with the reverse rate given by

$$R_b = [C][D] k_b, \quad k_b = \frac{k_f}{k_{\text{eq}}}, \; k_{\text{eq}} = \left(\frac{p}{RT}\right)^{\sum_{n=1}^{N} \nu_{nj}} \exp\left(\frac{\Delta S_j^0}{R} - \frac{\Delta H_j^0}{RT}\right), \tag{21}$$

where ΔS_j^0 and ΔH_j^0 are the entropy and enthalpy of formation differences across the reaction, respectively.

The ideal gas equation of state ($p = \rho RT/\overline{W}$) completes the description of the system. To solve (10)–(12), a method-of-lines approach is used where spatial derivatives are replaced by a finite difference operator of the form

$$\left[\frac{\partial \phi}{\partial x}\right]_i \approx \sum_{m=1}^{4} (\alpha_m \phi_{i-m} + \alpha_m \phi_{i+m}). \tag{22}$$

In the course of evaluating the time derivatives, S3D computes the various terms much as written here where various kernels (e.g., compute operand, apply derivative

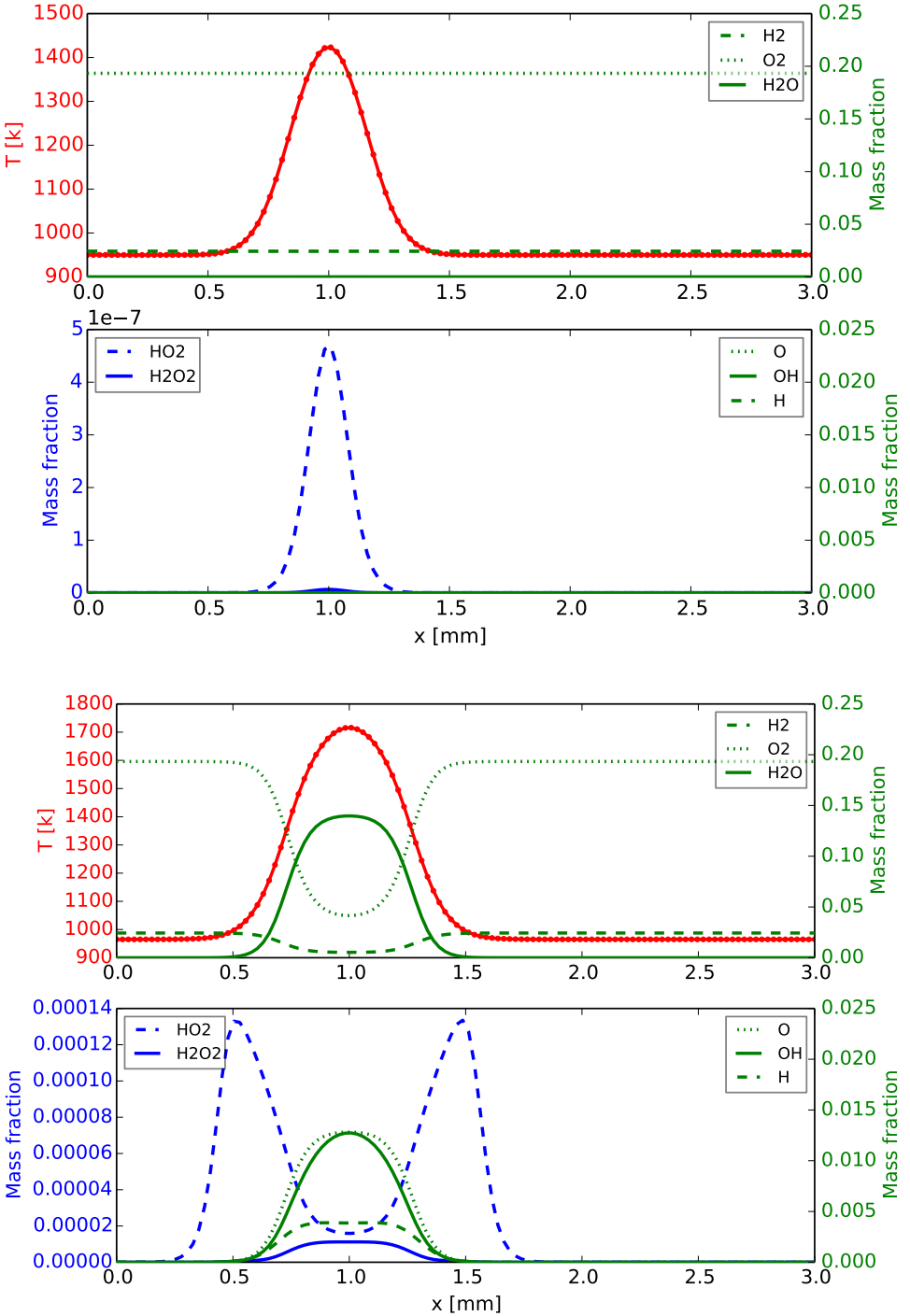


Figure 1. Spatial profiles of temperature and species mass fractions at $t = 5.5\,\mu$s (top) and $t = 30\,\mu$s (bottom) from reference solution obtained with 6,4-RK integrator.

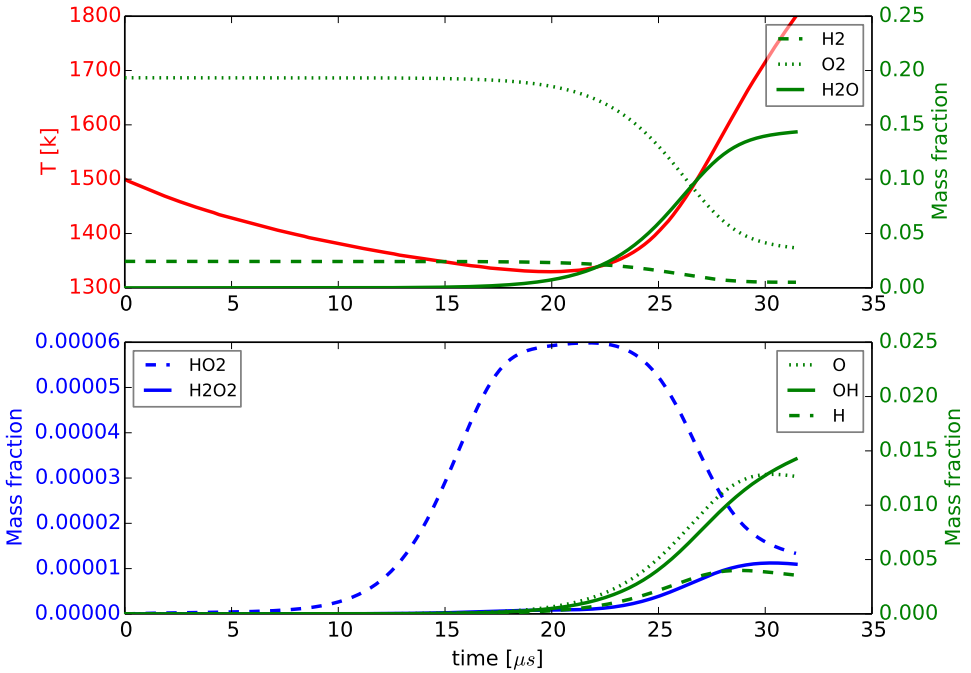


Figure 2. Temporal evolution of maximum temperature and species mass fractions at grid point coinciding with maximum temperature from reference solution obtained with 6,4-RK integrator.

operator, and compute diffusion velocity) operate on the entire solution grid until all of the time derivatives are completely assembled.

In the tests that follow, we will use a fixed time step and tolerate the extra computational cost as a necessary expense to remove one aspect that would make the results more difficult to interpret; in future work we plan to study the combination of SDC and adaptive time step control. The canonical problem is a one-dimensional simulation of a homogeneous mixture composed of hydrogen and air mixed in a stoichiometric ratio with a Gaussian temperature hot spot placed in the center of the domain according to

$$T(x) = T_0 + (T^* - T_0)\frac{1}{\sigma\sqrt{2\pi}}e^{-(x-x^*)^2/(2\sigma^2)}. \tag{23}$$

Solutions for this problem obtained using S3D and the native integrator used historically in S3D (the 6,4-RK algorithm) are shown in Figures 1 and 2. The problem is one-dimensional; 120 grid points are used to spatially resolve the ignition process, and a fixed time step of 5 ns is used in all cases. Figure 1 shows that the initial spatial temperature profile drives formation of a broad pool of hot radicals, led by HO_2 that is eventually consumed as the mixture proceeds towards ignition

and takes the first steps towards the formation of a front. In the time histories shown in Figure 2, the peak temperature decreases due to diffusive processes along with the slow buildup in HO_2 followed by an increase in H_2O_2, OH, and O and finally a rapid rise in temperature. The chemical mechanism is that of Li et al. [29]; *CHEMKIN's* [25] *tranlib* is used to evaluate transport coefficients for a mixture-averaged diffusion formulation. This test case has a relatively long "soaking" period, requiring approximately 5000 time steps before the onset of thermal runaway at 20 μs. This provides ample opportunity for small errors to compound into a large effect on the solution yet is relatively manageable for experimentation. A similar test case, a zero-dimensional ethylene-air ignition problem, is used by Spafford et al. [37] to study the effects of single precision on chemical reaction rate evaluation in the context of porting S3D kernels to a graphics coprocessor, where the test case proved sufficiently sensitive to the accuracy of the function evaluation that evaluating the reaction rates in single precision is insufficient to achieve an acceptable solution.

3. Soft error injection response

In this section we look at injecting two types of soft errors into major work arrays (those of the dimension of the solution grid) during the computation of the solution:

(A) scaling a single value within a work array by a large factor (i.e., multiplying by 10^4) and

(B) reversing the value of a bit at any position within the array (i.e., the value at any grid point could have any bit within it flipped, including the sign bit, the mantissa, and the exponent positions).

We use the type-A errors to explore the sensitivity of the solution to various intermediate values and to study how continued SDC sweeps can correct for such errors. Type-A errors produce a moderate response in that they typically produce a perturbed state that is incorrect but still physically plausible — the circumstance where silent data corruption is intuitively likely. Type-B errors are more realistic but can result in perturbed states that are physically inconsistent (e.g., negative temperatures). We use the bit flip approach, described in Section 2.2, for a comprehensive assessment of the technique integrated into the application code at the end of this section. In all cases, we limit our study to the work arrays that form return values of basic "simulation kernels" (which will be described in the following subsection). In other words, we leave persistent variables (e.g., stencil coefficients), control flow and instruction logic, and the solution vector at the start of the time step unperturbed.

3.1. ***Work array sensitivity.*** The S3D algorithm computes several quantities that are stored in work arrays during the evaluation of the right-hand-side function, and the sensitivity of the solution to perturbations varies widely between quantities.

To demonstrate this, we modified the code that evaluates the temporal derivatives of given quantities so that the results of kernel functions are perturbed. That is, evaluation of the time derivative involves application of several kernels called as functions that manipulate a set of work arrays:

$$\mathcal{R}^k(\vec{W}) \leftarrow \text{kernel}_k(\mathcal{I}^k(\vec{W}), \vec{q}, \dots), \tag{24}$$

where $\vec{W}$ is the vector of multidimensional work arrays, $\mathcal{R}^k(\vec{W})$ is the subset of the work arrays altered by the k-th kernel, $\mathcal{I}^k(\vec{W})$ is the subset of the work arrays used as input to the k-th kernel, $\vec{q}$ is the vector of conservative state variables at the start of the time step, and the $(\dots)$ represents the constants that complete the closure for the kernel. In this nomenclature we apply a perturbation function $\mathcal{P}$ that applies a single bit error (as discussed near the end of Section 2.2) to the return values of each kernel immediately after each kernel completes:

$$\widehat{\mathcal{R}}^k(\vec{W}) \leftarrow \mathcal{P}[\mathcal{R}^k(\vec{W})]. \tag{25}$$

The scratch/work arrays to which the error injection was applied have dimension $8 \cdot$ (nx, ny, nz), (nx, ny, nz, 3, 3), (nx, ny, nz, ns, 3), and (nx, ny, nz, ns) whereas the carryover arrays with dimension (nx, ny, nz, ns, 2) and the various setup arrays of smaller dimension as well as executable code have not been made vulnerable. Comparing perturbed runs to the baseline calculation described in Section 2.4 (again with a fixed 5 ns time step and the 6,4-RK time integration method), we obtain a sensitivity profile for the various work arrays. Figure 3 indicates the difference in the maximum temperature in the simulation domain at a fixed time near the end of the ignition delay when the various quantities are subjected individually to a one-time perturbation. The perturbation is applied to the output work array at the grid point where the temperature is maximum by multiplying the value by 10^4 immediately after the value is calculated during the first substage function evaluation for the time step beginning at $t = 5.5\,\mu$s. We observe, as many others have previously (e.g., [16]), that such error injection can result in different categories of behavior:

(1) The simulation fails in a detectable manner before completion, frequently as soon as the perturbation is injected. This is typically due to an unrealizable condition (e.g., temperature outside physical bounds or the sum of the species mass fractions becoming much larger than unity).

(2) There is no detectable effect on the calculated ignition delay.

(3) The calculation proceeds without apparent error to completion, and the calculated ignition delay is altered, with the size of the error depending on the size of the perturbation earlier in the calculation.

While the first type may slow scientific progress due to frequent restarts, it is the final type — the silent, undetectable errors that alter the result of the calculation —

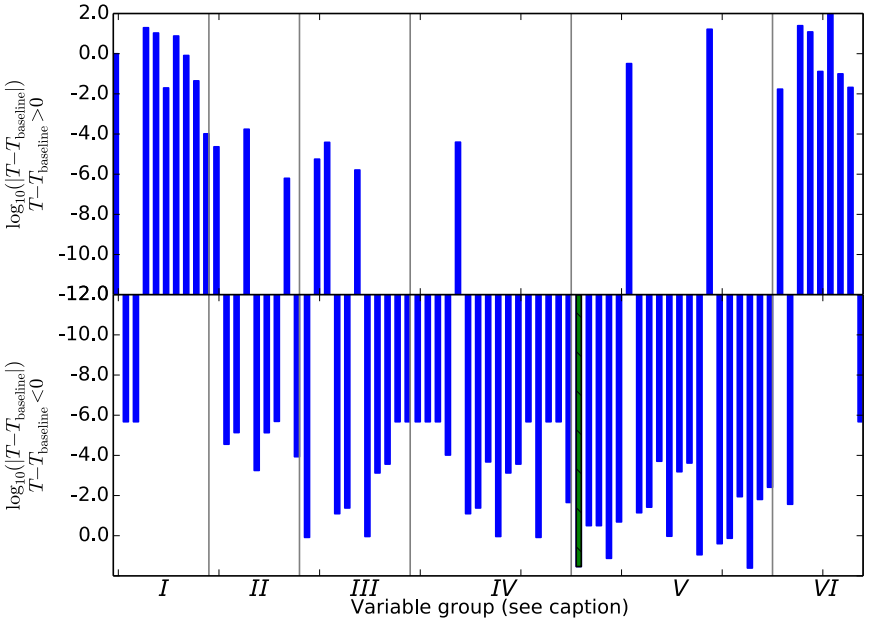


Figure 3. Difference in maximum temperature in domain from baseline at fixed time near end of ignition delay resulting from one-time perturbation of work arrays during calculation using traditional (6,4-RK) integration algorithm. Derivative of density is highlighted in green crosshatch. Work arrays where perturbation resulted in simulation crash are not shown. Variable groups are as follows: Group I, primaries (u, γ, c_p, Y_α); Group II, enthalpies (h_α); Group III, gradients (∇u, ∇T, ∇Y_α); Group IV, diffusive fluxes (τ_{ij}, J_α, J_T); Group V, second derivative operands and results (momentum, energy, species); and Group VI, reactions (S_α).

that are the most serious. Of the 93 kernel return values perturbed individually, for 74 of those variables the calculation proceeds to completion. The remainder result in simulation crashes (e.g., from out-of-bounds temperature extremes) and are not shown in Figure 3. The error in the temperature at the end of the solution ranged from 70 K below the correct temperature to 93 K above the correct temperature; this corresponds to impacting the calculated ignition delay by more than 5%. While it is difficult to make generalizations, perturbations that increased the reaction rate involving known ignition promoters for this mechanism (O, OH, and H) resulted in a significant temperature increase (hence, shorter ignition delay). Conversely, perturbations that increased the transport rates and hence hindered the buildup of radicals led to a decrease in temperature (hence, longer ignition delay). The perturbation that increased the source term for the continuity equation led to a decrease in temperature and is indicated in green in Figure 3 and will be considered in detail in the following subsection as representative of the error injections that led to silent data corruption.

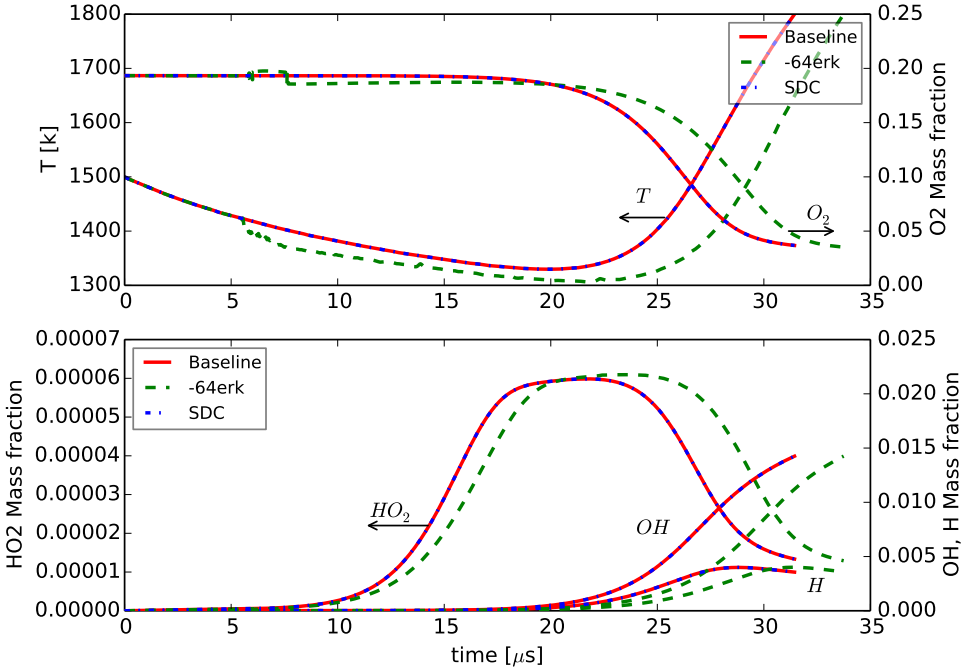


Figure 4. The effect of perturbation of the continuity equation source term on solution temporal evolution using 6,4-RK and SDC integration schemes; temporal plots were extracted at a fixed spatial location where error is injected. Notably, the SDC solution is indistinguishable from the baseline solution while the Runge–Kutta solution is silently and significantly corrupted.

3.2. ***Solution after injection of perturbation.*** Modifying the term that forms the density time derivative in the RHS evaluation, that is,

$$\frac{\partial \rho}{\partial t} = \frac{\partial(-\rho u)}{\partial x}, \tag{26}$$

results in a greater than 5% increase in the eventual predicted ignition delay and a significant change in the temperature at the end of the baseline ignition delay as highlighted in Figure 3 using the 6,4-RK integration method. Figure 4 shows the temporal evolution of the solution for temperature and key species for the baseline, unperturbed case, for the 6,4-RK integration and for SDC integration. The SDC integration is performed using three Gauss–Lobatto quadrature nodes and four correction sweeps. Figure 5 compares the spatial profiles at two different times: the time step after the error is injected and the time step when the baseline case reaches the ignition criterion. The perturbation grows over time after the injection (at $t = 5.5\,\mu$s). In keeping with the silent nature of the corruption, by inspection of the portion of the time history after the fault injection, it is difficult to tell that an error has occurred. Similarly, while it is obvious from looking at the spatial

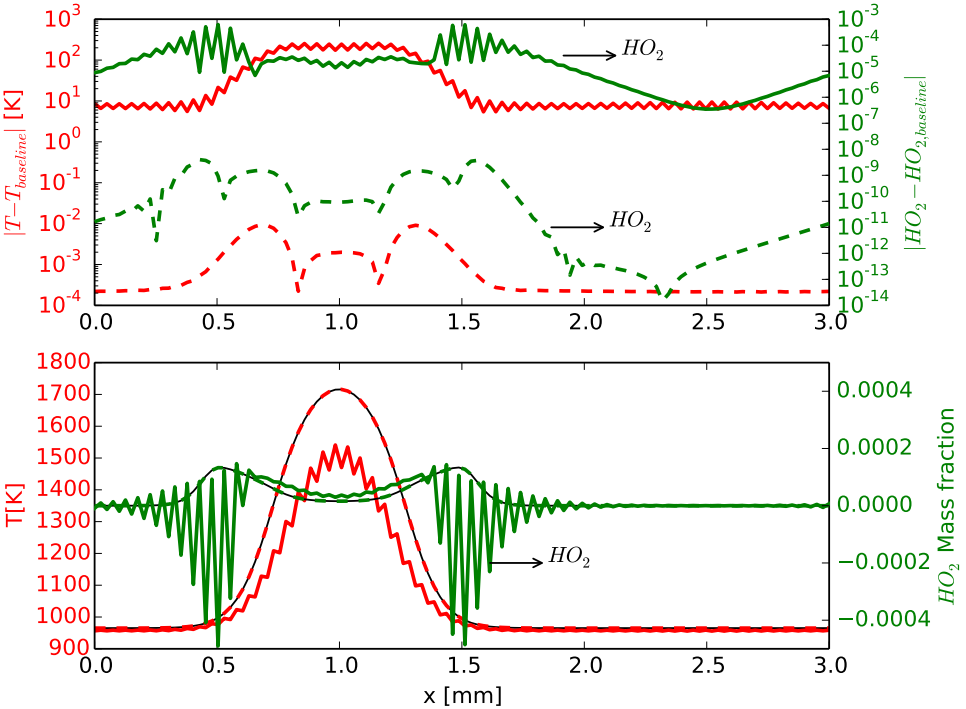


Figure 5. Effect of perturbation of continuity equation source term on solution using 6,4-RK and SDC integration schemes at time of baseline case ignition. Solid lines are the 6,4-RK solution, and dashed lines are the SDC solution for temperature (red) and HO_2 mass fraction (green). The upper plot shows the difference between the computed solution and the baseline while the lower plot shows the computed solution alongside the baseline (in solid black).

profiles at later times in Figure 5 that the solution is contaminated by ringing, it is not clear how to distinguish this from under-resolved physics [5]. Conversely, the solution traces obtained when using SDC with a fixed number of iterations are indistinguishable from the baseline, unperturbed case. This is an empirical demonstration of the tendency of the SDC iterations to recover from soft errors that result in silently corrupted data when using the traditional integration algorithm.

In Figure 6, the residual as given in (4) is shown over time; the curves shown for $|\mathcal{R}_k|$ are the magnitude of the residual for the k-th correction iteration. There is one value per time step plotted obtained at the end of the time step; the lower portion of the figure is an enlargement of the upper portion. We observe that the error injection can be detected by monitoring the residual, which increases sharply when the error is injected. In this experiment the number of SDC correction iterations is held constant. While this is sufficient to damp the error injected to the point where the solution is not qualitatively deteriorated, the residual at the final iteration has

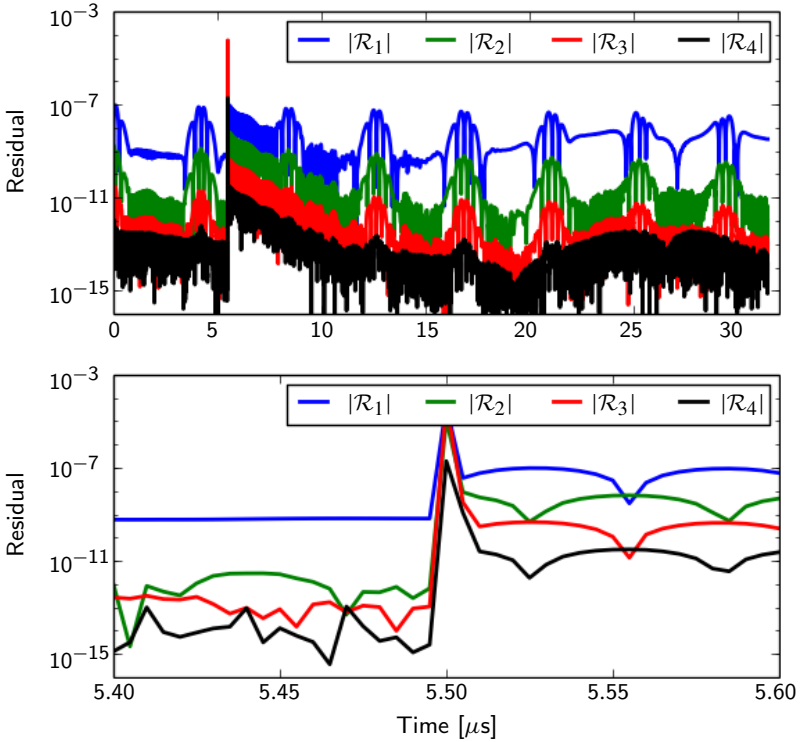


Figure 6. SDC residual response to soft perturbation for four correction sweeps given by $\mathcal{R}_1$–$\mathcal{R}_4$. The lower plot is a zoom in on the region of the fault injection; note that the fault is clearly evident by examining the residual.

not reached its final value prior to the error injection. It is several time steps later that the residual after the final time step reaches approximately the same magnitude as the final residual prior to the error injection. In the next subsection we will look at the response of a linear problem to shed more light on how further SDC iterations reduce the error in a contaminated solution.

3.3. ***Response of linear problem to perturbation.*** In Figures 7 and 8, a similar experiment is performed on the linear test problem

$$y'(t) = y(t), \qquad y(0) = 1 \tag{27}$$

over the interval [0, 1]. Three quadrature nodes are used, and four correction sweeps, including the initial explicit Euler predictor, are performed, giving a formally fourth-order method. The baseline behavior is shown in Figure 7 for comparison to the perturbed solution in Figure 8. A perturbation to the solution is introduced by using $y' = (1.5)y$ for the derivative evaluation during the third SDC sweep at the second quadrature node. For the unperturbed case, the solution error decreases monotonically with iteration count as seen in Figure 7. However, when the error is

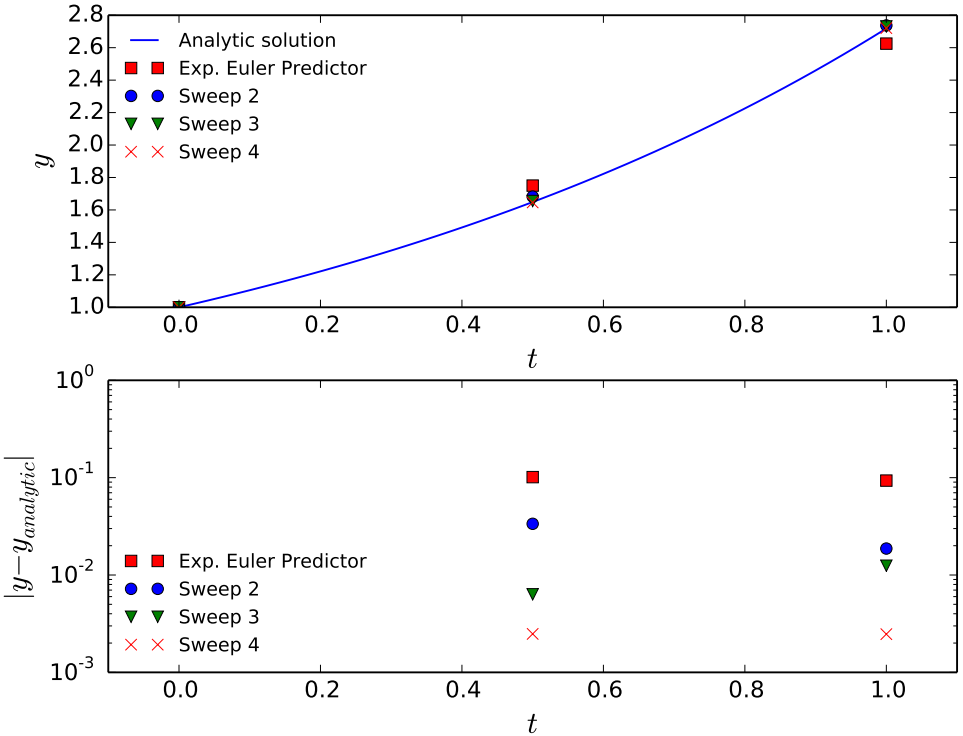


Figure 7. SDC iteration behavior for the linear problem ($y' = Ay$).

injected during the third sweep, we see the error jump up again to near the error in the initial predictor (Sweep 4 in Figure 8). After subsequent sweeps the error is reduced until after Sweep 7 the error in the solution is less than before the error is injected. Figure 9 demonstrates that the error damping is geometric for a wide range of perturbation magnitudes. The horizontal axis in Figure 9 corresponds to the size of the multiplicative perturbation to the derivative computation $y' = sy$. We find that across a wide range of s, both larger and smaller than unity, the error is damped with a consistent ratio. Also of note in Figure 9, we look at continuing the SDC iterations beyond the number of passes necessary for convergence of the reference solution. Even for large perturbations that result in errors several orders of magnitude larger than the reference solution converged error, the converged solution remains the same. This feature of SDC — the ability to recover from such large excursions from the true solution — leads to its natural resilience.

3.4. ***Response to multiple errors.*** In this subsection we conduct an experiment to assess the potential of the SDC iterations to recover from soft errors in a more realistic scenario. We use the baseline test case described above, running the simulation until a fixed time t_{ign}. We then extract the maximum temperature in

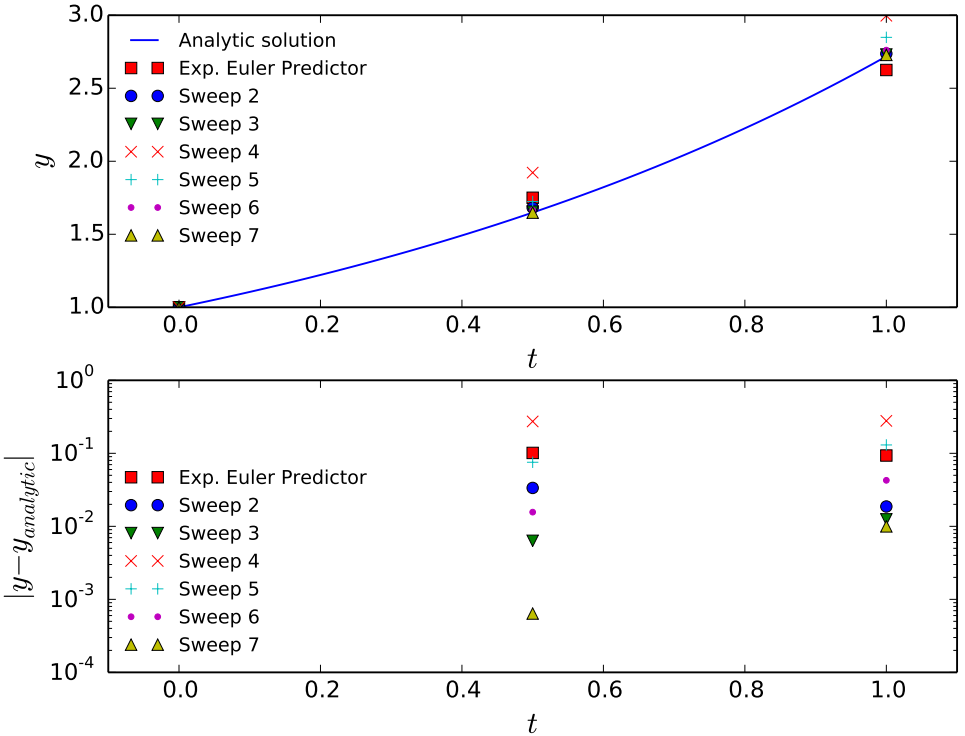


Figure 8. Effect of perturbation of the linear problem on SDC iteration convergence. Error is injected during Sweep 3 which results in an error larger than the initial predictor but is then damped by Sweeps 5 and 6.

the domain as a global measure of the simulation result. We set up our fault injection framework to inject bit flips into random bits in the return values of randomly selected kernels at random times, as discussed in Section 3. Specifically, we injected one fault every 5580 calls to the error injection callback per rank; this corresponds to approximately one fault every 10 time steps using the baseline RK time-advancement algorithm without error injection. Within this window, each process (MPI rank) chooses a random location where the fault will be injected. At the start of the fault injection window, each rank initializes a counter zero and chooses a random number in the range (0, 5580) to be the fault call. The counter is incremented each time the error injection callback is executed, and when it equals the fault call, a random bit within the valid range of the argument pointer is flipped. The counter continues to increment with successive calls, but without error injection, until it reaches the window size when it is reset and a new fault call count is chosen for the next window. For this one-dimensional calculation five MPI ranks were used.

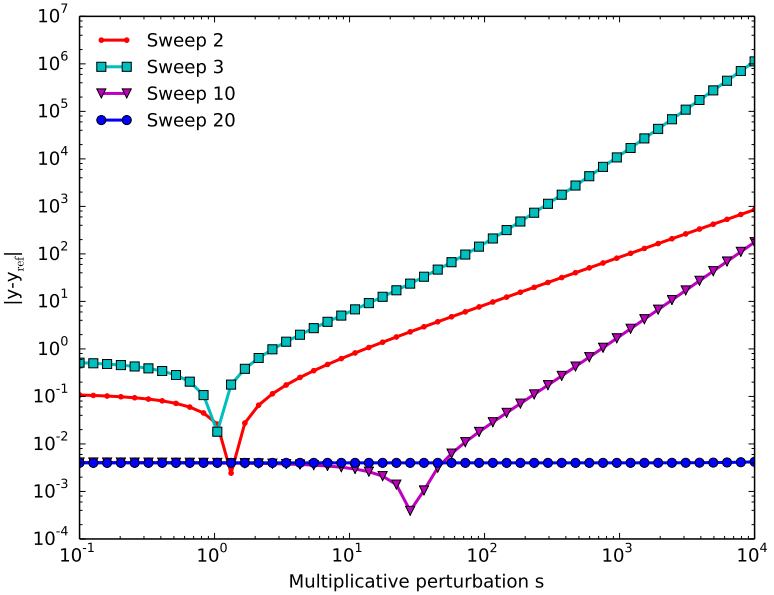


Figure 9. Effect of perturbation magnitude on SDC convergence rate. Reference solution y_{ref} is the analytic solution with $s = 1$.

When faults are injected randomly across the variable array, there is the potential that some faults will result in immediate crashes of the program as identified as the first type in Section 3.1, i.e., flipping the sign bit of major variables or changing the most significant bit in the exponent. These types of faults will cause the program to experience an unrecoverable error that is easily detectable. The test code solves a transport equation for total energy and computes temperature by using a Newton search to solve (17). Hence, a bounds check on the temperature is likely to pick up out-of-bounds issues on any of the variables that participate in (17). The code historically monitors the temperature range during the solution of (17) and terminates if it goes out of bounds. In order to allow the simulation to continue without a full restart, we cache the solution vector at the start of every outer time step and allow the simulation to restart from that point rather than terminating and restarting from a save file.

