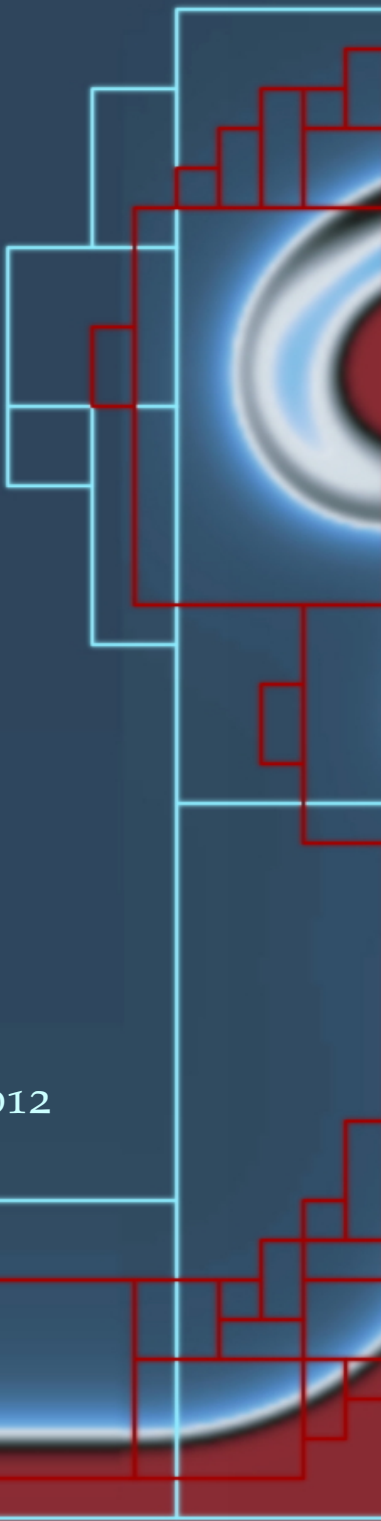
Communications in Applied Mathematics and Computational Science



vol. 7 no. 1 2012

mathematical sciences publishers

Communications in Applied Mathematics and Computational Science

map.berkeley.edu/camcos

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

The subscription price for 2012 is US $75/year for the electronic version, and $105/year for print and electronic. Subscriptions, requests for back issues from the last three years and changes of subscribers address should be sent to Mathematical Sciences Publishers, Department of Mathematics, University of California, Berkeley, CA 94720-3840, USA.

Communications in Applied Mathematics and Computational Science, at Mathematical Sciences Publishers, Department of Mathematics, 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 Mathematical Sciences Publishers.

PUBLISHED BY
mathematical sciences publishers
http://msp.org/
A NON-PROFIT CORPORATION
Typeset in LaTeX

COMM. APP. MATH. AND COMP. SCI.
Vol. 7, No. 1, 2012

msp

AN EMBEDDED BOUNDARY METHOD FOR THE NAVIER–STOKES EQUATIONS ON A TIME-DEPENDENT DOMAIN

GREGORY H. MILLER AND DAVID TREBOTICH

We present a new conservative Cartesian grid embedded boundary method for the solution of the incompressible Navier–Stokes equations in a time-dependent domain. It is a Godunov-projection fractional step scheme in which hyperbolic advection and a variety of implicit and explicit Helmholtz operations are performed on time-stationary domains. The transfer of data from one fixed domain to another uses third-order interpolation. The method is second order accurate in L_1 and first order in L_∞. The algorithm is verified on flow geometries with prescribed boundary motion.

1. Introduction

The incompressible Navier–Stokes equations on a time-dependent domain

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla P + \nu \Delta \boldsymbol{u} \tag{1.1a}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{1.1b}$$

approximate fluid behavior in a range of important applications. Here $\boldsymbol{u}(\boldsymbol{x}, t)$ is the velocity of the fluid, whose density is assumed to be unity, $\boldsymbol{x}$ is the spatial coordinate, t is time, P is pressure, and ν is the kinematic viscosity. We are particularly concerned with reaction-diffusion equations in porous media where reactive transport can alter the subsurface pore structure due to precipitation or dissolution. Other motivating applications include the dynamics of biological membranes and lipid bilayer analogs, and modeling rod-climbing and die-swell behavior of certain viscoelastic fluids. In these examples, the evolution of the fluid domain is coupled

This material is based upon work supported by the National Science Foundation under grant number DMS-0810939, and is supported as part of the Center for Nanoscale Control of Geologic CO_2, an Energy Frontier Research Center funded by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, and the Office of Advanced Scientific Computing, under Award Number DE-AC02-05CH11231, and by DOE grant number DE-SC0001981.

MSC2010: 35Q30, 35R37, 65M08.

Keywords: Navier–Stokes, embedded boundary, finite volume, moving domain.

to the motion of the fluid. Prescribed domain motion occurs in pumps, stirred vessels, and other mechanical systems.

There are two categories of approaches to discretizing moving domains: (i) gridding schemes that conform to the domain boundary, e.g., unstructured grids obeying Lagrangian dynamics; and (ii) structured, Cartesian grids where the domain influences the solution through a forcing as in the immersed boundary method [30] or the immersed interface method [19], or through cut cell methods where the finite volume quadrature is modified on those Cartesian cells overlain by the domain boundary, otherwise known as embedded boundary methods. Cut cell methods are confronted by a small-cell stability problem: finite volume discretizations are unstable on cells whose volume fraction vanishes. Approaches to this problem include cell merging techniques (Noh's "blending" [27]), the h-box technique that references a cell of nonvanishing size [2], and hybridization — use of a stable but nonconservative quadrature with subsequent reestablishment of conservation in a neighborhood [4]. Our approach is an embedded boundary method, with cut-cell stability through hybridization. This strategy has proven accurate, robust, and scalable in large scale simulations of reactive transport in fixed irregular domains [33].

Projection methods [7; 9; 8] use the unique Hodge decomposition of a vector field to determine the divergence-free component, and the gradient of a potential that can be associated with the pressure gradient. Godunov-projection methods are fractional step methods that first compute an intermediate velocity with a high-order Godunov approach, which is made discrete divergence-free by a Hodge projection. Other approaches can achieve high order without reference to the intermediate state, for example computation of $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ via an Adams–Bashforth approach. Our approach is based on the second-order projection method of Bell et al. [1], with a second-order unsplit Godunov method for the intermediate velocity [11], and using approximate projections after Lai [17]. For hyperbolic flow problems, high-order Godunov methods do a superior job of resolving steep gradients. Minion and Brown [3] compare a number of approaches to solving incompressible Navier–Stokes. Their examples show that the Godunov-projection approach does a good job of resolving incompressible Navier–Stokes flows with steep gradients without introducing spurious high-frequency oscillations created by some other approaches. This is a significant benefit for reacting flows where steep gradients exist and reaction rates can be sensitive to high-frequency oscillation.

There have been many recent developments in projection methods for the moving domain Navier–Stokes problem. Pan and coworkers [28] use a Godunov-projection method with multiblock structured ALE (arbitrary Lagrangian–Eulerian) grids. Udaykumar et al. [39] use an Adams–Bashforth projection approach with finite volume discretization. They locate the interface with Lagrangian marker

particles, and address the small cell problem with cell merging. Marella et al. [22] employ a similar method, with interface information derived from a discrete level set. Tan et al. [36] also use level sets to represent the interface, and combine the immersed interface method with an Adams–Bashforth projection method. Liau et al. [20] combine an Adams–Bashforth projection method with the immersed boundary method. Chiu et al. [5] use the immersed boundary method with a different second-order projection discretization. All of these methods claim or demonstrate second-order accuracy. Strict conservation is necessary to accurately capture wave behavior [18], a property essential for combustion and reactive flows. Such conservation is readily obtained with ALE and finite volume methods but is a very delicate issue for immersed boundary methods [16].

In this work we present a new conservative Godunov-projection method on Cartesian grids for the solution of the incompressible Navier–Stokes equations (1.1a) on a time-dependent domain $\Omega(t)$, with boundary conditions

$$\boldsymbol{u} = \boldsymbol{s}(\boldsymbol{x}, t) \tag{1.2a}$$

on moving walls, where $\boldsymbol{s}$ is the velocity of the boundary;

$$\boldsymbol{u} = \boldsymbol{u}_{\text{in}}(\boldsymbol{x}, t) \tag{1.2b}$$

on inflow boundaries; and

$$\boldsymbol{n} \cdot \nabla \boldsymbol{u} = 0 \tag{1.2c}$$

on outflow boundaries where $\boldsymbol{n}$ is normal to the domain boundary. We represent the domain boundary as the zero of a distance function level set, and derive all geometric descriptions at the moving front from the discrete level set. In this work, the boundary motion is prescribed.

We discretize space in uniform Cartesian cells which we label with index $\boldsymbol{i}$, an integer vector in D space dimensions. The center of cell $\boldsymbol{i}$ has spatial coordinate $\boldsymbol{x} = h(\boldsymbol{i} + \frac{1}{2}\mathbf{1})$ where h is the length of the cell, and $\mathbf{1}$ is the vector of ones in $\mathbb{Z}^D$. Time is discretized in uniform increments Δt, and $t^n = n\Delta t$ is the time at step n. $\boldsymbol{u}_{\boldsymbol{i}}^n$ denotes the value of fluid velocity $\boldsymbol{u}$ at the center of cell $\boldsymbol{i}$ at time t^n, and with $\boldsymbol{e}_d$ the d-th unit basis vector, $\boldsymbol{u}_{\boldsymbol{i}+1/2\boldsymbol{e}_d}^{n+1/2}$ denotes the fluid velocity at the half time step $t^{n+1/2}$ and the center of $\boldsymbol{i}$'s cell face in direction $+d$. With this discretization, an outline of the approach is:

(1) Extrapolate $\boldsymbol{u}_{\boldsymbol{i}}^n$ to $\Omega^{n+1/2}$, the fluid domain at time $t^{n+1/2}$. For those cells $\boldsymbol{i}$ in $\Omega^{n+1/2} \setminus \Omega^n$ (Figures 1 and 2), this extrapolation is based on the algorithm proposed by McCorquodale et al. [25]: three cells, whose centers together with $\boldsymbol{i}$ are collinear and approximately aligned with the interface normal, define a quadratic interpolation function determining $\boldsymbol{u}_{\boldsymbol{i}}$.

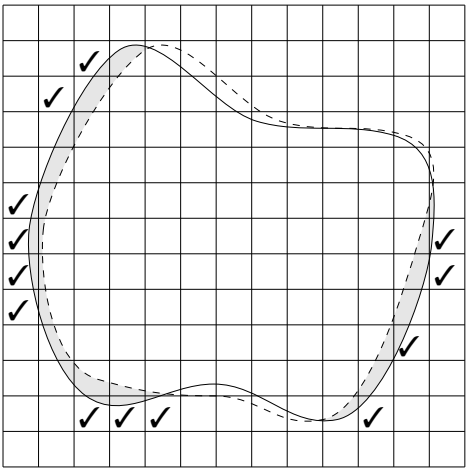


Figure 1. Newly uncovered cells. Domain boundary $\delta\Omega^n$ is shown with a dashed curve; Ω^n is the enclosed volume, and domain boundary $\delta\Omega^{n+1}$ is shown with the solid curve. The region $\Omega^{n+1} \setminus \Omega^n$ (shaded) contains fluid at t^{n+1} but not at t^n; it is a newly uncovered region. If a cell contains a newly uncovered region, and also contains fluid at time t^n, then the value of the field in Ω^{n+1} is copied from the same cell in Ω^n. But, if a newly uncovered region does not contain t^n values, the values in the extended domain must be estimated by extrapolation. Such cells are indicated with check marks.

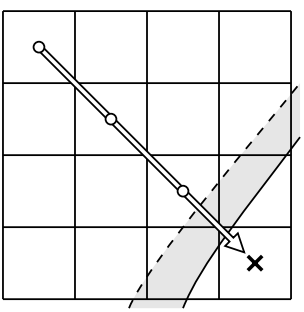


Figure 2. Extrapolation to newly uncovered cells. $\delta\Omega^n$ is shown as a dashed curve, and $\delta\Omega^{n+1}$ is a solid curve. Symbol X indicates the cell center of a newly uncovered cell. The arrow is aligned with the vector comprised of values 0 and ± 1 that is most nearly parallel the normal to $\delta\Omega^{n+1}$. Points along that arrow (open circles) are used to construct a quadratic, i.e., third-order, extrapolation polynomial.

(2) On $\Omega^{n+1/2}$, use high-order Godunov methods to compute time- and edge-centered values $\boldsymbol{u}^{n+1/2}_{\boldsymbol{i}+1/2\boldsymbol{e}_j}$, $j = 1, \ldots, D$ [11], and make this field discrete divergence-free with a MAC projection [15].

(3) Compute a nonconservative but stable flux difference, a conservative but unstable flux difference, and a stable hybrid flux difference for the hyperbolic treatment of $\boldsymbol{u}_t = -\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ [4].

(4) Modify the hybrid field $\boldsymbol{u}\cdot\nabla\boldsymbol{u}$ so that it obeys global conservation, i.e., so that $\boldsymbol{u}_t+\boldsymbol{u}\cdot\nabla\boldsymbol{u}=0$ is equivalent to the mathematically identical conservation form $\boldsymbol{u}_t+\nabla\cdot(\boldsymbol{u}\boldsymbol{u})=0$ in the weak sense.

(5) Extrapolate $\boldsymbol{u}\cdot\nabla\boldsymbol{u}$ and lagged estimate $\nabla P^{n-\frac{1}{2}}$ to Ω^{n+1}. On Ω^{n+1} solve the heat equation $\boldsymbol{u}_t=\nu\Delta\boldsymbol{u}+\boldsymbol{f}$ with source term $\boldsymbol{f}=-\nabla P-\boldsymbol{u}\cdot\nabla\boldsymbol{u}$;

$$\tilde{\boldsymbol{u}}=\mathscr{L}_{\mathrm{TGA}}\big(\boldsymbol{u}^n,\ -(\nabla P)^{n-1/2}-(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n+1/2}\big), \tag{1.3}$$

where $\mathscr{L}_{\mathrm{TGA}}$ is a particular discretization of the heat operator defined later by (2.49).

(6) Make $\boldsymbol{u}^{n+1}$ discrete divergence free with a cell-centered projection $\mathbb{P}$ (to be defined by (2.6)). The projection computes $\nabla P^{n+1/2}$ on Ω^{n+1}:

$$\boldsymbol{u}^*=\tilde{\boldsymbol{u}}+\Delta t(\nabla P)^{n-1/2}, \tag{1.4a}$$

$$\boldsymbol{u}^{n+1}=\mathbb{P}(\boldsymbol{u}^*), \tag{1.4b}$$

$$(\nabla P)^{n+1/2}=\frac{1}{\Delta t}(\boldsymbol{I}-\mathbb{P})(\boldsymbol{u}^*). \tag{1.4c}$$

In Section 2 additional details of the algorithm will be presented, with emphasis on those aspects that are new to this work. We will emphasize the algorithm as implemented on a single grid. The components of this algorithm have been shown elsewhere to operate on a hierarchy of nested grids, enabling an adaptive mesh capability. Our implementation includes this capability, and runs in 2D and 3D with SIMD parallelism. A numerical demonstration of convergence rates is presented in Section 3.

2. Algorithm details

In Section 2.1 the existence of a Hodge decomposition for the moving domain problem is described. This discussion justifies the projections used in outline steps (2) and (6). The implementation of the projection for cell-centered $\boldsymbol{u}$ has been described in [37] and implementation details related to adaptive meshes are given by [23; 24].

Next, in Section 2.2 the high-order Godunov approach to computing edge- and time-centered values $\boldsymbol{u}^{n+1/2}_{\boldsymbol{i}+1/2\boldsymbol{e}_d}$, outline step (2), is described.

In Section 2.3 the treatment of $\boldsymbol{u}\cdot\nabla\boldsymbol{u}$ as a hyperbolic update is described. This includes the stable and conservative forms mentioned in outline step (3), and the conservation property enforced in outline step (4).

The moving domain heat problem employed in outline step (5) was first published by McCorquodale et al. [25]. They demonstrate numerically that on a single domain Ω^{n+1} one can discretize the heat problem on time interval $[t^n,t^{n+1}]$, using

specially constructed boundary conditions and extrapolated source terms and initial conditions. In Section 2.4 we present a theoretical justification for that method.

Finally, in Section 2.5 we present some details on the construction of geometric terms used to define the quadratures underlying the solution to Poisson's equation (projection), the Helmholtz equation (heat), and the treatment of $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ as a hyperbolic source term on a moving domain. We use an idea due to Ligocki et al. [21] that derives geometric information using a hierarchical application of the divergence theorem. Our implementation is entirely new and differs from theirs by including some relevant inequality constraints. The specialization of that approach is described in the case that the primary source of geometric information is a discretized distance function.

2.1. ***Hodge projection on a moving domain.*** To implement a projection method on a moving domain, Trebotich and Colella [12; 37] decompose a vector field $\boldsymbol{w}$ into three components:

$$\boldsymbol{w} = \underbrace{\boldsymbol{v} + \nabla\theta}_{\boldsymbol{u}} + \nabla\phi, \tag{2.1a}$$

$$\Delta\theta = 0, \tag{2.1b}$$

$$\nabla \cdot \boldsymbol{v} = 0. \tag{2.1c}$$

In the context of incompressible Navier–Stokes, $\boldsymbol{u}$ is a divergence-free velocity field, consisting of a vorticity-carrying component $\boldsymbol{v}$ and an incompressible potential flow $\nabla\theta$. $\nabla\phi$ is the gradient of a potential, which can be used to determine ∇P. The boundary conditions for this decomposition are

(1) moving walls:

$$\boldsymbol{n} \cdot \boldsymbol{v} = 0, \tag{2.2a}$$

$$\boldsymbol{n} \cdot \nabla\theta = \boldsymbol{n} \cdot \boldsymbol{s}, \tag{2.2b}$$

$$\boldsymbol{n} \cdot \nabla\phi = \boldsymbol{n} \cdot (\boldsymbol{w} - \boldsymbol{s}); \tag{2.2c}$$

(2) inflow boundaries:

$$\boldsymbol{v} = 0, \tag{2.3a}$$

$$\boldsymbol{n} \cdot \nabla\theta = \boldsymbol{u}_0(\boldsymbol{x}, t) \quad \text{(prescribed)}, \tag{2.3b}$$

$$\boldsymbol{n} \cdot \nabla\phi = \boldsymbol{n} \cdot (\boldsymbol{w} - \boldsymbol{u}_0); \tag{2.3c}$$

(3) outflow boundaries:

$$\boldsymbol{n} \cdot \nabla \boldsymbol{v} = 0, \tag{2.4a}$$

$$\boldsymbol{n} \cdot \nabla\theta = \bar{\boldsymbol{u}}_{\text{out}}, \tag{2.4b}$$

$$\phi = 0. \tag{2.4c}$$

Here $\bar{\boldsymbol{u}}_{\text{out}}$ is the average outflow velocity given by conservation over the entire domain. These boundary conditions with $\boldsymbol{u} = \boldsymbol{v} + \nabla\theta$ are equivalent to the boundary conditions (1.2) of our problem. The Trebotich–Colella decomposition is solvable: the θ equation is well posed without null space, and $\boldsymbol{w} - \nabla\theta$ has boundary conditions compatible with the Hodge decomposition (e.g., [6]). Therefore, $\boldsymbol{v}$, $\nabla\theta$ and $\nabla\phi$ can be determined uniquely. A projection in this framework is accomplished by

$$\phi: \quad \Delta\phi = \nabla\cdot(\boldsymbol{w} - \nabla\theta), \tag{2.5a}$$

$$\boldsymbol{v} = (\boldsymbol{w} - \nabla\theta) - \nabla\phi. \tag{2.5b}$$

The existence of decomposition (2.1a) does not require explicit determination of potential θ. Instead,

$$\phi: \quad \Delta\phi = \Delta(\phi + \theta) = \nabla\cdot\boldsymbol{w}, \tag{2.6a}$$

$$\boldsymbol{u} = \boldsymbol{w} - \nabla\phi, \tag{2.6b}$$

or

$$\boldsymbol{u} = \mathbb{P}(\boldsymbol{w}), \tag{2.6c}$$

follows directly by application of (2.1b) to (2.5). The boundary conditions for projection (2.6) are

(1) moving walls:

$$\boldsymbol{n}\cdot\boldsymbol{u} = \boldsymbol{n}\cdot\boldsymbol{s}, \tag{2.7a}$$

$$\boldsymbol{n}\cdot\nabla\phi = \boldsymbol{n}\cdot(\boldsymbol{w} - \boldsymbol{s}); \tag{2.7b}$$

(2) inflow boundaries:

$$\boldsymbol{n}\cdot\boldsymbol{u} = \boldsymbol{u}_0(\boldsymbol{x}, t) \quad \text{(prescribed)}, \tag{2.8a}$$

$$\boldsymbol{n}\cdot\nabla\phi = \boldsymbol{n}\cdot(\boldsymbol{w} - \boldsymbol{u}_0); \tag{2.8b}$$

(3) outflow boundaries:

$$\boldsymbol{n}\cdot\nabla\boldsymbol{u} = 0, \tag{2.9a}$$

$$\phi = 0. \tag{2.9b}$$

These match (1.2) on $\boldsymbol{u}$, and for ϕ are identical to (2.1a).

Trebotich and Colella raise two concerns regarding the application of the Hodge decomposition to moving domains [12; 37]. The first is over boundary conditions, but as shown above, the existence of their velocity decomposition makes a Hodge decomposition with boundary conditions (2.7)–(2.9) viable. Second, they object to the use of a discrete projection that does not commute with the discrete PDE operators. While it is true that these discrete operators do not commute because of the boundary conditions on the discrete divergence, that property is not essential

to the success of the method. If we assume that the mixed derivative $\boldsymbol{u}_{xt}$ exists and is C^0, then the *differential* operators $\nabla\cdot$ and $\partial/\partial t$ commute [35] and, without recourse to discretization, the governing PDE gives

$$\Delta P = \nabla\cdot(\nu\Delta\boldsymbol{u} - \boldsymbol{u}\cdot\nabla\boldsymbol{u}) \tag{2.10}$$

with boundary condition

$$\boldsymbol{n}\cdot\nabla P = \boldsymbol{n}\cdot\left(\nu\Delta\boldsymbol{u} - \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \frac{\partial\boldsymbol{u}}{\partial t}\right). \tag{2.11}$$

This assumption on $\boldsymbol{u}_{xt}$ is required as well in the fixed-domain case (e.g., [8, Equation (2′)]).

If

$$\boldsymbol{w} = \boldsymbol{u}^* \approx \boldsymbol{u}^n + \Delta t(\nu\Delta\boldsymbol{u} - \boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n+1/2} \tag{2.12a}$$

$$= \boldsymbol{u}^{n+1} + \Delta t\nabla P^{n+1/2} + \mathbb{O}(\Delta t^3) + \mathbb{O}(h^2) \tag{2.12b}$$

(see (1.4a)), then with $\phi \approx \Delta t\, P$ and $\boldsymbol{u} = \boldsymbol{s}$ on $\delta\Omega$, the linear problems

$$\phi: \quad \Delta\phi = \nabla\cdot\boldsymbol{w}, \tag{2.13a}$$

$$\boldsymbol{n}\cdot\nabla\phi = \Delta t\boldsymbol{n}\cdot\nabla P^{n+1/2} \quad \text{on } \delta\Omega^{n+1}, \tag{2.13b}$$

$$\boldsymbol{n}\cdot\boldsymbol{w} = \boldsymbol{n}\cdot[\boldsymbol{s}^{n+1} + \Delta t\nabla P^{n+1/2}] \quad \text{on } \delta\Omega^{n+1} \tag{2.13c}$$

and

$$\phi: \quad \Delta\phi = \nabla\cdot\boldsymbol{w}, \tag{2.14a}$$

$$\boldsymbol{n}\cdot\nabla\phi = 0 \quad \text{on } \delta\Omega^{n+1}, \tag{2.14b}$$

$$\boldsymbol{n}\cdot\boldsymbol{w} = \boldsymbol{n}\cdot\boldsymbol{s}^{n+1} \quad \text{on } \delta\Omega^{n+1} \tag{2.14c}$$

are equivalent to $\mathbb{O}(\Delta t^3) + \mathbb{O}(h^2)$. The former (2.13) is the physical problem to be solved; the latter (2.14) is the Hodge decomposition we implement, and whose existence and uniqueness is addressed above. This approach amounts to placing the inhomogeneous boundary condition due to the moving domain in the divergence of velocity on the right-hand side of the Poisson's equation and solving the homogeneous (Neumann) problem for the pressure. The same approach maps true inflow conditions to $\boldsymbol{n}\cdot\boldsymbol{w} = \boldsymbol{u}_0$ and $\boldsymbol{n}\cdot\nabla\phi = 0$. For the outflow, conditions $\boldsymbol{n}\cdot\nabla\boldsymbol{w} = 0$ and $\phi = 0$ are literal. This discussion has used the time centering corresponding to the cell-centered projection of outline step (6). The MAC projection (outline step (2)) is entirely analogous.

2.2. ***High-order Godunov advection.*** The computation of $\boldsymbol{u}^{n+1/2}_{\boldsymbol{i}+1/2\boldsymbol{e}_d}$ is based on an adaptation of the embedded boundary method for hyperbolic PDEs [11]. It is a three-step process:

I. In the first step, cell-centered velocities $\boldsymbol{u}_{\boldsymbol{i}}^{n}$ are averaged to edges $\boldsymbol{u}_{\boldsymbol{i}+1/2\boldsymbol{e}_d}^{n}$, and this velocity field is used to resolve Riemann problems in an advective calculation. First, the velocity is extrapolated to faces with upwind characteristics:

$$\boldsymbol{u}_{\boldsymbol{i}}^{\pm d} = \boldsymbol{u}_{\boldsymbol{i}}^{n} \pm \frac{1}{2}\min\Big(1 \mp (\boldsymbol{e}_d \cdot \boldsymbol{u}_{\boldsymbol{i}}^{n})\frac{\Delta t}{h},\ 1\Big)(\delta_d \boldsymbol{u}^{n})_{\boldsymbol{i}} + \frac{\Delta t}{2}(\nu \Delta \boldsymbol{u}^{n})_{\boldsymbol{i}}. \tag{2.15}$$

This initial extrapolation does not include transverse derivatives or the pressure gradient. δ is a difference operator using van Leer [40] limiting:

$$\delta_d(u) = \begin{cases} \delta_d^{\mathrm{vL}}(u) & \text{if } (u_{\boldsymbol{i}+\boldsymbol{e}_d} - u_{\boldsymbol{i}})(u_{\boldsymbol{i}} - u_{\boldsymbol{i}-\boldsymbol{e}_d}) > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{2.16}$$

$$\begin{aligned} \delta_d^{\mathrm{vL}}(u) = {} & \operatorname{sign}(u_{\boldsymbol{i}+\boldsymbol{e}_d} - u_{\boldsymbol{i}-\boldsymbol{e}_d}) \\ & \times \min\big(2\left|u_{\boldsymbol{i}} - u_{\boldsymbol{i}-\boldsymbol{e}_d}\right|,\ 2\left|u_{\boldsymbol{i}+\boldsymbol{e}_d} - u_{\boldsymbol{i}}\right|,\ \tfrac{1}{2}\left|u_{\boldsymbol{i}+\boldsymbol{e}_d} - u_{\boldsymbol{i}-\boldsymbol{e}_d}\right|\big). \end{aligned} \tag{2.17}$$

Further, $\boldsymbol{u}_{\boldsymbol{i}}^{+d}$ is the value of velocity extrapolated to the right side of cell $\boldsymbol{i}$ in direction d, and $\boldsymbol{u}_{\boldsymbol{i}+\boldsymbol{e}_d}^{-d}$ is the value extrapolated to the same edge from cell $\boldsymbol{i}+\boldsymbol{e}_d$. A single-valued result is obtained by resolving the Riemann problem, which amounts to upwinding:

$$\bar{\boldsymbol{u}}_{\boldsymbol{i}+1/2\boldsymbol{e}_d}^{n+1/2} = \begin{cases} \boldsymbol{u}_{\boldsymbol{i}}^{+d} & \text{if } (\boldsymbol{e}_d \cdot \boldsymbol{u})_{\boldsymbol{i}+1/2\boldsymbol{e}_d}^{n} > 0, \\ \boldsymbol{u}_{\boldsymbol{i}+\boldsymbol{e}_d}^{-d} & \text{if } (\boldsymbol{e}_d \cdot \boldsymbol{u})_{\boldsymbol{i}+1/2\boldsymbol{e}_d}^{n} < 0, \\ \frac{1}{2}\big(\boldsymbol{u}_{\boldsymbol{i}}^{+d} + \boldsymbol{u}_{\boldsymbol{i}+\boldsymbol{e}_d}^{-d}\big) & \text{if } (\boldsymbol{e}_d \cdot \boldsymbol{u})_{\boldsymbol{i}+1/2\boldsymbol{e}_d}^{n} = 0. \end{cases} \tag{2.18}$$

The output of this Riemann problem is used to provide transverse flux corrections. In 2D (see Figure 3),

$$\boldsymbol{u}_{\boldsymbol{i}}^{\pm d} := \boldsymbol{u}_{\boldsymbol{i}}^{\pm d} - \frac{\Delta t}{2h}\big(\bar{\boldsymbol{u}}_{\boldsymbol{i}+1/2\boldsymbol{e}_{d'}}^{n+1/2} - \bar{\boldsymbol{u}}_{\boldsymbol{i}-1/2\boldsymbol{e}_{d'}}^{n+1/2}\big)(\boldsymbol{e}_d \cdot \boldsymbol{u}_{\boldsymbol{i}}^{n}), \quad d' \neq d, \tag{2.19}$$

followed by another Riemann solution. In 3D the transverse flux correction is more complicated [32].

II. A discrete MAC projection is used to make the advected velocities divergence-free:

$$\Phi:\ \Delta\Phi = \nabla \cdot \bar{\boldsymbol{u}}; \qquad (\nabla \cdot \bar{\boldsymbol{u}})_{\boldsymbol{i}} = \frac{1}{h}\sum_d \big(\bar{\boldsymbol{u}}_{\boldsymbol{i}+1/2\boldsymbol{e}_d}^{n+1/2} - \bar{\boldsymbol{u}}_{\boldsymbol{i}-1/2\boldsymbol{e}_d}^{n+1/2}\big), \tag{2.20a}$$

$$\boldsymbol{e}_d \cdot \boldsymbol{u} = \boldsymbol{e}_d \cdot \bar{\boldsymbol{u}} - \nabla_d \Phi. \tag{2.20b}$$

This projection only affects the normal component of the edge velocities.

III. The third step repeats step I, but the velocity used to judge upwind direction in the Riemann problem is the divergence-free edge velocity computed in step II. In this step the normal velocity components are not changed, but the transverse

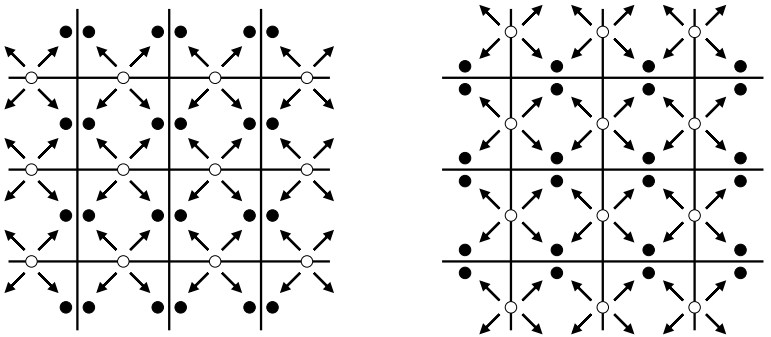


Figure 3. Transverse flux correction in 2D. Double-valued edge states $\boldsymbol{u}^{\pm d}$ are indicated by filled circles, and single-valued states $\bar{\boldsymbol{u}}$ are indicated by open circles. Differences in $\bar{\boldsymbol{u}}$ across a given cell provide flux correction to the states $\boldsymbol{u}^{\pm d}$ associated with that cell, but in transverse directions.

ones are. Finally, these transverse components are corrected to account for the pressure gradient computed in II. In 2D,

$$\boldsymbol{e}_{d'}\cdot\boldsymbol{u}^{n+1/2}_{\boldsymbol{i}+1/2\boldsymbol{e}_d} := \boldsymbol{e}_{d'}\cdot\boldsymbol{u}^{n+1/2}_{\boldsymbol{i}+1/2\boldsymbol{e}_d} - \tfrac{1}{4}\big[(\nabla_{d'}\Phi)_{\boldsymbol{i}+1/2\boldsymbol{e}_{d'}} + (\nabla_{d'}\Phi)_{\boldsymbol{i}+\boldsymbol{e}_d+1/2\boldsymbol{e}_{d'}} + (\nabla_{d'}\Phi)_{\boldsymbol{i}-1/2\boldsymbol{e}_{d'}} + (\nabla_{d'}\Phi)_{\boldsymbol{i}+\boldsymbol{e}_d-1/2\boldsymbol{e}_{d'}}\big], \quad (2.21)$$

where $d' \neq d$ is the transverse direction. The generalization to 3D is straightforward.

The extension of this algorithm to embedded boundary geometries is described in [11]. One change is to employ one-sided differences where the data does not support centered stencils. Another concerns the determination of so-called covered-edge values. Covered edges are those edges of irregular cells which are not in contact with the fluid. For these edges, the upwind characteristic tracing step provides a single edge value on the fluid side of the edge. The value on the side opposite the fluid is obtained by extrapolation from edge values interior to the domain (Figure 4); see [11, §5.2] for details.

2.3. ***Hyperbolic step.*** We are interested here in a formulation of $\boldsymbol{u}\cdot\nabla\boldsymbol{u}$ that is consistent with the hyperbolic split of the Navier–Stokes equations

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla\cdot\boldsymbol{F} = 0 \quad (2.22)$$

with $\boldsymbol{F} \equiv \boldsymbol{u}\boldsymbol{u}$, $\nabla\cdot\boldsymbol{F} = \boldsymbol{u}\cdot\nabla\boldsymbol{u}$ when $\nabla\cdot\boldsymbol{u} = 0$. For this hyperbolic equation, one has a discretization

$$\boldsymbol{u}^{n+1}_{\text{nonconservative}} = \boldsymbol{u}^n - \frac{\Delta t}{h}(\boldsymbol{D}\boldsymbol{F})^{\text{nc}} \quad (2.23)$$

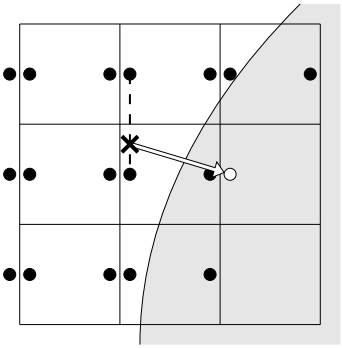


Figure 4. Covered edge calculation, illustrated in 2D for the case of $\hat{y}$-side edges. Closed circles indicate edge values calculated by the 1D advection algorithm described here, though modified to use one-sided differences near boundaries. The open circle indicates an exterior covered edge, in this case on the right side of an edge. This right-edge covered value is extrapolated from right-edge uncovered values by interpolation (dashed line), and extrapolation in the direction of the interface normal (arrow), using the cell-centered gradient. When the uncovered values are modified to account for transverse flux correction, this calculation is repeated so the covered edge value also includes transverse corrections.

with $\boldsymbol{DF}$ a flux difference which we approximate by

$$(\boldsymbol{DF})_{\boldsymbol{i}}^{\mathrm{nc}} = \sum_{d}^{D} \frac{1}{h}\left(u_{d,\boldsymbol{i}+1/2\boldsymbol{e}_d}^{n+1/2}\boldsymbol{u}_{\boldsymbol{i}+1/2\boldsymbol{e}_d}^{n+1/2} - u_{d,\boldsymbol{i}-1/2\boldsymbol{e}_d}^{n+1/2}\boldsymbol{u}_{\boldsymbol{i}-1/2\boldsymbol{e}_d}^{n+1/2}\right). \tag{2.24}$$

This discretization is second-order accurate in regular cells, but not consistent in cut cells. It is stable in both cases.

A conservative discretization of the conservation law on the irregular control volume comes from the space-time integration over the fluid in an irregular cell:

$$\begin{aligned} 0 &= \int_{t^n}^{t^{n+1}} dt \int_{\Omega_{\boldsymbol{i}}(t)} dV\left(\frac{\partial}{\partial t}, \nabla\right)\cdot(\boldsymbol{u}, \boldsymbol{F}) \qquad (2.25)\\ &\approx \kappa_{\boldsymbol{i}}^{n+1} h^D \boldsymbol{u}_{\boldsymbol{i}}^{n+1} - \kappa_{\boldsymbol{i}}^{n} h^D \boldsymbol{u}_{\boldsymbol{i}}^{n} \\ &\quad + \Delta t h^{D-1} \sum_{d}^{D}\left(\alpha_{\boldsymbol{i}+1/2\boldsymbol{e}_d}\boldsymbol{F}_{d,\boldsymbol{i}+1/2\boldsymbol{e}_d} - \alpha_{\boldsymbol{i}-1/2\boldsymbol{e}_d}\boldsymbol{F}_{d,\boldsymbol{i}-1/2\boldsymbol{e}_d}\right) + A_{\boldsymbol{i},\mathrm{EB}}\boldsymbol{n}_{\boldsymbol{i},EB}\cdot(\boldsymbol{u}, \boldsymbol{F})_{\boldsymbol{i},\mathrm{EB}}, \end{aligned}$$

where

$$\Omega_{\boldsymbol{i}}(t) = \Omega(t) \cap [h\boldsymbol{i},\ h(\boldsymbol{i}+\boldsymbol{1})] \tag{2.26}$$

is the fluid-occupied volume of cell $\boldsymbol{i}$ at time t. Subscript EB denotes that the object is located on the embedded boundary, and EB will be used also as an abbreviation.

Here κ denotes a volume fraction,

$$\kappa_{\boldsymbol{i}}^{n} = \frac{1}{h^D} \int_{\Omega_{\boldsymbol{i}}(t^n)} dV; \tag{2.27}$$

α a space-time area fraction (also known as "aperture"),

$$\alpha_{\boldsymbol{i}-1/2\boldsymbol{e}_d} = \frac{1}{h^{D-1}\Delta t} \int_{t^n}^{t^{n+1}} dt \int_{\delta\Omega_{\boldsymbol{i}}(t)\cap\{\boldsymbol{x}|x_d=hi_d\}} dA; \tag{2.28}$$

and A_{EB} is the space-time area of the EB. $\boldsymbol{n}_{\mathrm{EB}}$ is the unit normal in $\mathbb{R}^{D+1}$. The $D+1$ components of $A_{\mathrm{EB}}\boldsymbol{n}_{\mathrm{EB}}$ can be determined from the condition $\mathrm{div}(\boldsymbol{e}_i) = 0$ for each of the $D+1$ directions i, giving

$$\begin{aligned}\kappa_{\boldsymbol{i}}^{n+1} \underset{\mathrm{cent}}{\boldsymbol{u}}_{\boldsymbol{i}}^{n+1} = \kappa_{\boldsymbol{i}}^{n} \underset{\mathrm{cent}}{\boldsymbol{u}}_{\boldsymbol{i}}^{n} - \frac{\Delta t}{h} \sum_{d}^{D} (\alpha_{\boldsymbol{i}+1/2\boldsymbol{e}_d} \boldsymbol{F}_{d,\boldsymbol{i}+1/2\boldsymbol{e}_d} - \alpha_{\boldsymbol{i}-1/2\boldsymbol{e}_d} \boldsymbol{F}_{d,\boldsymbol{i}-1/2\boldsymbol{e}_d}) \\ - (\kappa_{\boldsymbol{i}}^{n} - \kappa_{\boldsymbol{i}}^{n+1}) \boldsymbol{u}_{\boldsymbol{i},\mathrm{EB}} - \frac{\Delta t}{h} \sum_{d}^{D} (\alpha_{\boldsymbol{i}-1/2\boldsymbol{e}_d} - \alpha_{\boldsymbol{i}+1/2\boldsymbol{e}_d}) \boldsymbol{F}_{d,\boldsymbol{i},\mathrm{EB}}.\end{aligned} \tag{2.29}$$

Here we have written $\underset{\mathrm{cent}}{\boldsymbol{u}}$ to emphasize that the centering is at the centroid $\underset{\mathrm{cent}}{\boldsymbol{x}}$ for (2.29) to be consistent (Figure 5);

$$\underset{\mathrm{cent}}{\boldsymbol{x}}_{\boldsymbol{i}}^{n} = \frac{1}{h^D \kappa_{\boldsymbol{i}}^{n}} \int_{\Omega_{\boldsymbol{i}}(t^n)} \boldsymbol{x} dV. \tag{2.30}$$

However, the elliptic operators we use are based on a cell-centered discretization $\underset{\mathrm{cc}}{\boldsymbol{u}}$, which suggests the modification

$$\underset{\mathrm{cc}}{\boldsymbol{u}}_{\boldsymbol{i}}^{n+1} - \underset{\mathrm{cc}}{\boldsymbol{u}}_{\boldsymbol{i}}^{n} = -\frac{\Delta t}{h} (\boldsymbol{D}\boldsymbol{F})^{c}, \tag{2.31}$$

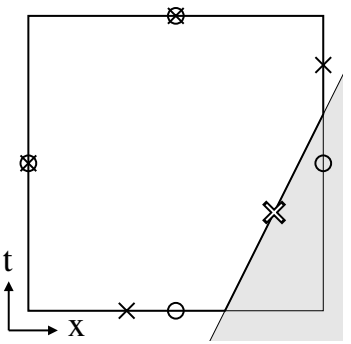


Figure 5. Centerings: centers (open circles) and centroids (crosses). In regular domains, the discretization relies on centered quantities. A convergent stencil in irregular domains uses centroid-centered quantities.

$$(\boldsymbol{DF})^c = \frac{1}{\kappa_{\boldsymbol{i}}^{n+1}}\Bigg[\frac{h}{\Delta t}\Big(\kappa_{\boldsymbol{i}}^{n}(\underset{\text{cc}}{\boldsymbol{x}}-\underset{\text{cent}}{\boldsymbol{x}})_{\boldsymbol{i}}^{n}\cdot(\nabla\boldsymbol{u})_{\boldsymbol{i}}^{n}-\kappa_{\boldsymbol{i}}^{n+1}(\underset{\text{cc}}{\boldsymbol{x}}-\underset{\text{cent}}{\boldsymbol{x}})_{\boldsymbol{i}}^{n+1}\cdot(\nabla\boldsymbol{u})_{\boldsymbol{i}}^{n}+(\kappa_{\boldsymbol{i}}^{n+1}-\kappa_{\boldsymbol{i}}^{n})\underset{\text{cc}}{\boldsymbol{u}}_{\boldsymbol{i}}^{n}\Big)$$
$$+\sum_{d}^{D}(\alpha_{\boldsymbol{i}+1/2\boldsymbol{e}_d}\boldsymbol{F}_{d,\boldsymbol{i}+1/2\boldsymbol{e}_d}-\alpha_{\boldsymbol{i}-1/2\boldsymbol{e}_d}\boldsymbol{F}_{d,\boldsymbol{i}-1/2\boldsymbol{e}_d})$$
$$+\frac{h}{\Delta t}(\kappa_{\boldsymbol{i}}^{n}-\kappa_{\boldsymbol{i}}^{n+1})\boldsymbol{u}_{\boldsymbol{i},\text{EB}}+\sum_{d}^{D}(\alpha_{\boldsymbol{i}-1/2\boldsymbol{e}_d}-\alpha_{\boldsymbol{i}+1/2\boldsymbol{e}_d})\boldsymbol{F}_{d,\boldsymbol{i},\text{EB}}\Bigg]. \quad (2.32)$$

Equation (2.31) has $\mathbb{O}(h)$ discretization error in irregular cells (when $\kappa < 1$), and is second-order in regular cells. The velocity at the centroid of the EB is $\boldsymbol{s}(\boldsymbol{x}, t)$, the prescribed boundary condition (1.2a). Fluxes at the centroids of cell faces are calculated by interpolating the velocity field to the centroid

$$\underset{\text{cent}}{\boldsymbol{x}}_{\boldsymbol{i}-1/2\boldsymbol{e}_d} = \frac{1}{h^{D-1}\Delta t\alpha_{\boldsymbol{i}-1/2\boldsymbol{e}_d}}\int_{t^n}^{t^{n+1}}dt\int_{\delta\Omega_{\boldsymbol{i}}(t)\cap\{\boldsymbol{x}|x_d=hi_d\}}\boldsymbol{x}\,dA, \quad (2.33a)$$

$$\underset{\text{cent}}{t}_{\boldsymbol{i}-1/2\boldsymbol{e}_d} = \frac{1}{h^{D-1}\Delta t\alpha_{\boldsymbol{i}-1/2\boldsymbol{e}_d}}\int_{t^n}^{t^{n+1}}tdt\int_{\delta\Omega_{\boldsymbol{i}}(t)\cap\{\boldsymbol{x}|x_d=hi_d\}}dAk. \quad (2.33b)$$

The data interpolated is taken from all available data in a 5^D-cell region centered at the point where $\boldsymbol{F}$ is required. This makes $\boldsymbol{F}$ on an irregular edge, say $\boldsymbol{i}+1/2\boldsymbol{e}_d$ independent of the cell, $\boldsymbol{i}$ or $\boldsymbol{i}+\boldsymbol{e}_d$, that shares it. Interpolation is second order in space and time, and implemented by solving an overdetermined set of linear equations with Householder decomposition.

To make the method stable we employ the hybridized flux difference

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n - \frac{\Delta t}{h}\left(\kappa^{n+1}(\boldsymbol{DF})^{\text{c}} + (1-\kappa^{n+1})(\boldsymbol{DF})^{\text{nc}}\right). \quad (2.34)$$

In the limit that cells become regular on $[t^n, t^{n+1}]$ the conservative, nonconservative, and hybrid flux differences are all equivalent to the stable second-order result, and (2.34) reduces to (2.31).

The generalized mass difference is redistributed. The mass excess is

$$\delta\boldsymbol{m} = h^D\kappa^{n+1}(\boldsymbol{u}^{n+1}-\boldsymbol{u}_{\text{unstable}}^{n+1})$$
$$= \Delta t h^{D-1}\kappa^{n+1}(1-\kappa^{n+1})\left((\boldsymbol{DF})^{\text{c}}-(\boldsymbol{DF})^{\text{nc}}\right). \quad (2.35)$$

The negative of this quantity is to be distributed in a volume-weighted sense to neighboring cells [4; 29; 26]. Let $\tilde{\boldsymbol{u}}$ be $\boldsymbol{u}^{n+1}$ evaluated by (2.34), then modified by redistribution. Then $(\boldsymbol{u}^n-\tilde{\boldsymbol{u}})/\Delta t$ is what we refer to in outline step (4) as a conservation-preserving calculation of $\boldsymbol{u}\cdot\nabla\boldsymbol{u}$.

2.4. ***The time-dependent heat problem.*** For the problem

$$u_t = K\Delta u + f \quad \boldsymbol{x} \in \Omega(t), \tag{2.36a}$$

$$u(\boldsymbol{x}, t^n) = u_0(\boldsymbol{x}) \quad \boldsymbol{x} \in \Omega(t^n), \tag{2.36b}$$

$$u(\boldsymbol{x}, t) = u_{\rm bc}(\boldsymbol{x}, t) \quad \boldsymbol{x} \text{ on } \delta\Omega(t), \tag{2.36c}$$

McCorquodale et al. [25] propose the following algorithm:

(1) Interpolate the boundary conditions $u_{\rm bc}(\boldsymbol{x}, t^n)$ to the boundary $\delta\Omega^{n+1}$ with

$$u_{\rm bc}^{\rm interp}(\boldsymbol{x}') = u_{\rm bc}(\boldsymbol{x}, t^n) + (\boldsymbol{x}' - \boldsymbol{x}) \cdot \nabla u_0(\boldsymbol{x}', t^n), \tag{2.37}$$

where $\boldsymbol{x}'$ on $\delta\Omega^{n+1}$, $\boldsymbol{x}$ on $\Delta\Omega^n$, and $|\boldsymbol{x} - \boldsymbol{x}'|$ is $\mathbb{O}(h)$. Specifically, let $\boldsymbol{i}$ be the cell containing $\boldsymbol{x}$, and let $\boldsymbol{i}'$ be the cell containing $\boldsymbol{x}'$. For a given $\boldsymbol{i}$, cell $\boldsymbol{i}'$ is chosen to be the neighbor of $\boldsymbol{i}$ with greatest boundary area (Figure 6).

(2) On $\delta\Omega^{n+1}$, boundary conditions for any time in $[t^n, t^{n+1}]$ are obtained by linear interpolation of $u_{\rm bc}^{\rm interp}(\boldsymbol{x})$ and $u_{\rm bc}(\boldsymbol{x}, t^{n+1})$.

(3) Extrapolate u_0 to Ω^{n+1} using the approach described in outline step (1).

(4) Extrapolate $f(\boldsymbol{x}, t^{n+1/2})$ from $\Omega^{n+1/2}$ to Ω^{n+1} with this same procedure.

(5) On Ω^{n+1}, solve the heat equation by the method of Twizell et al. [38]:

$$\begin{aligned}\boldsymbol{u}^{n+1} = {} & (\boldsymbol{I} - \mu_1 \Delta t \Delta_1^h)^{-1} (\boldsymbol{I} - \mu_2 \Delta t \Delta_2^h)^{-1} \\ & \times \left((\boldsymbol{I} + \mu_3 \Delta t \Delta_3^h) \boldsymbol{u}^{n,\rm extrap} + (\boldsymbol{I} + \mu_4 \Delta t \Delta_4^h) \Delta t \boldsymbol{f}^{n+1/2,\rm extrap} \right),\end{aligned} \tag{2.38}$$

with $\mu_1 = \mu_2 = 1 - 1/\sqrt{2}$, $\mu_3 = \sqrt{2} - 1$, and $\mu_4 = \sqrt{2} - 3/2$. Here $\boldsymbol{u}^{n,\rm extrap}$ is the field $\boldsymbol{u}$ centered at time t^n, but extrapolated from Ω^n to Ω^{n+1}, and

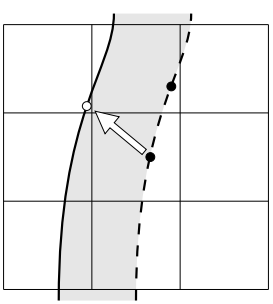


Figure 6. Extrapolation of boundary conditions. The dashed curve is $\delta\Omega^n$, and boundary conditions are known at the centroid of EB segments in each cell. The solid curve is $\delta\Omega^{n+1}$, and boundary conditions are needed at the centroids of this EB in each cell. For each t^{n+1} centroid (e.g., the open circle), the neighboring cell with the greatest boundary area is chosen. In this picture there are two candidates (closed circles). The boundary condition is then extrapolated using the inner product of the cell-centered gradient in the t^n cell and the relative coordinates (arrow).

likewise $\boldsymbol{f}^{n+1/2,\text{extrap}}$ is the source term f centered at $t^{n+1/2}$ and extrapolated from $\Omega^{n+1/2}$ to Ω^{n+1}. Δ_1^h is the discrete Laplacian on Ω^{n+1} with boundary conditions at t^{n+1}; Δ_2^h has boundary conditions at $t^{n+1} - \mu_1 \Delta t$ (by interpolation); and Δ_3^h has boundary conditions at t^n. The boundary conditions on Δ_4^h are homogeneous Dirichlet.

A justification of this algorithm follows.

The ODE

$$u' = A(t)u + f(t) \tag{2.39}$$

has solution

$$u^{n+1} = R(\Delta t)u^n + R(\Delta t)\int_0^{\Delta t} R^{-1}(s)\, f(s)\, ds, \tag{2.40}$$

where R is an integrating factor:

$$R(\Delta t) = \exp\left(\int_{t^n}^{t^n+\Delta t} A(\tau)\, d\tau\right). \tag{2.41}$$

Expanding A in a Taylor series,

$$A(t^n + s) = \sum_{i=0}^{\infty} A_i s^i, \tag{2.42}$$

facilitates constructing an approximation to R:

$$R(\Delta t) \approx \frac{1 + \mu_3\alpha_3\Delta t}{(1 - \mu_1\alpha_1\Delta t)(1 - \mu_2\alpha_2\Delta t)}, \tag{2.43}$$

where

$$\mu_1 = \mu_2 = 1 - 1/\sqrt{2}, \tag{2.44a}$$

$$\mu_3 = \sqrt{2} - 1. \tag{2.44b}$$

These coefficients μ_i minimize the discretization error of this approximation in the case that A is independent of time, which is the case described by Twizell et al. [38]. (Those authors introduce a factor ϵ of order machine precision to lift the degeneracy of (2.44a) and enable a partial fraction representation of (2.43). McCorquodale et al. [25] include this ϵ factor but do not use partial fractions.) The factors α_i are different time centerings of $A(t)$:

$$\alpha_1 = A_0 + A_1 c_1 \Delta t, \tag{2.45a}$$

$$\alpha_2 = A_0 + A_1 c_2 \Delta t, \tag{2.45b}$$

$$\alpha_3 = A_0 + A_1 c_3 \Delta t, \tag{2.45c}$$

with coefficients c_i to be determined. When A is time-varying, the approximation to $R(\Delta t)$ differs from (2.41) by $\mathbb{O}(\Delta t^3)$ provided

$$(c_1 + c_2)(2 - \sqrt{2}) + 2c_3(\sqrt{2} - 1) = 1. \tag{2.46}$$

The solution (see [25])

$$c_1 = 1, \tag{2.47a}$$
$$c_2 = 1/\sqrt{2}, \tag{2.47b}$$
$$c_3 = 0 \tag{2.47c}$$

satisfies this consistency requirement, but not uniquely. Expanding $f(t)$ in a Taylor series about $t^{n+1/2}$ leads to a discretization of that source term. Combined,

$$u^{n+1} = (1 - \mu_1\alpha_1\Delta t)^{-1}(1 - \mu_2\alpha_2\Delta t)^{-1} \times \left((1 + \mu_3\alpha_3\Delta t)u^n + (1 + \mu_4\alpha_4\Delta t)\Delta t f^{n+1/2}\right), \tag{2.48}$$

where $\mu_4 = \sqrt{2} - 3/2$, and α_4 is an arbitrary centering of A.

