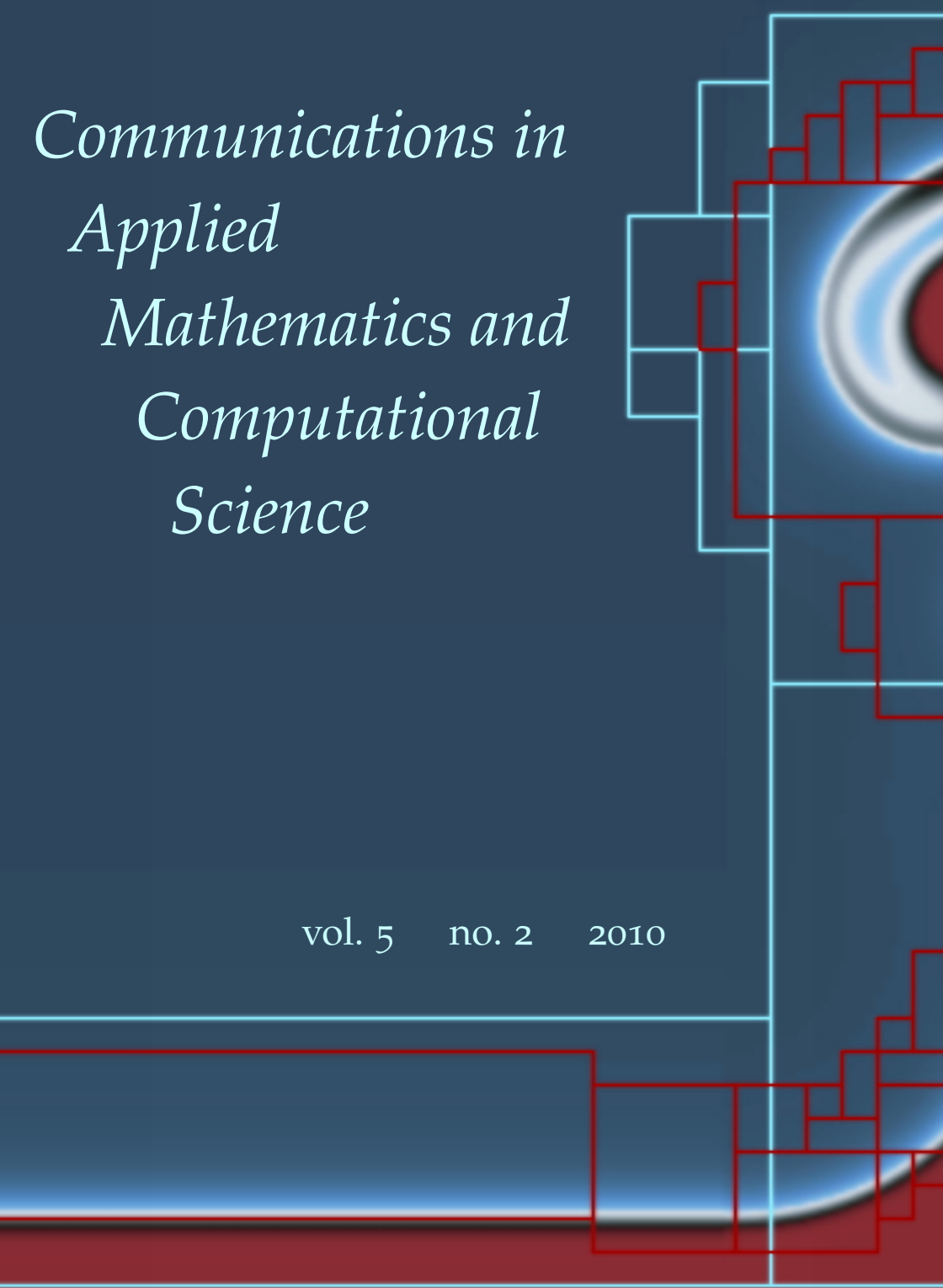
Communications in Applied Mathematics and Computational Science

vol. 5 no. 2 2010



mathematical sciences publishers

Communications in Applied Mathematics and Computational Science

pjm.math.berkeley.edu/camcos

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

The subscription price for 2010 is US $70/year for the electronic version, and $100/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://www.mathscipub.org
A NON-PROFIT CORPORATION
Typeset in LaTeX

COMM. APP. MATH. AND COMP. SCI.
Vol. 5, No. 2, 2010

ON THE ACCURACY OF FINITE-VOLUME SCHEMES FOR FLUCTUATING HYDRODYNAMICS

ALEKSANDAR DONEV, ERIC VANDEN-EIJNDEN,
ALEJANDRO GARCIA AND JOHN BELL

This paper describes the development and analysis of finite-volume methods for the Landau–Lifshitz Navier–Stokes (LLNS) equations and related stochastic partial differential equations in fluid dynamics. The LLNS equations incorporate thermal fluctuations into macroscopic hydrodynamics by the addition of white-noise fluxes whose magnitudes are set by a fluctuation-dissipation relation. Originally derived for equilibrium fluctuations, the LLNS equations have also been shown to be accurate for nonequilibrium systems. Previous studies of numerical methods for the LLNS equations focused primarily on measuring variances and correlations computed at equilibrium and for selected nonequilibrium flows. In this paper, we introduce a more systematic approach based on studying discrete equilibrium structure factors for a broad class of explicit linear finite-volume schemes. This new approach provides a better characterization of the accuracy of a spatiotemporal discretization as a function of wavenumber and frequency, allowing us to distinguish between behavior at long wavelengths, where accuracy is a prime concern, and short wavelengths, where stability concerns are of greater importance. We use this analysis to develop a specialized third-order Runge–Kutta scheme that minimizes the temporal integration error in the discrete structure factor at long wavelengths for the one-dimensional linearized LLNS equations. Together with a novel method for discretizing the stochastic stress tensor in dimension larger than one, our improved temporal integrator yields a scheme for the three-dimensional equations that satisfies a discrete fluctuation-dissipation balance for small time steps and is also sufficiently accurate even for time steps close to the stability limit.

MSC2000: 35K05, 65C30, 65N12, 65N40.
Keywords: finite-volume scheme, hydrodynamics.
Donev's work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. The work of Bell and Garcia was supported by the Applied Mathematics Research Program of the U. S. Department of Energy under contract no. DE-AC02-05CH11231. The work of Vanden-Eijnden was supported by the National Science Foundation through grants DMS02-09959, DMS02-39625, and DMS07-08140, and by the Office of Naval Research through grant N00014-04-1-0565.

1. Introduction

Recently the fluid dynamics community has considered increasingly complex physical, chemical, and biological phenomena at the microscopic scale, including systems for which significant interactions occur across multiple scales. At a molecular scale, fluids are not deterministic; the state of the fluid is constantly changing and stochastic, even at thermodynamic equilibrium. As simulations of fluids push toward the microscale, these random thermal fluctuations play an increasingly important role in describing the state of the fluid, especially when investigating systems where the microscopic fluctuations drive a macroscopic phenomenon such as the evolution of instabilities, or where the thermal fluctuations drive the motion of suspended microscopic objects in complex fluids. Some examples in which spontaneous fluctuations can significantly affect the dynamics include the breakup of droplets in jets [56; 27; 42], Brownian molecular motors [4; 58; 24; 54], Rayleigh–Bénard convection (both single species [65] and mixtures [60]), Kolmogorov flows [14; 15; 52], Rayleigh–Taylor mixing [41; 40], combustion and explosive detonation [57; 49], and reaction fronts [55].