Furthermore, to deal with the final type of error (those leading to silent corruption), we modified the SDC algorithm to monitor its convergence through the reduction in the residual. We propose a strategy for mitigating soft errors — hardware-introduced faults that are stochastic and transient in nature — based on monitoring the behavior of the SDC correction through the residual to identify when a soft error has occurred and continuing iteration until the residual drops to the prescribed tolerance. In the case of nonrecoverable errors detected during the correction iterations, we restart

	Mean	Minimum	Maximum	Span	Variance
RK	1737.32	1728.92	1758.82	29.90	0.95
SDC	1737.30	1736.70	1743.69	7.00	0.04

Table 1. Variance in temperature at the end of calculation with error injection using baseline Runge–Kutta integration and the SDC approach of the same order. The result from both methods without error injection is 1737.25.

the time step. For each correction iteration (after the first) we compute

$$\mathcal{R}_1 = \frac{\max|\vec{R}_n|}{\max|\vec{R}_1|}, \tag{28}$$

$$\mathcal{R}_{n-1} = \frac{\max|\vec{R}_n|}{\max|\vec{R}_{n-1}|} \tag{29}$$

and continue the correction sweeps until $\mathcal{R}_1 < 10^{-5}$ and $\mathcal{R}_{n-1} > 0.9$, that is, until the residual is small compared to the residual from the first correction pass and is also not changing significantly between successive correction passes. The tolerance values for $\mathcal{R}_1$ and $\mathcal{R}_{n-1}$ were chosen to be consistent with the ratios found in the baseline case without fault injection at the end of the SDC iterations. The maximum number of correction passes is limited to eight, after which the time step is accepted. In practice, only a few time steps encountered this limit.

We conducted 1500 independent runs using both the baseline RK time integration and the proposed SDC method; the distribution of the temperature at the end of the calculation is shown in Figure 10 and Table 1. The data in the table demonstrates that the temperature values using SDC are significantly more clustered near the reference value than those computed using RK. There are some occurrences where the error introduced is sufficiently large that maximum SDC iteration limit is reached before without fully damping the error, which accounts for the nonzero variance in the sample of the SDC solutions. Despite this, the width of the distribution is far narrower than the corresponding baseline distribution. In a production environment, two alternatives to narrow the distribution further are available: more SDC iterations could be allowed, or the time step could be restarted if the iteration limit is reached. For this test, the rate of error injection is significantly magnified from realistic error rates, so either option is likely acceptable with minimal computational cost under realistic error rates for a target platform. This is meant to be illustrative: given the uncertainty in the error rates for future architectures, we demonstrate that the simulation can make progress and the effect of those errors are mitigated, but it is difficult to assess computational cost without knowing what the error rates are. This is left for future work as more realistic predictions and measurements of soft error rates on extreme-scale architectures become available. Satisfyingly, the resilient

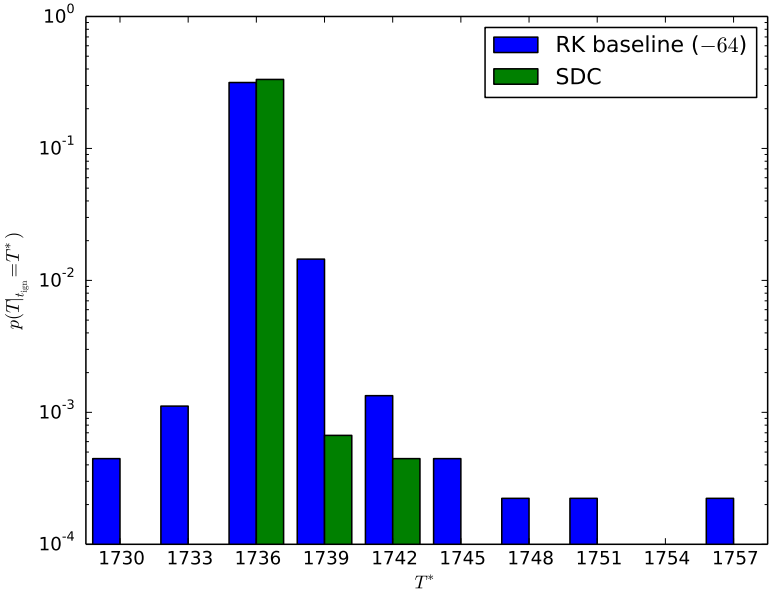


Figure 10. Distribution of temperature at end of calculation with error injection using baseline Runge–Kutta integration and SDC approach of same the order.

form of SDC does not add extra cost beyond a general formulation when there are no hardware faults. In the presence of extreme error rates, the algorithm still makes progress, with the vast majority of runs resulting in no silent data corruption and a clear path to including the remaining outliers available.

4. Conclusion

Natural extensions to a generic SDC algorithm have been proposed that are demonstrated to provide improved algorithmic resilience. It is shown that, in the face of a single transient error, continued SDC iterations beyond those normally required provide a viable approach to error recovery. In the case of elevated rates of stochastic errors, the algorithm can still make progress. In addition, although it is not explored here, the method provides a mechanism for detecting stuck bit errors that could potentially be used to trigger restarting the affected time step using different memory for the work arrays. When no errors are introduced, the suggested formulation reverts to a generic SDC algorithm, so there is no significant cost penalty for the modifications. The formulation is predicated on the ability to protect the integrity of the solution state between successive time steps as well as the program control flow. However, the work arrays used by the application code during a time step that typically comprise the majority of the memory usage can be exposed to significant error rates. This provides an opportunity for savings, where the need

for error correction is potentially reduced without resorting to measures such as redundant calculation that increase computational cost irrespective of the actual error rate realized. As such, the method is a way for application developers to design for potential increased soft error rates on future hardware without the penalty of degraded performance on less error-prone architectures.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research. Work at Lawrence Berkeley National Laboratory was supported by the Co-Design Program of the U.S. DOE Office of Advanced Scientific Computing Research under contract DE-AC02005CH11231.

References

[1] A. R. Benson, S. Schmit, and R. Schreiber, *Silent error detection in numerical time-stepping schemes*, Int. J. High Perform. C. **29** (2015), no. 4, 403–421.

[2] S. Borkar, *Design challenges of technology scaling*, IEEE Micro **19** (1999), no. 4, 23–29.

[3] A. Bourlioux, A. T. Layton, and M. L. Minion, *High-order multi-implicit spectral deferred correction methods for problems of reactive flow*, J. Comput. Phys. **189** (2003), no. 2, 651–675.

[4] P. G. Bridges, M. Hoemmen, K. B. Ferreira, M. A. Heroux, P. Soltero, and R. Brightwell, *Cooperative application/OS DRAM fault recovery*, Euro-Par 2011: parallel processing workshops (Bordeaux, 2011), vol. II, Lecture Notes in Computer Science, no. 7156, Springer, Berlin, 2012, pp. 241–250.

[5] D. L. Brown and M. L. Minion, *Performance of under-resolved two-dimensional incompressible flow simulations*, J. Comput. Phys. **122** (1995), no. 1, 165–183.

[6] J. H. Chen, A. Choudhary, B. de Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W. K. Liao, K. L. Ma, J. Mellor-Crummey, N. Podhorszki, R. Sankaran, S. Shende, and C. S. Yoo, *Terascale direct numerical simulations of turbulent combustion using S3D*, Comput. Sci. Disc. **2** (2009), no. 1, 015001.

[7] S. Chen, G. Bronevetsky, M. Casas-Guix, and L. Peng, *Comprehensive algorithmic resilience for numeric applications*, technical note LLNL-CONF-618412, Lawrence Livermore National Laboratory, Livermore, CA, 2013.

[8] A. Christlieb, B. Ong, and J.-M. Qiu, *Comments on high-order integrators embedded within integral deferred correction methods*, Commun. Appl. Math. Comput. Sci. **4** (2009), 27–56.

[9] C. Constantinescu, *Impact of deep submicron technology on dependability of VLSI circuits*, International Conference on Dependable Systems and Networks (Washington, DC, 2002), IEEE, Los Alamitos, CA, 2002, pp. 205–209.

[10] V. Degalahal, R. Ramanarayanan, N. Vijaykrishnan, Y. Xie, and M. J. Irwin, *The effect of threshold voltages on the soft error rate*, 5th International Symposium on Quality Electronic Design (San Jose, 2004), IEEE, Los Alamitos, CA, 2004, pp. 503–508.

[11] D. A. Donzis and K. Aditya, *Asynchronous finite-difference schemes for partial differential equations*, J. Comput. Phys. **274** (2014), 370–392.

[12] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT **40** (2000), no. 2, 241–266.

[13] T. Echekki and J. H. Chen, *Direct numerical simulation of autoignition in non-homogeneous hydrogen-air mixtures*, Combust. Flame **134** (2003), no. 3, 169–191.

[14] J. Elliott, F. Mueller, M. Stoyanov, and C. Webster, *Quantifying the impact of single bit flips on floating point arithmetic*, technical note ORNL/TM-2013/282, Oak Ridge National Laboratory, Oak Ridge, TN, 2013.

[15] M. Emmett and M. L. Minion, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci. **7** (2012), no. 1, 105–132.

[16] B. Fang, K. Pattabiraman, M. Ripeanu, and S. Gurumurthi, *Evaluating the error resilience of parallel programs*, 44th annual IEEE/IFIP International Conference on Dependable Systems and Networks (Atlanta, 2014), IEEE, Los Alamitos, CA, 2014, pp. 720–725.

[17] R. W. Grout, A. Gruber, H. Kolla, P.-T. Bremer, J. C. Bennett, A. Gyulassy, and J. H. Chen, *A direct numerical simulation study of turbulence and flame structure in transverse jets analysed in jet-trajectory based coordinates*, J. Fluid Mech. **706** (2012), 351–383.

[18] R. W. Grout, A. Gruber, C. S. Yoo, and J. H. Chen, *Direct numerical simulation of flame stabilization downstream of a transverse fuel jet in cross-flow*, P. Combust. Inst. **33** (2011), no. 1, 1629–1637.

[19] A. Gruber, R. Sankaran, E. R. Hawkes, and J. H. Chen, *Turbulent flame–wall interaction: a direct numerical simulation study*, J. Fluid Mech. **658** (2010), 5–32.

[20] E. Hairer and G. Wanner, *Solving ordinary differential equations, II: stiff and differential-algebraic problems*, 2nd ed., Springer Series in Computational Mathematics, no. 14, Springer, 1996.

[21] E. R. Hawkes and J. H. Chen, *Evaluation of models for flame stretch due to curvature in the thin reaction zones regime*, P. Combust. Inst. **30** (2005), no. 1, 647–655.

[22] E. R. Hawkes, R. Sankaran, J. C. Sutherland, and J. H. Chen, *Scalar mixing in direct numerical simulations of temporally evolving plane jet flames with skeletal CO/H_2 kinetics*, P. Combust. Inst. **31** (2007), no. 1, 1633–1640.

[23] M. A. Heroux, *Scalable computing challenges: an overview*, presentation at 2009 SIAM Annual Meeting, Sandia National Laboratories, Livermore, CA, 2009.

[24] A. A. Hwang, I. A. Stefanovici, and B. Schroeder, *Cosmic rays don't strike twice: understanding the nature of DRAM errors and the implications for system design*, Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems (London, 2012), ACM, New York, 2012, pp. 111–122.

[25] R. J. Kee, F. M. Rupley, E. Meeks, and J. A. Miller, *CHEMKIN-III: A FORTRAN chemical kinetics package for the analysis of gas-phase chemical and plasma kinetics*, technical note SAND96-8216, Sandia National Laboratories, Livermore, CA, 1996.

[26] C. A. Kennedy, M. H. Carpenter, and R. M. Lewis, *Low-storage, explicit Runge–Kutta schemes for the compressible Navier–Stokes equations*, Appl. Numer. Math. **35** (2000), no. 3, 177–219.

[27] A. T. Layton and M. L. Minion, *Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics*, J. Comput. Phys. **194** (2004), no. 2, 697–715.

[28] ———, *Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations*, BIT **45** (2005), no. 2, 341–373.

[29] J. Li, Z. Zhao, A. Kazakov, and F. L. Dryer, *An updated comprehensive kinetic model of hydrogen combustion*, Int. J. Chem. Kinet. **36** (2004), no. 10, 566–575.

[30] J. Mayo, R. Armstrong, and J. Ray, *Efficient, broadly applicable silent-error tolerance for extreme-scale resilience*, technical note SAND2012-8131, Sandia National Laboratories, Livermore, CA, 2012.

[31] M. L. Minion, *Semi-implicit spectral deferred correction methods for ordinary differential equations*, Commun. Math. Sci. **1** (2003), no. 3, 471–500.

[32] A. Nonaka, J. B. Bell, M. S. Day, C. Gilet, A. S. Almgren, and M. L. Minion, *A deferred correction coupling strategy for low Mach number flow with complex chemistry*, Combust. Theor. Model. **16** (2012), no. 6, 1053–1088.

[33] R. Sankaran, E. R. Hawkes, J. H. Chen, T. Lu, and C. K. Law, *Structure of a spatially developing turbulent lean methane–air Bunsen flame*, P. Combust. Inst. **31** (2007), no. 1, 1291–1298.

[34] R. Sankaran, H. G. Im, E. R. Hawkes, and J. H. Chen, *The effects of non-uniform temperature distribution on the ignition of a lean homogeneous hydrogen–air mixture*, P. Combust. Inst. **30** (2005), no. 1, 875–882.

[35] B. Schroeder, E. Pinheiro, and W.-D. Weber, *DRAM errors in the wild: a large-scale field study*, Commun. ACM **54** (2011), no. 2, 100–107.

[36] J. Sloan, R. Kumar, and G. Bronevetsky, *An algorithmic approach to error localization and partial recomputation for low-overhead fault tolerance*, 43rd annual IEEE/IFIP International Conference on Dependable Systems and Networks (Budapest, 2013), IEEE, Los Alamitos, CA, 2013.

[37] K. Spafford, J. Meredith, J. Vetter, J. Chen, R. Grout, and R. Sankaran, *Accelerating S3D: a GPGPU case study*, Euro-Par 2009: parallel processing workshops (Delft, Netherlands, 2009), Lecture Notes in Computer Science, no. 6043, Springer, Berlin, 2010, pp. 122–131.

[38] V. Sridharan and D. Liberty, *A study of DRAM failures in the field*, SC '12: International Conference on High Performance Computing, Networking, Storage and Analysis (Salt Lake City, 2012), IEEE, Los Alamitos, CA, 2012.

[39] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi, *Feng Shui of supercomputer memory positional effects in DRAM and SRAM faults*, SC '13: International Conference on High Performance Computing, Networking, Storage and Analysis (Denver, 2013), IEEE, Los Alamitos, CA, 2013.

[40] M. Stoyanov and C. Webster, *Numerical analysis of fixed point algorithms in the presence of hardware faults*, SIAM J. Sci. Comput. **37** (2015), no. 5, C532–C553.

[41] J. Wei, A. Thomas, G. Li, and K. Pattabiraman, *Quantifying the accuracy of high-level fault injection techniques for hardware faults*, 44th annual IEEE/IFIP International Conference on Dependable Systems and Networks (Atlanta, 2014), IEEE, Los Alamitos, CA, 2014, pp. 375–382.

[42] C. S. Yoo, R. Sankaran, and J. H. Chen, *Three-dimensional direct numerical simulation of a turbulent lifted hydrogen jet flame in heated coflow: flame stabilization and structure*, J. Fluid Mech. **640** (2009), 453–481.

Received March 3, 2016. Revised September 9, 2016.

RAY W. GROUT: ray.grout@nrel.gov
Computational Science Center, National Renewable Energy Laboratory, 15013 Denver West Parkway, Golden, CO 80401, United States

HEMANTH KOLLA: hnkolla@sandia.gov
Sandia National Laboratories, P.O. Box 969, M.S. 9158, 7011 East Ave, Livermore, CA 94551-0969, United States

MICHAEL L. MINION: mlminion@lbl.gov
Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, MS 50A-3141, 1 Cyclotron Road, Berkeley, CA 94720, United States

JOHN B. BELL: jbbell@lbl.gov
Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, MS 50A-3141, 1 Cyclotron Road, Berkeley, CA 94720, United States

COMM. APP. MATH. AND COMP. SCI.
Vol. 12, No. 1, 2017
dx.doi.org/10.2140/camcos.2017.12.51

msp

A FOURTH-ORDER CARTESIAN GRID EMBEDDED BOUNDARY METHOD FOR POISSON'S EQUATION

DHARSHI DEVENDRAN, DANIEL T. GRAVES,
HANS JOHANSEN AND TERRY LIGOCKI

In this paper, we present a fourth-order algorithm to solve Poisson's equation in two and three dimensions. We use a Cartesian grid, embedded boundary method to resolve complex boundaries. We use a weighted least squares algorithm to solve for our stencils. We use convergence tests to demonstrate accuracy and we show the eigenvalues of the operator to demonstrate stability. We compare accuracy and performance with an established second-order algorithm. We also discuss in depth strategies for retaining higher-order accuracy in the presence of nonsmooth geometries.

1. Introduction

There are many numerical approaches to solve Poisson's equation in complex geometries. Green function approaches [26; 16; 8], such as the fast multipole method, are fast and near-optimal in complexity, but they are not conservative. Also, they cannot be easily extended to variable and tensor coefficient Poisson operators, which are important in the earth sciences and multimaterial problems.

Another popular approach is to use the finite element method, which has a number of advantages. These advantages include negative-definite discrete operators, higher-order accuracy, and ease of extension to variable coefficients. The conditioning and accuracy of the discrete finite element operator can be strongly mesh-dependent, however [6]. Unfortunately, generating meshes with higher-order conforming elements for complex three-dimensional domains is still an expensive, globally coupled computation, and an open area of research [30].

This motivates the need for simpler grid generation. Cut cells are a simple way of addressing this. In a cut cell (or embedded boundary) method, the discrete domain is the intersection of the complex geometry with a regular Cartesian grid. Such

Research at LBNL was supported financially by the Office of Advanced Scientific Computing Research of the US Department of Energy under contract number DE-AC02-05CH11231.
MSC2010: 65M08, 65M50.
Keywords: Poisson equation, finite volume methods, high order, embedded boundary.

intersections are local, and can be calculated very efficiently in parallel, enabling fast computation of solution-dependent moving boundaries [1; 33; 27]. Cut cells have been used successfully to solve Poisson's equation in finite volume [19; 32] and finite difference [14; 24] discretizations.

For many problems, such as heat and mass transfer, discrete conservation is important. Finite volume methods are discretely conservative by construction because they are in discrete flux-divergence form [22]. Previous finite volume methods for Poisson's equation are first order in truncation error near the embedded boundary and second order in solution error [19; 32]. Our finite volume discretization of Poisson's equation is third order in truncation error and fourth order in solution error. The discretization is in flux-divergence form and therefore strongly conservative.

The second-order, finite volume, strongly conservative Schwartz et al. algorithm [32] has been used in many larger applications, including incompressible Navier–Stokes with moving boundaries [27], compressible Navier–Stokes [15] and a DNA-transport application [37]. We compare our algorithm to the Schwartz et al. algorithm by comparing both eigenvalue spectrums and the number of degrees of freedom that are required to achieve a given degree of accuracy.

Realistic boundaries can have discontinuities in their derivatives. For example, it is common to make a geometric description out of the intersection of several simpler geometries. We show that these discontinuities can have profound effects upon accuracy. We provide a strategy for maintaining higher-order accuracy in the presence of geometric discontinuities using geometric regularization with a smoothing length which can be controlled. We show that the rate at which this smoothing length converges with grid refinement matters greatly.

2. Algorithm

The algorithm is described in several steps. First, we introduce the embedded boundary finite volume discretization for Poisson's equation. Then we obtain a Taylor-series-based interpolant of the solution and operator that is compatible with cell- and face-averaged quantities, and achieves the desired order of accuracy. Lastly, we introduce a weighted least-squares approach that uses nearest neighbor values to stably interpolate the quantities needed by the finite volume operator.

2.1. ***Finite volume discretization.*** Given a charge density ρ, Poisson's equation for the potential ϕ can be written as

$$\nabla \cdot (\nabla \phi) = \rho. \tag{1}$$

Integrating this over a control volume V and applying the divergence theorem yields

$$\int_{\partial V} \nabla \phi \cdot \hat{\boldsymbol{n}}\, dA = \int_V \rho\, dV, \tag{2}$$

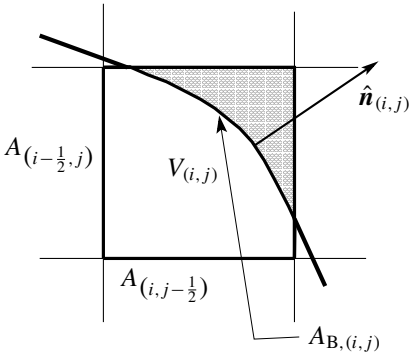


Figure 1. Illustration of cut cell notation. The shaded region is outside the solution domain. The volume $V_{\boldsymbol{i}} = V_{(i,j)}$ is connected to other volumes via the faces aligned with the coordinate planes. The EB face is formed by the intersection of the embedded boundary and the cell.

where $\hat{\boldsymbol{n}}$ is the outward-facing unit normal to the surface.

Our volumes are rectangular control volumes on a Cartesian mesh, cut by an embedded boundary. Formally, the underlying description of space is given by

$$\Upsilon_{\boldsymbol{i}} = \left[(\boldsymbol{i} - \tfrac{1}{2}\boldsymbol{u})h, (\boldsymbol{i} + \tfrac{1}{2}\boldsymbol{u})h\right], \quad \boldsymbol{i} \in \mathbb{Z}^{\mathcal{D}},$$

where $\mathcal{D}$ is the dimensionality of the problem, h is the mesh spacing, and $\boldsymbol{u}$ is the vector whose entries are all one. Note we use bold font $\boldsymbol{u} = (u_1, \ldots, u_d, \ldots, u_{\mathcal{D}})$ to indicate a vector quantity. Given an irregular domain Ω, we obtain control volumes $V_{\boldsymbol{i}} = \Upsilon_{\boldsymbol{i}} \cap \Omega$ and faces $A_{\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}_d}$, which are the intersection of the boundary of $\partial V_{\boldsymbol{i}}$ with the coordinate planes $\left\{\boldsymbol{x} : x_d = \left(i_d \pm \frac{1}{2}\right)h\right\}$ ($\boldsymbol{e}_d$ is the unit vector in the d direction). We also define $A_{\mathrm{B},\boldsymbol{i}}$ to be the intersection of the boundary of the irregular domain with the Cartesian control volume: $A_{\mathrm{B},\boldsymbol{i}} = \partial\Omega \bigcap \Upsilon_{\boldsymbol{i}}$. Figure 1 illustrates a volume cut by an embedded boundary.

We define a flux function to be the gradient of the potential ($\boldsymbol{F} \equiv \nabla\phi$). Given a volume $V_{\boldsymbol{i}}$ we can rewrite the integral form of Poisson's equation (2) as a sum of integrals over each face in the volume,

$$\int_{V_{\boldsymbol{i}}} \nabla \cdot (\nabla\phi)\, dV = \sum_{d=1}^{\mathcal{D}} \left(\int_{A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}} F_d\, dA - \int_{A_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d}} F_d\, dA + \int_{A_{\mathrm{B},\boldsymbol{i}}} F_d \hat{\boldsymbol{n}}_d\, dA \right). \quad (3)$$

We use the following notation to denote the averages of ϕ over a computational volume:

$$\langle\phi\rangle_{\boldsymbol{i}} = \frac{1}{|V_{\boldsymbol{i}}|} \int_{V_{\boldsymbol{i}}} \phi\, dV. \quad (4)$$

The average flux over a coordinate face is defined as

$$\langle F_d \rangle_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d} = \frac{1}{|A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}|} \int\limits_{A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}} F_d \, dA,$$

and the average flux at the irregular face is given by

$$\langle F_d \, \hat{\boldsymbol{n}}_d \rangle_{\mathrm{B},\boldsymbol{i}} = \frac{1}{|A_{\mathrm{B},\boldsymbol{i}}|} \int\limits_{A_{\mathrm{B},\boldsymbol{i}}} F_d \hat{\boldsymbol{n}}_d \, dA.$$

To create a conservative divergence operator, we discretize our divergence operator as a sum of average fluxes. We define the volume fraction κ to be the fraction of the volume of the cell inside the solution domain, so that

$$\kappa = h^{-\mathcal{D}} |V_{\boldsymbol{i}}|. \tag{5}$$

Given a flux function $\boldsymbol{F}$, the κ-weighted divergence of the flux is defined to be the volume average of the divergence multiplied by κ:

$$\begin{aligned} \kappa L(\phi)_{\boldsymbol{i}} &= \kappa \langle \nabla \cdot \boldsymbol{F} \rangle_{\boldsymbol{i}} \\ &= \frac{1}{h^{\mathcal{D}}} \int_{V_i} \nabla \cdot \boldsymbol{F} \, dV \\ &= \frac{1}{h^{\mathcal{D}}} \sum_{d=1}^{\mathcal{D}} \left(|A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}| \langle F_d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} - |A_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d}| \langle F_d \rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d} + |A_{\mathrm{B},\boldsymbol{i}}| \langle F_d \, \hat{\boldsymbol{n}}_d \rangle_{\mathrm{B},\boldsymbol{i}} \right). \end{aligned} \tag{6}$$

We weigh the conservative divergence this way to avoid small-κ numerical instabilities. Implicit algorithms for Poisson's equation (1) solve the discrete system

$$\kappa \langle \nabla \cdot \nabla \phi \rangle_{\boldsymbol{i}} = \kappa \langle \rho \rangle_{\boldsymbol{i}} \tag{7}$$

for ϕ [19; 32], which avoids very large negative eigenvalues from terms with κ^{-1}. Up to this point, no approximations have been made.

The accuracy of the method is dependent only upon the accuracy of the discretization of the average fluxes. Previous conservative algorithms for embedded boundaries compute fluxes that are second order [29; 27; 32; 13; 15; 28; 11]. In those algorithms, the face-averages of $\nabla\phi$ are approximated to second order by pointwise values at the centroids of faces. These fluxes are constructed by pointwise differencing those cell-centered values of ϕ.

2.2. ***Taylor series expansions for average quantities.*** In our discretization, we use the cell-averages of ϕ directly in the local polynomial expansion of ϕ that matches the boundary conditions to some order of accuracy. We use this polynomial expansion to construct a more accurate approximation to the face-averaged flux.

Throughout this paper, we use the following compact "multi-index" notation:

$$(\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{p}}=\prod_{d=1}^{\mathcal{D}}(x_d-\bar{x}_d)^{p_d},\qquad \boldsymbol{p}!=\prod_{d=1}^{\mathcal{D}}p_d!\,.$$

Given a point in space $\bar{\boldsymbol{x}}$, and a $\mathcal{D}$-dimensional integer vector $\boldsymbol{p}$, we define $m_{\boldsymbol{i}}^{\boldsymbol{p}}(\bar{\boldsymbol{x}})$ to be the $\boldsymbol{p}$-th moment of the volume $V_{\boldsymbol{i}}$ relative to the point $\bar{\boldsymbol{x}}$:

$$m_{\boldsymbol{i}}^{\boldsymbol{p}}(\bar{\boldsymbol{x}})=\int_{V_{\boldsymbol{i}}}(\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{p}}\,dV. \tag{8}$$

The volume of the cut cell $V_{\boldsymbol{i}}$ is $|V_{\boldsymbol{i}}|=m_{\boldsymbol{i}}^{\boldsymbol{z}}$, where $\boldsymbol{z}$ is the zero vector. We define the face moments $m_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}^{\boldsymbol{p}}(\bar{\boldsymbol{x}})$ to be the $\boldsymbol{p}$-th moments (relative to the point $\bar{\boldsymbol{x}}$) of the faces $A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}$:

$$m_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}^{\boldsymbol{p}}(\bar{\boldsymbol{x}})=\int_{A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}}(\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{p}}\,dA. \tag{9}$$

We define two moments corresponding to the embedded boundary face $A_{\mathrm{B},\boldsymbol{i}}$,

$$m_{\mathrm{B},\boldsymbol{i}}^{\boldsymbol{p}}(\bar{\boldsymbol{x}})=\int_{A_{\mathrm{B},i}}(\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{p}}\,dA \tag{10}$$

and

$$m_{\mathrm{B},\boldsymbol{i},d}^{\boldsymbol{p}}(\bar{\boldsymbol{x}})=\int_{A_{\mathrm{B},i}}(\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{p}}\,\hat{\boldsymbol{n}}_d(\boldsymbol{x})\,dA, \tag{11}$$

where $\hat{\boldsymbol{n}}_d$ is the d-th component of the outward-facing unit normal to the EB face.

For some integer Q, suppose we want an $O(h^Q)$ approximation to the flux $\boldsymbol{F}=\nabla\phi$. Given a sufficiently smooth function ϕ, we can approximate ϕ in the neighborhood of $\bar{\boldsymbol{x}}$ using a multidimensional Taylor expansion:

$$\phi(\boldsymbol{x})=\sum_{|\boldsymbol{q}|<=Q}\frac{1}{\boldsymbol{q}!}\phi^{(\boldsymbol{q})}(\bar{\boldsymbol{x}})(\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{q}}+O(h^{Q+1}), \tag{12}$$

where we use the multi-index partial derivative notation

$$\phi^{(\boldsymbol{q})}=\partial^{\boldsymbol{q}}\phi=\frac{\partial^{q_1}}{\partial x_1^{q_1}}\cdots\frac{\partial^{q_{\mathcal{D}}}}{\partial x_{\mathcal{D}}^{q_{\mathcal{D}}}}\phi. \tag{13}$$

If we put the expansion (12) into one of the integrals in (3) over a coordinate-aligned face and set $c_q = \frac{1}{\boldsymbol{q}!}\phi^{(\boldsymbol{q})}(\bar{\boldsymbol{x}})$, we get

$$\int_{A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}} \frac{\partial \phi}{\partial x_d}\, dA = \sum_{|\boldsymbol{q}|<=Q} q_d c_{\boldsymbol{q}} \int_{A_{i\pm\frac{1}{2}\boldsymbol{e}_d}} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{q}-\boldsymbol{e}_d}\, dA + O(h^Q) \tag{14}$$

$$= \sum_{|\boldsymbol{q}|<=Q} q_d c_{\boldsymbol{q}} m^{\boldsymbol{q}-\boldsymbol{e}_d}_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}(\bar{\boldsymbol{x}}) + O(h^Q). \tag{15}$$

The flux equation at the irregular boundary becomes

$$\int_{A_{\mathrm{B},\boldsymbol{i}}} \frac{\partial \phi}{\partial x_d}\hat{\boldsymbol{n}}_d\, dA = \sum_{|\boldsymbol{q}|<=Q} q_d c_{\boldsymbol{q}} \int_{A_{\mathrm{B},\boldsymbol{i}}} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{q}-\boldsymbol{e}_d}\hat{\boldsymbol{n}}_d\, dA + O(h^Q) \tag{16}$$

$$= \sum_{|\boldsymbol{q}|<=Q} q_d c_{\boldsymbol{q}} m^{\boldsymbol{q}-\boldsymbol{e}_d}_{\mathrm{B},\boldsymbol{i},d}(\bar{\boldsymbol{x}}) + O(h^Q). \tag{17}$$

All the moments can be generated to any order as shown in Schwartz, et al. [33]. Therefore, generating an $O(h^Q)$ algorithm for Poisson's equation reduces to finding the coefficients $c_{\boldsymbol{q}}$.

2.3. ***Weighted least-squares interpolants.*** We define $\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ to be the set of volumes in the neighborhood of face $\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d$. Our neighborhood algorithm is described in Section 2.3.3. We put the expansion (12) into (4) for every volume $V_{\boldsymbol{j}} \in \mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$:

$$\langle\phi\rangle_{\boldsymbol{j}} = \frac{1}{|V_j|}\sum_{|\boldsymbol{q}|<=Q} c_{\boldsymbol{q}} \int_{V_j} (\boldsymbol{x}-\bar{\boldsymbol{x}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d})^{\boldsymbol{q}}\, dV \tag{18}$$

$$= \frac{1}{|V_j|}\sum_{|\boldsymbol{q}|<=Q} c_{\boldsymbol{q}} m^{\boldsymbol{q}}_{\boldsymbol{j}}(\bar{\boldsymbol{x}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}), \tag{19}$$

where $\bar{\boldsymbol{x}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = h\left(\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d\right)$ is the center of the target face. This forms a system of equations for the coefficients $c_{\boldsymbol{q}}$.

Define C to be a column vector composed of the Taylor coefficients $c_{\boldsymbol{q}}$. In C, the powers of $\boldsymbol{q}$ are listed in lexicographical order. For example, in two dimensions, for $Q = 2$,

$$C = \begin{bmatrix} c^{(0,0)} \\ c^{(1,0)} \\ c^{(2,0)} \\ c^{(0,1)} \\ c^{(1,1)} \\ c^{(0,2)} \end{bmatrix}. \tag{20}$$

Define Φ to be the column vector of all $\langle\phi\rangle_{\boldsymbol{j}}$ such that $V_{\boldsymbol{j}} \in \mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$. Define M to be the matrix of the volume moments of the neighbors normalized by their

volumes. Each row of M corresponds to a particular neighbor. In particular, suppose $\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = \{V_{j_1}, \ldots, V_{j_N}\}$. Then the l-th row of M are the moments for neighbor V_{j_l}. Each column of M corresponds to a particular power $\boldsymbol{q}$. Because the volume of $V_{\boldsymbol{j}}$ is $m^{\boldsymbol{z}}_{\boldsymbol{j}}$, the first column consists of ones.

For example, in two dimensions, with $Q = 2$, the moment matrix M takes the form

$$M = \begin{bmatrix} 1 & \frac{m^{(1,0)}_{j_1}}{m^{(0,0)}_{j_1}} & \frac{m^{(2,0)}_{j_1}}{m^{(0,0)}_{j_1}} & \frac{m^{(0,1)}_{j_1}}{m^{(0,0)}_{j_1}} & \frac{m^{(1,1)}_{j_1}}{m^{(0,0)}_{j_1}} & \frac{m^{(0,2)}_{j_1}}{m^{(0,0)}_{j_1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \frac{m^{(1,0)}_{j_N}}{m^{(0,0)}_{j_N}} & \frac{m^{(2,0)}_{j_N}}{m^{(0,0)}_{j_N}} & \frac{m^{(0,1)}_{j_N}}{m^{(0,0)}_{j_N}} & \frac{m^{(1,1)}_{j_N}}{m^{(0,0)}_{j_N}} & \frac{m^{(0,2)}_{j_N}}{m^{(0,0)}_{j_N}} \end{bmatrix}. \tag{21}$$

Extending both C and M to $Q = 4$ simply requires adding the extra moments in Pascal's triangle in lexicographical order. All moments in the system of equations are centered around the target face at $\bar{\boldsymbol{x}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = h\left(\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}_d\right)$. The system of equations formed by (19) over the neighborhood $\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ takes the form

$$\Phi = MC. \tag{22}$$

Say there are P coefficients we need and N neighbors in $\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$. If $N > P$, we have an over-determined system that we can solve by weighted least squares. We define a weighting matrix W and use it to multiply both sides of our system

$$W\Phi = WMC.$$

The choice of weighting matrix is discussed in Section 2.3.2. Taking the Moore–Penrose pseudoinverse, we solve for the Taylor coefficients

$$C = (WM)^{\dagger} W\Phi,$$

and use these coefficients to compute the flux at the face. Recall from (15) that we need to shift and transform the coefficients to compute the average gradient at the face. Define G to be the row vector $G = [\cdots q_d m^{\boldsymbol{q}-\boldsymbol{e}^d}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} \cdots]$, where $|\boldsymbol{q}| <= Q$. Then, (15) becomes

$$|A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}|\langle F_d\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = GC.$$

We express this flux calculation as a stencil. Because these are all linear operators, we know we can express the flux as a column vector $S_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ acting on the solution, $|A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}|\langle F_d\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = S^T_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}\Phi$, where

$$S_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = G(WM)^{\dagger} W. \tag{23}$$

At every face in the domain, we solve for the stencil $S_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$. For faces near the domain boundaries and the embedded boundary we add boundary equations to the

system (22). This is discussed in Section 2.3.1. We solve for our stencils using the singular value decomposition framework from LAPACK [5].