The choice $c_1 = 1$ is optimal in that the final implicit solve will satisfy its given boundary conditions exactly. An interpretation of this result is that the α_3 operation carries u^n to $u^{n+\mu_3}$, then the α_2 operation carries the solution to $u^{n+1-\mu_1}$, with the final operation α_1 terminating at u^{n+1}. This suggests that $0 \le c_3 \le \mu_3/2$ in order that $\mu_3 \le c_2 \le \mu_3 + \mu_2$, i.e., that the boundary conditions lie within the interval of the associated operation.

Connecting ODE (2.39) to the heat PDE by the method of lines, this analysis suggests

$$\boldsymbol{u}^{n+1} = (\boldsymbol{I} - \mu_1\mathscr{L}_1\Delta t)^{-1}(\boldsymbol{I} - \mu_2\mathscr{L}_2\Delta t)^{-1} \times \left((\boldsymbol{I} + \mu_3\mathscr{L}_3\Delta t)\boldsymbol{u}^n + (\boldsymbol{I} + \mu_4\mathscr{L}_4\Delta t)\Delta t\, \boldsymbol{f}^{n+1/2}\right), \tag{2.49a}$$

which we abbreviate as

$$\boldsymbol{u}^{n+1} = \mathscr{L}_{\mathrm{TGA}}\left(\boldsymbol{u}^n,\ \boldsymbol{f}^{n+1/2}\right): \tag{2.49b}$$

the solution at t^{n+1} to $\boldsymbol{u}_t = \mathscr{L}\boldsymbol{u} + \boldsymbol{f}$. When $\mathscr{L}$ is a negative definite operator, this discretization is L_0 stable and second-order accurate in time. Since

$$\mu_1 + \mu_2 + \mu_3 = 1,$$

the principle of superposition requires that boundary conditions on $\mathscr{L}_4$ be homogeneous. It remains to be shown that all operators $\mathscr{L}_i$ can be discretized on the domain Ω^{n+1} to $\mathbb{O}(h^2)$. The operators $\mathscr{L}_i$ must be centered correctly, as given by (2.45) and (2.47), to second-order in time, except for $\mathscr{L}_4$, which may be first-order in time.

Consider the heat equation

$$\begin{aligned} u_t &= K\Delta u + f, \\ u(\boldsymbol{x},0) &= u_0(\boldsymbol{x}), \\ u(\boldsymbol{x},t) &= u_{\mathrm{bc}}(\boldsymbol{x},t) \quad \text{on } \delta\Omega. \end{aligned} \tag{2.50}$$

Let $\boldsymbol{x}^0$ be a point on $\delta\Omega$, and let $\boldsymbol{x}^1$ be an arbitrary point $\mathbb{O}(h)$ away from $\boldsymbol{x}^0$. Then a Taylor series expansion gives

$$u(\boldsymbol{x}^1,t) = u(\boldsymbol{x}^0,t) + (\boldsymbol{x}^1-\boldsymbol{x}^0)\cdot\nabla u(\boldsymbol{x}^0,0) + \mathbb{O}(h^2) + \mathbb{O}(h\Delta t). \tag{2.51}$$

Therefore, if Ω^{n+1} and $\Omega(t)$ are close (in the sense that for any point $\boldsymbol{x}$ on $\delta\Omega^{n+1}$ there is a point $\boldsymbol{x}'$ on $\delta\Omega(t)$ with $|\boldsymbol{x}-\boldsymbol{x}'| = \mathbb{O}(h)$), and if one uses Dirichlet boundary conditions on Ω^{n+1} given by

$$u_{\mathrm{bc}}(\boldsymbol{x}^{n+1},t) := u_{\mathrm{bc}}(\boldsymbol{x}(t),t) + (\boldsymbol{x}^{n+1}-\boldsymbol{x}(t))\cdot\nabla u_0,$$

and if $\Delta t \propto h$, then the solution at $\boldsymbol{x}(t)$ on $\delta\Omega(t)$ will be obtained to second order in h.

The solution on the interior of a domain Ω is a linear functional of its boundary conditions, initial conditions, and forcing. For example,

$$\begin{aligned} u(\boldsymbol{x},t) = &\int_\Omega dV' G(\boldsymbol{x}\,|\,\boldsymbol{x}',t)u_0(\boldsymbol{x}') + \int_0^t dt' \int_\Omega dV' G(\boldsymbol{x}\,|\,\boldsymbol{x}',t-t')f(\boldsymbol{x}',t') \\ &+ \int_0^t dt' \int_{\delta\Omega} dS' \boldsymbol{n}'\cdot\nabla' G(\boldsymbol{x}'\,|\,\boldsymbol{x},t-t')u_{\mathrm{bc}}(\boldsymbol{x}',t') \end{aligned} \tag{2.52}$$

solves (2.50), where G is the Green's function solving

$$\begin{aligned} G_t &= K\Delta G + \delta(\boldsymbol{x}-\boldsymbol{x}'), \\ G(\boldsymbol{x}\,|\,\boldsymbol{x}',0) &= 0, \\ G(\boldsymbol{x}\,|\,\boldsymbol{x}',t) &= 0 \quad \text{for all } \boldsymbol{x} \text{ on } \delta\Omega. \end{aligned} \tag{2.53}$$

Therefore, on a domain Ω differing from Ω^{n+1} by $\mathbb{O}(h)$, where u_0 and f are continued by high-order interpolation, and where u_{bc} is second-order accurate, the solution interior to Ω will be second-order accurate. Solved by a discrete method, the error will be the lower of the order of the method or h^2, in the present case $O(h^2)+O(\Delta t)$ for the solution by forward or backward Euler, and $O(h^2)+O(\Delta t^2)$ embedded in the Twizell et al. framework (2.49a).

The discretization of this heat solver is based on the conservative but unstable discretization of the Laplacian for time-stationary geometries

$$\Delta u = \nabla\cdot\boldsymbol{F}, \quad \boldsymbol{F} = \nabla u, \tag{2.54a}$$

$$u^{n+1} = u^n + \mathscr{L}_{\boldsymbol{i}}(\boldsymbol{u}), \tag{2.54b}$$

$$\mathscr{L}_{\boldsymbol{i}}(\boldsymbol{u}) = \frac{\nu \Delta t}{\kappa_{\boldsymbol{i}} h} \sum_{d}^{D} \left(\left(\alpha_{\boldsymbol{i}+{}^1\!/_2 \boldsymbol{e}_d} \boldsymbol{F}^{n+{}^1\!/_2}_{\boldsymbol{i}+{}^1\!/_2 \boldsymbol{e}_d} - \alpha_{\boldsymbol{i}-{}^1\!/_2 \boldsymbol{e}_d} \boldsymbol{F}^{n+{}^1\!/_2}_{\boldsymbol{i}-{}^1\!/_2 \boldsymbol{e}_d} \right) + \left(\alpha_{\boldsymbol{i}-{}^1\!/_2 \boldsymbol{e}_d} - \alpha_{\boldsymbol{i}+{}^1\!/_2 \boldsymbol{e}_d} \right) \boldsymbol{F}_{d,\boldsymbol{i},\mathrm{EB}} \right). \tag{2.54c}$$

Note that while $\mathscr{L}_{\boldsymbol{i}}$ is unstable in the limit $\kappa_{\boldsymbol{i}} \to 0$, $\kappa_{\boldsymbol{i}} \mathscr{L}_{\boldsymbol{i}}$ is stable. The overall sequence can be written in a stable manner as follows:

$$\boldsymbol{\psi}^1 = \boldsymbol{\kappa}(\boldsymbol{I} + \mu_4 \mathscr{L}) \boldsymbol{f}^{n+1/2}, \tag{2.55a}$$
$$\boldsymbol{\psi}^2 = \boldsymbol{\kappa}(\boldsymbol{I} + \mu_3 \mathscr{L}) \boldsymbol{u}^n, \tag{2.55b}$$
$$\boldsymbol{\psi}^3 = \Delta t \boldsymbol{\psi}^1 + \boldsymbol{\psi}^2, \tag{2.55c}$$
$$\boldsymbol{\psi}^4 = [\boldsymbol{\kappa}(\boldsymbol{I} - \mu_2 \mathscr{L})]^{-1} \boldsymbol{\psi}^3, \tag{2.55d}$$
$$\boldsymbol{\psi}^5 = \boldsymbol{\kappa} \boldsymbol{\psi}^4, \tag{2.55e}$$
$$\boldsymbol{u}^{n+1} = [\boldsymbol{\kappa}(\boldsymbol{I} - \mu_1 \mathscr{L})]^{-1} \boldsymbol{\psi}^5. \tag{2.55f}$$

2.5. ***Computation of space-time geometry.*** We base our geometry calculation on a hierarchical application of the divergence theorem proposed by Ligocki et al. [21], here specialized to the case where geometric information is to be determined from cell- and time-centered discrete values of a level set function ψ. This method assumes only that ψ is a sufficiently differentiable level set, not necessarily a distance function.

2.5.1. *Governing equations.* In D dimensions use the multiindex convention

$$\boldsymbol{x}^{\boldsymbol{p}} = x_1^{p_1} x_2^{p_2} \cdots x_D^{p_D}, \tag{2.56a}$$
$$\boldsymbol{p}! = p_1! p_2! \cdots p_D!, \tag{2.56b}$$
$$\nabla^{\boldsymbol{r}} = \frac{\partial^{r_1}}{\partial x_1^{r_1}} \frac{\partial^{r_2}}{\partial x_2^{r_2}} \cdots \frac{\partial^{r_D}}{\partial x_D^{r_D}}, \tag{2.56c}$$

and in this application all components of a multiindex are nonnegative. We will say multiindex integer $\boldsymbol{p}$ is even if all p_i are even, and for the magnitude, $P = |\boldsymbol{p}| = \sum p_i$, etc.

Consider the volume integral of $\nabla \cdot (\boldsymbol{x}^{\boldsymbol{p}} \boldsymbol{e}_d) = p_d \boldsymbol{x}^{\boldsymbol{p}-\boldsymbol{e}_d}$ with the divergence theorem:

$$p_d \int_V \boldsymbol{x}^{\boldsymbol{p}-\boldsymbol{e}_d} dV = \int_{A_d^+} \boldsymbol{x}^{\boldsymbol{p}} dA - \int_{A_d^-} \boldsymbol{x}^{\boldsymbol{p}} dA + \int_{A_{\mathrm{EB}}} \boldsymbol{x}^{\boldsymbol{p}} \boldsymbol{n} \cdot \boldsymbol{e}_d dA \tag{2.57}$$

where EB denotes the embedded boundary, and $\boldsymbol{n}$ is the unit normal vector. With the boundary having curvature, $\boldsymbol{n}$ is spatially varying. Account for this spatial variance with a truncated Taylor series:

$$p_d \int_V \boldsymbol{x}^{\boldsymbol{p}-\boldsymbol{e}_d} dV - n_d \int_{A_{\mathrm{EB}}} \boldsymbol{x}^{\boldsymbol{p}} dA$$
$$= \int_{A_d^+} \boldsymbol{x}^{\boldsymbol{p}} dA - \int_{A_d^-} \boldsymbol{x}^{\boldsymbol{p}} dA + \sum_{1 \le |\boldsymbol{r}| \le R} \frac{\nabla^{\boldsymbol{r}} n_d}{\boldsymbol{r}!} \int_{A_{\mathrm{EB}}} \boldsymbol{x}^{\boldsymbol{r}+\boldsymbol{p}} dA + \mathbb{O}(h^{D+R+P}). \quad (2.58)$$

Here V designates a generalized volume, and A designates a codimension-1 subspace — a generalized area. In (2.58), $\boldsymbol{n}$ is the normal to the EB in the space of volume V. This equation expresses moments on $[-h/2, h/2]^D$ and on the codimension-1 EB in terms of higher moments on lower-dimensional spaces. Each of these lower-dimensional spaces can be analyzed with a similar specialization of (2.58). For example, if the area A_{d^+} is bounded by subspaces L (lines), we have

$$p_{d'} \int_{A_{d^+}} \boldsymbol{x}^{\boldsymbol{p}-\boldsymbol{e}_{d'}} dA - n_{d'} \int_{L_{\mathrm{EB}}} \boldsymbol{x}^{\boldsymbol{p}} dL$$
$$= \int_{L_{d'}^+} \boldsymbol{x}^{\boldsymbol{p}} dL - \int_{L_{d'}^-} \boldsymbol{x}^{\boldsymbol{p}} dL + \sum_{1 \le |\boldsymbol{r}| \le R} \frac{\nabla^{\boldsymbol{r}} n_{d'}}{\boldsymbol{r}!} \int_{L_{\mathrm{EB}}} \boldsymbol{x}^{\boldsymbol{r}+\boldsymbol{p}} dL + \mathbb{O}(h^{D'+R+P}), \quad (2.59)$$

where $\boldsymbol{n}$ is the interface normal in the subspace A_{d^+}, and $D' = D - 1$ if we consider p_d (the component of $\boldsymbol{p}$ in the dimension orthogonal to space A_{d^+}) to be zero. (This assumption can be made without loss of generality. If $M(\boldsymbol{p})$ is a given moment on surface $A_{d^\pm}$ with $p_d = 0$, then $M(\boldsymbol{p} + k\boldsymbol{e}_d) = (\pm h/2)^k M(\boldsymbol{p})$: the generation of moments for which $p_d \neq 0$ is trivial.) Equation (2.58) can be applied as many times as needed until the subspaces contain trivial normal vectors $\boldsymbol{n}$: when the space V of (2.58) is 1D, the normal vector is ± 1 and has no derivative.

To interpret the order $D' + R + P$, begin by specifying S as the desired order of accuracy. On the original space $R = S - 1$, and $P = 0, 1$ is required at a minimum to obtain the centroid of the EB. However, with $R = 1$ and $P = 1$, EB moments with $P = 2$ are required on the right hand sides. This causes the maximum P, $P_{\max}$, to depend on S and D' in a systematic way:

$$P_{\max}(D') = S - 1 + [D - \max(D',\ 2)] \qquad D \ge 2, \quad (2.60)$$

and, for each magnitude $P = 0, \ldots, P_{\max}$,

$$R = \max(S - 1 - P,\ 0). \quad (2.61)$$

Table 1 displays some convergence results in multiple dimensions for the case $S = 2$.

2.5.2. *Order of operations.* For each dimension, the system of equations implied by (2.58) is overdetermined and nonsingular. Ligocki et al. propose evaluating this hierarchical system in a particular way, grouping equations on a common subspace and with common P. This makes each overdetermined set small, minimizing the

2D		
$1/h$	error	rate
16	$-8.143 \cdot 10^{-4}$	
32	$-2.226 \cdot 10^{-4}$	1.87
64	$-5.409 \cdot 10^{-5}$	2.04
128	$-1.378 \cdot 10^{-5}$	1.97

3D		
$1/h$	error	rate
16	$-3.108 \cdot 10^{-3}$	
32	$-7.873 \cdot 10^{-4}$	1.98
64	$-1.958 \cdot 10^{-4}$	2.01
128	$-4.898 \cdot 10^{-5}$	2.00

(2+1)D		
$1/h$	error	rate
16	$-6.705 \cdot 10^{-5}$	
32	$-1.796 \cdot 10^{-5}$	1.90
64	$-4.416 \cdot 10^{-6}$	2.02
128	$-1.115 \cdot 10^{-6}$	1.98

(3+1)D		
$1/h$	error	rate
16	$-2.107 \cdot 10^{-4}$	
32	$-5.330 \cdot 10^{-5}$	1.98
64	$-1.327 \cdot 10^{-5}$	2.01
128	$-3.315 \cdot 10^{-6}$	2.00

Table 1. Convergence of EB area for sections of a hypersphere for order $S = 2$. In 2D the area of a unit circle is computed on one quadrant. In 3D, the area of a unit sphere in one octant. In 2+1D, a section of the unit sphere from the midplane to $x_2 = 1/16$. In 3+1D, the area of a unit hypersphere from the midplane to $x_3 = 1/16$. Calculations used cell-centered values of the signed distance function to derive all quantities.

cumulative cost of the associated linear algebra. Here, we first describe the order of operation as described by Ligocki et al., then discuss constraints and modifications to the operation order that are made to accommodate them.

To illustrate these ideas, consider the 2D case. Let us write as a subscript $[\cdot\cdot]$ to indicate that the volume being integrated over is $[-h/2, h/2] \times [-h/2, h/2]$, and $[+\cdot]$ to indicate the $+x_0$ edge on which the integral runs $[-h/2, h/2]$ in the x_1 direction (Figure 7). We will write $(p_0 p_1)$ to represent a given moment. Thus,

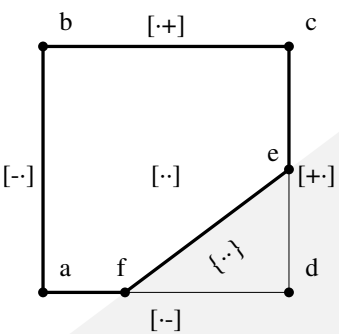


Figure 7. Notation for 2D example. The fluid region $abcde$ is denoted $[\cdot\cdot]$; the EB $\{\cdot\cdot\}$, $\overline{ef}$, separates the fluid from the shaded exterior region def. The 1D subregion $[+.]$ is the line segment $\overline{cd}$, etc. The calculation begins with $P = 0$ moments on the 1D subregions, e.g., $(00)_{[.-]} = af$, then the $P = 1$ moments; e.g., $(10)_{[.-]} = x_f^2/2 - h^2/8$, and $(01)_{[.-]} = -(af)h/2$ which is simply $(00)_{[.-]}$ multiplied by the x_1 coordinate of the edge, $-h/2$.

$(10)_{[\cdot -]}$ is the first x moment of the bottom edge of the cell.

In support of the 2D computation, we need the $P = 0$ and $P = 1$ moments over each edge (each of the four 1D bounding spaces). These quantities are determined by interpolation of the discrete level set data using stencils and methods described below (Section 2.5.4).

Once these 1D moments are known, one can proceed to evaluate the moments in 2D. In volume $[\cdot\cdot]$ we are interested in the $P = 0$ moment $(00)_{[\cdot\cdot]}$ which gives the volume fraction. We are also interested in the $P = 0$ and $P = 1$ moments over the EB, which together specify the centroid. The EB in volume $[\cdot\cdot]$ will be written $\{\cdot\cdot\}$. The first block of equations come from (2.58) with $P = 1$. In order, these are from $\boldsymbol{p} = (1, 0)$ with $d = 0$, then $d = 1$, followed by $\boldsymbol{p} = (0, 1)$ with $d = 0$, then $d = 1$:

$$\begin{aligned} 1(00)_{[\cdot\cdot]} - n_{0[\cdot\cdot]}(10)_{\{\cdot\cdot\}} &= (10)_{[+\cdot]} - (10)_{[-\cdot]}, \\ -n_{1[\cdot\cdot]}(10)_{\{\cdot\cdot\}} &= (10)_{[\cdot+]} - (10)_{[\cdot-]}, \\ -n_{0[\cdot\cdot]}(01)_{\{\cdot\cdot\}} &= (01)_{[+\cdot]} - (01)_{[-\cdot]}, \\ 1(00)_{[\cdot\cdot]} - n_{1[\cdot\cdot]}(01)_{\{\cdot\cdot\}} &= (01)_{[\cdot+]} - (01)_{[\cdot-]}. \end{aligned} \tag{2.62}$$

With the unknowns on the left hand side, there are 4 equations to determine 3 variables. The next set of equations come from (2.58) with $P = 0$, $\boldsymbol{p} = (0, 0)$, with $d = 0$ followed by $d = 1$:

$$\begin{aligned} -n_{0[\cdot\cdot]}(00)_{\{\cdot\cdot\}} = \quad & (00)_{[+\cdot]} - (00)_{[-\cdot]} + n^{(10)}_{0[\cdot\cdot]}(10)_{\{\cdot\cdot\}} + n^{(01)}_{0[\cdot\cdot]}(01)_{\{\cdot\cdot\}}, \\ -n_{1[\cdot\cdot]}(00)_{\{\cdot\cdot\}} = \quad & (00)_{[\cdot+]} - (00)_{[\cdot-]} + n^{(10)}_{1[\cdot\cdot]}(10)_{\{\cdot\cdot\}} + n^{(01)}_{1[\cdot\cdot]}(01)_{\{\cdot\cdot\}}; \end{aligned} \tag{2.63}$$

two equations in one unknown. This system requires the normal and its gradient, which may be constructed from a degree-2 Taylor series expansion of the level set.

From these computations, volume fraction, centroids and apertures are, e.g.,

$$\kappa = \frac{(00)_{[\cdot\cdot]}}{h^2}, \tag{2.64}$$

$$\boldsymbol{x}^{\text{cent}}_{[-\cdot]} = \frac{1}{(00)_{[-\cdot]}} \begin{pmatrix} (10)_{[-\cdot]} \\ (01)_{[-\cdot]} \end{pmatrix}, \tag{2.65}$$

$$\boldsymbol{x}^{\text{cent}}_{\{\cdot\cdot\}} = \frac{1}{(00)_{\{\cdot\cdot\}}} \begin{pmatrix} (10)_{\{\cdot\cdot\}} \\ (01)_{\{\cdot\cdot\}} \end{pmatrix}, \tag{2.66}$$

$$\alpha_{[-\cdot]} = \frac{(00)_{[-\cdot]}}{h}; \tag{2.67}$$

see (2.27), (2.33a), (2.28). While an EB area is calculated by this method, finite volume discretizations use the projected area and the normal that come from the requirement that $\nabla \cdot (\boldsymbol{e}_i) = 0$ [29].

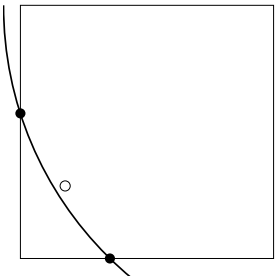


Figure 8. Recentering to improve accuracy. When the 1D edges of a given volume are evaluated, the intersections of the edges with $\psi = 0$ are discovered (filled circles). In the evaluation of higher-dimensional volumes, here a 2D face, the mean of the intersection points of those edges associated with this face gives a centering point (open circle) which approximates the centroid of the EB.

Ligocki (personal communication) noted that the quality of the least squares solutions can be dramatically improved by recentering the calculation from the center of a given $[-h/2, h/2]^D$ volume to a point close to the centroid of the EB. Specifically, we recenter the linear equation systems and the constraint equations prior to solution of the over-determined data fitting equations by Householder reduction, then recenter the computed result to the center of the given volume. The estimated centroid is the average of the intersections of $\psi = 0$ with the 1D edges of the volume being evaluated (Figure 8).

2.5.3. *Incorporation of constraints.* The moments appearing in this expansion are subject to certain inequality constraints. If $\bar{\boldsymbol{p}}$ is even, then the corresponding volume integral is nonnegative and, if not on the EB, can be bounded from above:

$$0 \leq \int_V \boldsymbol{x}^{\bar{\boldsymbol{p}}} dV \leq \frac{h^{P+D} \boldsymbol{p}!}{2^P (\boldsymbol{p}+\mathbf{1})!}. \tag{2.68}$$

If $\boldsymbol{p}$ differs from an even multiindex $\bar{\boldsymbol{p}}$ by addition of a unit basis vector $\boldsymbol{e}_j$, then

$$\min_V(x_j) \int_V \boldsymbol{x}^{\bar{\boldsymbol{p}}} dV \leq \int_V \boldsymbol{x}^{\bar{\boldsymbol{p}}+\boldsymbol{e}_j} dV \leq \max_V(x_j) \int_V \boldsymbol{x}^{\bar{\boldsymbol{p}}} dV, \tag{2.69}$$

by the mean value theorem.

In the second-order 2D example above, simple positivity constraints are

$$(00)_{[\cdot\cdot]} \geq 0, \tag{2.70a}$$

$$(00)_{\{\cdot\cdot\}} \geq 0, \tag{2.70b}$$

and there is a physical constraint

$$(00)_{[\cdot\cdot]} \leq h^2; \tag{2.70c}$$

volume fraction is positive but less than or equal to one, and the EB area is positive. Constraints of the second type are

$$-\frac{h}{2}(00)_{\{..\}} \le (10)_{\{..\}} \le +\frac{h}{2}(00)_{\{..\}}, \tag{2.71a}$$

$$-\frac{h}{2}(00)_{\{..\}} \le (01)_{\{..\}} \le +\frac{h}{2}(00)_{\{..\}}. \tag{2.71b}$$

Constraints of type (2.68) can be implemented with any organization of the divergence theorem hierarchy. However, to incorporate those derived from the mean value theorem (2.69) while minimizing the overall least squares problem, it is necessary to solve for all necessary moments of a given volume simultaneously. This can be seen by noting that constraints (2.71) combine EB area values $(00)_{\{..\}}$ and EB moment values $(10)_{\{..\}}$ and $(01)_{\{..\}}$ which are determined in different blocks (2.63) and (2.62), respectively, of the Ligocki et al. algorithm.

Incorporation of constraints in that setting means that the first linear system is solved without constraints, then constraints may be incorporated in subsequent solves. This would be analogous to weighing system (2.62) in preference to (2.63). This relative priority cannot be justified. To correct this weighting problem we solve simultaneously for all moments of a given subspace: (i) 1D moments as above, (ii) solve system (2.62), (2.63) together. We explicitly weigh each equation by h^{-P} so that, unconstrained, they carry similar weights as in the Ligocki et al. method.

All linear systems are solved with Householder $\boldsymbol{QR}$ reduction. The constrained least squares problem is equivalent to the constrained positive definite quadratic programming problem solved by Goldfarb and Idnani [13; 14]: minimize

$$(\boldsymbol{Ax}-\boldsymbol{b})^T(\boldsymbol{Ax}-\boldsymbol{b})$$

with respect to $\boldsymbol{x}$ subject to linear inequality constraints. Their method begins with the Cholesky $\boldsymbol{LL}^T$ decomposition of the Hessian $\boldsymbol{A}^T\boldsymbol{A}$, and with $\boldsymbol{Q}$ unitary the setup phase of their method is trivial: $\boldsymbol{L}=\boldsymbol{R}^T$, the transpose of $\boldsymbol{R}$ from the Householder decomposition. The quadratic form $(\boldsymbol{Ax}-\boldsymbol{b})^T(\boldsymbol{Ax}-\boldsymbol{b})$ never need be explicitly constructed.

2.5.4. *Stencils.* Here algorithms are described that determine the moments on 1D subspaces, and derivatives of the normal vector, from cell- and time-centered level set discretizations.

Nominally, we assume that the EB $\psi=0$ will intersect each 1D edge at most once. If this is true, then interpolated values of ψ at the corners of a cell determine which edges are intersected by the EB, which are covered (by the wall), and which are regular. It is important to the robustness of the method that these corner values be accurate, and that each edge's notion of the corner be identical: the corner

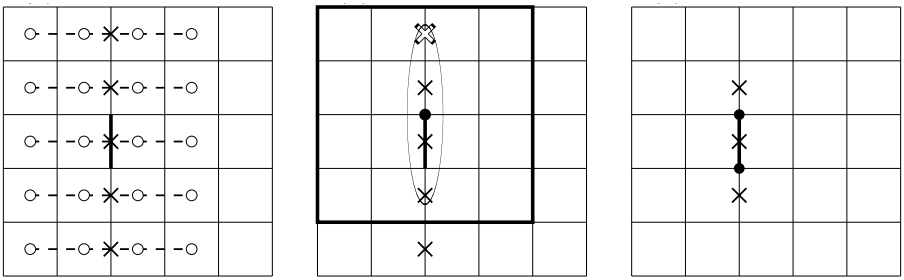


Figure 9. Stencils for the construction of 1D moments for the case where all dimensions are spatial, illustrated in 2D for the case of order $S = 3$ or 4. To achieve a final order S, stencils of half width $K = \lceil S/2 \rceil$ are constructed. (a) To compute the moments on the left edge of the center cell (bold line), a stencil consists of $2K$ points transverse to the edge, and $2K + 1$ points in the direction of the edge (open circles). The first step is to interpolate to the line in the transverse direction. The points being interpolated lie on the dashed lines, and the resulting interpolants are given by crosses. The calculation of the derivatives of $\boldsymbol{n}$ for the center cell is based on a least squares fit of all $(2K + 1)^D$ cells (the squares) to determine Taylor coefficients in a centered expansion of ψ. (b) The top $2K$ interpolants are interpolated to deduce the value at the top end of the bold line segment (filled circle). The polynomial given by this filled circle and the bottom $2K$ crosses is identical to the polynomial given by all crosses alone, so the top cross may be dropped when the filled circle is added to the list of support points. Similarly, the bottom $2K$ points are used to interpolate the value at the bottom of the line segment. The result is that the corner values of the cell are computed from a symmetric $(2K)^D$ set of points, and for all cells that share a given corner the stencil is identical (e.g., the value at the corner indicated by the filled circle is determined by the set of points in the bold square, regardless of the edge under consideration). (c) The resulting $2K + 1$ interpolation points — equivalently, the $2K + 1$ crosses of part (a) — define an interpolation polynomial whose roots are the intersection of $\psi = 0$ with the given edge.

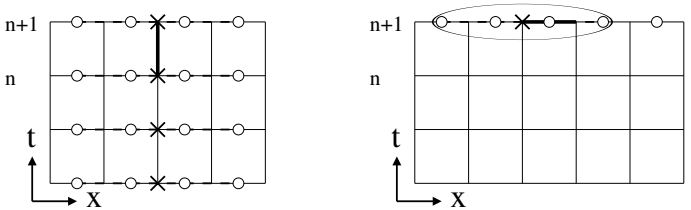


Figure 10. Stencils for the case where one dimension is temporal, illustrated in 1D+1D for the case of order $S = 4$. Let $K = \lceil S/2 \rceil$. Because data is centered at time levels, stencils for time and space edges are different. (a) To find moments on a temporal edge (bold line segment) S time levels are interpolated (crosses) each from $2K$ spatial interpolations (circles on dashed lines). (b) For a spatial edge, data on a single time level is treated by the stencil described in Figure 9. To evaluate derivatives of ψ at the center of a space-time volume, the stencil uses $2K + 1$ points in each spatial direction and $S + 1$ time levels.

values must be cell- and edge-invariant. In order that quantities like the aperture α be invariant, it is also important that the intersection point of $\psi = 0$ with a given edge be cell-invariant. These symmetry considerations impact the interpolation algorithms by rounding up the stencil width in some cases.

To compute the moments on 1D edges, one finds the intersection of the edge with $\psi = 0$ (say a point $\boldsymbol{\zeta}$), then constructs the moments explicitly. In a frame where the cell center is at the origin, one has, for example,

$$(p_0 p_1)_{[-\cdot]} = \left(-\frac{h}{2}\right)^{p_0} \times \begin{cases} \int_{-h/2}^{\zeta} y^{p_1} dy & \text{if } n_y = -1, \\ \int_{\zeta}^{h/2} y^{p_1} dy & \text{if } n_y = +1, \end{cases} \tag{2.72}$$

The intersection point $\boldsymbol{\zeta}$ is determined by constructing an interpolating polynomial using data interpolated to the line coincident with the edge. We seek its roots with bisection until Smale's criterion [34] indicates that Newton–Raphson will converge quadratically. Roots are then refined with Newton–Raphson.

For the general case of arbitrary dimension D and arbitrary order S, $\mathbb{O}(h^S)$ accuracy on the 1-dimensional subspaces requires an interpolation polynomial with S support points. The symmetry invariance requirement of the method modifies this stencil. If $K = \lceil S/2 \rceil$, then $2K$ support points are required in the transverse direction and $2K + 1$ in the normal direction (Figure 9).

The support requirements in the case of space-time interpolation are simpler since data exists on the time edges so interpolation to integer time levels is not required (Figure 10).

To achieve order S accuracy, $S - 1$ order derivatives of the normal vector are required, which are based on S order derivatives of the discrete level set using

$$\boldsymbol{n}^{(\boldsymbol{p})} = \frac{d^{\boldsymbol{p}}}{d\boldsymbol{x}^{\boldsymbol{p}}} \frac{\nabla \psi}{\sqrt{(\nabla \psi) \cdot (\nabla \psi)}}. \tag{2.73}$$

These derivatives are based on a Taylor series centered at the center of the relevant subspace, fit to data with stencil width $S + 1$. Where possible the stencil is made symmetric by rounding up to width $2K + 1$.

2.5.5. *Underresolved and nonconforming geometry.* Underresolved geometries may fail under the standard algorithm. The geometry in Figure 11 will fail because the interpolated value of ψ at the corners of the square cell are all positive. The algorithm therefore misses the fact that the EB crosses the left edge twice. One way to detect these problems is to estimate the minimum and maximum values of ψ on the cell. If these have different signs, then the cell is irregular even when the corner values have uniform sign, and even if ψ is not a distance function.

An algorithm to estimate the range of values the differentiable function ψ takes on the cell is given by Rivlin [31]. The basic idea is to sample the domain $\Omega_{\boldsymbol{i}}$ by

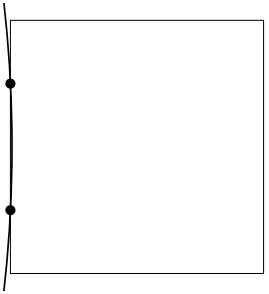


Figure 11. Irregular cells whose interpolated corner values have uniform sign. In this example, the cell is a square of length h and $\psi = 0$ is a circle centered $5\,{}^1\!/_2 h$ units to the left of the cell center. The radius is chosen so the circle intersects the left cell boundary at $\pm h/4$. The area to be measured is $\approx 2.08 \times 10^{-3} h^2$.

overlaying it with a grid of length δ. If $\psi(\boldsymbol{\xi})$ is an extremum in cell $\boldsymbol{i}$, and $\boldsymbol{x_k}$ is a point on the δ-grid, then

$$\psi(\boldsymbol{x_k}) = \psi(\boldsymbol{\xi}) + \sum_{|\boldsymbol{r}|=2} \frac{(\boldsymbol{x_k} - \boldsymbol{\xi})^{\boldsymbol{r}}}{\boldsymbol{r}!} \psi^{\boldsymbol{r}}(\boldsymbol{\chi}), \tag{2.74a}$$

$$\max_{\boldsymbol{x} \in \Omega_{\boldsymbol{i}}} |\psi(\boldsymbol{x}) - \psi(\boldsymbol{\xi})| \le \Delta \equiv \frac{\delta^2}{4} \max_{\boldsymbol{\chi}} \sum_{|\boldsymbol{r}|=2} \frac{1}{\boldsymbol{r}!} |\psi^{(\boldsymbol{r})}(\boldsymbol{\chi})| \tag{2.74b}$$

for some $\boldsymbol{\chi} \in [\boldsymbol{x_k}, \boldsymbol{\xi}]$, and so

$$\min_{\boldsymbol{x} \in \Omega_{\boldsymbol{i}}} \psi(\boldsymbol{x}) > \min_{\boldsymbol{x_k}} \psi(\boldsymbol{x_k}) - \Delta, \tag{2.75a}$$

$$\max_{\boldsymbol{x} \in \Omega_{\boldsymbol{i}}} \psi(\boldsymbol{x}) < \min_{\boldsymbol{x_k}} \psi(\boldsymbol{x_k}) + \Delta. \tag{2.75b}$$

We estimate Δ using the Taylor series, which we center at the center of cell $\Omega_{\boldsymbol{i}}$:

$$\psi(\boldsymbol{x}) = \sum_{P} \sum_{|\boldsymbol{p}|=P} \frac{\boldsymbol{x}^{\boldsymbol{p}}}{\boldsymbol{p}!} \psi^{(\boldsymbol{p})}(\boldsymbol{0}), \tag{2.76a}$$

$$\Delta = \frac{\delta^2}{4} \max_{\boldsymbol{\chi}} \sum_{|\boldsymbol{r}|=2} \frac{1}{\boldsymbol{r}!} \left| \sum_{P} \sum_{|\boldsymbol{p}|=P} \frac{\boldsymbol{\chi}^{\boldsymbol{p}-\boldsymbol{r}}}{(\boldsymbol{p}-\boldsymbol{r})!} \psi^{(\boldsymbol{p})}(\boldsymbol{0}) \right|, \tag{2.76b}$$

$$\le \frac{\delta^2}{4} \sum_{|\boldsymbol{r}|=2} \frac{1}{\boldsymbol{r}!} \sum_{P} \sum_{|\boldsymbol{p}|=P} \frac{(\frac{1}{2}\boldsymbol{h})^{\boldsymbol{p}-\boldsymbol{r}}}{(\boldsymbol{p}-\boldsymbol{r})!} \left| \psi^{(\boldsymbol{p})}(\boldsymbol{0}) \right| \tag{2.76c}$$

where $\boldsymbol{h}$ is the vector cell edge lengths. In support of (2.73), derivatives of ψ through order S are known. So, for any order $S \ge 2$ sufficient information will be available to employ Rivlin's method. Given a desired tolerance Δ, (i) approximate the Taylor series by least squares, (ii) estimate δ from (2.76c), then compute the

order S	relative error
2	−0.595
3	−0.0281
4	−0.0304
5	0.00832
6	−0.000256

Table 2. Relative area error $(A^h - A)/A$ (where A is exact and A^h is computed) using one level of bisection to resolve Figure 11. Without subdivision, the relative error is 1.

bounds by sampling the polynomial. If the product of bounds $\psi_{\min}\psi_{\max}$ is negative, then subdivision is applied. Otherwise, the cell is regular $\kappa = 1$ or covered $\kappa = 0$.

For the situation in Figure 11, a single bisection (in all directions) permits identification of the cell as an irregular one. The resulting volume calculations are summarized in Table 2.

3. Results

We demonstrate the method and show its convergence by computing the flow past a sphere in a bounded domain, Figure 12. In arbitrary units, the domain has length 4 and height 2. The top and bottom boundaries are stationary no-slip walls, the right boundary is outflow, and the left domain boundary is inflow with velocity having a Poiseuille profile with maximum velocity 1.5. Viscosity is 0.1. A sphere centered at (1, 1) obstructs the flow. Its radius depends on time as

$$0.2 + 0.1 \cos \omega t,$$

with $\omega = \pi/1.2$. The finest discretization of the domain is 1024×512, with $\Delta t = 1.5 \times 10^{-3}$ fixed. To determine rates of convergence we also use coarser grids: a 512×256 grid with $\delta t = 3.0 \times 10^{-3}$, etc., through the coarsest discretization

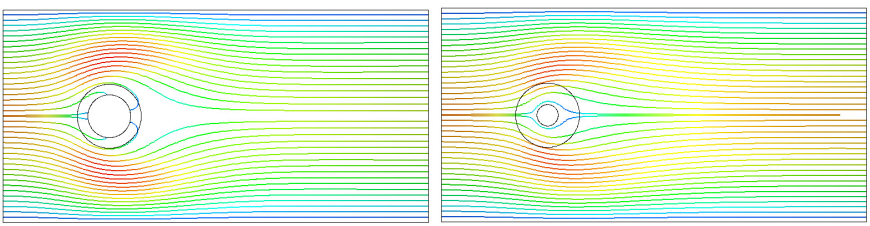


Figure 12. Flow past a shrinking sphere on 2:1 domain. Circles represent initial and final sphere surface. Curves are streamlines. Color corresponds to $|\boldsymbol{u}|$ from 0 (blue) to 1.7 (red). Note that the streamlines attach to the sphere because it is moving. Times are 0.6 and 1.2, respectively.

N_x	$\|u_x \text{ error}\|_\infty$	rate	$\|u_x \text{ error}\|_1$	rate
64/128	5.28×10^{-1}		4.37×10^{-3}	
128/256	2.24×10^{-1}	1.24	9.24×10^{-4}	2.24
256/512	8.76×10^{-2}	1.35	2.22×10^{-4}	2.06
512/1024	4.64×10^{-2}	0.92	5.36×10^{-5}	2.05
N_x	$\|u_y \text{ error}\|_\infty$	rate	$\|u_y \text{ error}\|_1$	rate
64/128	4.01×10^{-1}		3.12×10^{-3}	
128/256	1.86×10^{-1}	1.11	7.07×10^{-4}	2.14
256/512	8.28×10^{-2}	1.16	1.68×10^{-4}	2.07
512/1024	3.97×10^{-2}	1.06	3.95×10^{-5}	2.09

Table 3. Richardson error convergence study for flow past a shrinking sphere.

of 64×32 with $\Delta t = 2.4 \times 10^{-2}$. The maximum CFL over the course of this simulation is 0.8.

Errors and rates of convergence are shown in Table 3 after 352 time steps on the finest grid through 22 time steps on the coarsest. In L_1 the velocity is second-order accurate, while in L_∞ it is first-order. The errors reported are Richardson estimates obtained by comparing computations with different resolution:

$$\|u\|_1^{h,2h} = \frac{1}{V}\int_\Omega |u^h - u^{2h}| dV = \frac{\sum_{\boldsymbol{i}} \kappa_{\boldsymbol{i}} |u^h - u^{2h}|_{\boldsymbol{i}}}{\sum_{\boldsymbol{i}} \kappa_{\boldsymbol{i}}}, \tag{3.1a}$$

$$\|u\|_\infty^{h,2h} = \max_{\boldsymbol{x}\in\Omega} |u^h - u^{2h}| = \max_{\boldsymbol{i}} |u^h - u^{2h}|_{\boldsymbol{i}}. \tag{3.1b}$$

In these expressions, $\boldsymbol{i}$ is a cell index in the $2h$-grid, and

$$|u^h - u^{2h}|_{\boldsymbol{i}} = \left| \boldsymbol{u}_{\boldsymbol{i}}^{2h} - \frac{1}{2^D} \sum_{\boldsymbol{j}} \frac{\kappa_{\boldsymbol{j}}^h}{\kappa_{\boldsymbol{i}}^{2h}} \boldsymbol{u}_{\boldsymbol{j}}^h \right| \tag{3.2}$$

with the sum being over h-grid cells $\boldsymbol{j}$ that lie in the $2h$-grid cell. The convergence rate is given by

$$r = \frac{1}{\ln 2} \ln \frac{\|u\|^{2h,4h}}{\|u\|^{h,2h}}. \tag{3.3}$$

The first-order convergence in L_∞ is expected because of the discretization error of the quadrature formula (2.31) for the hyperbolic part of the governing equations. As anticipated by Colella [10], the truncation error in irregularly shaped finite volumes is lower order than regularly shaped volumes. Thus, any fully conservative and consistent finite volume hyperbolic method based on a quadrature rule consisting of one point per bounding surface will be first-order in L_∞. This expectation applies also to approaches like cell merging.

Acknowledgments

This work builds on a flow solver for the incompressible Navier–Stokes equations on a fixed domain, first written by D. T. Graves at the Lawrence Berkeley National Laboratory. We thank him for many helpful suggestions in the course of this project. T. J. Ligocki, also at LBNL, was a valuable resource for computational geometry.

References

[1] J. B. Bell, P. Colella, and H. M. Glaz, *A second-order projection method for the incompressible Navier–Stokes equations*, J. Comput. Phys. **85** (1989), no. 2, 257–283. MR 90i:76002 Zbl 0681.76030

[2] M. J. Berger, C. Helzel, and R. J. Leveque, *h-box methods for the approximation of hyperbolic conservation laws on irregular grids*, SIAM J. Numer. Anal. **41** (2003), no. 3, 893–918. MR 2004g:65103 Zbl 1066.65082

[3] D. L. Brown and M. L. Minion, *Performance of under-resolved two-dimensional incompressible flow simulations, II*, J. Comput. Phys. **138** (1997), no. 1, 734–765. Zbl 0914.76063

[4] I.-L. Chern and P. Colella, *A conservative front tracking method for hyperbolic conservation laws*, technical report UCRL-97200, Lawrence Livermore National Laboratory, 1987.

[5] P. H. Chiu, R. K. Lin, and T. W. H. Sheu, *A differentially interpolated direct forcing immersed boundary method for predicting incompressible Navier–Stokes equations in time-varying complex geometries*, J. Comput. Phys. **229** (2010), no. 12, 4476–4500. MR 2011b:76067 Zbl 05718209

[6] A. J. Chorin and J. E. Marsden, *A mathematical introduction to fluid mechanics*, 3rd ed., Texts in Applied Mathematics, no. 4, Springer, New York, 1993. MR 94c:76002 Zbl 0774.76001

[7] A. J. Chorin, *Numerical solution of the Navier–Stokes equations*, Math. Comp. **22** (1968), 745–762. MR 39 #3723 Zbl 0198.50103

[8] ———, *On the convergence of discrete approximations to the Navier–Stokes equations*, Math. Comp. **23** (1969), 341–353. MR 39 #3724 Zbl 0184.20103

[9] ———, *Numerical solutions of incompressible flow problems*, Numerical solutions of nonlinear problems (J. M. Ortega and W. C. Rheinboldt, eds.), Studies in Numerical Analysis, no. 2, Soc. Indust. Appl. Math., Philadelphia, 1970, pp. 64–71. MR 42 #2732

[10] P. Colella, *Volume-of-fluid methods for partial differential equations*, Godunov methods (E. F. Toro, ed.), Kluwer, New York, 2001, pp. 161–177. MR 1963590 Zbl 0989.65118

[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 2006i:65142 Zbl 1120.65324

[12] P. Colella and D. P. Trebotich, *Numerical simulation of incompressible viscous flow in deforming domains*, Proc. Natl. Acad. Sci. USA **96** (1999), no. 10, 5378–5381. MR 2000a:76116 Zbl 0938.76063

[13] D. Goldfarb and A. Idnani, *Dual and primal-dual methods for solving strictly convex quadratic programs*, Numerical Analysis: Proceedings of the third IIMAS workshop (J. Hennart, ed.), Springer, New York, 1982, pp. 226–239.

[14] ———, *A numerically stable dual method for solving strictly convex quadratic programs*, Math. Programming **27** (1983), no. 1, 1–33. MR 84k:90058 Zbl 0537.90081

[15] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface*, Phys. Fluids **8** (1965), 2182–2189.

[16] T. Ikeno and T. Kajishima, *Finite-difference immersed boundary method consistent with wall conditions for incompressible turbulent flow simulations*, J. Comput. Phys. **226** (2007), no. 2, 1485–1508. MR 2009e:76127 Zbl 1173.76374

[17] M. F. Lai, *A projection method for reacting flow in the zero Mach number limit*, Ph.D. thesis, University of California, Berkeley, 1993. MR 2691063

[18] P. D. Lax, *Weak solutions of nonlinear hyperbolic equations and their numerical computation*, Comm. Pure Appl. Math. **7** (1954), no. 1, 159–193. MR 16,524g Zbl 0055.19404

[19] 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 95g:65139 Zbl 0811.65083

[20] C.-C. Liau, Y.-W. Chang, C.-A. Lin, and J. M. McDonough, *Simulating flows with moving rigid boundary using immersed-boundary method*, Comput. Fluids **39** (2010), no. 1, 152–167.

[21] T. J. Ligocki, P. O. Schwartz, J. Percelay, and P. Colella, *Embedded boundary grid generation using the divergence theorem, implicit functions, and constructive solid geometry*, J. Phys. Conf. Ser. **125** (2008), 1–5.

[22] S. Marella, S. Krishnan, H. Liu, and H. S. Udaykumar, *Sharp interface Cartesian grid method. I. An easily implemented technique for 3D moving boundary computations*, J. Comput. Phys. **210** (2005), no. 1, 1–31. MR 2006e:65161

[23] D. F. Martin and K. L. Cartwright, *Solving Poisson's equation using adaptive mesh refinement*, technical report UCB/ERL M96/66, Electronics Research Laboratory, University of California, Berkeley, 1987.

[24] D. F. Martin, P. Colella, and D. Graves, *A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions*, J. Comput. Phys. **227** (2008), no. 3, 1863–1886. MR 2009g:76085 Zbl 1137.76040

[25] P. McCorquodale, P. Colella, and H. Johansen, *A Cartesian grid embedded boundary method for the heat equation on irregular domains*, J. Comput. Phys. **173** (2001), no. 2, 620–635. MR 2002h:80009 Zbl 0991.65099

[26] G. H. Miller and P. Colella, *A conservative three-dimensional Eulerian method for coupled solid-fluid shock capturing*, J. Comput. Phys. **183** (2002), no. 1, 26–82. MR 2003j:76080 Zbl 1057.76558

[27] W. F. Noh, *CEL: a time-dependent, two-space-dimensional, coupled Eulerian-Lagrangian code*, Methods in computational physics (B. Alder, S. Fernbach, and M. Rotenberg, eds.), vol. 3, Academic Press, New York, 1964, pp. 117–179.

[28] H. Pan, L. S. Pan, D. Xu, T. Y. Ng, and G. R. Liu, *A projection method for solving incompressible viscous flows on domains with moving boundaries*, Internat. J. Numer. Methods Fluids **45** (2004), no. 1, 53–78. MR 2048282 Zbl 1072.76044

[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 96d:76081 Zbl 0842.76056

[30] C. S. Peskin, *Numerical analysis of blood flow in the heart*, J. Computational Phys. **25** (1977), no. 3, 220–252. MR 58 #9389 Zbl 0403.76100

[31] T. J. Rivlin, *Bounds on a polynomial*, J. Res. Nat. Bur. Standards Sect. B **74B** (1970), 47–54. MR 43 #2174

[32] J. Saltzman, *An unsplit 3D upwind method for hyperbolic conservation laws*, J. Comput. Phys. **115** (1994), no. 1, 153–168. MR 1300337 Zbl 0813.65111

[33] C. Shen, D. Trebotich, S. Molins, D. T. Graves, B. V. Straalen, D. T. Graves, T. Ligocki, and C. I. Steefel, *High performance computations of subsurface reactive transport processes at the pore scale*, Proceedings of SciDAC 2011, Denver, CO, 2011, available at https://seesar.lbl.gov/anag/publications/treb/SciDAC2011_sim.pdf.

[34] S. Smale, *Newton's method estimates from data at one point*, The merging of disciplines: new directions in pure, applied, and computational mathematics (R. Ewing, K. Gross, and C. Martin, eds.), Springer, New York, 1986, pp. 185–196. MR 88e:65076 Zbl 0613.65058

[35] S. K. Stein and A. Barcellos, *Calculus and analytic geometry*, 5th ed., McGraw-Hill, New York, 1992.

[36] Z. Tan, K. M. Lim, and B. C. Khoo, *A level set-based immersed interface method for solving incompressible viscous flows with the prescribed velocity at the boundary*, Internat. J. Numer. Methods Fluids **62** (2010), no. 3, 267–290. MR 2010m:76061 Zbl 05662103

[37] D. P. Trebotich and P. Colella, *A projection method for incompressible viscous flow on moving quadrilateral grids*, J. Comput. Phys. **166** (2001), no. 2, 191–217. MR 2001m:76076 Zbl 1030.76044

[38] E. H. Twizell, A. B. Gumel, and M. A. Arigu, *Second-order, L_0-stable methods for the heat equation with time-dependent boundary conditions*, Adv. Comput. Math. **6** (1996), no. 3-4, 333–352 (1997). MR 97m:65164 Zbl 0872.65084

[39] H. S. Udaykumar, R. Mittal, P. Rampunggoon, and A. Khanna, *A sharp interface Cartesian grid method for simulating flows with complex boundaries*, J. Comput. Phys. **174** (2001), no. 1, 345–380.

[40] B. van Leer, *Towards the ultimate conservative difference scheme, V: A second-order sequel to Godunov's method*, J. Comput. Phys. **32** (1979), 101–136.

Received January 12, 2011. Revised July 1, 2011.

GREGORY H. MILLER: grgmiller@ucdavis.edu
Department of Chemical Engineering and Materials Science, University of California, 1 Shields Ave, Davis, CA 95616, United States

DAVID TREBOTICH: treb@lbl.gov
Applied Numerical Algorithms Group, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

COMM. APP. MATH. AND COMP. SCI.
Vol. 7, No. 1, 2012

msp

SLENDER BODY THEORY FOR STOKES FLOWS WITH REGULARIZED FORCES

RICARDO CORTEZ AND MICHAEL NICHOLAS

Existing slender body theories for the dynamics of a thin tube in a Stokes flow differ in the way the asymptotic errors depend on a small parameter defined as the radius of the body over its length. Examples are the theory of Lighthill, that of Keller and Rubinow, and that of Johnson. Slender body theory is revisited here in the more general setting of forces which are localized but smoothly varying within a small neighborhood of the filament centerline, rather than delta distributions along the centerline. Physically, this means that the forces are smoothly distributed over the cross-section of the body. The regularity in the forces produces a final expression that has built-in smoothing which helps eliminate instabilities encountered in computations with unsmoothed formulas. Consistency with standard theories is verified in the limit as the smoothing parameter vanishes, where the original expressions are recovered. In addition, an expression for the fluid velocity at locations off the slender body is derived and used to compute the flow around a filament.