Numerical schemes based on a particle representation of a fluid (e.g., molecular dynamics, direct simulation Monte Carlo [2]) inherently include spontaneous fluctuations due to the irregular dynamics of the particles. However, by far the most common numerical schemes in computational fluid dynamics are based on solving partial differential equations. To incorporate thermal fluctuations into macroscopic hydrodynamics, Landau and Lifshitz introduced an extended form of the compressible Navier–Stokes equations obtained by adding white-noise stochastic flux terms to the standard deterministic equations. While they were originally developed for equilibrium fluctuations, specifically the Rayleigh and Brillouin spectral lines in light scattering, the validity of the Landau–Lifshitz Navier–Stokes (LLNS) equations for nonequilibrium systems has been assessed [28] and verified in molecular simulations [33; 51; 53]. The LLNS system is one of the more complex examples in a broad family of PDEs with stochastic fluxes. Many members of this family arise from the LLNS equations in a variety of approximations (e.g., stochastic heat equation) while others are stochastic variants of well known PDEs, such as the stochastic Burger's equation [12], which can be derived from the continuum limit of an asymmetric excluded random walk.

Several numerical approaches for fluctuating hydrodynamics have been proposed. The earliest work by Garcia et al. [32] developed a simple scheme for the stochastic heat equation and the linearized one-dimensional LLNS equations. Ladd et al. [45] have included stress fluctuations in (isothermal) Lattice Boltzmann methods for some time, and recently a better theoretical foundation has been established [1; 26]. Moseler and Landman [56] included the stochastic stress tensor of the

LLNS equations in the lubrication equations and obtain good agreement with their molecular dynamics simulation in modeling the breakup of nanojets. Sharma and Patankar [61] developed a fluid-structure coupling between a fluctuating incompressible solver and suspended Brownian particles. Coveney, De Fabritiis, Delgado-Buscalioni and coworkers have also used the isothermal LLNS equations in a hybrid scheme, coupling a continuum fluctuating solver to a molecular dynamics simulation of a liquid [29; 35; 23]. Atzberger et al. [7] have developed a version of the immersed boundary method that includes fluctuations in a pseudospectral method for the incompressible Navier–Stokes equations. Voulgarakis and Chu [63] developed a staggered scheme for the isothermal LLNS equations as part of a multiscale method for biological applications, and a similar staggered scheme was also described in [22].

Recently, Bell et al. [13] introduced a centered scheme for the LLNS equations based on interpolation schemes designed to preserve fluctuations combined with a third-order Runge–Kutta (RK3) temporal integrator. In that work, the principal diagnostic used for evaluation of the numerical method was the accuracy of the local (cell) variance and spatial (cell-to-cell) correlation structure for equilibrium and selected nonequilibrium scenarios (e.g., constant temperature gradient). The metric established by those types of tests is, in some sense, simultaneously too crude and too demanding. It is too crude in the sense that it provides only limited information from detailed simulations that cannot be directly linked to specific properties of the scheme. On the other hand, such criteria are too demanding in the sense that they place requirements on the discretization integrated over all wavelengths, requiring that the method perform well at high wavenumbers where a deterministic PDE solver performs poorly. Furthermore, although Bell et al. [13] demonstrate that RK3 is an effective algorithm, compared with other explicit schemes for the compressible Navier–Stokes equations, the general development of schemes for the LLNS equations has been mostly trial and error.

Here, our goal is to establish a more rational basis for the analysis and development of explicit finite-volume scheme for stochastic partial differential equations (SPDEs) with a stochastic flux. The approach is based on analysis of the structure factor (equilibrium fluctuation spectrum) of the discrete system. The structure factor is, in essence, the stationary spatiotemporal correlations of hydrodynamic fluctuations as a function of spatial wavenumber and temporal frequency; the static structure factor is the integral over frequency (i.e., the spatial spectrum). By analyzing the structure factor for a numerical scheme, we are able to develop notions of accuracy for a given discretization at long wavelengths. Furthermore, in many cases the theoretical analysis for the structure factor is tractable (with the aid of symbolic manipulators) allowing us to determine optimal coefficients for a given numerical scheme. We perform this optimization as a two-step procedure. First, a

spatial discretization is developed that satisfies a discrete form of the fluctuation-dissipation balance condition. Then, a stable temporal integrator is proposed and the covariances of the random numbers are chosen so as to maximize the order of temporal accuracy of the small-wavenumber static structure factor. We focus primarily on explicit schemes for solving the LLNS equations because even at the scales where thermal fluctuations are important, the limitation on time step imposed by stability is primarily due to the hyperbolic terms. That is, when the cell size is comparable to the length scale for molecular transport (e.g., mean free path in a dilute gas) the time step for these compressible hydrodynamic equations is limited by the acoustic CFL (Courant–Friedrichs–Lewy) condition. At even smaller length scales the viscous terms further limit the time step yet the validity of a continuum representation for the fluid starts to break down at those atomic scales.

The paper is divided into roughly two parts: The first half (Sections 2–4) defines notation, develops the formalism, and derives the expressions for analyzing a general class of linear stochastic PDEs from the LLNS family of equations. The main result in the first half, how to evaluate the structure factor for a numerical scheme, appears in Section 3B. The second half applies this analysis to systems of increasing complexity, starting with the stochastic heat equation (Section 5A), followed by the LLNS system in one dimension (Section 6) and three dimensions (Section 7). The paper closes with a summary and concluding remarks, followed by an Appendix on the semi-implicit Crank–Nicolson method.

2. Landau–Lifshitz Navier–Stokes equations

We consider the accuracy of explicit finite-volume methods for solving the Landau–Lifshitz Navier–Stokes (LLNS) system of stochastic partial differential equations (SPDEs) in d dimensions, given in conservative form by

$$\partial_t \boldsymbol{U} = -\boldsymbol{\nabla}\cdot[\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{\mathcal{Z}}(\boldsymbol{U}, \boldsymbol{r}, t)], \tag{1}$$

where $\boldsymbol{U}(\boldsymbol{r}, t) = [\rho, \boldsymbol{j}, e]^T$ is a vector of *conserved variables* that are a function of the spatial position $\boldsymbol{r}$ and time t. The conserved variables are the densities of mass ρ, momentum $\boldsymbol{j} = \rho\boldsymbol{v}$, and energy $e = \varepsilon(\rho, T) + \frac{1}{2}\rho v^2$, expressed in terms of the *primitive variables*, mass density ρ, velocity $\boldsymbol{v}$, and temperature T; here ε is the internal energy density. The deterministic flux is taken from the traditional compressible Navier–Stokes–Fourier equations and can be split into *hyperbolic* and *diffusive fluxes*:

$$\boldsymbol{F}(\boldsymbol{U}) = \boldsymbol{F}_H(\boldsymbol{U}) + \boldsymbol{F}_D(\boldsymbol{U}),$$

where

$$\boldsymbol{F}_H = \begin{bmatrix} \rho\boldsymbol{v} \\ \rho\boldsymbol{v}\boldsymbol{v}^T + P\boldsymbol{I} \\ (e+P)\boldsymbol{v} \end{bmatrix} \quad \text{and} \quad \boldsymbol{F}_D = -\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\sigma} \\ \boldsymbol{\sigma}\cdot\boldsymbol{v} + \boldsymbol{\xi} \end{bmatrix},$$