Putting the flux stencils from (23) into (6), we get the stencil for our operator κL:

$$\kappa L_{\boldsymbol{i}} = \frac{1}{h^{\mathcal{D}}}\left(\sum_{d=1}^{\mathcal{D}}(S_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} - S_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d}) + S_{\boldsymbol{i}}^{\mathrm{EB}}\right), \tag{24}$$

where $S_{\boldsymbol{i}}^{\mathrm{EB}}$, the stencil for the embedded boundary flux, is discussed in Section 2.3.1.

2.3.1. *Boundary conditions.* Boundary conditions for this algorithm are used in two ways. First, we need to calculate the fluxes at the boundary to complete our finite volume discretization (see (24)). Second, including the boundary conditions in the interpolant (22) provides additional rows in the matrix that can compensate for ill-conditioned rows from small cells.

To calculate boundary fluxes, we need different procedures for different types of boundary conditions. For Neumann boundary conditions, the flux is the specified boundary condition. For Dirichlet boundary conditions, on the other hand, we need to compute a stencil to calculate the flux. For Dirichlet domain boundary faces, we solve for the flux stencil as we would for any other face. For Dirichlet boundary conditions on embedded boundary faces, we follow the same procedure except that we use the polynomial expansion from (17), where the derivatives of the normal to the boundary are included.

We add equations that contain boundary condition information to the system (19) used to compute polynomial coefficients. This is done for two reasons: first, it increases the rank of the interpolation matrix if there are small cells. Second, in combination with the weighting matrix described in Section 2.3.2, it "smoothly" incorporates boundary conditions into interior cells that are near the embedded boundary, so the interpolants at nearby faces are more consistent with each other. We have found that this greatly improves the spectrum of the resulting operator.

Specifically, suppose a volume $V_{\boldsymbol{j}}$ in the neighbor set $\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ contains a domain face $\boldsymbol{j}+\frac{1}{2}\boldsymbol{e}_d$ which has a Dirichlet boundary condition $\langle\phi\rangle_{\boldsymbol{j}+\frac{1}{2}\boldsymbol{e}_d} = \phi_{\mathrm{DB}}$. We add the equation

$$\phi_{\mathrm{DB}} = \frac{1}{|A_{\boldsymbol{j}+\frac{1}{2}\boldsymbol{e}_d}|}\int_{A_{\boldsymbol{j}+\frac{1}{2}\boldsymbol{e}_d}} \phi\, dA \tag{25}$$

$$= \frac{1}{|A_{\boldsymbol{j}+\frac{1}{2}\boldsymbol{e}_d}|}\sum_{|\boldsymbol{q}|<=Q} c_{\boldsymbol{q}}\int_{A_{\boldsymbol{j}+\frac{1}{2}\boldsymbol{e}_d}} (\boldsymbol{x}-\bar{\boldsymbol{x}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d})^{\boldsymbol{q}}\, dA \tag{26}$$

$$= \frac{1}{|A_{\boldsymbol{j}+\frac{1}{2}\boldsymbol{e}_d}|}\sum_{|\boldsymbol{q}|<=Q} c_{\boldsymbol{q}} m^{\boldsymbol{q}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}(\bar{\boldsymbol{x}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}) \tag{27}$$

to the system (19). If V_j contains an embedded boundary face with a Dirichlet boundary condition $\langle\phi\rangle_{\boldsymbol{i}}^{\mathrm{EB}} = \phi_{\mathrm{EB}}$ we add the appropriate form of (17):

$$\phi_{\mathrm{EB}} = \frac{1}{|A_{\mathrm{B}_i}|} \sum_{|\boldsymbol{q}|<=Q} c_{\boldsymbol{q}} \int_{A_{\mathrm{B}_i}} (\boldsymbol{x} - \bar{\boldsymbol{x}})^{\boldsymbol{q}} n_d \, dA \tag{28}$$

$$= \frac{1}{|A_{\mathrm{B}_i}|} \sum_{|\boldsymbol{q}|<=Q} c_{\boldsymbol{q}} m^{\boldsymbol{q}}_{\mathrm{B}_i,d}(\bar{\boldsymbol{x}}) \tag{29}$$

to our equation set (19). The extension of this process to Neumann boundary conditions is straightforward.

2.3.2. *Weighting matrix.* Using weighted least squares adds a great deal of flexibility to the least-squares system solver, in that we do not need to carefully choose neighbor sets that are both optimally minimal, and produce a well-conditioned interpolation. As the simplest choice, we use a diagonal weighting matrix W in (23), which amounts to assigning relative importance to the various equations in the system: larger weights mean that equation will more heavily influence the solution to the system [35]. If the volume being weighted is $\boldsymbol{j}$ and the target face is $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}_d$, the weight value $W_{\boldsymbol{j},\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ is chosen to be

$$W_{\boldsymbol{j},\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = (D_{\boldsymbol{j},\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d})^{-5},$$

where $D_{\boldsymbol{j},\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ is the Euclidean distance between the volume center and the target face center. We have found that the choice of weighting function strongly influences the locality of the resulting stencil, and thus the eigenvalues and stability of the resulting operator. Using this weighting, we find that our stencil values in the interior appear to be a perturbation off of a standard second-order stencil, while operator eigenvalues are stable despite small or missing neighboring cells near the embedded boundary.

To understand the effect of the weighting matrix power, we can apply discrete Fourier analysis [23] to the Poisson problem in a one-dimensional periodic domain. For an eigenmode $\phi = e^{i\beta x}$, the exact differential operator is $\partial_{xx}\phi = \lambda\phi$, where $\lambda = -\beta^2$. Our discrete operator based on (23) can be written as:

$$L\Phi = \frac{1}{h}(S_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} - S_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d})\Phi, \quad L\Phi \equiv \sum_{i=-N/2}^{N/2} s_i\phi_i \,. \tag{30}$$

From this it is straightforward to calculate the eigenvalues λ_L of our discrete operator, which we have plotted in Figure 2 for $N = 14$. Note that the weight power p in $W = D^p$ has a dramatic effect for $p \geq -4$, and relatively little effect for $p < -4$. We believe this is because the entries of Vandermonde-like matrix M in (22), which increase like $O(D^4)$, can be counteracted with $W = D^{-5}$ or greater. This allows us

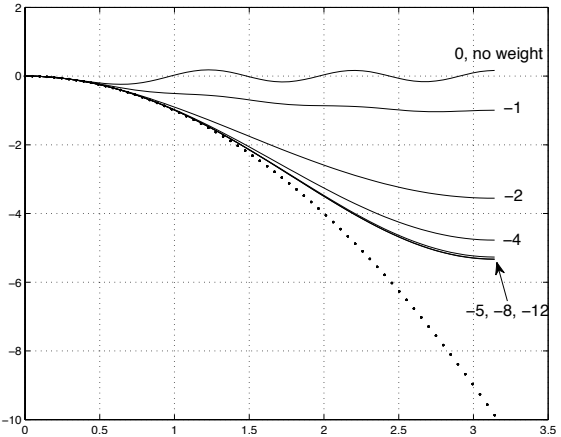


Figure 2. Discrete Fourier analysis of the effect of the weighting matrix power in one dimension, plotting wave number $\beta \in [0, \pi)$ versus operator eigenvalue λ for mode $\phi = e^{i\beta x}$. The dotted line is the exact differential operator, $\lambda = -\beta^2$. The solid lines represent the discrete operator (30) with $N = 14$ points in the stencil, and different weight powers p, for $W = D^p$.

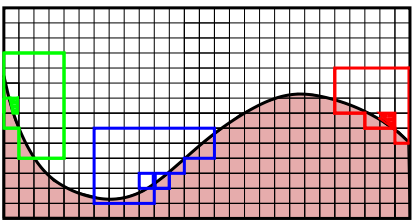


Figure 3. Neighbors of faces cut by the embedded boundary. Geometric constraints can greatly alter the number of neighbors available within a given radius.

to not be concerned with how many neighbors are in our stencil, and maintains the desired accuracy and stability properties of the differential operator. This analysis can be extended to two or more dimensions, and for other operators, which will be the focus of a future paper.

2.3.3. *Neighborhood algorithm.* We define the neighborhood of the face to be the set of valid cells within a discrete radius $R = 3$ cells of either cell of the face:

$$\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = \left\{ \boldsymbol{j} : i_d - j_d < R \text{ or } j_d - (i_d + 1) < R \text{ for any } 1 \le d \le \mathcal{D} \right\}. \tag{31}$$

We use this many cells because we need enough cells in the system (22) so that the system will be over-determined even in the case where the embedded boundary cuts out half of the cells in the neighborhood. Figure 3 illustrates how the number of neighbor volumes can vary due to geometric constraints. We detect if there are not enough cells for any given face and, for that face, we use a larger R.

2.4. ***Multigrid algorithm.*** Once our stencils are calculated, we use the Chombo infrastructure [9; 10], which uses the Martin and Cartwright multigrid algorithm [25], to solve the system. The bottom solver for our multigrid algorithm is the PETSc algebraic multigrid solver [4; 2; 3]. For eigenvalue calculations, we use the SLEPc infrastructure [18; 17; 7]. For details of our adaptive multigrid algorithm, see Devendran et al. [12].

3. Convergence tests

To validate our algorithm, we present convergence tests to show that our algorithm is converging at expected rates. We test both truncation error T and solution error ϵ. We evaluate convergence using the L_1, L_2, and L_∞ norms. Given an error field E^h, whose resolution is h, defined on volumes V_i in Ω, and a norm operator $\|\cdot\|$, the rate of convergence ϖ is defined as

$$\varpi = \log_2\left(\frac{\|E^{2h}\|}{\|E^h\|}\right).$$

Given a computational domain Ω, we define the L_∞ norm of a field to be the maximum value of that field while the L_1 and L_2 norms are integral norms. These take the form

$$\begin{aligned}
\|E\|_\infty &= \max_{i\in\Omega} |E_i|,\\
\|E\|_1 &= \frac{1}{|V_\Omega|}\int_\Omega |E_i|\,dV = \frac{1}{|V_\Omega|}\sum_{i\in\Omega} |E_i||V_i|,\\
\|E\|_2 &= \left(\frac{1}{|V_\Omega|}\int_\Omega |E_i|^2\,dV\right)^{\frac{1}{2}} = \left(\frac{1}{|V_\Omega|}\sum_{i\in\Omega} |E_i|^2|V_i|\right)^{\frac{1}{2}},
\end{aligned}$$

where $|V_\Omega|$ is the volume of the whole domain.

Given a smooth input potential ϕ^e, we compute the truncation error T by comparing the discrete operator L with the exact average Poisson operator L^e:

$$T = \kappa(L(\phi^e) - L^e(\phi^e)), \tag{32}$$

where $\kappa L(\phi)$ is given in (6) and

$$L^e(\phi^e)_i = \int_i \nabla\cdot(\nabla\phi^e)\,dV. \tag{33}$$

We weight the operator this way because the volume fraction κ can be arbitrarily small and because this is the form of the operator that is used in the solution process (see (7)). The solution error ϵ is given by comparing the computed solution ϕ to

$\mathcal{D}$	norm	$\|\epsilon^{2h}\|$	ϖ	$\|\epsilon^{h}\|$
2	L_∞	$1.290 \cdot 10^{-4}$	2.60	$2.130 \cdot 10^{-5}$
2	L_1	$5.336 \cdot 10^{-6}$	3.99	$3.358 \cdot 10^{-7}$
2	L_2	$1.200 \cdot 10^{-5}$	3.55	$1.022 \cdot 10^{-6}$
3	L_∞	$9.222 \cdot 10^{-4}$	3.86	$6.334 \cdot 10^{-5}$
3	L_1	$1.071 \cdot 10^{-5}$	4.00	$6.687 \cdot 10^{-7}$
3	L_2	$2.507 \cdot 10^{-5}$	3.66	$1.984 \cdot 10^{-6}$

Table 1. Truncation error convergence rates with Dirichlet boundary conditions on the embedded boundary and Neumann boundary conditions on the domain. The geometry is the exterior of the ellipse shown in Figure 6, and $h = 1/128$.

the exact solution ϕ^e:

$$\epsilon = \phi - \phi^e. \tag{34}$$

We expect the truncation error to be larger at the embedded boundary since the operator is formally third order in the cut cells. Potential theory tells us that these truncation errors at the boundary should be smoothed out in solution error. We therefore expect solution error to be uniformly fourth order in all norms.

For these tests, we need a smooth geometry and preferably one whose curvature varies. Our computational domain is the unit cube. Given a center point $\boldsymbol{x}_0$, we use the exterior of an ellipse of the form

$$\sum_{d=1}^{\mathcal{D}} \frac{x_d^2 - x_{0,d}^2}{r_d^2} = 0, \tag{35}$$

where $\boldsymbol{r} = (0.25, 0.5, 0.75)$ and $\boldsymbol{x}_0 = (0.5, 0.5, 0.5)$. A picture of this ellipse is given in Figure 6. We generate our geometric moments to $O(h^6)$ so that our results would only reflect the accuracy of our Poisson discretization. In these tests our finest grid spacing is $128^{\mathcal{D}}$ ($h = 1/128$), and the exact potential field is given by

$$\phi^e = \prod_{d=1}^{\mathcal{D}} \cos(\pi x_d). \tag{36}$$

3.1. ***Truncation error.*** In Table 1, we present truncation error rates for the case where the domain has Neumann boundary conditions and the irregular boundary has Dirichlet boundary conditions ($\phi|_{\partial\Omega} = \phi^e$). In Table 2, we present truncation error rates for the case where the domain has Dirichlet boundary conditions and the irregular boundary has Neumann boundary conditions ($\nabla\phi \cdot \hat{\boldsymbol{n}} = \nabla\phi^e \cdot \hat{\boldsymbol{n}}$). For the two examples, we present convergence rates for both two and three dimensions. The third-order truncation error at the embedded boundary dominates the error on the domain, and the L_∞ error reflects this. The truncation error in the L_1 norm, on the other hand, is fourth-order because the embedded boundary only has codimension one.

$\mathcal{D}$	norm	$\|\epsilon^{2h}\|$	ϖ	$\|\epsilon^{h}\|$
2	L_∞	$1.979\cdot10^{-4}$	2.99	$2.485\cdot10^{-5}$
2	L_1	$1.423\cdot10^{-5}$	3.95	$9.184\cdot10^{-7}$
2	L_2	$3.897\cdot10^{-5}$	3.46	$3.530\cdot10^{-6}$
3	L_∞	$4.490\cdot10^{-4}$	2.22	$9.645\cdot10^{-5}$
3	L_1	$2.698\cdot10^{-5}$	3.95	$1.747\cdot10^{-6}$
3	L_2	$6.697\cdot10^{-5}$	3.44	$6.161\cdot10^{-6}$

Table 2. Truncation error convergence rates with Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain. The geometry is the exterior of the ellipse shown in Figure 6, and $h = 1/128$.

$\mathcal{D}$	norm	$\|\epsilon^{2h}\|$	ϖ	$\|\epsilon^{h}\|$
2	L_∞	$1.626\cdot10^{-7}$	3.94	$1.060\cdot10^{-8}$
2	L_1	$8.934\cdot10^{-8}$	3.91	$5.952\cdot10^{-9}$
2	L_2	$1.032\cdot10^{-7}$	3.93	$6.783\cdot10^{-9}$
3	L_∞	$3.060\cdot10^{-7}$	3.97	$1.954\cdot10^{-8}$
3	L_1	$1.955\cdot10^{-7}$	3.95	$1.265\cdot10^{-8}$
3	L_2	$2.154\cdot10^{-7}$	3.96	$1.386\cdot10^{-8}$

Table 3. Solution error convergence rates with Dirichlet boundary conditions on the embedded boundary and Neumann boundary conditions on the domain. The geometry is the exterior of an ellipse and $h = 1/128$.

3.2. ***Solution error.*** In [19; 20], the authors show that a method can have a lower-order truncation error on the embedded boundary (which is a codimension one set) than in the interior and still maintain the proper order for the solution error. We solve $\kappa L\phi = \kappa L(\phi^e)$ and compute the solution error. For this test ϕ^e is given by (36). In Table 3 we present solution error rates for the case where the domain has Neumann boundary conditions and the irregular boundary has Dirichlet boundary conditions ($\phi|_{\partial\Omega} = \phi_e$). We present solution error rates for the case where the domain and the irregular boundary have Neumann boundary conditions ($\nabla\phi\cdot\hat{\boldsymbol{n}} = \nabla\phi^e\cdot\hat{\boldsymbol{n}}$) in Table 4. In both cases, we show uniform fourth-order convergence rates in all norms. We also run this test at much higher resolutions in Section 5.1.

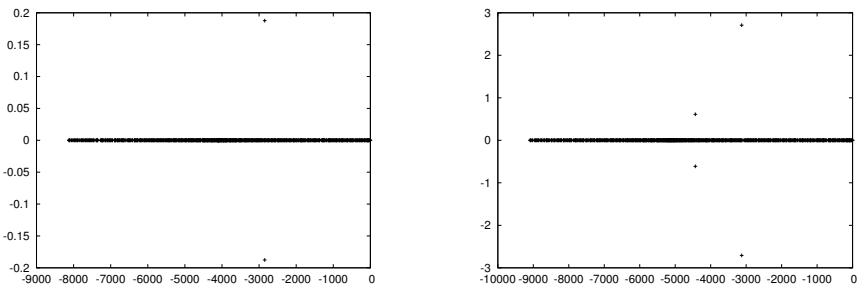
4. Operator eigenvalues

In this section, we compare the spectrum of our algorithm to the widely used, second-order algorithm presented by Schwartz et al. [32].

The eigenvalues of the Poisson operator will depend upon the geometry and resolution of the problem as well as the operator boundary conditions. Due to limitations in computational resources, we are only able to show the spectrum for

$\mathcal{D}$	norm	$\|\epsilon^{2h}\|$	ϖ	$\|\epsilon^{h}\|$
2	L_∞	$1.835 \cdot 10^{-7}$	3.96	$1.176 \cdot 10^{-8}$
2	L_1	$6.904 \cdot 10^{-8}$	3.95	$4.459 \cdot 10^{-9}$
2	L_2	$8.678 \cdot 10^{-8}$	3.96	$5.558 \cdot 10^{-9}$
3	L_∞	$3.879 \cdot 10^{-7}$	3.86	$2.669 \cdot 10^{-8}$
3	L_1	$9.325 \cdot 10^{-8}$	3.92	$6.175 \cdot 10^{-9}$
3	L_2	$1.315 \cdot 10^{-7}$	3.94	$8.559 \cdot 10^{-9}$

Table 4. Solution error convergence rates with Neumann boundary conditions the embedded boundary and Dirichlet boundary conditions on the domain. The geometry is the exterior of an ellipse and $h = 1/128$.



Second-order operator from [32]. Fourth-order operator.

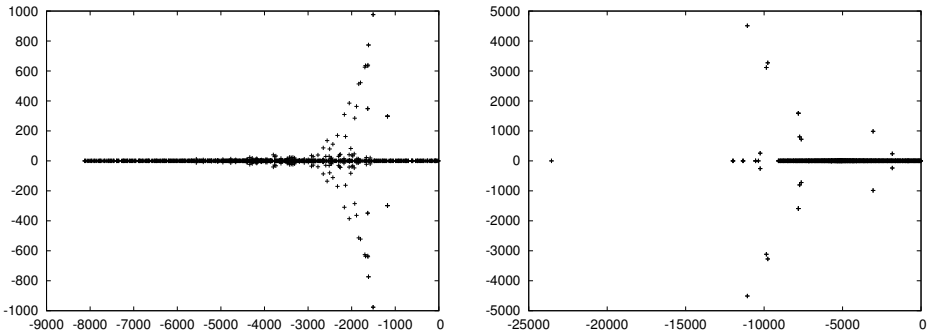
Figure 4. Plots of eigenvalues for the two-dimensional Poisson operators with Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. Real and imaginary parts on the x- and y-axes, respectively, with similar scales for each operator. The geometry implicit function is described by (35) and the resolution is 32^2.

coarse two-dimensional problems (our resolution is 32^2). We use the Krylov–Schur module in SLEPc [18] to compute the eigenvalues.

We present the spectrum for our fourth-order operator with Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain in Figure 4, right, We present the spectrum for the second-order operator with identical conditions in Figure 4, left. We also present the fourth-order spectrum for Dirichlet boundary conditions on the embedded boundary and Neumann boundary conditions on the domain in Figure 5, right. and the second-order spectrum in Figure 5, left. In both cases, Dirichlet boundary conditions on the embedded boundary introduce more complex eigenvalues. In both cases, all the eigenvalues have negative real components and are therefore stable.

5. Effect on accuracy of geometric differentiability

Fundamentally, the appeal of a higher-order method is that one can achieve higher accuracy with fewer degrees of freedom. To reliably achieve this rate of convergence,



Second-order operator from [32]. Fourth-order operator.

Figure 5. Plots of eigenvalues for the two-dimensional Poisson operators with Dirichlet boundary conditions on the embedded boundary and Neumann boundary conditions on the domain boundary. Real and imaginary parts on the x- and y-axes, respectively, with different scales for each operator. The fourth-order operator is very similar to second-order one, but with a few eigenvalues with significantly larger real or imaginary components. The geometry implicit function is described by (35) and the resolution is 32^2.

however, one needs a sufficiently smooth description of the geometry. To achieve $O(h^P)$ accurate fluxes, Schwartz, Percelay et al. [33] show that all geometric moments in the calculation must also converge at $O(h^P)$.

Unfortunately, geometric descriptions are not always sufficiently smooth. This is not necessarily catastrophic. In [20], it is shown that large truncation errors can be ameliorated under certain circumstances; specifically, $O(1)$ truncation errors at a Dirichlet boundary condition will not prevent second-order solution error convergence. Similarly, $O(h)$ truncation errors at a Neumann boundary will not prevent second-order solution error convergence.

These competing effects present a bit of a complex picture. To see how our algorithm fits into this picture, we compare our algorithm to the widely used Schwartz et al. [32] algorithm for Poisson's equation. We compare the two algorithms using both a smooth and a nonsmooth geometric description. These comparisons are done for both Dirichlet and Neumann boundary conditions at the embedded boundary. Because some of the techniques used in this section are resource-intensive, we restrict our comparisons to two dimensions so we can achieve much higher resolutions.

All of these tests are done with an exact potential

$$\phi_e = \prod_{d=1}^{\mathcal{D}} \sin(\pi x_d),$$

and a charge distribution $\rho = \nabla \cdot \nabla \phi_e$. The calculation domain is the unit square and there are Dirichlet boundary conditions on the domain boundary. In all of these results we present both the resolution and the number points in Ω at that

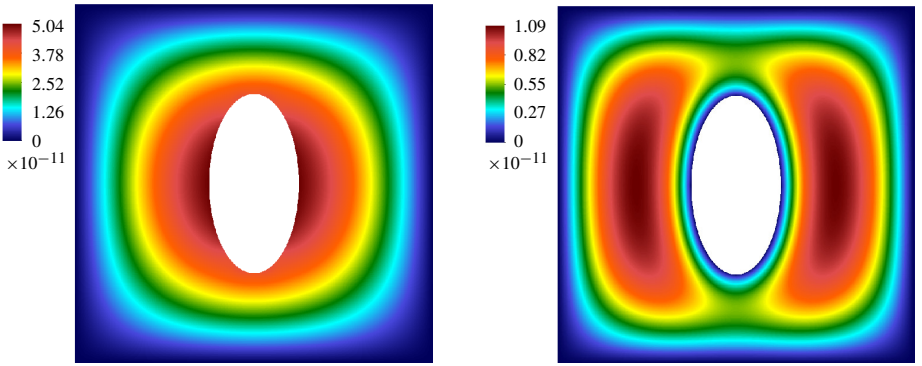


Figure 6. Solution error for the ellipse geometry in two dimensions using the current fourth-order algorithm with a resolution of 512^2. Left, using Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. Right, using Dirichlet boundary conditions both on the embedded and on the domain boundary.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	952	$1.419\cdot10^{-3}$	$3.354\cdot10^{-4}$	$4.144\cdot10^{-4}$	—
Schwartz	64^2	3752	$3.511\cdot10^{-4}$	$8.645\cdot10^{-5}$	$1.070\cdot10^{-4}$	1.95
Schwartz	128^2	14884	$8.639\cdot10^{-5}$	$2.186\cdot10^{-5}$	$2.709\cdot10^{-5}$	1.98
Schwartz	256^2	59312	$2.083\cdot10^{-5}$	$5.443\cdot10^{-6}$	$6.741\cdot10^{-6}$	2.00
Schwartz	512^2	236832	$5.167\cdot10^{-6}$	$1.354\cdot10^{-6}$	$1.677\cdot10^{-6}$	2.00
Schwartz	1024^2	946432	$1.272\cdot10^{-6}$	$3.380\cdot10^{-7}$	$4.185\cdot10^{-7}$	2.00
current	32^2	952	$2.786\cdot10^{-6}$	$1.041\cdot10^{-6}$	$1.316\cdot10^{-6}$	—
current	64^2	3752	$1.833\cdot10^{-7}$	$6.897\cdot10^{-8}$	$8.670\cdot10^{-8}$	3.91
current	128^2	14884	$1.176\cdot10^{-8}$	$4.459\cdot10^{-9}$	$5.557\cdot10^{-9}$	3.95
current	256^2	59312	$7.431\cdot10^{-10}$	$2.833\cdot10^{-10}$	$3.512\cdot10^{-10}$	3.97
current	512^2	236832	$5.045\cdot10^{-11}$	$1.941\cdot10^{-11}$	$2.396\cdot10^{-11}$	3.86

Table 5. Error vs. refinement comparison with the second-order Schwartz et al. algorithm with the elliptical geometry. This uses Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. The convergence rates ϖ are calculated using L_1.

resolution This number of points represents the number of degrees of freedom in the calculation.

5.1. ***Accuracy vs. resolution for a smooth geometry.*** First we compare our algorithm to the Schwartz et al. algorithm with a smooth geometry. Here, our geometry is the exterior of the ellipse whose implicit function is described by (35). Figure 6, left, shows a solution error plot with Neumann boundary conditions at the embedded boundary. Figure 6, right, shows a solution error plot with Dirichlet boundary conditions. Both cases show that the solution error is distributed throughout the domain.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	952	$5.415\cdot10^{-3}$	$1.322\cdot10^{-3}$	$1.715\cdot10^{-3}$	—
Schwartz	64^2	3752	$1.590\cdot10^{-3}$	$3.592\cdot10^{-4}$	$4.665\cdot10^{-4}$	1.87
Schwartz	128^2	14884	$4.581\cdot10^{-4}$	$9.569\cdot10^{-5}$	$1.243\cdot10^{-4}$	1.90
Schwartz	256^2	59312	$1.259\cdot10^{-4}$	$2.498\cdot10^{-5}$	$3.247\cdot10^{-5}$	1.93
Schwartz	512^2	236832	$3.430\cdot10^{-5}$	$6.449\cdot10^{-6}$	$8.381\cdot10^{-6}$	1.95
Schwartz	1024^2	946432	$9.289\cdot10^{-6}$	$1.647\cdot10^{-6}$	$2.142\cdot10^{-6}$	1.96
current	32^2	952	$7.190\cdot10^{-7}$	$3.169\cdot10^{-7}$	$3.841\cdot10^{-7}$	—
current	64^2	3752	$4.146\cdot10^{-8}$	$1.907\cdot10^{-8}$	$2.300\cdot10^{-8}$	4.05
current	128^2	14884	$2.527\cdot10^{-9}$	$1.173\cdot10^{-9}$	$1.400\cdot10^{-9}$	4.02
current	256^2	59312	$1.564\cdot10^{-10}$	$7.370\cdot10^{-11}$	$8.717\cdot10^{-11}$	3.99
current	512^2	236832	$1.093\cdot10^{-11}$	$5.195\cdot10^{-12}$	$6.109\cdot10^{-12}$	3.82

Table 6. Error vs. refinement comparison with the second-order Schwartz et al. algorithm with the elliptical geometry. This uses Dirichlet boundary conditions everywhere. The convergence rates ϖ are calculated using L_1.

Tables 5 and 6 show norms of our solution error at many resolutions for Neumann and Dirichlet boundary conditions, respectively, for both algorithms. For both Neumann and Dirichlet boundary conditions, we show the expected convergence rate of 4 for both Neumann and Dirichlet boundary conditions at the irregular faces.

We also get much smaller errors even with greatly reduced resolution. For example, in the Neumann case, we get an order of magnitude smaller errors at 64^2 (less than one thousand degrees of freedom) than Schwartz et al. get at 1024^2 (over one million degrees of freedom). We expect a different cross-over point depending on both resolution and the complexity of the boundary. Although our approach requires significantly more computation and memory, both due to setup (SVD-based solvers) and solution (using larger stencils), this impact is on a smaller-dimension domain (codimension 2 and 1 when $\mathcal{D}$ is 3 and 2, respectively) only near the embedded boundary. For a given size problem, there is likely a cross-over point where our algorithm delivers the same accuracy with many fewer total points.

5.2. ***Accuracy vs. resolution for a nonsmooth geometry.*** Now we compare our algorithm to the Schwartz et al. algorithm with a geometry that is only piecewise smooth. The geometry is given by the exterior of four or circles as shown in Figure 7. The implicit function is C^1 discontinuous. Figure 8, left, shows a solution error plot with Neumann boundary conditions. Figure 8, right, shows a solution error plot with Dirichlet boundary conditions. In both cases, the solution error is concentrated near the discontinuities in the geometry; in the Dirichlet case, it is concentrated in a very small area.

For Neumann boundary conditions at the embedded boundary, Tables 7 and 8 compare our solution errors with the Schwartz et al. algorithm for Dirichlet boundary

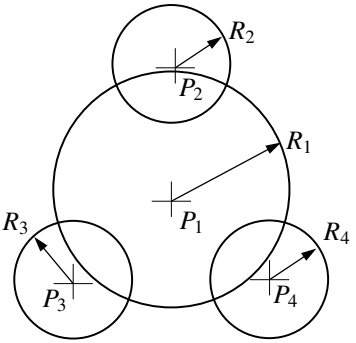


Figure 7. Diagram for our four circle geometry. The computational geometry is the region not covered by these four circles in the unit square, with centers $P_1 = (0.5, 0.5, 0.5)$, $P_2 = (0.5, 0.735, 0.5)$, $P_3 = (0.2965, 0.3825, 0.5)$, $P_4 = (0.7035, 0.3825, 0.5)$, and radii $R_1 = 0.2$, $R_2 = R_3 = R_4 = 0.1$.

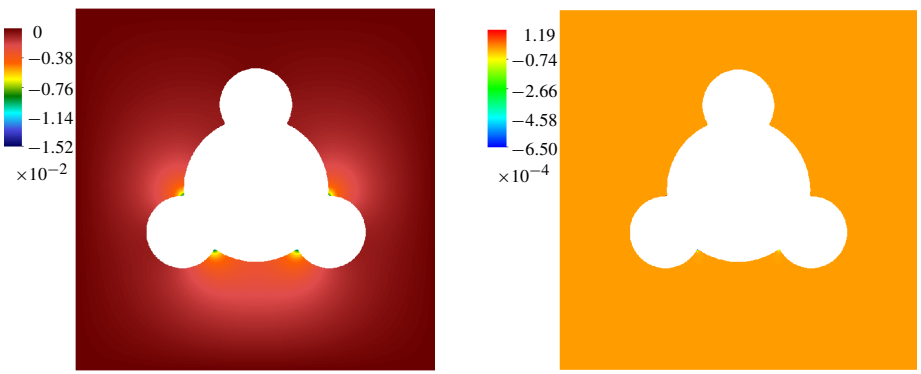


Figure 8. Solution error for the four circle geometry in two dimensions using the current fourth-order algorithm with a resolution of 512^2. Left, using Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. Right, using Dirichlet boundary conditions both on the embedded and on the domain boundary; the error is concentrated very near the cusps in the geometry.

conditions. The errors for the two algorithms are comparable for Neumann boundary conditions, though the higher-order algorithm does show better results with Dirichlet boundary conditions. For Neumann boundary conditions on the irregular faces, we do not even converge at second order. For Dirichlet boundary conditions, we show better convergence but it is generally less than fourth order.

5.3. ***Singular solutions and error characteristics.*** One might be tempted to ascribe this loss in accuracy to a poor approximation of geometric moments. After all, the implicit function from which the moments are generated is not smooth near the corner. To test this theory, we use the refinement algorithm described in [33] to refine the cells near circle intersections by a factor of 1024^2 in each direction. Since we

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	862	$3.525\cdot10^{-2}$	$3.006\cdot10^{-3}$	$4.625\cdot10^{-3}$	—
Schwartz	64^2	3370	$2.231\cdot10^{-2}$	$9.703\cdot10^{-4}$	$1.585\cdot10^{-3}$	1.63
Schwartz	128^2	13318	$1.291\cdot10^{-2}$	$5.649\cdot10^{-4}$	$1.134\cdot10^{-3}$	0.78
Schwartz	256^2	52930	$1.776\cdot10^{-3}$	$8.885\cdot10^{-5}$	$1.325\cdot10^{-4}$	2.66
Schwartz	512^2	211136	$3.576\cdot10^{-3}$	$1.403\cdot10^{-4}$	$2.192\cdot10^{-4}$	-0.66
Schwartz	1024^2	843316	$3.033\cdot10^{-3}$	$1.417\cdot10^{-4}$	$2.089\cdot10^{-4}$	-0.01
current	32^2	862	$5.751\cdot10^{-2}$	$4.869\cdot10^{-3}$	$7.208\cdot10^{-3}$	—
current	64^2	3370	$3.096\cdot10^{-2}$	$1.420\cdot10^{-3}$	$2.721\cdot10^{-3}$	1.77
current	128^2	13318	$2.817\cdot10^{-2}$	$2.132\cdot10^{-3}$	$3.204\cdot10^{-3}$	-0.58
current	256^2	52930	$1.969\cdot10^{-2}$	$1.383\cdot10^{-3}$	$2.020\cdot10^{-3}$	0.62
current	512^2	211136	$1.517\cdot10^{-2}$	$6.754\cdot10^{-4}$	$1.042\cdot10^{-3}$	1.03

Table 7. Error vs. refinement comparison with the second-order Schwartz et al. algorithm for the four circle geometry. This uses Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. The convergence rates ϖ are calculated using L_1.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	862	$2.191\cdot10^{-2}$	$1.333\cdot10^{-3}$	$1.750\cdot10^{-3}$	—
Schwartz	64^2	3370	$1.026\cdot10^{-2}$	$3.717\cdot10^{-4}$	$4.908\cdot10^{-4}$	1.84
Schwartz	128^2	13318	$2.850\cdot10^{-3}$	$9.703\cdot10^{-5}$	$1.300\cdot10^{-4}$	1.93
Schwartz	256^2	52930	$5.719\cdot10^{-4}$	$2.654\cdot10^{-5}$	$3.491\cdot10^{-5}$	1.87
Schwartz	512^2	211136	$8.178\cdot10^{-4}$	$6.686\cdot10^{-6}$	$9.117\cdot10^{-6}$	1.98
Schwartz	1024^2	843316	$8.979\cdot10^{-4}$	$1.620\cdot10^{-6}$	$2.330\cdot10^{-6}$	2.04
current	32^2	862	$1.818\cdot10^{-2}$	$1.604\cdot10^{-4}$	$5.435\cdot10^{-4}$	—
current	64^2	3370	$2.797\cdot10^{-3}$	$3.228\cdot10^{-5}$	$1.089\cdot10^{-4}$	2.31
current	128^2	13318	$2.317\cdot10^{-2}$	$4.557\cdot10^{-6}$	$7.391\cdot10^{-5}$	2.82
current	256^2	52930	$2.705\cdot10^{-3}$	$2.014\cdot10^{-7}$	$4.966\cdot10^{-6}$	4.49
current	512^2	211136	$6.502\cdot10^{-4}$	$5.940\cdot10^{-8}$	$2.263\cdot10^{-6}$	1.76

Table 8. Error vs. refinement comparison with the second-order Schwartz et al. algorithm with the four circle geometry. This uses Dirichlet boundary conditions everywhere.The convergence rates ϖ are calculated using L_1.

know the geometric moments of the uncut subcells exactly and only one subcell contains the discontinuity, this increases the accuracy of the geometric moments dramatically. When we run this test, the solution errors do not change. The reason that our accuracy degrades for the four circle geometry is that the solution to the error equation is singular at these points. With homogeneous boundary conditions, the solution of the Poisson equation is singular near corners whose angle is greater than $\pi/2$ [21].