1. Introduction

Slender body theories give asymptotic solutions of slender bodies (thin tubular bodies) in a viscous fluid where the small parameter of the expansion is the radius a^* of the tube divided by its length L^*. The goal is to develop an asymptotic formula that relates the velocity of the slender body's surface to forces that are consistent with that motion and are exerted along the centerline (see Figure 1). In our derivation, we nondimensionalize all spatial variables by the length of the tube, L^*, so that $a = a^*/L^*$ is the dimensionless slenderness parameter. In all derivations, the tube dimensionless length is taken to be 1 and we consider a tube with constant circular cross sections (constant a).

Different versions of the theory were developed independently in the 1970's by Lighthill [17] and by Keller and Rubinow [16] using stokeslets and dipole distributions. Johnson developed a slender body theory based on Wu's exact solution of the flow around a spheroidal body (see [13]). Later Johnson [15] made improvements

MSC2000: 76D07, 76Z99.
Keywords: slender body theory, Stokes flow.

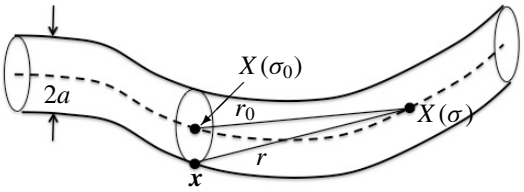


Figure 1. Schematic of a portion of the slender body. All spatial variables are scaled by the tube length so that a is a dimensionless slenderness parameter.

to the theories by adding higher order singularities near the slender body endpoints. Slender body formulations of this type have been used in numerous applications; our focus is on biological ones such as ciliary motion [11; 12], and swimming flagella [13].

The Keller–Rubinow slender body formulation [16] relies on the exact cancellation of integrals that have the same asymptotic singularity. While this is a mathematically elegant formulation, its numerical implementation is unstable to high wave number perturbations [20; 24]. Generally, it is not possible to achieve the same singularity cancellation numerically without problems related to cancellation errors and instability. Roughly speaking, to overcome this problem and stabilize the computation, the integrands in [20; 24] were regularized by replacing r^{-1} with $(r^2+\delta^2)^{-1/2}$ using a clever choice of δ that preserved the order of the asymptotic expansion of the final formula.

In Lighthill's theory [17; 18; 19], a portion of the filament containing the singularity is removed from the integration and replaced with a local term. The remaining integral is no longer singular but care must be taken in the numerical evaluation of it since the kernel has large gradients near the endpoints of the removed piece. A drawback of this formulation is that removing a piece of the integration curve interrupts the periodicity of the problem for a closed filament, eliminating the benefits of the trapezoid rule in periodic domains or the use of spectral methods to approximate the integral.

We address these issues here by re-deriving both theories for the case of force and dipole fields distributed not as delta distributions along the centerline of the body, but distributed over the cross-section of the slender body. This is accomplished by defining a smooth localized spherically symmetric function $\phi_\delta(r)$ (like a narrow Gaussian with standard deviation proportional to the slender body radius) centered at every point $X(s)$ of the body centerline and letting the force be given by $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{f}(s)\phi_\delta(|\boldsymbol{x}-X(s)|)$. While the maximum of the force is at the centerline of the slender body, the force is distributed over the entire cross-section of the body, which leads to a regular expression for the velocity of the body. This implies that the regularized Lighthill formulation can be implemented without removing the

piece of the curve where the singularity was. The regularization parameter, δ, is also dimensionless after scaling by the tube length, and is assumed to satisfy $\delta \sim a$.

The solution of the Stokes equations for regularized forces and dipoles is derived in Section 2, including the specific regularizing functions that are used. This solution is used to generate the near-field and far-field expansions for the asymptotic solution. Section 3 contains the derivation of Lighthill's theory for regularized forces and shows that the final expression collapses to Lighthill's formula when the regularization parameter δ vanishes. However, for $\delta > 0$, additional simplifications to the final formula are possible that circumvent the drawbacks discussed above. Section 4 shows the matched asymptotic analysis corresponding to Keller and Rubinow's theory, including simplifications to the final expression for $\delta > 0$. In Section 5 we show validation studies and numerical simulations that compare the two theories both in the case of a closed filament as well as the case of a swimming organism.

2. The flow due to regularized forces

The incompressible Stokes equations in $\mathbb{R}^3$ are

$$\mu \Delta \boldsymbol{u} = \nabla p - \boldsymbol{F}, \tag{1}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2}$$

where μ is the fluid viscosity, $\boldsymbol{u}$ is the fluid velocity, p is the pressure, and $\boldsymbol{F}$ is the body force. The boundary conditions associated with the flow are:

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{v}(\sigma) \qquad \text{for } \boldsymbol{x} \text{ on the surface of the slender body at cross section } \sigma,$$

$$\boldsymbol{u}(\boldsymbol{x}) \to 0 \qquad \text{as } |\boldsymbol{x}| \to \infty,$$

where $\boldsymbol{v}(\sigma)$ is a translational velocity of the cross section at σ.

Consider the problem in (1)–(2) in the case when the force exerted by the filament on the fluid is given by the function $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{f}\ \phi_\delta(\boldsymbol{x})$, where ϕ_δ is a radially symmetric smooth function whose integral over $\mathbb{R}^3$ is 1. For example, ϕ_δ may be a normal density function with standard deviation proportional to the parameter δ. The function ϕ_δ provides the spatial dependence of the body force. The following definitions will be convenient for the derivation of the exact solution of these equations.

Definitions. Let the regularized Green's function $G_\delta(\boldsymbol{x})$ be the free-space solution of $\Delta G_\delta = \phi_\delta$ and let $B_\delta(\boldsymbol{x})$ be the free-space solution of $\Delta B_\delta = G_\delta$.

The function $G_\delta(\boldsymbol{x})$ is a smooth function that is bounded everywhere and closely approximates the Green's function $G(\boldsymbol{x}) = -(4\pi|\boldsymbol{x}|)^{-1}$ for $|\boldsymbol{x}| > \delta$. Similarly $B_\delta(\boldsymbol{x})$ is smooth and approximates $B(\boldsymbol{x}) = -|\boldsymbol{x}|/8\pi$, the solution of the equation $\Delta B(\boldsymbol{x}) = G(\boldsymbol{x})$.

Taking the divergence of (1) and using (2) we have that $\nabla \cdot \boldsymbol{F} = \Delta p$, which gives

$$p(\boldsymbol{x}) = \boldsymbol{f} \cdot \nabla G_\delta(\boldsymbol{x}). \tag{3}$$

The equation for $\boldsymbol{u}$ now becomes $\mu \Delta \boldsymbol{u} = (\boldsymbol{f} \cdot \nabla)\nabla G_\delta - \boldsymbol{f}\ \phi_\delta$, whose particular solution is

$$\mu \boldsymbol{u}(\boldsymbol{x}) = (\boldsymbol{f} \cdot \nabla)\nabla B_\delta(\boldsymbol{x}) - \boldsymbol{f}\ G_\delta(\boldsymbol{x}).$$

This is referred to as *regularized stokeslet* flow. To this particular solution one can add $[\boldsymbol{D}\phi_\delta - (\boldsymbol{D} \cdot \nabla)\nabla G_\delta]$, which represents a regularized dipole flow whose divergence is zero everywhere and is harmonic outside the support of ϕ_δ. In this way we obtain the more general solution

$$\mu \boldsymbol{u}(\boldsymbol{x}) = (\boldsymbol{f} \cdot \nabla)\nabla B_\delta(\boldsymbol{x}) - \boldsymbol{f}\ G_\delta(\boldsymbol{x}) + \boldsymbol{D}\phi_\delta - (\boldsymbol{D} \cdot \nabla)\nabla G_\delta + \mu \boldsymbol{U}, \tag{4}$$

where $\boldsymbol{U}$ is a constant flow that may depend on $\boldsymbol{f}$ and $\boldsymbol{D}$.

Using the fact that G_δ and B_δ are radially symmetric, (4) can be written as

$$8\pi\mu(\boldsymbol{u}(\boldsymbol{x}) - \boldsymbol{U}) = \boldsymbol{f}\ \frac{H_1(|\boldsymbol{x}|)}{|\boldsymbol{x}|} + (\boldsymbol{f}\cdot\boldsymbol{x})\boldsymbol{x}\frac{H_2(|\boldsymbol{x}|)}{|\boldsymbol{x}|^3} - 2\boldsymbol{D}\ \frac{H_3(|\boldsymbol{x}|)}{|\boldsymbol{x}|^3} + 6(\boldsymbol{D}\cdot\boldsymbol{x})\boldsymbol{x}\frac{H_4(|\boldsymbol{x}|)}{|\boldsymbol{x}|^5}$$

where the smoothing functions depend on the blob ϕ_δ. They are defined by the relations

$$H_1(r) = 8\pi\big(B'_\delta(r) - rG_\delta(r)\big), \qquad H_2(r) = 8\pi\big(rB''_\delta(r) - B'_\delta(r)\big),$$
$$H_3(r) = 4\pi r^2\big(G'_\delta(r) - r\phi_\delta(r)\big), \qquad H_4(r) = \tfrac{4}{3}\pi r^2\big(G'_\delta(r) - rG''_\delta(r)\big).$$

One can check that for fixed $\delta > 0$ we have

- $\lim_{r\to\infty} H_k(r) = 1$, for $k = 1, 2, 3, 4$;
- for $r \ll \delta$,

$$H_1(r) = O\Big(\frac{r}{\delta}\Big), \quad H_2(r) = O\Big(\Big(\frac{r}{\delta}\Big)^3\Big), \quad H_3(r) = O\Big(\Big(\frac{r}{\delta}\Big)^3\Big), \quad H_4(r) = O\Big(\Big(\frac{r}{\delta}\Big)^5\Big).$$

The velocity formula has been derived for arbitrary isolated force $\boldsymbol{f}$ and dipole strength $\boldsymbol{D}$. In the case of a filament given by $X(\sigma)$, where σ is the arclength parameter with $0 \le \sigma \le 1$, the velocity at any point $\boldsymbol{x}$ is given by

$$8\pi\mu(\boldsymbol{u}(\boldsymbol{x}) - \boldsymbol{U}) = \int_0^1 \boldsymbol{f}(\sigma)\frac{H_1(r)}{r} + (\boldsymbol{f}\cdot\boldsymbol{r})\boldsymbol{r}\frac{H_2(r)}{r^3} - 2\boldsymbol{D}(\sigma)\frac{H_3(r)}{r^3} + 6(\boldsymbol{D}\cdot\boldsymbol{r})\boldsymbol{r}\frac{H_4(r)}{r^5}\,d\sigma, \tag{5}$$

where $\boldsymbol{r} = \boldsymbol{x} - X(\sigma)$ and $r = |\boldsymbol{r}|$. The constant field $\boldsymbol{U}$ and the dipole strength distribution $\boldsymbol{D}(\sigma)$ give the degrees of freedom needed to enforce boundary conditions at $|\boldsymbol{x}| \to \infty$ and at the filament surface.

2.1. ***Choice of blob function.*** Before proceeding we need to choose the regularizing function. Throughout the paper we will use radially symmetric blobs with infinite support. It turns out to be convenient (although not necessary) to choose one regularization for the stokeslets and a different one for the dipoles [1] in order to achieve simplified expressions. Using

$$\phi_\delta(r) = \frac{15\delta^4}{8\pi(r^2+\delta^2)^{7/2}} \quad \text{and} \quad \psi(r) = \frac{3\delta^2}{4\pi(r^2+\delta^2)^{5/2}}, \tag{6}$$

for the stokeslets and dipoles, respectively, the functions in (5) are

$$\begin{aligned} H_1(r) &= r\left(\frac{1}{\sqrt{r^2+\delta^2}} + \frac{\delta^2}{(r^2+\delta^2)^{3/2}}\right), & H_2(r) &= r\left(\frac{1}{\sqrt{r^2+\delta^2}} - \frac{\delta^2}{(r^2+\delta^2)^{3/2}}\right), \\ H_3(r) &= r^3\left(\frac{1}{(r^2+\delta^2)^{3/2}} - \frac{3\delta^2}{(r^2+\delta^2)^{5/2}}\right), & H_4(r) &= \frac{r^5}{(r^2+\delta^2)^{5/2}} \end{aligned} \tag{7}$$

and are shown in Figure 2. Other choices of regularization are possible, including Gaussians and functions with compact support. In all cases, the regularization parameter δ is chosen to be $\delta = O(a)$.

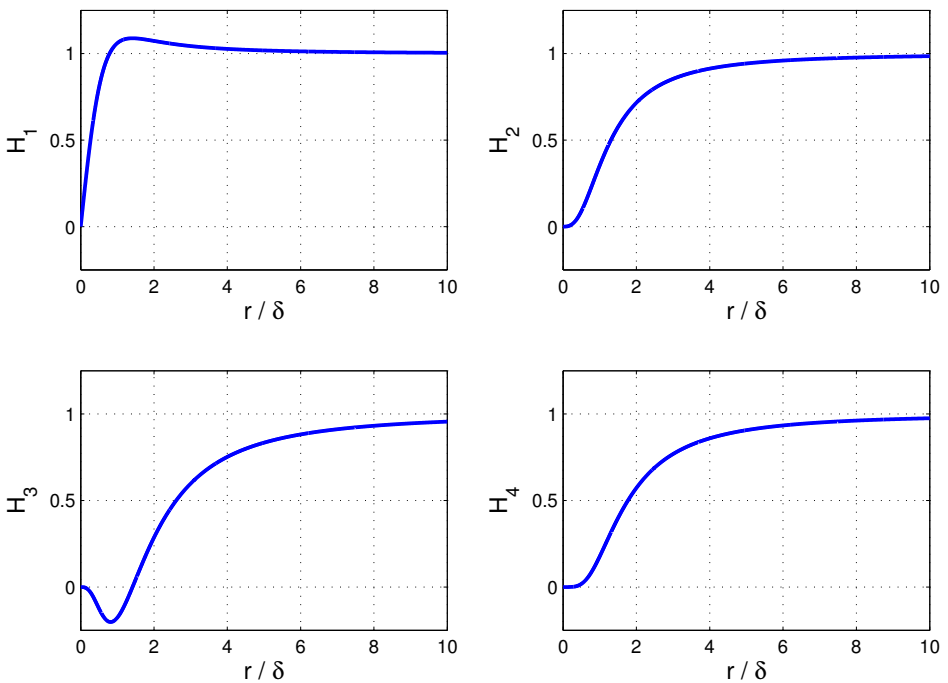


Figure 2. Graphs of the smoothing functions H_1–H_4 as functions of r/δ.

3. Lighthill's theory

We consider first a construction of the slender body velocity following the strategy in [17; 18; 19]. In Lighthill's theory, it is sufficient to choose the constant flow $\boldsymbol{U} = \boldsymbol{0}$ and to consider the evaluation of the velocity at a point $\boldsymbol{x}$ on the surface of the slender body. The velocity is given by (5):

$$8\pi\mu\boldsymbol{u}(\boldsymbol{x}) = \int_0^1 \boldsymbol{f}(\sigma)\frac{H_1(r)}{r} + (\boldsymbol{f}\cdot\boldsymbol{r})\boldsymbol{r}\frac{H_2(r)}{r^3} - 2\boldsymbol{D}(\sigma)\frac{H_3(r)}{r^3} + 6(\boldsymbol{D}\cdot\boldsymbol{r})\boldsymbol{r}\frac{H_4(r)}{r^5}\,d\sigma,$$

where $\boldsymbol{r} = \boldsymbol{x} - X(\sigma)$.

Specifically, consider the integral evaluated on the surface of the cross-section at $\sigma = \sigma_0$ and select an intermediate length scale represented by q with $a \ll q \ll 1$ and make the following assumptions:

(1) q is large enough that for $r > q$, the dipole contribution is negligible due to the high singularity, and the stokeslet contribution does not vary significantly on the cross-section at σ_0.

(2) q is small enough that the portion of the slender body corresponding to $|\sigma - \sigma_0| < q$ is straight (has zero curvature), and $\boldsymbol{f}(\sigma)$ and $\boldsymbol{D}(\sigma)$ do not vary significantly from their values at σ_0.

Although the velocity formula above is evaluated at a point $\boldsymbol{x}$ on the surface of the slender body, the goal is to reduce this formula to one that is evaluated at the centerline point $X(\sigma_0)$, corresponding to the center of the cross-section containing $\boldsymbol{x}$. The result would be an expression involving only centerline points but consistent with the correct boundary conditions on the surface of the slender body. The velocity expression will be separated into two pieces: the near field corresponding to $|\sigma - \sigma_0| < q$ and the far field. The first assumption will be used to simplify the far field and the second one to simplify the near field.

Let $\hat{\boldsymbol{b}}$ be a unit vector normal to the centerline at $X(\sigma_0)$ and write (see Figure 1)

$$\boldsymbol{x} = X(\sigma_0) + a\hat{\boldsymbol{b}}, \quad \boldsymbol{r}_0 = X(\sigma_0) - X(\sigma), \quad \boldsymbol{r} = \boldsymbol{x} - X(\sigma).$$

3.1. ***The far field.*** The far field is expressed as

$$\int_{|\boldsymbol{r}|>q} \boldsymbol{f}(\sigma)\frac{H_1(r)}{r} + (\boldsymbol{f}\cdot\boldsymbol{r})\boldsymbol{r}\frac{H_2(r)}{r^3} - 2\boldsymbol{D}(\sigma)\frac{H_3(r)}{r^3} + 6(\boldsymbol{D}\cdot\boldsymbol{r})\boldsymbol{r}\frac{H_4(r)}{r^5}d\sigma.$$

In the far field, the dipole contribution is insignificant and the stokeslet contribution is independent of the evaluation point on the cross-section centered at $X(\sigma_0)$. Consequently, one choose $\boldsymbol{D} = 0$ and $\boldsymbol{r} = \boldsymbol{r}_0$ in the integral above to get

$$8\pi\mu\boldsymbol{u}_{far}(\sigma_0) = \int_{|\boldsymbol{r}_0|>q} \boldsymbol{f}(\sigma)\frac{H_1(r_0)}{r_0} + (\boldsymbol{f}\cdot\boldsymbol{r}_0)\boldsymbol{r}_0\frac{H_2(r_0)}{r_0^3}d\sigma + O(a). \tag{8}$$

The error term in this equation is due to the fact that

$$(\boldsymbol{f}\cdot\boldsymbol{r})\boldsymbol{r} = (\boldsymbol{f}_0\cdot\boldsymbol{r}_0)\boldsymbol{r}_0 + a[(\boldsymbol{f}_0\cdot\hat{\boldsymbol{b}})\boldsymbol{r}_0 + (\boldsymbol{f}_0\cdot\boldsymbol{r}_0)\hat{\boldsymbol{b}}] + a^2(\boldsymbol{f}_0\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{b}}. \tag{9}$$

and that in the far field of point $X(\sigma_0)$ the shape of the slender body is arbitrary so there is no reason to expect any cancellation from symmetries.

3.2. ***The near field.*** Now, the near-field contribution is given by

$$8\pi\mu\boldsymbol{u}_{\text{near}}(\sigma_0) = \int_{|\boldsymbol{r}|<q} \boldsymbol{f}(\sigma)\frac{H_1(r)}{r} + (\boldsymbol{f}\cdot\boldsymbol{r})\boldsymbol{r}\frac{H_2(r)}{r^3} - 2\boldsymbol{D}(\sigma)\frac{H_3(r)}{r^3} + 6(\boldsymbol{D}\cdot\boldsymbol{r})\boldsymbol{r}\frac{H_4(r)}{r^5}\,d\sigma.$$

We consider q small enough that the force does not vary significantly from $\boldsymbol{f}(\sigma_0)$ and the dipole strength does not vary significantly from $\boldsymbol{D}(\sigma_0)$. Then for a straight filament, the vector $\boldsymbol{r}_0 = (\sigma_0 - \sigma)\boldsymbol{s}$ is an odd function, so that the term proportional to a in (9) will provide no contribution to the integral. We are left with

$$8\pi\mu\boldsymbol{u}_{\text{near}}(\sigma_0) = \int_{|\boldsymbol{r}|<q}\Bigg(\boldsymbol{f}_0\frac{H_1(r)}{r} + \big((\boldsymbol{f}_0\cdot\boldsymbol{r}_0)\boldsymbol{r}_0 + a^2(\boldsymbol{f}_0\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{b}}\big)\frac{H_2(r)}{r^3} - 2\boldsymbol{D}_0\frac{H_3(r)}{r^3} + 6\big((\boldsymbol{D}_0\cdot\boldsymbol{r}_0)\boldsymbol{r}_0 + a^2(\boldsymbol{D}_0\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{b}}\big)\frac{H_4(r)}{r^5}\Bigg)\,d\sigma.$$

In order for the entire cross-section of the filament at σ_0 to move with the same velocity, the integral cannot depend on the vector $\hat{\boldsymbol{b}}$, which is a unit vector normal to the filament at σ_0 but otherwise arbitrary. We therefore enforce the condition that

$$\int_{|\boldsymbol{r}|<q} (\boldsymbol{f}_0\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{b}}\,\frac{H_2(r)}{r^3} + 6(\boldsymbol{D}_0\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{b}}\,\frac{H_4(r)}{r^5}d\sigma = 0,$$

which is used to determine the strength of the dipole $\boldsymbol{D}_0$ as a function of $\boldsymbol{f}_0$. This approach is exactly analogous to the problem of a sphere moving at constant velocity in a Stokes fluid. A single stokeslet at the center of the sphere is not sufficient to provide the correct velocity, but a stokeslet plus a dipole at the center will suffice, provided the dipole strength is related to the stokeslet strength in a way that cancels the dependence on the evaluation point on the surface [2].

Computing the last integral exactly and neglecting terms containing $O(a^2/q^2)$ and $O(\delta^2/q^2)$, we have that

$$\frac{2}{(a^2+\delta^2)}(\boldsymbol{f}_0\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{b}} + \frac{8}{(a^2+\delta^2)^2}(\boldsymbol{D}_0\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{b}} = 0$$

from which we deduce that

$$\boldsymbol{D}_0 = -\frac{a^2+\delta^2}{4}\,\boldsymbol{f}_n, \tag{10}$$

where $\boldsymbol{f}_n := \boldsymbol{f}_0 - (\boldsymbol{f}_0 \cdot \boldsymbol{s})\boldsymbol{s}$represents the component of $\boldsymbol{f}_0$ normal to the filament. Note that the dipole strength is strictly normal to the filament; it does not have a tangential component.

The near-field velocity is thus given by

$$8\pi\mu \boldsymbol{u}_{\text{near}}(\sigma_0) = \int_{|r|<q} \boldsymbol{f}_0 \frac{H_1(r)}{r} + (\boldsymbol{f}_0 \cdot \boldsymbol{r}_0)\boldsymbol{r}_0 \frac{H_2(r)}{r^3} - 2\boldsymbol{D}_0 \frac{H_3(r)}{r^3} + 6(\boldsymbol{D}_0 \cdot \boldsymbol{r}_0)\boldsymbol{r}_0 \frac{H_4(r)}{r^5}\, d\sigma.$$

Before proceeding, we decompose the force into its normal and tangential components, $\boldsymbol{f}_0 = \boldsymbol{f}_n + \boldsymbol{f}_\tau$, and note that to leading order, $\boldsymbol{r}_0 = (\sigma_0 - \sigma)\boldsymbol{s}$, so that $(\boldsymbol{f}_0 \cdot \boldsymbol{r}_0)\boldsymbol{r}_0 = (\sigma - \sigma_0)^2 \boldsymbol{f}_\tau$. Using (10), we get

$$8\pi\mu \boldsymbol{u}_{\text{near}}(\sigma_0) = 2(\boldsymbol{f}_n + 2\boldsymbol{f}_\tau)\left[\ln\frac{2q}{\beta} - \frac{a^2}{2\beta^2}\right] + 2\boldsymbol{f}_n\left[1 - \frac{\delta^2}{2\beta^2}\right] + O(\epsilon^2), \quad (11)$$

where we have defined $\beta^2 = a^2 + \delta^2$ and $\epsilon = \max(a/q, \delta/q)$.

At this point, one can combine (8) and (11) to get the velocity. However, there is something unattractive about these expressions: they depend on a choice of q. But aside from some scaling requirements, q is arbitrary.

Lighthill devised a way to eliminate this ambiguity in a way that can be adjusted to the present context. Since the far field is simply the integral of the stokeslet field, we compute to leading order for any number θ satisfying $0 < \theta < q$,

$$\int_{\theta<r_0<q} \boldsymbol{f}_0 \frac{H_1(r_0)}{r_0} + (\boldsymbol{f}_0 \cdot \boldsymbol{r}_0)\boldsymbol{r}_0 \frac{H_2(r_0)}{r_0^3}\, d\sigma = 2(\boldsymbol{f}_n + 2\boldsymbol{f}_\tau)\ln\frac{2q}{\theta + \sqrt{\theta^2+\delta^2}} + 2\boldsymbol{f}_n\left[1 - \frac{\theta}{\sqrt{\theta^2+\delta^2}}\right]$$

and since (11) can be written as

$$8\pi\mu \boldsymbol{u}_{\text{near}}(\sigma_0) = 2(\boldsymbol{f}_n + 2\boldsymbol{f}_\tau)\ln\frac{2q}{\beta e^{a^2/2\beta^2}} + 2\boldsymbol{f}_n\left[1 - \frac{\delta^2}{2\beta^2}\right],$$

we can define the number θ by making the identification

$$\theta + \sqrt{\theta^2+\delta^2} = \beta e^{a^2/2\beta^2},$$

so that

$$8\pi\mu \boldsymbol{u}_{\text{near}}(\sigma_0) = 2\boldsymbol{f}_n\left[\frac{\theta}{\sqrt{\theta^2+\delta^2}} - \frac{\delta^2}{2\beta^2}\right] + \int_{\theta<r_0<q} \boldsymbol{f}_0 \frac{H_1(r_0)}{r_0} + (\boldsymbol{f}_0 \cdot \boldsymbol{r}_0)\boldsymbol{r}_0 \frac{H_2(r_0)}{r_0^3}\, d\sigma.$$

By writing the near field in this way and adding it to the far field in (8), we get a final expression which is independent of q:

$$8\pi\mu\boldsymbol{u}(\sigma_0) = 2\boldsymbol{f}_n\left[1 - \frac{2\delta^2}{\beta^2 e^{a^2/\beta^2} + \delta^2} - \frac{\delta^2}{2\beta^2}\right] + \int_{\theta<r_0} \boldsymbol{f}(\sigma)\frac{H_1(r_0)}{r_0} + (\boldsymbol{f}\cdot\boldsymbol{r}_0)\boldsymbol{r}_0\frac{H_2(r_0)}{r_0^3}\,d\sigma, \quad (12)$$

where $\boldsymbol{r}_0 = X(\sigma_0) - X(\sigma)$, $r_0 = |\boldsymbol{r}_0|$, $\beta^2 = a^2 + \delta^2$, and

$$\theta = \frac{1}{2\beta}\left(\beta^2 e^{a^2/2\beta^2} - \delta^2 e^{-a^2/2\beta^2}\right).$$

Formula (12) indicates that the velocity of the filament at $X(\sigma_0)$ has two contributions. One is the integral of the regularized stokeslet field evaluated at the centerline, with the portion $|\sigma - \sigma_0| < \theta$ excluded. The other one is a local term proportional to the component of force normal to the filament at σ_0.

3.3. ***The limit*** $\delta \to 0$**.** Notice that as the regularization δ vanishes, we have that $\beta \to a$ and therefore $\theta \to \theta_0 = a\sqrt{e}/2 \approx 0.824a$. In this limit, $H_1(r) \to 1$, $H_2(r) \to 1$ and the local term reduces to $2\boldsymbol{f}_n$, so that the entire expression for velocity is in agreement with Lighthill's basic theorem of flagellar hydrodynamics [18; 19]. The expression derived here is therefore more general since it includes the previously developed case of delta force distributions and extends it to the case of regular forces distributed over the body's cross-section. Figure 3 shows the relative size of the bracketed expression in the local term of (12) and of the parameter θ as functions of the regularization parameter δ.

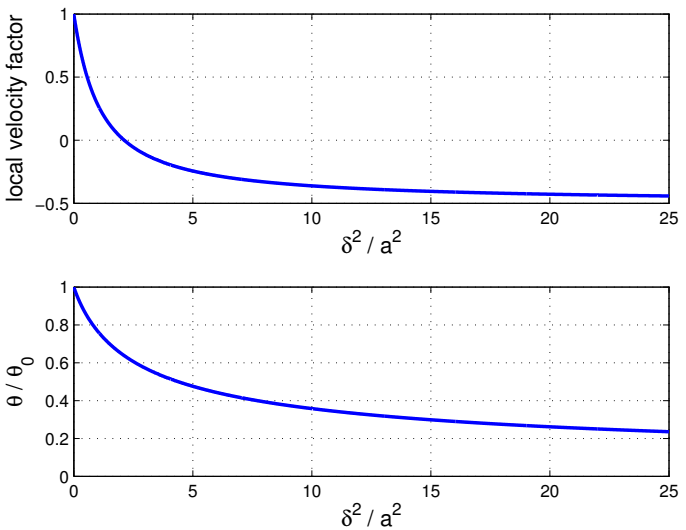


Figure 3. Graphs of the bracketed factor in the local term of the velocity in (12) and $\theta/\theta_0 = 2\theta/a\sqrt{e}$ as functions of δ^2/a^2.

3.4. ***Simplifications for $\delta > 0$.*** In the case $\delta = 0$, the excluded part of the integral in (12) is necessary because the stokeslet field is not integrable in any neighborhood of σ_0. For a closed filament, spectral methods are efficient numerical techniques for computing the integral accurately; however, the missing piece of the integral gets in the way of an efficient implementation of such methods. However, in the regularized case (when $\delta > 0$), the velocity expression is integrable and one can write it as the integral from $\sigma = 0$ to $\sigma = 1$ by simply adding and subtracting the excluded piece. By assumption in the region $|\sigma - \sigma_0| < \theta$, the force is considered constant: $\boldsymbol{f}(\sigma) = \boldsymbol{f}_0$ and $(\boldsymbol{f}\cdot\boldsymbol{r}_0)\boldsymbol{r}_0 = r_0^2\ \boldsymbol{f}_\tau$. After some simplification we have

$$8\pi\mu\boldsymbol{u}(\sigma_0) = 2\boldsymbol{f}_n\left[1 - \frac{\delta^2}{2\beta^2}\right] - (\boldsymbol{f}_n + 2\boldsymbol{f}_\tau)\left[2\ln(\beta/\delta) + 1 - \frac{\delta^2}{\beta^2}\right] + \int_0^1 \frac{\boldsymbol{f}(\sigma)}{\sqrt{r_0^2+\delta^2}} + \frac{(\boldsymbol{f}\cdot\boldsymbol{r}_0)\boldsymbol{r}_0}{r_0^2\sqrt{r_0^2+\delta^2}}\,d\sigma, \quad (13)$$

where $\boldsymbol{r}_0 = X(\sigma_0) - X(\sigma)$, $r_0 = |\boldsymbol{r}_0|$, and $\beta^2 = a^2 + \delta^2$.

This formula shows that the integral is that of the standard stokeslet but with the singular factor $1/r$ replaced by $1/\sqrt{r^2+\delta^2}$. The regularization of the forces is also expected to impact the local terms in the velocity since it is in a neighborhood of the centerline where the forces are most significantly changed. The final formula shows the appropriate form of the local terms in order for the velocity to be consistent with these forces.

4. Keller–Rubinow theory

Lighthill's slender body theory is developed in such a way that the errors depend linearly on the body radius. Keller and Rubinow's theory [16] develops the relationship between velocity and force with errors $O(a^2 \ln a)$ in regions away from the endpoints. In order to achieve this improvement, the near-field and far-field flows are evaluated at fluid locations rather than at a point on the slender body surface. The two expressions are then matched asymptotically at an intermediate distance. In this section, it will be convenient to start with (5) and set the dipole strength equal to a multiple of the force, $\boldsymbol{D} = A\boldsymbol{f}$.

4.1. ***The far field.*** Let $\boldsymbol{x}$ be a point in the fluid far from the slender body. For the far field solution we consider a flow which decays to zero as $|\boldsymbol{x}| \to \infty$. In this case it is possible to choose $A = 0$ and $\boldsymbol{U} = \boldsymbol{0}$ to get

$$8\pi\mu\boldsymbol{u}(\boldsymbol{x}) = \int_0^1 \boldsymbol{f}(\sigma)\,\frac{H_1(r)}{r} + (\boldsymbol{f}(\sigma)\cdot\boldsymbol{r})\boldsymbol{r}\,\frac{H_2(r)}{r^3}\,d\sigma \quad (14)$$

where $\boldsymbol{r} = \boldsymbol{x} - X(\sigma)$ and $r = |\boldsymbol{r}|$.

4.2. ***The near field.*** We now consider a point $\boldsymbol{x}_0$ in the fluid so close to the slender body that the latter can be viewed as a long, thin, straight cylinder of radius a. Without loss of generality we assume that it extends in the $\hat{\boldsymbol{z}}$ direction from $z=-q$ to $z=q$ and that the evaluation point is given by

$$\boldsymbol{x}_0 = (x, y, 0),$$

with $|\boldsymbol{x}_0| = \rho \ll q$. The forces acting along the straight tube are assumed to be constant and equal to $\boldsymbol{f}_0$. Then the velocity at $\boldsymbol{x}_0$ satisfies

$$8\pi\mu(\boldsymbol{u}(\boldsymbol{x}_0) - \boldsymbol{U}) = \int_{-q}^{q} \boldsymbol{f}_0\left(\frac{H_1(|\boldsymbol{y}|)}{|\boldsymbol{y}|} - \frac{2AH_3(|\boldsymbol{y}|)}{|\boldsymbol{y}|^3}\right) + (\boldsymbol{f}_0\cdot\boldsymbol{y})\boldsymbol{y}\left(\frac{H_2(|\boldsymbol{y}|)}{|\boldsymbol{y}|^3} + \frac{6AH_4(|\boldsymbol{y}|)}{|\boldsymbol{y}|^5}\right) dz, \quad (15)$$

where

$$\boldsymbol{y} = \boldsymbol{x}_0 - (0, 0, z) = (x, y, -z).$$

In order to simplify the notation, let $\boldsymbol{s}$ be a unit vector in the positive $\hat{\boldsymbol{z}}$ direction (tangent to the filament). Then

$$(\boldsymbol{f}_0\cdot\boldsymbol{y})\boldsymbol{y} = (\boldsymbol{f}_0\cdot\boldsymbol{x}_0)\,\boldsymbol{x}_0 - z\,[(\boldsymbol{f}_0\cdot\boldsymbol{x}_0)\,\boldsymbol{s} + (\boldsymbol{f}_0\cdot\boldsymbol{s})\,\boldsymbol{x}_0] + z^2(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}.$$

Using the specific form of the functions $H_1(r)$–$H_4(r)$ in (7), we find that the inner velocity is given by (see Appendix A):

$$\begin{aligned} 8\pi\mu(\boldsymbol{u}(\boldsymbol{x}_0) - \boldsymbol{U}) = \boldsymbol{f}&\left[\ln\frac{4q^2}{|\boldsymbol{x}_0|^2+\delta^2} + \frac{2\delta^2}{|\boldsymbol{x}_0|^2+\delta^2} - 2A\left(\frac{2|\boldsymbol{x}_0|^2-\delta^2}{(|\boldsymbol{x}_0|^2+\delta^2)^2} - \frac{1}{q^2}\right)\right] \\ &+ (\boldsymbol{f}\cdot\boldsymbol{x}_0)\boldsymbol{x}_0\left[\frac{2}{|\boldsymbol{x}_0|^2+\delta^2} + 6A\frac{4}{3(|\boldsymbol{x}_0|^2+\delta^2)^2}\right] \\ &+ (\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}\left[\ln\frac{4q^2}{|\boldsymbol{x}_0|^2+\delta^2} - 2 + 6A\left(\frac{2}{3(|\boldsymbol{x}_0|^2+\delta^2)} - \frac{1}{q^2}\right)\right]. \end{aligned}$$

4.2.1. *No-slip boundary condition.* We consider the slender body velocity given by the unknown $\boldsymbol{v}(\sigma_0)$. The goal is to develop an expression for $\boldsymbol{v}(\sigma_0)$. The near-field boundary condition must be consistent with a uniform velocity at every point of a cross-section of the slender body. Then we must impose the condition $\boldsymbol{u}(\boldsymbol{x}_0) = \boldsymbol{v}(\sigma_0)$ for all $\boldsymbol{x}_0$ with magnitude $\rho = a$ (i.e. on the surface of the slender body). For this we notice that the second term on the right side of the last equation is the only one that is not radially symmetric. This leads us to choose

$$A = -\frac{a^2+\delta^2}{4}.$$

With this value of A we can solve for the constant flow $\boldsymbol{U}$ in terms of $\boldsymbol{v}(\sigma_0)$ and substitute it back. The final inner velocity expression at a point $\boldsymbol{x}_0$ in the fluid is

$$
\begin{aligned}
&8\pi\mu\boldsymbol{u}(\boldsymbol{x}_0)\\
&= 8\pi\mu\boldsymbol{v}(\sigma_0) - (\boldsymbol{f}_0 + (\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\left(\ln\frac{|\boldsymbol{x}_0|^2+\delta^2}{a^2+\delta^2} + \frac{(|\boldsymbol{x}_0|^2-\delta^2)(|\boldsymbol{x}_0|^2-a^2)}{(|\boldsymbol{x}_0|^2+\delta^2)^2}\right)\\
&\qquad + 2(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}\frac{|\boldsymbol{x}_0|^2(|\boldsymbol{x}_0|^2-a^2)}{(|\boldsymbol{x}_0|^2+\delta^2)^2} + 2(\boldsymbol{f}_0\cdot\boldsymbol{x}_0)\boldsymbol{x}_0\frac{|\boldsymbol{x}_0|^2-a^2}{(|\boldsymbol{x}_0|^2+\delta^2)^2}. \qquad (16)
\end{aligned}
$$

4.3. ***Matching.*** The fluid velocity obtained from the inner expansion (16) is bounded as the filament is approached, i.e., as $|\boldsymbol{x}_0|\to a$. However, it grows logarithmically as $|\boldsymbol{x}_0|$ increases. The outer expansion velocity given by (14) also has a logarithmic term as $\boldsymbol{x}$ approaches a point $X(\sigma_0)$ on the filament. To match the solutions at an intermediate distance, let $\boldsymbol{x}_0 = \boldsymbol{x} - X(\sigma_0)$ with $|\boldsymbol{x}_0| = \rho$ and assume $\delta \sim a \ll \rho \ll 1$. Here $X(\sigma_0)$ is the point on the filament which is closest to $\boldsymbol{x}$ and $\boldsymbol{s}$ is the unit tangent at $X(\sigma_0)$. Then dropping the higher order terms in $O(a^2/\rho^2)$ and $O(\delta^2/\rho^2)$, the inner expansion becomes

$$
\begin{aligned}
&8\pi\mu\boldsymbol{u}(\boldsymbol{x})\\
&\quad = 8\pi\mu\boldsymbol{v}(\sigma_0) - (\boldsymbol{f}_0 + (\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\left[\ln\frac{\rho^2+\delta^2}{a^2+\delta^2} + 1\right] + 2(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s} + \frac{2(\boldsymbol{f}_0\cdot\boldsymbol{x}_0)\boldsymbol{x}_0}{\rho^2}.
\end{aligned}
$$

The outer expansion velocity is given by (14), which we rewrite as

$$
8\pi\mu\boldsymbol{u}(\boldsymbol{x}) = \int_0^1 J(\boldsymbol{r},\rho,\delta,\boldsymbol{f}(\sigma))\,d\sigma, \qquad (17)
$$

where $\boldsymbol{r} = \boldsymbol{x} - X(\sigma)$. Setting the two expressions equal to each other we get

$$
\begin{aligned}
&\int_0^1 J(\boldsymbol{r},\rho,\delta,\boldsymbol{f}(\sigma))\,d\sigma\\
&\quad = 8\pi\mu\boldsymbol{v}(\sigma_0) - (\boldsymbol{f}_0 + (\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\left[\ln\frac{\rho^2+\delta^2}{a^2+\delta^2} + 1\right] + 2(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s} + \frac{2(\boldsymbol{f}_0\cdot\boldsymbol{x}_0)\boldsymbol{x}_0}{\rho^2}.
\end{aligned}
$$

The inner expansion of the far field is found by expanding the left-hand side of this equation in powers of ρ (see Appendix B). This yields

$$
\begin{aligned}
&\int_0^1\left(J(\boldsymbol{r}_0,0,\delta,\boldsymbol{f}(\sigma)) - \boldsymbol{f}_0\frac{H_1(|\sigma-\sigma_0|)}{|\sigma-\sigma_0|} - (\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}\frac{H_2(|\sigma-\sigma_0|)}{|\sigma-\sigma_0|}\right)d\sigma \qquad (18)\\
&\quad + (\boldsymbol{f}_0 + (\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\big[\ln(4\sigma_0(1-\sigma_0)) - \ln(\rho^2+\delta^2)\big] - 2(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s} + \frac{2(\boldsymbol{f}_0\cdot\boldsymbol{x}_0)\boldsymbol{x}_0}{\rho^2}\\
&\quad = 8\pi\mu\boldsymbol{v}(\sigma_0) - (\boldsymbol{f}_0 + (\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\left[\ln\frac{\rho^2+\delta^2}{a^2+\delta^2} + 1\right] + 2(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s} + \frac{2(\boldsymbol{f}_0\cdot\boldsymbol{x}_0)\boldsymbol{x}_0}{\rho^2},
\end{aligned}
$$

where $\boldsymbol{r}_0 = X(\sigma_0) - X(\sigma)$.

Since $\boldsymbol{x}_0$ is an arbitrary point where the asymptotic matching is done, the final result should not depend on it (or on its magnitude ρ). Fortunately all the terms

containing ρ cancel out of the last equation so that the change of variables $t=\sigma-\sigma_0$ gives

$$8\pi\mu\boldsymbol{v}(\sigma_0)=\int_{-\sigma_0}^{1-\sigma_0}\Bigg(\frac{\boldsymbol{f}(\sigma_0+t)\ H_1(r_0)}{r_0}-\frac{\boldsymbol{f}_0\ H_1(|t|)}{|t|} + \frac{(\boldsymbol{f}(\sigma_0+t)\cdot\boldsymbol{r}_0)\boldsymbol{r}_0\ H_2(r_0)}{r_0^3}-\frac{(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}\ H_2(|t|)}{|t|}\Bigg)dt + (\boldsymbol{f}_0+(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\big[\ln(4\sigma_0(1-\sigma_0))-\ln\left(a^2+\delta^2\right)+1\big]-4(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}. \quad (19)$$

Our velocity formula can be simplified using the functions H_1–H_4 in (7) (see Appendix C) so that up to $O(\delta^2\ln\delta)$ the final expression for the filament velocity becomes

$$8\pi\mu\boldsymbol{v}(\sigma_0)=\int_{-\sigma_0}^{1-\sigma_0}\frac{\boldsymbol{f}(\sigma_0+t)}{\sqrt{r_0^2+\delta^2}}+\frac{(\boldsymbol{f}(\sigma_0+t)\cdot\boldsymbol{r}_0)\boldsymbol{r}_0}{r_0^2\sqrt{r_0^2+\delta^2}}-\frac{\boldsymbol{f}_0+(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}}{\sqrt{t^2+\delta^2}}\,dt + (\boldsymbol{f}_0+(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\big[\ln(4\sigma_0(1-\sigma_0))-\ln\left(a^2+\delta^2\right)\big] - (\boldsymbol{f}_0+(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})+2(\boldsymbol{f}_0-(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}), \quad (20)$$

where $\boldsymbol{r}_0=X(\sigma_0)-X(t)$. This is the regularized Keller–Rubinow formula.

4.4. ***The limit $\delta\to 0$.*** The first integral in (20) can be evaluated even when $\delta=0$ because its singular behavior has been explicitly extracted. It is easy to see that in the limit $\delta\to 0$, the expression in (20) converges to the one obtained in [16]:

$$8\pi\mu\boldsymbol{v}(\sigma_0)=\int_{-\sigma_0}^{1-\sigma_0}\frac{\boldsymbol{f}(\sigma_0+t)}{r_0}+\frac{(\boldsymbol{f}(\sigma_0+t)\cdot\boldsymbol{r}_0)\boldsymbol{r}_0}{r_0^3}-\frac{\boldsymbol{f}_0+(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}}{|t|}\,dt + (\boldsymbol{f}_0+(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\big[\ln(4\sigma_0(1-\sigma_0))-\ln a^2\big] - (\boldsymbol{f}_0+(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})+2(\boldsymbol{f}_0-(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}).$$

4.5. ***Simplifications for $\delta>0$.*** We note that since none of the functions in (20) is singular, one can evaluate the third term of the integral explicitly and include the result as part of the local terms, leaving only the integral of the stokeslet kernel as in the case of Lighthill's theory, (13). This also provides a way of comparing the two theories directly. The result is

$$8\pi\mu\boldsymbol{v}(\sigma_0) = 2\boldsymbol{f}_n-(\boldsymbol{f}_n+2\boldsymbol{f}_\tau)\,[2\ln(\beta/\delta)+1]+\int_{-\sigma_0}^{1-\sigma_0}\frac{\boldsymbol{f}(\sigma_0+t)}{\sqrt{r_0^2+\delta^2}}+\frac{(\boldsymbol{f}(\sigma_0+t)\cdot\boldsymbol{r}_0)\boldsymbol{r}_0}{r_0^2\sqrt{r_0^2+\delta^2}}dt. \quad (21)$$

This expression, found by the method of matched asymptotics, can be compared with (13), which was found by different means. The differences appear only in the local terms. We note that this final expression does not rely on the cancellation of singular terms.

In computations, although (21) does not contain singularities, the function is nearly singular (or spiky) so that there is a computational advantage to using (20) instead of (21).

4.6. ***The velocity of the fluid.*** The velocity field at an arbitrary point $\boldsymbol{x}$ in the fluid can also be evaluated using the asymptotic matching. The fluid velocity is given by the sum of the inner solution, (16), and the outer solution, (14), minus the inner expansion of the outer solution, given by the left-hand side of (18). After some cancellation, the final expression is

$$8\pi\mu\boldsymbol{u}(\boldsymbol{x}) = \int_0^1 \boldsymbol{f}(t)\,\frac{H_1(r)}{r} + (\boldsymbol{f}(t)\cdot\boldsymbol{r})\boldsymbol{r}\,\frac{H_2(r)}{r^3}\,dt \tag{22}$$
$$+\boldsymbol{f}(s)\left[1-\frac{(|\boldsymbol{x}_0|^2-\delta^2)(|\boldsymbol{x}_0|^2-a^2)}{(|\boldsymbol{x}_0|^2+\delta^2)^2}\right]-(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}\,\frac{a^2+\delta^2}{|\boldsymbol{x}_0|^2+\delta^2}-2(\boldsymbol{f}\cdot\boldsymbol{x}_0)\boldsymbol{x}_0\,\frac{a^2+\delta^2}{(|\boldsymbol{x}_0|^2+\delta^2)^2},$$

where $\boldsymbol{r}=\boldsymbol{x}-X(s+t)$, $r=|\boldsymbol{r}|$, and $\boldsymbol{x}_0=\boldsymbol{x}-X(s)$. Here $X(s)$ is the filament point closest to $\boldsymbol{x}$.

4.7. ***Periodic filaments.*** Equation (20) is valid for points along the filament that are far from the endpoints $s=0$ and $s=1$ relative to δ. In the case of a periodic filament, the choice of parametrization should be irrelevant. Therefore the equation can be evaluated at a valid point, say $s=1/2$, and the result should be valid for any point on the filament. The forces $\boldsymbol{f}(t)$ and parametrization $X(t)$ are periodic functions but one last modification is necessary because the function $1/\sqrt{t^2+\delta^2}$, which appears in the integrand of (20), is not periodic in t. In order to replace it with a periodic function, we use the identity

$$\int_{-1/2}^{1/2}\frac{1}{\sqrt{t^2+\delta^2}}\,dt=\int_{-1/2}^{1/2}\frac{dt}{\sqrt{(1/\pi^2)\sin^2(\pi t)+\delta^2}}-\ln(16/\pi^2)+O(\delta^2\ln\delta).$$

So, for a periodic filament, the final expression is

$$8\pi\mu\boldsymbol{v}(s)=\int_{-1/2}^{1/2}\frac{\boldsymbol{f}(s+t)}{(|\boldsymbol{r}_0|^2+\delta^2)^{1/2}}+\frac{(\boldsymbol{f}(s+t)\cdot\boldsymbol{r}_0)\boldsymbol{r}_0}{|\boldsymbol{r}_0|^2\,(|\boldsymbol{r}_0|^2+\delta^2)^{1/2}}-\frac{[\boldsymbol{f}(s)+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}]}{\sqrt{\pi^{-2}\sin^2(\pi t)+\delta^2}}\,dt$$
$$+2(\boldsymbol{f}-(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s})-(\boldsymbol{f}+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s})\left[\ln\frac{(a^2+\delta^2)\pi^2}{16}+1\right]. \tag{23}$$

We use this formulation, rather than (20), for periodic filaments.

We note that in this case, the local terms (outside the integral in (23)) can also be written as

$$2(\boldsymbol{f}-(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s})-(\boldsymbol{f}+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s})\left[\ln(a^2)+1+\ln\frac{\pi^2(a^2+\delta^2)}{16a^2}\right].$$

It is clear that the choice of δ affects significantly the local drag. This is to be expected, since the regularization mostly affects the near-field velocity. However, it turns out that the local terms are exactly equal to the ones in [16], i.e.,

$$2(\boldsymbol{f} - (\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}) - (\boldsymbol{f} + (\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s})(\ln a^2 + 1),$$

if δ is chosen so that $\ln \dfrac{\pi^2(a^2+\delta^2)}{16a^2} = 0$, or approximately $\delta = 0.788124a$.

5. Numerical examples

5.1. ***Validation studies.*** We consider first two validation studies by computing the velocity of the fluid around a translating slender body. The first test problem is that of a torus with centerline radius $R = 1/2\pi$ and cross-sectional radius a. The torus is slender when $\epsilon = a/R = 2\pi a \ll 1$. We consider here a torus translating in an arbitrary direction. The second validation study is that of a straight slender body of length 1 and radius a.

5.1.1. *A slender torus.* Consider a torus whose centerline is in the xy-plane. This is a particularly good test problem since the geometry of the slender body is that of a cylindrical tube with constant cross-section, which is exactly what our formulas have assumed. There is no exact solution; however, by placing various fundamental solutions along the centerline, Johnson and Wu [14] developed $O(\epsilon^2 \log \epsilon)$ asymptotic approximations to fluid velocities under various conditions. They give asymptotic formulas for the force per unit length on the centerline of the torus that produces a given translation. We make two comparisons. First, we compute forces by setting the centerline velocity to the prescribed value and using either (13) for Lighthill or (21) for Keller–Rubinow to solve for the forces. We also compute the centerline velocity when using the forces from [14] in the regularized theories. Second, we compute the fluid velocity using the regularized Keller–Rubinow formula (22) and compare with the results from [14].

In the first comparison, we set the centerline velocity to a constant and invert a trapezoid rule discretization of (13) to solve for the forces in Lighthill's theory. We do the same with (21) to solve for the forces in Keller–Rubinow's theory. The force per unit length is then compared with its asymptotic value given in [14]. The results for a horizontal translation velocity $(0, 1, 0)$ are shown in Table 1 and for a vertical translation velocity $(0, 0, 1)$ in Table 2. Recall that δ is a numerical parameter for the regularization of the integrals, and therefore should be related to the discretization size of the centerline. In these examples we discretize the centerline with N points in such a way that the point separation is approximately equal to the tube cross-section a; then δ is chosen proportional to $1/N$. The tables show results for two different values of δ for each slenderness value. The tables show that both theories give comparable results for the normal component of the

a	F_n^{JW}	F_n^{KR}	F_n^L	F_t^{JW}	F_t^{KR}	F_t^L	numerical parameters
0.01	2.2370	2.2539	2.2814	1.4213	1.4415	1.3822	$N = 100, \delta = 0.5/N$
		2.2177	2.2351		1.4008	1.3624	$N = 100, \delta = 0.4/N$
0.005	2.0149	2.0300	2.0453	1.2058	1.2191	1.1808	$N = 200, \delta = 0.5/N$
		1.9982	2.0080		1.1916	1.1665	$N = 200, \delta = 0.4/N$
0.0025	1.8245	1.8373	1.8468	1.0533	1.0630	1.0360	$N = 400, \delta = 0.5/N$
		1.8104	1.8166		1.0428	1.0249	$N = 400, \delta = 0.4/N$
0.00125	1.6636	1.6745	1.6809	0.93775	0.94528	0.92500	$N = 800, \delta = 0.5/N$
		1.6517	1.6559		0.92956	0.91605	$N = 800, \delta = 0.4/N$

Table 1. Comparison of the resultant force per unit length for the case of a torus of centerline radius $R = 1/2\pi$ and cross-sectional radius a translating horizontally (in the y-direction) with unit speed. The slenderness parameter defined in [14] is $\epsilon = 2\pi a$. The forces shown are for the normal and tangential components (subscripts) from [14] (JW), regularized Keller–Rubinow (KR), and regularized Lighthill (L). The number of points N discretizing the centerline is varied so that the point separation equals a. The Lighthill forces were found by inverting (13) while the Keller–Rubinow forces were found by inverting (21).

force. The Keller–Rubinow theory gives slightly better results for the tangential force and the given discretization parameters. By comparing (13) and (21), one can see that the dependence of the normal component of force on the velocity is identical for both theories, while there is a difference in the tangential component

a	F_n^{JW}	F_n^{KR}	F_n^L	numerical parameters
0.01	2.3503	2.3729	2.3729	$N = 100, \delta = 0.5/N$
		2.3259	2.3259	$N = 100, \delta = 0.4/N$
0.005	2.0806	2.0982	2.0982	$N = 200, \delta = 0.5/N$
		2.0614	2.0614	$N = 200, \delta = 0.4/N$
0.0025	1.8664	1.8805	1.8805	$N = 400, \delta = 0.5/N$
		1.8509	1.8509	$N = 400, \delta = 0.4/N$
0.00125	1.6922	1.7038	1.7038	$N = 800, \delta = 0.5/N$
		1.6795	1.6795	$N = 800, \delta = 0.4/N$

Table 2. Comparison of the resultant force per unit length for the case of a torus of centerline radius $R = 1/2\pi$ and cross-sectional radius a translating vertically (in the z-direction) with unit speed. The slenderness parameter defined in [14] is $\epsilon = 2\pi a$. The forces shown are for normal to the tube in the z-direction from [14] (JW), regularized Keller–Rubinow (KR), and regularized Lighthill (L). The number of points N discretizing the centerline is varied so that the point separation equals a. The Lighthill forces were found by inverting (13) while the Keller–Rubinow forces were found by inverting (21). Notice that these two equations are identical for the normal component of force since they only differ in the tangential force component. This example has zero tangential force so the two theories give the same answers.