$P = P(\rho, T)$ is the pressure, the viscous stress tensor is

$$\boldsymbol{\sigma} = \eta \overline{\nabla} \boldsymbol{v} = \eta \left[(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T) - \frac{2\,(\nabla \cdot \boldsymbol{v})}{d} \boldsymbol{I} \right]$$

for $d \geq 2$ (we have assumed zero bulk viscosity) and $\boldsymbol{\sigma} = \eta v_x$ for $d = 1$, and the heat flux is $\boldsymbol{\xi} = \mu \nabla T$. We denote the adjoint (conjugate transpose) of a matrix or linear operator $\boldsymbol{M}$ with $\boldsymbol{M}^\star = \overline{\boldsymbol{M}}^T$. As postulated by Landau and Lifshitz [46; 28], the *stochastic flux*

$$\mathcal{Z} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\Sigma} \\ \boldsymbol{\Sigma} \cdot \boldsymbol{v} + \boldsymbol{\Xi} \end{bmatrix}$$

is composed of the stochastic stress tensor $\boldsymbol{\Sigma}$ and stochastic heat flux vector $\boldsymbol{\Xi}$, assumed to be mutually uncorrelated random Gaussian fields with the following covariance (where bars denote means):

$$\begin{aligned} \left\langle \boldsymbol{\Sigma}(\boldsymbol{r}, t) \boldsymbol{\Sigma}^\star(\boldsymbol{r}', t') \right\rangle &= \boldsymbol{C}_{\boldsymbol{\Sigma}} \delta(t - t') \delta(\boldsymbol{r} - \boldsymbol{r}'), \\ &\qquad \text{where } C^{(\boldsymbol{\Sigma})}_{ij,kl} = 2\bar{\eta} k_B \overline{T} \left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk} - \tfrac{2}{d_f}\delta_{ij}\delta_{kl} \right); \\ \left\langle \boldsymbol{\Xi}(\boldsymbol{r}, t) \boldsymbol{\Xi}^\star(\boldsymbol{r}', t') \right\rangle &= \boldsymbol{C}_{\boldsymbol{\Xi}} \delta(t - t') \delta(\boldsymbol{r} - \boldsymbol{r}'), \qquad \text{where } C^{(\boldsymbol{\Xi})}_{i,j} = 2\bar{\mu} k_B \overline{T}^2 \delta_{ij}. \end{aligned} \tag{2}$$

In the LLNS system, the *hyperbolic* or *advective* fluxes are responsible for transporting the conserved quantities at the speed of sound or fluid velocity, without dissipation. On the other hand, the *diffusive* or *dissipative* fluxes are the ones responsible for damping the thermal fluctuations generated by the *stochastic* or *fluctuating* fluxes. At equilibrium a steady state is reached in which a *fluctuation-dissipation balance* condition is satisfied.

In the original formulation, Landau and Lifshitz only considered adding stochastic fluxes to the linearized Navier–Stokes equations, which leads to a well-defined system of SPDEs whose equilibrium solutions are random Gaussian fields. Derivations of the equations of fluctuating hydrodynamics through careful asymptotic expansions of the underlying microscopic (particle) dynamics give equations for the Gaussian fluctuations around the solution to the usual deterministic Navier–Stokes equations [47], in the spirit of the Central Limit Theorem. Therefore, numerical solutions should, in principle, consist of two steps: first solving the nonlinear deterministic equations for the *mean* solution, and then solving the linearized equations for the *fluctuations* around the mean. If the fluctuations are small perturbations, it makes sense numerically to try to combine these two steps into one and simply consider nonlinear equations with added thermal fluctuations. There is also hope that this might capture effects not captured in the two-system approach, such as fluctuation-

driven transport in nonequilibrium systems [59], or the effect of fluctuations on the very long-time dynamics of the mean (e.g., shock drift [13]) and hydrodynamic instabilities [65; 56; 40].

The linearized equations of fluctuating hydrodynamics can be given a well defined interpretation with the use of generalized functions or distributions [19]. However, the nonlinear fluctuating hydrodynamic equations (1) must be treated with some care since they have not been derived from first principles [28] and are in fact mathematically ill defined due to the high irregularity of white-noise fluctuating stresses [34]. More specifically, because the solution of these equations is itself a distribution the interpretation of the nonlinear terms requires giving a precise meaning to products of distributions, which cannot be defined in general and requires introducing some sort of regularization. Although written formally as an SPDE, the LLNS equations are usually interpreted in a finite volume context, where the issues of regularity, at first sight, disappear. However, in finite volume form the level of fluctuations becomes increasingly large as the volume shrinks and the nonlinear terms diverge leading to an "ultraviolet catastrophe" of the kind familiar in other fields of physics [34; 16]. Furthermore, because the noise terms are Gaussian, it is possible for rare events to push the system to states that are not thermodynamically valid such as negative T or ρ. For that reason, we will focus on the linearized LLNS equations, which can be given a well-defined interpretation. Since the fluctuations are expected to be a small perturbation of the deterministic solution, the nonlinear equations should behave similarly to the linearized equations anyway, at least near equilibrium for sufficiently large cells.

To simplify the exposition we assume the fluid to be a monoatomic ideal gas; the generalization of the results for an arbitrary fluid is tedious but straightforward. For an ideal gas the equation of state may be written as

$$P = \rho\,(k_B T/m) = \rho c^2,$$

where c is the isothermal speed of sound. The internal energy density is $\varepsilon = \rho c_v T$, where c_v is the heat capacity at constant volume, which may be written as $c_v = d_f k_B/2m$ where d_f is the number of degrees of freedom of the molecules (for monoatomic gases there are $d_f = d$ translational degrees of freedom), and $c_p = (1 + 2/d_f)c_v$ is the heat capacity at constant pressure. For analytical calculations, it is convenient to convert the LLNS system from conserved variables to primitive variables, since the primitive variables are uncorrelated at equilibrium and the equations (1) simplify considerably:

$$\begin{aligned} D_t \rho &= -\rho \boldsymbol{\nabla}\cdot \boldsymbol{v}, \\ \rho\,(D_t \boldsymbol{v}) &= -\boldsymbol{\nabla} P + \boldsymbol{\nabla}\cdot(\boldsymbol{\sigma} + \boldsymbol{\Sigma}), \\ \rho c_p\,(D_t T) &= D_t P + \boldsymbol{\nabla}\cdot(\boldsymbol{\xi} + \boldsymbol{\Xi}) + (\boldsymbol{\sigma} + \boldsymbol{\Sigma}) : \boldsymbol{\nabla} \boldsymbol{v}, \end{aligned} \tag{3}$$

where $D_t\Box = \partial_t\Box + \boldsymbol{v}\cdot\boldsymbol{\nabla}(\Box)$ denotes the familiar advective derivative. Note that in the fully nonlinear numerical implementation, however, we continue to use the conserved variables to ensure that the physical conservation laws are strictly obeyed.

Linearizing (3) around a reference uniform equilibrium state $\rho = \rho_0 + \delta\rho$, $\boldsymbol{v} = \boldsymbol{v}_0 + \delta\boldsymbol{v}$, $T = T_0 + \delta T$, and dropping the deltas for notational simplicity,

$$\boldsymbol{U} = \begin{bmatrix} \delta\rho \\ \delta\boldsymbol{v} \\ \delta T \end{bmatrix} \to \begin{bmatrix} \rho \\ \boldsymbol{v} \\ T \end{bmatrix},$$

we obtain the linearized LLNS system for the equilibrium thermal fluctuations,

$$\partial_t\boldsymbol{U} = -\boldsymbol{\nabla}\cdot[\boldsymbol{F}\boldsymbol{U} - \boldsymbol{\mathcal{Z}}] = -\boldsymbol{\nabla}\cdot[\boldsymbol{F}_H\boldsymbol{U} + \boldsymbol{F}_D\boldsymbol{\nabla}\boldsymbol{U} - \boldsymbol{\mathcal{Z}}], \tag{4}$$

where

$$\boldsymbol{F}_H\boldsymbol{U} = \begin{bmatrix} \rho_0\boldsymbol{v} + \rho\boldsymbol{v}_0 \\ \left(c_0^2\rho_0^{-1}\rho + c_0^2T_0^{-1}T\right)\boldsymbol{I} + \boldsymbol{v}_0\boldsymbol{v}^T \\ c_0^2c_v^{-1}\boldsymbol{v} + T\boldsymbol{v}_0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{F}_D\boldsymbol{\nabla}\boldsymbol{U} = \begin{bmatrix} 0 \\ \rho_0^{-1}\eta_0\overline{\boldsymbol{\nabla}}\boldsymbol{v} \\ \rho_0^{-1}c_v^{-1}\mu_0\boldsymbol{\nabla}T \end{bmatrix},$$