Given a truncation error T (defined in (32)) and the solution error ϵ (defined in (34)), the error equation can be written

$$L(\epsilon) = T. \tag{37}$$

The boundary conditions for ϵ are homogeneous analogs of the boundary conditions for ϕ (if ϕ's boundary conditions are inhomogeneous Dirichlet, ϵ's boundary conditions are homogeneous Dirichlet). Because our equation is linear, we can separate the truncation error into two parts. We define T^{s} to be the truncation error in cells within the stencil width w of the singular points. We define the nonsingular component of the truncation error to be $T^{\mathrm{ns}} = T - T^{\mathrm{s}}$. We then compute the convergence rate of the solution error ϵ^{ns} in the absence of the singular points of the truncation error by solving

$$L\epsilon^{\mathrm{ns}} = T^{\mathrm{ns}} \tag{38}$$

with the appropriate homogeneous boundary conditions.

We are given a set of M circles $\{C_1, \ldots, C_M\}$, which intersect at the set of volumes $\mathcal{V}^{\mathrm{s}} = \{P_1, \ldots, P_M\}$. The singular part of the truncation error T^{s} is given by

$$T^{\mathrm{s}}_{\boldsymbol{v}} = \begin{cases} T & \text{if } \boldsymbol{v} \in \mathcal{V}^{\mathrm{s}}, \\ 0 & \text{otherwise.} \end{cases} \tag{39}$$

We solve (38) using both the current fourth-order algorithm and the second-order Schwartz et al. algorithm. The comparisons with the Schwartz et al. algorithm are given in Tables 9 and 10. Again, we show that the current algorithm has comparable errors at 32^2 resolution to the Schwartz et al. algorithm at 1024^2 resolution. Once we remove the singular part of the truncation error, we once again show consistent fourth-order accuracy with both Dirichlet and Neumann boundary conditions on the irregular faces.

6. Geometric regularization and accuracy

We recognize that the technique of removing the singular parts of the truncation error is not generally useful to larger applications. The tests presented in Section 5.3 are predicated upon knowing a priori the singular points and the exact solution. For high-order methods to be more generally useful, they must produce much better accuracy than lower-order methods in the presence of geometric discontinuities without this prior knowledge. In this section, we present a method to smooth the geometric description over a controlled length scale. We then show that, if one is careful about how this length scale converges with grid refinement, she can retain superior accuracy compared to lower-order methods even when the input implicit function is only C^0.

algorithm	resolution	# points	$L_\infty(\epsilon^{\text{ns}})$	$L_1(\epsilon^{\text{ns}})$	$L_2(\epsilon^{\text{ns}})$	ϖ
Schwartz	32^2	862	$1.329\cdot10^{-3}$	$3.557\cdot10^{-4}$	$4.370\cdot10^{-4}$	—
Schwartz	64^2	3370	$2.881\cdot10^{-4}$	$8.740\cdot10^{-5}$	$1.073\cdot10^{-4}$	2.02
Schwartz	128^2	13320	$7.068\cdot10^{-5}$	$2.084\cdot10^{-5}$	$2.552\cdot10^{-5}$	2.06
Schwartz	256^2	52930	$1.777\cdot10^{-5}$	$5.171\cdot10^{-6}$	$6.327\cdot10^{-6}$	2.01
Schwartz	512^2	211136	$4.382\cdot10^{-6}$	$1.278\cdot10^{-6}$	$1.564\cdot10^{-6}$	2.01
Schwartz	1024^2	843316	$1.033\cdot10^{-6}$	$3.222\cdot10^{-7}$	$3.946\cdot10^{-7}$	1.98
current	32^2	862	$2.353\cdot10^{-6}$	$7.242\cdot10^{-7}$	$8.939\cdot10^{-7}$	—
current	64^2	3370	$1.864\cdot10^{-7}$	$5.838\cdot10^{-8}$	$7.354\cdot10^{-8}$	3.63
current	128^2	13318	$1.133\cdot10^{-8}$	$3.783\cdot10^{-9}$	$4.735\cdot10^{-9}$	3.94
current	256^2	52930	$7.215\cdot10^{-10}$	$2.405\cdot10^{-10}$	$2.993\cdot10^{-10}$	3.97
current	512^2	211136	$5.392\cdot10^{-11}$	$1.649\cdot10^{-11}$	$2.046\cdot10^{-11}$	3.86

Table 9. Convergence of the nonsingular part of the solution error vs. refinement for the current algorithm and for the Schwartz et al. algorithm. This uses Dirichlet boundary conditions on the domain boundary and Neumann boundary conditions on the embedded boundary.The convergence rates ϖ are calculated using L_1.

algorithm	resolution	# points	$L_\infty(\epsilon^{\text{ns}})$	$L_1(\epsilon^{\text{ns}})$	$L_2(\epsilon^{\text{ns}})$	ϖ
Schwartz	32^2	862	$6.121\cdot10^{-3}$	$1.440\cdot10^{-3}$	$1.875\cdot10^{-3}$	—
Schwartz	64^2	3370	$1.552\cdot10^{-3}$	$3.911\cdot10^{-4}$	$5.066\cdot10^{-4}$	1.88
Schwartz	128^2	13320	$4.119\cdot10^{-4}$	$1.002\cdot10^{-4}$	$1.321\cdot10^{-4}$	1.96
Schwartz	256^2	52930	$1.355\cdot10^{-4}$	$2.669\cdot10^{-5}$	$3.515\cdot10^{-5}$	1.90
Schwartz	512^2	211136	$3.215\cdot10^{-5}$	$6.797\cdot10^{-6}$	$8.913\cdot10^{-6}$	1.97
Schwartz	1024^2	843316	$9.172\cdot10^{-6}$	$1.640\cdot10^{-6}$	$2.152\cdot10^{-6}$	2.05
current	32^2	862	$5.126\cdot10^{-7}$	$1.741\cdot10^{-7}$	$2.236\cdot10^{-7}$	—
current	64^2	3370	$2.684\cdot10^{-8}$	$1.080\cdot10^{-8}$	$1.338\cdot10^{-8}$	4.01
current	128^2	13318	$1.586\cdot10^{-9}$	$6.380\cdot10^{-10}$	$7.764\cdot10^{-10}$	4.08
current	256^2	52930	$9.650\cdot10^{-11}$	$3.882\cdot10^{-11}$	$4.683\cdot10^{-11}$	4.03
current	512^2	211136	$6.676\cdot10^{-12}$	$2.693\cdot10^{-12}$	$3.232\cdot10^{-12}$	3.84

Table 10. Convergence of the nonsingular part of the solution error vs. refinement for both the current algorithm and the Schwartz et al. algorithm. This uses Dirichlet boundary conditions everywhere.The convergence rates ϖ are calculated using L_1.

6.1. ***Smoothing the geometric description.*** Recall that, to generate our geometric moments using the algorithm described in [33], we must start with an implicit function $I(\boldsymbol{x})$ whose zero surface (or contour, in two dimensions) forms the embedded boundary. Consider the geometry described in Figure 7. The implicit function for each circle C_i, with radius r_i and center $\boldsymbol{y}_i$ is given by

$$C_i(\boldsymbol{x}) = r_i^2 - \sum_{d=1}^{\mathcal{D}} (x_d - y_{i,d})^2.$$

The overall implicit function at any point is given by taking the maximum of the four functions:

$$I(\boldsymbol{x}) = \max_{1<=d<=4} C_i(\boldsymbol{x}). \tag{40}$$

Since our geometry is smooth away from specific intersection locations, we wish to only smooth within a length scale δ from the intersections of implicit function zero surfaces. To smooth this description we could use a mollifying function and integrate the convolution directly as in [36]. This has the advantage that the length scale over which the smoothing happens is well defined. These functions can be delicate, however, to integrate numerically. Shapiro [34] presents an alternative approach called R-functions (named for V. L. Rvačev, the originator of the concept [31]), in which logical functions such as maxima, minima and absolute values are replaced by differentiable functions. Though this method is far more numerically tractable, the functions that Shapiro presents do not have a well-defined length scale over which they smooth. The smoothing method described here provides both a well-defined smoothing length and is numerically tractable.

One way to write the maxima function used in (40) is by using an absolute value:

$$\max(a, b) = \tfrac{1}{2}(a + b + |a - b|).$$

Let us define a function $\max_\delta$ which smooths the function max over a length scale δ,

$$\max_\delta(a, b) = \tfrac{1}{2}(a + b + A_\delta(a - b)),$$

where A_δ is the convolution of the absolute value function with a sufficiently smooth function $\psi_\delta(x)$ with compact support in contained within $x \in [-\delta, \delta]$:

$$A_\delta(x) = \int_{-\infty}^{\infty} \psi_\delta(x - y)|y|\,dy = \int_{0}^{\infty} \psi_\delta(x - y)y\,dy - \int_{-\infty}^{0} \psi_\delta(x - y)y\,dy.$$

Since our algorithm is fourth order in fluxes, we use geometric moments to fourth order. The algorithm in [33] requires that the implicit function have derivatives to fourth order. This implies that the mollifier ψ_δ needs to be C^4 and these derivatives must also have compact support. We also require

$$\int_{-\infty}^{\infty} \psi(y)\,dy = 1.$$

Our choice of ψ_δ is

$$\psi_\delta(x) = \begin{cases} \dfrac{4}{3\delta}\cos^4\left(\dfrac{\pi x}{2\delta}\right) & \text{if } -\delta \le x \le \delta, \\ 0 & \text{otherwise.} \end{cases}$$

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	862	$1.989\cdot10^{-2}$	$2.026\cdot10^{-3}$	$3.003\cdot10^{-3}$	—
Schwartz	64^2	3368	$2.593\cdot10^{-3}$	$2.686\cdot10^{-4}$	$3.564\cdot10^{-4}$	2.91
Schwartz	128^2	13316	$1.171\cdot10^{-3}$	$1.207\cdot10^{-4}$	$1.657\cdot10^{-4}$	1.15
Schwartz	256^2	52916	$2.522\cdot10^{-4}$	$3.012\cdot10^{-5}$	$4.093\cdot10^{-5}$	2.00
Schwartz	512^2	211062	$6.144\cdot10^{-5}$	$7.472\cdot10^{-6}$	$1.014\cdot10^{-5}$	2.01
Schwartz	1024^2	843004	$1.417\cdot10^{-5}$	$1.798\cdot10^{-6}$	$2.436\cdot10^{-6}$	2.05
current	32^2	862	$1.525\cdot10^{-1}$	$1.166\cdot10^{-2}$	$1.905\cdot10^{-2}$	—
current	64^2	3368	$1.739\cdot10^{-3}$	$5.812\cdot10^{-5}$	$1.102\cdot10^{-4}$	7.64
current	128^2	13316	$7.054\cdot10^{-5}$	$3.077\cdot10^{-6}$	$5.886\cdot10^{-6}$	4.23
current	256^2	52916	$3.593\cdot10^{-6}$	$4.053\cdot10^{-8}$	$8.884\cdot10^{-8}$	6.24
current	512^2	211062	$1.425\cdot10^{-7}$	$9.227\cdot10^{-9}$	$1.473\cdot10^{-8}$	2.13

Table 11. Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz et al. algorithm. The boundary conditions are Dirichlet on the domain boundary and Neumann on the embedded boundary. Here we set the geometric regularization length to a constant $\delta = 0.01$. The convergence rates ϖ are calculated using L_1.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	862	$1.184\cdot10^{-2}$	$1.869\cdot10^{-3}$	$2.522\cdot10^{-3}$	—
Schwartz	64^2	3368	$1.715\cdot10^{-3}$	$4.142\cdot10^{-4}$	$5.414\cdot10^{-4}$	2.17
Schwartz	128^2	13316	$5.922\cdot10^{-4}$	$1.023\cdot10^{-4}$	$1.356\cdot10^{-4}$	2.01
Schwartz	256^2	52916	$1.355\cdot10^{-4}$	$2.676\cdot10^{-5}$	$3.527\cdot10^{-5}$	1.93
Schwartz	512^2	211062	$3.215\cdot10^{-5}$	$6.784\cdot10^{-6}$	$8.892\cdot10^{-6}$	1.97
Schwartz	1024^2	843004	$8.063\cdot10^{-6}$	$1.644\cdot10^{-6}$	$2.159\cdot10^{-6}$	2.04
current	32^2	862	$7.904\cdot10^{-3}$	$4.768\cdot10^{-5}$	$2.992\cdot10^{-4}$	—
current	64^2	3368	$9.380\cdot10^{-5}$	$1.418\cdot10^{-6}$	$4.421\cdot10^{-6}$	5.07
current	128^2	13316	$2.921\cdot10^{-6}$	$9.434\cdot10^{-9}$	$5.098\cdot10^{-8}$	7.23
current	256^2	52916	$2.745\cdot10^{-7}$	$2.839\cdot10^{-10}$	$2.146\cdot10^{-9}$	5.05
current	512^2	211062	$3.223\cdot10^{-9}$	$3.365\cdot10^{-12}$	$3.125\cdot10^{-11}$	6.39

Table 12. Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz et al. algorithm. The boundary conditions are Dirichlet everywhere. Here we set the geometric regularization length to a constant $\delta = 0.01$. The convergence rates ϖ are calculated using L_1.

which fulfills these requirements. We need to integrate only where the mollifier is nonzero. If a and b are signed distance functions, then δ is the length scale over which $A_\delta(a, b)$ represents a smoothing of the absolute value function.

6.2. ***Regularization length scale and grid refinement.*** Now we investigate how one picks the length scale δ. For the piecewise-smooth geometric description presented in Section 5.2, we present four different choices for δ and see how the accuracy changes with grid refinement. First we use a constant $\delta = 0.01$. Second,

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	828	$6.864\cdot10^{-3}$	$1.503\cdot10^{-3}$	$1.960\cdot10^{-3}$	—
Schwartz	64^2	3238	$1.628\cdot10^{-3}$	$3.894\cdot10^{-4}$	$5.070\cdot10^{-4}$	1.94
Schwartz	128^2	12810	$5.351\cdot10^{-4}$	$1.014\cdot10^{-4}$	$1.345\cdot10^{-4}$	1.94
Schwartz	256^2	50918	$1.404\cdot10^{-4}$	$2.669\cdot10^{-5}$	$3.516\cdot10^{-5}$	1.92
Schwartz	512^2	203052	$4.085\cdot10^{-5}$	$6.808\cdot10^{-6}$	$8.933\cdot10^{-6}$	1.97
Schwartz	1024^2	810964	$9.834\cdot10^{-6}$	$1.640\cdot10^{-6}$	$2.153\cdot10^{-6}$	2.05
current	32^2	828	$9.448\cdot10^{-5}$	$1.965\cdot10^{-6}$	$5.407\cdot10^{-6}$	—
current	64^2	3326	$9.659\cdot10^{-7}$	$1.622\cdot10^{-8}$	$4.080\cdot10^{-8}$	6.92
current	128^2	13266	$2.179\cdot10^{-8}$	$8.458\cdot10^{-10}$	$1.295\cdot10^{-9}$	4.26
current	256^2	52886	$2.110\cdot10^{-8}$	$9.797\cdot10^{-11}$	$2.689\cdot10^{-10}$	3.10
current	512^2	211088	$1.028\cdot10^{-8}$	$2.417\cdot10^{-11}$	$1.331\cdot10^{-10}$	2.01

Table 13. Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz et al. algorithm. The boundary conditions are Dirichlet everywhere. Here we set the geometric regularization length to $\delta = 4h$. The convergence rates ϖ are calculated using L_1.

we choose delta to vary linearly with h ($\delta = 4h$). Third, we choose $\delta = \sqrt{R_1 h}$, where $R_1 = 0.2$ is the radius of the largest circle in Figure 7. Finally, we choose

$$\delta = 0.1\sqrt[4]{R_1^3 h}.$$

The convergence rates for the four choices are quite different.

First, we set our geometric regularization length to a constant $\delta = 0.01$. Tables 11 and 12 show error rates for Neumann and Dirichlet boundary conditions at the cut faces, respectively. With this fixed δ, the current algorithm shows much smaller errors than Schwartz et al. In the L_1 norm, we get better error rates at 32^2 than Schwartz, et al. gets at 1024^2. We also show excellent convergence rates with both Neumann and Dirichlet boundary conditions at the irregular faces.

Next, we set our geometric regularization length to $\delta = 4h$. Tables 13 and 14 show the error rates for Dirichlet and Neumann boundary conditions at the cut faces, respectively. With δ converging linearly with grid refinement, the improvement over Schwartz et al. is far more modest, especially with Neumann boundary conditions at the cut faces. Our convergence rates in this case (again, especially with Neumann boundary conditions), are erratic.

Next, we set our geometric regularization length to $\delta = \sqrt{R_1 h}$. Tables 15 and 16 show the error rates for Neumann and Dirichlet boundary conditions at the cut faces, respectively. With this formulation of δ, we once again get much better error rates than Schwartz et al. Here again, in the L_1 norm, we get better error rates at 32^2 than Schwartz, et al. gets at 1024^2. Our convergence rates here for Dirichlet boundary conditions at the embedded boundary are fourth order. For Neumann boundary conditions at the irregular faces, our convergence rates here are more erratic.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	828	$1.232\cdot10^{-3}$	$3.357\cdot10^{-4}$	$4.141\cdot10^{-4}$	—
Schwartz	64^2	3238	$4.626\cdot10^{-4}$	$1.379\cdot10^{-4}$	$1.736\cdot10^{-4}$	1.28
Schwartz	128^2	12810	$2.328\cdot10^{-4}$	$5.009\cdot10^{-5}$	$6.474\cdot10^{-5}$	1.46
Schwartz	256^2	50918	$1.477\cdot10^{-4}$	$2.196\cdot10^{-5}$	$2.940\cdot10^{-5}$	1.19
Schwartz	512^2	203052	$8.893\cdot10^{-5}$	$9.725\cdot10^{-6}$	$1.335\cdot10^{-5}$	1.17
Schwartz	1024^2	810964	$4.877\cdot10^{-5}$	$4.166\cdot10^{-6}$	$5.785\cdot10^{-6}$	1.22
current	32^2	828	$1.683\cdot10^{-3}$	$1.122\cdot10^{-4}$	$2.043\cdot10^{-4}$	—
current	64^2	3326	$6.971\cdot10^{-6}$	$5.483\cdot10^{-7}$	$9.162\cdot10^{-7}$	7.67
current	128^2	13266	$6.512\cdot10^{-7}$	$6.887\cdot10^{-8}$	$9.896\cdot10^{-8}$	2.99
current	256^2	52886	$8.852\cdot10^{-7}$	$7.875\cdot10^{-8}$	$1.125\cdot10^{-7}$	-0.193
current	512^2	211088	$4.898\cdot10^{-7}$	$3.394\cdot10^{-8}$	$5.000\cdot10^{-8}$	1.21

Table 14. Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz et al. algorithm. The boundary conditions are Dirichlet on the domain boundary and Neumann on the embedded boundary. Here we set the geometric regularization length to $\delta = 4h$. The convergence rates ϖ are calculated using L_1.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	846	$1.578\cdot10^{-3}$	$5.115\cdot10^{-4}$	$6.401\cdot10^{-4}$	—
Schwartz	64^2	3312	$5.006\cdot10^{-4}$	$1.460\cdot10^{-4}$	$1.845\cdot10^{-4}$	1.80
Schwartz	128^2	13088	$2.112\cdot10^{-4}$	$4.692\cdot10^{-5}$	$6.028\cdot10^{-5}$	1.63
Schwartz	256^2	52022	$8.354\cdot10^{-5}$	$1.484\cdot10^{-5}$	$1.940\cdot10^{-5}$	1.66
Schwartz	512^2	20751	$3.258\cdot10^{-5}$	$4.992\cdot10^{-6}$	$6.646\cdot10^{-6}$	1.57
Schwartz	1024^2	82879	$1.000\cdot10^{-5}$	$1.449\cdot10^{-6}$	$1.944\cdot10^{-6}$	1.78
current	32^2	846	$6.139\cdot10^{-5}$	$4.725\cdot10^{-6}$	$8.113\cdot10^{-6}$	—
current	64^2	3330	$1.274\cdot10^{-6}$	$8.435\cdot10^{-8}$	$1.691\cdot10^{-7}$	5.80
current	128^2	13252	$4.698\cdot10^{-7}$	$4.297\cdot10^{-8}$	$6.226\cdot10^{-8}$	0.973
current	256^2	52794	$8.422\cdot10^{-8}$	$7.356\cdot10^{-9}$	$1.057\cdot10^{-8}$	2.54
current	512^2	210856	$2.127\cdot10^{-8}$	$1.846\cdot10^{-9}$	$2.645\cdot10^{-9}$	1.99

Table 15. Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz et al. algorithm. The boundary conditions are Dirichlet on the domain boundary and Neumann on the embedded boundary. Here we set the geometric regularization length to $\delta = \sqrt{R_1 h}$. The convergence rates ϖ are calculated using L_1.

Finally we set geometric regularization length to $\delta = 0.1\sqrt[4]{R_1^3 h}$. Tables 17 and 18 show the error rates for Dirichlet and Neumann boundary conditions at the cut faces, respectively. We see excellent convergence rates and error values for both types of boundary condition.

Clearly, how the regularization length varies with grid refinement is an important concern. We suspect that the optimal formulation will depend upon the nature of the partial differential equation as well as its boundary conditions.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	846	$6.581\cdot10^{-3}$	$1.496\cdot10^{-3}$	$1.993\cdot10^{-3}$	—
Schwartz	64^2	3312	$1.890\cdot10^{-3}$	$4.014\cdot10^{-4}$	$5.254\cdot10^{-4}$	1.89
Schwartz	128^2	13088	$4.148\cdot10^{-4}$	$9.815\cdot10^{-5}$	$1.295\cdot10^{-4}$	2.03
Schwartz	256^2	52022	$1.355\cdot10^{-4}$	$2.626\cdot10^{-5}$	$3.447\cdot10^{-5}$	1.90
Schwartz	512^2	207510	$3.214\cdot10^{-5}$	$6.751\cdot10^{-6}$	$8.842\cdot10^{-6}$	1.95
Schwartz	1024^2	828794	$8.573\cdot10^{-6}$	$1.643\cdot10^{-6}$	$2.157\cdot10^{-6}$	2.03
current	32^2	846	$5.402\cdot10^{-6}$	$2.143\cdot10^{-7}$	$3.879\cdot10^{-7}$	—
current	64^2	3330	$2.792\cdot10^{-7}$	$1.039\cdot10^{-8}$	$1.807\cdot10^{-8}$	4.36
current	128^2	13252	$1.421\cdot10^{-8}$	$5.089\cdot10^{-10}$	$7.016\cdot10^{-10}$	4.35
current	256^2	52794	$2.260\cdot10^{-9}$	$3.414\cdot10^{-11}$	$7.429\cdot10^{-11}$	3.89
current	512^2	210856	$4.074\cdot10^{-10}$	$2.406\cdot10^{-12}$	$5.238\cdot10^{-12}$	3.82

Table 16. Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz et al. algorithm. The boundary conditions are Dirichlet everywhere. Here we set the geometric regularization length to $\delta = \sqrt{R_1 h}$. The convergence rates ϖ are calculated using L_1.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	846	$8.863\cdot10^{-3}$	$1.775\cdot10^{-3}$	$2.356\cdot10^{-3}$	—
Schwartz	64^2	3312	$1.700\cdot10^{-3}$	$4.134\cdot10^{-4}$	$5.400\cdot10^{-4}$	2.10
Schwartz	128^2	13088	$6.486\cdot10^{-4}$	$1.026\cdot10^{-4}$	$1.361\cdot10^{-4}$	2.01
Schwartz	256^2	52022	$1.753\cdot10^{-4}$	$2.692\cdot10^{-5}$	$3.557\cdot10^{-5}$	1.93
Schwartz	512^2	207510	$3.698\cdot10^{-5}$	$6.802\cdot10^{-6}$	$8.921\cdot10^{-6}$	1.98
Schwartz	1024^2	828794	$1.002\cdot10^{-5}$	$1.640\cdot10^{-6}$	$2.152\cdot10^{-6}$	2.05
current	32^2	846	$2.322\cdot10^{-3}$	$1.896\cdot10^{-5}$	$9.508\cdot10^{-5}$	—
current	64^2	3330	$8.003\cdot10^{-5}$	$1.197\cdot10^{-6}$	$3.742\cdot10^{-6}$	3.98
current	128^2	13252	$3.913\cdot10^{-6}$	$7.621\cdot10^{-9}$	$6.239\cdot10^{-8}$	7.29
current	256^2	52794	$6.676\cdot10^{-7}$	$7.522\cdot10^{-10}$	$6.327\cdot10^{-9}$	3.34
current	512^2	210856	$5.136\cdot10^{-8}$	$5.920\cdot10^{-11}$	$4.836\cdot10^{-10}$	3.66

Table 17. Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz et al. algorithm. The boundary conditions are Dirichlet everywhere. Here we set the geometric regularization length to $\delta = 0.1\sqrt[4]{R_1^3 h}$. The convergence rates ϖ are calculated using L_1.

7. Conclusions

We present a fourth-order, conservative discretization of Poisson's equation in the presence of complex geometry. We show that our algorithm converges at the expected rate for smooth solutions and geometries. We show that our algorithm has a similar eigenvalue spectrum to the a widely used second-order algorithm but is much more accurate with a sufficiently smooth geometric description. We show that the effect of geometric discontinuities on error rates can be profound.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	846	$9.291\cdot10^{-3}$	$1.131\cdot10^{-3}$	$1.540\cdot10^{-3}$	—
Schwartz	64^2	3312	$2.222\cdot10^{-3}$	$2.422\cdot10^{-4}$	$3.190\cdot10^{-4}$	2.22
Schwartz	128^2	13088	$1.348\cdot10^{-3}$	$1.408\cdot10^{-4}$	$1.940\cdot10^{-4}$	7.82
Schwartz	256^2	52022	$3.738\cdot10^{-4}$	$4.130\cdot10^{-5}$	$5.694\cdot10^{-5}$	1.76
Schwartz	512^2	207510	$1.141\cdot10^{-4}$	$1.087\cdot10^{-5}$	$1.500\cdot10^{-5}$	1.92
Schwartz	1024^2	828794	$3.202\cdot10^{-5}$	$3.167\cdot10^{-6}$	$4.370\cdot10^{-6}$	1.77
current	32^2	846	$4.085\cdot10^{-2}$	$3.202\cdot10^{-3}$	$5.722\cdot10^{-3}$	—
current	64^2	3330	$1.254\cdot10^{-3}$	$3.802\cdot10^{-5}$	$7.107\cdot10^{-5}$	6.39
current	128^2	13252	$1.189\cdot10^{-4}$	$6.829\cdot10^{-6}$	$1.186\cdot10^{-5}$	2.47
current	256^2	52794	$1.318\cdot10^{-5}$	$6.356\cdot10^{-7}$	$1.223\cdot10^{-6}$	3.42
current	512^2	210856	$1.134\cdot10^{-6}$	$5.400\cdot10^{-8}$	$8.949\cdot10^{-8}$	3.55

Table 18. Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz et al. algorithm. The boundary conditions are Dirichlet on the domain boundary and Neumann on the embedded boundary. Here we set the geometric regularization length to $\delta = 0.1\sqrt[4]{R_1^3 h}$. The convergence rates ϖ are calculated using L_1.

Even in the presence of these discontinuities, however, higher-order convergence can be recovered if one removes the singular parts of the solution or smooths the geometric description. To retain higher-order accuracy, how the smoothing length scale varies with grid refinement is an important concern. We present one such refinement scheme which performs quite well for both Neumann and Dirichlet boundary conditions at cut faces.

Acknowledgment

The authors would like to thank Dr. Phillip Colella for his technical advice and insight.

References

[1] M. J. Aftosmis, M. J. Berger, and J. E. Melton, *Robust and efficient cartesian mesh generation for component-based geometry*, AIAA Journal **36** (1998), no. 6, 952–960.

[2] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, and H. Zhang, *PETSc users manual*, Tech. Report ANL-95/11 - Revision 3.5, Argonne National Laboratory, 2014.

[3] ———, *PETSc Web page*, 2014.

[4] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, Modern software tools in scientific computing (E. Arge, A. M. Bruaset, and H. P. Langtangen, eds.), Birkhäuser, 1997, pp. 163–202. Zbl

[5] V. A. Barker, L. S. Blackford, J. Dongarra, J. Du Croz, S. Hammarling, M. Marinova, J. Waśniewski, and P. Yalamov, *LAPACK95 users' guide*, Software, Environments, and Tools, no. 13, SIAM, Philadelphia, PA, 2001. MR Zbl

[6] S. C. Brenner and L. R. Scott, *The mathematical theory of finite element methods*, 3rd ed., Texts in Applied Mathematics, no. 15, Springer, New York, 2008. MR Zbl

[7] C. Campos, J. E. Roman, E. Romero, and A. Tomas, *SLEPc users manual*, Tech. Report DSIC-II/24/02 - Revision 3.3, D. Sistemes Informàtics i Computació, Universitat Politècnica de València, 2012.

[8] H. Cheng, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm in three dimensions*, J. Comput. Phys. **155** (1999), no. 2, 468–498. MR Zbl

[9] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Modiano, D. B. Serafini, and B. V. Straalen, *Chombo software package for AMR applications: design document*, Tech. Report LBNL-6616E, LBNL, July 2014.

[10] P. Colella, D. T. Graves, T. J. Ligocki, G. Miller, D. Modiano, P. Schwartz, B. V. Straalen, J. Pillod, D. Trebotich, and M. Barad, *EBChombo software package for Cartesian grid, embedded boundary application*, Tech. Report LBNL-6615E, LBNL, 2014.

[11] P. Colella, D. T. Graves, B. J. Keen, and D. Modiano, *A Cartesian grid embedded boundary method for hyperbolic conservation laws*, J. Comput. Phys. **211** (2006), no. 1, 347–366. MR Zbl

[12] D. Devendran, D. T. Graves, and H. Johansen, *A hybrid multigrid algorithm for Poisson's equation using an adaptive, fourth order treatment of cut cells*, Tech. Report LBNL-1004329, LBNL, 2014.

[13] Z. Dragojlovic, F. Najmabadi, and M. Day, *An embedded boundary method for viscous, conducting compressible flow*, J. Comput. Phys. **216** (2006), no. 1, 37–51. MR Zbl

[14] F. Gibou and R. Fedkiw, *A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem*, J. Comput. Phys. **202** (2005), no. 2, 577–601. MR Zbl

[15] D. T. Graves, P. Colella, D. Modiano, J. Johnson, B. Sjogreen, and X. Gao, *A Cartesian grid embedded boundary method for the compressible Navier–Stokes equations*, Commun. Appl. Math. Comput. Sci. **8** (2013), no. 1, 99–122. MR Zbl

[16] L. Greengard and J.-Y. Lee, *A direct adaptive Poisson solver of arbitrary order accuracy*, J. Comput. Phys. **125** (1996), no. 2, 415–424. MR Zbl

[17] V. Hernández, J. E. Román, and V. Vidal, *SLEPc: scalable library for eigenvalue problem computations*, High performance computing for computational science (Berlin) (J. M. L. M. Palma, A. A. Sousa, J. Dongarra, and V. Hernández, eds.), Lecture Notes in Computer Science, no. 2565, Springer, 2003, pp. 377–391. Zbl

[18] V. Hernandez, J. E. Roman, and V. Vidal, *SLEPc: a scalable and flexible toolkit for the solution of eigenvalue problems*, ACM Trans. Math. Software **31** (2005), no. 3, 351–362. MR

[19] H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys. **147** (1998), no. 1, 60–85. MR Zbl

[20] H. S. Johansen, *Cartesian grid embedded boundary finite difference methods for elliptic and parabolic partial differential equations on irregular domains*, Ph.D. thesis, University of California, Berkeley, 1997.

[21] L. D. Landau and E. M. Lifshitz, *Fluid mechanics*, 2nd ed., Course of Theoretical Physics, no. 6, Pergamon, Oxford, 1987. MR

[22] R. J. LeVeque, *Numerical methods for conservation laws*, Birkhäuser, Basel, 1990. MR Zbl

[23] ———, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, SIAM, Philadelphia, 2007. MR Zbl

[24] R. J. LeVeque and Z. L. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal. **31** (1994), no. 4, 1019–1044. MR Zbl

[25] D. F. Martin and K. L. Cartwright, *Solving Poisson's equation using adaptive mesh refinement*, Tech. Report UCB/ERI M96/66, University of California, Berkeley, 1996.

[26] A. McKenney, L. Greengard, and A. Mayo, *A fast Poisson solver for complex geometries*, J. Comput. Phys. **118** (1995), no. 2, 348–355. MR Zbl

[27] G. H. Miller and D. Trebotich, *An embedded boundary method for the Navier–Stokes equations on a time-dependent domain*, Commun. Appl. Math. Comput. Sci. **7** (2012), no. 1, 1–31. MR Zbl

[28] A. Nonaka, D. Trebotich, G. Miller, D. Graves, and P. Colella, *A higher-order upwind method for viscoelastic flow*, Commun. Appl. Math. Comput. Sci. **4** (2009), 57–83. MR Zbl

[29] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome, *An adaptive Cartesian grid method for unsteady compressible flow in irregular regions*, J. Comput. Phys. **120** (1995), no. 2, 278–304. MR Zbl

[30] S. Z. Pirzadeh, *Advanced unstructured grid generation for complex aerodynamic applications*, AIAA Journal **48** (2010), no. 5, 904–915.

[31] V. L. Rvačev, *An analytic description of certain geometric objects*, Dokl. Akad. Nauk SSSR **153** (1963), 765–767, In Russian; translated in Soviet Math. Dokl. **4** (1963), 1750–1753. MR Zbl

[32] P. Schwartz, M. Barad, P. Colella, and T. Ligocki, *A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions*, J. Comput. Phys. **211** (2006), no. 2, 531–550. MR Zbl

[33] P. Schwartz, J. Percelay, T. J. Ligocki, H. Johansen, D. T. Graves, D. Devendran, P. Colella, and E. Ateljevich, *High-accuracy embedded boundary grid generation using the divergence theorem*, Commun. Appl. Math. Comput. Sci. **10** (2015), no. 1, 83–96. MR Zbl

[34] V. Shapiro, *Semi-analytic geometry with R-functions*, Acta Numer. **16** (2007), 239–303. MR Zbl

[35] G. Strang, *Linear algebra and its applications*, Academic, New York, 1976. MR Zbl

[36] E. Tadmor and J. Tanner, *Adaptive mollifiers for high resolution recovery of piecewise smooth data from its spectral information*, Found. Comput. Math. **2** (2002), no. 2, 155–189. MR Zbl

[37] D. Trebotich, G. H. Miller, and M. D. Bybee, *A penalty method to model particle interactions in DNA-laden flows*, Journal of Nanoscience and Nanotechnology **8** (2008), no. 7, 3749–3756.

Received March 25, 2016. Revised December 19, 2016.