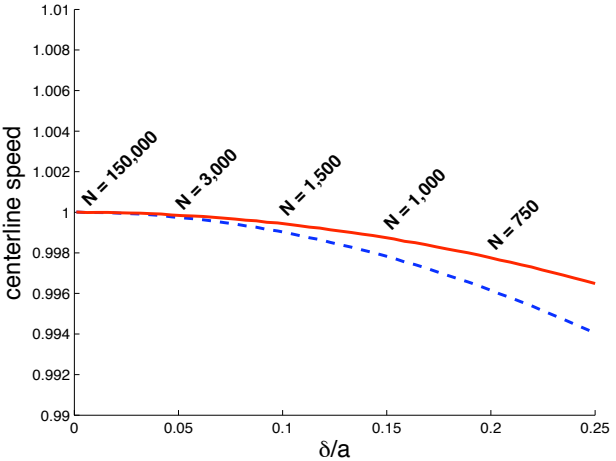


Figure 4. Centerline velocity of the torus as a function of δ when the force per unit length applied is the one given in [14] for a torus moving horizontally with constant velocity $(u, v, w) = (0, 1, 0)$. The solid curve corresponds to the Lighthill theory and the dashed curve to the Keller–Rubinow theory. N is the number of quadrature points along the centerline that were required to numerically resolve the integrals for the given value of δ.

of force. The test problem in Table 2 results in strictly normal force, which is why both theories give the same solution.

A different comparison was performed by using the values of the force per unit length given in [14] and applying those forces in the regularized theories. Although one expects the forces from each theory to be similar for a given centerline velocity boundary condition, they are not identical. So, using the asymptotic forces from [14] in the regularized theories does not guarantee the correct centerline velocity. We set $a = 0.01$ and use the asymptotic forces to compute the centerline velocities from (12) and (21) for different values of δ. The results for the torus translating horizontally in the y direction are shown in Figure 4. The solid curve corresponds to the Lighthill theory and the dashed curve to the Keller–Rubinow theory. For $a = 0.01$, the asymptotic error in [14] is $O(\epsilon^2 \log \epsilon) \approx 0.011$ where $\epsilon = 2\pi a$, which indicates that the velocity values are acceptable for $\delta \in (0, a/4]$. The figure also shows the number N of quadrature points on the centerline required to resolve the integrals for each δ. Note that N can be extremely large for $\delta \approx 0$; however, if we use, say, the value $\delta = 0.25a$, we can benefit tremendously by reducing the number of quadrature points due to the regularization of the integrands.

Finally, we compute the velocity in the fluid using the Keller–Rubinow theory, (22), and compare it to the asymptotic formula given in [14]. Figure 5 shows the y-component of the fluid velocity along the line $(x, 0, 0)$ where x varies form the surface of the torus to a distance of about $10a$. The left panel shows the result for $a = 0.01$, a value that allows the computation of the integrals with $N = 200$ points

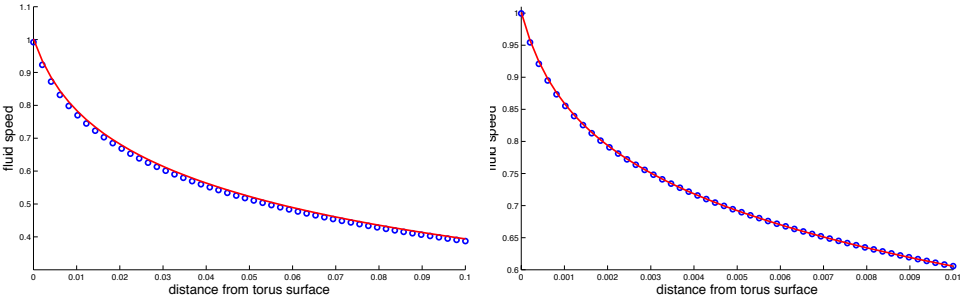


Figure 5. Fluid velocities resulting from tori moving horizontally with constant velocity $(u, v, w) = (0, 1, 0)$. The tori are given by centerline radius $R = 1/2\pi$ and cross-sectional radius $a = 0.01$ (left) and $a = 0.001$ (right). The plots show the y component of velocity v at points $(x, 0, 0)$ as x varies from the torus surface to $x = 10a$. The velocities for the Johnson and Wu theory [14] are shown as circles. The solid line represents the velocities for the regularized Keller–Rubinow theory with $\delta = 0.4/N$ and $N = 200$ (left) and $N = 2000$ (right).

on the centerline. The right panel shows the results for $a = 0.001$ and $N = 2000$. In this case, the forces used in the Keller–Rubinow theory were computed by inverting (21) by enforcing the velocity boundary condition on a curve on the outer surface of the torus (the curve corresponding to $r = R + a$, $z = 0$). The circles represent the asymptotic theory in [14] while the solid line is from Keller–Rubinow. Note that the agreement is better for the more slender torus.

5.1.2. *A straight slender body.* The second validation problem is the one of a straight slender filament of unit length. Chwang and Wu [4] developed an exact solution for the translation of a prolate spheroid, which we use as a reference. We emphasize that our formulas have been derived for a cylindrical tube of constant circular cross-section, as depicted in Figure 6, so the geometry is not exactly the same as the solution in [4]. For this reason, we cannot expect our solution to converge to the one in [4]; however, a qualitative comparison is instructive. For the regularized theories, we use a slender cylinder whose axis coincides with the x-axis and whose radius is $a = 0.01$. The reference is the exact solution for the prolate spheroid $4x^2 + (y^2 + z^2)/a^2 = 1$. The two slender bodies have the same cross section when $x = 0$ only as shown in Figure 6. At other values of x, the cylinder is wider than the prolate spheroid. We invert a discrete version of (22) based on the trapezoid rule with $N = 401$ points, enforcing the velocity boundary condition of $(u, v, w) = (1, 1, 0)$ on the surface of the cylinder $(y = a)$.

We then select points in the fluid along a straight line perpendicular to the slender bodies and emanating from the center of them, given by $(0, y, 0)$ for $y \in [a, 10a]$ (see Figure 6 for reference). We compute the fluid velocity there using both the regularized Keller–Rubinow theory for the cylindrical tube and the exact solution of Chwang and Wu for the prolate spheroid. The results are shown on the left panel of

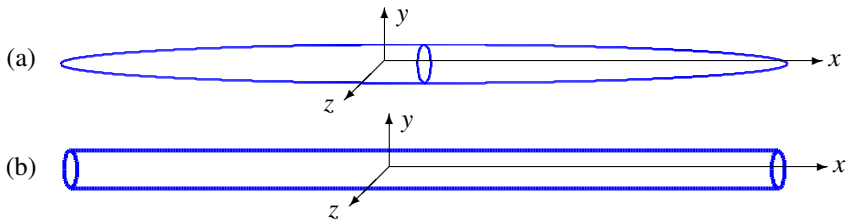


Figure 6. Difference in the shape of slender bodies. Panel (a) shows the prolate spheroid for which an exact solution is known [4]. Panel (b) shows the shape addressed in our work. The figures use a radius of $a = 0.025$ for visualization purposes. The cross-sections are equal only at $x = 0$.

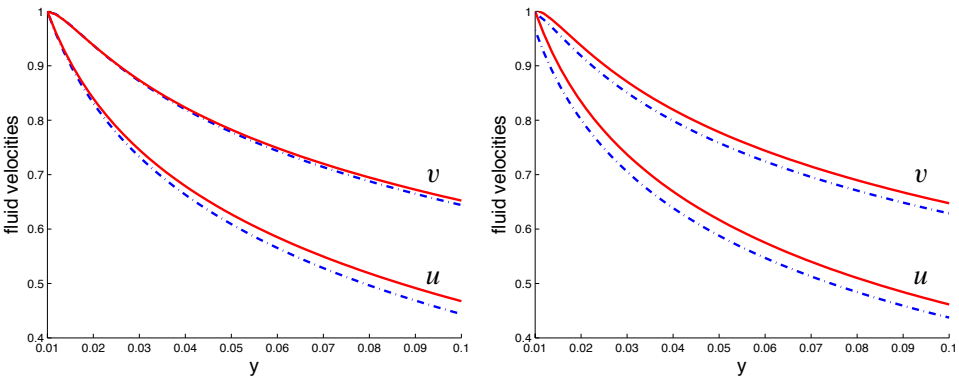


Figure 7. Fluid velocities resulting from a straight slender body moving with constant velocity $(u, v, w) = (1, 1, 0)$. The axis of the slender body is the x axis. The length of the body is 1 and the radius is $a = 0.01$. The number of quadrature points was $N = 401$ and $\delta = 0.5/N$. The left panel shows the x and the y components of velocity (u, v) at points $(0, y, 0)$ for $y \in [a, 10a]$. The right panel shows the same velocity components at points $(0.25, y, 0)$, at a cross section halfway between the center and the nose of the tube. The solid curves are from the regularized Keller–Rubinow theory while the dashed lines are the exact solution of a prolate spheroid given in [4].

Figure 7. The solid curves are for the straight cylinder using the regularized Keller–Rubinow theory and the dashed curves are the exact prolate spheroid solution. The right panel of the figure shows similar results but computing the fluid velocity along the line $(0.25, y, 0)$ for $a \leq y \leq 10a$, which is halfway between the center and the nose of the slender bodies. Here we do not expect the solutions to agree due to the fact that the two slender bodies are different. Specifically, the velocities given by Chwang and Wu are not equal to 1 at the point $(0.25, a, 0)$ since that is not on the surface of the prolate spheroid, but it is on the surface of the slender cylinder. In spite of this, the curves agree qualitatively.

5.2. ***Application: Closed filaments.*** We apply both theories to the problem of closed filaments with a normal force proportional to curvature. Our filaments are

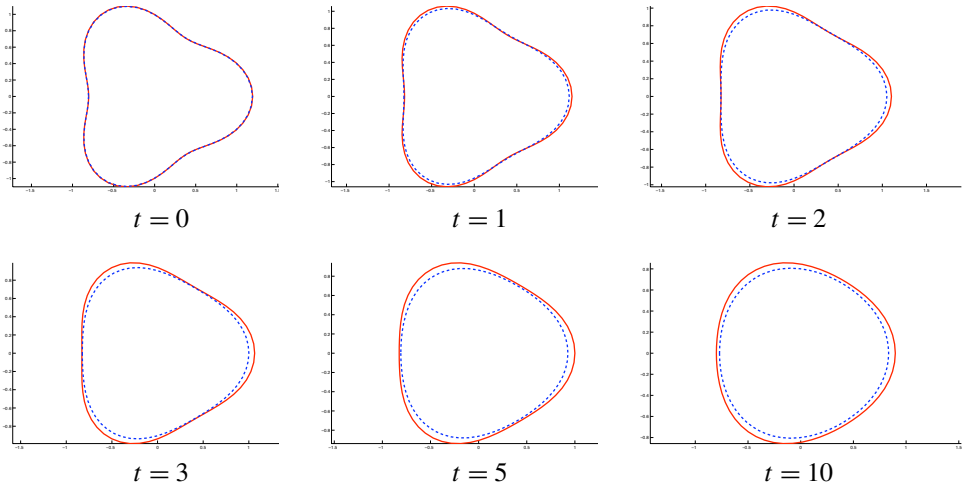


Figure 8. A closed filament $r = 1 + B\cos(n\theta)$ with normal forces at various times with $B = 0.2$, $N = 64$, $n = 3$. The slender body radius was $a = 0.1$ and the regularization parameter set to $\delta = 0.1$. The dashed lines correspond to Lighthill's theory, and the solid lines are results for the Keller–Rubinow method.

defined in cylindrical coordinates by

$$r = 1 + B\cos(n\theta) \quad \text{and} \quad z = 0$$

for various integers n, and we assign a force

$$F(\theta, t) = -\frac{1}{10}\kappa(\theta, t)\nu(\theta, t)\left(L(t) - \frac{3\pi}{2}\right)$$

where $\kappa(\theta, t)$ is the curvature of the filament, $\nu(\theta, t)$ is the inward unit normal, and $L(t)$ is the arclength of the filament. We expect such forces to restore the filaments to circular shapes. The filaments are discretized using N points and the forces are computed at those locations. Since the filaments and forces are smooth and periodic, we approximate derivatives along the filament using FFT interpolation and evaluate the integrals with a trapezoid rule. We update positions at each time step with a Runge-Kutta method. Results for both the Lighthill and Keller–Rubinow theories, (13) and (23) respectively, can be seen in Figure 8 for $r = 1 + B\cos(n\theta)$ with $B = 0.2$, $N = 64$, $n = 3$. The slender body radius was $a = 0.1$ and the regularization parameter set to $\delta = 0.1$. The differences in the shapes are due to the local terms in the expressions since the integrals are identical. The figure shows that for this set of parameters, the filament approaches the circular shape faster with Lighthill's method.

It has been noted in the Introduction that subtracting the singularity of the Keller–Rubinow integral like in (20) or (23) can lead to numerical instabilities in filaments with high frequency components. This is the case when the parameter δ is chosen

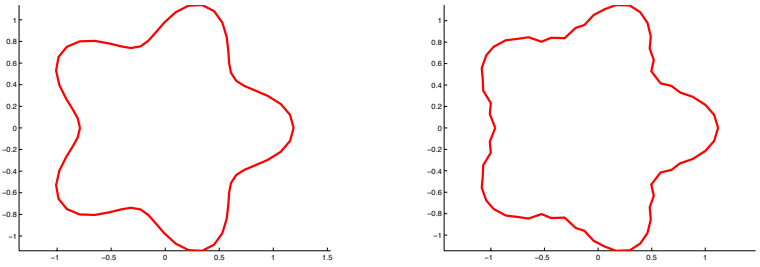


Figure 9. Solution at time $t = 1$ using the regularized theory of Keller–Rubinow, (21). The closed filament was given initially by $r = 1 + B\cos(n\theta)$ with $B = 0.3$, $N = 64$, $n = 5$. The slender body radius was $a = 0.1$ and two different regularizations were used. Note the formation of instabilities in the filament with less regularization.

too small for the regularization to provide stability. Figure 9 shows a closed filament with wave number $n = 5$ after a short simulation time using two different values of the regularization parameter δ. The smaller value is too small to prevent instabilities. Figure 10 shows that the filament computation remains stable for long times when the larger regularization parameter is used. The computations were done using (23).

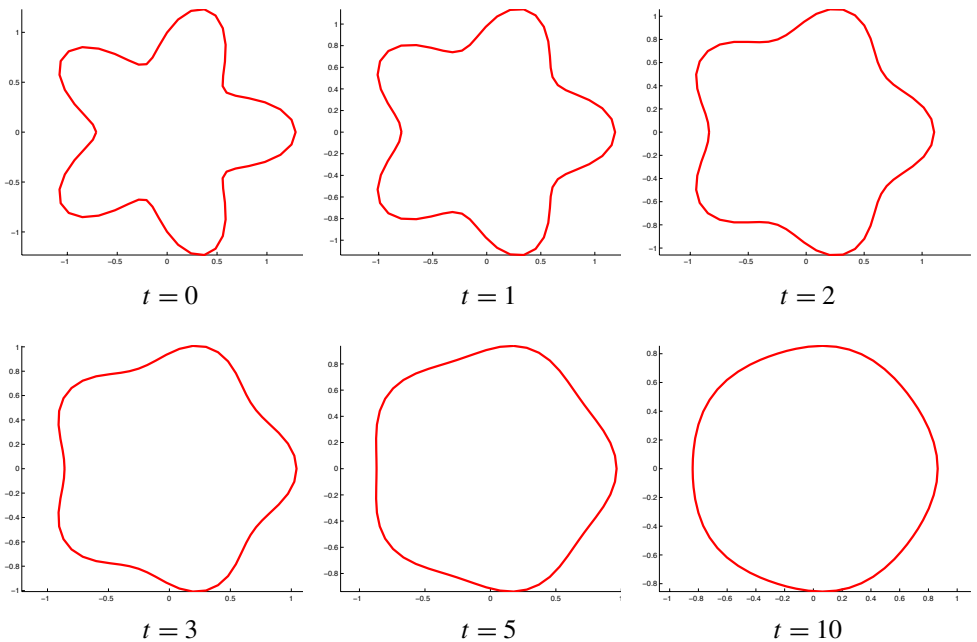


Figure 10. Solution at various times using the regularized theory of Keller and Rubinow, (21). The closed filament was given initially by $r = 1 + B\cos(n\theta)$ with $B = 0.3$, $N = 64$, $n = 5$. The slender body radius was $a = 0.1$ and the regularization parameter set to $\delta = 0.1$.

5.3. ***Application: Swimming organism.*** Biological applications of slender body theory also call for open filaments such as in the case of cilia or flagella. Sperm swimming in an infinite fluid, for example, often exhibit planar beat in which a wave travels along the flagellum from head to tail. Internal mechanisms in the flagellum produce time-dependent forces along it that result in swimming motions through interactions with the fluid. Relevant mathematical analysis and computational modeling of swimming "filaments" or cylindrical tubes can be found in [10; 23; 3; 4; 14; 6]. Our goal is to show the applicability of our result to this type of motion, so we present an idealized swimming microorganism modeled as a single sinusoidal filament with growing amplitude from head to tail. Simulations of the motion and flow field around flagella in which the shape was represented parametrically are found, for example, in [7; 9; 22; 21].

In time, the organism moves according to a traveling wave translating down the body. There are two types of forces involved. The organism is defined by N points equally distributed along the length which lies in the xy-plane so that the organism points are $(x_k, y_k, 0)$ for $k = 1, 2, \ldots, N$. Consecutive points are connected by springs whose resting lengths are given by their initial position. The first type of force is the spring force (Hooke's law) that develops as the end points of the springs move causing them to contract or stretch. The second type of force is due to an imposed time-dependent curvature of the filament consistent with the idealized shape

$$y(x) = Y_0\, x\ \sin\left(\frac{2\pi}{L}(x-t)\right), \qquad 0 \le x \le L.$$

This is done by writing a discrete energy function

$$\mathscr{E}_h(x_1, y_1, \ldots, x_k, y_k, \ldots, x_N, y_N) = \frac{h}{2} S_1 \sum_{n=1}^{N} \left(\frac{(\vec{x}_{n+1} - \vec{x}_n) \times (\vec{x}_n - \vec{x}_{n-1})}{h^3} - \kappa_n \right)^2$$

where h is the separation between contiguous points, S_1 is a constant and κ_n is the target curvature at point n. The cross product is known to approximate curvature [9]. Then the curvature force is defined as

$$\vec{F}_k = -\left(\frac{\partial \mathscr{E}_h}{\partial x_k}, \frac{\partial \mathscr{E}_h}{\partial y_k}, 0 \right).$$

By defining forces this way, we guarantee that the net force and net torque are identically zero. More details of this force can be found in [9; 8; 5]. The goal of this example is show an application of the methodology developed here even if the biological aspects are not developed.

The flagellum is defined by N points equally distributed along the length. Their velocities are computed using the trapezoid rule on the Keller–Rubinow integral in (21), and the position is updated at each time step with Euler's method. The

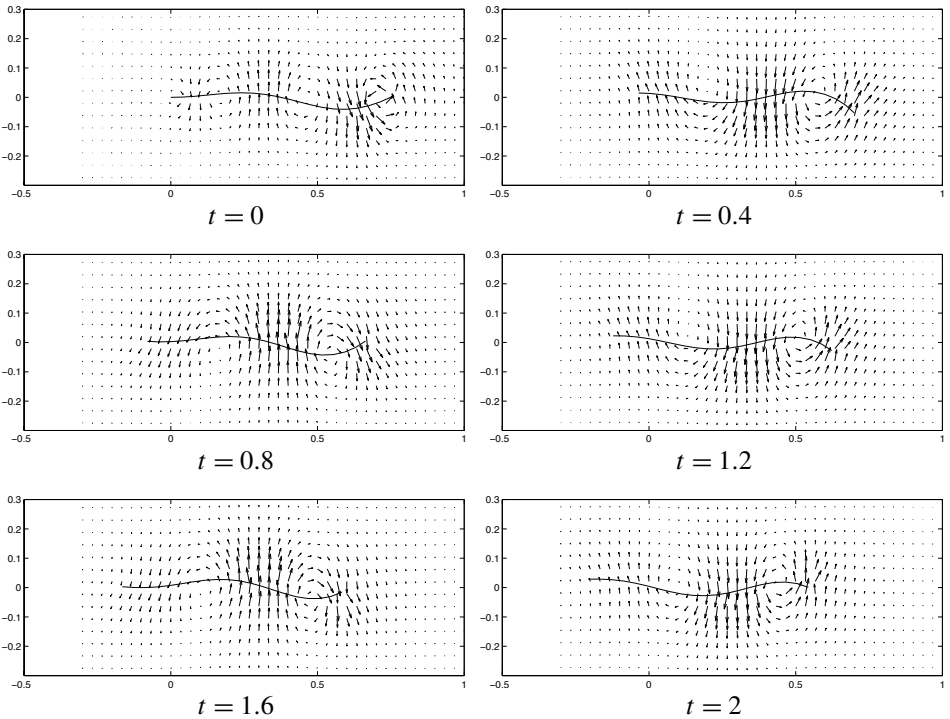


Figure 11. Snapshots of the flagella at different times. The motion is governed by (21), and the fluid velocities are calculated using (22). For these calculations, the length of the flagella is $L = 0.75$, and we have used $N = 20$ nodes with $\delta = 2h = 2L/(N-1)$ and $a = 2\delta$.

fluid velocities are computed using the trapezoid rule on the integral in (22). All necessary derivatives along the filament are calculated with simple finite differences. Figure 11 shows the results for the parameters $L = 0.75$, $N = 20$, $\delta = 2h = 2L/(N-1)$ and $a = 2\delta$. This results in a dimensionless oscillation period of $T = 0.75$ so that the snapshots in the figure cover more than two periods. The time step was set to $\Delta t = 10^{-5}$. All frames cover exactly the same spatial domain so that the swimming motion (leftward) of the organism is appreciable. The fluid motion in the plane of the organism shows the rotations that are typical of this motion [9; 8; 5].

6. Conclusions

We have derived a regularized formulation of the slender body theories developed by Lighthill and by Keller and Rubinow. The main purpose is to provide a modified version of each theory that retains the asymptotic order of the original formula but results in expressions that are more amenable to computation. Specifically, the

Keller and Rubinow theory relies on the exact cancellation of singular functions, which is not possible to accomplish numerically without some type of regularization. In the case of Lighthill's theory, the advantage of our approach is that the gap that is removed from the line integral can be restored. The uninterrupted integral along the centerline of the slender body is important especially in closed filaments where one can take advantage of the periodicity of the problem using high-order quadratures.

The results show that both regularized theories result in the same integral along the filament, (13) and (21), which allows a direct comparison between them. The two theories differ by the local terms only due to the way in which they are derived and the order of the asymptotic expansions. The regularization parameter δ should be considered a numerical parameter that removes the singularity of the original expressions in a way that maintains the asymptotic order and stabilizes the computation of the integrals. The validation studies show that comparable results can be obtained with $\delta \approx 0$ and $\delta = O(a)$ while the latter case provides a substantial advantage in the number of quadrature points needed to compute the resulting integrals accurately. We show by example that sufficient regularization (i.e. large enough δ) is necessary to stabilize high wave numbers in the representation of the filaments.

The theory presented here involves regularized stokeslets and dipoles along the centerline of the slender body. The inclusion of other elements, such as rotlets for a torque load, is also possible since regularized versions of them are available [1].

Appendix A: Details of the inner velocity expansion

To compute the inner velocity in (15), we will need the following approximations for $|\boldsymbol{x}_0| = \rho \ll q$:

$$J_1 = \int_{-q}^{q} \frac{H_1(\sqrt{|\boldsymbol{x}_0|^2+z^2})\,dz}{(|\boldsymbol{x}_0|^2+z^2)^{1/2}} \approx \ln\frac{4q^2}{|\boldsymbol{x}_0|^2+\delta^2} + \frac{2\delta^2}{|\boldsymbol{x}_0|^2+\delta^2} + \frac{\left(|\boldsymbol{x}_0|^2-\delta^2\right)}{2q^2}$$

$$J_{2a} = \int_{-q}^{q} \frac{H_3(\sqrt{|\boldsymbol{x}_0|^2+z^2})\,dz}{(|\boldsymbol{x}_0|^2+z^2)^{3/2}} \approx \frac{2(|\boldsymbol{x}_0|^2-\delta^2)}{(|\boldsymbol{x}_0|^2+\delta^2)^2} - \frac{1}{q^2}$$

$$J_{2b} = \int_{-q}^{q} \frac{H_2(\sqrt{|\boldsymbol{x}_0|^2+z^2})\,dz}{(|\boldsymbol{x}_0|^2+z^2)^{3/2}} \approx \frac{2}{(|\boldsymbol{x}_0|^2+\delta^2)} - \frac{1}{q^2}$$

$$J_3 = \int_{-q}^{q} \frac{H_4(\sqrt{|\boldsymbol{x}_0|^2+z^2})\,dz}{(|\boldsymbol{x}_0|^2+z^2)^{5/2}} \approx \frac{4}{3(|\boldsymbol{x}_0|^2+\delta^2)^2} - \frac{1}{2q^4}$$

$$J_4 = \int_{-q}^{q} \frac{z H_2(\sqrt{|\boldsymbol{x}_0|^2+z^2})dz}{(|\boldsymbol{x}_0|^2+z^2)^{3/2}} = 0$$

$$J_5 = \int_{-q}^{q} \frac{z H_4(\sqrt{|\boldsymbol{x}_0|^2+z^2})dz}{(|\boldsymbol{x}_0|^2+z^2)^{5/2}} = 0$$

$$J_6 = \int_{-q}^{q} \frac{z^2 H_2(\sqrt{|\boldsymbol{x}_0|^2+z^2})\,dz}{(|\boldsymbol{x}_0|^2+z^2)^{3/2}} \approx \ln\frac{4q^2}{|\boldsymbol{x}_0|^2+\delta^2} - 2 + \frac{3(|\boldsymbol{x}_0|^2+\delta^2)}{2q^2}$$

$$J_7 = \int_{-q}^{q} \frac{z^2 H_4(\sqrt{|\boldsymbol{x}_0|^2+z^2})\,dz}{(|\boldsymbol{x}_0|^2+z^2)^{5/2}} \approx \frac{2}{3(|\boldsymbol{x}_0|^2+\delta^2)} - \frac{1}{q^2}$$

Neglecting terms of order $O(a^2/q^2)$, $O(\delta^2/q^2)$ and $O(|\boldsymbol{x}_0|^2/q^2)$, the velocity can be written as

$$\begin{aligned}
8\pi\mu(\boldsymbol{u}(\boldsymbol{x}_0)-\boldsymbol{U}) &= \boldsymbol{f}(J_1-2AJ_{2a})+(\boldsymbol{f}\cdot\boldsymbol{x}_0)\boldsymbol{x}_0\,(J_{2b}+6AJ_3)+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}\,(J_6+6AJ_7)\\
&= \boldsymbol{f}\left[\ln\frac{4q^2}{|\boldsymbol{x}_0|^2+\delta^2}+\frac{2\delta^2}{|\boldsymbol{x}_0|^2+\delta^2}-2A\left(\frac{2|\boldsymbol{x}_0|^2-\delta^2}{(|\boldsymbol{x}_0|^2+\delta^2)^2}-\frac{1}{q^2}\right)\right]\\
&\quad+(\boldsymbol{f}\cdot\boldsymbol{x}_0)\boldsymbol{x}_0\left[\frac{2}{|\boldsymbol{x}_0|^2+\delta^2}+6A\left(\frac{4}{3(|\boldsymbol{x}_0|^2+\delta^2)^2}\right)\right]\\
&\quad+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}\left[\ln\frac{4q^2}{|\boldsymbol{x}_0|^2+\delta^2}-2+6A\left(\frac{2}{3(|\boldsymbol{x}_0|^2+\delta^2)}-\frac{1}{q^2}\right)\right].
\end{aligned}$$

When the evaluation point is on the surface, $|\boldsymbol{x}_0| = a$, the velocity must be independent of the particular surface point, so that the coefficient of $(\boldsymbol{f}\cdot\boldsymbol{x}_0)\boldsymbol{x}_0$ must vanish. This leads to

$$A = -\frac{a^2+\delta^2}{4},$$

so that this choice of A is consistent with the boundary conditions at the filament surface, we have for $|\boldsymbol{x}_0| = a$

$$8\pi\mu\boldsymbol{U} = 8\pi\mu\boldsymbol{v}(s) - (\boldsymbol{f}+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s})\left[\ln\frac{4q^2}{a^2+\delta^2}+1\right]+4(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s},$$

and the final velocity expression is (16):

$$\begin{aligned}
8\pi\mu\boldsymbol{u}(\boldsymbol{x}_0) &= 8\pi\mu\boldsymbol{v}(\sigma_0)-(\boldsymbol{f}_0+(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\left[\ln\frac{|\boldsymbol{x}_0|^2+\delta^2}{a^2+\delta^2}+\frac{(|\boldsymbol{x}_0|^2-\delta^2)(|\boldsymbol{x}_0|^2-a^2)}{(|\boldsymbol{x}_0|^2+\delta^2)^2}\right]\\
&\quad+2(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}\frac{|\boldsymbol{x}_0|^2(|\boldsymbol{x}_0|^2-a^2)}{(|\boldsymbol{x}_0|^2+\delta^2)^2}+2(\boldsymbol{f}_0\cdot\boldsymbol{x}_0)\boldsymbol{x}_0\frac{|\boldsymbol{x}_0|^2-a^2}{(|\boldsymbol{x}_0|^2+\delta^2)^2}+O(\epsilon^2),
\end{aligned}$$

where

$$\epsilon = \max\left(\frac{|\boldsymbol{x}_0|}{q},\frac{\delta}{q}\right).$$

Appendix B: Matching

Consider the matching equation

$$\int_0^1 J(\boldsymbol{r},\rho,\delta,\boldsymbol{f}(\sigma))\,d\sigma = 8\pi\mu\boldsymbol{v}(s) - (\boldsymbol{f}+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s})\left[\ln\frac{\rho^2+\delta^2}{a^2+\delta^2}+1\right] + 2(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s} + \frac{2(\boldsymbol{f}\cdot\boldsymbol{x}_0)\boldsymbol{x}_0}{\rho^2}. \quad (24)$$

Since the terms containing ρ came from integrals J_1, J_{2b} and J_6, it is natural to consider writing

$$\begin{aligned}\int_0^1 J(\boldsymbol{r},\rho,\delta,\boldsymbol{f}(\sigma))\,d\sigma &= \int_0^1 \big(J(\boldsymbol{r},\rho,\delta,\boldsymbol{f}(\sigma)) - \boldsymbol{f}(s)K_1(\rho,\delta) - (\boldsymbol{f}\cdot\boldsymbol{x}_0)\boldsymbol{x}_0K_2(\rho,\delta) - (\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}K_3(\rho,\delta)\big)\,d\sigma \\ &\quad + \int_0^1 [\boldsymbol{f}(s)K_1(\rho,\delta) + (\boldsymbol{f}\cdot\boldsymbol{x}_0)\boldsymbol{x}_0K_2(\rho,\delta) + (\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}K_3(\rho,\delta)]d\sigma, \quad (25)\end{aligned}$$

where, setting $t=\sigma-s$, we define

$$K_1(\rho,\delta) = \frac{H_1(\sqrt{t^2+\rho^2})}{(t^2+\rho^2)^{1/2}}, \qquad K_2(\rho,\delta) = \frac{H_2(\sqrt{t^2+\rho^2})}{(t^2+\rho^2)^{3/2}},$$
$$K_3(\rho,\delta) = \frac{H_2(\sqrt{t^2+\rho^2})\,t^2}{(t^2+\rho^2)^{3/2}}.$$

We approximate the first integral on the right-hand side of (25) by setting $\rho=0$ so that the outer solution is approximated by

$$\begin{aligned}\int_0^1 J(\boldsymbol{r},\rho,\delta,\boldsymbol{f}(\sigma))\,d\sigma &\approx \int_0^1 [J(\boldsymbol{r}_0,0,\delta,\boldsymbol{f}(\sigma)) - \boldsymbol{f}(s)K_1(0,\delta) - (\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}K_3(0,\delta)]d\sigma \\ &\quad + \int_0^1 [\boldsymbol{f}(s)K_1(\rho,\delta) + (\boldsymbol{f}\cdot\boldsymbol{x}_0)\boldsymbol{x}_0K_2(\rho,\delta) + (\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}K_3(\rho,\delta)]d\sigma, \quad (26)\end{aligned}$$

where $\boldsymbol{r}_0 = X(s)-X(s+t)$.

Then, by using the integrals

$$\begin{aligned}\int_0^1 K_1(\rho,\delta)\,d\sigma &= \int_{-s}^{1-s}\frac{(t^2+\rho^2+2\delta^2)\,dt}{(t^2+\rho^2+\delta^2)^{3/2}} \\ &= \ln(4s(1-s)) - \ln(\rho^2+\delta^2) + O(\rho^2) + O(\delta^2),\end{aligned}$$

$$\begin{aligned}\int_0^1 K_2(\rho,\delta)\,d\sigma &= \int_{-s}^{1-s}\frac{dt}{(t^2+\rho^2+\delta^2)^{3/2}} \\ &= \frac{2}{\rho^2+\delta^2} + O(\rho^2) = \frac{2}{\rho^2} + O(\rho^2) + O(\delta^2/\rho^2),\end{aligned}$$

$$\int_0^1 K_3(\rho,\delta)\,d\sigma = \int_{-s}^{1-s} \frac{t^2\,dt}{(t^2+\rho^2+\delta^2)^{3/2}} = \ln(4s(1-s)) - 2 - \ln(\rho^2+\delta^2) + O(\rho^2),$$

(see the expressions in (7)), Equation (24) becomes

$$\int_0^1 \big(J(\boldsymbol{r}_0,0,\delta,\boldsymbol{f}(\sigma)) - \boldsymbol{f}_0 K_1(0,\delta) - (\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s}K_3(0,\delta)\big)\,d\sigma + (\boldsymbol{f}_0 + (\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\big[\ln(4s(1-s)) - \ln(\rho^2+\delta^2)\big] - 2(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s} + \frac{2(\boldsymbol{f}_0\cdot\boldsymbol{x}_0)\boldsymbol{x}_0}{\rho^2} = 8\pi\mu\boldsymbol{v}(s) - (\boldsymbol{f}_0 + (\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s})\left[\ln\frac{\rho^2+\delta^2}{a^2+\delta^2} + 1\right] + 2(\boldsymbol{f}_0\cdot\boldsymbol{s})\boldsymbol{s} + \frac{2(\boldsymbol{f}_0\cdot\boldsymbol{x}_0)\boldsymbol{x}_0}{\rho^2}.$$

Appendix C: Simplifying the velocity expression

The filament velocity formula (19) can be simplified by using (7) and the identity

$$\int_{-s}^{1-s} \frac{dt}{(t^2+a^2+\delta^2)^{1/2}} = \ln(4s(1-s)) - \ln\left(a^2+\delta^2\right) + O(a^2+\delta^2);$$

we can write the filament velocity as

$$\begin{aligned}8\pi\mu\boldsymbol{v}(s) = {}& \int_{-s}^{1-s} \frac{\boldsymbol{f}(s+t)}{(|\boldsymbol{r}_0|^2+\delta^2)^{1/2}} + \frac{(\boldsymbol{f}(s+t)\cdot\boldsymbol{r}_0)\boldsymbol{r}_0}{|\boldsymbol{r}_0|^2\,(|\boldsymbol{r}_0|^2+\delta^2)^{1/2}} - \frac{\boldsymbol{f}(s)+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}}{(t^2+\delta^2)^{1/2}}\,dt \\ & + \delta^2\int_{-s}^{1-s} \frac{\boldsymbol{f}(s+t)}{(|\boldsymbol{r}_0|^2+\delta^2)^{3/2}} - \frac{(\boldsymbol{f}(s+t)\cdot\boldsymbol{r}_0)\boldsymbol{r}_0}{|\boldsymbol{r}_0|^2\,(|\boldsymbol{r}_0|^2+\delta^2)^{3/2}} - \frac{\boldsymbol{f}(s)-(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}}{(t^2+\delta^2)^{3/2}}\,dt \\ & + \int_{-s}^{1-s} \frac{\boldsymbol{f}(s)+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}}{(t^2+a^2+\delta^2)^{1/2}}\,dt - (\boldsymbol{f}+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}) + 2(\boldsymbol{f}-(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}),\end{aligned}$$

where $\boldsymbol{r}_0 = X(s) - X(s+t)$.

Lemma 6.1 below shows that the second integral in this expression is $O(\delta^2\ln\delta)$, so that the final expression for the filament velocity up to this order is

$$\begin{aligned}8\pi\mu\boldsymbol{v}(s) = {}& \int_{-s}^{1-s} \frac{\boldsymbol{f}(s+t)}{(|\boldsymbol{r}_0|^2+\delta^2)^{1/2}} + \frac{(\boldsymbol{f}(s+t)\cdot\boldsymbol{r}_0)\boldsymbol{r}_0}{|\boldsymbol{r}_0|^2\,(|\boldsymbol{r}_0|^2+\delta^2)^{1/2}} - \frac{\boldsymbol{f}(s)+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}}{(t^2+\delta^2)^{1/2}}\,dt \\ & + \int_{-s}^{1-s} \frac{\boldsymbol{f}(s)+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}}{(t^2+a^2+\delta^2)^{1/2}}\,dt - (\boldsymbol{f}+(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}) + 2(\boldsymbol{f}-(\boldsymbol{f}\cdot\boldsymbol{s})\boldsymbol{s}).\end{aligned}$$

Lemma 6.1. *Let a filament be defined by the curve $X(t)$, where t is the arclength parameter. Let $f(t)$ be a smooth function, $\boldsymbol{r}(t) = X(t) - X(0)$ and $\delta \ll \ell$. Then*

$$I(\delta) = \delta^2\int_{-\ell}^{\ell} \frac{f(t)}{(|\boldsymbol{r}|^2+\delta^2)^{3/2}} - \frac{f(0)}{(t^2+\delta^2)^{3/2}}\,dt = -\left(\tfrac{1}{4}K_0^2 f_0 + f_0''\right)\delta^2\ln\delta + O(\delta^2)$$

where $f_0 = f(0)$, $f_0' = f'(0)$, $f_0'' = f''(0)$ and K_0 is the filament curvature at $X(0)$.

Proof. Consider first the integral

$$I_1(\delta) = \delta^2 \int_0^\ell \frac{f(t)}{(|\boldsymbol{r}(t)|^2 + \delta^2)^{3/2}}\, dt$$

and write for $t \ll 1$

$$\begin{aligned}
\boldsymbol{r}(t) &= t\hat{\tau} + \tfrac{1}{2}t^2 K_0 \hat{n} + \tfrac{1}{6}t^3 (K_0' \hat{n} - K_0^2 \hat{\tau}) + O(t^4),\\
|\boldsymbol{r}(t)|^2 &= \boldsymbol{r} \cdot \boldsymbol{r} = t^2 - \tfrac{1}{12} K_0^2 t^4 + O(t^5),\\
|\boldsymbol{r}(t)| &= t - \tfrac{1}{24} K_0^2 t^3 + O(t^4),\\
|\boldsymbol{r}(t)|_t &= 1 - \tfrac{1}{8} K_0^2 t^2 + O(t^3).
\end{aligned}$$

where $\hat{\tau}$ and $\hat{n}$ are the tangent and normal unit vectors at $X(0)$. Using these expressions we write for $t \ll 1$

$$\begin{aligned}
f(t) &= f_0 + t f_0' + \tfrac{1}{2} t^2 f_0'' + O(t^3)\\
&= f_0 |\boldsymbol{r}(t)|_t \left[1 + \tfrac{1}{8} K_0^2 |\boldsymbol{r}(t)|^2\right] + f_0' |\boldsymbol{r}(t)| |\boldsymbol{r}(t)|_t + \tfrac{1}{2} f_0'' |\boldsymbol{r}(t)|^2 |\boldsymbol{r}(t)|_t + O(t^3)\\
&= |\boldsymbol{r}(t)|_t \left[f_0 + |\boldsymbol{r}(t)| f_0' + \tfrac{1}{2} |\boldsymbol{r}(t)|^2 \left(f_0'' + \tfrac{1}{4} K_0^2 f_0\right)\right] + O(t^3)\\
&= |\boldsymbol{r}(t)|_t \, \mathcal{A}(|\boldsymbol{r}(t)|) + O(t^3).
\end{aligned}$$

Then

$$\begin{aligned}
I_1(\delta) &= \delta^2 \int_0^\ell \frac{f(t) - |\boldsymbol{r}(t)|_t \, \mathcal{A}(|\boldsymbol{r}(t)|)}{(|\boldsymbol{r}(t)|^2 + \delta^2)^{3/2}}\, dt + \delta^2 \int_0^\ell \frac{|\boldsymbol{r}(t)|_t \, \mathcal{A}(|\boldsymbol{r}(t)|)}{(|\boldsymbol{r}(t)|^2 + \delta^2)^{3/2}}\, dt\\
&= J_1(\delta) + J_2(\delta).
\end{aligned}$$

By construction we know that $J_1(\delta) = O(\delta^2)$. On the other hand, we can let $y = |\boldsymbol{r}(t)|$ and write

$$J_2(\delta) = \delta^2 \int_0^R \frac{\mathcal{A}(y)}{(y^2 + \delta^2)^{3/2}}\, dy$$

where $R = |\boldsymbol{r}(\ell)|$. So

$$\begin{aligned}
J_2(\delta) = f_0 \int_0^R \frac{\delta^2}{(y^2+\delta^2)^{3/2}}\, dy + f_0' \int_0^R \frac{\delta^2 y}{(y^2+\delta^2)^{3/2}}\, dy\\
+ \tfrac{1}{2}\left(f_0'' + \tfrac{1}{4} K_0^2 f_0\right) \int_0^R \frac{\delta^2 y^2}{(y^2+\delta^2)^{3/2}}\, dy
\end{aligned}$$

or

$$\begin{aligned}
J_2(\delta) = f_0 \frac{R}{\sqrt{R^2+\delta^2}} - f_0' \left[\frac{\delta^2}{\sqrt{R^2+\delta^2}} - \delta\right]\\
+ \tfrac{1}{2}\left(f_0'' + \tfrac{1}{4} K_0^2 f_0\right)\left[\delta^2 \ln\left(R + \sqrt{R^2+\delta^2}\right) - \delta^2 \ln \delta - \frac{\delta^2 R}{\sqrt{R^2+\delta^2}}\right].
\end{aligned}$$

Now the second half of the original integral is

$$I_2(\delta) = \delta^2 \int_{-\ell}^{0} \frac{f(t)}{(|\boldsymbol{r}(t)|^2+\delta^2)^{3/2}}\,dt = \delta^2 \int_{0}^{\ell} \frac{f(-t)}{(|\boldsymbol{r}(-t)|^2+\delta^2)^{3/2}}\,dt.$$

Using the same approach we conclude that

$$\begin{aligned} I_2(\delta) &= f_0 \frac{P}{\sqrt{P^2+\delta^2}} + f_0' \left[\frac{\delta^2}{\sqrt{P^2+\delta^2}} - \delta\right] \\ &\quad + \tfrac{1}{2}\big(f_0'' + \tfrac{1}{4}K_0^2 f_0\big)\left[\delta^2 \ln\big(P+\sqrt{P^2+\delta^2}\big) - \delta^2 \ln\delta - \frac{\delta^2 P}{\sqrt{P^2+\delta^2}}\right] + O(\delta^2), \end{aligned}$$

where $P = |\boldsymbol{r}(-\ell)|$.

Combining the results we have

$$\begin{aligned} I(\delta) &= f_0\left[\frac{R}{\sqrt{R^2+\delta^2}} + \frac{P}{\sqrt{P^2+\delta^2}} - \frac{2\ell}{\sqrt{\ell^2+\delta^2}}\right] + f_0'\left[\frac{\delta^2}{\sqrt{P^2+\delta^2}} - \frac{\delta^2}{\sqrt{R^2+\delta^2}}\right] \\ &\quad + \tfrac{1}{2}\big(f_0'' + \tfrac{1}{4}K_0^2 f_0\big)\left[\delta^2 \ln\big(P+\sqrt{P^2+\delta^2}\big) + \delta^2 \ln\big(R+\sqrt{R^2+\delta^2}\big)\right. \\ &\qquad \left. - 2\delta^2 \ln\delta - \frac{\delta^2 P}{\sqrt{P^2+\delta^2}} - \frac{\delta^2 R}{\sqrt{R^2+\delta^2}}\right] + O(\delta^2) \\ &= -\big(f_0'' + \tfrac{1}{4}K_0^2 f_0\big)\delta^2 \ln\delta + O(\delta^2). \end{aligned}$$

□

References

[1] J. Ainley, S. Durkin, R. Embid, P. Boindala, and R. Cortez, *The method of images for regularized Stokeslets*, J. Comput. Phys. **227** (2008), no. 9, 4600–4616. MR 2009e:76050 Zbl 05276040

[2] G. K. Batchelor, *An introduction to fluid dynamics*, Cambridge University Press, Cambridge, 1967. MR 2000j:76001 Zbl 0152.44402

[3] ———, *Slender-body theory for particles of arbitrary cross-section in Stokes flow*, J. Fluid Mech. **44** (1970), 419–440. MR 47 #1375 Zbl 0216.52401

[4] A. T. Chwang and T. Y. Wu, *A note on the helical movements of micro-organisms*, Proc. Roy. Soc. Lond. B **178** (1971), 327–346.

[5] R. Cortez, *A vortex/impulse method for immersed boundary motion in high Reynolds number flows*, J. Comput. Phys. **160** (2000), no. 1, 385–400. MR 2000m:76087 Zbl 0964.76067

[6] R. D. Dresdner and D. F. Katz, *Relationships of mammalian sperm motility and morphology to hydrodynamic aspects of cell function*, Biol. Reprod. **25** (1981), no. 5, 920–930.

[7] L. Fauci and A. McDonald, *Sperm motility in the presence of boundaries*, Bull. Math. Biol. **57** (1995), 679–699.

[8] L. J. Fauci and A. L. Fogelson, *Truncated Newton methods and the modeling of complex immersed elastic structures*, Comm. Pure Appl. Math. **46** (1993), no. 6, 787–818. MR 94e:92018 Zbl 0741.76103

[9] L. J. Fauci and C. S. Peskin, *A computational model of aquatic animal locomotion*, J. Comput. Phys. **77** (1988), no. 1, 85–108. MR 90e:92007 Zbl 0641.76140

[10] J. Gray and G. J. Hancock, *The propulsion of sea-urchin spermatozoa*, J. Exp. Biol. **32** (1955), no. 4, 802–814.

[11] S. Gueron and N. Liron, *Ciliary motion modeling, and dynamic multicilia ineractions*, Biophys. J. **63** (1992), 1045–1058.

[12] ———, *Simulations of three-dimensional ciliary beats and cilia interactions*, Biophys. J. **65** (1993), 499–507.

[13] R. E. Johnson and C. J. Brokaw, *A comparison between resistive-force theory and slender-body theory*, Biophys. J. **25** (1979), 113–127.

[14] R. E. Johnson and T. Y. Wu, *The asymptotic solution formula for uniform flow past a slender torus*, J. Fluid Mech. **95** (1979), 263–277.

[15] R. E. Johnson, *An improved slender-body theory for Stokes flow*, J. Fluid Mech. **99** (1980), no. 2, 411–431. MR 81g:76045 Zbl 0447.76037

[16] J. B. Keller and S. I. Rubinow, *Slender-body theory for slow viscous flow*, J. Fluid Mech. **75** (1976), 705–714.

[17] J. Lighthill, *Mathematical biofluiddynamics*, Regional Conference Series in Applied Mathematics, no. 17, Society for Industrial and Applied Mathematics, Philadelphia, 1975. MR 57 #9113 Zbl 0312.76076

[18] ———, *Flagellar hydrodynamics*, SIAM Rev. **18** (1976), no. 2, 161–230. MR 53 #2413 Zbl 0366.76099

[19] ———, *Helical distributions of Stokeslets: The centenary of a paper on slow viscous flow by the physicist h. a. lorentz*, J. Engrg. Math. **30** (1996), no. 1-2, 35–78. MR 97e:76099 Zbl 0883.76099

[20] M. J. Shelley and T. Ueda, *The Stokesian hydrodynamics of flexing, stretching filaments*, Phys. D **146** (2000), no. 1-4, 221–245. MR 2001j:74024 Zbl 1049.76016

[21] D. J. Smith, E. A. Gaffney, J. R. Blake, and J. C. Kirkman-Brown, *Human sperm accumulation near surfaces: a simulation study*, J. Fluid Mech. **621** (2009), 289–320. Zbl 1171.76476

[22] D. Smith, J. Blake, and E. Gaffney, *Fluid mechanics of nodal flow due to embryonic primary cilia*, J. R. Soc. Interface **5** (2008), 567–573.

[23] G. Taylor, *The action of waving cylindrical tails in propelling microscopic organisms*, Proc. Roy. Soc. London. Ser. A. **211** (1952), 225–239. MR 14,104e Zbl 0046.18904

[24] A.-K. Tornberg and M. J. Shelley, *Simulating the dynamics and interactions of flexible fibers in Stokes flows*, J. Comput. Phys. **196** (2004), no. 1, 8–40. MR 2004m:76164 Zbl 1115.76413

Received July 9, 2010. Revised August 29, 2011.

RICARDO CORTEZ: rcortez@tulane.edu
Mathematics Department, Tulane University, 6823 St. Charles Ave., New Orleans, LA 70118, United States
http://tulane.edu/sse/math/faculty/ricardo-cortez.cfm

MICHAEL NICHOLAS: mnichol@tulane.edu
Mathematics Department, Tulane University, 6823 St. Charles Ave., New Orleans, LA 70118, United States
http://faculty.carthage.edu/mnicholas

COMM. APP. MATH. AND COMP. SCI.
Vol. 7, No. 1, 2012

msp

NUMERICAL METHOD FOR EXPECTATIONS OF PIECEWISE DETERMINISTIC MARKOV PROCESSES

ADRIEN BRANDEJSKY, BENOÎTE DE SAPORTA AND FRANÇOIS DUFOUR

We present a numerical method to compute expectations of functionals of a piecewise deterministic Markov process. We discuss time dependent functionals as well as deterministic time horizon problems. Our approach is based on the quantization of an underlying discrete-time Markov chain. We obtain bounds for the rate of convergence of the algorithm. The approximation we propose is easily computable and is flexible with respect to some of the parameters defining the problem. An example illustrates the paper.

1. Introduction

The aim of this paper is to propose a practical numerical method to approximate some expectations related to a piecewise deterministic Markov process thanks to the quantization of a discrete-time Markov chain naturally embedded within the continuous-time process.

Piecewise deterministic Markov processes (PDMP's) have been introduced by M. H. A. Davis in [5] as a general class of stochastic models. PDMP's are a

This work was supported by ARPEGE program of the French National Agency of Research (ANR), project "FAUTOCOES", number ANR-09-SEGI-004.
MSC2010: primary 60J25, 65C20; secondary 60K10.
Keywords: expectation, piecewise deterministic Markov processes, quantization, numerical method.

family of Markov processes involving deterministic motion punctuated by random jumps. The motion depends on three local characteristics namely the flow Φ, the jump rate λ and the transition measure Q, which specifies the postjump location. Starting from the point x, the motion of the process follows the flow $\Phi(x,t)$ until the first jump time T_1, which occurs either spontaneously in a Poisson-like fashion with rate $\lambda(\Phi(x,t))$ or when the flow $\Phi(x,t)$ hits the boundary of the state space. In either case, the location of the process at the jump time T_1 is selected by the transition measure $Q(\Phi(x,T_1),\cdot)$ and the motion restarts from this new point X_{T_1} denoted by Z_1. We define similarly the time S_2 until the next jump, $T_2 = T_1 + S_2$ with the next postjump location defined by $Z_2 = X_{T_2}$ and so on. Thus, associated to the PDMP we have the discrete-time Markov chain $(Z_n, S_n)_{n\in\mathbb{N}}$, given by the postjump locations and the interjump times. A suitable choice of the state space and the local characteristics Φ, λ and Q provides stochastic models covering a great number of problems of operations research as described in [5, Section 33].