and $\boldsymbol{\mathcal{Z}}(\boldsymbol{r},t)$ is a random Gaussian field with a covariance

$$\left\langle \boldsymbol{\mathcal{Z}}(\boldsymbol{r},t)\boldsymbol{\mathcal{Z}}^{\star}(\boldsymbol{r}',t')\right\rangle = \boldsymbol{C}_{\boldsymbol{\mathcal{Z}}}\delta(t-t')\delta(\boldsymbol{r}-\boldsymbol{r}'),$$

where the covariance matrix is block diagonal,

$$\boldsymbol{C}_{\boldsymbol{\mathcal{Z}}} = \begin{bmatrix} 0 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \rho_0^{-2}\boldsymbol{C}_{\boldsymbol{\Sigma}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \rho_0^{-2}c_v^{-2}\boldsymbol{C}_{\boldsymbol{\Xi}} \end{bmatrix},$$

and $\boldsymbol{C}_{\boldsymbol{\Sigma}}$ and $\boldsymbol{C}_{\boldsymbol{\Xi}}$ are given in (2). Equation (4) is a system of linear SPDEs with additive noise that can be analyzed within a general framework, as we develop next. We note that the stochastic "forcing" in (4) is essentially a divergence of white noise, modeling conservative *intrinsic* (thermal) fluctuations [47], rather than the more common *external* fluctuations modeled through white noise forcing [21; 39].

The next two sections develop the tools for analyzing finite volume schemes for linearized SPDEs, such as the LLNS system, specifically how to predict the equilibrium spectrum of the fluctuations (i.e., structure factor) from the spatial and temporal discretization used by the numerical algorithm. These analysis tools are demonstrated for simple examples in Section 5A and applied to the LLNS system in Sections 6 and 7.

3. Explicit methods for linear stochastic partial differential equations

In this section, we develop an approach for analyzing the behavior of explicit discretizations for a broad class of SPDEs, motivated by the linearized form of the LLNS equations. In particular, we consider a general linear SPDE for the stochastic field $\boldsymbol{\mathcal{U}}(\boldsymbol{r},t)\equiv\boldsymbol{\mathcal{U}}(t)$ of the form

$$d\boldsymbol{\mathcal{U}}(t)=\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{U}}(t)\,dt+\boldsymbol{\mathcal{K}}\,d\boldsymbol{\mathcal{B}}(t), \tag{5}$$

with periodic boundary conditions on the torus $\boldsymbol{r}\in\mathcal{V}=[0,H]^d$, where $\boldsymbol{\mathcal{L}}$ (the *generator*) and $\boldsymbol{\mathcal{K}}$ (the *filter*) are time-independent linear operators, and $\boldsymbol{\mathcal{B}}$ is a cylindrical Wiener process (Brownian sheet), and the initial condition at $t=0$ is $\boldsymbol{\mathcal{U}}_0$. As common in the physics literature, we will abuse notation and write

$$\partial_t\boldsymbol{\mathcal{U}}=\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{U}}+\boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{W}},$$

where $\boldsymbol{\mathcal{W}}=d\boldsymbol{\mathcal{B}}(t)/dt$ is spatiotemporal white noise, that is, a random Gaussian field with zero mean and covariance

$$\left\langle\boldsymbol{\mathcal{W}}(\boldsymbol{r},t)\boldsymbol{\mathcal{W}}^\star(\boldsymbol{r}',t')\right\rangle=\delta(t-t')\delta(\boldsymbol{r}-\boldsymbol{r}'). \tag{6}$$

The so-called mild solution [19] of (5) is a generalized process

$$\boldsymbol{\mathcal{U}}(t)=e^{t\boldsymbol{\mathcal{L}}}\boldsymbol{\mathcal{U}}_0+\int_0^t e^{(t-s)\boldsymbol{\mathcal{L}}}\boldsymbol{\mathcal{K}}\,d\boldsymbol{\mathcal{B}}(s), \tag{7}$$

where the integral denotes a stochastic convolution. If the operator $\boldsymbol{\mathcal{L}}$ is dissipative, that is, $\lim_{t\to\infty}e^{t\boldsymbol{\mathcal{L}}}\boldsymbol{\mathcal{U}}_0=\boldsymbol{0}$ for all $\boldsymbol{\mathcal{U}}_0$, then at long times t' the solution to (5) is a Gaussian process with mean zero and covariance

$$\boldsymbol{C}_{\boldsymbol{\mathcal{U}}}(t)=\left\langle\boldsymbol{\mathcal{U}}(t')\boldsymbol{\mathcal{U}}^\star(t'+t)\right\rangle=\int_{-\infty}^0 e^{-s\boldsymbol{\mathcal{L}}}\boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{K}}^\star e^{(t-s)\boldsymbol{\mathcal{L}}^\star}ds,\quad t\geq 0. \tag{8}$$

This means that (5) has a unique invariant measure (equilibrium or stationary distribution) that is Gaussian with mean zero and covariance given in (8).

In general, the field $\boldsymbol{\mathcal{U}}(\boldsymbol{r},t)$ is only a generalized function of the spatial coordinate $\boldsymbol{r}$ and cannot be evaluated pointwise. For the cases we will consider here, specifically, translationally invariant problems where $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{K}}$ are differential operators, this difficulty can be avoided by transforming (5) to Fourier space via the Fourier series transform

$$\boldsymbol{\mathcal{U}}(\boldsymbol{r},t)=\sum_{\boldsymbol{k}\in\widehat{\mathcal{V}}}e^{i\boldsymbol{k}\cdot\boldsymbol{r}}\widehat{\boldsymbol{\mathcal{U}}}(\boldsymbol{k},t), \tag{9}$$

$$\widehat{\boldsymbol{\mathcal{U}}}(\boldsymbol{k},t)=\frac{1}{V}\int_{\boldsymbol{r}\in\mathcal{V}}e^{-i\boldsymbol{k}\cdot\boldsymbol{r}}\boldsymbol{\mathcal{U}}(\boldsymbol{r},t)d\boldsymbol{r}, \tag{10}$$

where $V = |\mathcal{V}| = H^d$ is the volume of the system, and each *wavevector* $\boldsymbol{k} \equiv \boldsymbol{k}(\kappa)$ is expressed in terms of the integer *wave index* $\kappa \in \mathbb{Z}^d$, giving the set of discrete wavevectors

$$\widehat{\mathcal{V}} = \left\{ \boldsymbol{k} = 2\pi\kappa/H \mid \kappa \in \mathbb{Z}^d \right\}.$$