DHARSHI DEVENDRAN: pdevendran@gmail.com
Applied Numerical Algorithms Group (ANAG), Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

DANIEL T. GRAVES: DTGraves@lbl.gov
Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

HANS JOHANSEN: hjohansen@lbl.gov
Applied Numerical Algorithms Group (ANAG), Computational Research Division, Lawrence Berkeley National Laboratory, MS 50A1148, One Cyclotron Road, Berkeley, CA 94720, United States

TERRY LIGOCKI: TJLigocki@lbl.gov
Applied Numerical Algorithms Group (ANAG), Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

COMM. APP. MATH. AND COMP. SCI.
Vol. 12, No. 1, 2017
dx.doi.org/10.2140/camcos.2017.12.81

msp

A CENTRAL-UPWIND GEOMETRY-PRESERVING METHOD FOR HYPERBOLIC CONSERVATION LAWS ON THE SPHERE

ABDELAZIZ BELJADID AND PHILIPPE G. LEFLOCH

We introduce a second-order, central-upwind finite volume method for the discretization of nonlinear hyperbolic conservation laws on the two-dimensional sphere. The semidiscrete version of the proposed method is based on a technique of local propagation speeds, and the method is free of any Riemann solver. The main advantages of our scheme are its high resolution of discontinuous solutions, its low numerical dissipation, and its simplicity of implementation. We do not use any splitting approach, which is often applied to upwind schemes in order to simplify the resolution of Riemann problems. The semidiscrete form of our scheme is strongly built upon the analytical properties of nonlinear conservation laws and the geometry of the sphere. The curved geometry is treated here in an analytical way so that the semidiscrete form of the proposed scheme is consistent with a geometric compatibility property. Furthermore, the time evolution is carried out by using a total-variation diminishing Runge–Kutta method. A rich family of (discontinuous) stationary solutions is available for the conservation laws under consideration when the flux is nonlinear and foliated (in a suitable sense). We present a series of numerical tests, encompassing various nontrivial steady state solutions and therefore providing a good validation of the accuracy and efficiency of the proposed central-upwind finite volume scheme. Our numerical tests confirm that the scheme is stable and succeeds in accurately capturing discontinuous steady state solutions to conservation laws posed on the sphere.

1. Introduction

Nonlinear hyperbolic problems involving conservation laws, or more generally balance laws, arise in continuum physics and in many engineering applications. One of the most important partial differential equations (PDEs) is Burgers' equation, which plays a crucial role in designing numerical methods and arises in a variety of applications. For instance, it arises in the modeling of water infiltration in unsaturated soil and fluid flows through porous media, which are significant in petroleum and environmental engineering problems and in traffic flow problems [11;

MSC2010: primary 35L65, 65M08; secondary 76L05.
Keywords: hyperbolic conservation law, shock wave, geometry-compatible flux, central-upwind scheme.

37; 31]. In general, the solutions to hyperbolic PDEs can develop sharp gradients (or discontinuities) in finite time, even when starting from smooth initial conditions. For example, in multiphase flow in unsaturated porous media, the wetting front can be very sharp [28; 29]. High-resolution shock-capturing techniques are required since they have the ability to capture sharp gradients within a few computational cells with low levels of numerical diffusion and oscillation.

Various classes of so-called shock-capturing schemes have been proposed. In particular, upwind and central schemes have been used to numerically solve hyperbolic conservation laws. Generally, it can be stated that the difference between these schemes is that upwind methods use characteristic-related information, while central methods do not. The use of characteristic information in upwind schemes can improve the results but renders these schemes, in some cases, computationally expensive. Central schemes are widely used (see, e.g., [32]) after the pioneering work of Nessyahu and Tadmor [34], where a second-order finite volume central method on a staggered grid in spacetime was first proposed. This strategy leads to high resolution and the simplicity of the Riemann-solver free method. As observed by Kurganov and Tadmor [20], this scheme suffers from excessive numerical viscosity when a small time step is considered.

In order to improve the performance of central schemes, some characteristic information can still be used. Kurganov et al. [16] proposed the central-upwind schemes which are based on information obtained from the local speeds of wave propagation. The central-upwind schemes can be considered as a generalization of central schemes originally developed by Kurganov and Tadmor [20; 21] and Kurganov and Levy [14]. The central-upwind schemes are simple, since they use no Riemann solvers, and they have proven their effectiveness in multiple studies, as shown in [18; 19; 15]. Kurganov and Petrova [17] extended the central-upwind schemes to triangular grids for solving two-dimensional Cartesian systems of conservation laws. Next, Beljadid et al. [4] proposed a two-dimensional well-balanced and positivity-preserving cell-vertex central-upwind scheme for the computation of shallow water equations with source terms due to bottom topography.

Several studies have been recently done for hyperbolic conservation laws posed on curved manifolds. The solutions of conservation laws including the systems on manifolds and on spacetimes were studied in [35; 33] and by LeFloch and coauthors [1; 2; 5; 6; 23; 26; 27]. More recently, hyperbolic conservation laws for an evolving surface were investigated by Dziuk, Kröner, and Müller [10], Giesselman [12], and Dziuk and Elliott [9]. Earlier on, for such problems, Ben-Artzi and LeFloch [6] and LeFloch and Okutmustur [27] established a general well-posedness theory for conservation laws on manifolds. In fact, several physically relevant classes of conservation laws in curved spaces were extensively investigated in recent years and we refer the interested reader to [8; 13; 22; 24; 25].

Burgers' equation provides a simple, yet challenging, equation which admits discontinuous solutions, and it provides a simplified setup for the design and validation of shock-capturing numerical methods. Burgers' equation and its generalizations to a curved manifold have been widely used in the physical and mathematical literature. In [3], we have used a class of Burgers-type equations on the sphere and adopted the methodology first proposed by Ben-Artzi, Falcovitz, and LeFloch [5], which uses second-order approximations based on generalized Riemann problems. In [3], a scheme was proposed which uses piecewise linear reconstructions based on solution values at the centers of the computational cells and on values of Riemann solutions at the cell interfaces. A second-order approximation based on a generalized Riemann solver was then proposed, together with a total-variation diminishing Runge–Kutta method (TVDRK3) with operator splitting for the temporal integration.

The finite volume method developed in [5] is strongly linked to the structure of the governing equation on the sphere. The geometric dimensions are considered in an analytical way which leads to discrete forms of schemes that respect exactly the geometric compatibility property. The splitting approach which is used in these schemes simplifies the resolution of the Riemann problem, but it increases the computational cost.

In the present study, we propose a new finite volume method which is less expensive in terms of computational cost. This scheme is free of any Riemann solver and does not use any splitting approach, while such a splitting is widely used in upwind schemes when one needs to simplify the resolution of Riemann problems. The present paper provides the first study of *geometry-preserving, central-upwind schemes for conservation laws on a curved geometry*.

Burgers' equation and its generalizations will be used in the present paper in order to develop and validate the new finite volume method. We design in full detail a geometry-compatible central-upwind scheme for scalar nonlinear hyperbolic conservation laws on the sphere. This system has a simple appearance, but it generates solutions that have a very rich wave structure (due to the curved geometry), and its solutions provide an effective framework for assessing numerical methods. Our goal is to develop and validate a finite volume method which is free of any Riemann problem and is consistent with the geometric compatibility (or divergence-free) condition, at the discrete level. As we prove, the proposed scheme is efficient and accurate for discontinuous solutions and implies only negligible geometric distortions on the solutions.

An outline of the paper is as follows. In Section 2, the governing equations related to this study are presented. Section 3 is devoted to the derivation of the semidiscrete version of our scheme. In Section 4, the coordinate system and the nonoscillatory reconstruction are described. In Section 5, we present the geometry-compatible flux vectors and some particular steady state solutions as well as confined

solutions, which will be used to validate the performance of the proposed method. In Section 6, we demonstrate the high-resolution of the proposed central-upwind scheme thanks to a series of numerical experiments. Finally, some concluding remarks are provided.

2. Governing equations

We consider nonlinear hyperbolic equations posed on the sphere $\mathbb{S}^2$ and based on the flux vector $F = F(x, u)$, depending on the function $u(t, x)$ and the space variable x. This flux is assumed to satisfy the following geometric compatibility condition: for any arbitrary constant value $\bar{u} \in \mathbb{R}$,

$$\nabla \cdot (F(\cdot, \bar{u})) = 0. \tag{2-1}$$

We also assume that the flux takes the form

$$F(x, u) = n(x) \wedge \Phi(x, u), \tag{2-2}$$

where $n(x)$ is the unit normal vector to the sphere and the function $\Phi(x, u)$ is a vector field in $\mathbb{R}^3$, restricted to $\mathbb{S}^2$ and defined by

$$\Phi(x, u) = \nabla h(x, u). \tag{2-3}$$

Here, $h = h(x, u)$ is a smooth function depending on the space variable x and the state variable $u(t, x)$. Observe that (for instance by Claim 2.2 in [5]) the conditions (2-2) and (2-3) for the flux vector are sufficient to ensure the validity of the geometric compatibility condition (2-1).

Here we are going to develop and validate a new geometry-preserving central-upwind scheme which approximates solutions to the hyperbolic conservation law

$$\partial_t u + \nabla \cdot F(x, u) = 0, \quad (x, t) \in \mathbb{S}^2 \times \mathbb{R}_+, \tag{2-4}$$

where $\nabla \cdot F$ is the divergence of the vector field F. Given any data u_0 prescribed on the sphere, we consider the following initial condition for the unknown function $u = u(t, x)$:

$$u(0, x) = u_0(x), \quad x \in \mathbb{S}^2. \tag{2-5}$$

Equation (2-4) can be rewritten, using general local coordinates and the index of summation j, in the form

$$\partial_t u + \frac{1}{\sqrt{|g|}} \partial_j (\sqrt{|g|} F^j(x, u)) = 0, \tag{2-6}$$

or

$$\partial_t (\sqrt{|g|} u) + \partial_j (\sqrt{|g|} F^j(x, u)) = 0, \tag{2-7}$$

where in local coordinates $x = (x^j)$, the derivatives are denoted by $\partial_j = \frac{\partial}{\partial x^j}$, F^j are the components of the flux vector, and g is the metric.

The conservation law (2-4) becomes

$$\partial_t v + \partial_j(\sqrt{|g|}F^j(x, v/\sqrt{|g|})) = 0, \tag{2-8}$$

where $v = u\sqrt{|g|}$. This form will be used in the derivation of the semidiscrete form of the proposed scheme. For the latitude-longitude grid on the sphere, the divergence operator of the flux vector is

$$\nabla \cdot F = \frac{1}{\cos\phi}\left(\frac{\partial}{\partial\phi}(F_\phi \cos\phi) + \frac{\partial F_\lambda}{\partial\lambda}\right), \tag{2-9}$$

where F_ϕ and F_λ are the flux components in the latitude (ϕ) and longitude (λ) directions on the sphere, respectively.

3. Derivation of the proposed method

Discretization of the divergence operator. We will describe the derivation of the new central-upwind scheme in detail for the three steps: reconstruction, evolution, and projection. We will develop and give a semidiscrete form of the proposed method for a general computational grid used to discretize the sphere. We assume the discretization of the sphere $\mathbb{S}^2 = \bigcup_{j=1}^{j=N} C_j$, where C_j are the computational cells with areas $|C_j|$. We denote by m_j the number of cell sides of C_j and by $C_{j1}, C_{j2}, \ldots, C_{jm_j}$ the neighboring computational cells that share with C_j the common sides $(\partial C_j)_1, (\partial C_j)_2, \ldots, (\partial C_j)_{m_j}$, respectively. The length of each cell interface $(\partial C_j)_k$ is denoted by l_{jk}. The discrete value of the state variable $u(t, x)$ inside the computational cell C_j at a point $G_j \in C_j$ is denoted by $\boldsymbol{u}_j^n$ at step n. The longitude and latitude coordinates of the suitable point G_j to use inside each computational cell C_j are presented on page 97. These coordinates should be chosen according to the reconstruction of the state variable $u(t, x)$ over the computational cells used on the sphere. Finally, we use the notations Δt and $t_n = n\Delta t$ for the time step and the time at step n, respectively. To obtain the semidiscrete form of the proposed scheme, a first-order explicit development in time will be used. The resulting ODE can be numerically solved using a higher-order SSP ODE solver such as Runge–Kutta of the multistep methods. In the numerical experiments, the third-order TVD Runge–Kutta method proposed by Shu and Osher [36] is used.

In this section, we will present a general form of the discretization of the divergence operator for a general computational grid on the sphere. The approximation of the flux divergence can be written using the divergence theorem as

$$[\nabla \cdot F(x, u)]^{\text{approx}} = \frac{I_j}{|C_j|}, \quad I_j = \left[\oint_{\partial C_j} F(x, u) \cdot \nu(x)\, ds\right]^{\text{approx}}, \tag{3-1}$$

where $\nu(x)$ is the unit normal vector to the boundary ∂C_j of the computational cell C_j and ds is the infinitesimal length along ∂C_j.

The scalar potential function h is used to obtain the following approximation along each side of the computational cell C_j.

Claim 3.1. *For a three-dimensional flux $\Phi(x,u)$ given by (2-3), where $h = h(x,u)$ is a smooth function in the neighborhood of the sphere $\mathbb{S}^2$, the total approximate flux through the cell interface e is given by*

$$\oint_{e^1}^{e^2} F(x,u)\cdot \nu(x)\, ds = -(h(e^2,u_j) - h(e^1,u_j)), \tag{3-2}$$

where e^1 and e^2 are the initial and final endpoints of the side e in the sense of integration and u_j is the estimate value of the variable u along the side e.

Namely, the flux vector is written in the form $F(x,u) = n(x)\wedge\Phi(x,u)$ and we can derive the approximation of the integral along each cell side of C_j

$$\begin{aligned}\oint_{e^1}^{e^2} F(x,u)\cdot \nu(x)\, ds &= \oint_{e^1}^{e^2} (n(x)\wedge\Phi(x,u))\cdot\nu(x)\, ds\\ &= -\oint_{e^1}^{e^2} \Phi(x,u)\cdot(n(x)\wedge\nu(x))\, ds = -\oint_{e^1}^{e^2} \nabla h(x,u)\cdot\tau(x)\, ds\\ &= -\oint_{e^1}^{e^2} \nabla_{\partial C_j} h(x,u)\, ds = -(h(e^2,u_j) - h(e^1,u_j)),\end{aligned} \tag{3-3}$$

where $\tau(x)$ is the unit vector tangent to the boundary ∂C_j.

Remark 3.2. Using the discrete approximations based on Claim 3.1, if a constant value of the state variable $u(t,x) = u_j = \bar{u}$ is considered, one obtains

$$\begin{aligned}[\nabla\cdot F(x,u)]^{\text{approx}} &= \frac{1}{|C_j|}\left[\oint_{\partial C_j} F(x,u)\cdot\nu(x)\, ds\right]^{\text{approx}}\\ &= -\sum_{e\in\partial C_j} (h(e^2,\bar{u}) - h(e^1,\bar{u})) = 0.\end{aligned} \tag{3-4}$$

This confirms that the discrete approximation of the divergence operator respects the divergence-free condition which is the geometric requirement that the proposed scheme should satisfy.

Reconstruction method and approximation of the one-sided local speeds of propagation of the waves. In the following, we will present the reconstruction of the proposed central-upwind scheme and the approximation used to obtain the maximum of the directional local speeds of propagation of the waves at cell interfaces inward and outward of computational cells. The semidiscrete form of the proposed scheme for (2-4) will be derived by using the approximation of the cell averages of the

solution. At each time $t = t_n$, the computed solution is

$$\boldsymbol{u}_j^n \approx \frac{1}{|C_j|}\int_{C_j} u(x, t_n)\, dV_g, \tag{3-5}$$

where $dV_g = \sqrt{g}\, dx^1\, dx^2$.

The discrete values $\boldsymbol{u}_j^n$ of the solution at time $t = t_n$ are used to construct a conservative piecewise polynomial function with possible discontinuities at the interfaces of the computational cells C_j:

$$\tilde{u}^n(x) = \sum_j w_j^n(x)\chi_j(x), \tag{3-6}$$

where $w_j^n(x)$ is a polynomial in two variables (λ and ϕ) and χ_j is the characteristic function which is defined using the Kronecker symbol δ_{jk} and, for any point of spatial coordinate x inside the computational cell C_k, we consider $\chi_j(x) = \delta_{jk}$.

To prevent oscillations, minmod-type reconstruction can be used to obtain the polynomial function $w_j^n(x)$ for each computational cell. Page 97 describes the reconstruction method used for the proposed central-upwind scheme on the sphere.

The maximum of the directional local speeds of propagation of the waves at the k-th interface inward and outward of the computational cell C_j are denoted by a_{jk}^{in} and a_{jk}^{out}, respectively. When the solution evolves over a time step Δt, the discontinuities move inward and outward at the k-th interface of the computational cell C_j with maximum distances $a_{jk}^{\text{in}}\Delta t$ and $a_{jk}^{\text{out}}\Delta t$, respectively. These distances of propagation are used at the computational cells to delimit different areas in which the solution is still smooth and the areas in which the solution may not be smooth when it evolves from the time level t_n to t_{n+1}.

We define the domain D_j as the part inside the cell C_j in which the solution is still smooth; see Figure 1. Two other types of domains are defined: the first type includes the "rectangular" domains D_{jk}, $k = 1, 2, \ldots, m_j$, along each side of C_j of width $(a_{jk}^{\text{out}} + a_{jk}^{\text{in}})\Delta t$ and length $l_{jk} + O(\Delta t)$, and the second type includes the domains denoted by E_{jk}, $k = 1, 2, \ldots, m_j$, around the cell vertices of computational cells. These domains are decomposed into two subdomains $D_{jk} = D_{jk}^+ \cup D_{jk}^-$ and $E_{jk} = E_{jk}^+ \cup E_{jk}^-$, where the subdomains with the superscript plus signs "+" and minus signs "−" are the domains inside and outside of the cell C_j, respectively. For purely geometrical reasons, the areas of the three types of subdomains are of orders $|D_j| = O(1)$, $|D_{jk}| = O(\Delta t)$, and $|E_{jk}| = O(\Delta t^2)$.

We consider the projection of the flux vector $\widetilde{F}$ according to the normal to the k-th cell interface $(\partial C_j)_k$:

$$f_{jk} = N_{jk} \cdot \widetilde{F}, \tag{3-7}$$

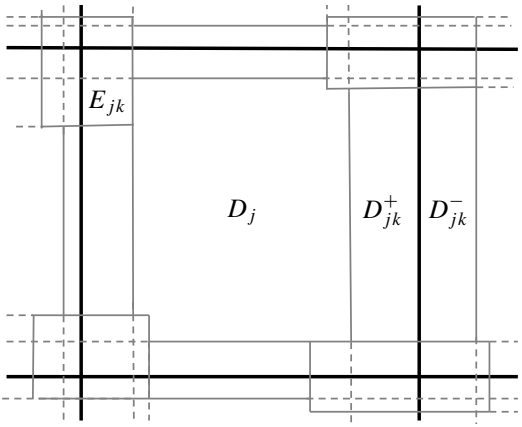


Figure 1. Schematic view of the decomposition of the control volume. The thick black lines are the limits of computational cells, and the thin gray lines are used for the decomposition of control volumes.

where N_{jk} is the unit normal vector to the cell interface $(\partial C_j)_k$ and $\widetilde{F}$ has the components $\sqrt{g}F^j(x, v/\sqrt{g})$ which are used in (2-8).

The one-sided local speeds of propagation of the waves at the k-th cell interface $(\partial C_j)_k$, inward and outward of the computational cell C_j, are estimated by

$$\begin{aligned} a_{jk}^{\text{out}} &= \max\left\{\frac{\partial f_{jk}}{\partial v}(M_{jk}, u_j(M_{jk})), \frac{\partial f_{jk}}{\partial v}(M_{jk}, u_{jk}(M_{jk})), 0\right\}, \\ a_{jk}^{\text{in}} &= -\min\left\{\frac{\partial f_{jk}}{\partial v}(M_{jk}, u_j(M_{jk})), \frac{\partial f_{jk}}{\partial v}(M_{jk}, u_{jk}(M_{jk})), 0\right\}, \end{aligned} \tag{3-8}$$

where $u_j(M_{jk})$ is the value of the state variable u at the midpoint M_{jk} of $(\partial C_j)_k$, which is obtained from the nonoscillatory reconstruction for the computational cell C_j and $u_{jk}(M_{jk})$ is the value of u at the same point M_{jk} using the nonoscillatory reconstruction for the neighboring cell C_{jk}.

Evolution and projection steps. In this section, the techniques used for the hyperbolic conservation laws and shallow water systems in a Cartesian framework [16; 18; 19; 15; 17; 4] will be extended to the case of hyperbolic conservation laws on the sphere. The computed cell averages $\bar{\boldsymbol{u}}_j^{n+1}$ of the numerical solution at time step t_{n+1} over the computational cells C_j are used to obtain the piecewise linear reconstruction $\widetilde{w}^{n+1}$ which should satisfy the conservative requirement

$$\bar{\boldsymbol{u}}_j^{n+1} = \frac{1}{|C_j|}\int_{C_j} \widetilde{w}^{n+1}(x)\, dV_g. \tag{3-9}$$

The average of the function $\widetilde{w}^{n+1}$ over the domain D_j is denoted by

$$\bar{w}^{n+1}(D_j) = \frac{1}{|D_j|}\int_{D_j} \widetilde{w}^{n+1}(x)\, dV_g. \tag{3-10}$$

Note that it is possible to derive the fully discrete form of the proposed scheme but it is impractical to use and, for simplicity, we will develop the semidiscrete form of the scheme. The ODE for approximating the cell averages of the solutions is derived by letting the time step Δt go to zero. This eliminates some terms because of their orders, and we keep the more consistent terms:

$$\begin{aligned}\frac{d\bar{\boldsymbol{u}}_j}{dt}(t_n) &= \lim_{\Delta t\to 0} \frac{\bar{\boldsymbol{u}}_j^{n+1} - \bar{\boldsymbol{u}}_j^{n}}{\Delta t} \\ &= \lim_{\Delta t\to 0} \frac{1}{\Delta t}\left[\frac{1}{|C_j|}\int_{D_j} \widetilde{w}^{n+1}(x)\, dV_g + \frac{1}{|C_j|}\sum_{k=1}^{m_j}\int_{D_{jk}^+} \widetilde{w}^{n+1}(x)\, dV_g \right. \\ &\qquad \left. + \frac{1}{|C_j|}\sum_{k=1}^{m_j}\int_{E_{jk}^+} \widetilde{w}^{n+1}(x)\, dV_g - \bar{\boldsymbol{u}}_j^{n}\right]. \end{aligned} \tag{3-11}$$

Since the areas of domains E_{jk} with $k = 1, 2, \ldots, m_j$ are of order Δt^2, we obtain

$$\int_{E_{jk}^+} \widetilde{w}^{n+1}(x)\, dV_g = O(\Delta t^2). \tag{3-12}$$

This approximation allows us to deduce that the third term on the right-hand side of (3-11) is of order Δt^2 and the result for the limit of this term vanishes for the ODE.

The second term in (3-11), in which we use the "rectangular" domains D_{jk}^+, will be estimated by using the assumption that the spatial derivatives of $\widetilde{w}^{n+1}$ are bounded independently of Δt. Under this assumption, the following claim gives an estimation of this term with an error of order Δt^2 for each $k \in [1, m_j]$.

Claim 3.3. *Consider the reconstruction given by (3-6), its evolution $\widetilde{w}^{n+1}$ over the global domain, and the definitions given at the bottom of page 86 for the domains D_{jk} and D_{jk}^+. If we assume that the spatial derivatives of $\widetilde{w}^{n+1}$ are bounded independently of Δt, then*

$$\int_{D_{jk}^+} \widetilde{w}^{n+1}(x)\, dV_g = |D_{jk}^+|\bar{w}^{n+1}(D_{jk}) + O(\Delta t^2). \tag{3-13}$$

Proof. It is obvious that, for the cases $|D_{jk}^+| = 0$ or $|D_{jk}^-| = 0$, (3-13) is valid. We assume that $|D_{jk}^+||D_{jk}^-| \neq 0$, and we consider

$$R = \int_{D_{jk}^+} \widetilde{w}^{n+1}(x)\, dV_g - |D_{jk}^+|\bar{w}^{n+1}(D_{jk}).$$

We have

$$R = \int_{D_{jk}^+} \widetilde{w}^{n+1}(x)\,dV_g - \frac{|D_{jk}^+|}{|D_{jk}|}\left(\int_{D_{jk}^+} \widetilde{w}^{n+1}(x)\,dV_g + \int_{D_{jk}^-} \widetilde{w}^{n+1}(x)\,dV_g\right)$$
$$= \frac{|D_{jk}^+|}{|D_{jk}|}\left[\frac{|D_{jk}^-|}{|D_{jk}^+|}\int_{D_{jk}^+} \widetilde{w}^{n+1}(x)\,dV_g - \int_{D_{jk}^-} \widetilde{w}^{n+1}(x)\,dV_g\right]$$
$$= \frac{|D_{jk}^+|}{|D_{jk}|}\left[\frac{a_{jk}^{\text{out}}}{a_{jk}^{\text{in}}}\int_{-a_{jk}^{\text{in}}\Delta t}^{0} \widetilde{w}^{n+1}(s)\tilde{l}_{jk}\,ds - \int_0^{a_{jk}^{\text{out}}\Delta t} \widetilde{w}^{n+1}(s)\tilde{l}_{jk}\,ds\right], \tag{3-14}$$

where $\tilde{l}_{jk}$ is the length of the domain D_{jk} and s is a variable along the outward axis orthogonal to the k-th cell interface; see Figures 1 and 3.

One obtains after the change of variable in the first integral of the last equality in (3-14)

$$R = \frac{|D_{jk}^+|}{|D_{jk}|}\tilde{l}_{jk}\int_0^{a_{jk}^{\text{out}}\Delta t}\left(\widetilde{w}^{n+1}\left(-\frac{a_{jk}^{\text{in}}}{a_{jk}^{\text{out}}}s\right) - \widetilde{w}^{n+1}(s)\right)ds.$$

Using the mean value theorem on the function $\widetilde{w}^{n+1}$, we obtain

$$R = -\frac{|D_{jk}^+|}{|D_{jk}|}\tilde{l}_{jk}\int_0^{a_{jk}^{\text{out}}\Delta t}\frac{a^{\text{in}}+a^{\text{out}}}{a^{\text{out}}}s\frac{\partial \widetilde{w}^{n+1}}{\partial s}(c_s)\,ds,$$

where $c_s \in [\min(s, -sa_{jk}^{\text{in}}/a_{jk}^{\text{out}}), \max(s, -sa_{jk}^{\text{in}}/a_{jk}^{\text{out}})]$.

We denote by M the upper bound of the spatial derivative of the function $\widetilde{w}^{n+1}$ over the domain D_{jk}. Therefore,

$$|R| \le Ml\frac{|D_{jk}^+|}{|D_{jk}^-|}\int_0^{a_{jk}^{\text{out}}\Delta t} s\,ds = \frac{Ml}{2}|D_{jk}^+||D_{jk}^-|.$$

Since $\tilde{l}_{jk} = l_{jk} + O(\Delta t)$ and both the areas $|D_{jk}^+|$ and $|D_{jk}^-|$ are of order Δt, we obtain $R = O(\Delta t^2)$. □

Using (3-13) in Claim 3.3,

$$\frac{1}{|C_j|}\sum_{k=1}^{m_j}\int_{D_{jk}^+} \widetilde{w}^{n+1}(x)\,dV_g = \frac{1}{|C_j|}\sum_{k=1}^{m_j}|D_{jk}^+|\overline{w}^{n+1}(D_{jk}) + O(\Delta t^2)$$
$$= \frac{\Delta t}{|C_j|}\sum_{k=1}^{m_j} a_{jk}^{\text{in}}(l_{jk} + O(\Delta t))\overline{w}^{n+1}(D_{jk}) + O(\Delta t^2). \tag{3-15}$$

Therefore, (3-11) can be written as

$$\frac{d\bar{\boldsymbol{u}}_j}{dt}(t_n) = \lim_{\Delta t\to 0}\frac{1}{\Delta t}\left[\frac{|D_j|}{|C_j|}\overline{w}^{n+1}(D_j) - \bar{\boldsymbol{u}}_j^n\right] + \sum_{k=1}^{m_j}\lim_{\Delta t\to 0}\frac{|D_{jk}^+|}{\Delta t|C_j|}\overline{w}^{n+1}(D_{jk}), \tag{3-16}$$

where

$$\overline{w}^{n+1}(D_{jk}) = \frac{1}{|D_{jk}|}\int_{D_{jk}} \widetilde{w}^{n+1}(x)\, dV_g. \tag{3-17}$$

In order to derive the semidiscrete form of the proposed scheme from (3-16), one needs to compute the average values $\overline{w}^{n+1}(D_{jk})$ and $\overline{w}^{n+1}(D_j)$. To compute $\overline{w}^{n+1}(D_{jk})$, (2-4) is integrated over the spacetime control volume $D_{jk} \times [t_n, t_{n+1}]$. After integration by parts and applying the divergence theorem to transform the surface integral of the divergence operator to the boundary integral and using the approximation (3-2) of the flux through the cell interfaces, we obtain

$$\overline{w}^{n+1}(D_{jk}) = \frac{1}{|D_{jk}|}\left[\int_{D^+_{jk}} w^n_j(x) dV_g + \int_{D^-_{jk}} w^n_{jk}(x) dV_g\right] - \frac{1}{|D_{jk}|}\int_{t_n}^{t_{n+1}}\int_{D_{jk}} \nabla\cdot F(x,u)\, dV_g, \tag{3-18}$$

and

$$\begin{aligned}
&\int_{D_{jk}} \nabla\cdot F(x,u)\, dV_g \\
&= \left[\int_{\partial D_{jk}} F(x,u)\cdot \nu(x)\, ds\right]^{\text{approx}} \\
&= \sum_{i=1}^{i=4}\int_{(\partial D_{jk})_i} F(x,u)\cdot\nu(x)\, ds \\
&= -\left[-h(e^2_{jk}, u_j(M_{jk})) + h(e^1_{jk}, u_j(M_{jk})) + h(e^2_{jk}, u_{jk}(M_{jk})) - h(e^1_{jk}, u_{jk}(M_{jk}))\right] \\
&\qquad + O(\Delta t),
\end{aligned} \tag{3-19}$$

where $(\partial D_{jk})_i$, $i = 1, 2, 3, 4$, are the four edges of the domain D_{jk}, e^2_{jk} and e^1_{jk} are the initial and final endpoints of the cell interface $(\partial C_j)_k$, and as mentioned before w^n_j and w^n_{jk} are the piecewise polynomial reconstructions in the computational cells C_j and C_{jk} at time t_n, respectively.

The term on the right-hand side of (3-19) of order $O(\Delta t)$ corresponds to the global result of the integration along the two edges of the domain D_{jk} having the length $(a^{\text{in}}_{jk} + a^{\text{out}}_{jk})\Delta t$ and the rest of the integration due to the difference between the length of the domain D_{jk} and the length of the cell interface $(\partial C_j)_k$.

In order to compute the spatial integrals in (3-18), Gaussian quadrature can be applied. In our case, the midpoint rule is used for simplicity:

$$\int_{D^+_{jk}} w^n_{jk}\, dV_g + \int_{D^-_{jk}} w^n_{jk}\, dV_g \approx l_{jk}\Delta t[a^{\text{in}}_{jk} u_j(M_{jk}) + a^{\text{out}}_{jk} u_{jk}(M_{jk})]. \tag{3-20}$$

Equations (3-18), (3-19), and (3-20) lead to

$$\lim_{\Delta t\to 0}\overline{w}^{n+1}(D_{jk}) = \frac{l_{jk}}{a^{\text{in}}_{jk}+a^{\text{out}}_{jk}}[a^{\text{in}}_{jk}u_j(M_{jk})+a^{\text{out}}_{jk}u_{jk}(M_{jk})] + \frac{1}{a^{\text{in}}_{jk}+a^{\text{out}}_{jk}}\big[-h(e^2_{jk},u_j(M_{jk}))+h(e^1_{jk},u_j(M_{jk})) + h(e^2_{jk},u_{jk}(M_{jk}))-h(e^1_{jk},u_{jk}(M_{jk}))\big]. \quad (3\text{-}21)$$

Therefore, we find

$$\lim_{\Delta t\to 0}\sum_{k=1}^{m_j}\frac{|D^+_{jk}|}{\Delta t|C_j|}\overline{w}^{n+1}(D_{jk}) = \sum_{k=1}^{m_j}\frac{a^{\text{in}}_{jk}l_{jk}}{|C_j|(a^{\text{in}}_{jk}+a^{\text{out}}_{jk})}[a^{\text{in}}_{jk}u_j(M_{jk})+a^{\text{out}}_{jk}u_{jk}(M_{jk})] + \sum_{k=1}^{m_j}\frac{a^{\text{in}}_{jk}}{|C_j|(a^{\text{in}}_{jk}+a^{\text{out}}_{jk})}\big[-h(e^2_{jk},u_j(M_{jk}))+h(e^1_{jk},u_j(M_{jk})) + h(e^2_{jk},u_{jk}(M_{jk}))-h(e^1_{jk},u_{jk}(M_{jk}))\big]. \quad (3\text{-}22)$$

Now the average value $\overline{w}^{n+1}(D_j)$ will be computed. Equation (2-4) is integrated over the spacetime control volume $D_j\times[t_n,t_{n+1}]$, and after integration by parts and using the divergence theorem to transform the surface integral to a boundary integral and using (3-2), one obtains

$$\overline{w}^{n+1}(D_j) = \frac{1}{|D_j|}\int_{D_j} w^n_j\,dV_g - \frac{1}{|D_j|}\int_{t_n}^{t_{n+1}}\int_{D_j}\nabla\cdot F(x,u)\,dV_g = \frac{1}{|D_j|}\int_{D_j} w^n_j\,dV_g - \frac{\Delta t}{|D_j|}\left(\sum_{k=1}^{m_j}[-h(e^2_{jk},u_j(M_{jk})) + h(e^1_{jk},u_j(M_{jk}))] + O(\Delta t)\right). \quad (3\text{-}23)$$

The last term in (3-23) includes $O(\Delta t)$ since e^1_{jk} and e^2_{jk} are corners of C_j, not D_j.