We are interested in the approximation of expectations of the form

$$\boldsymbol{E}_x\left[\int_0^{T_N} l(X_t)\,dt + \sum_{j=1}^{N} c(X_{T_j^-})\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}\right]$$

where $(X_t)_{t\geq 0}$ is a PDMP and l and c are some nonnegative, real-valued, bounded functions and ∂E is the boundary of the domain. Such expectations are discussed by M. H. A. Davis in [5, Chapter 3]. They often appear as cost or reward functions in optimization problems. The first term is referred to as the running cost while the second may be called the boundary jump cost. Besides, they are quite general since Davis shows how a "wide variety of apparently different functionals" can be obtained from the above specific form. For example, this wide variety includes quantities such as a mean exit time and even, for any fixed $t \geq 0$, the distribution of X_t (that is, $\boldsymbol{E}_x[\mathbb{1}_F(X_t)]$ where F is a measurable set).

There are surprisingly few works in the literature devoted to the actual computation of such expectations, using other means than direct Monte Carlo simulations. Davis showed that these expectations satisfy integrodifferential equations. However, the set of partial differential equations that is obtained is unusual. Roughly speaking, these differential equations are basically transport equations with a nonconstant velocity and they are coupled by the boundary conditions and by some integral terms involving kernels that are derived from the properties of the underlying stochastic process. The main difficulty comes from the fact that the domains on which the equations have to be solved vary from one equation to another making their numerical resolution highly problem specific. Another similar approach has been recently investigated in [4; 7]. It is based on a discretization of the Chapman Kolmogorov equations satisfied by the distribution of the process $(X_t)_{t\geq 0}$. The

authors propose an approximation of such expectations based on finite volume methods. Unfortunately, their method is only valid if there are no jumps at the boundary. Our approach is completely different and does not rely on differential equations, but on the fact that such expectations can be computed by iterating an integral operator G. This operator only involves the embedded Markov chain $(Z_n, S_n)_{n\in\mathbb{N}}$ and conditional expectations. It is therefore natural to propose a computational method based on the quantization of this Markov chain, following the same idea as [6].

There exists an extensive literature on quantization methods for random variables and processes. The interested reader may for instance consult [8], [9] and the references within. Quantization methods have been developed recently in numerical probability or optimal stochastic control with applications in finance (see [1; 2; 9], for instance). The quantization of a random variable X consists in finding a finite grid such that the projection $\widehat{X}$ of X on this grid minimizes some L^p norm of the difference $X-\widehat{X}$. Roughly speaking, such a grid will have more points in the areas of high density of X. As explained for instance in [9, Section 3], under some Lipschitz-continuity conditions, bounds for the rate of convergence of functionals of the quantized process towards the original process are available.

In the present work, we develop a numerical method to compute expectations of functionals of the above form where the cost functions l and c satisfy some Lipschitz-continuity conditions. We first recall the results presented by Davis according to whom, the above expectation may be computed by iterating an operator denoted by G. Consequently, it appears natural to follow the idea developed in [6] namely to express the operator G in terms of the underlying discrete-time Markov chain $(Z_n, S_n)_{n\in\mathbb{N}}$ and to replace it by its quantized approximation. Moreover, in order to prove the convergence of our algorithm, we replace the indicator function $\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}$ contained within the functional by some Lipschitz continuous approximation. Bounds for the rate of convergence are then obtained. However, and this is the main contribution of this paper, we then tackle two important aspects that had not been investigated in [6].

The first aspect consists in allowing c and l to be time-dependent functions, although still Lipschitz continuous, so that we may compute expectations of the form

$$\boldsymbol{E}_x\left[\int_0^{T_N} l(X_t,t)\,dt+\sum_{j=1}^{N} c(X_{T_j^-},T_j)\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}\right].$$

This important generalization has huge applicative consequences. For instance, it allows discounted cost or reward functions such as $l(x,t)=e^{-\delta t}l(x)$ and $c(x,t)=e^{-\delta t}c(x)$ where δ is some interest rate. To compute the above expectation, our strategy consists in considering, as suggested by Davis in [5], the time-augmented

process $\widetilde{X}_t = (X_t, t)$. Therefore, a natural way to deal with the time-dependent problem is to apply our previous approximation scheme to the time-augmented process $(\widetilde{X}_t)_{t\geq 0}$. However, it is far from obvious, that the assumptions required by our numerical method still hold for this new PDMP $(\widetilde{X}_t)_{t\geq 0}$.

The second important generalization is to consider the deterministic time horizon problem. Indeed, it seems crucial, regarding the applications, to be able to approximate

$$\boldsymbol{E}_x\left[\int_0^{t_f} l(X_t,t)\,dt + \sum_{T_f\leq t_f} c(X_{T_f^-},T_f)\mathbb{1}_{\{X_{T_f^-}\in\partial E\}}\right]$$

for some fixed $t_f > 0$ regardless of how many jumps occur before this deterministic time. To compute this quantity, we start by choosing a time N such that $\boldsymbol{P}(T_N < t_f)$ be small so that the previous expectation boils down to

$$\boldsymbol{E}_x\left[\int_0^{T_N} l(X_t,t)\mathbb{1}_{\{t\leq t_f\}}dt + \sum_{j=1}^{N} c(X_{T_j^-},T_j)\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}\mathbb{1}_{\{T_j\leq t_f\}}\right].$$

At first sight, this functional seems to be of the previous form. Yet, one must recall that Lipschitz continuity conditions have been made concerning the cost functions so that the indicator functions $\mathbb{1}_{\{\cdot\leq t_f\}}$ prevent a direct application of the earlier results. We deal with the two indicator functions in two different ways. On the one hand, we prove that it is possible to relax the regularity condition on the running cost function so that our algorithm still converges in spite of the first indicator function. On the other hand, since the same reasoning cannot be applied to the indicator function within the boundary jump cost term, we bound it between two Lipschitz continuous functions. This provides bounds for the expectation of the deterministic time horizon functional.

An important advantage of our method is that it is flexible. Indeed, as pointed out in [1], a quantization based method is "obstacle free" which means, in our case, that it produces, once and for all, a discretization of the process independently of the functions l and c since the quantization grids merely depend on the dynamics of the process. They are only computed once, stored off-line and may therefore serve many purposes. Once they have been obtained, we are able to approximate very easily and quickly any of the expectations described earlier. This flexibility is definitely an important advantage of our scheme over standard methods such as Monte Carlo simulations since, with such methods, we would have to run the whole algorithm for each expectation we want to compute. This point is illustrated in Section 6 where we easily solve an optimization problem that would be very laboriously handled by Monte Carlo simulations.

The paper is organized as follows. We first recall, in Section 2, the definition of a PDMP and state our assumptions. In Section 3, we introduce the recursive method to compute the expectation. Section 4 presents the approximation scheme and a bound for the rate of convergence. The main contribution of the paper lies in Section 5, which contains generalizations to time-dependent parameters and deterministic time-horizon problems. The paper is illustrated by a numerical example in Section 6; a conclusion (Section 7) is followed by some appendixes containing technical results.

2. Definitions and assumptions

For all metric space E, we denote by $\mathcal{B}(E)$ its Borel σ-field and $B(E)$ the set of real-valued, bounded and measurable functions defined on E. For $a, b \in \mathbb{R}$, set $a \wedge b = \min(a, b)$, $a \vee b = \max(a, b)$, and $a^+ = a \vee 0$.

Definition of a PDMP. In this first section, let us define a piecewise deterministic Markov process and introduce some general assumptions. Let M be a finite set called the set of the modes that will represent the different regimes of evolution of the PDMP. For each $m \in M$, the process evolves in E_m, an open subset of $\mathbb{R}^d$. Let

$$E = \{(m, \zeta), m \in M, \zeta \in E_m\}.$$

This is the state space of the process $(X_t)_{t\in\mathbb{R}^+} = (m_t, \zeta_t)_{t\in\mathbb{R}^+}$. Let ∂E be its boundary and $\overline{E}$ its closure and for any subset Y of E, Y^c denotes its complement.

A PDMP is defined by its local characteristics $(\Phi_m, \lambda_m, Q_m)_{m\in M}$.

- For each $m \in M$, $\Phi_m : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is a continuous function called the flow in mode m. For all $t \in \mathbb{R}$, $\Phi_m(\cdot, t)$ is an homeomorphism and $t \to \Phi_m(\cdot, t)$ is a semigroup; i.e., for all $\zeta \in \mathbb{R}^d$, $\Phi_m(\zeta, t + s) = \Phi_m(\Phi_m(\zeta, s), t)$. For all $x = (m, \zeta) \in E$, define the deterministic exit time from E:

  $$t^*(x) = \inf\{t > 0 \text{ such that } \Phi_m(\zeta, t) \in \partial E_m\}.$$

  We use here and throughout the convention $\inf \varnothing = +\infty$.

- For all $m \in M$, the jump rate $\lambda_m : \overline{E}_m \to \mathbb{R}^+$ is measurable, and for all $(m, \zeta) \in E$, there exists $\epsilon > 0$ such that

  $$\int_0^\epsilon \lambda_m(\Phi_m(\zeta, t))\, dt < +\infty.$$

- For all $m \in M$, Q_m is a Markov kernel on $(\mathcal{B}(\overline{E}), \overline{E}_m)$ that satisfies

  $$Q_m(\zeta, \{(m, \zeta)\}^c) = 1 \quad \text{for all } \zeta \in \overline{E}_m.$$

From these characteristics, it can be shown (see [5]) that there exists a filtered probability space $(\Omega, \mathscr{F}, \mathscr{F}_t, (\boldsymbol{P}_x)_{x\in E})$ on which a process $(X_t)_{t\in\mathbb{R}^+}$ is defined. Its motion, starting from a point $x \in E$, may be constructed as follows. Let T_1 be a nonnegative random variable with survival function

$$\boldsymbol{P}_x(T_1 > t) = \begin{cases} e^{-\Lambda(x,t)} & \text{if } 0 \le t < t^*(x), \\ 0 & \text{if } t \ge t^*(x), \end{cases}$$

where for $x = (m, \zeta) \in E$ and $t \in [0, t^*(x)]$,

$$\Lambda(x,t) = \int_0^t \lambda_m(\Phi_m(\zeta, s))\, ds.$$

One then chooses an E-valued random variable Z_1 according to the distribution $Q_m(\Phi_m(\zeta, T_1), \cdot)$. The trajectory of X_t for $t \le T_1$ is

$$X_t = \begin{cases} (m, \Phi_m(\zeta, t)) & \text{if } t < T_1, \\ Z_1 & \text{if } t = T_1. \end{cases}$$

Starting from the point $X_{T_1} = Z_1$, one then selects in a similar way $S_2 = T_2 - T_1$ the time between T_1 and the next jump, Z_2 the next postjump location and so on. Davis shows, in [5], that the process so defined is a strong Markov process $(X_t)_{t\ge 0}$ with jump times $(T_n)_{n\in\mathbb{N}}$ (with $T_0 = 0$). The process $(\Theta_n)_{n\in\mathbb{N}} = (Z_n, S_n)_{n\in\mathbb{N}}$ where $Z_n = X_{T_n}$ is the postjump location and $S_n = T_n - T_{n-1}$ (with $S_0 = 0$) is the n-th interjump time is clearly a discrete-time Markov chain.

The following assumption about the jump-times is standard (see [5, Section 24], for example):

Assumption 2.1. For all $(x,t) \in E \times \mathbb{R}^+$, $\boldsymbol{E}_x\left[\sum_k \mathbb{1}_{\{T_k<t\}}\right] < +\infty$.

It implies in particular that T_k goes to infinity a.s. when k goes to infinity.

Notation and assumptions. For notational convenience, any function h defined on E will be identified with its component functions h_m defined on E_m. Thus, one may write

$$h(x) = h_m(\zeta) \text{ when } x = (m, \zeta) \in E.$$

We also define a generalized flow $\Phi : E \times \mathbb{R}^+ \to E$ such that

$$\Phi(x,t) = (m, \Phi_m(\zeta, t)) \text{ when } x = (m, \zeta) \in E.$$

Define on E the following distance, for $x = (m, \zeta)$ and $x' = (m', \zeta') \in E$:

$$|x - x'| = \begin{cases} +\infty & \text{if } m \ne m', \\ |\zeta - \zeta'| & \text{otherwise.} \end{cases} \tag{1}$$

For any function w in $B(\overline{E})$, introduce the notation

$$Qw(x) = \int_E w(y)Q(x,dy), \qquad C_w = \sup_{x\in\overline{E}} |w(x)|,$$

and for any Lipschitz continuous function w in $B(E)$, denote by $[w]^E$, or if there is no ambiguity by $[w]$, its Lipschitz constant:

$$[w]^E = \sup_{x\neq y\in E} \frac{|w(x)-w(y)|}{|x-y|},$$

with the convention $\dfrac{1}{\infty} = 0$.

Remark 2.2. For $w \in B(\overline{E})$ and from the definition of the distance on E, one has $[w] = \max_{m\in M}[w_m]$.

Definition 2.3. Denote by $\boldsymbol{L}_c(E)$ the set of functions $w \in B(E)$ that are Lipschitz continuous along the flow; i.e., the real-valued, bounded, measurable functions defined on E and satisfying the following conditions:

- For all $x \in E$, the map $w(\Phi(x,\cdot)) : [0, t^*(x)) \to \mathbb{R}$ is continuous, and the limit $\lim_{t\to t^*(x)} w(\Phi(x,t))$ exists and is denoted by $w\big(\Phi(x,t^*(x))\big)$.
- There exists $[w]_1^E \in \mathbb{R}^+$ such that for all $x, y \in E$ and $t \in [0, t^*(x) \wedge t^*(y)]$, one has
$$\big|w(\Phi(x,t)) - w(\Phi(y,t))\big| \leq [w]_1^E |x-y|.$$
- There exists $[w]_2^E \in \mathbb{R}^+$ such that for all $x \in E$ and $t, u \in [0, t^*(x)]$, one has
$$\big|w(\Phi(x,t)) - w(\Phi(x,u))\big| \leq [w]_2^E |t-u|.$$
- There exists $[w]_*^E \in \mathbb{R}^+$ such that for all $x, y \in E$, one has
$$\big|w(\Phi(x,t^*(x))) - w(\Phi(y,t^*(y)))\big| \leq [w]_*^E |x-y|.$$

Denote by $\boldsymbol{L}_c(\partial E)$ the set of real-valued, bounded, measurable functions defined on ∂E satisfying the following condition:

- There exists $[w]_*^{\partial E} \in \mathbb{R}^+$ such that for all $x, y \in E$, one has
$$\big|w(\Phi(x,t^*(x))) - w(\Phi(y,t^*(y)))\big| \leq [w]_*^{\partial E} |x-y|.$$

Remark 2.4. When there is no ambiguity, we will use the notation $[w]_i$ instead of $[w]_i^E$ for $i \in \{1, 2, *\}$ and $[w]_*$ instead of $[w]_*^{\partial E}$.

Remark 2.5. In Definition 2.3, we used the generalized flow for notational convenience. For instance, the definition of $[w]_1$ is equivalent to the following: for all

$m \in M$, there exists $[w_m]_1 \in \mathbb{R}^+$ such that for all $\zeta, \zeta' \in E_m$ and $t \in [0, t^*(m,\zeta) \wedge t^*(m,\zeta')]$, one has

$$\left|w_m(\Phi_m(\zeta,t)) - w_m(\Phi_m(\zeta',t))\right| \leq [w_m]_1 |\zeta - \zeta'|.$$

Let $[w]_1 = \max_{m\in M}[w_m]_1$.

Definition 2.6. For all $u \geq 0$, denote by $\boldsymbol{L}_c^u(E)$ the set of functions $w \in B(E)$ Lipschitz continuous along the flow until time u; i.e., the real-valued, bounded, measurable functions defined on E and satisfying the following conditions:

- For all $x \in E$, the map $w(\Phi(x,\cdot)) : [0, t^*(x) \wedge u) \to \mathbb{R}$ is continuous. If $t^*(x) \leq u$, then $\lim_{t\to t^*(x)} w(\Phi(x,t))$ exists and is denoted by $w\big(\Phi(x,t^*(x))\big)$.
- There exists $[w]_1^{E,u} \in \mathbb{R}^+$ such that for all $x, y \in E$ and $t \in [0, t^*(x) \wedge t^*(y) \wedge u]$, one has
$$\left|w(\Phi(x,t)) - w(\Phi(y,t))\right| \leq [w]_1^{E,u} |x - y|.$$
- There exists $[w]_2^{E,u} \in \mathbb{R}^+$ such that for all $x \in E$ and $t, t' \in [0, t^*(x) \wedge u]$, one has
$$\left|w(\Phi(x,t)) - w(\Phi(x,t'))\right| \leq [w]_2^{E,u} |t - t'|.$$
- There exists $[w]_*^{E,u} \in \mathbb{R}^+$ such that for all $x, y \in E$, if $t^*(x) \leq u$ and $t^*(y) \leq u$, one has
$$\left|w(\Phi(x,t^*(x))) - w(\Phi(y,t^*(y)))\right| \leq [w]_*^{E,u} |x - y|.$$

Remark 2.7. For all $u \leq u'$, one has $\boldsymbol{L}_c^{u'}(E) \subset \boldsymbol{L}_c^u(E)$ with $[w]_i^{E,u} \leq [w]_i^{E,u'}$ where $i \in \{1, 2, *\}$.

Remark 2.8. Definitions 2.3 and 2.6 correspond respectively to the Lipschitz and local Lipschitz continuity along the flow that is, along the trajectories of the process. They can be replaced by (local) Lipschitz assumptions on the flow Φ, t^* and w in the classical sense.

We will require the following assumptions.

Assumption 2.9. The jump rate λ is bounded and there exists $[\lambda]_1 \in \mathbb{R}^+$ such that for all $x, y \in E$ and $t \in [0, t^*(x) \wedge t^*(y)]$, one has

$$\left|\lambda(\Phi(x,t)) - \lambda(\Phi(y,t))\right| \leq [\lambda]_1 |x - y|.$$

Assumption 2.10. The deterministic exit time from E, denoted by t^*, is assumed to be bounded and Lipschitz continuous on E.

Remark 2.11. Since the deterministic exit time t^* is bounded by C_{t^*}, one may notice that $\boldsymbol{L}_c^u(E)$ for $u \geq C_{t^*}$ is no other than $\boldsymbol{L}_c(E)$.

Remark 2.12. In most practical applications, the physical properties of the system ensure that either t^* is bounded, or the problem has a natural finite deterministic time horizon t_f. In the latter case, there is no loss of generality in considering that t^* is bounded by this deterministic time horizon. This leads to replacing C_{t^*} by t_f. An example of such a situation is presented in an industrial example in Section 6.2.

Assumption 2.13. The Markov kernel Q is Lipschitz in the following sense: there exists $[Q] \in \mathbb{R}^+$ such that for all $u \geq 0$ and for all function $w \in \boldsymbol{L}_c^u(E)$, one has

(1) for all $x, y \in E$ and $t \in [0, t^*(x) \wedge t^*(y) \wedge u)$,

$$\big|Qw(\Phi(x,t)) - Qw(\Phi(y,t))\big| \leq [Q][w]_1^{E,u}|x-y|.$$

(2) for all $x, y \in E$ such that $t^*(x) \vee t^*(y) \leq u$,

$$\big|Qw(\Phi(x,t^*(x))) - Qw(\Phi(y,t^*(y)))\big| \leq [Q]\big([w]_*^{E,u} + [w]_1^{E,u}\big)|x-y|.$$

Remark 2.14. Assumption 2.13 is slightly more restrictive that its counterpart in [6] (Assumption 2.5), because of the introduction of the state space $\boldsymbol{L}_c^u(E)$. This is to ensure that the time-augmented process still satisfies a similar assumption; see Section 5.1.

3. Expectation

From now on, we will assume that $Z_0 = x$ a.s. for some $x \in E$. For all fixed $N \in \mathbb{N}^*$, we intend to numerically approximate the quantity

$$J_N(l,c)(x) = \boldsymbol{E}_x\left[\int_0^{T_N} l(X_t)\,dt + \sum_{j=1}^{N} c(X_{T_j^-})\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}\right], \tag{2}$$

where $l \in B(E)$, $c \in B(\partial E)$ and X_{t-} is the left limit of X_t. Thus, $X_{T_j^-}$ is the j-th prejump location. Since the boundary jumps occur exactly at the deterministic exit times from E, one has,

$$J_N(l,c)(x) = \boldsymbol{E}_x\left[\int_0^{T_N} l(X_t)\,dt + \sum_{j=1}^{N} c\big(\Phi(Z_{j-1}, t^*(Z_{j-1}))\big)\mathbb{1}_{\{S_j = t^*(Z_{j-1})\}}\right].$$

In many applications, $J_N(l,c)(x)$ appears as a cost or a reward function. The first term, that depends on l, is called the running cost and the second one, that depends on c, is the boundary jump cost.

The rest of this section is devoted to formulating the expectation above in a way that will allow us to derive a numerical computation method. The Lipschitz continuity property will be a crucial point when it comes to proving the convergence of our approximation scheme. For this reason, the first step of our approximation is to replace the indicator function in $J_N(l,c)(x)$ by a Lipschitz continuous

function. Then we present a recursive method yielding the required expectation. This recursive formulation will be the basis of our numerical method.

3.1. ***Lipschitz continuity.*** We introduce a regularity assumption on l and c.

Assumption 3.1. We assume that $l \in \boldsymbol{L}_c(E)$ and $c \in \boldsymbol{L}_c(\partial E)$.

Moreover, we replace the indicator function in $J_N(l,c)(x)$ by a Lipschitz continuous function δ^A, with $A > 0$. Let then

$$J_N^A(l,c)(x) = \boldsymbol{E}_x\left[\int_0^{T_N} l(X_t)\,dt + \sum_{j=1}^{N} c\big(\Phi(Z_{j-1}, t^*(Z_{j-1}))\big)\delta^A(Z_{j-1}, S_j)\right],$$

where δ^A is a triangular approximation of the indicator function. It is defined on $E \times \mathbb{R}$ by

$$\delta^A(x,t) = \begin{cases} A\Big(t - \Big(t^*(x) - \dfrac{1}{A}\Big)\Big) & \text{for } t \in \Big[t^*(x) - \dfrac{1}{A};\ t^*(x)\Big], \\ -A\Big(t - \Big(t^*(x) + \dfrac{1}{A}\Big)\Big) & \text{for } t \in \Big[t^*(x);\ t^*(x) + \dfrac{1}{A}\Big], \\ 0 & \text{otherwise.} \end{cases}$$

For all $x \in E$, the function $\delta^A(x,t)$ goes to $\mathbb{1}_{\{t=t^*(x)\}}$ when A goes to infinity. The following proposition proves the convergence of $J_N^A(l,c)(x)$ towards $J_N(l,c)(x)$ with an error bound.

Proposition 3.2. *For all $x \in E$, $A > 0$, $N \in \mathbb{N}^*$, $l \in \boldsymbol{L}_c(E)$ and $c \in \boldsymbol{L}_c(\partial E)$, one has*

$$\big|J_N^A(l,c)(x) - J_N(l,c)(x)\big| \le \frac{NC_cC_\lambda}{A}.$$

Proof. For all $x \in E$, one has

$$\begin{aligned} &\big|J_N^A(l,c)(x) - J_N(l,c)(x)\big| \\ &\quad = \left|\boldsymbol{E}_x\left[\sum_{j=1}^{N} c\big(\Phi(Z_{j-1}, t^*(Z_{j-1}))\big)\big(\delta^A(Z_{j-1}, S_j) - \mathbb{1}_{\{S_j = t^*(Z_{j-1})\}}\big)\right]\right| \\ &\quad \le C_c \sum_{j=1}^{N} \boldsymbol{E}_x\big[|\delta^A(Z_{j-1}, S_j) - \mathbb{1}_{\{S_j = t^*(Z_{j-1})\}}|\big] \\ &\quad \le C_c \sum_{j=1}^{N} \boldsymbol{E}_x\big[\boldsymbol{E}\big[|\delta^A(Z_{j-1}, S_j) - \mathbb{1}_{\{S_j = t^*(Z_{j-1})\}}| \;\big|\; Z_{j-1}\big]\big]. \end{aligned}$$

We recall that the conditional law of S_j with respect to Z_{j-1} has density

$$s \to \lambda\big(\Phi(Z_{j-1}, s)\big)e^{-\Lambda(Z_{j-1}, s)}$$

on $[0; t^*(Z_{j-1}))$ and puts the weight $e^{-\Lambda(Z_{j-1},t^*(Z_{j-1}))}$ on the point $t^*(Z_{j-1})$. We also recall that λ is bounded thanks to Assumption 2.9. Finally, one has

$$\begin{aligned}
&\left|J_N^A(l,c)(x) - J_N(l,c)(x)\right| \\
&\qquad \le C_c \sum_{j=1}^{N} \boldsymbol{E}_x\left[\int_{t^*(Z_{j-1})-\frac{1}{A}}^{t^*(Z_{j-1})} \delta^A(Z_{j-1},s)\lambda\big(\Phi(Z_{j-1},s)\big)e^{-\Lambda(Z_{j-1},s)}\,ds\right] \\
&\qquad \le \frac{NC_cC_\lambda}{A}.
\end{aligned}$$

Hence the result. □

Consequently to this proposition, we consider, from now on, the approximation of $J_N^A(l,c)(x)$ for some fixed A, large enough to ensure that the previous error is as small as required. The suitable choice of A will be discussed in Section 4.2.

3.2. ***Recursive formulation.*** Davis shows in [5, Section 32] that the expectation $J_N^A(l,c)(x)$ we are interested in is obtained by merely iterating an operator that we will denote by G. The rest of this section is dedicated to presenting this method from which we will derive our approximation scheme in Section 4.

Definition 3.3. Introduce the functions L, C and F defined for all $x \in E$ and $t \in [0; t^*(x)]$ by

$$\begin{aligned}
L(x,t) &= \int_0^t l\big(\Phi(x,s)\big)\,ds, \\
C(x,t) &= c\big(\Phi(x,t^*(x))\big)\delta^A(x,t), \\
F(x,t) &= L(x,t) + C(x,t),
\end{aligned}$$

along with the operator $G\colon B(E) \to B(E)$ given by

$$Gw(x) = \boldsymbol{E}_x\big[F(x,S_1) + w(Z_1)\big].$$

Definition 3.4. Define the sequence of functions $(v_k)_{0\le k\le N}$ in $B(E)$ by

$$v_N(x) = 0, \quad v_k(x) = Gv_{k+1}(x).$$

Davis then shows in [5, Equation 32.33] that, for all $k \in \{0,\dots,N\}$,

$$v_{N-k}(x) = \boldsymbol{E}_x\left[\int_0^{T_k} l(X_t)\,dt + \sum_{j=1}^{k} c\big(\Phi(Z_{j-1},t^*(Z_{j-1}))\big)\delta^A(Z_{j-1},S_j)\right].$$

Thus, the quantity $J_N^A(l,c)(x)$ we intend to approximate is none other than $v_0(x)$.

Notice that, thanks to the Markov property of the chain $(Z_n, S_n)_{n\in\mathbb{N}}$, one has for all $k \in \{0, \dots, N-1\}$,

$$Gw(x) = \boldsymbol{E}\big[F(Z_k, S_{k+1}) + w(Z_{k+1})\big|Z_k = x\big]. \tag{3}$$

Hence, for all $k \in \{0, \dots, N\}$, let $V_k = v_k(Z_k)$ so that one has

$$V_N = 0, \quad V_k = \boldsymbol{E}\big[F(Z_k, S_{k+1}) + V_{k+1}\big|Z_k\big].$$

This backward recursion provides the required quantity

$$V_0 = J_N^A(l, c)(x).$$

Hence, we need to approximate the sequence of random variables $(V_k)_{0\le k\le N}$. This sequence satisfies a recursion that only depends on the chain $(Z_k, S_k)_{0\le k\le N}$. Therefore, it appears natural to propose an approximation scheme based on a discretization of this chain $(Z_k, S_k)_{0\le k\le N}$, called quantization, similarly to the ideas developed in [6] and [3].

4. Approximation scheme

Let us now turn to the approximation scheme itself. We explained in the previous section how the expectation we are interested in stems from the iteration of the operator G that only depends on the discrete-time Markov chain $(Z_k, S_k)_{0\le k\le N}$. The first step of our numerical method is therefore to discretize this chain in order to approximate the operator G.

4.1. ***Quantization of the chain*** $(Z_k, S_k)_{k\le N}$**.** Our approximation method is based on the quantization of the underlying discrete time Markov chain $(\Theta_k)_{k\le N} = (Z_k, S_k)_{k\le N}$. This quantization consists in finding an optimally designed discretization of the process to provide for each step k the best possible approximation of Θ_k by a random variable $\widehat{\Theta}_k$ which state space has a finite and fixed number of points. Here, *optimal* means that the distance between Θ_k and $\widehat{\Theta}_k$ in a suitably chosen L^p norm is minimal. For details on the quantization methods, we mainly refer to [9] but the interested reader can also consult [1], [2] and the references therein.

More precisely, consider X an $\mathbb{R}^q$-valued random variable such that $\|X\|_p < \infty$ and let K be a fixed integer. The optimal L_p-quantization of the random variable X consists in finding the best possible L_p-approximation of X by a random vector $\widehat{X} \in \{x^1, \dots, x^K\}$ taking at most K values: This procedure consists of two steps:

(1) Find a finite weighted grid $\Gamma \subset \mathbb{R}^q$ with $\Gamma = \{x^1, \dots, x^K\}$.

(2) Set $\widehat{X} = \widehat{X}^\Gamma$ where $\widehat{X}^\Gamma = \mathrm{proj}_\Gamma(X)$ with proj_Γ denotes the closest neighbor projection on Γ.

The asymptotic properties of the L_p-quantization are given by the following result; see [9], for example.

Theorem 4.1. *If* $\mathbb{E}[|X|^{p+\eta}] < +\infty$ *for some* $\eta > 0$ *then one has*

$$\lim_{K\to\infty} K^{p/q} \min_{|\Gamma|\leq K} \|X - \widehat{X}^{\Gamma}\|_p^p = J_{p,q} \int |h|^{q/(q+p)}(u)\, du,$$

where the law of X *is* $P_X(du) = h(u)\lambda_q(du) + \nu$ *with* $\nu \perp \lambda_q$, $J_{p,q}$ *a constant and* λ_q *the Lebesgue measure in* $\mathbb{R}^q$.

Remark that X needs to have finite moments up to the order $p + \eta$ to ensure the above convergence. In this work, we used the CLVQ quantization algorithm described in [1], Section 3.

There exists a similar procedure for the optimal quantization of a Markov chain $\{X_k\}_{k\in\mathbb{N}}$. There are two approaches to provide the quantized approximation of a Markov chain. The first one, based on the quantization at each time k of the random variable X_k is called the *marginal quantization*. The second one that enhances the preservation of the Markov property is called *Markovian quantization*. Remark that for the latter, the quantized Markov process is not homogeneous. These two methods are described in details in [9, Section 3]. In this work, we used the marginal quantization approach for simplicity reasons.

The quantization algorithm provides for each time step $0 \leq k \leq N$ a finite grid Γ_k of $E \times \mathbb{R}^+$ as well as the transition matrices $(\widehat{Q}_k)_{0\leq k\leq N-1}$ from Γ_k to Γ_{k+1}. Let $p \geq 1$ such that for all $k \leq N$, Z_k and S_k have finite moments at least up to order p and let proj_{Γ_k} be the closest-neighbor projection from $E \times \mathbb{R}^+$ onto Γ_k (for the distance associated to norm p). The quantized process $(\widehat{\Theta}_k)_{k\leq N} = (\widehat{Z}_k, \widehat{S}_k)_{k\leq N}$ takes values for each k in the finite grid Γ_k of $E \times \mathbb{R}^+$ and is defined by

$$(\widehat{Z}_k, \widehat{S}_k) = \text{proj}_{\Gamma_k}(Z_k, S_k). \tag{4}$$

Moreover, we also denote by Γ_k^Z and Γ_k^S, respectively, the projections of Γ_k on E and $\mathbb{R}^+$.

Some important remarks must be made concerning the quantization. On the one hand, the optimal quantization has nice convergence properties stated by Theorem 4.1. Indeed, the L^p-quantization error $\|\Theta_k - \widehat{\Theta}_k\|_p$ goes to zero when the number of points in the grids goes to infinity. However, on the other hand, the Markov property is not maintained by the algorithm and the quantized process is generally not Markovian. Although the quantized process can be easily transformed into a Markov chain (see [9]), this chain will not be homogeneous. It must be pointed out that the quantized process $(\widehat{\Theta}_k)_{k\in\mathbb{N}}$ depends on the starting point Θ_0 of the process.

In practice, we begin with the computation of the quantization grids which merely requires to be able to simulate the process. This step is quite time-consuming, especially when the number of points in the quantization grids is large. However, the grids are only computed once and for all and may be stored off-line. What is more, they only depend on the dynamics of the process, not on the cost functions l and c. Hence, the same grids may be used to compute different expectations of functionals as long as they are related to the same process. Our schemes are then based on the following simple idea: we replace the process by its quantized approximation within the operator G. The approximation is thus obtained in a very simple way since the quantized process has finite state space.

4.2. ***Approximation of the expectation and rate of convergence.*** We now use the quantization of the process $(\Theta_k)_{k\leq N} = (Z_k, S_k)_{k\leq N}$. In order to approximate the random variables $(V_k)_{k\leq N}$, we introduce a quantized version of the operator G. Notice that the quantized process is no longer an homogeneous Markov chain so that we have different operators for each time step k. Their definitions naturally stem from a remark made in the previous section: recall that for all $k \in \{1, \ldots, N\}$ and $x \in E$,

$$Gw(x) = \boldsymbol{E}\big[F(Z_{k-1}, S_k) + w(Z_k) \,\big|\, Z_{k-1} = x\big].$$

Definition 4.2. For all $k \in \{1, \ldots, N\}$, $w \in B(\Gamma_k^Z)$ and $z \in \Gamma_{k-1}^Z$, let

$$\widehat{G}_k w(z) = \boldsymbol{E}\big[F(z, \widehat{S}_k) + w(\widehat{Z}_k) \,\big|\, \widehat{Z}_{k-1} = z\big].$$

Introduce also the functions $(\hat{v}_k)_{0\leq k\leq N}$ by

$$\begin{aligned} \hat{v}_N(z) &= 0 && \text{for all } z \in \Gamma_N^Z, \\ \hat{v}_k(z) &= \widehat{G}_{k+1}\hat{v}_{k+1}(z) && \text{for all } k \in \{0, \ldots, N-1\} \text{ and } z \in \Gamma_k^Z. \end{aligned}$$

Finally, for all $k \in \{0, \ldots, N\}$, let

$$\widehat{V}_k = \hat{v}_k(\widehat{Z}_k).$$

Remark 4.3. The conditional expectation in $\widehat{G}_k w(z)$ is a finite sum. Thus, the numerical computation of the sequence $(\widehat{V}_k)_k$ will be easily performed as soon as the quantized process $(\widehat{\Theta}_k)_{k\leq N}$ has been obtained.

Remark 4.4. We have assumed that $Z_0 = x$ a.s. Thus, the quantization algorithm provides that $\widehat{Z}_0 = x$ a.s. too. Consequently, the random variable $\widehat{V}_0 = \hat{v}_0(\widehat{Z}_0)$ is, in fact, deterministic.

The following theorem states the convergence of $\widehat{V}_0$ towards $V_0 = J_N^A(l, c)(x)$ and provides a bound for the rate of convergence.

Theorem 4.5. *For all $k \in \{0, \dots, N\}$, one has $v_k \in \boldsymbol{L}_c(E)$. Moreover, the approximation error satisfies*

$$|J_N(l,c)(x) - \widehat{V}_0| \leq \varepsilon_N(l,c,X,A),$$

where

$$\varepsilon_N(l,c,X,A) = \sum_{k=0}^{N-1} \Big(2[v_{k+1}]\|Z_{k+1} - \widehat{Z}_{k+1}\|_p + \big(2[v_k] + [F]_1\big)\|Z_k - \widehat{Z}_k\|_p + [F]_2\|S_{k+1} - \widehat{S}_{k+1}\|_p\Big) + \frac{NC_cC_\lambda}{A}$$

with

$$\begin{aligned}
[F]_1 &= C_{t^*}[l]_1 + [c]_* + A[t]_*C_c,\\
[F]_2 &= C_l + AC_c.\\
C_{v_n} &\leq n\big(C_{t^*}C_l + C_c\big),\\
[v_n]_1 &\leq e^{C_{t^*}C_\lambda}\big(K(A, v_{n-1}) + nC_{t^*}[\lambda]_1(C_{t^*}C_l + C_c)\big) + C_{t^*}[l]_1,\\
[v_n]_2 &\leq e^{C_{t^*}C_\lambda}\big(C_{t^*}C_lC_\lambda + 2C_l + C_\lambda C_c + (2n-1)C_\lambda(C_{t^*}C_l + C_c)\big) + C_l,\\
[v_n]_* &\leq [v_n]_1 + [t^*][v_n]_2,\\
[v_n] &\leq K(A, v_{n-1}),
\end{aligned}$$

and for all $w \in \boldsymbol{L}_c(E)$, $K(A,w) = E_1 + E_2A + E_3[w]_1 + E_4C_w + [Q][w]_*$, where

$$\begin{aligned}
E_1 &= 2[l]_1C_{t^*} + C_l\big([t^*] + 2C_{t^*}^2[\lambda]_1\big) + [c]_*(1 + C_{t^*}C_\lambda)\\
&\quad + C_c\big(2[\lambda]_1C_{t^*} + C_\lambda C_{t^*}^2[\lambda]_1 + 2[t^*]C_\lambda\big),\\
E_2 &= C_cC_{t^*}C_\lambda[t^*],\\
E_3 &= (1 + C_{t^*}C_\lambda)[Q],\\
E_4 &= 2C_\lambda[t^*] + C_{t^*}[\lambda]_1(2 + C_{t^*}C_\lambda).
\end{aligned}$$

The choice of A. Proposition 3.2 suggests that A should be as large as possible. However, the constants $[F]_1$, $[F]_2$ and $[v_n]$ that appear in the bound of the approximation error proposed by the above Theorem 4.5 grow linearly with A. Thus, in order to control this error, it is necessary that the order of magnitude of the quantization error $\|\Theta_k - \widehat{\Theta}_k\|_p$ be at most $1/A$.

The convergence of the approximation scheme can be derived from Theorem 4.5. Indeed, on the one hand, one must remind that $V_0 = J_N^A(l,c)(x)$ is the expectation we intended to approximate and on the other hand, $\|\Theta_k - \widehat{\Theta}_k\|_p$ may become arbitrarily small when the number of points in the quantization grids goes to infinity (see [9], for example). An outline of the proof is presented in Appendix C.

5. Time-dependent functionals

We now turn to the main contribution of this paper and present two generalizations of the previous problem. On the one hand, we will consider time-dependent functionals of the form

$$\boldsymbol{E}_x\left[\int_0^{T_N} l(X_t,t)\,dt + \sum_{j=1}^{N} c(X_{T_j^-},T_j)\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}\right]$$

where l and c are Lipschitz continuous functions. On the other hand, we wish to replace the random time horizon T_N by a deterministic one, denoted by t_f:

$$\boldsymbol{E}_x\left[\int_0^{t_f} l(X_t,t)\,dt + \sum_{T_j\le t_f} c(X_{T_j^-},T_j)\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}\right].$$

We will reason as follows. As suggested by Davis in [5], we will introduce a transformation $(\widetilde{X}_t)_{t\ge 0}$ of the initial process $(X_t)_{t\ge 0}$ by including the time variable into the state space: $(\widetilde{X}_t) = (X_t, t)$. Indeed, we will see that both the expectation of the time-dependent functional and the one with deterministic time horizon are no other than expectations of time invariant functionals for the time-augmented process $(\widetilde{X}_t)_{t\ge 0}$. We therefore intend to apply the previously exposed approximation scheme to this new PDMP. However, it is far from obvious that the Lipschitz continuity assumptions 2.9, 2.13 and 2.10 still hold for this new process.

Thus, the rest of this section is organized as follows. First, we recall the precise definition of the time-augmented process and prove that it satisfies the Lipschitz continuity assumptions required by our approximation scheme. Then, we will see that the time-dependent functional case corresponds to a time invariant functional for the new transformed process and may therefore be obtained thanks to the earlier method. Finally, we consider the deterministic time horizon problem that features an additional hurdle namely the presence of non-Lipschitz continuous indicator functions.

5.1. ***The time-augmented process.*** Davis suggests, in [5, Section 31], that the case of the time-dependent functionals may be treated by introducing the time variable within the state space. Thus, it will be possible to apply our previous numerical method to the time-augmented process. However, and this is what we discuss in this section, it is necessary to check whether the Lipschitz continuity assumptions still hold. We first recall the definition of the time-augmented process given by Davis.

Definition 5.1. Introduce the new state space

$$\widetilde{E} = E \times \mathbb{R}^+$$

equipped with the norm defined by: for all $\xi = (x,t)$, $\xi' = (x',t') \in \widetilde{E}$, let

$$|\xi - \xi'| = |x - x'| + |t - t'| \tag{5}$$

where the norm on E is given by (1). On this state space, we define the process

$$\widetilde{X}_t = (X_t, t).$$

The local characteristics of the PDMP $(\widetilde{X}_t)_{t\geq 0}$, denoted by $(\tilde{\lambda}, \widetilde{Q}, \widetilde{\Phi})$, are given for all $\xi = (x,t) \in \widetilde{E}$ by

$$\begin{cases} \tilde{\lambda}(\xi) = \lambda(x), \\ \widetilde{\Phi}(\xi, s) = \big(\Phi(x,s), t+s\big) & \text{for } s \leq t^*(x), \\ \widetilde{Q}\big(\xi, A \times \{t\}\big) = Q(x,A) & \text{for all } A \in \mathcal{B}(E). \end{cases}$$

Moreover, we naturally define for all $\xi = (x,t) \in \widetilde{E}$

$$\tilde{t}^*(\xi) = \inf\{s > 0 \text{ such that } \widetilde{\Phi}(\xi, s) \in \partial \widetilde{E}\} = t^*(x)$$

Clearly, Assumptions 2.9 and 2.10 still hold with $[\tilde{\lambda}]_1 = [\lambda]_1$ and $[\tilde{t^*}] = [t^*]$. However, proving Assumption 2.13 is more intricate. We start with the following lemma.

Lemma 5.2. *Let $u, t \geq 0$ and $w \in \boldsymbol{L}_c^u(\widetilde{E})$. Denote by w_t the function of $B(E)$ defined by $w_t = w(\cdot, t)$. One has then $w_t \in \boldsymbol{L}_c^{t\wedge u}(E)$ with*

$$\begin{aligned} [w_t]_1^{E,t\wedge u} &\leq [w]_1^{\widetilde{E},u}, \\ [w_t]_2^{E,t\wedge u} &\leq [w]_1^{\widetilde{E},u} + [w]_2^{\widetilde{E},u}, \\ [w_t]_*^{E,t\wedge u} &\leq (1 + [t^*])[w]_*^{\widetilde{E},u}. \end{aligned}$$

Proof. Let $u, t \geq 0$ and $w \in \boldsymbol{L}_c^u(\widetilde{E})$. For $x, x' \in E$ and $s \leq t^*(x) \wedge t^*(x') \wedge t \wedge u$, one has

$$\big|w_t(\Phi(x,s)) - w_t(\Phi(x',s))\big| = \big|w\big(\widetilde{\Phi}((x,t-s),s)\big) - w\big(\widetilde{\Phi}((x',t-s),s)\big)\big|.$$

We now use the fact that $w \in \boldsymbol{L}_c^u(\widetilde{E})$ which yields since $s \leq u$

$$\big|w_t(\Phi(x,s)) - w_t(\Phi(x',s))\big| \leq [w]_1^{\widetilde{E},u}\big|(x,t-s) - (x',t-s)\big| = [w]_1^{\widetilde{E},u}\big|x - x'\big|.$$

Hence, $[w_t]_1^{E,t\wedge u} \leq [w]_1^{\widetilde{E},u}$, and similarly one obtains $[w_t]_2^{E,t\wedge u} \leq [w]_1^{\widetilde{E},u} + [w]_2^{\widetilde{E},u}$.

On the other hand, for $x, x' \in E$ such that $t^*(x) \vee t^*(x') \leq t \wedge u$, one has

$$\begin{aligned}
&\left|w_t(\Phi(x,t^*(x))) - w_t(\Phi(x',t^*(x')))\right| \\
&\quad = \left|w\big(\widetilde{\Phi}((x,t-t^*(x)),t^*(x))\big) - w\big(\widetilde{\Phi}((x',t-t^*(x')),t^*(x'))\big)\right| \\
&\quad = \Big|w\big(\widetilde{\Phi}\big((x,t-t^*(x)),\tilde{t}^*(x,t-t^*(x))\big)\big) \\
&\qquad\qquad - w\big(\widetilde{\Phi}\big((x',t-t^*(x')),\tilde{t}^*(x',t-t^*(x'))\big)\big)\Big|;
\end{aligned}$$

moreover since $w \in \boldsymbol{L}_c^u(\widetilde{E})$ and $\tilde{t}^*(x,t-t^*(x)) \vee \tilde{t}^*(x',t-t^*(x')) \leq u$ one has

$$\left|w_t(\Phi(x,t^*(x))) - w_t(\Phi(x',t^*(x')))\right| \leq [w]_*^{\widetilde{E},u}\left|(x,t-t^*(x)) - (x',t-t^*(x'))\right|.$$

We conclude thanks to the Lipschitz continuity assumption 2.10 on t^*, which yields $\left|(x,t-t^*(x)) - (x',t-t^*(x'))\right| \leq (1+[t^*])|x-x'|$. One obtains $[w_t]_*^{E,t\wedge u} \leq [w]_*^{\widetilde{E},u}\big(1+[t^*]\big)$ and $w_t \in \boldsymbol{L}_c^{t\wedge u}(E)$. □

The next proposition proves that Assumption 2.13 holds for the time-augmented process $(\widetilde{X})_{t\geq 0}$.

Proposition 5.3. *Let $w \in \boldsymbol{L}_c^u(\widetilde{E})$.*

(1) *For all $\xi, \xi' \in \widetilde{E}$ and $s \in [0, \tilde{t}^*(\xi) \wedge \tilde{t}^*(\xi') \wedge u]$,*

$$\left|\widetilde{Q}w\big(\widetilde{\Phi}(\xi,s)\big) - \widetilde{Q}w\big(\widetilde{\Phi}(\xi',s)\big)\right| \leq ([Q]\vee 1)[w]_1^{\widetilde{E},u}|\xi-\xi'|.$$

(2) *For all $\xi, \xi' \in \widetilde{E}$ such that $\tilde{t}^*(\xi) \vee \tilde{t}^*(\xi') \leq u$,*

$$\begin{aligned}
&\left|\widetilde{Q}w\big(\widetilde{\Phi}(\xi,\tilde{t}^*(\xi))\big) - \widetilde{Q}w\big(\widetilde{\Phi}(\xi',\tilde{t}^*(\xi'))\big)\right| \\
&\qquad\qquad \leq ([Q]\vee 1)(1+[t^*])\big([w]_*^{\widetilde{E},u} + [w]_1^{\widetilde{E},u}\big)|\xi-\xi'|.
\end{aligned}$$

In other words, Assumption 2.13 is satisfied with $[\widetilde{Q}] = ([Q]\vee 1)(1+[t^*])$.

Proof. As in the previous lemma, for all $t \geq 0$, we will denote by w_t the function of $B(E)$ defined by $w_t = w(\cdot,t)$. For $\xi = (x,t) \in \widetilde{E}$ and $w \in \boldsymbol{L}_c^u(\widetilde{E})$, one has, by the definition of $\widetilde{Q}$,

$$\widetilde{Q}w(\xi) = \int_{\xi'\in\widetilde{E}} w(\xi')\widetilde{Q}\big((x,t),d\xi'\big) = \int_{z\in E} w(z,t)Q\big(x,dz\big) = Qw_t(x). \quad (6)$$

We may now check the regularity assumption on $\widetilde{Q}$. Let $\xi = (x,t)$ and $\xi' = (x',t') \in \widetilde{E}$. Let $s \in [0; \tilde{t}^*(\xi) \wedge \tilde{t}^*(\xi') \wedge u]$. Thanks to the definition of $\widetilde{\Phi}$ and (6) one has

$$\begin{aligned}
\left|\widetilde{Q}w(\widetilde{\Phi}(\xi,s)) - \widetilde{Q}w(\widetilde{\Phi}(\xi',s))\right| &= \left|\widetilde{Q}w(\Phi(x,s),t+s) - \widetilde{Q}w(\Phi(x',s),t'+s)\right| \\
&= \left|Qw_{t+s}(\Phi(x,s)) - Qw_{t'+s}(\Phi(x',s))\right|.
\end{aligned}$$

We split this into the sum of two differences:

$$\begin{aligned}&\big|Qw_{t+s}(\Phi(x,s))-Qw_{t'+s}(\Phi(x',s))\big|\\&\qquad\le\big|Qw_{t+s}(\Phi(x,s))-Qw_{t+s}(\Phi(x',s))\big|+\big|Q(w_{t+s}-w_{t'+s})(\Phi(x',s))\big|.\end{aligned}$$

On the one hand, we recall that thanks to Lemma 5.2, $w_{t+s}\in \boldsymbol{L}_c^{(t+s)\wedge u}(E)$, so that, since $s\le(t+s)\wedge u$, we may use the Lipschitz continuity assumption 2.13 on Q and the first term is bounded as follows:

$$\big|Qw_{t+s}(\Phi(x,s))-Qw_{t+s}(\Phi(x',s))\big|\le[Q][w_{t+s}]_1^{E,(t+s)\wedge u}|x-x'|.$$

Lemma 5.2 also provides $[w_{t+s}]_1^{E,(t+s)\wedge u}\le[w]_1^{\widetilde{E},u}$. On the other hand, and more basically, the second term in the equation above satisfies

$$\big|Q(w_{t+s}-w_{t'+s})(\Phi(x',s))\big|\le[w]_1^{\widetilde{E},u}|t-t'|.$$

We obtain

$$\big|\widetilde{Q}w\big(\widetilde{\Phi}(\xi,s)\big)-\widetilde{Q}w\big(\widetilde{\Phi}(\xi',s)\big)\big|\le([Q]\vee 1)[w]_1^{\widetilde{E},u}|\xi-\xi'|.$$

We reason similarly to bound $\big|\widetilde{Q}w\big(\widetilde{\Phi}(\xi,\tilde{t}^*(\xi))\big)-\widetilde{Q}w\big(\widetilde{\Phi}(\xi',\tilde{t}^*(\xi'))\big)\big|$, where $\xi=(x,t)$ and $\xi'=(x',t')\in\widetilde{E}$ are such that $\tilde{t}^*(\xi)\vee\tilde{t}^*(\xi')\le u$. Equation (6) yields

$$\begin{aligned}&\big|\widetilde{Q}w\big(\widetilde{\Phi}(\xi,\tilde{t}^*(\xi))\big)-\widetilde{Q}w\big(\widetilde{\Phi}(\xi',\tilde{t}^*(\xi'))\big)\big|\\&\qquad=\big|Qw_{t+t^*(x)}\big(\Phi(x,t^*(x))\big)-Qw_{t'+t^*(x')}\big(\Phi(x',t^*(x'))\big)\big|,\end{aligned}$$

which we now split as follows:

$$\begin{aligned}&\big|Qw_{t+t^*(x)}\big(\Phi(x,t^*(x))\big)-Qw_{t'+t^*(x')}\big(\Phi(x',t^*(x'))\big)\big|\\&\qquad\le\big|Qw_{t+t^*(x)}\big(\Phi(x,t^*(x))\big)-Qw_{t+t^*(x)}\big(\Phi(x',t^*(x'))\big)\big|\\&\qquad\qquad+\big|(Qw_{t+t^*(x)}-Qw_{t'+t^*(x')})\big(\Phi(x',t^*(x'))\big)\big|.\end{aligned}$$

Thanks to Lemma 5.2, $w_{t+t^*(x)}\in\boldsymbol{L}_c^{(t+t^*(x))\wedge u}(E)$. We assume, without loss of generality, that $t^*(x)\ge t^*(x')$, so $t^*(x)\vee t^*(x')\le(t+t^*(x))\wedge u$. Therefore, the first term in the above equation is bounded, thanks to the Lipschitz continuity assumption 2.13 on Q and Lemma 5.2, by

$$[Q]\big((1+[t^*])[w]_*^{\widetilde{E},u}+[w]_1^{\widetilde{E},u}\big)|x-x'|.$$

It is more straightfoward to obtain a bound for the second term, of the form

$$[w]_1^{\widetilde{E},u}\big|t-t'+t^*(x)-t^*(x')\big|\le[w]_1^{\widetilde{E},u}\big(|t-t'|+[t^*]|x-x'|\big).$$

We obtain

$$\begin{aligned}
&\left|\widetilde{Q}w\big(\widetilde{\Phi}(\xi,\tilde{t}^*(\xi))\big)-\widetilde{Q}w\big(\widetilde{\Phi}(\xi',\tilde{t}^*(\xi'))\big)\right| \\
&\quad\le [Q](1+[t^*])[w]_*^{\widetilde{E},u}|x-x'|+[w]_1^{\widetilde{E},u}\big([Q]|x-x'|+|t-t'|+[t^*]|x-x'|\big) \\
&\quad\le \big([Q]\vee 1\big)\big(1+[t^*]\big)\big([w]_*^{\widetilde{E},u}+[w]_1^{\widetilde{E},u}\big)|\xi-\xi'|.
\end{aligned}$$

Hence the result. □

Consequently, we may apply our numerical method to the time-augmented process $(\widetilde{X}_t)_{t\ge 0}$. In other words, for $l\in \boldsymbol{L}_c(\widetilde{E})$, $c\in \boldsymbol{L}_c(\partial\widetilde{E})$ and $\xi\in\widetilde{E}$, our approximation scheme may be used to compute

$$\widetilde{J}_N(l,c)(\xi)=\boldsymbol{E}_\xi\left[\int_0^{T_N} l(\widetilde{X}_t)\,dt+\sum_{j=1}^{N} c(\widetilde{X}_{T_j^-})\mathbb{1}_{\{\widetilde{X}_{T_j^-}\in\partial\widetilde{E}\}}\right]. \tag{7}$$

We will now see that the time-dependent functional and the deterministic time horizon problems boil down to computing such quantities $\widetilde{J}_N(l,c)(\xi)$ for suitably chosen functions l and c.