In Fourier space, the SPDE (5) becomes an infinite system of uncoupled stochastic ordinary differential equations (SODEs),

$$d\widehat{\boldsymbol{\mathcal{U}}}(t) = \widehat{\boldsymbol{\mathcal{L}}}\widehat{\boldsymbol{\mathcal{U}}}(t)dt + \widehat{\boldsymbol{\mathcal{K}}}d\widehat{\boldsymbol{\mathcal{B}}}(t), \tag{11}$$

one SODE for each $\boldsymbol{k} \in \widehat{\mathcal{V}}$. The invariant distribution of (11) is a zero-mean Gaussian random process, characterized fully by the covariance obtained from the spatial Fourier transform of (8),

$$\boldsymbol{\mathcal{S}}(\boldsymbol{k}, t) = V \left\langle \widehat{\boldsymbol{\mathcal{U}}}(\boldsymbol{k}, t')\widehat{\boldsymbol{\mathcal{U}}}^{\star}(\boldsymbol{k}, t'+t) \right\rangle = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\omega t} \boldsymbol{\mathcal{S}}(\boldsymbol{k}, \omega)\, d\omega, \tag{12}$$

where the *dynamic structure factor* (space-time spectrum) is

$$\boldsymbol{\mathcal{S}}(\boldsymbol{k}, \omega) = V \left\langle \widehat{\boldsymbol{\mathcal{U}}}(\boldsymbol{k}, \omega)\widehat{\boldsymbol{\mathcal{U}}}^{\star}(\boldsymbol{k}, \omega) \right\rangle = (\widehat{\boldsymbol{\mathcal{L}}} - i\omega)^{-1}(\widehat{\boldsymbol{\mathcal{K}}}\widehat{\boldsymbol{\mathcal{K}}}^{\star})(\widehat{\boldsymbol{\mathcal{L}}}^{\star} + i\omega)^{-1}, \tag{13}$$

which follows directly from the space-time $(\boldsymbol{k}, \omega)$ Fourier transform of the SPDE (5). By integrating the dynamic spectrum over all frequencies ω, one gets the *static structure factor*

$$\boldsymbol{\mathcal{S}}(\boldsymbol{k}) = \boldsymbol{\mathcal{S}}(\boldsymbol{k}, t = 0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \boldsymbol{\mathcal{S}}(\boldsymbol{k}, \omega)\, d\omega, \tag{14}$$

which is the spatial spectrum of an equilibrium snapshot of the fluctuating field and is the Fourier equivalent of $\boldsymbol{C}_{\boldsymbol{\mathcal{U}}}(t = 0)$. Note that the dynamic structure factor of spatiotemporal white noise is unity independent of the wavevector and wavefrequency: $\boldsymbol{\mathcal{S}}_{\boldsymbol{\mathcal{W}}}(\boldsymbol{k}, \omega) = \boldsymbol{I}$.

3A. ***Discretization.*** For the types of equations we will consider in this paper, the invariant measure is spatially white, specifically, $\boldsymbol{\mathcal{S}}(\boldsymbol{k})$ is diagonal and independent of $\boldsymbol{k}$. The associated fluctuating field $\boldsymbol{\mathcal{U}}$ cannot be evaluated pointwise, therefore, it is more natural to use *finite-volume* cell averages, denoted here by $\boldsymbol{U}$. In the deterministic setting, for uniform periodic grids there is no important difference between finite-volume and finite-difference methods. Our general approach can likely be extended also to analysis of stochastic finite-element discretizations, however, such methods have yet to be developed for the LLNS equations and here we focus on finite-volume methods. For notational simplicity, we will discuss problems in one spatial dimension ($d = 1$), with (mostly) obvious generalizations to higher dimensions.

Space is discretized into N_c identical cells of length $\Delta x = H/N_c$, and the value $\boldsymbol{U}_j$ stored in cell $1 \le j \le N_c$ is the average of the corresponding variable over the cell

$$\boldsymbol{U}_j(t) = \frac{1}{\Delta x} \int_{(j-1)\Delta x}^{j\Delta x} \boldsymbol{\mathcal{U}}(x,t)\,dx. \tag{15}$$

Time is discretized with a time step Δt, approximating cell averages of $\boldsymbol{\mathcal{U}}(x,t)$ pointwise in time with $\boldsymbol{U}^n = \{\boldsymbol{U}_1^n, \dots, \boldsymbol{U}_{N_c}^n\}$,

$$\boldsymbol{U}_j^n \approx \boldsymbol{U}_j(n\Delta t),$$

where $n \ge 0$ enumerates the time steps. The white noise $\boldsymbol{\mathcal{W}}(x,t)$ cannot be evaluated pointwise in either space or time and is discretized using a spatiotemporal average

$$\overline{\boldsymbol{\mathcal{W}}}_j^n(t) = \frac{1}{\Delta x \Delta t} \int_{n\Delta t}^{(n+1)\Delta t} \int_{(j-1)\Delta x}^{j\Delta x} \boldsymbol{\mathcal{W}}(x,t)\,dx\,dt, \tag{16}$$

which is a normal random variable with zero mean and variance $(\Delta x \Delta t)^{-1}$, independent between different cells and time steps. Note that for certain types of equations the dynamic structure factor may be white in frequency as well. In this case, a pointwise-in-time discretization is not appropriate and one can instead use a spatiotemporal average as done for white noise in (16).

We will study the accuracy of explicit linear finite-volume schemes for solving the SPDE (5). Rather generally, such methods are specified by a linear recursion of the form

$$\boldsymbol{U}^{n+1} = (\boldsymbol{I} + \boldsymbol{L}\Delta t)\,\boldsymbol{U}^n + \sqrt{\frac{\Delta t}{\Delta x}}\,\boldsymbol{K}\boldsymbol{W}^n, \tag{17}$$

where $\boldsymbol{L}$ and $\boldsymbol{K}$ are consistent stencil discretizations of the continuum differential operators $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{K}}$ (note that $\boldsymbol{L}$ and $\boldsymbol{K}$ may involve powers of Δt in general). Here

$$\boldsymbol{W}^n = (\Delta x \Delta t)^{1/2}\,\overline{\boldsymbol{\mathcal{W}}}^n \tag{18}$$

is a vector of standard normal variables with mean zero and variance one.

Without the random forcing, the deterministic equation $\boldsymbol{\mathcal{U}}_t = \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{U}}$ and the associated discretization can be studied using classical tools and notions of stability, consistency, and convergence. Under the assumption that the discrete generator $\boldsymbol{L}$ is dissipative, the initial condition $\boldsymbol{U}^0$ will be damped and the equilibrium solution will simply be a constant. The addition of the random forcing, however, leads to a nontrivial invariant measure (equilibrium distribution) of $\boldsymbol{U}^n$ determined by an interplay between the (discretized) fluctuations and dissipation. Because of the dissipative nature of the generator, any memory of the initial condition will eventually disappear and the long time dynamics is guaranteed to follow an ergodic trajectory that samples the unique invariant measure. In order to characterize

the accuracy of the stochastic integrator, we will analyze how well the discrete invariant measure (equilibrium distribution) reproduces the invariant measure of the continuum SPDE (this is a form of *weak convergence*). Note that due to ergodicity, ensemble averages can either be computed by averaging the power spectrum of the fields over multiple samples or averaging over time (after sufficiently many initial equilibration steps). In the theory we will consider the limit $n \to \infty$ and then average over different realizations of the noise $\boldsymbol{W}$ to obtain the discrete structure factors. In numerical calculations, we perform temporal averaging.

Regardless of the details of the iteration (17), $\boldsymbol{W}^n$ will always be a Gaussian random vector generated anew at each step n using a random number generator. The discretized field $\boldsymbol{U}^n$ is therefore a linear combination of Gaussian variates and it is therefore a Gaussian vector-valued stochastic process. In particular, the invariant measure (equilibrium distribution) of $\boldsymbol{U}^n$ is fully characterized by the covariance

$$\boldsymbol{C}^{(\boldsymbol{U})}_{j,j',n} = \lim_{N_s \to \infty} \left\langle \boldsymbol{U}_j^{N_s} \left(\boldsymbol{U}_{j'}^{N_s+n}\right)^{\star} \right\rangle, \tag{19}$$

which we would like to compare to the covariance of the continuum Gaussian field $\boldsymbol{C}_{\boldsymbol{\mathcal{U}}}(t = n\Delta t)$ given by (8). This comparison is best done in the Fourier domain by using the spatial discrete Fourier transform, defined for a spatially discrete field $\boldsymbol{U}$ (for example, $\boldsymbol{U} \equiv \boldsymbol{U}^n$ or $\boldsymbol{U} \equiv \boldsymbol{U}(t)$) via

$$\boldsymbol{U}_j = \sum_{k \in \widehat{\mathcal{V}}_d} \widehat{\boldsymbol{U}}_k e^{ij\Delta k}, \tag{20}$$

$$\widehat{\boldsymbol{U}}_k = \frac{1}{V} \sum_{j=0}^{N_c-1} \boldsymbol{U}_{j+1} e^{-ij\Delta k} \Delta x, \tag{21}$$

where we have denoted the discrete *dimensionless* wavenumber

$$k\Delta x = 2\pi\kappa/N_c,$$

and the wave index is now limited to the first N_c values,

$$\widehat{\mathcal{V}}_d = \{k = 2\pi\kappa/H \mid 0 \le \kappa < N_c\} \subset \mathcal{V}.$$