Using the previous equality,

$$\frac{1}{\Delta t}\left[\frac{|D_j|}{|C_j|}\overline{w}^{n+1}(D_j)-\bar{u}^n_j\right] = \frac{1}{\Delta t}\left\{\frac{1}{|C_j|}\int_{D_j} w^n_j\,dV_g - \frac{\Delta t}{|C_j|}\left(\sum_{k=1}^{m_j}\big[-h(e^2_{jk},u_j(M_{jk})) + h(e^1_{jk},u_j(M_{jk}))\big] + O(\Delta t)\right) - \bar{u}^n_j\right\}, \quad (3\text{-}24)$$

which leads to

$$\lim_{\Delta t\to 0}\frac{1}{\Delta t}\left[\frac{|D_j|}{|C_j|}\overline{w}^{n+1}(D_j)-\bar{u}_j^n\right]=-\frac{1}{|C_j|}\sum_{k=1}^{m_j}a_{jk}^{\text{in}}l_{jk}u_j(M_{jk})$$
$$-\frac{1}{|C_j|}\sum_{k=1}^{m_j}[-h(e_{jk}^2,u_j(M_{jk}))+h(e_{jk}^1,u_j(M_{jk}))]. \quad (3\text{-}25)$$

Equations (3-22) and (3-25) are used together to obtain the semidiscrete form

$$\frac{d\bar{\boldsymbol{u}}_j}{dt}=-\frac{1}{|C_j|}\sum_{k=1}^{m_j}a_{jk}^{\text{in}}l_{jk}u_j(M_{jk})-\frac{1}{|C_j|}\sum_{k=1}^{m_j}[-h(e_{jk}^2,u_j(M_{jk}))+h(e_{jk}^1,u_j(M_{jk}))]$$
$$+\sum_{k=1}^{m_j}\frac{a_{jk}^{\text{in}}l_{jk}}{|C_j|(a_{jk}^{\text{in}}+a_{jk}^{\text{out}})}[a_{jk}^{\text{in}}u_j(M_{jk})+a_{jk}^{\text{out}}u_{jk}(M_{jk})]$$
$$+\sum_{k=1}^{m_j}\frac{a_{jk}^{\text{in}}}{|C_j|(a_{jk}^{\text{in}}+a_{jk}^{\text{out}})}\big[-h(e_{jk}^2,u_j(M_{jk}))+h(e_{jk}^1,u_j(M_{jk}))$$
$$+h(e_{jk}^2,u_{jk}(M_{jk}))-h(e_{jk}^1,u_{jk}(M_{jk}))\big]. \quad (3\text{-}26)$$

This equation can be rewritten in the form

$$\frac{d\bar{\boldsymbol{u}}_j}{dt}=\frac{1}{|C_j|}\sum_{k=1}^{m_j}\frac{a_{jk}^{\text{in}}a_{jk}^{\text{out}}l_{jk}}{a_{jk}^{\text{in}}+a_{jk}^{\text{out}}}(u_{jk}(M_{jk})-u_j(M_{jk}))$$
$$+\frac{a_{jk}^{\text{in}}a_{jk}^{\text{out}}}{|C_j|(a_{jk}^{\text{in}}+a_{jk}^{\text{out}})}\big\{a_{jk}^{\text{in}}[h(e_{jk}^2,u_j(M_{jk}))-h(e_{jk}^1,u_j(M_{jk}))]$$
$$+a_{jk}^{\text{out}}[h(e_{jk}^2,u_{jk}(M_{jk}))-h(e_{jk}^1,u_{jk}(M_{jk}))]\big\}, \quad (3\text{-}27)$$

which can be rewritten as

$$\frac{d\bar{\boldsymbol{u}}_j}{dt}=-\frac{1}{|C_j|}\sum_{k=1}^{m_j}\frac{a_{jk}^{\text{in}}\boldsymbol{H}(u_{jk}(M_{jk}))+a_{jk}^{\text{out}}\boldsymbol{H}(u_j(M_{jk}))}{a_{jk}^{\text{in}}+a_{jk}^{\text{out}}}$$
$$+\frac{1}{|C_j|}\sum_{k=1}^{m_j}\frac{a_{jk}^{\text{in}}a_{jk}^{\text{out}}l_{jk}}{a_{jk}^{\text{in}}+a_{jk}^{\text{out}}}[u_{jk}(M_{jk})-u_j(M_{jk})], \quad (3\text{-}28)$$

where $\boldsymbol{H}(u_j(M_{jk}))$ and $\boldsymbol{H}(u_{jk}(M_{jk}))$ are given by

$$\begin{aligned}\boldsymbol{H}(u_j(M_{jk}))&=-[h(e_{jk}^2,u_j(M_{jk}))-h(e_{jk}^1,u_j(M_{jk}))],\\ \boldsymbol{H}(u_{jk}(M_{jk}))&=-[h(e_{jk}^2,u_{jk}(M_{jk}))-h(e_{jk}^1,u_{jk}(M_{jk}))].\end{aligned} \quad (3\text{-}29)$$

The function $\boldsymbol{H}$ is defined in the form (3-29) in order to be consistent with the total approximate flux through the cell interface as presented by (3-2) in Claim 3.1.

Remark 3.4. If the value of $a_{jk}^{\text{in}} + a_{jk}^{\text{out}}$ in (3-28) is zero or very close to zero (smaller than 10^{-8} in our numerical experiments), we avoid division by zero or by a very small number using the following approximations

$$\frac{a_{jk}^{\text{in}} \boldsymbol{H}(u_{jk}(M_{jk})) + a_{jk}^{\text{out}} \boldsymbol{H}(u_j(M_{jk}))}{a_{jk}^{\text{in}} + a_{jk}^{\text{out}}} \approx \frac{1}{2}\left[\sum_{k=1}^{m_j} \boldsymbol{H}(u_j(M_{jk})) + \sum_{k=1}^{m_j} \boldsymbol{H}(u_{jk}(M_{jk}))\right],$$

$$\frac{a_{jk}^{\text{in}} a_{jk}^{\text{out}}}{|C_j|(a_{jk}^{\text{in}} + a_{jk}^{\text{out}})} \sum_{k=1}^{m_j} l_{jk}[u_{jk}(M_{jk}) - u_j(M_{jk})] \approx 0. \tag{3-30}$$

These approximations are obtained using similar extreme distances of the propagation of the waves at the cell interface inward and outward of the computational cell to define the domains D_j, D_{jk}, and E_{jk}. The semidiscretization (3-28) and (3-29) is a system of ODEs which has to be integrated in time using an accurate and stable temporal scheme. In our numerical examples reported in Section 6, we used the third-order total-variation diminishing Runge–Kutta method.

The geometry-compatible condition. In the semidiscrete form (3-28) and (3-29) of the proposed scheme, if we consider a constant value of the function $u \equiv \bar{u}$, the second term in the right-hand side of (3-28) vanishes. For this constant function, we obtain for each interface cell k

$$u_j(M_{jk}) = u_{jk}(M_{jk}) = \bar{u} \tag{3-31}$$

and

$$\boldsymbol{H}(u_j(M_{jk})) = \boldsymbol{H}(u_{jk}(M_{jk})). \tag{3-32}$$

The first term in the right-hand side of (3-28) becomes

$$-\frac{1}{|C_j|} \sum_{k=1}^{m_j} \frac{a_{jk}^{\text{in}} \boldsymbol{H}(u_{jk}(M_{jk})) + a_{jk}^{\text{out}} \boldsymbol{H}(u_j(M_{jk}))}{a_{jk}^{\text{in}} + a_{jk}^{\text{out}}} = -\frac{1}{|C_j|} \sum_{k=1}^{m_j} \boldsymbol{H}(u_j(M_{jk})). \tag{3-33}$$

Since we have

$$\sum_{k=1}^{m_j} \boldsymbol{H}(u_j(M_{jk})) = \sum_{k=1}^{m_j} \boldsymbol{H}(u_{jk}(M_{jk})) = -\sum_{k=1}^{m_j} [h(e_{jk}^2, \bar{u}) - h(e_{jk}^1, \bar{u})] = 0, \tag{3-34}$$

we conclude that the first term on the right-hand side of (3-28) will be canceled, which confirms that the proposed scheme respects the geometry-compatibility condition.

Remark 3.5. In the formulation of the proposed central-upwind finite volume method, the midpoint rule is used to compute the spatial integrals. The proposed scheme is second-order accurate, and we obtain the error in the form $E \sim C(\Delta x)^2$ where the magnitude of the constant C is also important as well as the order of

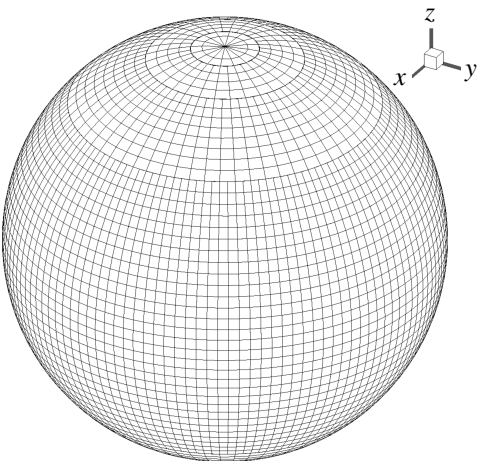


Figure 2. Type of grid used on the sphere.

accuracy of the scheme [30]. For the same second-order accuracy of the proposed schemes using Gaussian quadrature and the midpoint rule, the parameter C obtained using Gaussian quadrature is small in comparison to the value obtained for the case using the midpoint rule. The use of Gaussian quadrature will improve the accuracy of the scheme compared to the midpoint rule. Gaussian quadrature will not have any impact on the geometry-compatibility condition of the proposed scheme.

4. Formulation using the latitude-longitude grid on the sphere

Computational grid on the sphere. The geometry-compatible scheme was developed in the previous section for scalar nonlinear hyperbolic conservation laws using a general grid on the sphere. However, in order to prevent oscillations, an appropriate piecewise linear reconstruction should be proposed according to the computational grid used in the proposed method. In the following, we will describe the computational grid and the nonoscillatory piecewise linear reconstruction used in our numerical experiments. The position of each point on the sphere can be represented by its longitude $\lambda \in [0, 2\pi]$ and its latitude $\phi \in [-\pi/2, \pi/2]$. The grid considered in our numerical examples is shown in Figure 2. The coordinates are singular at the south and north poles, corresponding to $\phi = -\pi/2$ and $\phi = \pi/2$, respectively. The Cartesian coordinates are denoted by $x = (x_1, x_2, x_3)^T \in \mathbb{R}^3$ for standard orthonormal basis vectors $\boldsymbol{i}_1$, $\boldsymbol{i}_2$, and $\boldsymbol{i}_3$.

The unit tangent vectors in the directions of longitude and latitude at each point x on the sphere with coordinates (λ, ϕ) are given by

$$\begin{aligned} \boldsymbol{i}_\lambda &= -(\sin\lambda)\boldsymbol{i}_1 + (\cos\lambda)\boldsymbol{i}_2, \\ \boldsymbol{i}_\phi &= -(\sin\phi)(\cos\lambda)\boldsymbol{i}_1 - (\sin\phi)(\sin\lambda)\boldsymbol{i}_2 + (\cos\phi)\boldsymbol{i}_3. \end{aligned} \tag{4-1}$$

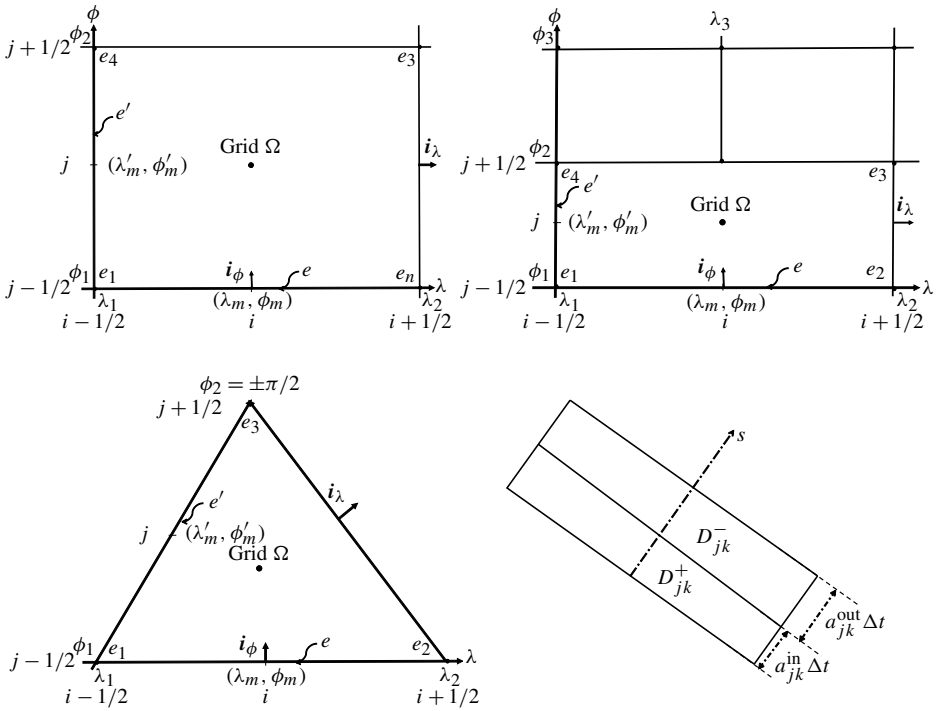


Figure 3. Types of grids used on the sphere. Bottom right: the domain $D_{jk} = D_{jk}^+ \cup D_{jk}^-$.

The unit normal vector to the sphere at the same point $x \in \mathbb{S}^2$ is given by

$$n(x) = (\cos\phi)(\cos\lambda)\boldsymbol{i}_1 + (\cos\phi)(\sin\lambda)\boldsymbol{i}_2 + (\sin\phi)\boldsymbol{i}_3. \tag{4-2}$$

In spherical coordinates, for any vector field F represented by $F = F_\lambda \boldsymbol{i}_\lambda + F_\phi \boldsymbol{i}_\phi$, the equation of conservation law (2-4) can be rewritten as

$$\partial_t u + \frac{1}{\cos\phi}\left(\frac{\partial}{\partial\phi}(F_\phi \cos\phi) + \frac{\partial F_\lambda}{\partial\lambda}\right) = 0. \tag{4-3}$$

The three general structures of the computational cells used as part of the discretization grid on the sphere are shown in Figure 3. When we go from the equator to the north or south poles, the cells are changed by a ratio of 2 at some special latitude circles to reduce the number of cells in order to satisfy the stability condition and to ensure consistency of precision in the entire domain of the sphere. For the stability condition, the CFL number defined as the maximum of the ratio $\nu_j \Delta t / L_j$ is used. The parameter $\nu_j = \max_k(a_{jk}^{\text{out}}, a_{jk}^{\text{in}})$ is the maximum of the directional local speeds of propagation of the waves, and L_j is the minimum length of the computational cell C_j in the longitude and latitude directions. The domain of each cell Ω is defined as $\Omega = \{(\lambda, \phi) : \lambda_1 \le \lambda \le \lambda_2,\ \phi_1 \le \phi \le \phi_2\}$. Near the north or south poles, a "triangular" cell is considered which is a special case of the standard "rectangular" cell shown in Figure 3 with zero length for the side located on the pole.

A nonoscillatory piecewise linear reconstruction. In this section, we describe the piecewise linear reconstruction used in the proposed scheme. For simplicity, in the notations, we will use the indices i and j for the cell centers along the longitude and latitude, respectively (see Figure 3). At each time step t_n, data cell average values $u^n_{i,j}$ in each cell of center (λ_i, ϕ_j) are locally replaced by a piecewise linear function. The obtained reconstruction is

$$u^n_{i,j}(\lambda, \phi) = u^n_{i,j} + (\lambda - \lambda_i)\mu^n_{i,j} + (\phi - \phi_j)\sigma^n_{i,j}, \tag{4-4}$$

where $\mu^n_{i,j}$ and $\sigma^n_{i,j}$ are the slopes in the directions of longitude and latitude, respectively. To prevent oscillations, we propose the following minmod-type reconstruction to obtain the slopes in the longitude and latitude directions:

$$\begin{aligned}
\mu^n_{i,j} &= \text{minmod}\left[\frac{u^n_{i+1,j} - u^n_{i,j}}{\lambda_{i+1} - \lambda_i}, \frac{u^n_{i+1,j} - u^n_{i-1,j}}{\lambda_{i+1} - \lambda_{i-1}}, \frac{u^n_{i,j} - u^n_{i-1,j}}{\lambda_i - \lambda_{i-1}}\right],\\
\sigma^n_{i,j} &= \text{minmod}\left[\frac{u^n_{i,j+1} - u^n_{i,j}}{\phi_{j+1} - \phi_j}, \frac{u^n_{i,j+1} - u^n_{i,j-1}}{\phi_{j+1} - \phi_{j-1}}, \frac{u^n_{i,j} - u^n_{i,j-1}}{\phi_j - \phi_{j-1}}\right],
\end{aligned} \tag{4-5}$$

where the minmod function is defined as

$$\begin{aligned}
&\text{minmod}(\kappa_1, \kappa_2, \kappa_3)\\
&\quad = \begin{cases} \kappa \min(|\kappa_1|, |\kappa_2|, |\kappa_3|) & \text{if } \kappa = \text{sign}(\kappa_1) = \text{sign}(\kappa_2) = \text{sign}(\kappa_3),\\ 0 & \text{otherwise.} \end{cases}
\end{aligned} \tag{4-6}$$

At each step, we compute the average values of the state variable u in the computational cells. The same values are used as the values of u at the cell centers of coordinates (λ_i, ϕ_j). The suitable points, inside the cells which respect these conditions for the linear reconstruction used in this study, should have the spherical coordinates

$$\begin{aligned}
\lambda_i &= \frac{\lambda_1 + \lambda_2}{2},\\
\phi_j &= \frac{\phi_2 \sin(\phi_2) - \phi_1 \sin(\phi_1) + \cos\phi_2 - \cos\phi_1}{\sin\phi_2 - \sin\phi_1},
\end{aligned} \tag{4-7}$$

where λ_1, λ_2, ϕ_1, and ϕ_2 correspond to the longitude and latitude coordinates of the cell nodes as shown in Figure 3.

5. Geometry-compatible flux vectors and particular solutions of interest

Classes of geometry-compatible flux vectors. We have introduced, in [3], two classes of flux vector fields for (2-9). In this classification, the structure of the potential function $h(x, u)$ was used to distinguish between *foliated* and *generic* fluxes. In the proposed classification, the parametrized level sets defined by $\Gamma_{C,u} = \{x \in \mathbb{R}^3 : h(x, u) = C\}$, where $C \in \mathbb{R}$, are used for the flux vector $F(x, u) =$

$n(x) \wedge \nabla h(x, u)$ associated to the potential function h. The flux F is called a *foliated flux field* if the associated family of level sets $\{\Gamma_{C,u}\}_{C \in \mathbb{R}}$ in $\mathbb{R}^3$ is independent of the state variable u. In other words, for any two parameters u_1 and u_2, one can find two real numbers C_1 and C_2 such that $\Gamma_{C_1,u_1} = \Gamma_{C_2,u_2}$. For the generic flux field, the potential function $h = h(x, u)$ does not have this structure.

The dependency of the potential function on the space variable x generates the propagation of the waves, while the dependency on the state variable u leads to the formation of shocks in the solutions. The foliated flux with linear behavior generates the spatially periodic solutions while the foliated flux with nonlinear behavior can generate nontrivial stationary solutions. In our analysis in [3], we have concluded that the new classification introduced and the character of linearity of the flux are sufficient to predict the late-time asymptotic behavior of the solutions. For a linear foliated flux, the solutions are simply transported along the level sets. The generic flux generates large variations in solutions, which converge to constant values within independent domains on the sphere. For the nonlinear foliated flux, the solution converges to its constant average in each level set. For this flux, any steady state solution should be constant along each level set. This type of nontrivial stationary solutions are used in our numerical experiments to demonstrate the performance of the proposed central-upwind finite volume method.

Particular solutions of interest. The nontrivial steady state solutions which will be used in our numerical experiments are obtained using nonlinear foliated fluxes. We are particularly interested in nonlinear foliated fluxes based on a scalar potential function of the form

$$h(x, u) = \varphi(x \cdot a) f(u), \tag{5-1}$$

where $x \cdot a$ denotes the scalar product of the vector x and some constant vector $a = (a_1, a_2, a_3)^T \in \mathbb{R}^3$, while f is a function of the state variable u and φ is a function of one variable. This scalar potential function leads to the gradient-type flux vector field $\Phi(x, u) = \varphi'(x \cdot a) f(u) a$, where φ' is the derivative of the function φ. The flux is obtained using (2-2) as

$$F(x, u) = \varphi'(x \cdot a) f(u) n(x) \wedge a. \tag{5-2}$$

For this foliated flux vector and any function $\tilde{u}$ which depends on one variable, the function defined as $u_0(x) = \tilde{u}(x \cdot a) = \tilde{u}(a_1 x_1 + a_2 x_2 + a_3 x_3)$ is a steady state solution to the conservation law (2-9) associated to the flux vector $F(x, u)$. Arbitrary functions φ and values of the vector a are used to construct nonlinear foliated fluxes and the corresponding nontrivial stationary solutions. In the following, ∇ will be used as the standard gradient operator defined using the variable x and, if other variables are used, they will be specified in the notation by ∇_y for the gradient operator using any other variable y.

In order to prove that the function $u_0(x)$ is a steady state solution of (2-9), the Claim 3.2 in [5] will be used. This claim states that, for any smooth function $h(x,u)$ defined on $\mathbb{S}^2$ with the associated gradient $\Phi = \nabla h$, if the function u_0 defined on $\mathbb{S}^2$ satisfies the condition $\nabla_y h(y, u_0(x))_{|y=x} = \nabla H(x)$, where H is a smooth function defined in a neighborhood of $\mathbb{S}^2$, then the function u_0 is a steady state solution of the conservation law (2-9) associated to the flux vector $F(x,u) = n(x) \wedge \Phi(x,u)$. This result will be used to prove the following corollary related to nontrivial stationary solutions which are obtained using the nonlinear foliated flux vectors.

Claim 5.1 (a family of steady state solutions). *Consider the foliated flux vector $F(x,u) = n(x) \wedge \Phi(x,u)$ with $\Phi = \nabla h$ and $h(x,u) = \varphi(x \cdot a) f(u)$, where $a = (a_1, a_2, a_3)^T \in \mathbb{R}^3$, f is a function of the state variable u, and the function φ depends on one variable. For any function $\tilde{u}$ which depends on one variable, the function defined as $u_0(x) = \tilde{u}(x \cdot a) = \tilde{u}(a_1x_1 + a_2x_2 + a_3x_3)$ is a steady state solution to the conservation law (2-9) associated to the flux $F(x,u)$.*

Proof. We consider the function

$$H(x) = H_0(a_1x_1 + a_2x_2 + a_3x_3), \tag{5-3}$$

where H_0 is defined by

$$H_0(\mu) = \int_{\mu_0}^{\mu} \varphi'(\mu) f(\tilde{u}(\mu))\, d\mu, \tag{5-4}$$

for some reference value μ_0.

The function $h(x,u) = \varphi(x \cdot a) f(u)$ is smooth in $\mathbb{R}^3$, and one obtains

$$\nabla_y h(y, u_0(x))_{|y=x} = \varphi'(x \cdot a) f(\tilde{u}(x \cdot a)) \sum_{k=1}^{k=3} a_k i_k, \tag{5-5}$$

which leads to

$$\nabla_y h(y, u_0(x))_{|y=x} = \nabla H(x). \tag{5-6}$$

As mentioned before, according to Claim 3.2 in [5], the condition (5-6) is sufficient to conclude that the function $u_0(x)$ is a steady state solution of the conservation law (2-9). □

We will consider the nonlinear foliated flux vectors based on the scalar potential functions of the form $h(x,u) = \varphi(x_1) f(u)$, where the function φ is not constant. For this flux, any nonconstant function which depends on x_1 only is a nontrivial steady state solution of (2-9). Another form of nonlinear foliated flux is used in our numerical tests which is obtained by using the scalar potential function of the form $h(x,u) = \varphi(x_1 + x_2 + x_3) f(u)$. This case leads to steady state solutions of the form $u_0(x) = \tilde{u}(x_1 + x_2 + x_3)$. In this paper we will consider discontinuous steady state solutions to test the performance of the proposed central-upwind method. Claim 5.1

will be used to obtain discontinuous steady state solutions for some particular flux vector fields. We will use the nonlinear foliated flux vectors which are obtained by using the scalar potential function of the form $h(x,u)=\varphi(x\cdot a)f(u)$, where $f(u)=u^2/2$. For these flux vectors, the function defined as $u_0(x)=\chi(x\cdot a)\tilde{u}(x\cdot a)$ is a discontinuous stationary solution of (2-9), where $\chi(x\cdot a)=\pm 1$.

In Tests 7 and 8, the proposed central-upwind scheme is employed to compute *confined solutions* of the conservation law (2-9). In these cases, we consider the flux vector $F(x,u)$ which vanishes outside a domain Θ in the sphere $\mathbb{S}^2$. If the initial condition $u_0(x)$ vanishes outside of Θ, then the solution should vanish outside the domain Θ for all time. However, the solution can evolve inside the domain Θ depending on the type of flux and the initial condition considered inside of Θ. This case is observed in Test 7 presented in Section 6, where we choose the initial condition which is not stationary inside the domain Θ but vanishes outside this domain. In Test 8, we will consider the flux vector $F(x,u)$ which vanishes outside Θ and is defined inside this domain using the scalar potential function $h(x,u)=\varphi(x\cdot a)f(u)$. This leads to a flux vector F which satisfies the conditions mentioned in Claim 5.1. For this case, we will consider an initial condition of the form $u_0(x)=\tilde{u}(x\cdot a)$ inside a domain Θ and that vanishes outside this domain. The solution should be stationary inside Θ and should vanish outside this domain.

6. Numerical experiments

In this section, we demonstrate the performance of the proposed central-upwind scheme on a variety of numerical examples. Different types of nonlinear foliated fluxes are used to construct some particular and interesting solutions. In Example 6.1, four numerical tests are performed using different discontinuous steady state solutions of the conservation law (2-9) with the nonlinear foliated flux vectors based on the scalar potential functions of the form $h(x,u)=\varphi(x_1)f(u)$. In Example 6.2, two numerical tests are performed using different discontinuous steady state solutions in the spherical cap of (2-9) which are obtained by using the nonlinear foliated flux corresponding to the scalar potential function of the form $h(x,u)=\varphi(x_1+x_2+x_3)f(u)$. In Example 6.3, two numerical tests are performed where the proposed scheme is employed to compute confined solutions.

Example 6.1 (discontinuous steady state solutions). First, we consider the potential function $h(x,u)=x_1 f(u)$, where $f(u)=u^2/2$, which leads to the nonlinear foliated flux vector $F(x,u)=f(u)n(x)\wedge i_1$. We take the following discontinuous steady state solution of (2-9) as initial condition (Test 1):

$$u_2(x)=\begin{cases}\gamma x_1^3 & \text{if } -1\le x_1\le 0.5,\\ -\gamma x_1^2/(2x_1+1) & \text{if } 0.5\le x_1\le 1,\end{cases} \tag{6-1}$$

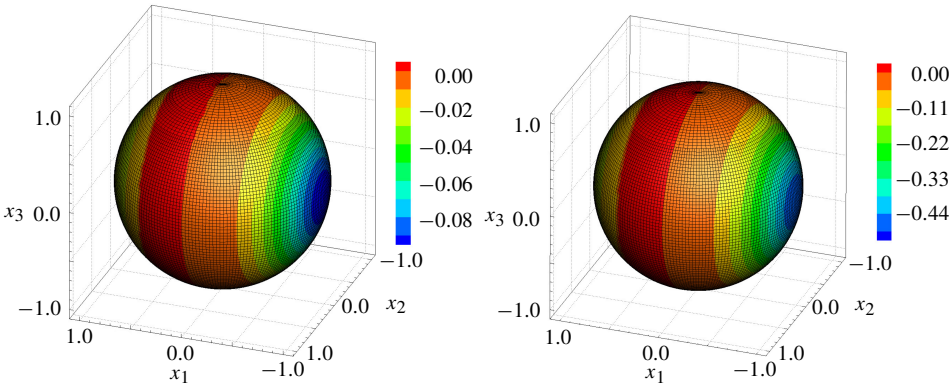


Figure 4. Solutions on the entire sphere at time $t = 5$ for Test 1 (left) and Test 2 (right).

where γ is an arbitrary constant which controls the amplitude and shocks of the solution. This solution has a single closed curve of discontinuity on the sphere.

The numerical solution is computed using a grid with an equatorial longitude step $\Delta\lambda = \pi/96$ and a latitude step $\Delta\phi = \pi/96$ and CFL $= 0.1$. Figure 4, left, shows the numerical solution with $\gamma = 0.1$ which is computed using the proposed scheme at a global time $t = 5$. The numerical solution remains nearly unchanged in time using the proposed scheme. The numerical solution error defined by using the L^2 norm is computed by summation over all grid cells on the sphere. For Test 1, the error is $u_{\text{error}} = 1.5 \times 10^{-4}$ at time $t = 5$, which is small compared to the full range of the numerical solution $u_{\max} - u_{\min} = 0.1$.

Another test is performed using the steady state solution (6-1) as the initial condition with $\gamma = 0.5$ (Test 2) and the same computational grid used in Test 1 and CFL $= 0.6$. As shown in Figure 4, right, the solution remains nearly unchanged up to a global time $t = 5$. The error using the L^2 norm is $u_{\text{error}} = 2.7 \times 10^{-3}$, which is small compared to the full range of the solution $u_{\max} - u_{\min} = 0.5$.

Now we consider a new test (Test 3) using the following steady state solution, with more discontinuities, which is defined in three domains separated by two closed curves on the sphere:

$$u_2(x) = \begin{cases} \gamma x_1^4 & \text{if} \quad -1 \le x_1 \le -0.5, \\ 0.5\gamma x_1^3 & \text{if} \quad -0.5 < x_1 < 0.5, \\ -0.25\gamma x_1^2 & \text{if} \quad 0.5 \le x_1 \le 1. \end{cases} \tag{6-2}$$

The numerical solution is computed using CFL $= 0.1$ and the same grid on the sphere used in the previous tests. As shown in Figure 5, left, the numerical solution which is obtained at time $t = 5$ using the proposed method based on the initial condition (6-2) with $\gamma = 0.1$ remains nearly unchanged. The error is $u_{\text{error}} = 9.6 \times 10^{-5}$, which is small compared to the full range $u_{\max} - u_{\min} = 0.1$.

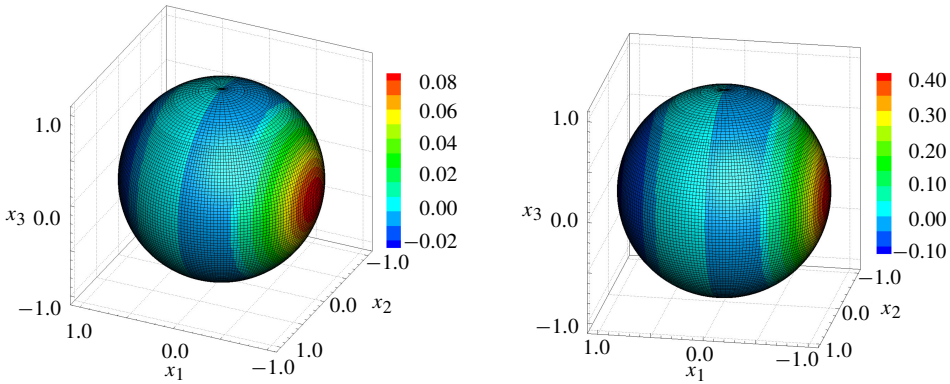


Figure 5. Solutions on the entire sphere at time $t = 5$ for Test 3 (left) and Test 4 (right).

For $\gamma = 0.5$ (Test 4), we used the same computational grid and CFL $= 0.6$. As is shown in Figure 5, right, again for this test the numerical solution at time $t = 5$ remains nearly unchanged. The error using the L^2 norm is $u_{\text{error}} = 1.9 \times 10^{-3}$, which is small compared to the full range of the solution $u_{\max} - u_{\min} = 0.6$.

Example 6.2 (discontinuous steady state solutions in a spherical cap). In the following, the performance of the proposed finite volume method will be analyzed using some particular steady state solutions in a spherical cap. The scalar potential function $h(x, u) = (x_1 + x_2 + x_3) f(u)$ is considered with $f(u) = u^2/2$. This leads to the nonlinear foliated flux $F(x, u) = f(u)n(x) \wedge (i_1 + i_2 + i_3)$. The function of the form $u(x) = \chi(\theta)\tilde{u}(\theta)$ is a steady state solution of (2-9), where $\tilde{u}$ is an arbitrary real function depending on one variable and $\theta = x_1 + x_2 + x_3$. In this numerical example (Test 5), the following discontinuous steady state solution is considered as the initial condition:

$$u(0, x) = \begin{cases} 0.1/(\theta + 2) & \text{if } 0 \le \theta, \\ -0.1/(\theta - 2) & \text{otherwise.} \end{cases} \tag{6-3}$$

The numerical solution is computed by using a grid with an equatorial longitude step $\Delta\lambda = \pi/96$ and a latitude step $\Delta\phi = \pi/96$ and CFL $= 0.1$. Figure 6, left, shows the numerical solution, which remains nearly unchanged in time after being subjected to integration up to a global time $t = 5$ by the proposed scheme. The numerical solution error defined by using the L^2 norm is $u_{\text{error}} = 1.3 \times 10^{-3}$, which is small compared to the full range $u_{\max} - u_{\min} = 0.1$. The following numerical example (Test 6) is performed using the same nonlinear foliated flux considered in Test 5 and the steady state solution with more discontinuities defined by

$$u(0, x) = \begin{cases} 0.2\theta^3 & \text{if } 0.5 \le \theta, \\ 0.1\theta^2 & \text{if } \theta \le -0.5, \\ -0.025 & \text{otherwise.} \end{cases} \tag{6-4}$$

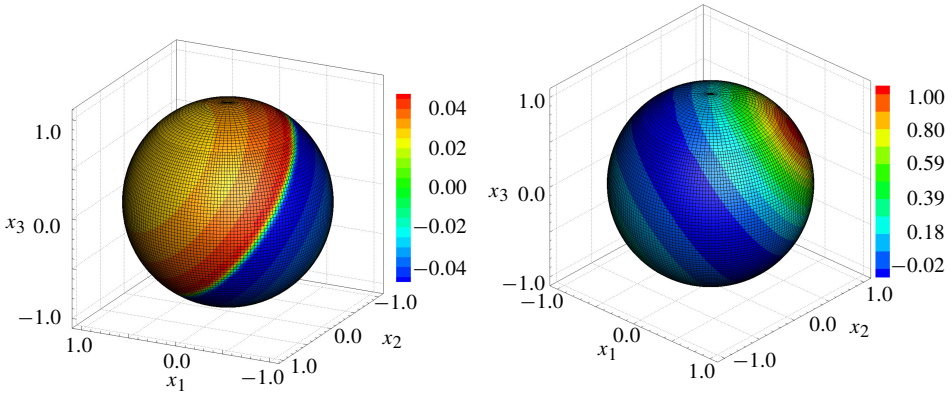


Figure 6. Solutions on the entire sphere at time $t = 5$ for Test 5 (left) and Test 6 (right).

The numerical solution is computed using the same grid used in Test 5 and CFL $= 0.9$. Figure 6, right, shows the numerical solution at time $t = 5$, which remains stationary with the error $u_{\text{error}} = 1.8 \times 10^{-3}$, negligible compared to the full range of the solution $u_{\max} - u_{\min} = 1.06$.

Example 6.3 (confined solutions). In this part, two numerical tests are performed using confined solutions of the conservation law (2-9) based on the flux vector which is obtained using the potential function

$$h(x, u) = \begin{cases} x_1^2 f_1(u) & \text{if } x_1 \le 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6-5}$$

In Test 7, we consider the function

$$u(x, 0) = \begin{cases} 0.1(1 + x_2^2)x_1 & \text{if } x_1 \le 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6-6}$$

The solution of the conservation law (2-9), which is obtained using the function (6-6) as the initial condition, is confined, and it vanishes outside the domain $x_1 \le 0$. The numerical solution is computed using the proposed scheme with an equatorial longitude step $\Delta\lambda = \pi/96$, a latitude step $\Delta\phi = \pi/96$, and CFL $= 0.1$. Figure 7, left, shows the numerical solution at time $t = 5$. The solution evolves in time inside the domain $x_1 \le 0$, but it vanishes outside this domain, which is in good agreement with the evolution of the analytical solution.