5.2. ***Lipschitz continuous cost functions.*** We first consider the time-dependent functional problem with Lipschitz continuous cost functions. Thus, let then $l\in \boldsymbol{L}_c(\widetilde{E})$, $c\in \boldsymbol{L}_c(\partial\widetilde{E})$ and $x\in E$, we wish to compute

$$\boldsymbol{E}_x\left[\int_0^{T_N} l(X_t,t)\,dt+\sum_{j=1}^{N} c(X_{T_j^-},T_j)\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}\right].$$

It is straightforward to show that this quantity may be expressed using the time-augmented process starting from the point $\xi_0=(x,0)$. Indeed, one has

$$\widetilde{J}_N(l,c)(\xi_0)=\boldsymbol{E}_x\left[\int_0^{T_N} l(X_t,t)\,dt+\sum_{j=1}^{N} c(X_{T_j^-},T_j)\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}\right],$$

where $\widetilde{J}_N(l,c)(\xi_0)$ is given by (7). Although they are time-dependent, the cost functions l and c are seen, in the left-hand side term, as time invariant functions of the time-augmented process. The expectation of the time-dependent functional is therefore obtained by computing the expectation of a time invariant functional for the transformed PDMP thanks to the approximation scheme described in Section 4. This is what expresses the following theorem, which proof stems from the previous discussion.

Theorem 5.4. *Let $l\in \boldsymbol{L}_c(\widetilde{E})$ and $c\in \boldsymbol{L}_c(\partial\widetilde{E})$ and apply the approximation scheme described in Section 4 to the time-augmented process $(\widetilde{X}_t)_{t\ge 0}$, one has then*

$$\left|\boldsymbol{E}_x\left[\int_0^{T_N} l(X_t,t)\,dt+\sum_{j=1}^{N} c(X_{T_j^-},T_j)\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}\right]-\hat{V}_0\right|\le \varepsilon_N(l,c,\widetilde{X},A),$$

where we denote by $\varepsilon_N(l,c,\widetilde{X},A)$ *the bound of the approximation error provided by Theorem 4.5 when our approximation scheme is applied with cost functions* l *and* c *to the time-augmented process* $(\widetilde{X}_t)_{t\ge 0}$.

Remark 5.5. The quantity $\varepsilon_N(l,c,\widetilde{X},A)$ is computed with respect to the process $(\widetilde{X}_t)_{t\ge 0}$ instead of $(X_t)_{t\ge 0}$, as presented in Theorem 4.5, so that

$$\begin{aligned}\varepsilon_N(l,c,\widetilde{X},A)=\sum_{k=0}^{N-1}\Big(&2[v_{k+1}]^{\widetilde{E}}\|\widetilde{Z}_{k+1}-\widehat{\widetilde{Z}}_{k+1}\|_p\\&+\big(2[v_k]^{\widetilde{E}}+[F]_1'+[F]_1''A\big)\|\widetilde{Z}_k-\widehat{\widetilde{Z}}_k\|_p\\&+\big([F]_2'+A[F]_2''\big)\|\widetilde{S}_{k+1}-\widehat{\widetilde{S}}_{k+1}\|_p\Big)+\frac{NC_cC_\lambda}{A}.\end{aligned}$$

where $(\widetilde{Z}_k,\widetilde{S}_k)_{k\in\mathbb{N}}$ denotes the sequence of the postjump locations and the interjump times of the time-augmented process $(\widetilde{X}_t)_{t\ge 0}$, and where

$$\begin{aligned}[F]_1'&=C_{t^*}[l]_1^{\widetilde{E}}+[c]_*^{\widetilde{E}},\\ [F]_1''&=[t^*]C_c,\\ [F]_2'&=C_l,\\ [F]_2''&=C_c,\\ C_{v_n}&\le n\big(C_{t^*}C_l+C_c\big),\\ [v_n]_1^{\widetilde{E}}&\le e^{C_{t^*}C_\lambda}\big(\widetilde{K}(A,v_{n-1})+nC_{t^*}[\lambda]_1(C_{t^*}C_l+C_c)\big)+C_{t^*}[l]_1^{\widetilde{E}},\\ [v_n]_2^{\widetilde{E}}&\le e^{C_{t^*}C_\lambda}\big(C_{t^*}C_lC_\lambda+2C_l+C_\lambda C_c+(2n-1)C_\lambda(C_{t^*}C_l+C_c)\big)+C_l,\\ [v_n]_*^{\widetilde{E}}&\le [v_n]_1^{\widetilde{E}}+[t^*][v_n]_2^{\widetilde{E}},\\ [v_n]^{\widetilde{E}}&\le \widetilde{K}(A,v_{n-1}),\end{aligned}$$

and for all $w\in\boldsymbol{L}_c(E)$ we have

$$\widetilde{K}(A,w)=\widetilde{E}_1+E_2A+\widetilde{E}_3[w]_1^{\widetilde{E}}+E_4C_w+[\widetilde{Q}][w]_*^{\widetilde{E}},$$

where

$$\begin{aligned}[\widetilde{Q}]&=([Q]\vee 1)(1+[t^*]),\\ \widetilde{E}_1&=2[l]_1^{\widetilde{E}}C_{t^*}+C_l\big([t^*]+2C_{t^*}^2[\lambda]_1\big)+[c]_*^{\widetilde{E}}(1+C_{t^*}C_\lambda)\\&\quad+C_c\big(2[\lambda]_1C_{t^*}+C_\lambda C_{t^*}^2[\lambda]_1+2[t^*]C_\lambda\big),\\ E_2&=C_cC_{t^*}C_\lambda[t^*],\end{aligned}$$

$$\widetilde{E}_3 = (1 + C_{t^*} C_\lambda)[\widetilde{Q}],$$
$$E_4 = 2C_\lambda [t^*] + C_{t^*} [\lambda]_1 (2 + C_{t^*} C_\lambda).$$

5.3. ***Deterministic time horizon.*** In the context of applications, it seems relevant to consider a deterministic time horizon t_f. For instance, one may want to estimate a mean cost over a given period no matter how many jumps occur during this period. Actually, we will choose a time horizon of the form $t_f \wedge T_N$ with N large enough to ensure the N-th jump will occur after time t_f with high probability: in other words, that $\boldsymbol{P}_x(T_N < t_f)$ be close to zero. For a discussion concerning the choice of such N, and in particular a theoretical bound of the probability $\boldsymbol{P}_x(T_N < t_f)$, we refer to [3]. Simply notice that in practice, this probability may be estimated through Monte Carlo simulations. We thus intend to approximate the following quantity for $l \in \boldsymbol{L}_c(\widetilde{E})$, $c \in \boldsymbol{L}_c(\partial \widetilde{E})$ and $x \in E$:

$$\boldsymbol{E}_x\left[\int_0^{T_N \wedge t_f} l(X_t, t)\, dt + \sum_{T_j \le t_f} c(X_{T_j^-}, T_j) \mathbb{1}_{\{X_{T_j^-} \in \partial E\}} \right]$$
$$= \boldsymbol{E}_x\left[\int_0^{T_N} l(X_t, t) \mathbb{1}_{\{t \le t_f\}} dt + \sum_{j=1}^{N} c(X_{T_j^-}, T_j) \mathbb{1}_{\{X_{T_j^-} \in \partial E\}} \mathbb{1}_{\{T_j \le t_f\}} \right].$$

The natural approach would consist in killing the process at time t_f as Davis suggests in [5, Section 31], and applying our method to the new process. However, the killed process will not necessarily fulfill our Lipschitz continuity assumptions because of the discontinuity introduced at time t_f.

A second idea would then be to use the previous results, to consider the time-augmented process, and to define $\tilde{l}(x,t) = l(x,t)\mathbb{1}_{\{t \le t_f\}}$ and $\tilde{c}(x,t) = c(x,t)\mathbb{1}_{\{t \le t_f\}}$. However, a similar problem appears. Indeed, such functions $\tilde{l}$ and $\tilde{c}$ are not Lipschitz continuous and our numerical method requires this assumption. In the rest of this section, we will see how to overcome this drawback. On the one hand, we prove that the Lipschitz continuity condition on l may be relaxed so that our numerical method may be used directly to approximate $\widetilde{J}_N(\tilde{l}, c)$ for any $c \in \boldsymbol{L}_c(\partial \widetilde{E})$. On the other hand, in the general case, we will deal with the non-Lipschitz continuity of $\tilde{c}$ by bounding it between two Lipschitz continuous functions.

5.3.1. *Direct estimation of the running cost term.* Let us explain how the Lipschitz continuity condition on the running cost function may be relaxed so that Theorem 4.5, stating the convergence of our approximation scheme, remains true when the running cost function is $\tilde{l}(x,t) = l(x,t)\mathbb{1}_{\{t \le t_f\}}$ with $l \in \boldsymbol{L}_c(\widetilde{E})$ and the boundary jump cost function is $c \in \boldsymbol{L}_c(\partial \widetilde{E})$ (although with slightly different constants in the bound of the convergence rate). Indeed, the running cost function $\tilde{l}$ appears inside an integral that will have a regularizing effect allowing us to derive

the required Lipschitz property of the functional in spite of the discontinuity of $\tilde{l}$. Details are provided in Appendix B.

Consequently, our approximation scheme may be used to compute $\widetilde{J}_N(\tilde{l},c)(\xi)$ for any $\xi \in \widetilde{E}$. We recall that $\widetilde{J}_N$ is defined by (7) and that for all $x \in E$, one has

$$\widetilde{J}_N(\tilde{l},c)(x,0) = \boldsymbol{E}_x\left[\int_0^{T_N \wedge t_f} l(X_t,t)\,dt + \sum_{j=1}^{N} c(X_{T_j^-},T_j)\mathbb{1}_{\{X_{T_j^-}\in\partial E\}}\right].$$

We now turn to the indicator function $\mathbb{1}_{\{T_j \le t_f\}}$ required within the boundary jump cost term.

5.3.2. *Bounds of the boundary jump cost term.* We explained how the Lipschitz continuity condition on l may be relaxed. However, when it comes to c, this condition cannot be avoided and our numerical method cannot be used directly with $\tilde{c}(x,t) = c(x,t)\mathbb{1}_{\{t\le t_f\}}$. We overcome this drawback by using Lipschitz continuous approximations of the indicator function. Indeed, for $B > 0$, we introduce the real-valued functions $\underline{u}_B$ and $\bar{u}_B$ defined on $\mathbb{R}$ by

$$\underline{u}_B(t) = \begin{cases} 1 & \text{if } t < t_f - 1/B, \\ -B(t-t_f) & \text{if } t_f - 1/B \le t < t_f, \\ 0 & \text{if } t_f \le t, \end{cases}$$

$$\bar{u}_B(t) = \begin{cases} 1 & \text{if } t < t_f, \\ -B(t-t_f)+1 & \text{if } t_f \le t < t_f + 1/B, \\ 0 & \text{if } t_f + 1/B \le t. \end{cases}$$

The following lemma is straightforward.

Lemma 5.6. *For all* $t \ge 0$, $\lim_{B\to+\infty} \underline{u}_B(t) = \mathbb{1}_{[0;t_f)}(t)$ *and* $\lim_{B\to+\infty} \bar{u}_B(t) = \mathbb{1}_{[0;t_f]}(t)$. *Furthermore, for all* $B > 0$, $\underline{u}_B$ *and* $\bar{u}_B$ *are Lipschitz continuous with Lipschitz constant* B. *Moreover,* $|\underline{u}_B - \mathbb{1}_{[0;t_f]}| \le 1$, $|\bar{u}_B - \mathbb{1}_{[0;t_f]}| \le 1$ *and*

$$\underline{u}_B \le \mathbb{1}_{[0;t_f]} \le \bar{u}_B.$$

Thus, define for $l \in \boldsymbol{L}_c(\widetilde{E})$

$$\tilde{l}(x,t) = l(x,t)\mathbb{1}_{\{t\le t_f\}} \tag{8}$$

and for $c \in \boldsymbol{L}_c(\partial\widetilde{E})$ and for all $B > 0$,

$$\underline{c}_B(x,t) = c(x,t)\underline{u}_B(t) \qquad \text{and} \qquad \bar{c}_B(x,t) = c(x,t)\bar{u}_B(t). \tag{9}$$

We now check that these functions satisfy our Lipschitz continuity conditions.

Proposition 5.7. *The functions* $\underline{c}_B$ *and* $\bar{c}_B$ *belong to* $\boldsymbol{L}_c(\partial\widetilde{E})$ *with* $[\underline{c}_B]_*, [\bar{c}_B]_* \le [c]_* + BC_c(1 \vee [t^*])$.

Proof. We prove the result for $\underline{c}_B$, the other case being similar. For all $\xi = (x,t), \xi' = (x',t') \in \widetilde{E}$, one has

$$\begin{aligned}
&\left|\underline{c}_B\big(\widetilde{\Phi}(\xi, t^*(\xi))\big) - \underline{c}_B\big(\widetilde{\Phi}(\xi', t^*(\xi'))\big)\right| \\
&\qquad = \left|c\big(\widetilde{\Phi}(\xi, \tilde{t^*}(\xi))\big)\underline{u}_B(t + \tilde{t^*}(\xi)) - c\big(\widetilde{\Phi}(\xi', \tilde{t^*}(\xi'))\big)\underline{u}_B(t' + \tilde{t^*}(\xi'))\right| \\
&\qquad \le [c]_*|\xi - \xi'| + C_c\left|\underline{u}_B(t + \tilde{t^*}(\xi)) - \underline{u}_B(t' + \tilde{t^*}(\xi'))\right| \\
&\qquad \le [c]_*|\xi - \xi'| + C_c B\big(|t - t'| + [\tilde{t^*}]|x - x'|\big) \\
&\qquad \le \big([c]_* + C_c B(1 \vee [t^*])\big)|\xi - \xi'|.
\end{aligned}$$

Hence the result. □

Therefore, the functions $\underline{c}_B$ and $\bar{c}_B$ are acceptable boundary jump cost functions and we may bound the deterministic horizon expectation by

$$\begin{aligned}
\widetilde{J}_N(\tilde{l}, \underline{c}_B)(x,0) &\le \boldsymbol{E}_x\left[\int_0^{T_N} l(X_t)\mathbb{1}_{\{t \le t_f\}} dt + \sum_{j=1}^{N} c(X_{T_j^-})\mathbb{1}_{\{X_{T_j^-} \in \partial E\}}\mathbb{1}_{\{T_j \le t_f\}}\right] \\
&\le \widetilde{J}_N(\tilde{l}, \bar{c}_B)(x,0).
\end{aligned}$$

The following proposition provides the convergence of the bounds.

Proposition 5.8. *For all $x \in E$, one has*

$$\begin{aligned}
&\lim_{B \to +\infty} \widetilde{J}_N(\tilde{l}, \underline{c}_B)(x,0) \\
&\qquad = \lim_{B \to +\infty} \widetilde{J}_N(\tilde{l}, \bar{c}_B)(x,0) \\
&\qquad = \boldsymbol{E}_x\left[\int_0^{T_N \wedge t_f} l(X_t, t)\, dt + \sum_{j=1}^{N} c(X_{T_j^-}, T_j)\mathbb{1}_{\{X_{T_j^-} \in \partial E\}}\mathbb{1}_{\{T_j \le t_f\}}\right].
\end{aligned}$$

Convergence holds for every $t_f > 0$ in the case of $\widetilde{J}_N(\tilde{l}, \bar{c}_B)(x,0)$ but only for almost every $t_f > 0$ with respect to the Lebesgue measure on $\mathbb{R}$ in the case of $\widetilde{J}_N(\tilde{l}, \underline{c}_B)(x,0)$.

Proof. Let $x \in E$. We first consider $\widetilde{J}_N(\tilde{l}, \bar{c}_B)(x,0)$.

$$\begin{aligned}
&\left|\boldsymbol{E}_x\left[\sum_{j=1}^{N} c(X_{T_j^-}, T_j)\mathbb{1}_{\{X_{T_j^-} \in \partial E\}}\mathbb{1}_{\{T_j \le t_f\}} - \sum_{j=1}^{N} \bar{c}_B(X_{T_j^-}, T_j)\mathbb{1}_{\{X_{T_j^-} \in \partial E\}}\right]\right| \\
&\qquad \le \boldsymbol{E}_x\left[\sum_{j=1}^{N} \left|c(X_{T_j^-}, T_j)\right|\left|\mathbb{1}_{\{T_j \le t_f\}} - \bar{u}_B(T_j)\right|\right] \\
&\qquad \le C_c \boldsymbol{E}_x\left[\sum_{j=1}^{N} \mathbb{1}_{\{t_f < T_j \le t_f + \frac{1}{B}\}}\right] \le C_c \sum_{j=1}^{N} \left(\varphi_j\left(t_f + \frac{1}{B}\right) - \varphi_j(t_f)\right),
\end{aligned}$$

where φ_j is the distribution function of T_j. For all $j \leq N$, the summand in this last expression goes to 0 as $B \to +\infty$, since φ_j is right-continuous; this shows the required convergence.

We now turn to the case of $\widetilde{J}_N(\tilde{l}, \underline{c}_B)(x, 0)$. Similar computations yield

$$\left| \boldsymbol{E}_x \left[\sum_{j=1}^{N} c(X_{T_j^-}, T_j) \mathbb{1}_{\{X_{T_j^-} \in \partial E\}} \mathbb{1}_{\{T_j \leq t_f\}} - \sum_{j=1}^{N} \underline{c}_B(X_{T_j^-}, T_j) \mathbb{1}_{\{X_{T_j^-} \in \partial E\}} \right] \right| \leq C_c \sum_{j=1}^{N} \left(\varphi_j(t_f) - \varphi_j\left(t_f - \frac{1}{B}\right) \right).$$

One cannot conclude as in the previous case, since φ_j need not be left-continuous. We therefore assume that t_f is not an atom of any of the laws of the random variables T_j. Then, for all $j \leq N$, the summand on the right-hand side tends to 0 as $B \to +\infty$, and the result follows. Indeed, the set of the atoms of T_j is at most countable, so the convergence holds for almost every t_f with respect to the Lebesgue measure on $\mathbb{R}$. □

5.3.3. *Bounds in the general case.* The previous results show that the deterministic horizon expectation may be bounded by applying our numerical method with $\tilde{l}$ and successively $\underline{c}_B$ and $\bar{c}_B$. In other words, we have shown:

Theorem 5.9. *Let $l \in \boldsymbol{L}_c(\widetilde{E})$ and $c \in \boldsymbol{L}_c(\partial \widetilde{E})$. Let $(\underline{V}_{k,B})_{0 \leq k \leq N}$ (respectively $(\overline{V}_{k,B})_{0 \leq k \leq N}$) be the sequence of random variables $(V_k)_{0 \leq k \leq N}$ described in Section 4 when applying our approximation scheme to the time-augmented process $(\widetilde{X}_t)_{t \geq 0}$ with cost functions $\tilde{l}$ and $\underline{c}_B$ (respectively $\bar{c}_B$) defined by (8) and (9). The bounds of the approximation error provided by Theorem 4.5 are respectively denoted by*

$$\varepsilon_N(l, \underline{c}_B, \widetilde{X}, A, B) \quad \text{and} \quad \varepsilon_N(l, \bar{c}_B, \widetilde{X}, A, B).$$

One has then

$$\begin{aligned} \underline{V}_{0,B} - \varepsilon_N(l, \underline{c}_B, \widetilde{X}, A, B) &\leq \boldsymbol{E}_x \left[\int_0^{T_N \wedge t_f} l(X_t, t)\, dt + \sum_{j=1}^{N} c(X_{T_j^-}, T_j) \mathbb{1}_{\{X_{T_j^-} \in \partial E\}} \mathbb{1}_{\{T_j \leq t_f\}} \right] \\ &\leq \overline{V}_{0,B} + \varepsilon_N(l, \bar{c}_B, \widetilde{X}, A, B). \end{aligned}$$

Remark 5.10. In the previous theorem, the quantity $\varepsilon_N(l, \underline{c}_B, \widetilde{X}, A, B)$ (and similarly $\varepsilon_N(l, \bar{c}_B, \widetilde{X}, A, B)$) is computed with respect to the process $(\widetilde{X}_t)_{t \geq 0}$ instead

of $(X_t)_{t\geq 0}$ as presented in Theorem 4.5 so that one has

$$\begin{aligned}
&\varepsilon_N(l, \underline{c}_B, \widetilde{X}, A, B) \\
&\quad = \sum_{k=0}^{N-1} \Big(2[v_{k+1}]^{\widetilde{E}} \|\widetilde{Z}_{k+1} - \widehat{\widetilde{Z}}_{k+1}\|_p \\
&\qquad + \big(2[v_k]^{\widetilde{E}} + [F]_1' + [F]_1'' A + [F]_1''' B\big) \|\widetilde{Z}_k - \widehat{\widetilde{Z}}_k\|_p \\
&\qquad + \big([F]_2' + [F]_2'' A\big) \|\widetilde{S}_{k+1} - \widehat{\widetilde{S}}_{k+1}\|_p \Big) + \frac{N C_c C_\lambda}{A}.
\end{aligned}$$

where $(\widetilde{Z}_k, \widetilde{S}_k)_{k\in\mathbb{N}}$ denotes the sequence of the postjump locations and the interjump times of the time-augmented process $(\widetilde{X}_t)_{t\geq 0}$ and with

$$\begin{aligned}
[F]_1''' &= C_c(1 \vee [t^*]), \\
[v_n]_1^{\widetilde{E}} &\leq e^{C_{t^*} C_\lambda} \big(\widetilde{K}(A, B, v_{n-1}) + n C_{t^*} [\lambda]_1 (C_{t^*} C_l + C_c)\big) + C_{t^*} [l]_1^{\widetilde{E}}, \\
[v_n]^{\widetilde{E}} &\leq \widetilde{K}(A, B, v_{n-1}),
\end{aligned}$$

and for all $w \in \boldsymbol{L}_c(E)$ we have

$$\widetilde{K}(A, B, w) = E_1' + E_2'' B + E_2 A + \widetilde{E}_3 [w]_1^{\widetilde{E}} + E_4 C_w + [\widetilde{Q}][w]_*^{\widetilde{E}},$$

where

$$\begin{aligned}
E_1' &= 2[l]_1^{\widetilde{E}} C_{t^*} + C_l \big([t^*] + 2C_{t^*}^2 [\lambda]_1\big) + [c]_*^{\widetilde{E}} (1 + C_{t^*} C_\lambda) \\
&\qquad + C_c \big(2[\lambda]_1 C_{t^*} + C_\lambda C_{t^*}^2 [\lambda]_1 + 2[t^*] C_\lambda\big), \\
E_1'' &= C_c (1 \vee [t^*])(1 + C_{t^*} C_\lambda)
\end{aligned}$$

The other constants remain unchanged; see Remark 5.5 for their expressions.

Furthermore, it is important to stress the fact that applying twice our numerical method does not increase significantly the computing time. Indeed, the computation of the quantization grids is, by far, the most costly step. These grids, that only depend on the dynamics of the process, may be stored off-line and used for the approximation of both bounds.

The choice of B. We now discuss the choice of the parameter B, the discussion is quite similar to the one concerning the choice of A in Section 4.2. Proposition 5.8 suggests that B should be chosen as large as possible. However, choosing a large value for B will lead to large Lipschitz constants that will decrease the sharpness of the bounds $\varepsilon_N(l, \underline{c}_B, \widetilde{X})$ and $\varepsilon_N(l, \bar{c}_B, \widetilde{X})$ for the approximation error provided by Theorem 4.5. Indeed, it is easy to check that $[v_n]$ grows linearly with B (see the precise expressions of the Lipschitz constants above). Thus, in order to control the error proposed by Theorem 4.5, it is necessary that the order of magnitude of the quantization error $\|\Theta_n - \widehat{\Theta}_n\|_p$ be at most $1/B$.

6. Numerical results

6.1. ***A repair workshop model.*** We now present a repair workshop model adapted from [5, Section 21].

In a factory, a machine produces goods which daily value is $r(x)$, where $x \in [0; 1]$ represents a parameter of evolution of the machine, a setting chosen by the operator. For instance, x may be some load or some pace imposed on the machine. This machine, initially working, may break down with an age-dependent hazard rate $\lambda(t)$ and is then sent to the workshop for repair. Besides, the factory's management has decided that, whenever the machine has worked for a whole year without requiring repair, it is sent to the workshop for maintenance. The daily cost of maintenance is $q(x)$, while the daily cost of a repairs is $p(x)$, with reasonably $p(x) > q(x)$. We assume that after a repair or maintenance, both lasting a fixed time s, the machine is totally repaired and is not worn down.

We therefore consider three modes: the machine is working ($m = 1$), being repaired ($m = 2$), or undergoing maintenance ($m = 3$). The state of the process at time t will be denoted by $X_t = (m_t, \zeta_t, t)$, where ζ_t is the time since the last change of mode. (This component is required since the hazard rate λ is age-dependent.) The state space is $E = \big(\{1\}\times[0; 365]\times\mathbb{R}^+\big) \cup \big(\{2\}\times[0; s]\times\mathbb{R}^+\big) \cup \big(\{3\}\times[0; s]\times\mathbb{R}^+\big)$. In each mode, the flow is $\Phi_m\big((\zeta, t), u\big) = (\zeta + u, t + u)$. Concerning the transition kernel, one sees from the previous discussion that, for instance, from the point $(1, \zeta, t)$, the process can jump to the point $(2, 0, t)$ if $\zeta < 365$ and the jump is forced to $(3, 0, t)$ if $\zeta = 365$. Figure 1 presents the state space and an example of trajectory of the process.

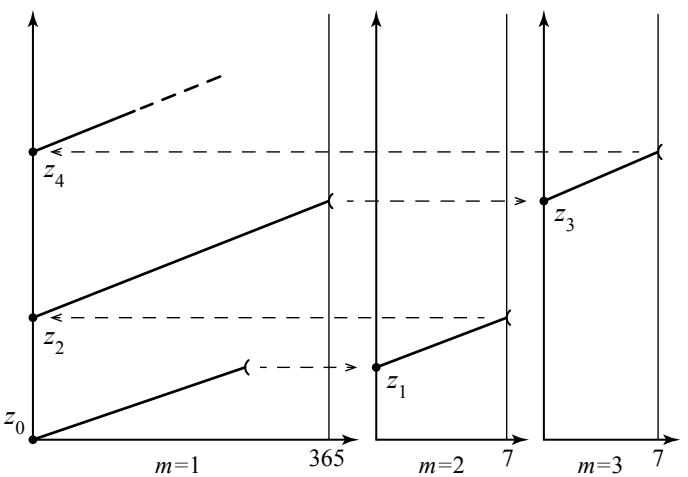


Figure 1. An example trajectory. The process starts from the point Z_0 in mode $m = 1$ (machine in service). The machine may be sent to the workshop for repairs ($m = 2$) or for maintenance ($m = 3$).

Our aim is to find the value of the setting x that maximizes the expected total benefits $B(x)$, that is, the discounted value (for an interest rate ρ) of production minus maintenance and reparation costs over a period $t_f = 5$ years:

$$B^* = \sup_{x\in[0;1]} B(x),$$

where

$$B(x) = \boldsymbol{E}_{(1,0,0)}\left[\int_0^{t_f} e^{-\rho t}\left(r(x)\mathbb{1}_{\{m_t=1\}} - p(x)\mathbb{1}_{\{m_t=2\}} - q(x)\mathbb{1}_{\{m_t=3\}}\right) dt\right].$$

We will use the following values $r(x) = x$, $p(x) = 100x^2$, $q(x) = 5$, $s = 7$ days, $\rho = \frac{0.03}{365}$ and λ represents a Weibull distribution with parameters $\alpha = 2$ et $\beta = 600$.

Our assumptions clearly hold so that we may run our numerical method. We first need to find $N \in \mathbb{N}$ such that $\boldsymbol{P}_{(1,0,0)}(T_N < t_f)$ be small. Monte Carlo simulations lead to the value $N = 18$. For a fixed $x \in [0; 1]$, we will therefore compute $\widetilde{J}_N(\tilde{l}, 0)(1, 0, 0)$ where $\tilde{l}(m, \zeta, t) = e^{-\rho t}\big(r(x)\mathbb{1}_{\{m=1\}} - p(x)\mathbb{1}_{\{m=2\}} - q(x)\mathbb{1}_{\{m=3\}}\big)\mathbb{1}_{\{t\le t_f\}}$. Finally, notice that we could have chosen r, p and q slightly more generally by allowing them to be time-dependent.

It is important to stress the fact that, once the Markov chain associated to the process is quantized, we will be able to compute the approximation of $B(x)$ almost instantly for any $x \in [0; 1]$ because the same grids are used for every computation. Thanks to this flexibility, we are able to draw the function $x \to B(x)$ and, thus, to solve the above optimization problem very easily. This is a very important advantage of our method. Indeed, if we computed $B(x)$ through standard methods such as Monte Carlo simulations, we would have to repeat the whole algorithm again and again for each value of x and solving the optimization problem would be intractable.

The following figure represents the approximation of the function B computed on a constant step grid of $[0; 1]$ with step 10^{-2}. This leads to the solution of the earlier optimization problem. Indeed, we obtain $B^* = B(x^*) = 537.84$ where $x^* = 0.78$ is the value of the setting x that maximizes the benefits of the factory.

Now let $x = 0.78$. The following table presents the values of $\widehat{V}_N$, which are the approximations of $B(x)$, for different number of points in the quantization grids. A reference value $B_{\text{Monte Carlo}} = 537.69$ is obtained via the Monte Carlo method (10^8 simulations).

Points in the quantization grids	$\widehat{V}_N$	relative error to 537.69
20 points	542.14	0.83%
50 points	539.57	0.35%
100 points	538.24	0.10%
500 points	537.84	0.03%

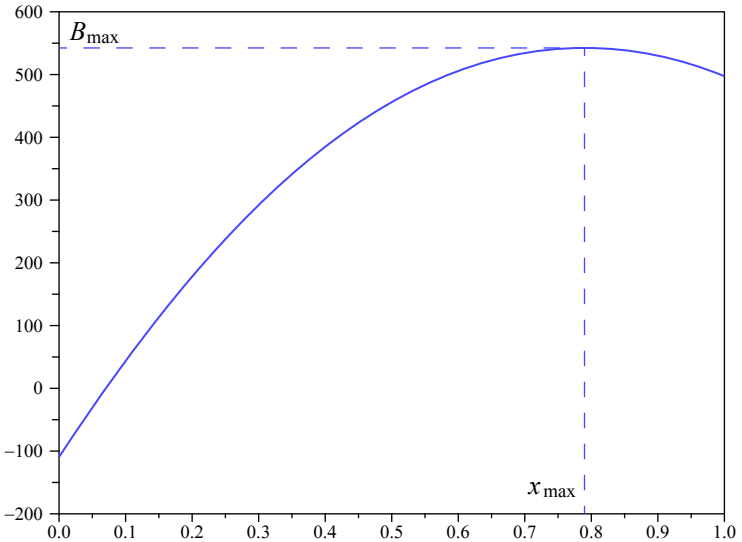


Figure 2. The function B drawn with 500 points in the quantization grids.

From a computational time point of view, we have already explained that the computation of large quantization grids is, by far, the most costly step since it may take up to several hours whereas the approximation of the expectation that follows is then almost instantaneous. However, we may notice, in the above table, that grids containing only 50 points yield a quite accurate result with merely 0.35% error. Such grids only require a few minutes to be designed.

Remark 6.1. We already noticed that the same grids may serve several purposes. For instance, we may also have been interested in the computation of the mean time spent by the machine in the workshop by taking $l(m, \zeta, t) = \mathbb{1}_{\{m \in \{2;3\}\}}$.

6.2. ***A corrosion model.*** We consider here a corrosion model for an aluminum metallic structure. This example was provided by Astrium. It concerns a small structure within a strategic ballistic missile. The missile is stored successively in three different environments which are more or less corrosive. It is made to have potentially large storage durations. The requirement for security is very strong. The mechanical stress exerted on the structure depends in part on its thickness. A loss of thickness will cause an overconstraint and therefore increase the risk of rupture. It is thus crucial to study the evolution of the thickness of the structure over time.

Let us describe more precisely the usage profile of the missile. It is stored successively in three different environments: the workshop ($m = 1$), the submarine in operation ($m = 2$) and the submarine in dry-dock ($m = 3$). This is because the

structure must be equipped and used in a given order. Then it goes back to the workshop and so on. The missile stays in each environment during a random duration with exponential distribution. Its parameter λ_m depends on the environment. The degradation law for the thickness loss then depends on the environment through two parameters, a deterministic transition period η_m and a random corrosion rate ρ uniformly distributed within a given range. Typically, the workshop and dry-dock are the most corrosive environments but the time spent in operation is more important. The randomness of the corrosion rate accounts for small variations and uncertainties in the corrosiveness of each environment.

In each environment $m \in \{1; 2; 3\}$, the thickness loss d_m evolves in time as

$$d_m(\rho, s) = \rho\big(s + \eta_m(e^{-s/(2\eta_m)} - 1)\big). \tag{10}$$

Here are the numerical values of the parameters of the corrosion model:

	environment 1	environment 2	environment 3
λ_m (h^{-1})	$(17520)^{-1}$	$(131400)^{-1}$	$(8760)^{-1}$
η_m (h)	30000	200000	40000
ρ (mm/h)	$[10^{-6}, 10^{-5}]$	$[10^{-7}, 10^{-6}]$	$[10^{-6}, 10^{-5}]$

Initially, the structure is in environment $m = 1$ and the thickness loss is null. One draws the corrosion rate ρ_0 uniformly distributed in the interval $[10^{-6}, 10^{-5}]$ and the time of the first change of environment T_1 exponentially distributed with parameter $\lambda_1 = (17520)^{-1}$ hours^{-1}. The corrosion starts according to (10) so that, for all $0 \le t \le T_1$, the loss of thickness is $d_1(\rho_0, t)$. The structure then moves to environment 2 and the process restarts similarly: a new corrosion rate ρ_{T_1} is drawn according to an uniform law on $[10^{-7}, 10^{-6}]$, the time of the second jump T_2 is drawn so that $T_2 - T_1$ is exponentially distributed with parameter $\lambda_2 = (131400)^{-1}$ hours^{-1} and for $T_1 \le t \le T_2$, the loss of thickness is $d_1(\rho_0, T_1) + d_2(\rho_{T_1}, t - T_1)$ and so on.

At each change of environment, a new corrosion rate ρ is drawn according to a uniform law on the corresponding interval. The thickness loss, however, evolves continuously.

We are interested in computing the mean loss of thickness in environment 2 until a given time $t_f = 18$ years.

Modeling by PDMP.

The state space E. The loss of thickness will be modeled by a PDMP whose modes are the different environments. Let then $M = \{1, 2, 3\}$. The PDMP $(X_t)_{t \ge 0}$ will contain the following components: the mode $m \in M$, the loss of thickness d, the time since the last jump s (this is to ensure that the Markov property is satisfied),

the corrosion rate ρ and the time t (since we consider the time-augmented process). Clearly, one has always $s \le t$, so we can reasonably consider the state space

$$E = \big\{(m,d,s,\rho,t) \in M \times \mathbb{R}^+ \times \mathbb{R}^+ \times [10^{-7};10^{-5}] \times \mathbb{R}^+ \text{ such that } s \le t\big\}.$$

The flow Φ. The flow is given for all $u \ge 0$ by

$$\Phi\left(\begin{pmatrix} m \\ d \\ s \\ \rho \\ t \end{pmatrix}, u\right) = \begin{pmatrix} m \\ d + d_m(\rho, s+u) - d_m(\rho,s) \\ s+u \\ \rho \\ t+u \end{pmatrix}.$$

The transition kernel Q. Let us now study the jumps of this process. When the process jumps from a point $x = (m,d,s,\rho,t) \in E$, m becomes $m+1$ modulo 3 (denoted $m+1[3]$), d and t remain unchanged, s becomes 0. Only ρ is randomly drawn, according to a uniform law on an interval $[\rho_{\min};\rho_{\max}]$ that depends on the new mode. One has then for $w \in B(E)$, $x = (m,d,s,\rho,t) \in E$, and $u \ge 0$,

$$\begin{aligned} Qw\big(\Phi\left(\begin{pmatrix} m \\ d \\ s \\ \rho \\ t \end{pmatrix}, u\right)\big) &= Qw \begin{pmatrix} cm \\ d + d_m(\rho, s+u) - d_m(\rho,s) \\ s+u \\ \rho \\ t+u \end{pmatrix} \\ &= \frac{1}{\rho_{\max} - \rho_{\min}} \int_{\rho_{\min}}^{\rho_{\max}} w \begin{pmatrix} m+1[3] \\ d + d_m(\rho, s+u) - d_m(\rho,s) \\ 0 \\ \tilde{\rho} \\ t+u \end{pmatrix} d\tilde{\rho}. \end{aligned} \tag{11}$$

The cost function l. The function $l \in B(E)$ will be the cost function to compute the mean loss of thickness in mode 2. It is defined as follows: for all $x = (m,d,s,\rho,t) \in E$ and $u \ge 0$,

$$l(\Phi(x,u)) = \rho\big(1 - \tfrac{1}{2} e^{-(s+u)/(2\eta_m)}\big)\mathbb{1}_{\{m=2\}} = \frac{d}{du}\big(d_m(\rho, s+u)\big)\mathbb{1}_{\{m=2\}}. \tag{12}$$

One then defines $\tilde{l}(\Phi(x,u)) = l(\Phi(x,u))\mathbb{1}_{\{t+u \le t_f\}}$, so that

$$\begin{aligned} L(x,u) &= \int_0^u \tilde{l}\big(\Phi(x,u')\big)\, du' = \int_0^{u \wedge (t_f - t)^+} l\big(\Phi(x,u')\big)\, du' \\ &= \big(d_m(\rho, s + u \wedge (t_f - t)^+) - d_m(\rho,s)\big)\mathbb{1}_{\{m=2\}}; \end{aligned}$$

that is indeed the thickness lost in mode 2 from the point $x = (m, d, s, \rho, t)$ during a time $u \wedge (t_f - t)^+$.

The assumptions. Assumptions 2.1 and 2.9 are clearly satisfied. It is easy to check, from (12), that $l \in \boldsymbol{L}_c(E)$, so Assumption 3.1 holds.

We now turn to Assumption 2.13 and we will see that, although it does not hold for any function $w \in \boldsymbol{L}_c^v(E)$, it holds for a sufficiently big subclass of functions. We first need to make a remark. Recall that for all $x = (m, d, s, \rho, t) \in E$ and for all $k \in \{0, \dots, N\}$, one has $v_{N-k}(x) = \boldsymbol{E}_x bigl[\int_0^{T_k} l\big(\Phi(x,u)\big)\mathbb{1}_{\{t+u\le t_f\}} du\bigr]$. Therefore, for all $k \in \{0, \dots, N\}$ the function v_k as well as the function $\tilde{l}$ satisfy

$$x = (m, d, s, \rho, t) \in E \text{ and } t \ge t_f \implies w(x) = 0. \tag{13}$$

The next step consists in proving that Assumption 2.13, although it is not satisfied for any function $w \in \boldsymbol{L}_c^v(E)$, holds for any function $w \in \boldsymbol{L}_c^v(E)$ that also satisfies condition (13). This is done in Lemma 6.2 and it is sufficient because in the proof of the theorem that ensures the convergence of our approximation scheme, Assumption 2.13 is only used with the functions $(v_k)_{k\in\{0,\dots,N\}}$ that do satisfy condition (13).

Lemma 6.2. *There exists $[Q] \in \mathbb{R}^+$ such that for all $v \ge 0$ and $w \in \boldsymbol{L}_c^v(E)$ that satisfies condition (13), one has for all $x, x' \in E$ and $0 \le u \le v$,*

$$\big|Qw\big(\Phi(x,u)\big) - Qw\big(\Phi(x',u)\big)\big| \le [Q][w]_1^{E,v}|x - x'|.$$

Proof. Let $x = (m, d, s, \rho, t)$ and $x' = (m', d', s', \rho', t') \in E$ with for instance $t \le t'$. First we may choose $m = m'$; otherwise, $|x - x'| = +\infty$ and there is nothing to prove. Now, we are facing three different cases:

- If $t_f \le t + u \le t' + u$, then one has $Qw\big(\Phi(x,u)\big) = Qw\big(\Phi(x',u)\big) = 0$ because w satisfies condition (13) and there is nothing to prove.
- If $t + u \le t_f \le t' + u$, notice that

$$Qw\big(\Phi(x',u)\big) = Qw\big(\Phi((m', d', s', \rho', t_f), u)\big) = 0$$

(this stems from condition (13)), so that we are reduced to the following case.

- We assume from now on that $t + u \le t' + u \le t_f$. We now intend to bound $\big|Qw\big(\Phi(x,u)\big) - Qw\big(\Phi(x',u)\big)\big|$. It is clear from (11) that we only need to prove that the function $(\rho, s) \to d_m(\rho, s)$, defined by (10), is Lipschitz continuous with respect to both its variables on the set $[10^{-7}; 10^{-5}] \times [0; t_f]$. Indeed, we have $s \le t$ and $s' \le t'$ so that $s, s', s + u, s' + u \le t_f$. Standard computations yield

$$|d_m(\rho, s) - d_m(\rho', s')| \le s|\rho - \rho'| + \tfrac{3}{2}\rho'|s - s'| \le t_f|\rho - \rho'| + \tfrac{3}{2}10^{-5}|s - s'|.$$

Hence the result. □

Assumption 2.10 is not satisfied because in our corrosion model, one has $t^*(x) = +\infty$ for all $x \in E$. Besides, we may notice that the previous proof would have been more straightforward if t^* had been bounded. Indeed in that case, we would have had $s, s', s+u, s'+u \leq C_{t^*}$ and the introduction of condition (13) would have been unnecessary. Nevertheless, we have been able to overcome the drawback of having t^* unbounded by noticing that somehow the deterministic time horizon t_f plays the part of the missing C_{t^*}. This is the meaning of condition (13): roughly speaking, we do not consider what happens beyond t_f.

More generally, we will now see that in our deterministic time horizon problem, the boundedness of t^* may be dropped and our results remain true replacing C_{t^*} by t_f. This is clear in the case of Proposition A.2 because the function $\tilde{l}$ satisfies the condition (13). Proposition A.7 remains also true replacing C_{t^*} by t_f. Indeed, on the one hand, it is clear that $L(x,u) \leq t_f C_l$. On the other hand, when computing $|v_n(\Phi(x,u)) - v_n(\Phi(x',u'))|$, we are facing three cases, as in the proof of Lemma 6.2:

- If $t_f \leq u \leq u'$, one has

$$v_n(\Phi(x,u)) = v_n(\Phi(x',u')) = 0,$$

by condition (13).

- If $u \leq t_f \leq u'$, one has

$$\big|v_n(\Phi(x,u)) - v_n(\Phi(x',u'))\big| = \big|v_n(\Phi(x,u)) - v_n(\Phi(x',t_f))\big|,$$

since $v_n(\Phi(x',u')) = v_n(\Phi(x',t_f)) = 0$ (condition (13) once again), so that we are reduced to the next case.

- If $u \leq u' \leq t_f$, the computations remain unchanged and t_f replaces C_{t^*} as a bound for u and u'.

Numerical results. The table below presents the values of the loss of thickness in environment 2 obtained through our approximation scheme with quantization grids of varying fineness, as well as the relative deviation with respect to the Monte Carlo value of 0.036755, obtained with 10^8 simulations.

Quantization grids	$\widehat{V}_0$	error	Quantization grids	$\widehat{V}_0$	error
20 points	0.038386	4.43%	2000 points	0.037041	0.77%
50 points	0.037804	2.85%	4000 points	0.037007	0.69%
100 points	0.037525	2.09%	6000 points	0.036973	0.57%
200 points	0.037421	1.81%	8000 points	0.036944	0.49%
500 points	0.037264	1.38%	10000 points	0.036911	0.40%
1000 points	0.037160	1.10%	12000 points	0.036897	0.36%

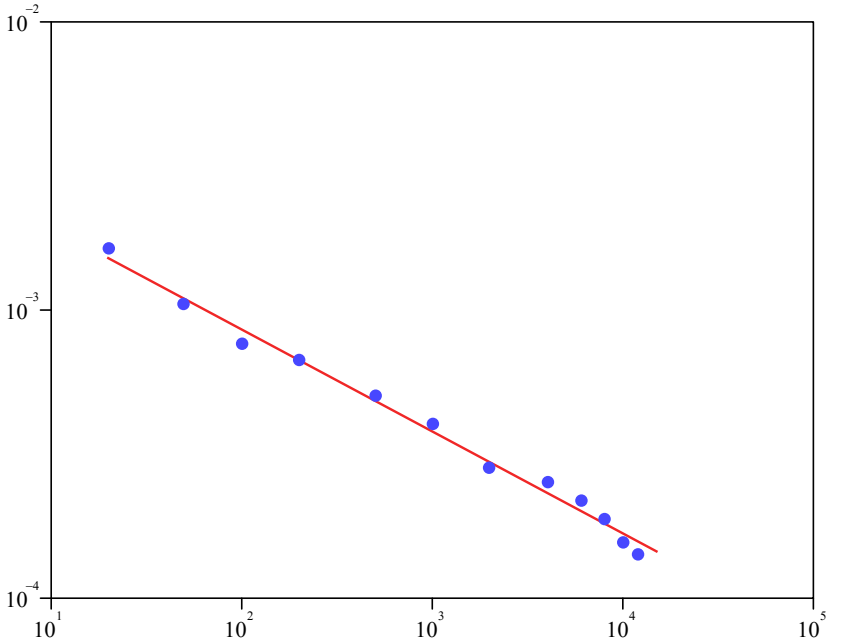


Figure 3. Log-log plot of error when approximating the loss of thickness in environment 2 versus number of points in the quantization grids. The empirical convergence rate, estimated through a regression model, is -0.35.

Figure 3 presents respectively the empirical convergence rate. The convergence rate, estimated through a regression model is -0.35. This is roughly the same order of magnitude as the rate of convergence of the optimal quantizer (see for instance [9]) since here the dimension is 3 (indeed, m is deterministic and $s = 0$ immediately after a jump so that we only quantize the variables ρ, d and t).

Finally, we show here the CPU time to compute the expectations from the quantization grids (computations are run with Matlab R2010b on a MacBook Pro 2.66 GHz i7 processor). The CPU time for 10^8 Monte Carlo simulations was approximately 16 000 s. It can be seen that, once the quantization grids are obtained, our approximation scheme performs very fast.

Quantization grids	CPU time (s)	Quantization grids	CPU time (s)
20 points	0.0059	2000 points	1.5
50 points	0.0085	4000 points	5.6
100 points	0.014	6000 points	13
200 points	0.034	8000 points	24
500 points	0.12	10000 points	35
1000 points	0.37	12000 points	54

7. Conclusion

We have presented an efficient and easy to implement numerical method to approximate expectations of functionals of piecewise deterministic Markov processes. We proved the convergence of our algorithm with bounds for the rate of convergence.

Although our method concerns time invariant functionals, we proved that we are able to tackle time-dependent problems such as Lipschitz continuous time-dependent functionals or deterministic time horizon expectations. Indeed, we proved that, thanks to the introduction of the time-augmented process, time-dependent problems may be seen, paradoxically, as special cases of the time invariant situation.

Our method is easy to implement because it merely requires to be able to simulate the process. Furthermore, although the computation of the quantization grids may be quite time-consuming, it may be performed preliminarily because the grids only depend on the dynamics of the process and not on the cost functions l and c. Therefore, they may be stored off-line and serve several purposes. As illustrated by the examples presented in Section 6, storing the grids provides to our approximation scheme efficiency and flexibility. Indeed, the computation of the expectation can be performed very quickly once the grids are available. Thus, if one decides for instance to modify the functional, the same grids may be used so that the new result is obtained very quickly. This flexibility is an important advantage over standard Monte Carlo simulations.

Appendix A. Lipschitz continuity of F, G and v_n

The first lemma and the first proposition of this section present mainly the Lipschitz continuity of the functions δ^A and F. They are stated without proof because they are quite straightforward.

Lemma A.1. *The function δ^A is Lipschitz continuous with respect to both its variables; i.e., for all $x, y \in E$ and $u, t \in \mathbb{R}$, one has*

$$|\delta^A(x,t) - \delta^A(y,t)| \leq A[t^*]|x-y|,$$
$$|\delta^A(x,t) - \delta^A(x,u)| \leq A|t-u|,$$

Moreover, one has for all $x \in E$ and $t, s \geq 0$ such that $t+s \leq t^*(x)$,

$$\delta^A(\Phi(x,s),t) = \delta^A(x,t+s).$$

Proposition A.2. *The function F introduced in Definition 3.3, is Lipschitz continuous with respect to both its variables. For all $x, y \in E$ and $u, v \in [0; t^*(x) \wedge t^*(y)]$, one has*

$$|F(x,u) - F(y,v)| \leq [F]_1|x-y| + [F]_2|u-v|,$$

with

$$[F]_1 = C_{t^*}[l]_1 + [c]_* + A[t^*]C_c, \quad [F]_2 = C_l + AC_c.$$

The next two lemmas are adapted from [6], the second one being a special case of Lemma A.1 there. Thus, they are stated without proof.

Lemma A.3. *For $h \in \boldsymbol{L}_c(E)$, $(x, y) \in E^2$, and $t \le t^*(x) \wedge t^*(y)$*

$$\left| \int_t^{t^*(x)} h(\Phi(x,s)) e^{-\Lambda(x,s)} ds - \int_t^{t^*(y)} h(\Phi(y,s)) e^{-\Lambda(y,s)} ds \right| \le \big(C_{t^*}[h]_1 + (C_{t^*}^2[\lambda]_1 + [t^*])C_h\big)|x-y|.$$

Lemma A.4. *For $h \in \boldsymbol{L}_c(\partial E) \cup \boldsymbol{L}_c(E)$ and $x, y \in E$, one has*

$$\left| e^{-\Lambda(x,t^*(x))} h\big(\Phi(x,t^*(x))\big) - e^{-\Lambda(y,t^*(y))} h\big(\Phi(y,t^*(y))\big) \right| \le \big([h]_* + C_h\big(C_{t^*}[\lambda]_1 + [t^*]C_\lambda\big)\big)|x-y|.$$

The following notation will be convenient later on. For $w \in \boldsymbol{L}_c(E)$, $x \in E$ and $t \in [0; t^*(x)]$, we define

$$\begin{aligned} G_t w(x) &= E_x\big[\big(F(x,S_1) + w(Z_1)\big)\mathbb{1}_{\{S_1 \ge t\}}\big] \\ &= E_x\big[\big(L(x,S_1) + C(x,S_1) + w(Z_1)\big)\mathbb{1}_{\{S_1 \ge t\}}\big]. \end{aligned}$$

In particular, $G_0 = G$. Since we know the law of (Z_1, S_1), it can be shown that

$$G_t w(x) = \Upsilon_1(x) + \Upsilon_2(x) + \Upsilon_3(x) + \Upsilon_4(x) + \Upsilon_5(x), \tag{14}$$

with

$$\begin{aligned} \Upsilon_1(x) &= e^{-\Lambda(x,t)} \int_0^t l \circ \Phi(x,s)\, ds, \\ \Upsilon_2(x) &= \int_t^{t^*(x)} l \circ \Phi(x,s) e^{-\Lambda(x,s)} ds. \\ \Upsilon_3(x) &= c \circ \Phi(x, t^*(x)) \int_t^{t^*(x)} \delta^A(x,s) \lambda \circ \Phi(x,s) e^{-\Lambda(x,s)} ds, \\ \Upsilon_4(x) &= \int_t^{t^*(x)} \big(\lambda Q w\big) \circ \Phi(x,s) e^{-\Lambda(x,s)} ds, \\ \Upsilon_5(x) &= e^{-\Lambda(x,t^*(x))} \big(Qw + c\big) \circ \Phi(x, t^*(x)). \end{aligned}$$

Proposition A.5. *For $w \in \boldsymbol{L}_c(E)$, $(x, y) \in E^2$ and $t \in [0; t^*(x) \wedge t^*(y)]$, one has*

$$\big|G_t w(x) - G_t w(y)\big| \le K(A, w)|x-y|,$$

where $K(A,w) = E_1 + E_2 A + E_3[w]_1 + E_4 C_w + [Q][w]_*$, *with*

$$\begin{aligned}
E_1 &= 2[l]_1 C_{t^*} + C_l\big([t^*] + 2C_{t^*}^2[\lambda]_1\big) + [c]_*(1 + C_{t^*}C_\lambda) \\
&\qquad + C_c\big(2[\lambda]_1 C_{t^*} + C_\lambda C_{t^*}^2[\lambda]_1 + 2[t^*]C_\lambda\big), \\
E_2 &= C_c C_{t^*} C_\lambda [t^*], \\
E_3 &= (1 + C_{t^*}C_\lambda)[Q], \\
E_4 &= 2C_\lambda[t^*] + C_{t^*}[\lambda]_1(2 + C_{t^*}C_\lambda).
\end{aligned}$$

Proof. Let $w \in \boldsymbol{L}_c(E)$, $(x,y) \in E^2$ and $t \in [0; t^*(x) \wedge t^*(y)]$. In view of (14), we naturally split $|G_t w(x) - G_y w(y)|$ into the sum of five differences.