Since the fields are real-valued, there is a redundancy in the Fourier coefficients $\widehat{\boldsymbol{U}}_k$ because of the Hermitian symmetry between κ and $N_c - \kappa$ (essentially, the second half of the wave indices correspond to negative k), and thus we will only consider $0 \le \kappa \le \lfloor N_c/2 \rfloor$, giving a (Nyquist) cutoff wavenumber $k_{\max} \approx \pi/\Delta x$.

What we would like to compare is the Fourier coefficients of the numerical approximation, $\widehat{\boldsymbol{U}}^n_k$, with the Fourier coefficients of the continuum solution

$$\widehat{\boldsymbol{\mathcal{U}}}_k(t = n\Delta t).$$

The invariant measure of $\widehat{\boldsymbol{U}}_k^n$ has zero mean and is characterized by the covariance obtained from the spatial Fourier transform of (19),

$$\boldsymbol{S}_{k,n} = V \lim_{N_s\to\infty} \left\langle \widehat{\boldsymbol{U}}_k^{N_s} \left(\widehat{\boldsymbol{U}}_k^{N_s+n}\right)^\star \right\rangle . \tag{22}$$

From the definition of the discrete Fourier transform it follows that for small Δk, that is, smooth Fourier basis functions on the scale of the discrete grid, $\widehat{\boldsymbol{U}}_k(t)$ converges to the Fourier coefficient $\widehat{\boldsymbol{\mathcal{U}}}(k, t = n\Delta t)$ of the continuum field. Therefore, $\boldsymbol{S}_{k,n}$ is the discrete equivalent (numerical approximation) to the continuum structure factor $\boldsymbol{\mathcal{S}}(k, t = n\Delta t)$. We define a discrete approximation to be *weakly consistent* if

$$\lim_{\Delta x,\Delta t\to 0} \boldsymbol{S}_{k,n=\lfloor t/\Delta t\rfloor} = \boldsymbol{\mathcal{S}}\,(k, t)\,,$$

for any chosen $k \in \widehat{\mathcal{V}}$ and t. This means that, given a sufficiently fine discretization, the numerical scheme can accurately reproduce the structure factor for a desired wave index and time lag. An alternative view is that a convergent scheme reproduces the slow (compared to Δt) and large-scale (compared to Δx) fluctuations, that is, it accurately reproduces the dynamic structure factor $\boldsymbol{\mathcal{S}}(k, \omega)$ for small $\Delta k = k\Delta x$ and $\Delta\omega = \omega\Delta t$. Our goal here is to quantify this for several numerical methods for solving stochastic conservation laws and optimize the numerical schemes by tuning parameters to obtain the best possible approximation to $\boldsymbol{\mathcal{S}}(k, \omega)$ for small k and ω.

Much of our analysis will be focused on the *discrete static structure factor*

$$\boldsymbol{S}_k = \boldsymbol{S}_{k,0} = V \lim_{N_s\to\infty} \left\langle \widehat{\boldsymbol{U}}_k^{N_s} \left(\widehat{\boldsymbol{U}}_k^{N_s}\right)^\star \right\rangle .$$

Note that for a spatially white field $\boldsymbol{\mathcal{U}}(x)$, the finite-volume averages $\boldsymbol{U}_j$ are independent Gaussian variates with mean zero and variance Δx^{-1}, and the discrete Fourier coefficients $\widehat{\boldsymbol{U}}_k$ are independent Gaussian variates with mean zero and variance V^{-1}. As a measure of the accuracy of numerical schemes for solving (5), we will compare the discrete static structure factors $\boldsymbol{S}_k$ with the continuum prediction $\boldsymbol{\mathcal{S}}(k)$, for all of the discrete wavenumbers (i.e., pointwise in Fourier space). It is expected that any numerical scheme will produce some artifacts at the largest wavenumbers because of the strong corrections due to the discretization; however, small wavenumbers ought to have much smaller errors because they evolve over time scales and length scales much larger than the discretization step sizes. Specifically, we propose to look at the series expansions

$$\boldsymbol{S}_k - \boldsymbol{\mathcal{S}}(k) = O(\Delta t^{p_1} k^{p_2}),$$

and optimize the numerical schemes by maximizing the powers p_1 and p_2. Next we describe the general formalism used to obtain explicit expressions for the discrete structure factors $\boldsymbol{S}_k$ for a general explicit method, and then illustrate the

formalism on some simple examples, before attacking the more complex equations of fluctuating hydrodynamics.

3B. ***Analysis of linear explicit methods.*** Regardless of the details of a particular scheme and the particular linear SPDE being solved, at the end of the time step a typical explicit scheme makes a linear combination of the values in the neighboring cells and random variates to produce an updated value,

$$\boldsymbol{U}_j^{n+1} = \boldsymbol{U}_j^n + \sum_{\Delta j=-w_D}^{\Delta j=w_D} \boldsymbol{\Phi}_{\Delta j} \boldsymbol{U}_{j+\Delta j}^n + \sum_{\Delta j=-w_S}^{\Delta j=w_S} \boldsymbol{\Psi}_{\Delta j} \boldsymbol{W}_{j+\Delta j}^n, \tag{23}$$

where w_D and w_S are the deterministic and stochastic stencil widths. The particular forms of the matrices of coefficients $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ depend on the scheme, and will involve powers of Δt and Δx. Here we assume that for each n the random increment $\boldsymbol{W}^n$ is an independent vector of N_s normal variates with covariance

$$\boldsymbol{C}_{\boldsymbol{W}} = \left\langle \boldsymbol{W}_j^n (\boldsymbol{W}_j^n)^\star \right\rangle$$

constant for all of the cells j and thus wavenumbers, where N_s is the total number of random numbers utilized per cell per stage. Computer algebra systems can be used to obtain explicit formulas for the matrices in (23); we have made extensive use of Maple for the calculations presented in this paper.

Assuming a translation invariant scheme, the iteration (23) can easily be converted from real space to an iteration in Fourier space,

$$\widehat{\boldsymbol{U}}_k^{n+1} = \widehat{\boldsymbol{U}}_k^n + \sum_{\Delta j=-w_D}^{\Delta j=w_D} \boldsymbol{\Phi}_{\Delta j} \widehat{\boldsymbol{U}}_k^n \exp(i \Delta j \Delta k) + \sum_{\Delta j=-w_S}^{\Delta j=w_S} \boldsymbol{\Psi}_{\Delta j} \widehat{\boldsymbol{W}}_k^n \exp(i \Delta j \Delta k), \tag{24}$$

where different wavenumbers are not coupled to each other. In general, any linear explicit method can be represented in Fourier space as a recursion of the form

$$\widehat{\boldsymbol{U}}_k^{n+1} = \boldsymbol{M}_k \widehat{\boldsymbol{U}}_k^n + \boldsymbol{N}_k \widehat{\boldsymbol{W}}_k^n, \tag{25}$$

where the explicit form of the matrices $\boldsymbol{M}_k$ and $\boldsymbol{N}_k$ depend on the particular scheme and typically contain various powers of $\sin \Delta k$, $\cos \Delta k$, and Δt, and

$$\boldsymbol{C}_{\widehat{\boldsymbol{W}}} = \left\langle \widehat{\boldsymbol{W}}_k^n \left(\widehat{\boldsymbol{W}}_k^n\right)^\star \right\rangle = N_c^{-1} \boldsymbol{C}_{\boldsymbol{W}}.$$