In the second numerical test (Test 8), we consider an initial condition which is a confined solution and steady state inside the domain $x_1 \le 0$. The following initial condition is considered:

$$u(x, 0) = \begin{cases} 0.1x_1 & \text{if } x_1 \le 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6-7}$$

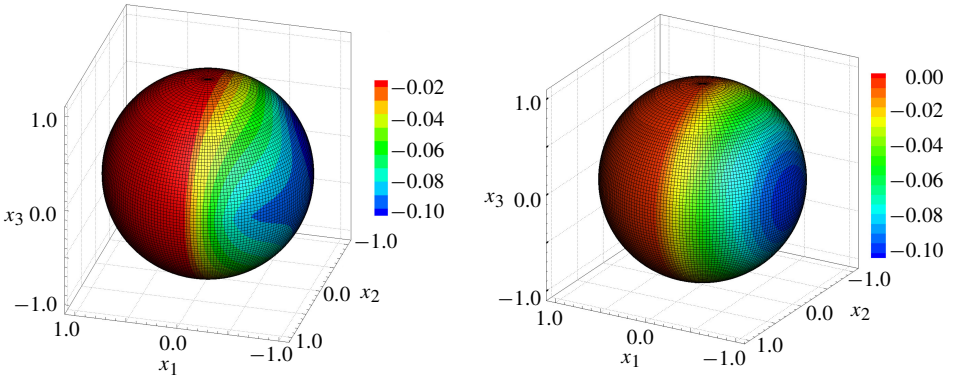


Figure 7. Solutions on the entire sphere at time $t = 5$ for Test 7 (left) and Test 8 (right)

The numerical solution is computed using the proposed central-upwind scheme with the same grid and CFL number which are used in Test 7. Figure 7, right, shows the numerical solution at time $t = 5$. The solution remains steady state in the domain $x_1 \leq 0$, and it vanishes outside this domain for all time as does the initial condition, which is in good agreement with the evolution of the analytical solution. The L^2 error of the numerical solution over the sphere is $u_{\text{error}} = 9.6 \times 10^{-5}$ at time $t = 5$, which is small compared to the full range of the solution $u_{\max} - u_{\min} = 0.1$.

7. Concluding remarks

We have introduced a new geometry-preserving, central-upwind scheme for the discretization of hyperbolic conservation laws posed on the sphere. The main advantage of the proposed scheme is its simplicity since it does not use any Riemann solver and, moreover, the semidiscrete form of our scheme is strongly connected to the geometry of the sphere. The use of Gaussian quadrature will improve the accuracy of the proposed method compared to the midpoint rule using the same computational grid. The Gaussian quadrature does not have an impact on the geometry-compatibility condition of the scheme.

A nonoscillatory reconstruction is used in which the gradient of each variable is computed using a minmod function in order to ensure stability. Our numerical experiments demonstrate the ability of the proposed scheme to avoid oscillations. The performance of the second-order version was tested using relevant numerical examples, and the results clearly demonstrated the scheme's potential and its ability to resolve discontinuous solutions to conservation laws posed on a curved geometry.

We observe that the formulation of the semidiscrete formulation is based on some approximations and assumptions. The scheme is more suitable for discontinuous solutions with shocks of average amplitude. However, the proposed method has the advantage of simplicity compared to the class of upwind schemes. As previously

mentioned, the first advantage is that the proposed scheme is free of any Riemann solver. The second advantage is related to the resolution: we do not use the splitting approach which is often applied in upwind schemes as a simplification technique in order to be able to solve the Riemann problems. This again renders the proposed numerical scheme less expensive compared to upwind methods.

The scheme we have developed here for nonlinear hyperbolic conservation laws could be extended to multidimensional hyperbolic conservation laws and to shallow water models posed on the sphere by extending the methodology in [4; 7] in which central-upwind schemes for solving two-dimensional Cartesian systems for shallow water models were designed. For shallow water systems, instead of the geometric compatibility condition used in the present study, the so-called C-property related to stationary solutions and introduced in [38] should be used in designing a well-balanced central-upwind scheme.

Acknowledgments

The second author was partially supported by the Innovative Training Networks grant 642768 ModCompShock.

References

[1] P. Amorim, M. Ben-Artzi, and P. G. LeFloch, *Hyperbolic conservation laws on manifolds: total variation estimates and the finite volume method*, Methods Appl. Anal. **12** (2005), no. 3, 291–323.

[2] P. Amorim, P. G. LeFloch, and B. Okutmustur, *Finite volume schemes on Lorentzian manifolds*, Commun. Math. Sci. **6** (2008), no. 4, 1059–1086.

[3] A. Beljadid, P. G. LeFloch, and A. Mohammadian, *A geometry-preserving finite volume method for conservation laws on curved geometries*, preprint, 2013.

[4] A. Beljadid, A. Mohammadian, and A. Kurganov, *Well-balanced positivity preserving cell-vertex central-upwind scheme for shallow water flows*, Comput. Fluids **136** (2016), 193–206.

[5] M. Ben-Artzi, J. Falcovitz, and P. G. LeFloch, *Hyperbolic conservation laws on the sphere: a geometry-compatible finite volume scheme*, J. Comput. Phys. **228** (2009), no. 16, 5650–5668.

[6] M. Ben-Artzi and P. G. LeFloch, *Well-posedness theory for geometry-compatible hyperbolic conservation laws on manifolds*, Ann. Inst. H. Poincaré (C) **24** (2007), no. 6, 989–1008.

[7] S. Bryson, Y. Epshteyn, A. Kurganov, and G. Petrova, *Well-balanced positivity preserving central-upwind scheme on triangular grids for the Saint-Venant system*, ESAIM Math. Model. Numer. Anal. **45** (2011), no. 3, 423–446.

[8] T. Ceylan, P. G. LeFloch, and B. Okutmustur, *The relativistic Burgers equation on a FLRW background and its finite volume approximation*, preprint, 2015. arXiv

[9] G. Dziuk and C. M. Elliott, *Finite elements on evolving surfaces*, IMA J. Numer. Anal. **27** (2007), no. 2, 262–292.

[10] G. Dziuk, D. Kröner, and T. Müller, *Scalar conservation laws on moving hypersurfaces*, Interfaces Free Bound. **15** (2013), no. 2, 203–236.

[11] M. Espedal, A. Fasano, and A. Mikelić, *Filtration in porous media and industrial application: lectures given at the 4th session of the Centro Internazionale Matematico Estivo*, Lect. Notes Math., no. 1734, Springer, Berlin, 2000.

[12] J. Giesselmann, *A convergence result for finite volume schemes on Riemannian manifolds*, M2AN Math. Model. Numer. Anal. **43** (2009), no. 5, 929–955.

[13] J. Giesselmann and P. G. LeFloch, *Formulation and convergence of the finite volume method for conservation laws on spacetimes with boundary*, preprint, 2016. arXiv

[14] A. Kurganov and D. Levy, *A third-order semidiscrete central scheme for conservation laws and convection-diffusion equations*, SIAM J. Sci. Comput. **22** (2000), no. 4, 1461–1488.

[15] ———, *Central-upwind schemes for the Saint-Venant system*, M2AN Math. Model. Numer. Anal. **36** (2002), no. 3, 397–425.

[16] A. Kurganov, S. Noelle, and G. Petrova, *Semidiscrete central-upwind schemes for hyperbolic conservation laws and Hamilton–Jacobi equations*, SIAM J. Sci. Comput. **23** (2001), no. 3, 707–740.

[17] A. Kurganov and G. Petrova, *Central-upwind schemes on triangular grids for hyperbolic systems of conservation laws*, Numer. Methods Partial Differential Equations **21** (2005), no. 3, 536–552.

[18] ———, *A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system*, Commun. Math. Sci. **5** (2007), no. 1, 133–160.

[19] A. Kurganov, G. Petrova, and B. Popov, *Adaptive semidiscrete central-upwind schemes for nonconvex hyperbolic conservation laws*, SIAM J. Sci. Comput. **29** (2007), no. 6, 2381–2401.

[20] A. Kurganov and E. Tadmor, *New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations*, J. Comput. Phys. **160** (2000), no. 1, 241–282.

[21] ———, *New high-resolution semi-discrete central schemes for Hamilton–Jacobi equations*, J. Comput. Phys. **160** (2000), no. 2, 720–742.

[22] P. G. LeFloch, *Hyperbolic systems of conservation laws: the theory of classical and nonclassical shock waves*, Birkhäuser, Basel, 2002.

[23] ———, *Hyperbolic conservation laws on spacetimes*, Nonlinear conservation laws and applications (A. Bressan, G.-Q. G. Chen, M. Lewicka, and D. Wang, eds.), IMA Vol. Math. Appl., no. 153, Springer, New York, 2011, pp. 379–391.

[24] ———, *Structure-preserving shock-capturing methods: late-time asymptotics, curved geometry, small-scale dissipation, and nonconservative products*, Advances in numerical simulation in physics and engineering: lecture notes of the XV 'Jacques-Louis Lions' Spanish–French School (C. Parés, C. Vázquez, and F. Coquel, eds.), SEMA SIMAI, no. 3, Springer, Cham, 2014, pp. 179–222.

[25] P. G. LeFloch and H. Makhlof, *A geometry-preserving finite volume method for compressible fluids on Schwarzschild spacetime*, Commun. Comput. Phys. **15** (2014), no. 3, 827–852.

[26] P. G. LeFloch, H. Makhlof, and B. Okutmustur, *Relativistic Burgers equations on curved spacetimes: derivation and finite volume approximation*, SIAM J. Numer. Anal. **50** (2012), no. 4, 2136–2158.

[27] P. G. LeFloch and B. Okutmustur, *Hyperbolic conservation laws on spacetimes: a finite volume scheme based on differential forms*, Far East J. Math. Sci. **31** (2008), no. 1, 49–83.

[28] R. Lenormand, *Pattern growth and fluid displacements through porous media*, Physica A **140** (1986), no. 1–2, 114–123.

[29] R. Lenormand and C. Zarcone, *Role of roughness and edges during imbibition in square capillaries*, conference paper SPE-13264-MS, Society of Petroleum Engineers, 1984.

[30] R. J. LeVeque, *Finite volume methods for hyperbolic problems*, Cambridge University, 2002.

[31] J. D. Logan, *An introduction to nonlinear partial differential equations*, Wiley, New York, 1994.

[32] A. Meister and J. Struckmeier, *Central schemes and systems of balance laws*, Hyperbolic partial differential equations: theory, numerics and applications, Vieweg, Braunschweig, 2002, pp. 59–114.

[33] K. W. Morton and T. Sonar, *Finite volume methods for hyperbolic conservation laws*, Acta Numer. **16** (2007), 155–238.

[34] H. Nessyahu and E. Tadmor, *Nonoscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys. **87** (1990), no. 2, 408–463.

[35] J. A. Rossmanith, D. S. Bale, and R. J. LeVeque, *A wave propagation algorithm for hyperbolic systems on curved manifolds*, J. Comput. Phys. **199** (2004), no. 2, 631–662.

[36] C.-W. Shu and S. Osher, *Efficient implementation of essentially nonoscillatory shock-capturing schemes*, J. Comput. Phys. **77** (1988), no. 2, 439–471.

[37] N. Su, J. P. C. Watt, K. W. Vincent, M. E. Close, and R. Mao, *Analysis of turbulent flow patterns of soil water under field conditions using burgers equation and porous suction-cup samplers*, Aust. J. Soil Res. **42** (2004), no. 1, 9–16.

[38] M. E. Vázquez-Cendón, *Improved treatment of source terms in upwind schemes for the shallow water equations in channels with irregular geometry*, J. Comput. Phys. **148** (1999), no. 2, 497–526.

Received March 28, 2016. Revised December 31, 2016.

ABDELAZIZ BELJADID: beljadid@mit.edu
Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, United States

PHILIPPE G. LEFLOCH: contact@philippelefloch.org
Laboratoire Jacques-Louis Lions & Centre National de la Recherche Scientifique, Université Pierre et Marie Curie (Paris 6), 4 Place Jussieu, 75258 Paris, France

COMM. APP. MATH. AND COMP. SCI.
Vol. 12, No. 1, 2017
dx.doi.org/10.2140/camcos.2017.12.109

msp

TIME-PARALLEL GRAVITATIONAL COLLAPSE SIMULATION

ANDREAS KREIENBUEHL, PIETRO BENEDUSI,
DANIEL RUPRECHT AND ROLF KRAUSE

This article demonstrates the applicability of the parallel-in-time method Parareal to the numerical solution of the Einstein gravity equations for the spherical collapse of a massless scalar field. To account for the shrinking of the spatial domain in time, a tailored load balancing scheme is proposed and compared to load balancing based on number of time steps alone. The performance of Parareal is studied for both the subcritical and black hole case; our experiments show that Parareal generates substantial speedup and, in the supercritical regime, can reproduce Choptuik's black hole mass scaling law.

1. Introduction

Einstein's field equations of general relativity (GR) consist of ten coupled, nonlinear, hyperbolic-elliptic partial differential equations (PDEs). Because gravity couples to all forms of energy, there is an enormous dynamic range of spatiotemporal scales in GR. Hence, usually only the application of advanced numerical methods can provide solutions and in numerical relativity [1; 3] extensive use of high-performance computing (HPC) is made [32; 26].

Today, almost all HPC architectures are massively parallel systems connecting large numbers of compute nodes by a high-speed interconnect. In numerical simulations, the power of these systems can only be harnessed by algorithms that feature a high degree of concurrency; every algorithm with strong serial dependencies can only provide inferior performance on massively parallel computers. For the solution of PDEs, parallelization strategies have been developed mainly for spatial solvers. However, in light of the rapid increase in the number of cores in supercomputers, methods that offer additional concurrency along the temporal axis have recently begun to receive more attention.

The idea of parallelization in time was introduced in 1964 [35]. In the 1980s and 1990s, time and spacetime multigrid methods were studied [22; 23; 24]. More recently, the now widely used time-parallel method Parareal was proposed [31].

MSC2010: 35Q76, 65M25, 65Y05, 83C57.
Keywords: Einstein–Klein–Gordon gravitational collapse, Choptuik scaling, Parareal, spatial coarsening, load balancing, speedup.

Other recently introduced parallel-in-time methods are PFASST [33; 12], RIDC [9], or MGRIT [13]. A historical overview is offered in [17].

Given the demonstrated potential of parallel-in-time integration methods for large-scale parallel simulations [42], these methods could be beneficial for the numerical relativity community. However, their application is not straightforward and often it is unclear a priori if good performance can be achieved. In this article, we therefore investigate the *principal applicability* of the time-parallel Parareal method to solving Einstein's equations describing spherical, gravitational collapse of a massless scalar field. The system is also referred to as an Einstein–Klein–Gordon system because it is equivalent to a Klein–Gordon equation expressed in the context of GR, i.e., on a back-reacting, curved geometry. It defines a basic gravitational field theory and is of interest therefore not only in numerical relativity but also in, e.g., quantum gravity [25; 44; 29]. A summary of numerically derived results is given in [21]; the work by Choptuik [7] brought forward novel, physical results and is of particular interest here because we will show that Parareal correctly reproduces the expected mass scaling law.

Mathematical theory shows that Parareal performs well for diffusive problems with constant coefficients [19]. For diffusive problems with space- or time-dependent coefficients, numerical experiments show that Parareal can converge quickly too [30]. However, given the theory for basic constant-coefficient hyperbolic PDEs [19], it can be expected that Parareal applied to convection-dominated problems converges too slowly for meaningful speedup to be possible. Special cases with reasonable performance are discussed in [16], and for certain hyperbolic PDEs it was found that some form of stabilization is required for Parareal to provide speedup [18; 40; 11; 6]. Surprisingly, no stabilization is required for the equations describing gravitational collapse; we demonstrate that plain Parareal can achieve significant speedup. A detailed analytical investigation of why this is the case would definitely be of interest but is left out for future work. One reason could be that we solve in characteristic coordinates for which the discretization is aligned with the directions of propagation [16; 28].

In Section 2 we define the system of Einstein field equations that we solve using Parareal. In addition, we give details on the numerical approach and discuss the interplay between Parareal and the particular structure of the spatial mesh. In Section 3 we discuss the Parareal method. Then, in Section 4 numerical results are presented. Finally, in Section 5 we conclude with a summary and discussion.

2. Equations

2.1. ***Gravitational collapse.*** The Einstein field equations in Planck units normalized to $4\pi G/c^4 = 1$ are

$$G_{\mu\nu} = 2T_{\mu\nu}, \tag{2.1.1}$$

where $\mu, \nu \in \{0, 1, 2, 3\}$ index time (via 0) and space (via 1, 2, and 3).[1] Once the nongravitational matter content is specified by a definition of the energy-momentum tensor $T_{\mu\nu}$, possibly along with equations of state that together satisfy the continuity equations $\nabla^\mu T_{\mu\nu} = 0$, (2.1.1) defines a set of ten partial differential equations for ten unknown metric tensor field components $g_{\mu\nu}$.[2] In all generality, the equations are coupled, nonlinear, and hyperbolic-elliptic in nature. Six of the ten equations are hyperbolic evolution equations, while the remaining four are elliptic constraints on the initial data; they represent the freedom to choose spacetime coordinates. For the matter content, we consider a minimally coupled massless scalar field ϕ with energy-momentum tensor

$$T_{\mu\nu} = \nabla_\mu \phi \, \nabla_\nu \phi - \tfrac{1}{2} g_{\mu\nu} g^{\alpha\beta} \, \nabla_\alpha \phi \, \nabla_\beta \phi. \tag{2.1.2}$$

For the metric tensor field $g_{\mu\nu}$ in spherical symmetry, it is natural to introduce a parametrization in terms of Schwarzschild coordinates (t, r). Here, t is the time coordinate of a stationary observer at infinite radius r, which measures the size of spheres centered at $r = 0$. In [7] the resulting Einstein field equations are analyzed numerically. In particular, adaptive mesh refinement [4] is used to resolve the black hole formation physics. In [20] the same investigation is carried out in double null or characteristic coordinates (τ, ρ) without mesh refinement (see, however, [39; 43]). Finally, in [29] the effect of quantum gravity modifications on the collapse is studied in adjusted characteristic coordinates. Here we use characteristic coordinates (τ, ρ) as well but exclude quantum gravity modifications. Also, for simplicity, we will refer to τ as a time coordinate and to ρ as a space coordinate.

Making the ansatz

$$g_{\mu\nu} \, \mathrm{d}x^\mu \, \mathrm{d}x^\nu = -2\partial_\rho r H \, \mathrm{d}\tau \, \mathrm{d}\rho + r^2 (\mathrm{d}\vartheta^2 + [\sin(\vartheta) \, \mathrm{d}\varphi]^2) \tag{2.1.3}$$

for the metric tensor field and using an auxiliary field h for the spacetime geometry along with an auxiliary field Φ for the matter content, the complete field equations are

$$\partial_\tau r = -\tfrac{1}{2} h, \qquad \partial_\tau \Phi = \frac{(H - h)(\Phi - \phi)}{2r} \tag{2.1.4}$$

for r and Φ and

$$\partial_\rho \phi = \frac{\partial_\rho r}{r} (\Phi - \phi), \qquad \partial_\rho H = \frac{\partial_\rho r}{r} H (\Phi - \phi)^2, \qquad \partial_\rho h = \frac{\partial_\rho r}{r} (H - h) \tag{2.1.5}$$

for ϕ, H, and h [20]. Overall the system can be seen as a wave equation for the massless scalar field ϕ on a back-reacting, curved geometry. Boundary conditions

[1] We omit the addition of the cosmological constant term $\Lambda g_{\mu\nu}$ on the left-hand side in (2.1.1) because observations suggest $0 < \Lambda \ll 1$ (see, e.g., [27]); the term's impact on black hole formation as studied here can be neglected.

[2] We use the Einstein summation convention.

at $(\tau, \rho = \tau)$ are $r = 0$ and regularity of Φ, ϕ, H, and h, which implies $\Phi = \phi$ and $H = h$ at the boundary [10; 28]. Consistent initial data at $(\tau = 0, \rho)$ are

$$r = \tfrac{1}{2}\rho, \qquad \Phi = (1 + \rho\,\partial_\rho)\phi, \tag{2.1.6}$$

where we choose for ϕ the Gaussian wave packet

$$\phi(0, \rho) = \phi_0 \frac{\rho^3}{1+\rho^3} \exp\left(-\left[\frac{\rho - \rho_0}{\delta_0}\right]^2\right). \tag{2.1.7}$$

We also performed tests for initial data similar in shape to the hyperbolic tangent function much like Choptuik did in [7] for purely serial time stepping. Since in this case we found Parareal's performance to resemble strongly that for the case of the Gaussian wave packet, we do not include these results here. The initial scalar field configuration is thus characterized by an amplitude ϕ_0, mean position ρ_0, and width δ_0. Depending on the value of these parameters, the solution to (2.1.4) and (2.1.5) can describe a bounce of the wave packet or black hole formation near the boundary at $r = 0$. A black hole appears when the outward null expansion

$$\Theta^+ = \frac{1}{r}\sqrt{\frac{2h}{H}}, \tag{2.1.8}$$

which measures the relative rate of change of a cross-sectional area element of a congruence of outgoing null curves, approaches zero [36]. The black hole mass is

$$M = \tfrac{1}{2}r, \tag{2.1.9}$$

evaluated at the point (τ^+, ρ^+) toward which Θ^+ vanishes.

2.2. ***Numerical solution.*** The numerical grid is depicted in Figure 1, left. It is parametrized by the characteristic coordinates τ and ρ, which are used for numerical integration; τ is used as the coordinate representing time and ρ as the coordinate representing space. Integration thus takes place on a right triangle with initial data defined along the lower right-hand leg. Clearly, the spatial domain becomes smaller as the solution is advanced in τ. Note that the domain is not exactly a right triangle because at the upper-most corner a small subtriangle is missing. This "buffer" zone of extent λ is needed for the spatial part of the numerical stencil to fit. The computational domain thus consists of all points $(\tau, \rho) \in [0, L - \lambda] \times [0, L]$ with $L = 80$, $\lambda = 0.625$, and $\rho \geq \tau$.

As a time-stepping method for the solution of the equations in (2.1.4), we use a second-order Lax–Wendroff Richtmyer two-step method on a fine spacetime grid [28]. To employ the time-parallel method Parareal (see Section 3), we need a second, computationally cheap, time-integration method. Here, we choose the explicit first-order Euler method on a coarse spacetime mesh. For Parareal to be efficient, the cost of the coarse method has to be small compared to that of the fine one: by choosing

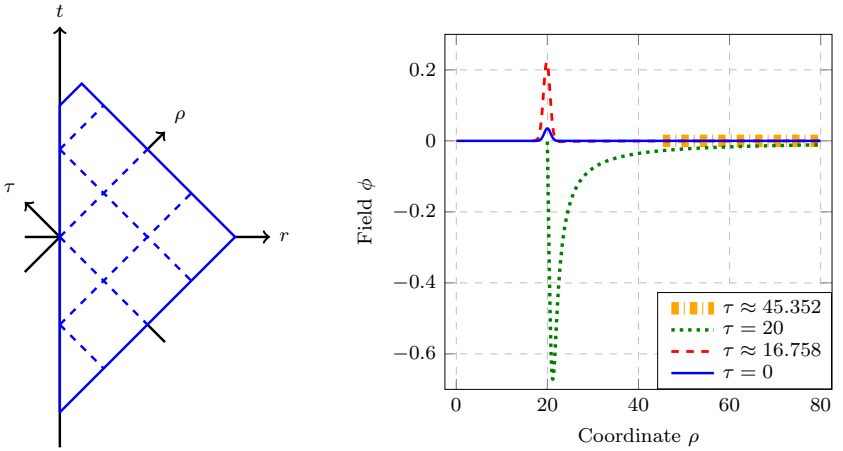


Figure 1. Left: the numerical domain. It is parametrized by the characteristic coordinates τ and ρ. Right: subcritical gravitational scalar field evolution and scalar field solution snapshots for a black-hole-free setting. The peak of the Gaussian evolves along the constant coordinate value $\rho \approx 20$, which is also when the bounce occurs in τ.

a simple first-order method on the coarse grid for $\mathscr{C}$, we obtain a good coarse-to-fine ratio (see Section 3.4). For optimal speedup, the right balance between the difference in accuracy and difference in cost between $\mathscr{C}$ and $\mathscr{F}$ has to be found.

For the integration in space of the equations in (2.1.5), we use a second-order Runge–Kutta method [28]. Snapshots of scalar field evolution resulting from the chosen fine grid discretization are shown in Figure 1, right, where ϕ evolves along constant lines of ρ until a bounce occurs at $r = 0$. The figure also shows how the size of the domain decreases during the evolution: for $\tau = 0$ the left boundary is at $\rho = 0$ while for $\tau = 20$ it is at $\rho = 20$.

2.3. ***Mass scaling.*** In practice, the simulation terminates when a black hole forms because H grows without bound in this case (see [10] for details). Figure 2, left, provides a simplified illustration of a black hole region (dotted portion) and shows where the simulation comes to a halt (dashed line). Thus, to determine the black hole mass M, we record minimal expansion values via the scalar $(r\Theta^+)_{\mathrm{mi}} = \min_\rho\{r\Theta^+\}$ derived from (2.1.8). The last such recorded minimal value before the termination of the simulation defines a characteristic coordinate (τ^+, ρ^+) (see again Figure 2, left), which we can use to define an r and M via (2.1.9). The scalar $(r\Theta^+)_{\mathrm{mi}}$ approaches 0 when (τ, ρ) nears (τ^+, ρ^+), as is shown in the lower portion of Figure 2, right.

Based on numerical experiments, Choptuik presents, among other things, a relation between the amplitude ϕ_0 of the Gaussian in (2.1.7) and the black hole mass M [7]. He shows that there is a critical value $\phi_0^\star$ such that for $\phi_0 < \phi_0^\star$ there is a bounce (subcritical case), while for $\phi_0 > \phi_0^\star$ there is a black hole (supercritical case).

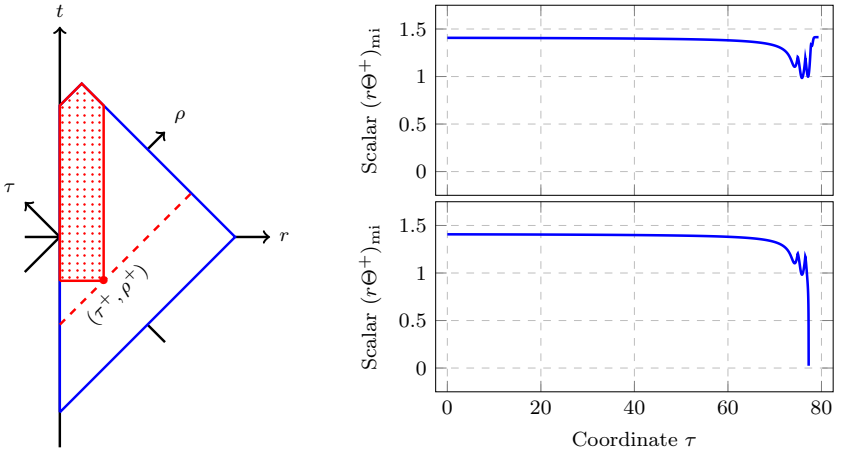


Figure 2. Illustrations to clarify supercritical gravitational collapse. Left: the simulation terminates at τ^+, when a black hole forms at ρ^+. Right: minimal weighted outward null expansion indicating a bounce (top) and black hole formation (bottom) are shown.

Based thereon, he demonstrates that the black hole mass scales with $\phi_0 - \phi_0^\star > 0$ according to the law $M \propto (\phi_0 - \phi_0^\star)^\gamma$ with γ being a positive constant of the same value for various initial data profiles. We demonstrate that Parareal can correctly capture this black hole mass scaling law although our coarse-level Euler method alone cannot. Also, Parareal requires less wall-clock time than $\mathcal{F}$, which can be beneficial for the investigation of the high-accuracy-demanding critical solution [7; 21] that requires the simulation of numerous black holes [20]. This analysis however is omitted in this article and left for future work.

3. Parareal

3.1. ***Algorithm.*** Parareal [31] is a method for the solution of initial value problems

$$\partial_\tau u(\tau) = f(\tau, u(\tau)), \quad u(0) = u_0, \ 0 \le \tau \le T. \tag{3.1.1}$$

Here, as is outlined in the previous section, f comes from discretizing (2.1.4) and (2.1.5), and $T = L - \lambda$ marks the end time. Parareal starts with a decomposition of the time domain into N_{pr} temporal subintervals (TSs) defined in terms of times τ^p such that

$$[\tau^1, \tau^2] \cup \cdots \cup [\tau^{N_{\mathrm{pr}}-1}, \tau^{N_{\mathrm{pr}}}] = [0, L - \lambda]. \tag{3.1.2}$$

Now denote by $\mathcal{F}$ some serial time-integration method of high accuracy and cost (in our case this is the second-order Lax–Wendroff Richtmyer two-step method) and by $\mathcal{C}$ a cheap and possibly much less accurate method (in our case this is the explicit first-order Euler method). Instead of running the fine method subinterval

by subinterval serially in time, Parareal performs the iteration

$$u_{[i+1]}^{p+1} = \mathscr{C}(u_{[i+1]}^{p}) - \mathscr{C}(u_{[i]}^{p}) + \mathscr{F}(u_{[i]}^{p}), \tag{3.1.3}$$

where superscripts index time or process number $p \in \{1, \ldots, N_{\mathrm{pr}}\}$ and subscripts iterations $i \in \{1, \ldots, N_{\mathrm{it}}\}$. The advantage is that the expensive computation of the fine method can be performed in parallel over all TSs at once. Here, we assume that the number of TSs is equal to the number N_{pr} of cores (or *processes*) used for the time direction. Good speedup can be obtained if $\mathscr{C}$ is fast in comparison to $\mathscr{F}$ but still accurate enough for Parareal to converge rapidly. See Section 3.4 for a more detailed discussion of Parareal's speedup.

In Section 2.2 we hinted at the interchangeability of the characteristic coordinates τ and ρ for the numerical integration. Therefore, theoretically, Parareal could *also* be used for the spatial integration to *simultaneously* parallelize both time and space. However, such an interweaving of two Parareal iterations is not discussed in this article; it is put aside for future work.

3.2. ***Spatial coarsening in Parareal.*** In order to make $\mathscr{C}$ cheaper and improve speedup, we not only use a less accurate time stepper for $\mathscr{C}$ but also employ a coarsened spatial discretization with a reduced number of degrees of freedom. Therefore, we need a spatial interpolation $\boldsymbol{I}$ and restriction $\boldsymbol{R}$ operator. In this case (see, e.g., [14]), the Parareal algorithm is given by

$$u_{[i+1]}^{p+1} = \boldsymbol{I}\mathscr{C}(\boldsymbol{R}u_{[i+1]}^{p}) - \boldsymbol{I}\mathscr{C}(\boldsymbol{R}u_{[i]}^{p}) + \mathscr{F}(u_{[i]}^{p}). \tag{3.2.1}$$

As the restriction operator $\boldsymbol{R}$, we use point injection. For the interpolation operator $\boldsymbol{I}$, we use polynomial (i.e., Lagrangian) interpolation of order 3, 5, and 7.[3] It has been shown that, even for simple toy problems, convergence of Parareal can deteriorate if spatial coarsening with low-order interpolation is used. As demonstrated in Section 4.1, this also holds true for the problem studied here.

3.3. ***Implementation.*** We have implemented two different realizations of Parareal. In a "standard" version $\mathscr{P}_{\mathrm{st}}$ (see Listing 1, left), the Parareal correction is computed on each TS up to a uniformly prescribed iteration number. In contrast, in the "modified" implementation $\mathscr{P}_{\mathrm{mo}}$ (see Listing 1, right), Parareal corrections are only performed on TSs where the solution may not yet have converged. Because Parareal always converges at a rate of at least one TS per iteration, we only iterate on a TS if its assigned MPI rank is greater than or equal to the current Parareal iteration number (see line 8 in Listing 1, right). Otherwise, no further iterations are needed or performed, and the process remains idle. Thus, as the iteration progresses, more and

[3] We also tested barycentric interpolation [5; 15] but found the performance in terms of runtimes and speedup (see Sections 3.4 and 4) to be inferior.

```
1  if p > 1 then // Initialization
2    Coarse(co; τ^1 → τ^p)
3  Interp(co ↦ fi[0])
4  if p < N_pr then // Prediction
5    Coarse(co; τ^p → τ^{p+1})
6  Interp(co ↦ fi[2])
7  for i = 1 : N_it do // Iteration
8    if p < N_pr then
9      Fine(fi[0]; τ^p → τ^{p+1})
10   fi[1] = fi[0]
11   fi[1] −= fi[2]
12   if p > 1 then
13     MPI_Recv(fi[0]; p ⇐ p−1)
14   else
15     Init(fi[0])
16   Restrict(fi[0] ↦ co)
17   if p < N_pr then
18     Coarse(co; τ^p → τ^{p+1})
19   Interp(co ↦ fi[2])
20   fi[1] += fi[2]
21   if p < N_pr then
22     MPI_Send(fi[1]; p ⇒ p+1)
```

```
1  if p > 1 then // Initialization
2    Coarse(co; τ^1 → τ^p)
3  Interp(co ↦ fi[0])
4  if p < N_pr then // Prediction
5    Coarse(co; τ^p → τ^{p+1})
6  Interp(co ↦ fi[2])
7  for i = 1 : N_it do // Iteration
8    if p >= i then
9      j = (i+1) % 2
10     k = i % 2
11     if p < N_pr then
12       Fine(fi[j]; τ^p → τ^{p+1})
13     if p > i then
14       MPI_Recv(fi[k]; p ⇐ p−1)
15       fi[j] −= fi[2]
16       Restrict(fi[k] ↦ co)
17       if p < N_pr then
18         Coarse(co; τ^p → τ^{p+1})
19       Interp(co ↦ fi[2])
20       fi[j] += fi[2]
21     if p < N_pr then
22       MPI_Send(fi[j]; p ⇒ p+1)
```

Listing 1. Pseudocode for the standard and modified Parareal implementations. Variable "co" denotes the coarse grid solution and "fi" an array of three fine grid buffers. Left: the standard Parareal implementation $\mathcal{P}_{\text{st}}$. Right: the modified Parareal implementation $\mathcal{P}_{\text{mo}}$.

more processes enter an idle state. In an implementation to be realized in future work, the criterion for convergence used here will be replaced by a check for some residual tolerance [2]. This could negatively affect the observed performance since it requires essentially one more iteration to compute the residual.[4] It also bears mentioning that it has very recently been demonstrated that parallel-in-time integration methods are good candidates to provide algorithm-based fault tolerance [34; 41].

Another difference between the standard and modified implementations is that in the former, after each time-parallel fine evolution, a copy of the fine-grid solution has to be created (see line 10 in Listing 1, left). In the modified Listing 1, right, this copying is circumvented by the use of two alternating indices "j" and "k" in lines 9 and 10, respectively. The iteration number determines their values, which in turn determines the fine-grid solution buffer that is used to send or receive data by means of the corresponding MPI routines (see lines 14 and 22 in Listing 1, right). The two implementations also have slightly different requirements in terms of storage.

[4] In [2] a version of Parareal is discussed that can be used to proceed the integration beyond a given end time. It is based on an optimized scheduling of those tasks which become idle in our implementation.

As can be seen in line 15 in Listing 1, left, in $\mathcal{P}_{\text{st}}$ on the first TS or, equivalently, for the first MPI rank, the fine-grid solution has to be assigned initial data at the beginning of each iteration. This requires one additional buffer to be held in storage. Other than that both implementations need one coarse-grid solution buffer and three fine-grid buffers for each TS.

3.4. ***Speedup.*** We denote by R_{co} the coarse and by R_{fi} the fine time stepper's runtime. Recalling that N_{it} denotes the number of iterations required for Parareal to converge given N_{pr} processes, Parareal's theoretically achievable speedup is

$$S = \left[\left(1 + \frac{N_{\text{it}}}{N_{\text{pr}}}\right)\frac{R_{\text{co}}}{R_{\text{fi}}} + \frac{N_{\text{it}}}{N_{\text{pr}}}\right]^{-1} \leq \min\left\{\frac{N_{\text{pr}}}{N_{\text{it}}}, \frac{R_{\text{fi}}}{R_{\text{co}}}\right\}, \tag{3.4.1}$$

as is discussed, e.g., in [33]. The estimate is valid only for the ideal case, where runtimes across subintervals are perfectly balanced. In the presence of load imbalances in time, however, i.e., differences in the runtimes of $\mathscr{C}$ and $\mathscr{F}$ across TSs, maximum speedup is reduced [30]. Because the spatial domain we consider is shrinking in time, a tailored decomposition of the time axis has to be used to provide well balanced computational load, as is discussed in the next section.