The first one, $|\Upsilon_1(x) - \Upsilon_1(y)|$, is bounded by

$$\begin{aligned}
|\Upsilon_1(x) - \Upsilon_1(y)| &\le C_{t^*} C_l \big|e^{-\Lambda(x,t)} - e^{-\Lambda(y,t)}\big| + \int_0^t \big(l\circ\Phi(x,s) - l\circ\Phi(y,s)\big)\,ds \\
&\le \big(C_{t^*}^2 C_l[\lambda]_1 + C_{t^*}[l]_1\big)|x-y|.
\end{aligned}$$

The differences $|\Upsilon_2(x) - \Upsilon_2(y)|$ and $|\Upsilon_4(x) - \Upsilon_4(y)|$ can be bounded thanks to Lemma A.3, with successively $h = l$ and $h = \lambda Q w$. Notice that $C_{\lambda Q w} \le C_\lambda C_w$ and $[\lambda Q w]_1 \le C_\lambda [Q][w]_1 + C_w[\lambda]_1$.

For the difference of the Υ_5 terms, we use Lemma A.4 with $h = Qw + c$. Notice that $C_{Qw+c} \le C_w + C_c$ and that $[Qw + c]_* \le [Q]\big([w]_* + [w]_1\big) + [c]_*$.

Finally, to bound $|\Upsilon_3(x) - \Upsilon_3(y)|$, we assume without loss of generality that $t^*(x) \le t^*(y)$ and we have

$$\begin{aligned}
&|\Upsilon_3(x) - \Upsilon_3(y)| \\
&\le C_c \int_t^{t^*(x)} \big|\delta^A(x,s)\lambda\circ\Phi(x,s)e^{-\Lambda(x,s)} - \delta^A(y,s)\lambda\circ\Phi(y,s)e^{-\Lambda(y,s)}\big|\,ds \\
&+ C_c \int_{t^*(x)}^{t^*(y)} \big|\delta^A(y,s)\lambda\circ\Phi(y,s)e^{-\Lambda(y,s)}\big|\,ds + [c]_* C_{t^*} C_\lambda |x-y| \\
&\le C_c \int_t^{t^*(x)} \big(C_\lambda\big|\delta^A(x,s) - \delta^A(y,s)\big| + [\lambda]_1|x-y| + C_\lambda\big|e^{-\Lambda(x,s)} - e^{-\Lambda(y,s)}\big|\big)\,ds \\
&\qquad + C_c[t^*]C_\lambda|x-y| + [c]_* C_{t^*} C_\lambda |x-y| \\
&\le \big(C_c C_{t^*}\big(C_\lambda A[t^*] + [\lambda]_1 + C_\lambda C_{t^*}[\lambda]_1\big) + C_c[t^*]C_\lambda + [c]_* C_{t^*} C_\lambda\big)|x-y|.
\end{aligned}$$

The result follows. □

The next lemma is stated without proof, as it is very close to [5, Lemma 51.7].

Lemma A.6. *For all* $x \in E$ *and* $t \in [0; t^*(x)]$, *one has*

$$v_n(\Phi(x,t)) = e^{\Lambda(x,t)} G_t v_{n-1}(x) - \int_0^t l\circ\Phi(x,s)\,ds.$$

Proposition A.7. *For all $n \in \{0, 1, \dots, N\}$, one has $v_n \in \boldsymbol{L}_c(E)$ and*

$$
\begin{aligned}
C_{v_n} &\leq n\big(C_{t^*} C_l + C_c\big), \\
[v_n]_1 &\leq e^{C_{t^*} C_\lambda}\big(K(A, v_{n-1}) + n C_{t^*}[\lambda]_1 (C_{t^*} C_l + C_c)\big) + C_{t^*}[l]_1, \\
[v_n]_2 &\leq e^{C_{t^*} C_\lambda}\big(C_{t^*} C_l C_\lambda + 2C_l + C_\lambda C_c + (2n-1) C_\lambda (C_{t^*} C_l + C_c)\big) + C_l, \\
[v_n]_* &\leq [v_n]_1 + [t^*][v_n]_2, \\
[v_n] &\leq K(A, v_{n-1}),
\end{aligned}
$$

Proof. Recall that for $x \in E$, one has from Definition 3.3

$$
v_n(x) = G v_{n-1}(x) = \boldsymbol{E}_x\left[L(x, S_1)\right] + \boldsymbol{E}_x\left[C(x, S_1)\right] + \boldsymbol{E}_x[v_{n-1}(Z_1)].
$$

Thus, $C_{v_n} \leq C_{t^*} C_l + C_c + C_{v_{n-1}} \leq n\big(c_{t^*} C_l + C_c\big)$ by induction.

Let us now turn to $[v_n]_1$. Lemma A.6 yields

$$
\begin{aligned}
&|v_n(\Phi(x,t)) - v_n(\Phi(y,t))| \\
&\quad \leq |e^{\Lambda(x,t)} G_t v_{n-1}(x) - e^{\Lambda(y,t)} G_t v_{n-1}(y)| + \int_0^t \left|l \circ \Phi(x,s) - l \circ \Phi(y,s)\right| ds \\
&\quad \leq e^{\Lambda(x,t)}\left|G_t v_{n-1}(x) - G_t v_{n-1}(y)\right| + \left|G_t v_{n-1}(y)\right|\left|e^{\Lambda(x,t)} - e^{\Lambda(y,t)}\right| \\
&\qquad\qquad + C_{t^*}[l]_1 |x - y|.
\end{aligned}
$$

The result follows using Proposition A.5 and noticing that

$$
\begin{aligned}
\Lambda(x,t) &\leq C_{t^*} C_\lambda, \\
|G_t v_{n-1}(y)| &\leq C_{t^*} C_l + C_c + C_{v_{n-1}} \leq n(C_{t^*} C_l + C_c), \\
|e^{\Lambda(x,t)} - e^{\Lambda(y,t)}| &\leq e^{C_{t^*} C_\lambda} C_{t^*}[\lambda]_1 |x - y|.
\end{aligned}
$$

We now turn to $[v_n]_2$. For $x \in E$ and $s, t \in [0, t^*(x)]$ with $s \leq t$, one has

$$
\begin{aligned}
\left|v_n(\Phi(x,t)) - v_n(\Phi(x,s))\right| &\leq e^{\Lambda(x,t)}\left|G_t v_{n-1}(x) - G_s v_{n-1}(x)\right| \\
&\quad + \left|G_s v_{n-1}(x)\right|\left|e^{\Lambda(x,t)} - e^{\Lambda(x,s)}\right| + C_l |t - s|.
\end{aligned}
$$

Moreover, from (14), one has

$$
\begin{aligned}
&\left|G_t v_{n-1}(x) - G_s v_{n-1}(x)\right| \\
&\quad \le E_x\left[\left|F(x, S_1) + v_{n-1}(Z_1)\right| \mathbb{1}_{\{s \le S_1 < t\}}\right] \\
&\quad \le \left| e^{-\Lambda(x,t)} \int_0^t l(\Phi(x,u))\, du - e^{-\Lambda(x,s)} \int_0^s l(\Phi(x,u))\, du \right| \\
&\qquad\qquad + \int_s^t \left| l(\Phi(x,u)) e^{-\Lambda(x,u)} \right| du \\
&\qquad\qquad + \left| c \circ \Phi(x, t^*(x)) \right| \int_s^t \left| \delta^{\mathcal{A}}(x,u) \lambda \circ \Phi(x,u) e^{-\Lambda(x,u)} \right| du \\
&\qquad\qquad + \int_s^t \left| (\lambda Q v_{n-1}) \circ \Phi(x,u) e^{-\Lambda(x,u)} \right| du \\
&\quad \le \left(C_{t^*} C_l \left| e^{-\Lambda(x,t)} - e^{-\Lambda(x,s)} \right| + C_l |t-s| \right) + (C_l |t-s|) \\
&\qquad\qquad + (C_c C_\lambda |t-s|) + (C_\lambda C_{v_{n-1}} |t-s|)
\end{aligned}
$$

and

$$
|e^{\Lambda(x,t)} - e^{\Lambda(x,s)}| \le e^{C_{t^*} C_\lambda} C_\lambda |t-s|.
$$

Finally, the bound for $[v_n]$ is a direct consequence from Proposition A.5. □

Appendix B. Relaxed assumption on the running cost function

In this section, we consider the approximation applied to the time-augmented process so that the local characteristics are $\tilde{\Phi}$, $\tilde{\lambda}$ and $\tilde{Q}$ defined in Section 5.1. Moreover, we consider a function $l \in \boldsymbol{L}_c(\tilde{E})$ and we define $\tilde{l} \in B(\tilde{E})$ by

$$
\text{for all } \xi = (x,t) \in \tilde{E},\ \tilde{l}(\xi) = l(x,t) \mathbb{1}_{\{t \le t_f\}}.
$$

We intend to prove that the convergence of our approximation scheme, stated by Theorem 4.5, remains true if we choose $\tilde{l}$ as the running cost function even though it does not fulfill the required Lipschitz conditions, i.e., $\tilde{l} \notin \boldsymbol{L}_c(\tilde{E})$. Indeed, the Lipschitz continuity of l is used four times in the proof of the theorem, once in Proposition A.2, twice in Proposition A.5 (when bounding the difference of the Υ_1 terms and the one of the Υ_2 ones) and once in Proposition A.7 (when bounding $[v_n]_1$). In each case, the Lipschitz continuity of the running cost function l is used to bound a term of the form

$$
\int_s^{s'} \left| \tilde{l} \circ \tilde{\Phi}(\xi, u) - \tilde{l} \circ \tilde{\Phi}(\xi', u) \right| du \tag{15}
$$

for $\xi, \xi' \in \widetilde{E}$ and $s, s' \in [0; \tilde{t}^*(\xi) \wedge \tilde{t}^*(\xi')]$, or of the form

$$\int_s^{\tilde{t}^*(\xi)\wedge\tilde{t}^*(\xi')} \left|\tilde{l} \circ \widetilde{\Phi}(\xi, u) e^{-\widetilde{\Lambda}(\xi,u)} - \tilde{l} \circ \widetilde{\Phi}(\xi', u) e^{-\widetilde{\Lambda}(\xi',u)}\right| du \tag{16}$$

for $\xi, \xi' \in \widetilde{E}$ and $s \in [0; \tilde{t}^*(\xi) \wedge \tilde{t}^*(\xi')]$ and where we use the natural notation $\widetilde{\Lambda}(\xi, u) = \int_0^u \tilde{\lambda}(\widetilde{\Phi}(\xi, v))\, dv$. Concerning this second form, Equation (16), notice that

$$\begin{aligned}
&\int_s^{\tilde{t}^*(\xi)\wedge\tilde{t}^*(\xi')} \left|\tilde{l} \circ \widetilde{\Phi}(\xi, u) e^{-\widetilde{\Lambda}(\xi,u)} - \tilde{l} \circ \widetilde{\Phi}(\xi', u) e^{-\widetilde{\Lambda}(\xi',u)}\right| du \\
&\quad \le \int_s^{\tilde{t}^*(\xi)\wedge\tilde{t}^*(\xi')} \left|\tilde{l} \circ \widetilde{\Phi}(\xi, u) - \tilde{l} \circ \widetilde{\Phi}(\xi', u)\right| du \\
&\qquad\qquad + C_l \int_s^{\tilde{t}^*(\xi)\wedge\tilde{t}^*(\xi')} \left|e^{-\widetilde{\Lambda}(\xi,u)} - e^{-\widetilde{\Lambda}(\xi',u)}\right| du \\
&\quad \le \int_s^{\tilde{t}^*(\xi)\wedge\tilde{t}^*(\xi')} \left|\tilde{l} \circ \widetilde{\Phi}(\xi, u) - \tilde{l} \circ \widetilde{\Phi}(\xi', u)\right| du + C_l C_{t^*}^2 [\lambda]_1 |\xi - \xi'|,
\end{aligned}$$

so that, to ensure that Theorem 4.5 remains true with $\tilde{l}$ as the running cost function, it is sufficient to be able to bound terms of the form (15). This is done in the following lemma.

Lemma B.1. *For $\xi = (x, t), \xi' = (x', t') \in \widetilde{E}$ and $s \in [0; \tilde{t}^*(\xi) \wedge \tilde{t}^*(\xi')]$, one has*

$$\int_0^s \left|\tilde{l} \circ \widetilde{\Phi}(\xi, u) - \tilde{l} \circ \widetilde{\Phi}(\xi', u)\right| du \le (C_{t^*}[l]_1 + C_l)|\xi - \xi'|.$$

Proof. Let $\xi = (x, t), \xi' = (x', t') \in \widetilde{E}$ and $s \in [0; \tilde{t}^*(\xi) \wedge \tilde{t}^*(\xi')]$. One has

$$\begin{aligned}
&\int_0^s \left|\tilde{l} \circ \widetilde{\Phi}(\xi, u) - \tilde{l} \circ \widetilde{\Phi}(\xi', u)\right| du \\
&\quad \le \int_0^s \left|l \circ \widetilde{\Phi}(\xi, u) \mathbb{1}_{\{t+u\le t_f\}} - l \circ \widetilde{\Phi}(\xi', u) \mathbb{1}_{\{t'+u\le t_f\}}\right| du \\
&\quad \le \int_0^s \left|l \circ \widetilde{\Phi}(\xi, u) - l \circ \widetilde{\Phi}(\xi', u)\right| du + C_l \int_0^s \left|\mathbb{1}_{\{t+u\le t_f\}} - \mathbb{1}_{\{t'+u\le t_f\}}\right| du
\end{aligned}$$

The left-hand side term is bounded by $C_{t^*}[l]_1|\xi - \xi'|$ since $l \in \boldsymbol{L}_c(\widetilde{E})$. For the right-hand side term, assume without loss of generality that $t \le t'$, one has

$$\left|\mathbb{1}_{\{t+u\le t_f\}} - \mathbb{1}_{\{t'+u\le t_f\}}\right| = \left|\mathbb{1}_{\{t-t_f\le u\}} - \mathbb{1}_{\{t'-t_f\le u\}}\right| = \mathbb{1}_{\{t-t_f\le u<t'-t_f\}},$$

so that the right-hand side term is bounded by $C_l|t - t'| \le C_l|\xi - \xi'|$. The result follows. □

Theorem 4.5 remains true if we choose $\tilde{l}$ as the running cost function. One only needs to slightly modify the Lipschitz constants given in propositions A.2, A.5 and A.7. The terms $C_{t^*}[l]_1$ have to be replaced by $C_{t^*}[l]_1 + C_l$.

Appendix C. Proof of Theorem 4.5

The Lipschitz continuity of the functions v_k is proved by Proposition A.7. Now let $A > 0$ and notice that

$$|J_N(l,c)(x) - \widehat{V}_0| \le |J_N(l,c)(x) - V_0| + |V_0 - \widehat{V}_0|.$$

Proposition 3.2 says that $|J_N(l,c)(x) - V_0| \le NC_cC_\lambda/A$ since $V_0 = J_N^A(l,c)(x)$. We now have to bound $|V_0 - \widehat{V}_0|$.

Some of the arguments of the proof are similar to the ones used in Theorem 5.1 from [6], thus we will not develop the details of the proof. Recall that $\|V_N - \widehat{V}_N\|_p = 0$ and let $k \in \{0, \dots, N-1\}$. In order to bound the approximation error, let us split it into three terms $\|V_k - \widehat{V}_k\|_p \le \Xi_1 + \Xi_2 + \Xi_3$, where

$$\begin{aligned}
\Xi_1 &= \|v_k(Z_k) - v_k(\widehat{Z}_k)\|_p, \\
\Xi_2 &= \|Gv_{k+1}(\widehat{Z}_k) - \widehat{G}_{k+1}v_{k+1}(\widehat{Z}_k)\|_p, \\
\Xi_3 &= \|\widehat{G}_{k+1}v_{k+1}(\widehat{Z}_k) - \widehat{G}_{k+1}\hat{v}_{k+1}(\widehat{Z}_k)\|_p.
\end{aligned}$$

The theorem is then a direct consequence from the three following lemmas, stated without proof, that provide bounds for each of these three terms.

Lemma C.1. *The first term, Ξ_1, is bounded by*

$$\|v_k(Z_k) - v_k(\widehat{Z}_k)\|_p \le [v_k]\|Z_k - \widehat{Z}_k\|_p.$$

Lemma C.2. *The second term, Ξ_2, is bounded by*

$$\begin{aligned}
&\left\|Gv_{k+1}(\widehat{Z}_k) - \widehat{G}_{k+1}v_{k+1}(\widehat{Z}_k)\right\|_p \\
&\quad \le [v_{k+1}]\|Z_{k+1} - \widehat{Z}_{k+1}\|_p + \big([v_k] + [F]_1\big)\|Z_k - \widehat{Z}_k\|_p + [F]_2\|S_{k+1} - \widehat{S}_{k+1}\|_p.
\end{aligned}$$

Lemma C.3. *The third term, Ξ_3, is bounded by*

$$\begin{aligned}
&\left\|\widehat{G}_{k+1}v_{k+1}(\widehat{Z}_k) - \widehat{G}_{k+1}\hat{v}_{k+1}(\widehat{Z}_k)\right\|_p \\
&\qquad\qquad \le [v_{k+1}]\|Z_{k+1} - \widehat{Z}_{k+1}\|_p + \|V_{k+1} - \widehat{V}_{k+1}\|_p.
\end{aligned}$$

Acknowledgements

This work was supported by ARPEGE program of the French National Agency of Research (ANR), project "FAUTOCOES", number ANR-09-SEGI-004. Besides, the authors gratefully acknowledge Astrium for its financial support.

References

[1] V. Bally and G. Pagès, *A quantization algorithm for solving multi-dimensional discrete-time optimal stopping problems*, Bernoulli **9** (2003), no. 6, 1003–1049. MR 2005f:60096

[2] V. Bally, G. Pagès, and J. Printems, *A quantization tree method for pricing and hedging multidimensional American options*, Math. Finance **15** (2005), no. 1, 119–168. MR 2005k:91142

[3] A. Brandejsky, B. de Saporta, and F. Dufour, *Numerical methods for the exit time of a piecewise-deterministic markov process*, Adv. Appl. Probab. **44** (2012), no. 1. arXiv 1012.2659

[4] C. Cocozza-Thivent, R. Eymard, and S. Mercier, *A finite-volume scheme for dynamic reliability models*, IMA J. Numer. Anal. **26** (2006), no. 3, 446–471. MR 2008h:60370 Zbl 1109.65011

[5] M. H. A. Davis, *Markov models and optimization*, Monographs on Statistics and Applied Probability, no. 49, Chapman & Hall, London, 1993. MR 96b:90002 Zbl 0780.60002

[6] B. de Saporta, F. Dufour, and K. Gonzalez, *Numerical method for optimal stopping of piecewise deterministic Markov processes*, Ann. Appl. Probab. **20** (2010), no. 5, 1607–1637. MR 2011g:60079 Zbl 1197.93162

[7] R. Eymard, S. Mercier, and A. Prignet, *An implicit finite volume scheme for a scalar hyperbolic problem with measure data related to piecewise deterministic Markov processes*, J. Comput. Appl. Math. **222** (2008), no. 2, 293–323. MR 2009j:65194 Zbl 1158.65008

[8] R. M. Gray and D. L. Neuhoff, *Quantization*, IEEE Trans. Inform. Theory **44** (1998), no. 6, 2325–2383. MR 99i:94029 Zbl 1016.94016

[9] G. Pagès, H. Pham, and J. Printems, *Optimal quantization methods and applications to numerical problems in finance*, Handbook of computational and numerical methods in finance, Birkhäuser, Boston, 2004, pp. 253–297. MR 2083055 Zbl 1138.91467

Received May 6, 2011. Revised November 8, 2011.

ADRIEN BRANDEJSKY: adrien.brandejsky@math.u-bordeaux1.fr
INRIA Bordeaux Sud-Ouest, team CQFD, F-33400 Talence, France

and

Université de Bordeaux, IMB, UMR 5251, F-33400 Talence, France

and

CNRS, IMB, UMR 5251, F-33400 Talence, France

BENOÎTE DE SAPORTA: saporta@math.u-bordeaux1.fr
Université de Bordeaux, Gretha, UMR 5113, F-33400 Talence, France

and

CNRS, Gretha, UMR 5113, IMB, UMR 5251, F-33400 Talence, France

and

INRIA Bordeaux Sud-Ouest, team CQFD, F-33400 Talence, France

FRANÇOIS DUFOUR: francois.dufour@math.u-bordeaux1.fr
Université de Bordeaux, IMB, UMR 5251, F-33400 Talence, France

and

CNRS, IMB, UMR 5251, F-33400 Talence, France

and

INRIA Bordeaux Sud-Ouest, team CQFD, F-33400 Talence, France

COMM. APP. MATH. AND COMP. SCI.
Vol. 7, No. 1, 2012

msp

TOWARD AN EFFICIENT PARALLEL IN TIME METHOD FOR PARTIAL DIFFERENTIAL EQUATIONS

MATTHEW EMMETT AND MICHAEL L. MINION

A new method for the parallelization of numerical methods for partial differential equations (PDEs) in the temporal direction is presented. The method is iterative with each iteration consisting of deferred correction sweeps performed alternately on fine and coarse space-time discretizations. The coarse grid problems are formulated using a space-time analog of the full approximation scheme popular in multigrid methods for nonlinear equations. The current approach is intended to provide an additional avenue for parallelization for PDE simulations that are already saturated in the spatial dimensions. Numerical results and timings on PDEs in one, two, and three space dimensions demonstrate the potential for the approach to provide efficient parallelization in the temporal direction.

1. Introduction

The last decade has seen an increase in research into the parallelization of numerical methods for ordinary and partial differential equations in the temporal direction beginning with the introduction of the parareal algorithm in 2001 [20] and the related PITA scheme in 2003 [11]. Both parareal and PITA are iterative methods where, in each iteration, each processor (corresponding to distinct time steps) uses both an accurate (or fine) method and a less computationally expensive (or coarse) method to propagate an improved solution through the time domain. Parallel speedup can be achieved because the fine solutions can be computed in parallel.

The main drawback of the parareal algorithm is that it has low parallel efficiency. Specifically, the parallel efficiency is formally bounded above by $1/K$, where K

Emmett was supported by the Director, DOE Office of Science, Office of Advanced Scientific Computing Research, Office of Mathematics, Information, and Computational Sciences, Applied Mathematical Sciences Program, under contract DE-SC0004011. Minion's work was supported by the Alexander von Humboldt Foundation and the Director, DOE Office of Science, Office of Advanced Scientific Computing Research, Office of Mathematics, Information, and Computational Sciences, Applied Mathematical Sciences Program, under contract DE-SC0004011 and by the National Science Foundation under contract DMS-0854961.

MSC2010: 65M99.

Keywords: parallel computing, time parallel, ordinary differential equations, partial differential equations, deferred corrections, parareal.

is the number of iterations needed to converge to the desired accuracy. Since K must be at least 2 for any meaningful stopping criteria, the efficiency of parareal is always less than $\frac{1}{2}$, and in practice can be much worse, particularly if many processors are used, or high temporal accuracy is desired.

For the parallel solution of partial differential equations (PDEs), parallelization in the spatial dimensions is well established, and hence temporal parallelization is only attractive if the temporal parallel efficiency exceeds that of (additional) spatial parallelization. In [21; 24] a method for the parallelization of ordinary differential equations (ODEs) is presented, similar in structure to the parareal method but utilizing a defect or deferred correction strategy in place of standard methods for ODEs as in parareal. Both the fine and coarse propagators in parareal are cast as spectral deferred correction (SDC) [10] sweeps using different temporal resolutions to improve the solution on each time step. Hence the method described in [21; 24] can be heuristically thought of in two different ways:

(1) A modification of the parareal algorithm that replaces direct solves in each iteration with a deferred correction procedure applied to solutions generated at previous iterations to reduce the cost of each parareal iteration.

(2) A time parallel version of the SDC method that incorporates a coarse and fine temporal discretization to achieve better parallel efficiency.

In this paper, the ideas introduced in [21; 24] are extended to the temporal parallelization of partial differential equations (PDEs). The key difference between the ODE method and the PDE method is that coarsening can be done in both space and time in the coarse discretization. This observation has been made previously for both the parareal and hybrid parareal/SDC methods [3; 2; 12; 13; 24], although details of how best to translate information from the coarse and fine discretizations have not been extensively explored. Furthermore, for the parareal method, reducing the cost of the coarse propagator does not alter the fact that the parallel efficiency is bounded by $1/K$. Here we present a procedure for using coarse grid information based on the full approximation scheme (FAS) technique developed for the solution of nonlinear equations by multigrid methods [8] that increases the accuracy of the coarse grid SDC sweeps. Hence the method introduced here can be considered a *parallel full approximation scheme in space and time* (or PFASST for short).

The numerical techniques used in the construction of the PFASST method are reviewed in Section 2, and the details of how these techniques are synthesized to construct the PFASST algorithm are outlined in Section 3. The computational cost and theoretical parallel efficiency and speedup of the PFASST algorithm is discussed in Section 4, followed by numerical results confirming the convergence properties and efficiency in Section 5. Finally, a short discussion of the current results and future research directions can be found in Section 6.

2. Method components

In this section, the components used to construct the PFASST algorithm are reviewed. First, a short description of spectral deferred corrections is presented and the method is cast in a concise notational formulation used later on. Then a description of the parareal and hybrid parareal/SDC method from [21; 24] is provided. Finally a short review of the full approximation scheme is included.

2.1. ***Spectral deferred corrections.*** Spectral deferred correction (SDC) methods are variants of the traditional defect correction methods (or the closely related deferred correction methods) for ODEs introduced in the 1960s [30; 26; 27; 9; 29; 4]. SDC is introduced in [10], and the method has been modified and analyzed extensively since (see [22; 23; 19; 18; 17], for instance).

For the following description, consider the ODE initial value problem

$$u'(t) = f(t, u(t)), \qquad u(0) = u_0, \tag{1}$$

where $t \in [0, T]$; $u_0,\ u(t) \in \mathbb{C}^N$; and $f : \mathbb{R} \times \mathbb{C}^N \to \mathbb{C}^N$. In the numerical examples presented in Section 5, a method of lines discretization based on a pseudospectral approach is used to reduce the PDE in question to a large system of ODEs. To describe SDC, it is convenient to use the equivalent Picard integral form of (1):

$$u(t) = u_0 + \int_0^t f\big(\tau, u(\tau)\big)\, d\tau. \tag{2}$$

As with traditional deferred correction methods, a single time step $[t_n, t_{n+1}]$ is divided into a set of intermediate substeps by defining intermediate points $t_m \in [t_n, t_{n+1}]$. In SDC, the intermediate points t_m correspond to the nodes in Gaussian quadrature rules. Here Gauss–Lobatto rules are used so that $\boldsymbol{t} = [t_0, \ldots, t_M]$ (with $t_n = t_0 < \cdots < t_M = t_{n+1}$) corresponds to the Gauss–Lobatto quadrature rule with $M + 1$ nodes (which has order $2M$).

SDC constructs higher-order accurate solutions within one full time step by iteratively approximating a series of correction equations at the intermediate nodes using lower-order methods. One attractive feature of SDC methods is that, since only lower-order methods are required, one can construct methods that employ operator splitting and/or multirate time-stepping and still achieve higher-order accuracy. (See [6; 22; 18; 7], for instance.)

The SDC method begins by computing a provisional solution $\boldsymbol{U}^0 = [U_1^0, \ldots, U_M^0]$, at each of the intermediate nodes with $U_m^0 \approx u(t_m)$. As described below, this initial approximation can be simply the solution at the beginning of the time step replicated at each node. The method then proceeds iteratively. Let $\boldsymbol{U}^k = [U_1^k, \ldots, U_M^k]$ denote the vector of solution values at each point $t_1 \ldots t_M$ and SDC iteration k, and $\boldsymbol{F}^k = [f(t_0, U_0^k), \ldots, f(t_M, U_M^k)]$ the vector of function values at each point

$t_0 \dots t_M$ and SDC iteration k. Note that $\boldsymbol{U}^k$ has M entries but $\boldsymbol{F}^k$ has $M+1$. To compute the approximations $\boldsymbol{U}^{k+1}$, one first computes the approximate integrals

$$\boldsymbol{S}_m^{m+1}\boldsymbol{F}^k = \sum_{j=0}^{M} q_{m,j} f(t_j, U_j^k) \approx \int_{t_m}^{t_{m+1}} f\big(\tau, U^k(\tau)\big)\, d\tau \tag{3}$$

for $m = 0 \dots M-1$. These approximations can be calculated by a matrix-vector multiplication by the $M \times M+1$ spectral integration matrix $\boldsymbol{S}$ with entries $q_{m,j}$. For a system of ODEs of size N, the integration matrix is applied component-wise, and hence $\boldsymbol{S}$ must be defined as the Kronecker product of the scalar integration matrix with the $N \times N$ identity matrix (see discussion in [14]).

Using these values, a first-order implicit time-stepping method similar to backward Euler for computing U^{k+1} at each substep can be written

$$U_{m+1}^{k+1} = U_m^{k+1} + \Delta t_m \big[f(t_{m+1}, U_{m+1}^{k+1}) - f(t_{m+1}, U_{m+1}^k) \big] + \boldsymbol{S}_m^{m+1}\boldsymbol{F}^k, \tag{4}$$

where $\Delta t_m = t_{m+1} - t_m$. The computational cost of each substep is essentially that of backward Euler (although if an iterative method is used to solve the implicit system, a very good initial guess is provided by the solution at iteration k). The process of solving (4) at each node t_m is referred to here as an *SDC sweep*.

The accuracy of the solution generated after k SDC sweeps done with such a first-order method is formally $O(\Delta t^k)$ as long as the spectral integration rule (here Gauss–Lobatto) is at least order k. In fact, if SDC converges, it converges to the solution of the spectral collocation or implicit Runge–Kutta method

$$\boldsymbol{U} = \boldsymbol{U}_0 + \Delta t\, \boldsymbol{S}\boldsymbol{F}, \tag{5}$$

where $\boldsymbol{U}_0 = [U_0, \dots, U_0]$. Hence SDC can be considered as an iterative method for solving the spectral collocation formulation. (See [14] for a technique to accelerate this convergence.)

For equations which can be split into stiff and nonstiff pieces, the above method is easily modified to create semi-implicit or IMEX schemes. Consider the ODE

$$u'(t) = f\big(t, u(t)\big) = f_E\big(t, u(t)\big) + f_I\big(t, u(t)\big), \qquad u(0) = u_0. \tag{6}$$

The first term on the right-hand side is assumed to be nonstiff (and hence treated explicitly), and the second is assumed to be stiff (and hence treated implicitly). Equation (4) can be easily modified to give a semi-implicit scheme:

$$\begin{aligned} U_{m+1}^{k+1} = U_m^{k+1} &+ \Delta t_m \big[f_I(t_{m+1}, U_{m+1}^{k+1}) - f_I(t_{m+1}, U_{m+1}^k) \big] \\ &+ \Delta t_m \big[f_E(t_m, U_m^{k+1}) - f_E(t_m, U_m^k) \big] + \boldsymbol{S}_m^{m+1}\boldsymbol{F}^k. \end{aligned} \tag{7}$$

Note that unlike semi-implicit or IMEX schemes based on Runge–Kutta (see [28; 1; 25; 15; 5], for instance), it is straightforward to construct semi-implicit

SDC schemes with very high formal order of accuracy. This semi-implicit form is used for all of the numerical examples presented in Section 5.

Following the ideas introduced in [14], we can write an SDC sweep more compactly by using matrix notation. Specifically, one can write all M steps of the forward Euler time-stepping scheme as

$$\boldsymbol{U} = \boldsymbol{U}_0 + \Delta t \boldsymbol{S}_E \boldsymbol{F}, \tag{8}$$

where the $M \times M+1$ matrix $\boldsymbol{S}_E$ has entries

$$\boldsymbol{S}_E(i,j) = \begin{cases} \Delta t_j / \Delta t & \text{if } j \le i, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Likewise, all M steps of the backward Euler scheme can be written

$$\boldsymbol{U} = \boldsymbol{U}_0 + \Delta t \boldsymbol{S}_I \boldsymbol{F}, \tag{10}$$

where the $M \times M+1$ matrix $\boldsymbol{S}_I$ has entries

$$\boldsymbol{S}_I(i,j) = \begin{cases} \Delta t_{j-1} / \Delta t & \text{if } 1 < j \le i+1, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The matrices $\boldsymbol{S}_E$ and $\boldsymbol{S}_I$ are also first order approximations to the integration matrix $\boldsymbol{S}$ that arise from using either the left-hand or right-hand rectangle rule approximation to the integrals $\boldsymbol{S}_m^{m+1}$ defined in (3).

Furthermore, let $\tilde{\boldsymbol{S}}_E = \boldsymbol{S} - \boldsymbol{S}_E$ and $\tilde{\boldsymbol{S}}_I = \boldsymbol{S} - \boldsymbol{S}_I$. Then, one SDC sweep using the semi-implicit time-stepping scheme in (7) can be compactly expressed as

$$\boldsymbol{U}^{k+1} = \boldsymbol{U}_0 + \Delta t \boldsymbol{S}_E \boldsymbol{F}^{k+1} + \Delta t \boldsymbol{S}_I \boldsymbol{F}^{k+1} + \Delta t (\tilde{\boldsymbol{S}}_E + \tilde{\boldsymbol{S}}_I) \boldsymbol{F}^k. \tag{12}$$

2.2. ***Parareal and SDC.*** The parareal method for the temporal parallelization of ODEs and PDEs was introduced in 2001 by Lions, Maday, and Turinici [20] and has sparked renewed interest in the construction of time parallel methods. In the parareal method, the time interval of interest $[0, T]$ is divided into N intervals with each interval being assigned to a different processor. On each interval $[t_n, t_{n+1}]$ for $n = 0 \ldots N-1$, the parareal method iteratively computes a succession of approximations $U_{n+1}^k \approx u(t_{n+1})$, where k denotes the iteration number.

The parareal algorithm can be described in terms of two numerical approximation methods typically denoted by $\mathcal{G}$ and $\mathcal{F}$. Both $\mathcal{G}$ and $\mathcal{F}$ propagate an initial value $U_n \approx u(t_n)$ by approximating the solution to (2) from t_n to t_{n+1} and can in principle be any self-starting ODE method. However, in order for the parareal method to be efficient, it must be the case that the $\mathcal{G}$ propagator is computationally less expensive than the $\mathcal{F}$ propagator, and hence $\mathcal{G}$ is typically a low-order method. Since the overall accuracy of parareal is limited by the accuracy of the $\mathcal{F}$ propagator,

$\mathcal{F}$ is typically higher-order and in addition may use a smaller time step than $\mathcal{G}$. For these reasons, $\mathcal{G}$ is referred to as the coarse propagator and $\mathcal{F}$ the fine propagator.

The parareal method begins by computing a first approximation in serial, U_{n+1}^0 for $n = 0 \ldots N-1$, often performed with the coarse propagator $\mathcal{G}$, i.e.,

$$U_{n+1}^0 = \mathcal{G}(t_{n+1}, t_n, U_n^0) \tag{13}$$

with $U_0^0 = u(0)$. Alternatively, one could use the parareal method with a coarser time discretization to compute the initial approximation [2]. The parareal method proceeds iteratively, alternating between the parallel computation of $\mathcal{F}(t_{n+1}, t_n, U_n^k)$ and an update of the initial conditions at each processor of the form

$$U_{n+1}^{k+1} = \mathcal{G}(t_{n+1}, t_n, U_n^{k+1}) + \mathcal{F}(t_{n+1}, t_n, U_n^k) - \mathcal{G}(t_{n+1}, t_n, U_n^k) \tag{14}$$

for $n = 0 \ldots N-1$. Although this step has serial dependencies, the computations on independent processors can be scheduled so that each processor can begin the computation of the new $\mathcal{G}$ value $\mathcal{G}(t_{n+1}, t_n, U_n^{k+1})$ as soon as $\mathcal{F}(t_{n+1}, t_n, U_n^k)$ has been computed and the new starting value U_n^{k+1} has been received from processor $n-1$ [24]. The calculation in (14), which requires computing the $\mathcal{G}$ propagator, is referred to as here the $\mathcal{G}$ correction sweep.

Note that after k iterations of the parareal method, the solution U_m^k for $m \leq k$ is exactly equal to the numerical solution given by using the $\mathcal{F}$ propagator in a serial manner. Hence after N iterations the parareal solution is exactly equal to applying $\mathcal{F}$ in serial, but in practice the iterations converge more quickly for large N. Since each iteration of the parareal method requires the application of both $\mathcal{F}$ and $\mathcal{G}$ (plus the cost of communication between processors), the parareal method can only provide parallel speedup compared to the using $\mathcal{F}$ in serial if the number of iterations required to converge to the specified criteria (denoted here by K) is significantly less than N.

The dominant cost of the parareal method is the computation of $\mathcal{F}$ in each iteration. It has been well documented that the parallel efficiency of parareal is bounded by $1/K$ where K is the number of iterations needed to converge. In [21; 24] a hybrid parareal/SDC method is introduced that replaces the $\mathcal{F}$ propagator in parareal with a deferred correction sweep using the solution from the last iteration (and the new initial value U_n^{k+1}). This reduces the cost of the $\mathcal{F}$ propagator in two ways. First, instead of using many steps of a standard method like Runge–Kutta, the same level of accuracy can be achieved by one step of SDC due to the spectral accuracy of SDC methods. Second, the SDC sweep has a computational cost of approximately M steps of a first-order method, rather than many steps of a higher-order method. In effect, the high-cost of SDC *per time step* is amortized over the parallel iterations.

Furthermore, a connection between the parareal correction sweep defined by (14) and an SDC sweep is explained in [21; 24]. Hence the hybrid parareal/SDC method also casts the $\mathcal{G}$ correction sweep in (14) as an SDC sweep, which not only provides an updated starting value for the next processor, but can also be used to further improve the solution in the interval $[t_n, t_{n+1}]$. The numerical experiments in [24] suggest that the hybrid parareal/SDC strategy has similar convergence behavior as standard parareal, but with a reduced parallel cost and higher parallel efficiency. Specifically, the parallel efficiency (as compared to the serial SDC method) is bounded not by $1/K$, but K_s/K_p where K_s is the number of iterations required of the serial SDC method to converge to a given tolerance and K_p is the number of iterations for the parallel iterations to converge.

Note there are some disadvantages to the parareal/SDC hybrid approach. Because $\mathcal{F}$ and $\mathcal{G}$ in parareal are only used to provide solutions at the values t_n, parareal can be used in a "black-box" fashion with standard time integration methods. The gain in efficiency in the parareal/SDC approach requires that SDC be adopted as the time integration method, although there is still a great deal of flexibility in how the coarse and fine SDC sweeps are formulated (that is in fact one of the aforementioned advantages of SDC). Also, the parareal/SDC approach requires that both coarse and fine function values be stored at the intermediate nodes (however, the storage required is similar to that of higher-order Runge–Kutta methods).

2.3. ***Full approximation scheme.*** As mentioned above, when parallelizing PDEs in the temporal direction, an obvious way to reduce the cost of the coarse propagator in parareal or a hybrid parareal/SDC approach is to reduce both the temporal and spatial resolution. In the hybrid parareal/SDC method, it is desirable to use information from the coarse SDC sweep to not only improve the initial condition passed forward in time to the next processor, but also to improve the fine resolution solution on the same processor. To do so, the coarse resolution problem must be initialized including fine information, and the coarse resolution solution must be interpolated somehow (in both time and space) once computed. In the PFASST algorithm described in the next section, the coarse and fine resolution solutions are connected in the same manner as the full approximation scheme (FAS) method popular in multigrid methods for nonlinear problems (see [8], for instance). In fact, although only two resolutions are used here, the PFASST method could be extended to use a hierarchy of fine and coarse space-time grids, reminiscent of multigrid methods. Results along these lines will be reported in the future.

To review the FAS procedure, consider a nonlinear equation of the form

$$A(\boldsymbol{x}) = \boldsymbol{b}, \tag{15}$$

where the solution vector $\boldsymbol{x}$ and the right side $\boldsymbol{b}$ correspond to spatial discretizations

of some function. Given an approximate solution $\tilde{\boldsymbol{x}}$, the corresponding residual equation is

$$A(\tilde{\boldsymbol{x}} + \boldsymbol{e}) = \boldsymbol{r} + A(\tilde{\boldsymbol{x}}), \tag{16}$$

where $\boldsymbol{e}$ is the error and $\boldsymbol{r} = \boldsymbol{b} - A(\tilde{\boldsymbol{x}})$ is the residual. In a multigrid approach, the residual equation (16) is solved on a coarser discretization level by introducing an operator T_F^G that restricts solutions at the fine resolution to the coarse. Then, assuming A^G is an appropriate approximation to A on the coarse level, the coarse residual equation becomes

$$A^G(\tilde{\boldsymbol{x}}^G + \boldsymbol{e}^G) = A^G(\tilde{\boldsymbol{x}}^G) + \boldsymbol{r}^G = A^G(\tilde{\boldsymbol{x}}^G) + T_F^G\big(\boldsymbol{b} - A(\tilde{\boldsymbol{x}})\big) \tag{17}$$

$$= \boldsymbol{b}^G + A^G(\tilde{\boldsymbol{x}}^G) - T_F^G A(\tilde{\boldsymbol{x}}), \tag{18}$$

where superscript G denotes the coarse level. With $\boldsymbol{y}^G = \tilde{\boldsymbol{x}}^G + \boldsymbol{e}^G$, the coarse FAS residual equation becomes

$$A^G(\boldsymbol{y}^G) = \boldsymbol{b}^G + \boldsymbol{\tau}, \tag{19}$$

with the FAS correction term

$$\boldsymbol{\tau} = A^G(\tilde{\boldsymbol{x}}^G) - T_F^G A(\tilde{\boldsymbol{x}}). \tag{20}$$

The addition of the FAS correction allows the coarse solution to attain a similar degree of accuracy as the fine solution, but at the resolution of the coarse level [8]. In particular, if the fine residual is zero (i.e., $\tilde{\boldsymbol{x}}$ is the fine solution), the FAS corrected coarse equation (19) becomes $A^G(\boldsymbol{y}^G) = A^G(\tilde{\boldsymbol{x}}^G)$, and the coarse solution $\boldsymbol{y}^G$ is the restriction of the fine solution.

Once the coarse solution $\boldsymbol{y}^G$ has been computed, the fine approximate solution is improved using an interpolation operator T_G^F

$$\tilde{\boldsymbol{x}} = T_G^F(\boldsymbol{y}^G - \tilde{\boldsymbol{x}}^G). \tag{21}$$

Returning to SDC methods, the FAS correction for coarse SDC iterations is determined by considering SDC as an iterative method for solving the collocation formulation given by (5), which can be written

$$\boldsymbol{U} - \Delta t\, \boldsymbol{S}\boldsymbol{F} = \boldsymbol{U}_0, \tag{22}$$

where $\boldsymbol{U}_0$, $\boldsymbol{S}$, and $\boldsymbol{F}$ are defined as in Section 2.1. Therefore, combining (20) and (22), the FAS correction for coarse SDC iterations is given by

$$\boldsymbol{\tau} = \Delta t\big(\boldsymbol{S}^G \boldsymbol{F}^G - T_F^G \boldsymbol{S}\boldsymbol{F}\big), \tag{23}$$

where $\boldsymbol{S}^G$ is the integration matrix defined by the coarse nodes, $\boldsymbol{F}^G$ is the vector of function values at the coarse level, and T_F^G is a space-time restriction operator. This allows the coarse SDC iterations to achieve the accuracy of the fine SDC iterations at the resolution of the coarse level, and ultimately allows the PFASST

algorithm to achieve similar accuracy as a serial computation performed on the fine level. The numerical experiments in Section 5 confirm the benefit of using the FAS correction term in the coarse SDC sweep.

3. PFASST

At this point we have reviewed the main ingredients of the PFASST algorithm: SDC, time parallel iterations, and FAS. Now we describe how these ingredients are combined to form the PFASST algorithm. As in parareal, the time interval of interest $[0, T]$ is divided into N uniform intervals $[t_n, t_{n+1}]$ which are assigned to the processors $\boldsymbol{P}_n$ where $n = 0 \dots N-1$. Each interval is subdivided by defining $M+1$ fine SDC nodes $\boldsymbol{t}_n = [t_{n,0} \cdots t_{n,M}]$ such that $t_n = t_{n,0} < \cdots < t_{n,M} = t_{n+1}$; and $\tilde{M}+1$ coarse SDC nodes $\tilde{\boldsymbol{t}}_n$ such that $t_n = \tilde{t}_{n,0} < \cdots < \tilde{t}_{n,\tilde{M}} = t_{n+1}$. The coarse SDC nodes $\tilde{\boldsymbol{t}}_n$ are chosen to be a subset of the fine SDC nodes $\boldsymbol{t}_n$ to facilitate interpolation and restriction between the coarse and fine levels. The solution at the m-th fine node on processor $\boldsymbol{P}_n$ during iteration k is denoted $U^k_{n,m}$. Similarly, the solution at the $\tilde{m}$-th coarse node on processor $\boldsymbol{P}_n$ during iteration k is denoted $\tilde{U}^k_{n,\tilde{m}}$. For brevity let

$$\boldsymbol{U}^k_n = [U^k_{n,1}, \cdots, U^k_{n,M}] \quad \text{and} \quad \boldsymbol{F}^k_n = [f(t_{n,0}, U^k_{n,0}), \cdots, f(t_{n,M}, U^k_{n,M})],$$

with analogous notation for the coarse level (marked with a tilde). Note that the use of point injection as the coarsening procedure with Gaussian quadrature nodes means that the coarse nodes may not correspond to Gaussian nodes. This is further discussed at the beginning of Section 5.

3.1. ***Initialization.*** In the parareal method, the processors are typically initialized by using the coarse grid propagator in serial to yield a low-accuracy initial condition for each processor. This means in practice that all processors except the first are idle until passed an initial condition from the previous processor. Here we employ a different initialization scheme wherein each processor begins coarse SDC sweeps during this idle time. Hence the number of coarse iterations (SDC sweeps) done on processor $\boldsymbol{P}_n$ in the initialization is equal to n rather than 1. This has the same total computational cost of doing one SDC sweep per processor in serial, but the additional SDC sweeps can improve the accuracy of the solution significantly, as will be demonstrated in Section 5.

Specifically, the initial data $u(0)$ is spread to each processor $\boldsymbol{P}_n$ and stored in the fine level at each fine SDC node so that $U^0_{n,m} = u(0)$ for $n = 0 \dots N-1$ and $m = 0 \dots M$. Next, the fine values $\boldsymbol{U}^0_n$ are restricted to the coarse level and stored in $\tilde{\boldsymbol{U}}^{0,0}_n$, where the two superscripts denote PFASST iteration and initialization iteration, respectively. Before beginning the initialization iterations, the function values $\boldsymbol{F}^0_n$ and $\tilde{\boldsymbol{F}}^{0,0}_n$ are computed and used to form the first FAS correction $\boldsymbol{\tau}^0_n$.

The initialization iterations for $j = 1 \ldots n$ on processor $\boldsymbol{P}_n$ are comprised of the following steps:

(1) Receive the new initial value $\tilde{U}_{n,0}^{0,j}$ from processor $\boldsymbol{P}_{n-1}$ if $n > 0$ and $j > 1$.

(2) Perform one or more coarse SDC sweeps using the values $\tilde{\boldsymbol{F}}_n^{0,j-1}$ computed previously and the FAS correction $\boldsymbol{\tau}_n^0$. This will yield updated values $\tilde{\boldsymbol{U}}_n^{0,j}$ and $\tilde{\boldsymbol{F}}_n^{0,j}$.

(3) Send $\tilde{U}_{n,\tilde{M}}^{0,j}$ to processor $\boldsymbol{P}_{n+1}$ (if $n < N-1$). This will be received as the new initial condition $\tilde{U}_{n+1,0}^{0,j+1}$ in the next iteration.

After processor $\boldsymbol{P}_n$ is finished computing the value $\tilde{U}_{n,\tilde{M}}^{0,n}$ and sending it to $\boldsymbol{P}_{n+1}$, the correction $\tilde{\boldsymbol{U}}_n^{0,n} - \tilde{\boldsymbol{U}}_n^{0,0}$ is interpolated to the fine grid to yield the initial value $\boldsymbol{U}_n^0$. The PFASST iterations on each processor are then begun immediately with this initial value.

3.2. ***PFASST iterations.*** The PFASST iterations for $k = 1 \ldots K$ on each processor $\boldsymbol{P}_n$ proceed as follows. Assuming that the fine solution and function values $\boldsymbol{U}_n^{k-1}$ and $\boldsymbol{F}_n^{k-1}$ are available, the iterations are comprised of the following steps:

(1) Perform one fine SDC sweep using the values $\boldsymbol{F}_n^{k-1}$. This will yield provisional updated values $\boldsymbol{U}_n^{k'}$ and $\boldsymbol{F}_n^{k'}$.

(2) Restrict the fine values $\boldsymbol{U}_n^{k'}$ to the coarse nodes to form $\tilde{\boldsymbol{U}}_n^{k'}$ and compute $\tilde{\boldsymbol{F}}_n^{k'}$.

(3) Compute the FAS correction $\boldsymbol{\tau}_n^k$ using $\boldsymbol{F}_n^{k'}$ and $\tilde{\boldsymbol{F}}_n^{k'}$.

(4) Receive the new initial value $U_{n,0}^k$ from processor $\boldsymbol{P}_{n-1}$ if $n > 0$.

(5) Perform n_G coarse SDC sweeps beginning with the values $\tilde{\boldsymbol{F}}_n^{k'}$, the FAS correction $\boldsymbol{\tau}_n^k$, and the restriction of the new initial value $\tilde{U}_{n,0}^k$. This will yield new values $\tilde{\boldsymbol{U}}_n^k$ and $\tilde{\boldsymbol{F}}_n^k$.

(6) Interpolate the coarse correction $\tilde{U}_{n,\tilde{M}}^{k'} - \tilde{U}_{n,\tilde{M}}^{k}$ (in space only) and add to $U_{n,M}^{k'}$ to yield the updated value $U_{n,M}^k$.

(7) Send $U_{n,M}^k$ to processor $\boldsymbol{P}_{n+1}$ (if $n < N-1$). This will be received as the new initial condition $U_{n+1,0}^{k+1}$ in the next iteration.

(8) Interpolate the coarse grid correction $\tilde{\boldsymbol{U}}_n^{k'} - \tilde{\boldsymbol{U}}_n^k$ in space and time at the remaining fine time nodes ($0 < m < M$) and add to $\boldsymbol{U}_n^{k'}$ to yield $\boldsymbol{U}_n^k$. Recompute new values $\boldsymbol{F}_n^k$

The majority of the overhead associated with FAS is done in step 8 above, which is delayed until after the new initial condition is sent in step 7. This minimizes the amount of computation done between receiving a new initial condition from the previous processor (step 4) and sending the data forward (step 7). Pseudocode for the PFASST algorithm can be found in the Appendix.

4. Parallel speedup and efficiency

In this section, the theoretical parallel efficiency and speedup of the PFASST algorithm are examined. In the implementation used for the numerical results presented here, the PFASST method will converge to the collocation formulation (22) using one time step per processor. The number of substeps used in the fine SDC method is denoted again by M (which is the number of fine SDC nodes used minus one). Let η_F denote the cost of the method used for each substep of the fine SDC sweep. Likewise, let η_G and $\tilde{M}$ be the corresponding constants for the coarse SDC sweep. Further, define $\Upsilon_F = M\eta_F$ and $\Upsilon_G = \tilde{M}\eta_G$ to be the cost of one fine/coarse SDC sweep (assuming that the cost of computing the integration term used in the sweep is negligible). Let n_G denote the number of coarse SDC sweeps performed per PFASST iteration. To take into consideration the overhead of parallelization, we define γ_F and γ_G as the cost of sending a coarse and fine solution from one processor to the next. Finally, we define $\Upsilon_{\mathbb{O}}$ as the cost of the interpolation and restriction done in FAS.