By iterating this recurrence relation, we can easily obtain (assuming $\widehat{\boldsymbol{U}}_k^0 = 0$)

$$\widehat{\boldsymbol{U}}_k^{n+1} = \sum_{l=0}^{n} (\boldsymbol{M}_k)^l \, \boldsymbol{N}_k \widehat{\boldsymbol{W}}_k^{n-l},$$

from which we can calculate

$$\boldsymbol{S}_k^n = V\left\langle \left(\widehat{\boldsymbol{U}}_k^n\right)\left(\widehat{\boldsymbol{U}}_k^n\right)^\star \right\rangle = \sum_{l=0}^{n-1} (\boldsymbol{M}_k)^l (\Delta x \boldsymbol{N}_k \boldsymbol{C}_{\boldsymbol{W}} \boldsymbol{N}_k^\star)(\boldsymbol{M}_k^\star)^l = \sum_{l=0}^{n-1} (\boldsymbol{M}_k)^l \widetilde{\boldsymbol{C}} (\boldsymbol{M}_k^\star)^l.$$

In order to calculate this sum explicitly, we will use the identity

$$\boldsymbol{M}_k \boldsymbol{S}_k^n \boldsymbol{M}_k^\star - \boldsymbol{S}_k^n = (\boldsymbol{M}_k)^n \widetilde{\boldsymbol{C}} (\boldsymbol{M}_k^\star)^n - \widetilde{\boldsymbol{C}} \tag{26}$$

to obtain a linear system for the entries of the matrix $\boldsymbol{S}_k^n$. If the deterministic method is stable, which means that all eigenvalues of the matrix $\boldsymbol{M}_k$ are below unity for all wavenumbers, then in the limit $n \to \infty$ the first term on the right side will vanish, to give

$$\boldsymbol{M}_k \boldsymbol{S}_k \boldsymbol{M}_k^\star - \boldsymbol{S}_k = -\Delta x \boldsymbol{N}_k \boldsymbol{C}_{\boldsymbol{W}} \boldsymbol{N}_k^\star. \tag{27}$$

If one assumes existence of a unique structure factor, Equation (27) can be most directly obtained from the condition of stationarity $\boldsymbol{S}_k^{n+1} = \boldsymbol{S}_k^n \equiv \boldsymbol{S}_k$,

$$\left\langle \left(\boldsymbol{M}_k \widehat{\boldsymbol{U}}_k^n + \boldsymbol{N}_k \widehat{\boldsymbol{W}}_k^n\right)\left(\boldsymbol{M}_k \widehat{\boldsymbol{U}}_k^n + \boldsymbol{N}_k \widehat{\boldsymbol{W}}_k^n\right)^\star \right\rangle = \left\langle \left(\widehat{\boldsymbol{U}}_k^n\right)\left(\widehat{\boldsymbol{U}}_k^n\right)^\star \right\rangle = V^{-1} \boldsymbol{S}_k,$$

giving a path to easily extend the analysis to more complicated situations such as multistep schemes.

Equation (27) is a linear system of equations for the equilibrium static structure factor produced by a given scheme, where the number of unknowns is equal to the square of the number of variables (field components). By simply deleting the subscripts k one obtains a more general but much larger linear system [36] for the real space equilibrium covariance of a snapshot of the discrete field $\boldsymbol{C}_{j,j'}^{(\boldsymbol{U})} = \boldsymbol{C}_{j,j',n=0}^{(\boldsymbol{U})}$:

$$\boldsymbol{M} \boldsymbol{C}_{\boldsymbol{U}} \boldsymbol{M}^\star - \boldsymbol{C}_{\boldsymbol{U}} = -\Delta x \boldsymbol{N} \boldsymbol{C}_{\boldsymbol{W}}^{(N_c)} \boldsymbol{N}^\star,$$

where

$$\boldsymbol{C}_{\boldsymbol{W}}^{(N_c)} = \left\langle \boldsymbol{W}^n (\boldsymbol{W}^n)^\star \right\rangle$$

is the covariance matrix of the random increments. Note that this relation continues to hold even for schemes that are not translation invariant such as generalizations to nonperiodic boundary conditions; however, the number of unknowns is now the square of the total number of degrees of freedom so that explicit solutions will in general not be possible. Based on standard wisdom for deterministic schemes, it is expected that schemes that perform well under periodic boundary conditions will also perform well in the presence of boundaries when the discretization is suitably modified only near the boundaries.

A similar approach to the one illustrated above for the static structure factor can be used to evaluate the discrete *dynamic* structure factor

$$\boldsymbol{S}_{k,\omega} = \lim_{N_s \to \infty} V\,(N_s \Delta t)\left\langle \widehat{\boldsymbol{U}}_{k,\omega}^{N_s} \left(\widehat{\boldsymbol{U}}_{k,\omega}^{N_s}\right)^\star \right\rangle$$

from the time-discrete Fourier transform

$$\widehat{\boldsymbol{U}}^{N_s}_{k,\omega} = \frac{1}{N_s}\sum_{l=0}^{N_s} \exp\left(-il\,\Delta\omega\right)\widehat{\boldsymbol{U}}^{l}_{k},$$

where $\Delta\omega = \omega\Delta t$, and the frequency is less than the Nyquist cutoff $\omega \leq \pi/\Delta t$. The calculation yields

$$\boldsymbol{S}_{k,\omega} = [\boldsymbol{I} - \exp\left(-i\,\Delta\omega\right)\boldsymbol{M}_k]^{-1}(\Delta x\,\Delta t\,\boldsymbol{N}_k\boldsymbol{C}_{\boldsymbol{W}}\boldsymbol{N}_k^{\star})[\boldsymbol{I} - \exp\left(i\,\Delta\omega\right)\boldsymbol{M}_k^{\star}]^{-1}. \quad (28)$$

Equation (28) can be seen as discretized forms of the continuum version (13) in the limits $\Delta k \to 0$, $\Delta t \to 0$ (the corresponding correlations in the time-domain are given in [36]).

Equations (27) and (28) are the main result of this section and we have used it to obtain explicit expressions for $\boldsymbol{S}_k$ and $\boldsymbol{S}_{k,\omega}$ for several equations and schemes. Many of our results are in fact rather general; however, for clarity and specificity, in the next sections we will illustrate the above formalism for several simple examples of stochastic conservation laws.