3.5. ***Load balancing.*** Because we integrate over a triangular computational space-time domain (see Figure 1, left), a straightforward, uniform partitioning of the time axis results in imbalanced computational load in time. The first load balancing (LB) strategy, to which henceforth we will refer as LB1, is based on this straightforward, basic decomposition of the time axis. It assigns to each TS the same *number* of time steps without regard to their computational *cost*. Because of the shrinking domain, TSs at later times carry fewer spatial degrees of freedom so that the per-process runtimes R_{co}^p and R_{fi}^p of the coarse and fine time steppers, respectively, are larger for the earlier TSs than for the later ones. Figure 3, left, shows how this partition leads to an imbalanced computational load in time because the portion extending across the "early-middle" TS $[e, m]$ covers a larger area and thus a larger number of grid points than the portion over the "middle-late" TS $[m, l]$.

Figure 3 suggests that early-in-time TSs should have a shorter extent in time than later ones. Thus, in the second strategy, to which in the following we will refer as LB2, we also consider the *cost* of time steps in order to balance the runtime $R_{\text{co}}^p + R_{\text{fi}}^p$ over all processes p. We use a decomposition of the time axis in TSs such that the sum of the total coarse and fine runtime is balanced over all TSs, i.e., such that $R_{\text{co}} + R_{\text{fi}} = N_{\text{pr}}(R_{\text{co}}^p + R_{\text{fi}}^p)$ for any process p. This is done by a bisection approach, making use of the fact the we use explicit rather than implicit time integrators (see the discussion in [30]) and thus that the cost of a time step from τ to $\tau + \Delta\tau$ is directly proportional to the number of spatial degrees of freedom at τ. Therefore, the total spacetime domain is first divided into two parts of roughly

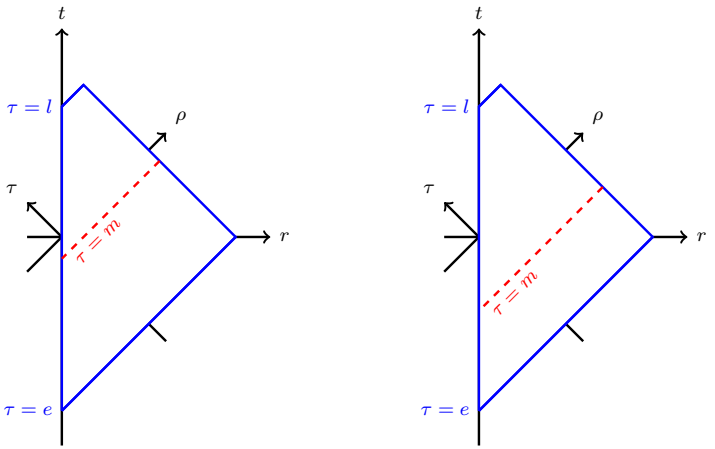


Figure 3. Illustration of two different approaches for the decomposition of the time domain. Left: imbalanced load in time from load balancing LB1. Right: balanced load in time from load balancing LB2.

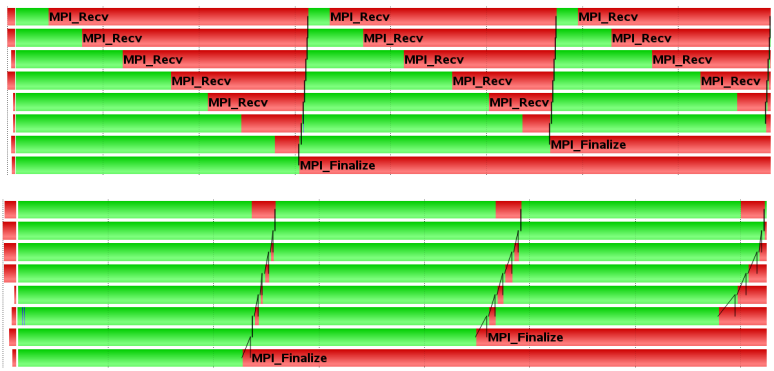


Figure 4. Vampir traces for the implementation $\mathcal{P}_{\text{mo}}$ with $(N_{\text{pr}}, N_{\text{it}}) = (8, 3)$ for two different load balancing strategies. Top: Vampir trace for LB1. The Parareal runtime is $R_{\text{pa}} = 7.964$ s. Bottom: Vampir trace for LB2. The Parareal runtime is $R_{\text{pa}} = 5.436$ s.

equal number of grid points as is sketched in Figure 3, right. Then, each part is divided again and again until the required number of TSs is reached. Note that this limits the possible numbers of TSs to powers of 2.

Figure 4 shows Vampir[5] traces for one simulation featuring LB1 (Figure 4, top) and one LB2 (Figure 4, bottom). The horizontal axes correspond to runtime, while the vertical axes depict MPI rank numbers from 1 (lower) to 8 (upper). In each case, three Parareal iterations are performed. Green regions indicate the coarse and fine integrators carrying out work. Time spent in MPI receives (including waiting time) is shown in red. We observe how LB1 leads to load imbalance and incurs

[5] https://www.vampir.eu/

significant wait times in processes handling a later TS. In contrast, the processes' idle times (shown in red) in MPI receives are almost invisible in the case of LB2. Elimination of wait times leads to a significant reduction in runtime and increase in speedup, as will be shown in Section 4.

4. Results

Speedup and runtime measurements were performed on the Cray XC40 supercomputer Piz Dora[6] at the Swiss National Supercomputing Center (CSCS) in Lugano. It features 1256 compute nodes, which all hold two 12-core Intel Xeon E5-2690v3 processors. This results in a total of 30144 compute cores and a peak performance of 1.254 PFlops; it occupies position 56 in the Top500 November 2014 list.[7] On Piz Dora, we used the GNU Compiler Collection[8] version 4.9.2 and the runtimes we provide do not include the cost of I/O operations. Some simulations measuring convergence were performed on a machine located at the Università della Svizzera italiana that is maintained by members of the Institute of Computational Science of the Faculty of Informatics.[9]

For the results presented in the following, we use a coarse grid resolution of $(\Delta\tau)_{\text{co}} = (\Delta\rho)_{\text{co}} = \Delta_{\text{co}} = L/2048 \approx 0.039$ and a fine grid resolution of $\Delta_{\text{fi}} = \Delta_{\text{co}}/8 \approx 0.005$. We have also determined a *reference* solution to approximately measure the serial fine stepper's discretization error. For this we have used again the serial fine time stepper but with a step size of $\Delta_{\text{re}} = \Delta_{\text{fi}}/4 \approx 0.001$.

4.1. ***Subcritical.*** First we consider the subcritical case, where no black holes form. Figure 5 shows for $N_{\text{pr}} = 256$ and two different sets of initial data parameters the relative defect

$$D_{[i]} = \frac{\|r_{[i]} - r_{\text{fi}}\|_2}{\|r_{\text{fi}}\|_2}, \tag{4.1.1}$$

which measures the difference between the Parareal solution $r_{[i]}$ after i iterations and the serial fine solution r_{fi} as a function of the characteristic coordinate τ.

In Figure 5, left, we use the initial data parameters $(\phi_0, \rho_0, \delta_0) = (0.035, 20, 1)$, which results in an "early" bounce of the wave packet at about $\tau = 20$. For the simulations in Figure 5, right, the values are $(\phi_0, \rho_0, \delta_0) = (0.01, 75, 1)$, which leads to a "late" bounce at about $\tau = 75$. Defects are plotted for $N_{\text{it}} \in \{1, 2, 3, 4\}$ along with the estimated discretization errors $\|r_{\text{co}} - r_{\text{re}}\|_2/\|r_{\text{fi}}\|_2$ of serial coarse and $\|r_{\text{fi}} - r_{\text{re}}\|_2/\|r_{\text{fi}}\|_2$ of serial fine solutions. We observe that in Figure 5, left, the data for $N_{\text{it}} = 3$ is somewhat jagged because for LB2 there are various start

[6] http://www.cscs.ch/computers/piz_daint_piz_dora/

[7] http://www.top500.org/list/2014/11

[8] https://gcc.gnu.org

[9] https://www.ics.usi.ch/index.php/ics-research/resources

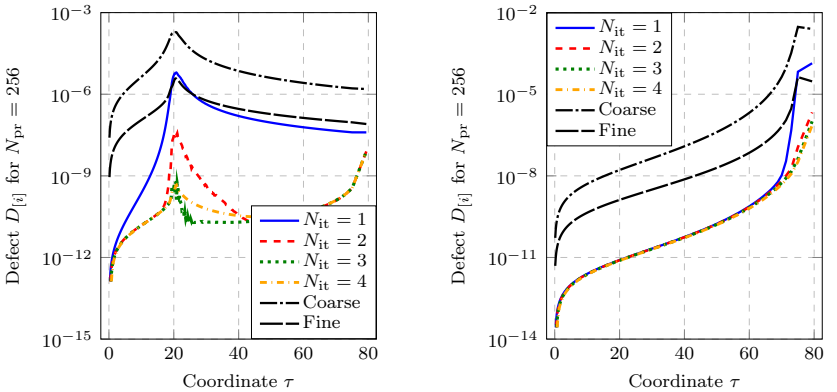


Figure 5. Defect in r between Parareal and the fine method over time for fixed $N_{pr} = 256$. Left: early bounce scenario. Right: late bounce situation.

and end times of TSs near the bounce region. In any case, Parareal converges in two iterations: for $N_{it} = 2$, the defect is below the discretization error for all τ. In fact, without the bounce region near $\tau = 20$, only one iteration would be required for convergence. For the late bounce scenario in Figure 5, right, we also observe that the rate of convergence at the final time $\tau = L - \lambda$ gives an indication of the convergence at all τ. In the following we thus focus on convergence at the final time. Convergence for the other evolved field Φ is not shown but was found to be at least as good as for r.[10]

Figure 6, left and middle, illustrate the defect of Parareal at the end of the simulation at $\tau = L - \lambda$ for various values of N_{pr} with third-order interpolation (left) and fifth-order interpolation (middle). For third-order interpolation, Parareal does not converge at all. The configuration stalls at a defect of about 10^{-2} until the iteration count equals N_{pr}. There, Parareal converges by definition but cannot provide any speedup. In contrast, Parareal shows good convergence behavior for fifth-order interpolation. For N_{pr} less than 64, the defect of Parareal falls below the approximate discretization error of the fine method after a single iteration. Otherwise, for $N_{pr} \geq 64$ up to $N_{pr} = 512$, two iterations are required.

The resulting speedups with correspondingly adjusted values for N_{it} are shown in Figure 6, right, for both load balancing strategies (see the discussion in Section 3.5). In addition, the projected speedup according to (3.4.1) is shown. The fine-to-coarse ratio R_{fi}/R_{co} was determined experimentally and found to be about 74. Up to $N_{pr} = 64$, for the advanced load balancing, speedup closely mirrors the theoretical curve while the basic load balancing performs significantly worse. For $N_{pr} \geq 64$, measured speedups fall short of the theoretical values, peak at $N_{pr} = 256$, and

[10]Convergence seems to be unaffected by the load balancing. In tests not documented here, we found that for LB1 it takes two iterations for Parareal to converge as well.

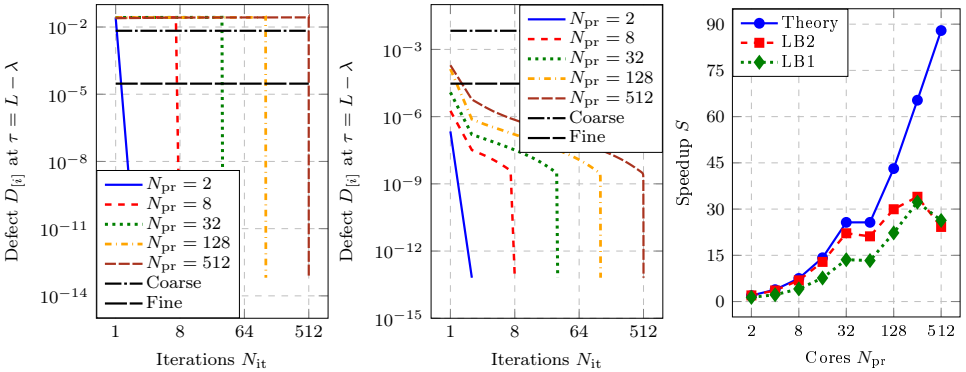


Figure 6. Parareal's performance for the subcritical case in terms of convergence for polynomial interpolation orders 3 and 5 and in terms of speedup. Left: defect for late bounce and interpolation order 3. Middle: defect for late bounce and interpolation order 5. Right: Parareal speedup for fifth-order interpolation.

then start to decrease. Note that the theoretical model (blue line in Figure 6, right) does take into account the scaling limit from the serial correction step according to Amdahl's law. The difference between theory and measured speedup is therefore due to other overheads (communication and transfer between meshes) as analyzed below.

Although the load balancing strategy LB2 results in significantly better speedup than the basic approach LB1, the peak value provided by both schemes is essentially the same. This is because, for increasingly large numbers of cores, the computational load per TS eventually becomes small and imbalances in computational load insignificant. Instead, runtime is dominated by overhead from, e.g., communication in time. The communication load is independent of the chosen load balancing and depends solely on the number of TSs; for every TS one message has to be sent and received once per iteration (save for the first and last TS). Therefore, it can be expected that ultimately both approaches to load balancing lead to comparable peak values. Below we demonstrate that the saturation in speedup is related to a significant increase in time spent in MPI routines; eventually, communication cost starts to dominate over the computational cost left on each time slice and the time parallelization saturates just as spatial parallelization does.

Figure 7 illustrates the reason behind the drop-off in speedup beyond $N_{pr} = 256$. First, define

$$R^p_{pa} = R^p_{co} + R^p_{fi} + \sum_{st} R^p_{st}, \tag{4.1.2}$$

where R^p_{st} denotes runtime spent in *stages* that are *different* from coarse and fine integration on the TS assigned to process p. For now, we consider only overhead from sending and receiving data as well as from interpolation; other overheads are

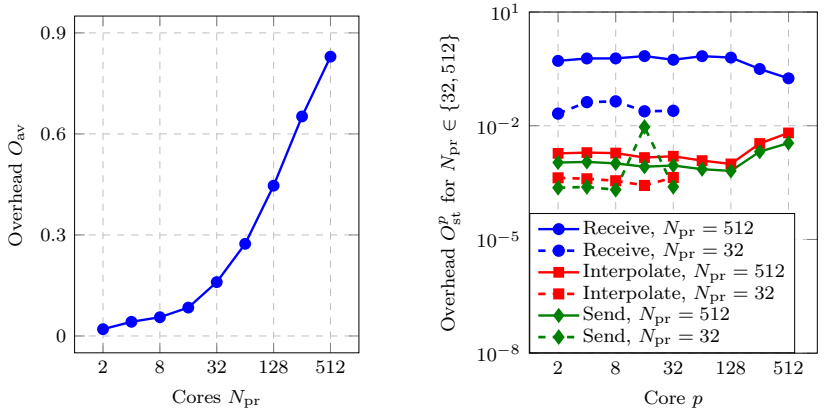


Figure 7. Overhead from communication and other sources increases with N_{pr}, which leads to Parareal's speedup decay. Left: average overhead. Right: overhead caused by three different Parareal stages.

not further analyzed here. Next, we introduce the *total* overhead on a TS as the sum of all stage runtimes or

$$O^p_{to} = \sum_{st} R^p_{st}, \tag{4.1.3}$$

which is also the runtime spent *neither* in the coarse *nor* fine integrator for a given p. The *average* overhead is now defined as the geometric mean value of O^p_{to} over all TSs, which is

$$O_{av} = \frac{\sum_{p=1}^{N_{pr}} O^p_{to}}{N_{pr}}. \tag{4.1.4}$$

Finally, we define the relative overhead for individual *stages* on a TS as

$$O^p_{st} = \frac{R^p_{st}}{R^p_{pa}}, \tag{4.1.5}$$

where R^p_{pa} is the runtime of Parareal at processor p. Ideally, as is assumed for the derivation of the speedup model given in (3.4.1), R^p_{co} and R^p_{fi} are the dominant costs. In this case, $R^p_{co} + R^p_{fi} \approx R^p_{pa}$ so that according to (4.1.2) we have $O^p_{to} \approx 0$ and therefore $O_{av} \approx 0$ by definition. However, as can be seen in Figure 7, left, O_{av} is small only for small values of N_{pr}. For $N_{pr} \geq 32$ it increases rapidly, which indicates that the overhead from communication and other sources starts to play a more dominant role when N_{pr} is increased.

Figure 7, right, shows the relative overhead from (4.1.5) for $N_{pr} \in \{32, 512\}$ and $p \in \{1, \ldots, N_{pr}\}$ for the three different stages st $\in$ {Interpolation, Send, Receive}; "Send" and "Receive" refer to the corresponding MPI routines. There is a significant

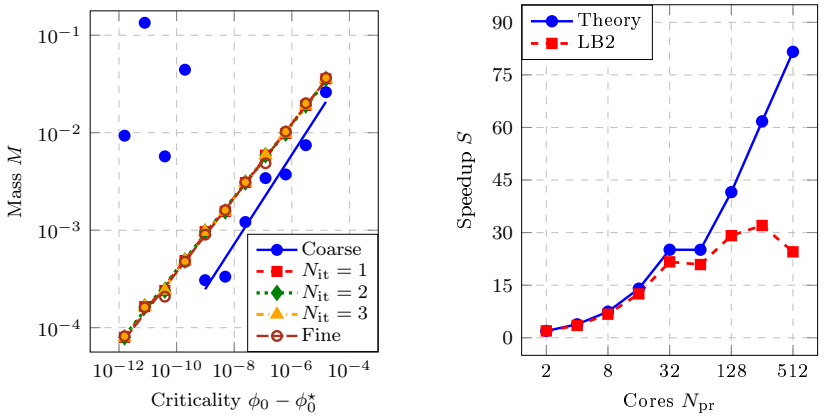


Figure 8. Parareal's performance for the supercritical case. Left: Choptuik scaling from Parareal. Right: Parareal speedup.

increase in relative overhead in all three stages as the number of cores grows, causing the eventual drop-off in speedup for increasing N_{pr}.

4.2. ***Supercritical.*** We consider now the more complex case in which a black hole forms at some time during the simulation. The goal is to compute the black hole's position via (2.1.8) so that its mass can be determined from (2.1.9) (see Section 2.3). Because the characteristic coordinates (τ, ρ) do not allow us to continue the simulation past the black hole formation event, we need a way to keep the simulation from terminating when Θ^+ approaches 0 (see Figure 2, right).

To avoid the need to adaptively modify the decomposition of the time domain, we carry out the supercritical case study using initial data parameter values near $(\phi_0, \rho_0, \delta_0) = (0.01, 75, 1)$, which we have also used for the results in Figure 5, right. With these parameters and in particular for $\phi_0 \geq 0.01$, for all investigated partitions of the time axis with $N_{\text{pr}} \leq 256$, the black hole generated by the fine time integrator forms in the *last* TS unless ϕ_0 becomes too large (ρ_0 and δ_0 are fixed). Thus, Parareal can be used over all TSs except for the last one, where only the fine method is executed to compute the black hole's position. The C++ implementation uses a try-throw-catch approach to prevent complete termination of the simulation; if the radicand in the definition of Θ^+ in (2.1.8) fails to be nonnegative, an exception is thrown such that the Parareal iteration can continue. As the Parareal iteration converges and better and better starting values are provided for $\mathcal{F}$ on the last TS, the accuracy of the computed black hole position improves. A more general implementation aiming at production runs would need to allow for black hole formation in TSs before the last one, but this is left for future work. In this article, the focus lies on investigating the *principal applicability* of Parareal to the simulation of gravitational collapse.

	$\phi_0^\star$		γ	
	Value	Error (%)	Value	Error (%)
Coarse	0.01057748	$7.25 \cdot 10^{-1}$	0.458	20.21
$N_{\text{it}} = 1$	0.01055915	$5.51 \cdot 10^{-1}$	0.377	1.05
$N_{\text{it}} = 2$	0.01050240	$1.01 \cdot 10^{-2}$	0.370	2.89
$N_{\text{it}} = 3$	0.01050135	$9.52 \cdot 10^{-5}$	0.381	0
Fine	0.01050134	0	0.381	0

Table 1. Approximate values and relative errors for the critical amplitude $\phi_0^\star$ and resulting straight line slope γ.

Figure 8, left, depicts the Choptuik scaling that results from solutions computed with Parareal for $N_{\text{pr}} = 256$ after the first three iterations. Table 1 lists the generated values of $\phi_0^\star$ and γ (see Section 2.3) and errors compared to the value provided by the fine integrator, which agrees with the result in [20]. As can be seen in Figure 8, left, the coarse integrator $\mathcal{C}$ alone cannot adequately resolve black holes with $\phi_0 - \phi_0^\star \lesssim 10^{-9}$ (they are too small for $\mathcal{C}$ to be "visible") and its γ is wrong by about 20%. This means that the coarse method is too "coarse" in the sense that, on its own, it cannot correctly capture the physics underlying the investigated problem. Nonetheless, *Parareal* is not only capable of generating the correct black hole physics but can do so after only one iteration.

Figure 8, right, visualizes the speedup achieved in the supercritical case including the theoretical estimate according to (3.4.1). The numbers of iterations required for Parareal to converge are derived from an analysis just like the one plotted in Figure 6, middle, for the subcritical case, and basically the values are identical. Up to 64 processes, good speedup close to the theoretical bound is observed. For larger core numbers, however, speedup reaches a plateau and performance is no longer increasing. As in the subcritical case, as N_{pr} increases, the computing times per TS eventually become too small and Parareal's runtime becomes dominated by, e.g., communication (see Figure 7). Even though the temporal parallelization eventually saturates, substantial acceleration of almost a factor of 30 using 128 cores in time is possible, corresponding to a parallel efficiency of about 23%.

5. Conclusion

The article assesses the performance of the parallel-in-time integration method Parareal for the numerical simulation of gravitational collapse of a massless scalar field in spherical symmetry. It gives an overview of the dynamics and physics described by the corresponding Einstein field equations and presents the employed numerical methods to solve them. Because the system is formulated and solved in characteristic coordinates, the computational spacetime domain is triangular so that

later time steps carry fewer spatial degrees of freedom. A strategy for balancing computational *cost* per subinterval instead of just number of steps is discussed, and its benefits are demonstrated by traces using the Vampir tool. Numerical experiments are presented for both the sub- and supercritical case. Parareal converges rapidly for both and, for the latter, correctly reproduces Choptuik's mass scaling law after only one iteration despite the fact that the used coarse integrator alone generates a strongly flawed mass scaling law. This underlines the capability of Parareal to quickly correct a coarse method that does not resolve the dynamics of the problem. The results given here illustrate that Parareal and presumably other parallel-in-time methods as well can be used to improve utilization of parallel computers for numerical studies of black hole formation.

Multiple directions for future research emerge from the presented results. Evaluating performance gains for computing the critical solution [7; 21] would be valuable. Next, more complex collapse scenarios such as in the Einstein–Yang–Mills system [8], axial symmetry [37], or binary black hole spacetimes [38] could be addressed. An extended implementation of Parareal could utilize a more sophisticated convergence criterion [2], more flexible black hole detection, and parallelism in space via, e.g., again Parareal. The latter would be possible because the integration along the characteristic we took to represent space is for the solution of initial value problems just like in the temporal direction. Another topic of interest is that of adaptive mesh refinement (J. Thornburg, personal communication, 2015): how it can be used efficiently in connection with Parareal or other time-parallel methods seems to be an open problem. As discussed in the introduction, a mathematical analysis of the convergence behavior of Parareal for Einstein's equations would be of great interest as well, particularly since the good performance is unexpected in view of the negative theoretical results for basic hyperbolic problems. Finally, incorporating a parallel-in-time integration method into a software library widely used for black hole or other numerical relativity simulations would be the ideal way to make this new approach available to a large group of domain scientists.[11]

Acknowledgments

We would like to thank Matthew Choptuik from the University of British Columbia in Vancouver, Canada and Jonathan Thornburg from the Indiana University in Bloomington for providing feedback and suggestions on an earlier version of the manuscript. We would also like to thank Jean-Guillaume Piccinali and Gilles Fourestey from the Swiss National Supercomputing Center (CSCS) in Lugano and

[11] A copy of the library Lib4PrM for the Parareal method can be obtained by cloning the Git repository https://scm.ti-edu.ch/repogit/lib4prm.

Andrea Arteaga from the Swiss Federal Institute of Technology Zurich (ETHZ) for discussions concerning the hardware at CSCS.

This research is funded by the Deutsche Forschungsgemeinschaft (DFG) as part of the "ExaSolvers" project in the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) and by the Swiss National Science Foundation (SNSF) under the lead agency agreement as grant SNSF-145271. The research of Kreienbuehl, Ruprecht, and Krause is also funded through the "Future Swiss Electrical Infrastructure" (FURIES) project of the Swiss Competence Centers for Energy Research (SCCER) at the Commission for Technology and Innovation (CTI).

References

[1] M. Alcubierre, *Introduction to $3+1$ numerical relativity*, International Series of Monographs on Physics, no. 140, Oxford University, 2008.

[2] E. Aubanel, *Scheduling of tasks in the parareal algorithm*, Parallel Comput. **37** (2011), no. 3, 172–182.

[3] T. W. Baumgarte and S. L. Shapiro, *Numerical relativity: solving Einstein's equations on the computer*, Cambridge University, 2010.

[4] M. J. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys. **53** (1984), no. 3, 484–512.

[5] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Rev. **46** (2004), no. 3, 501–517.

[6] F. Chen, J. S. Hesthaven, and X. Zhu, *On the use of reduced basis methods to accelerate and stabilize the parareal method*, Reduced order methods for modeling and computational reduction (A. Quarteroni and G. Rozza, eds.), Modeling, Simulation and Applications, no. 9, Springer, Cham, 2014, pp. 187–214.

[7] M. W. Choptuik, *Universality and scaling in gravitational collapse of a massless scalar field*, Phys. Rev. Lett. **70** (1993), no. 1, 9–12.

[8] M. W. Choptuik, E. W. Hirschmann, and R. L. Marsa, *New critical behavior in Einstein–Yang–Mills collapse*, Phys. Rev. D **60** (1999), no. 12, 124011.

[9] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput. **32** (2010), no. 2, 818–835.

[10] D. Christodoulou, *Bounded variation solutions of the spherically symmetric Einstein-scalar field equations*, Comm. Pure Appl. Math. **46** (1993), no. 8, 1131–1220.

[11] X. Dai and Y. Maday, *Stable parareal in time method for first- and second-order hyperbolic systems*, SIAM J. Sci. Comput. **35** (2013), no. 1, A52–A78.

[12] M. Emmett and M. L. Minion, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci. **7** (2012), no. 1, 105–132.

[13] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid*, SIAM J. Sci. Comput. **36** (2014), no. 6, C635–C661.

[14] P. F. Fischer, F. Hecht, and Y. Maday, *A parareal in time semi-implicit approximation of the Navier–Stokes equations*, Domain decomposition methods in science and engineering (Berlin, 2003) (R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. Xu, eds.), Lecture Notes in Computational Science and Engineering, no. 40, Springer, Berlin, 2005, pp. 433–440.

[15] M. S. Floater and K. Hormann, *Barycentric rational interpolation with no poles and high rates of approximation*, Numer. Math. **107** (2007), no. 2, 315–331.

[16] M. J. Gander, *Analysis of the parareal algorithm applied to hyperbolic problems using characteristics*, Bol. Soc. Esp. Mat. Apl. (2008), no. 42, 21–35.

[17] ———, *50 years of time parallel time integration*, Multiple shooting and time domain decomposition methods (Heidelberg, 2013) (T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds.), Contributions in Mathematical and Computational Sciences, no. 9, Springer, Cham, 2015, pp. 69–113.

[18] M. J. Gander and M. Petcu, *Analysis of a Krylov subspace enhanced parareal algorithm for linear problems*, ESAIM Proc. **25** (2008), 114–129.

[19] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput. **29** (2007), no. 2, 556–578.

[20] D. Garfinkle, *Choptuik scaling in null coordinates*, Phys. Rev. D **51** (1995), no. 10, 5558–5561.

[21] C. Gundlach and J. M. Martín-García, *Critical phenomena in gravitational collapse*, Living Rev. Relativ. **10** (2007), no. 5.

[22] W. Hackbusch, *Parabolic multigrid methods*, Proceedings of the sixth International Symposium on Computing Methods in Applied Sciences and Engineering (Versailles, 1983) (R. Glowinski and J.-L. Lions, eds.), North-Holland, Amsterdam, 1984, pp. 189–197.

[23] G. Horton, *The time-parallel multigrid method*, Comm. Appl. Numer. Methods **8** (1992), no. 9, 585–595.

[24] G. Horton, S. Vandewalle, and P. Worley, *An algorithm with polylog parallel complexity for solving parabolic partial differential equations*, SIAM J. Sci. Comput. **16** (1995), no. 3, 531–541.

[25] V. Husain, *Critical behaviour in quantum gravitational collapse*, Adv. Sci. Lett. **2** (2009), no. 2, 214–220.

[26] L. E. Kidder, M. A. Scheel, S. A. Teukolsky, E. D. Carlson, and G. B. Cook, *Black hole evolution by spectral methods*, Phys. Rev. D **62** (2000), no. 8, 084032.

[27] E. Komatsu, J. Dunkley, M. R. Nolta, C. L. Bennett, B. Gold, G. Hinshaw, N. Jarosik, D. Larson, M. Limon, L. Page, D. N. Spergel, M. Halpern, R. S. Hill, A. Kogut, S. S. Meyer, G. S. Tucker, J. L. Weiland, E. Wollack, and E. L. Wright, *Five-year Wilkinson microwave anisotropy probe observations: cosmological interpretation*, Astrophys. J. Suppl. S. **180** (2009), no. 2, 330–376.

[28] A. Kreienbuehl, *Quantum cosmology, polymer matter, and modified collapse*, Ph.D. thesis, University of New Brunswick, 2011.

[29] A. Kreienbuehl, V. Husain, and S. S. Seahra, *Modified general relativity as a model for quantum gravitational collapse*, Classical Quant. Grav. **29** (2012), no. 9, 095008.

[30] A. Kreienbuehl, A. Naegel, D. Ruprecht, R. Speck, G. Wittum, and R. Krause, *Numerical simulation of skin transport using Parareal*, Comput. Vis. Sci. **17** (2015), no. 2, 99–108.

[31] J.-L. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps "pararéel"*, C. R. Acad. Sci. Paris I **332** (2001), no. 7, 661–668.

[32] F. Löffler, J. Faber, E. Bentivegna, T. Bode, P. Diener, R. Haas, I. Hinder, B. C. Mundim, C. D. Ott, E. Schnetter, G. Allen, M. Campanelli, and P. Laguna, *The Einstein Toolkit: a community computational infrastructure for relativistic astrophysics*, Classical Quant. Grav. **29** (2012), no. 11, 115001.

[33] M. L. Minion, *A hybrid parareal spectral deferred corrections method*, Commun. Appl. Math. Comput. Sci. **5** (2010), no. 2, 265–301.

[34] A. S. Nielsen and J. S. Hesthaven, *Fault tolerance in the Parareal method*, Proceedings of the ACM Workshop on Fault-Tolerance for HPC at Extreme Scale (Kyoto, 2016), ACM, New York, 2016.

[35] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Comm. ACM **7** (1964), no. 12, 731–733.

[36] E. Poisson, *A relativist's toolkit: the mathematics of black-hole mechanics*, Cambridge University, 2004.

[37] F. Pretorius, *Numerical simulations of gravitational collapse*, Ph.D. thesis, University of British Columbia, 2002.

[38] ———, *Evolution of binary black-hole spacetimes*, Phys. Rev. Lett. **95** (2005), no. 12, 121101.

[39] F. Pretorius and L. Lehner, *Adaptive mesh refinement for characteristic codes*, J. Comp. Phys. **198** (2004), no. 1, 10–34.

[40] D. Ruprecht and R. Krause, *Explicit parallel-in-time integration of a linear acoustic-advection system*, Comput. Fluids **59** (2012), 72–83.

[41] R. Speck and D. Ruprecht, *Toward fault-tolerant parallel-in-time integration with PFASST*, Parallel Comput. **62** (2017), 20–37.

[42] R. Speck, D. Ruprecht, R. Krause, M. Emmett, M. Minion, M. Winkel, and P. Gibbon, *A massively space-time parallel N-body solver*, SC '12: proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (Salt Lake City, 2012), IEEE, Los Alamitos, CA, 2012.

[43] J. Thornburg, *Adaptive mesh refinement for characteristic grids*, Gen. Relativity Gravitation **43** (2011), no. 5, 1211–1251.

[44] J. Ziprick and G. Kunstatter, *Dynamical singularity resolution in spherically symmetric black hole formation*, Phys. Rev. D **80** (2009), no. 2, 024032.

Received April 24, 2016. Revised December 28, 2016.

ANDREAS KREIENBUEHL: akreienbuehl@lbl.gov
Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

PIETRO BENEDUSI: pietro.benedusi@usi.ch
Institute of Computational Science, Faculty of Informatics, Università della Svizzera italiana, Via Giuseppe Buffi 13, CH-6904 Lugano, Switzerland

DANIEL RUPRECHT: d.ruprecht@leeds.ac.uk
School of Mechanical Engineering, University of Leeds, Woodhouse Lane, Leeds, LS2 9JT, United Kingdom

ROLF KRAUSE: rolf.krause@usi.ch
Institute of Computational Science, Faculty of Informatics, Università della Svizzera italiana, Via Giuseppe Buffi 13, CH-6904 Lugano, Switzerland

Guidelines for Authors

Authors may submit manuscripts in PDF format on-line at the Submission page at msp.org/camcos.

Originality. Submission of a manuscript acknowledges that the manuscript is original and and is not, in whole or in part, published or under consideration for publication elsewhere. It is understood also that the manuscript will not be submitted elsewhere while under consideration for publication in this journal.

Language. Articles in CAMCoS are usually in English, but articles written in other languages are welcome.

Required items. A brief abstract of about 150 words or less must be included. It should be self-contained and not make any reference to the bibliography. If the article is not in English, two versions of the abstract must be included, one in the language of the article and one in English. Also required are keywords and subject classifications for the article, and, for each author, postal address, affiliation (if appropriate), and email address.

Format. Authors are encouraged to use LaTeX but submissions in other varieties of TeX, and exceptionally in other formats, are acceptable. Initial uploads should be in PDF format; after the refereeing process we will ask you to submit all source material.

References. Bibliographical references should be complete, including article titles and page ranges. All references in the bibliography should be cited in the text. The use of BibTeX is preferred but not required. Tags will be converted to the house format, however, for submission you may use the format of your choice. Links will be provided to all literature with known web locations and authors are encouraged to provide their own links in addition to those supplied in the editorial process.

Figures. Figures must be of publication quality. After acceptance, you will need to submit the original source files in vector graphics format for all diagrams in your manuscript: vector EPS or vector PDF files are the most useful.

Most drawing and graphing packages (Mathematica, Adobe Illustrator, Corel Draw, MATLAB, etc.) allow the user to save files in one of these formats. Make sure that what you are saving is vector graphics and not a bitmap. If you need help, please write to graphics@msp.org with details about how your graphics were generated.

White space. Forced line breaks or page breaks should not be inserted in the document. There is no point in your trying to optimize line and page breaks in the original manuscript. The manuscript will be reformatted to use the journal's preferred fonts and layout.

Proofs. Page proofs will be made available to authors (or to the designated corresponding author) at a Web site in PDF format. Failure to acknowledge the receipt of proofs or to return corrections within the requested deadline may cause publication to be postponed.

Communications in Applied Mathematics and Computational Science

vol. 12 no. 1 2017