If the PFASST iterations converge to the required accuracy in K_p iterations, the total cost on N processors is

$$C_p = Nn_G\Upsilon_G + (N-1)\gamma_G + K_p(\Upsilon_F + n_G\Upsilon_G + \Upsilon_{\mathbb{O}} + \gamma_F). \tag{24}$$

For simplicity, the communication costs γ_G and γ_F (which are relatively small in the numerical experiments in Section 5) are treated as overhead and are included in the $\Upsilon_{\mathbb{O}}$ term hereafter.

Let K_s denote the number of SDC iterations needed to compute the solution to the desired accuracy in serial using the fine SDC nodes. Then the cost of the serial SDC method will be approximately $C_s = NK_s\Upsilon_F$. Therefore, the parallel speedup S of the PFASST algorithm is

$$S = \frac{C_s}{C_p} = \frac{NK_s\Upsilon_F}{Nn_G\Upsilon_G + K_p(\Upsilon_F + n_G\Upsilon_G + \Upsilon_{\mathbb{O}})}. \tag{25}$$

Defining $\alpha = \Upsilon_G/\Upsilon_F$ and $\beta = \Upsilon_{\mathbb{O}}/\Upsilon_F$, (25) becomes

$$S = \frac{N}{\dfrac{Nn_G\alpha}{K_s} + \dfrac{K_p}{K_s}(1 + n_G\alpha + \beta)} \tag{26}$$

which gives a parallel efficiency of

$$E = \frac{1}{\dfrac{Nn_G\alpha}{K_s} + \dfrac{K_p}{K_s}(1 + n_G\alpha + \beta)}. \tag{27}$$

To achieve a parallel efficiency that is close to 1, the two quantities $Nn_G\alpha/K_s$ and K_p/K_s should be as small as possible. If the coarsening ratio between the

coarse and fine grids is two in both time and space, and the implicit solves in the method have a cost proportional to the total number of grid points in the problem, then the cost ratio α between the coarse and fine SDC sweeps is approximately $1/2^{D+1}$, where D is the spatial dimension of the problem. This means that the $Nn_G\alpha/K_s$ term is well approximated by $Nn_G/(2^{D+1}K_s)$. Furthermore, since in each PFASST iteration both coarse and fine SDC sweeps are done, K_p/K_s can actually be less than one (as shown in Section 5). The numerical experiments also show that increasing the number of coarse SDC sweeps, n_G, reduces K_p, but at the cost of increasing all terms containing α. Finally, it should be noted that the cost of the overhead from the FAS procedure (and communication) signified by $\Upsilon_{\mathbb{C}}$ is not necessarily small. Since in the numerical tests presented here, both the evaluation of the function values and the interpolation in FAS are done with the FFT, β is in fact close to 1.

In contrast to the standard parareal method, note that the efficiency is not automatically bounded above by $E < 1/K_p$ but rather $E < K_s/K_p$. That is, by combining the SDC and parareal iterations into one hybrid parareal/SDC iteration, the bound on the parallel efficiency is relaxed by a factor of K_s when compared to the standard parareal method.

5. Numerical examples

In this section numerical results are presented to demonstrate the performance of the PFASST method for several PDEs of varying complexity. Since we are most interested in temporal errors, a pseudospectral discretization in space is used for all examples to minimize spatial errors. Also periodic boundary conditions are prescribed so that the discrete fast Fourier transform (FFT) can be used to evaluate the spectral derivatives and to interpolate in space. Temporal interpolation is done using standard polynomial interpolation from coarse nodes to fine. In both space and time, coarse grids are formed by taking every other fine point, so that the restriction operator is simply point-wise injection.

In the numerical tests that follow, the convergence of PFASST iterates is at times compared with the convergence of a serial SDC method. For the serial SDC method, the number of iterations reported refers to how many SDC sweeps are performed during each time step. This relates to the order of the method since each SDC sweep raises the formal order of accuracy by one, up to the accuracy of the underlying quadrature rule. In all cases, the errors reported are computed by comparing to a temporally resolved run on the fine grid (i.e., the solution of the discretized ODE and not the solution to the underlying PDE).

In all the examples below, the fine nodes in time correspond to either 9 or 5 Gauss–Lobatto nodes. Hence there are 5 or 3 coarse nodes respectively. When

9 nodes are used on the fine level, the 5 coarse nodes do not correspond to the Lobatto nodes, and hence the underlying quadrature rule is of order 6 instead of order 8. For 5 fine nodes, the 3 coarse nodes are the Lobatto nodes (Simpson's rule). When errors from serial SDC runs on coarse nodes are reported, the nodes used correspond to the coarse PFASST level and not the coarse Lobatto rule. One could instead interpolate the solutions in time to coarse Lobatto nodes or use Clenshaw–Curtis quadrature nodes at each level so that coarse and fine nodes correspond. Numerical experiments (not reported here) suggest the convergence of the PFASST iterations is not improved when using Clenshaw–Curtis nodes, and the accuracy of the fine solution is reduced. A more careful examination of the impact of different interpolation and restriction strategies in space and time is in preparation.

5.1. ***Viscous Burger's equation.*** In the first set of examples, the convergence of the PFASST iterates are examined on a simple one-dimensional equation, namely the viscous Burger's (VB) equation

$$u_t + uu_x = \nu u_{xx}, \tag{28}$$

where $\nu = 0.005$ is the diffusion constant. The VB equation (28) is split into explicit and implicit parts according to $f_E = -uu_x$ and $f_I = \nu u_{xx}$ in (6). That is, the nonlinear advection term is treated explicitly while the linear diffusion term is treated implicitly. The domain is the unit interval, and the initial conditions are

$$u_0(x) = e^{-(x-0.5)^2/\sigma}, \tag{29}$$

with $\sigma = 0.004$, so that the solution has a full spatial spectrum. The periodic images of the Gaussian are included in the initial condition to ensure it is spatially smooth. The spatial discretization is chosen so that the solution is resolved on the fine and coarse resolutions: 512 points are used on the fine grid and 256 on the coarse. The temporal discretizations are done with 5 Gauss–Lobatto SDC nodes on the fine level, and 3 Gauss–Lobatto SDC nodes on the coarse level. Analysis of the performance of PFASST when the coarse level is not well-resolved is ongoing and will be presented elsewhere.

For the VB test, 64 processors are used with the time step on each processor being $\Delta t = 0.08/64$. Note that the real part of the quantity $-\Delta t \nu (2\pi k_{\max})^2$, where $k_{\max}$ is the largest Fourier wave number, is approximately -15.16 on the fine grid. Hence the use of a semi-implicit method (as opposed to a fully explicit method) avoids a substantial time step restriction.

Figure 1 shows the convergence of the PFASST algorithm and the serial SDC method for the VB test. The error is computed for each SDC and PFASST iteration at the end of each time step, and therefore the horizontal axis corresponds to the time step, which in turn corresponds to the processor number in the PFASST case.

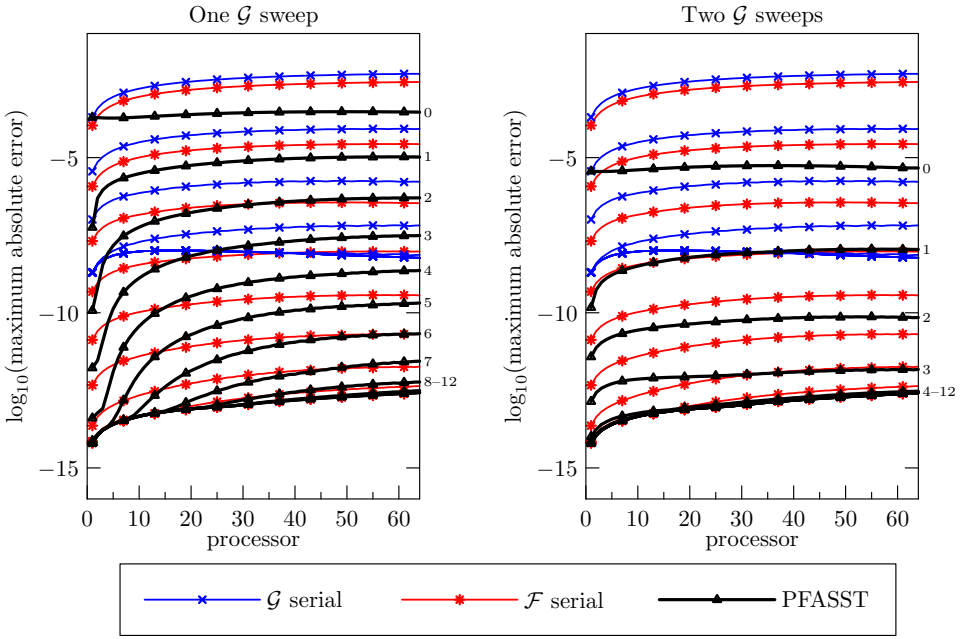


Figure 1. Maximum absolute error versus time (processor) for several PFASST iterations applied to the VB equation. The left and right panels correspond to one and two coarse SDC sweeps per PFASST iteration respectively. Each PFASST line represents the error at the end of the corresponding PFASST iteration, with 0 representing the solution after initialization. The $\mathcal{G}$ and $\mathcal{F}$ serial lines represent the error of serial SDC runs on the coarse and fine space-time discretizations with varying numbers of SDC sweeps (iterations) per time step.

Again, keep in mind that the number of iterations reported for the serial SDC runs refers to how many SDC sweeps are performed during each time step of the serial method. The PFASST algorithm is run using $n_G = 1$ (left panel) and $n_G = 2$ (right panel) coarse SDC sweeps per iteration. Several observations can be made from the data.

Concerning the convergence of the serial SDC method, note first that the minimum error is reached after 4 SDC iterations for the coarse grid, which is in agreement with the formal fourth-order accuracy expected with 3 Gauss–Lobatto nodes. Similarly, the fine serial SDC method very nearly attains minimum error after 8 iterations, which again is consistent with the formal eighth-order accuracy. Note that the convergence of SDC to the collocation solution can be slower for very stiff problems [14], as is the case for the examples in Section 5.2. Because of the spectral accuracy of the SDC method, the fine solution with 5 Gauss–Lobatto nodes is substantially more accurate than the coarse.

Turning to the convergence of the PFASST algorithm, the first thing to note is that the PFASST iterates do converge to the converged SDC solution on the fine

grid. Next, note that using two coarse SDC sweeps per PFASST iteration in this example reduces the total number of iterations needed to obtain a highly accurate solution from 9 iterations to 4 iterations, albeit at a higher cost per iteration. This behavior is similar to the parareal algorithm in that the overall speed of convergence of the algorithm depends on the accuracy of the coarse propagator [24]. Although not shown here, it has been observed that using multiple fine SDC sweeps per iteration does not have a significant effect on the rate of convergence of the PFASST algorithm for this example.

Note the effect of the iterative initialization procedure described in Section 3.1. The error after initialization (labeled 0 in each panel) is significantly less than that produced using the corresponding number of serial coarse SDC sweeps (one in the left panel and two in the right) at later times. Of course for the first processor, the initialization is exactly equivalent to the first step of a serial SDC method.

Figure 2 compares the convergence of the PFASST algorithm and serial SDC runs by considering error versus iteration computed at the final time. Additional

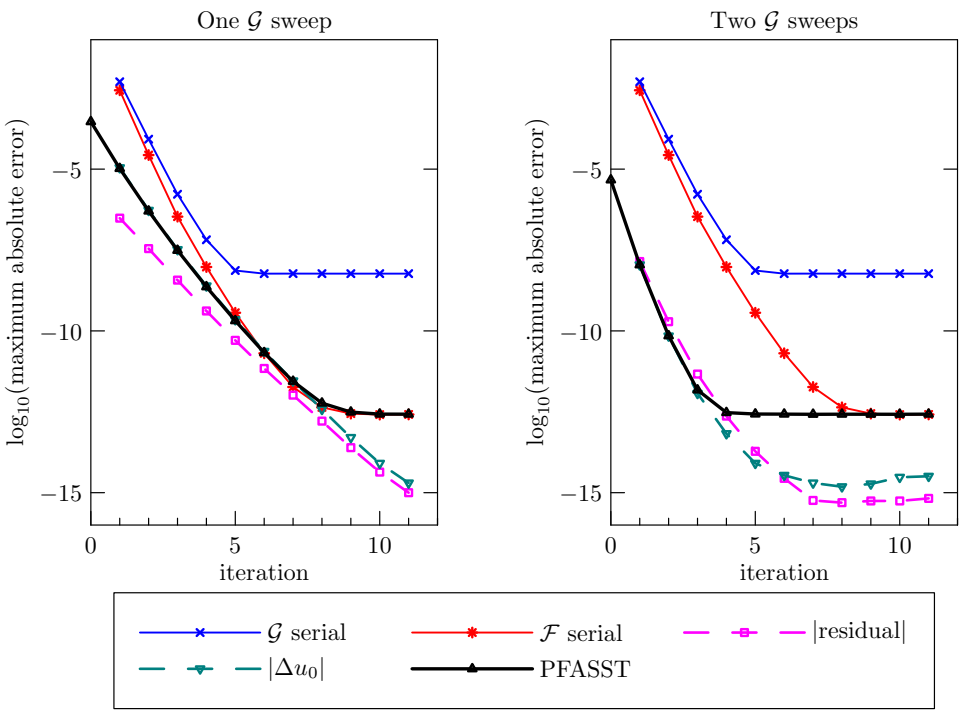


Figure 2. Maximum absolute error, residual, and change in fine initial condition computed at the final time interval for several PFASST iterations applied to the VB equation. The left and right panels correspond respectively to one and two coarse SDC sweeps per coarse PFASST iteration. The $\mathcal{G}$ and $\mathcal{F}$ serial lines represent the error of serial SDC runs on the coarse and fine space-time discretizations with varying numbers of SDC sweeps (iterations) per time step.

data corresponding to the residual and the maximum change in the initial condition at each processor is also included. It is again apparent that the PFASST algorithm achieves the accuracy of the fine serial run despite the relatively poor temporal accuracy of the coarse resolution. Also, the convergence of the PFASST algorithm is accelerated by using two coarse SDC sweeps per iteration. Since the problem is well-resolved at the finest level, the residual and change in initial condition are good indicators of the error at each iteration, until the accuracy of the time integration scheme is reached. Using the residual and change in initial condition to adaptively control the number of PFASST iterations performed will be explored elsewhere.

To demonstrate the importance of including the FAS correction term in the coarse sweeps, the above test was rerun omitting the FAS correction term defined in (23). Figure 3 shows the convergence of the algorithm without FAS corrections. It is clear that when FAS corrections are not included, the algorithm can only achieve accuracy comparable to the coarse level.

Finally, Figure 4 shows how the performance of the PFASST algorithm depends on the numbers of processors for the VB example. The left panel shows the convergence results for four different numbers of processors with the final time fixed (so

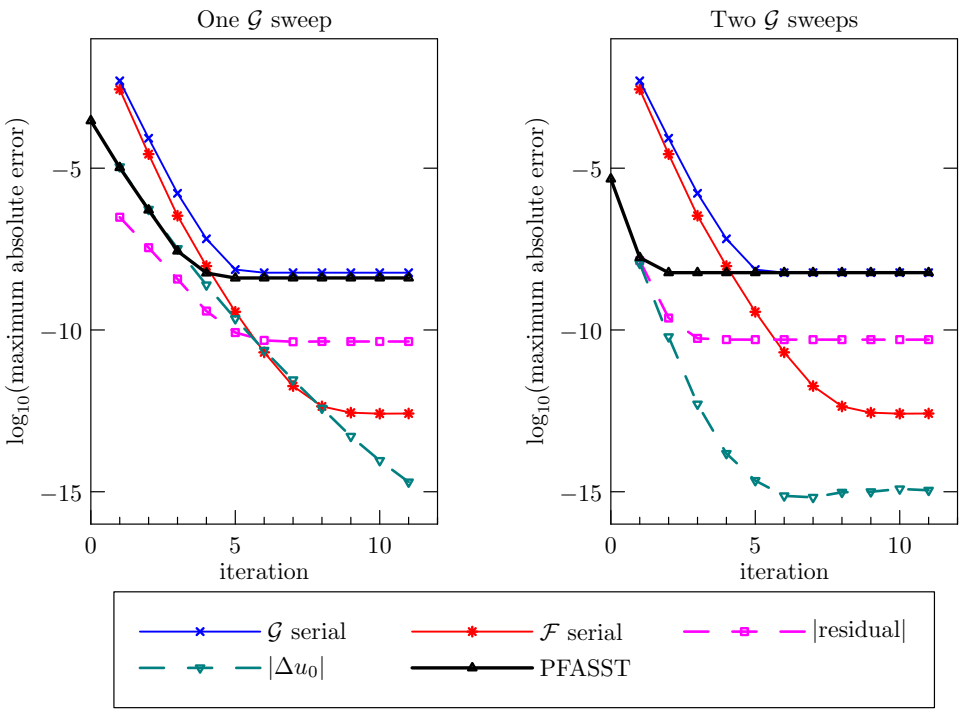


Figure 3. Maximum absolute error, residual, and change in fine initial condition computed at the final time for several PFASST iterations for the VB equation without FAS corrections. This figure should be compared to Figure 2.

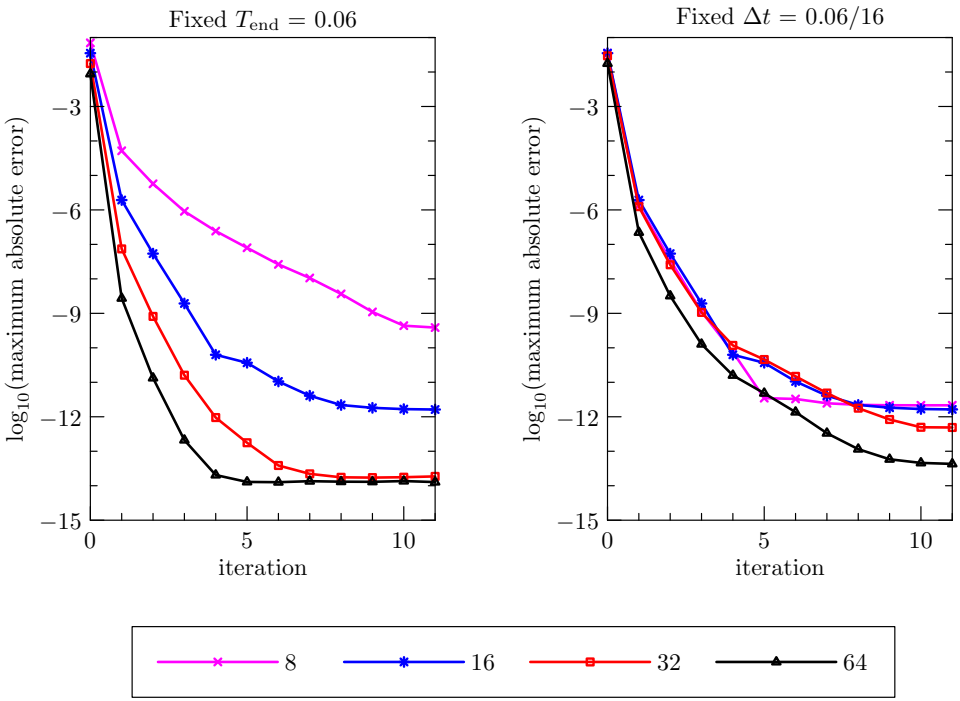


Figure 4. Maximum absolute error computed at the final time for several PFASST iterations and number of processors for the VB equation. The left panel shows PFASST errors at the final time for a fixed final time. The right panel shows PFASST errors at the final time for a fixed time step.

that the time step decreases as the number of processors increases). The right panel shows the convergence for the same numbers of processors with a fixed time step (so that the final integration time increases as the number of processors increases).

When the final time is fixed (left panel) the convergence of the PFASST algorithm improves as N increases in that the number of PFASST iterations required to achieve a given level of accuracy decreases. When the time step is fixed (right panel) the convergence is similar for each run despite the difference in simulation time. In summary, for this example the convergence depends largely on the time step used, not on the number of processors.

5.2. ***The Kuramoto–Silvashinsky equation.*** Next, the performance of the PFASST algorithm is explored for the Kuramoto–Silvashinsky equation

$$u_t + \tfrac{1}{2}|\nabla u|^2 + \nabla^2 u + \nabla^4 u = 0, \tag{30}$$

where the solution u is a function of two space variables and time. The KS equation arises as a model for interfacial instabilities in a variety of physical contexts and

has been shown to exhibit nontrivial dynamical behavior, both spatially and temporally, including chaos [16]. It contains a nonlinear term and high-order derivatives which, from a numerical perspective, make it a challenging equation to solve as it is nonlinear, very stiff, and highly sensitive to changes in the initial conditions or numerical error.

The KS equation (30) is split into explicit and implicit parts according to $f_E = -\frac{1}{2}|\nabla u|^2$ and $f_I = -\nabla^2 u - \nabla^4 u$ in (6). That is, the nonlinear term is treated explicitly while the linear antidiffusion and hyper-diffusion terms are treated implicitly. As with the VB equation, periodic boundary conditions are used, all spatial operators are evaluated spectrally, and the computational grid is chosen so that the solution is fairly well resolved on the fine grid. The domain size used throughout is a two-dimensional square domain with sides of length $L = 100.0$, with 512 points in each dimension on the fine grid and 256 on the coarse. The initial condition used throughout is shown in Figure 5. This initial condition was obtained by running the KS equation from a simple initial condition with only three Fourier modes to a final time of approximately 97, and subsequently removing high frequency modes (magnitude of wave-number greater than 121). This produces an initial condition with a broad spectrum of Fourier modes but without any fast initial transients. The temporal discretization uses 9 Gauss–Lobatto SDC nodes on the fine level, and 5 nodes on the coarse level. Note that the coarse nodes do not correspond to the 5-point Gauss–Lobatto rule.

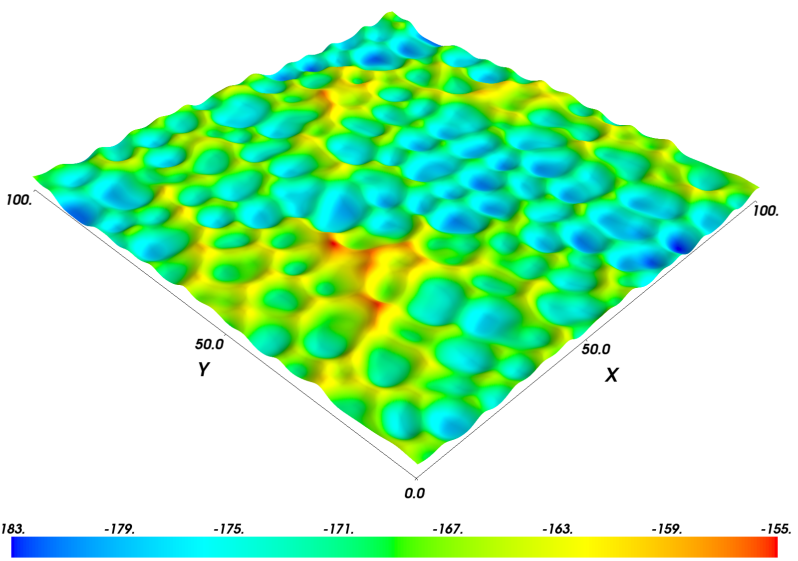


Figure 5. Initial condition for the KS and AD equation examples.

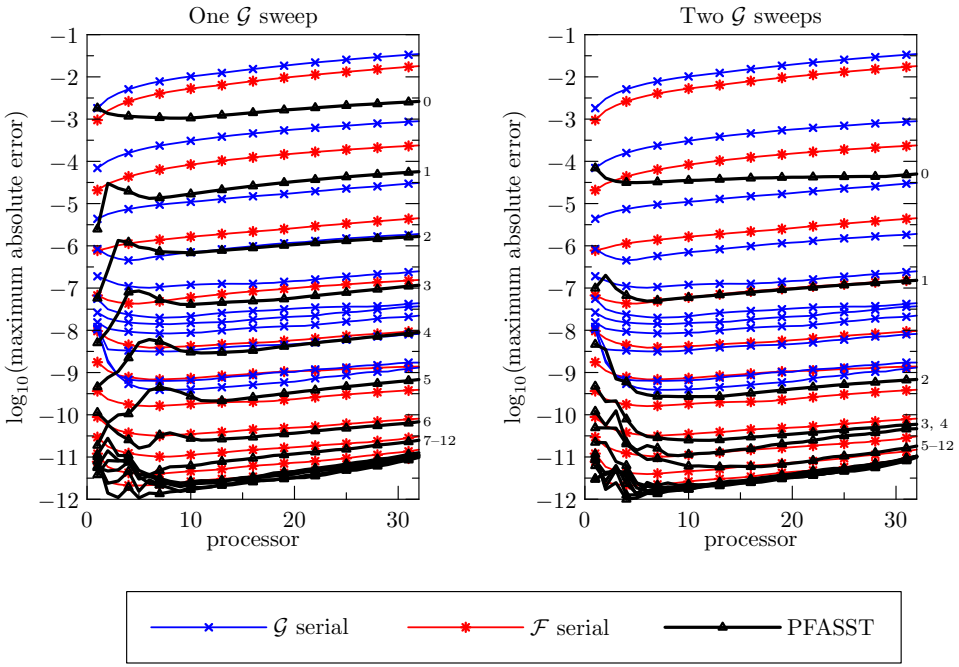


Figure 6. Maximum absolute error versus time (processor) for several PFASST iterations applied to the KS equation. The left and right panels show PFASST errors for one and two coarse SDC sweeps per PFASST iteration respectively. Each PFASST line represents the error at the end of the corresponding PFASST iteration, with 0 representing the solution after initialization. The $\mathcal{G}$ and $\mathcal{F}$ serial lines represent the error of serial SDC runs on the coarse and fine space-time discretizations with varying numbers of SDC sweeps (iterations) per time step.

For the KS test, 32 processors are used with the time step on each being $\Delta t = 1.0/32$. Figures 6 and 7 compare the convergence of the PFASST algorithm and serial SDC runs by considering the error versus processor (or simulation time) for each PFASST iteration, and the error versus iteration at the final time, respectively.

The results for KS shown in Figures 6 and 7 are qualitatively the same as those for VB shown in Figures 1 and 2. Again one can note the benefit of the iterative initialization procedure for the PFASST algorithm in that the error after initialization is considerably lower than a serial SDC method with the corresponding number of sweeps (except for at the first processor where they are identical). Note also that the PFASST method converges faster when using $n_G = 2$ coarse SDC sweeps per iteration instead of one, and this can improve the parallel efficiency (as discussed in Section 5.3).

For the KS equation, the minimum error achieved by both the serial and parallel methods is higher than in the VB case despite the use of 9 fine nodes for the KS equation instead of 5 for VB. To highlight the difficulty inherent in the parallel

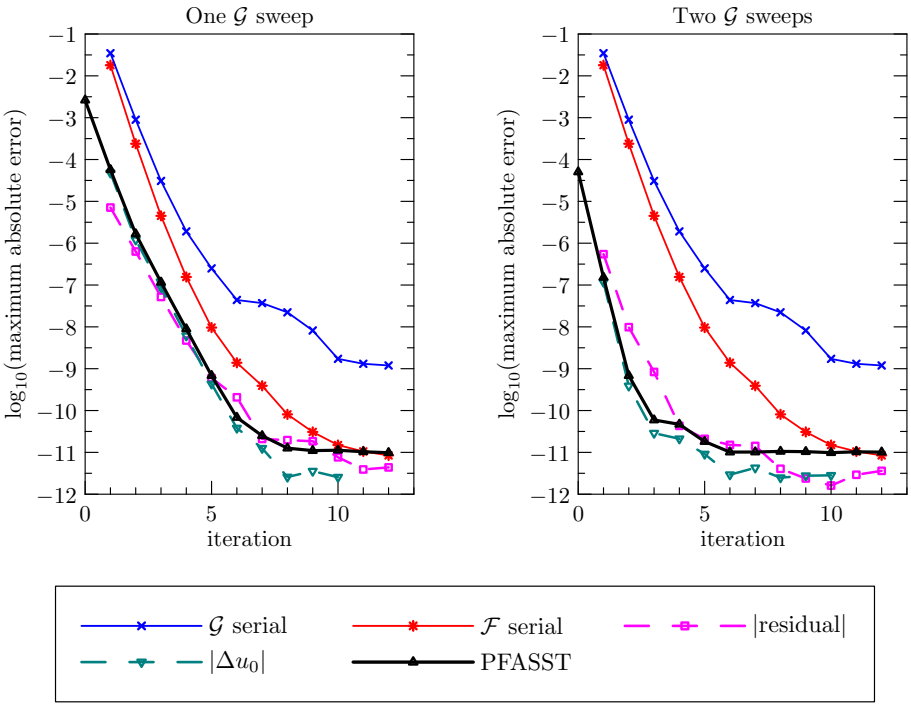


Figure 7. Maximum absolute error, residual, and change in fine initial condition computed at the final time for PFASST iterations applied to the KS equation. The left and right panels correspond to one and two coarse SDC sweeps per PFASST iteration respectively. The $\mathcal{G}$ and $\mathcal{F}$ serial lines represent the error of serial SDC runs on the coarse and fine space-time discretizations with varying numbers of SDC sweeps (iterations) per time step.

numerical approximations of the KS equation, a final numerical experiment is performed using the same initial conditions as the KS example (Figure 5), but with a simpler linear advection-diffusion equation

$$u_t + \nabla \cdot u = \nu \nabla^2 u \tag{31}$$

with $\nu = 0.02$. The domain size, spatial discretization, and temporal discretization are the same as in the KS example. Figure 8 shows the convergence of the PFASST algorithm for this linear advection/diffusion (AD) equation. It is clear that PFASST converges much faster than in the KS example, specifically in four and two iterations for one and two coarse SDC sweeps per iteration respectively.

5.3. ***Parallel timing results.*** The results in Section 5.2 show that the PFASST algorithm exhibits reasonable convergence behavior for a selection of PDEs in simple geometries. In this section, the parallel speedup and efficiency of the PFASST algorithm are explored. The PFASST algorithm has been implemented in F90 using MPI for communication between processors. The timing results correspond

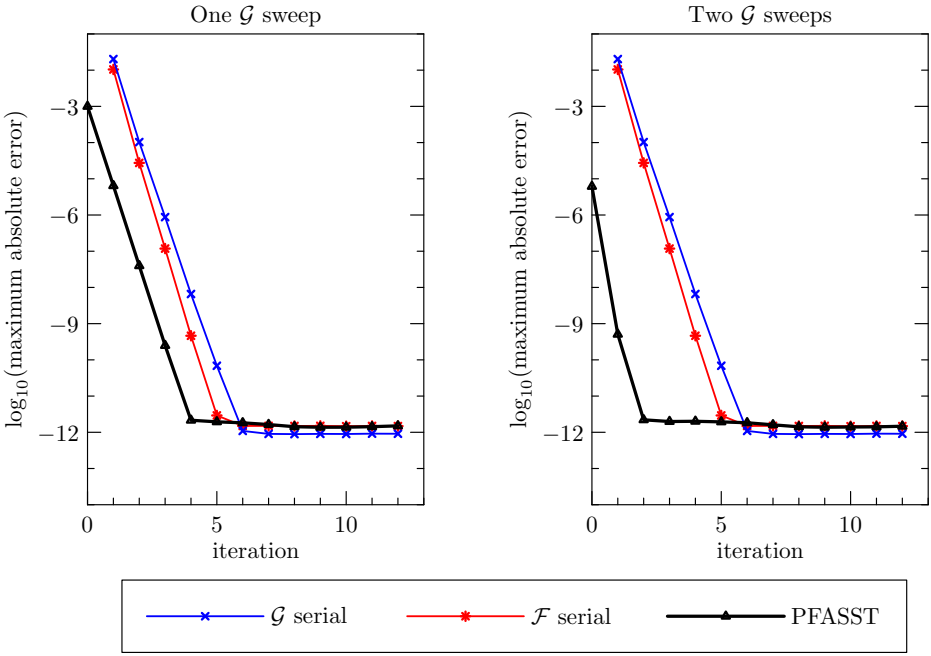


Figure 8. Maximum absolute error versus iteration at the final time for PFASST iterations applied to the AD equation. The left and right panels correspond to one and two coarse SDC sweeps per PFASST iteration respectively.

to numerical experiments performed using 8 and 16 processors on a multicore UNIX machine with 2×8 2GHz AMD Opteron cores, and run times reported were computed using the MPI `wtime` command. Preliminary timings have also been performed on a distributed memory cluster using up to 512 cores. Initial results suggest that communication costs across interconnects do not significantly impact the efficiency of the PFASST method. A more thorough analysis of these costs will be presented elsewhere.

For the following tests, the PFASST method is applied to a scalar VB equation (28) in three dimensions

$$u_t + u\nabla \cdot u = \nu \nabla^2 u, \tag{32}$$

hereafter referred to as the 3d NAD equation. The spatial resolution is 128 points in each dimension on the fine grid and 64 on the coarse. The initial condition used was a periodic image of

$$u_0(x) = (4\pi\nu)^{-3/2} e^{-(x-0.5)^2/(4\pi\nu)}. \tag{33}$$

The parameter values used were $\nu = 0.02$ and $T_{\text{end}} = 0.002$. The temporal discretizations are done with 5 Gauss–Lobatto SDC nodes on the fine level, and 3 Gauss–Lobatto SDC nodes on the coarse level.

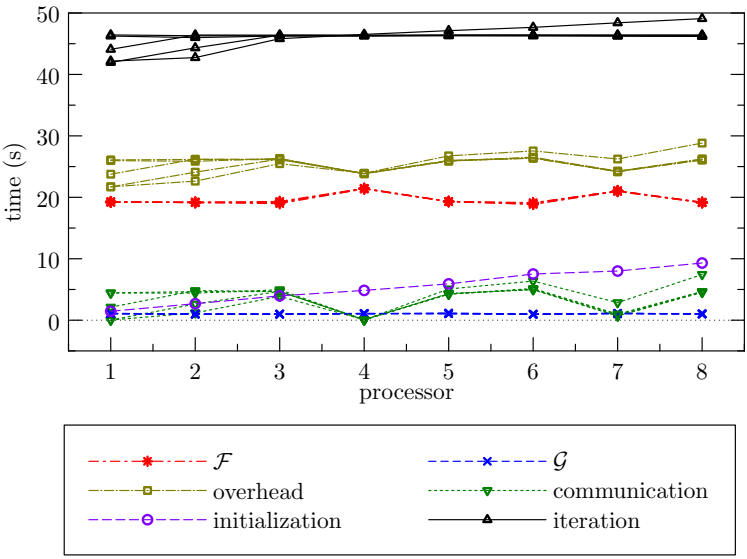


Figure 9. PFASST iteration timings for the third NAD equation on 8 processors. For all series except the predictor, each line corresponds to a different PFASST iteration from 1 to 8.

Figure 9 shows a timing breakdown of PFASST iterations for the 3d NAD equation using 8 processors. As expected, the predictor time grows linearly with the processor number. This "burn-in" time is unavoidable, but the slope could be reduced by introducing even coarser levels in the iterative initialization procedure. The ratio of the cost of fine to coarse SDC sweeps (denoted by α in Section 4) is approximately 16, which is consistent with a factor of two coarsening in both time and space. The communication cost associated with passing the fine solution forward in time from $\boldsymbol{P}_n$ to $\boldsymbol{P}_{n+1}$ is greater than the cost of a coarse SDC sweep, but less than that of a fine SDC sweep. Finally, the overhead of interpolation and restriction performed to compute the FAS correction is slightly more costly than a fine SDC sweep (i.e., $\beta > 1$). This is because the FFT is used for both spatial interpolation and the "explicit" computation of the nonlinear advective terms.

Table 1 shows the parallel speedup and efficiency of the PFASST algorithm for the 3d NAD equation. The number of iterations used for each method was chosen so that the accuracy of the solution at the final time was consistent between the runs and approximately equal to 10^{-13}. The number of PFASST iterations required is less than the number of serial SDC iterations required since, during each PFASST iteration, at least two SDC sweeps are performed (one on the fine level and one or more on the coarse level). That is, K_p/K_s is less than one (see (26)). The parallel speedup and efficiency achieved are well predicted by the theoretical formulas (26) and (27).

method	processors	iterations	time	speedup	efficiency
Serial SDC	$\Delta t = T_{\text{end}}/8$	7 SDC	1115.34s		
PFASST, one $\mathcal{G}$ sweep	8	5 PFASST	310.47s	3.59	0.45
PFASST, two $\mathcal{G}$ sweeps	8	3 PFASST	212.48s	5.25	0.66
Serial SDC	$\Delta t = T_{\text{end}}/16$	5 SDC	1606.69s		
PFASST, one $\mathcal{G}$ sweep	16	4 PFASST	216.20s	7.43	0.46
PFASST, two $\mathcal{G}$ sweeps	16	3 PFASST	202.13s	7.95	0.50

Table 1. Parallel speedup and efficiency of the PFASST algorithm for the 3d NAD equation.

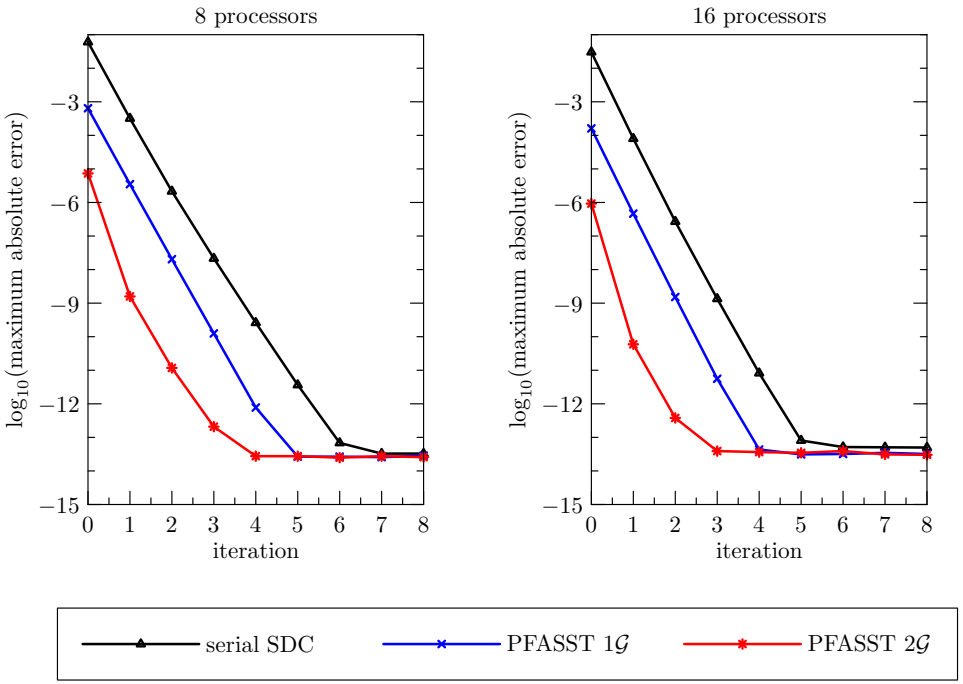


Figure 10. Maximum absolute error for the serial and PFASST iterations computed at the final time for the 3d NAD equation used for parallel timings.

Figure 10 compares the convergence of the PFASST runs and serial SDC runs by considering error versus iteration for the 3d NAD equation computed at the final time. All runs eventually achieve an accuracy of roughly 10^{-13}. This figure justifies the number of iterations used to perform the timings in Table 1.

6. Discussion

The preliminary results included in the previous section suggest that it is possible to achieve reasonable parallel efficiency in the temporal direction. Hence for PDE computations for which spatial parallelization has been saturated, parallelizing in

time appears attractive. Efforts to implement the PFASST algorithm within an existing spatial parallelization infrastructure are underway and will be reported on in the future. For problems in which many more time steps are desired than processor groups available, an algorithm for terminating the iterations at one time step so that the processors can begin computation on a new time step will be required. There are also several other immediate research directions we are pursuing that may increase the efficiency of the PFASST approach.

The test problems examined in this paper use spectrally accurate derivative and interpolation operators so that the convergence of the temporal scheme is easy to identify. One drawback of this approach is that the relative cost of interpolation and recomputing explicit function values in the FAS procedure is high. Of obvious interest is the performance of the PFASST method on problems discretized with finite-differences, finite-volumes, or finite-elements. Also, the performance of the PFASST approach on hyperbolic problems or those dominated by dispersive waves has not yet been fully investigated.

For the pseudospectral discretization used throughout the test cases here, the implicit equation at each substep is solved directly using the FFT. In practice, such equations for PDEs are usually solved using an iterative method such as a Krylov subspace or multigrid method. When combined with the PFASST algorithm, another avenue for a further gain in efficiency for PDEs employing iterative solvers is to reduce or vary the number of iterations of the implicit solver in different parts of the algorithm. For example, it may not be necessary to solve implicit equations within SDC substeps to full precision at every iteration, although it is difficult to predict *a priori* how a reduction in spatial solver accuracy will affect the convergence of the time parallel iterations.

Finally, the two-level method used in the paper can be easily extended to multiple levels as in standard multigrid methods. Such a method then resembles a space-time multigrid method where the "relaxation" operator in the temporal direction is an SDC sweep. Analysis and numerical testing of such a multilevel approach is ongoing.

Appendix: Pseudocode for the PFASST algorithm

See Figure 11 for a scheduling diagram of the algorithm.

Initialization on processor P_n.

Spread initial condition and compute FAS correction
SET: $\boldsymbol{U}^{0,0} = u(0)$, and evaluate $\boldsymbol{F}^{0,0}$
RESTRICT: fine $\boldsymbol{U}^{0,0}$ to coarse $\tilde{\boldsymbol{U}}^{0,0}$, and evaluate $\tilde{\boldsymbol{F}}^{0,0}$
COMPUTE: FAS correction $\boldsymbol{\tau}^0$ between $\boldsymbol{F}^{0,0}$ and $\tilde{\boldsymbol{F}}^{0,0}$
FOR $j = 0, \ldots, n-1$:

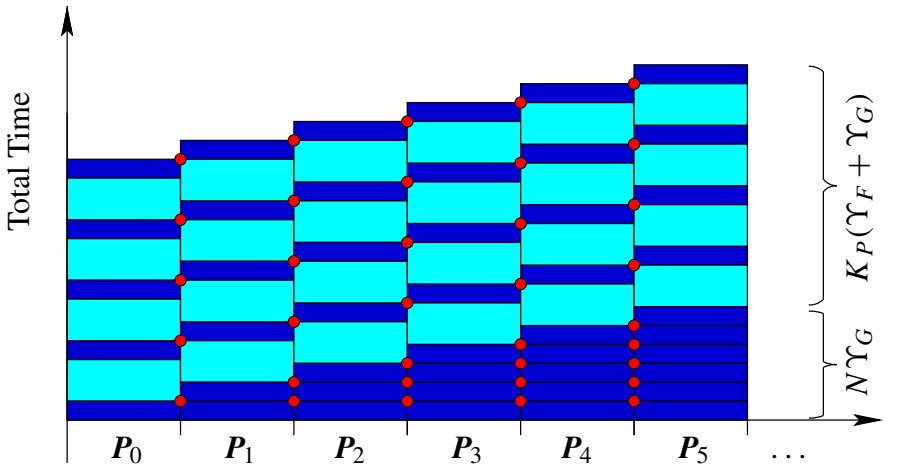


Figure 11. Cost diagram for the PFASST method with two space/time discretizations and the new iterative initialization procedure. Thinner darker rectangles correspond to coarse SDC sweeps, while finer correspond to fine SDC sweeps. Dots correspond to communication between processors.

Get new coarse initial value, sweep, and send forward
IF $j > 0$ and $n > 0$:
  RECEIVE: $\tilde{U}_0^{0,j} = \tilde{U}_{\tilde{M}}^{0,j}$ from $\boldsymbol{P}_{n-1}$
SWEEP: perform one SDC sweep with $\tilde{\boldsymbol{U}}^{0,j}$ and $\tilde{\boldsymbol{F}}^{0,j}$
UPDATE: during sweep, update $\tilde{\boldsymbol{U}}^{0,j+1}$ and evaluate $\tilde{\boldsymbol{F}}^{0,j+1}$ appropriately
IF $n < N-1$:
  SEND: $\tilde{U}_{\tilde{M}}^{0,j+1}$ to $\boldsymbol{P}_{n+1}$

PFASST iterations on processor $\boldsymbol{P_n}$.

SET: $\tilde{\boldsymbol{U}}^0$ to last coarse initialization $\tilde{\boldsymbol{U}}^{0,n-1}$, and evaluate $\tilde{\boldsymbol{F}}^0$
INTERPOLATE: coarse correction $\tilde{\boldsymbol{U}}^0 - \tilde{\boldsymbol{U}}^{0,0}$ to fine $\boldsymbol{U}^0$, and evaluate $\boldsymbol{F}^0$
FOR $k = 1, \ldots, K$
  Sweep on fine level, restrict, and compute FAS correction
  SWEEP: perform one SDC sweep with $\boldsymbol{U}^{k-1}$ and $\boldsymbol{F}^{k-1}$
  UPDATE: during sweep, update $\boldsymbol{U}^k$ and evaluate $\boldsymbol{F}^k$ appropriately
  RESTRICT: fine $\boldsymbol{U}^k$ to coarse $\tilde{\boldsymbol{U}}^k$, and evaluate $\tilde{\boldsymbol{F}}^k$
  COMPUTE: FAS correction $\boldsymbol{\tau}^k$ between $\boldsymbol{F}^k$ and $\tilde{\boldsymbol{F}}^k$
  Receive new fine initial condition, restrict
  IF $n > 0$:
    RECEIVE: $U_0^k = U_M^k$ from $\boldsymbol{P}_{n-1}$
  RESTRICT: fine U_0^k to coarse $\tilde{U}_0^k$
  Sweep on coarse level, interpolate final value, and send forward
  SWEEP: perform one SDC sweep with $\tilde{\boldsymbol{U}}^k$, $\tilde{\boldsymbol{F}}^k$, and $\boldsymbol{\tau}^k$

UPDATE: during sweep, update $\tilde{\boldsymbol{U}}^k$ and evaluate $\tilde{\boldsymbol{F}}^k$ appropriately
INTERPOLATE: coarse correction $\tilde{U}_{\tilde{M}}^k - \tilde{U}_{\tilde{M}}^{k-1}$ to fine U_M^k
IF $n < N - 1$:
  SEND: U_M^k to $\boldsymbol{P}_{n+1}$
Interpolate coarse to fine and set initial condition
INTERPOLATE: coarse correction $\tilde{\boldsymbol{U}}^k - \tilde{\boldsymbol{U}}^{k-1}$ to fine $\boldsymbol{U}^k$, and evaluate $\boldsymbol{F}^k$

References

[1] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri, *Implicit-explicit Runge–Kutta methods for time-dependent partial differential equations*, Appl. Numer. Math. **25** (1997), no. 2-3, 151–167. MR 98i:65054 Zbl 0896.65061

[2] G. Bal, *On the convergence and the stability of the parareal algorithm to solve partial differential equations*, Domain decomposition methods in science and engineering (R. Kornhuber et al., eds.), Lect. Notes Comput. Sci. Eng., no. 40, Springer, Berlin, 2005, pp. 425–432. MR 2235769 Zbl 1066.65091

[3] G. Bal and Y. Maday, *A "parareal" time discretization for non-linear PDE's with application to the pricing of an American put*, Recent developments in domain decomposition methods (L. F. Pavarino and A. Toselli, eds.), Lect. Notes Comput. Sci. Eng., no. 23, Springer, Berlin, 2002, pp. 189–202. MR 1962689

[4] K. Böhmer, P. Hemker, and H. J. Stetter, *The defect correction approach*, Defect correction methods (K. Böhmer and H. J. Stetter, eds.), Comput. Suppl., no. 5, Springer, Vienna, 1984, pp. 1–32. MR 86i:65007 Zbl 0551.65034

[5] S. Boscarino, *Error analysis of IMEX Runge–Kutta methods derived from differential-algebraic systems*, SIAM J. Numer. Anal. **45** (2007), no. 4, 1600–1621. MR 2008g:65086 Zbl 1152.65088

[6] 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. MR 2004f:76084 Zbl 1061.76053

[7] E. L. Bouzarth and M. L. Minion, *A multirate time integrator for regularized Stokeslets*, J. Comput. Phys. **229** (2010), no. 11, 4208–4224. MR 2011b:65099 Zbl 05712958

[8] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*, 2nd ed., Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. MR 2001h:65002 Zbl 0958.65128

[9] J. W. Daniel, V. Pereyra, and L. L. Schumaker, *Iterated deferred corrections for initial value problems*, Acta Ci. Venezolana **19** (1968), 128–135. MR 40 #8270

[10] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT **40** (2000), no. 2, 241–266. MR 2001e:65104 Zbl 0959.65084

[11] C. Farhat and M. Chandesris, *Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications*, Internat. J. Numer. Methods Engrg. **58** (2003), no. 9, 1397–1434. MR 2004h:65154 Zbl 1032.74701

[12] 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 (R. Kornhuber et al., eds.), Lect. Notes Comput. Sci. Eng., no. 40, Springer, Berlin, 2005, pp. 433–440. MR 2235770 Zbl 02143574

[13] M. J. Gander, *Analysis of the parareal algorithm applied to hyperbolic problems using characteristics*, Bol. Soc. Esp. Mat. Apl. SẽMA (2008), no. 42, 21–35. MR 2009b:65268

[14] J. Huang, J. Jia, and M. Minion, *Accelerating the convergence of spectral deferred correction methods*, J. Comput. Phys. **214** (2006), no. 2, 633–656. MR 2006k:65173 Zbl 1094.65066

[15] C. A. Kennedy and M. H. Carpenter, *Additive Runge–Kutta schemes for convection-diffusion-reaction equations*, Appl. Numer. Math. **44** (2003), no. 1-2, 139–181. MR 2003m:65111 Zbl 1013.65103

[16] I. G. Kevrekidis, B. Nicolaenko, and J. C. Scovel, *Back in the saddle again: a computer assisted study of the Kuramoto–Sivashinsky equation*, SIAM J. Appl. Math. **50** (1990), no. 3, 760–790. MR 91c:58095 Zbl 0722.35011

[17] A. T. Layton, *On the choice of correctors for semi-implicit Picard deferred correction methods*, Appl. Numer. Math. **58** (2008), no. 6, 845–858. MR 2009e:65116 Zbl 1143.65057

[18] 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. MR 2004k:76089 Zbl 1100.76048

[19] ———, *Implications of the choice of predictors for semi-implicit Picard integral deferred correction methods*, Commun. Appl. Math. Comput. Sci. **2** (2007), 1–34. MR 2008e:65252 Zbl 1131.65059

[20] J.-L. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps "pararéel"*, C. R. Acad. Sci. Paris Sér. I Math. **332** (2001), no. 7, 661–668. MR 2002c:65140

[21] M. L. Minion and S. A. Williams, *Parareal and spectral deferred corrections*, AIP Conference Proceedings, vol. 1048, 2008, pp. 388–391.

[22] M. L. Minion, *Higher-order semi-implicit projection methods*, Numerical simulations of incompressible flows (M. Hafez, ed.), World Sci. Publ., River Edge, NJ, 2003, pp. 126–140. MR 1984431 Zbl 1079.76056

[23] ———, *Semi-implicit projection methods for incompressible flow based on spectral deferred corrections*, Appl. Numer. Math. **48** (2004), no. 3-4, 369–387. MR 2056924 Zbl 1035.76040

[24] ———, *A hybrid parareal spectral deferred corrections method*, Commun. Appl. Math. Comput. Sci. **5** (2010), no. 2, 265–301. MR 2765386 Zbl 1208.65101

[25] L. Pareschi and G. Russo, *Implicit-explicit Runge–Kutta schemes for stiff systems of differential equations*, Recent trends in numerical analysis (D. Trigiante, ed.), Adv. Theory Comput. Math., no. 3, Nova Scientific, Huntington, NY, 2001, pp. 269–288. MR 2005a:65065 Zbl 1018.65093

[26] V. Pereyra, *On improving an approximate solution of a functional equation by deferred corrections*, Numer. Math. **8** (1966), 376–391. MR 34 #3814 Zbl 0173.18103

[27] ———, *Iterated deferred corrections for nonlinear operator equations*, Numer. Math. **10** (1967), 316–323. MR 36 #4812 Zbl 0258.65059

[28] J. W. Shen and X. Zhong, *Semi-implicit Runge–Kutta schemes for the non-autonomous differential equations in reactive flow computations*, Proceedings of the 27th AIAA Fluid Dynamics Conference, AIAA, June 1996, pp. 17–20.

[29] H. J. Stetter, *Economical global error estimation*, Stiff differential systems (R. A. Willoughby, ed.), Plenum, New York, 1974, pp. 245–258. MR 53 #9655

[30] P. Zadunaisky, *A method for the estimation of errors propagated in the numerical solution of a system of ordinary differential equations*, The Theory of Orbits in the Solar System and in Stellar Systems. Proceedings of International Astronomical Union, Symposium 25 (G. Contopoulos, ed.), 1964, pp. 281–287.

Received December 21, 2011. Revised January 18, 2012.

MATTHEW EMMETT: memmett@unc.edu
Department of Mathematics, University of North Carolina, CB 3250 Phillips Hall, Chapel Hill, NC 27599, United States

MICHAEL L. MINION: minion@unc.edu
Department of Mathematics, University of North Carolina, CB 3250 Phillips Hall, Chapel Hill, NC 27599, United States

Guidelines for Authors

Authors may submit manuscripts in PDF format on-line at the Submission page at msp.berkeley.edu/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. 7 no. 1 2012