3B1. *Discrete fluctuation-dissipation balance.* We consider first the static structure factors for very small time steps. In the limit $\Delta t \to 0$, temporal terms of order two or more can be ignored so that all time-integration methods behave like an explicit first-order Euler iteration as in (17),

$$\widehat{\boldsymbol{U}}^{n+1}_{k} = \left(\boldsymbol{I} + \Delta t\,\widehat{\boldsymbol{L}}^{(0)}_{k}\right)\widehat{\boldsymbol{U}}^{n}_{k} + \sqrt{\frac{\Delta t}{\Delta x}}\,\widehat{\boldsymbol{K}}^{(0)}_{k}\,\widehat{\boldsymbol{W}}_{k}, \quad (29)$$

where $\boldsymbol{L}^{(0)} = \boldsymbol{L}\,(\Delta t = 0)$ can be thought of as the spatial discretization of the generator $\boldsymbol{\mathcal{L}}$, and $\boldsymbol{K}^{(0)} = \boldsymbol{K}\,(\Delta t = 0)$ is the spatial discretization of the filtering operator $\boldsymbol{\mathcal{K}}$. Comparing to (25) we can directly identify $\boldsymbol{M}_k = \boldsymbol{I} + \Delta t\,\widehat{\boldsymbol{L}}^{(0)}_{k}$ and $\boldsymbol{N}_k = \sqrt{\Delta t/\Delta x}\,\widehat{\boldsymbol{K}}^{(0)}_{k}$ and substitute these into (27). Keeping only terms of order Δt on both sides we obtain the condition

$$\widehat{\boldsymbol{L}}^{(0)}_{k}\,\boldsymbol{S}^{(0)}_{k} + \boldsymbol{S}^{(0)}_{k}\left(\widehat{\boldsymbol{L}}^{(0)}_{k}\right)^{\star} = -\widehat{\boldsymbol{K}}^{(0)}_{k}\,\boldsymbol{C}_{\boldsymbol{W}}\left(\widehat{\boldsymbol{K}}^{(0)}_{k}\right)^{\star}, \quad (30)$$

where $\boldsymbol{S}^{(0)}_{k} = \lim_{\Delta t\to 0}\boldsymbol{S}_k$ (see also a related real-space derivation using Ito's calculus in [6], as well as in [36, Section VIII]). It can be shown that if $\widehat{\boldsymbol{L}}^{(0)}_{k}$ is definite, (30) has a unique solution. Assuming that $\boldsymbol{W}$ is as given in (18), that is, that $\boldsymbol{C}_{\boldsymbol{W}} = \boldsymbol{I}$, and that the spatial discretizations of the generator and filter operators satisfy a *discrete fluctuation-dissipation balance*

$$\widehat{\boldsymbol{L}}^{(0)}_{k} + \left(\widehat{\boldsymbol{L}}^{(0)}_{k}\right)^{\star} = -\widehat{\boldsymbol{K}}^{(0)}_{k}\left(\widehat{\boldsymbol{K}}^{(0)}_{k}\right)^{\star}, \quad (31)$$

we see that $\boldsymbol{S}^{(0)}_{k} = \boldsymbol{I}$ is the solution to (30), that is, at equilibrium the discrete fields are spatially white. The discrete fluctuation-dissipation balance condition can also

be written in real space:

$$\boldsymbol{L}^{(0)} + (\boldsymbol{L}^{(0)})^\star = -\boldsymbol{K}^{(0)}(\boldsymbol{K}^{(0)})^\star. \tag{32}$$

The condition (32) is the discrete equivalent of the continuum fluctuation-dissipation balance condition [44]

$$\mathcal{L} + \mathcal{L}^\star = -\mathcal{K}\mathcal{K}^\star, \tag{33}$$

which ensures that $\boldsymbol{\mathcal{S}}(k) = \boldsymbol{I}$, that is, that the invariant measure of the SPDE is spatially white. We observe that adding a skew adjoint component to $\mathcal{L}$ does not alter the fluctuation-dissipation balance above, as is the case with nondissipative (advective) terms. Numerous equations [47] modeling conservative thermal systems satisfy condition (33), including the linearized LLNS equations (with some additional prefactors). In essence, the fluctuations injected at all scales by the spatially white forcing $\mathcal{W}$ are filtered by $\mathcal{K}$ and then dissipated by $\mathcal{L}$ at just equal rates.

Assuming a spatial discretization satisfies the discrete fluctuation-dissipation balance condition, it is possible to extend the above analysis to higher powers of Δt and analyze the corrections to the structure factors for finite time steps. Some general conclusions can be reached in this way, for example, the Euler method is first-order accurate, predictor-corrector methods are at least second-order accurate, while the Crank–Nicolson semi-implicit method gives $\boldsymbol{S}_k = \boldsymbol{I}$ for any time step. We will demonstrate these results for specific examples in the next section, including the spatial truncation errors as well.

4. Linear stochastic conservation laws

The remainder of this paper is devoted to the study of the accuracy of finite-volume methods for solving linear stochastic PDEs in conservation form,

$$\partial_t \mathcal{U} = -\nabla\cdot[(\boldsymbol{A}\mathcal{U} - \boldsymbol{C}\nabla\mathcal{U}) - \boldsymbol{E}\mathcal{W}], \tag{34}$$

where $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{E}$ are constants, and $\mathcal{W}$ is Gaussian spatiotemporal white noise. The white noise forcing and its divergence here need to be interpreted in the (weak) sense of distributions since they lack the regularity required for the classical definitions. The linearization of the LLNS equations (1) leads to a system of the form (34), as do a number of other classical PDEs [47], such as the *stochastic advection-diffusion equation*

$$\partial_t T = -\boldsymbol{a}\cdot\nabla T + \mu\nabla^2 T + \sqrt{2\mu}\nabla\cdot\mathcal{W}, \tag{35}$$

where $T(\boldsymbol{r}, t) \equiv \boldsymbol{U}(\boldsymbol{r}, t)$ is a scalar stochastic field, $\boldsymbol{A} \equiv \boldsymbol{a}$ is the advective velocity, $\boldsymbol{C} \equiv \mu\boldsymbol{I}$, $\mu > 0$ is the diffusion coefficient, and $\boldsymbol{E} \equiv \sqrt{2\mu}\boldsymbol{I}$. The simplest case is the *stochastic heat equation*, obtained by taking $\boldsymbol{a} = \boldsymbol{0}$.

A key feature of the type of system considered here is that the noise is intrinsic to the system and appears in the flux as opposed to commonly treated systems that include an external stochastic forcing term, such as the form of a stochastic heat equation considered in [21]. Since white noise is more regular than the spatial derivative of white noise, external noise leads to more regular equilibrium fields (e.g., continuous functions in one dimension). Intrinsic noise, on the other hand, leads to very irregular equilibrium fields. Notationally, it is convenient to write (34) as

$$\partial_t \boldsymbol{\mathcal{U}} = -\boldsymbol{\mathcal{D}}(\boldsymbol{A}\boldsymbol{\mathcal{U}} - \boldsymbol{C}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{U}} - \boldsymbol{E}\boldsymbol{\mathcal{W}}), \tag{36}$$

defining the divergence $\boldsymbol{\mathcal{D}} \equiv \nabla\cdot$ and gradient $\boldsymbol{\mathcal{G}} \equiv \nabla$ operators, $\boldsymbol{\mathcal{D}}^\star = -\boldsymbol{\mathcal{G}}$. In the types of equations that appear in hydrodynamics, such as the LLNS equations, the operator $\boldsymbol{\mathcal{D}}\boldsymbol{A}$ is skew-adjoint, $(\boldsymbol{\mathcal{D}}\boldsymbol{A})^\star = -\boldsymbol{\mathcal{D}}\boldsymbol{A}$ (hyperbolic or advective flux), $\boldsymbol{C} \succeq \boldsymbol{0}$ (dissipative or diffusive flux), and $\boldsymbol{E}\boldsymbol{E}^\star = 2\boldsymbol{C}$, that is, $\boldsymbol{E}^\star = (2\boldsymbol{C})^{1/2}$. Therefore, the generator $\boldsymbol{\mathcal{L}} = -\boldsymbol{\mathcal{D}}\boldsymbol{A} + \boldsymbol{\mathcal{D}}\boldsymbol{C}\boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{D}}\boldsymbol{A})^\star - \boldsymbol{\mathcal{D}}\boldsymbol{C}\boldsymbol{\mathcal{D}}^*$ and filter $\boldsymbol{\mathcal{K}} = \boldsymbol{\mathcal{D}}\boldsymbol{E}$ satisfy the fluctuation-dissipation balance condition (33) and the equilibrium distribution is spatially white. Note that even though advection makes some of the eigenvalues of $\boldsymbol{\mathcal{L}}$ complex, the generator is dissipative and (34) has a unique invariant measure because the real part of all of the eigenvalues of $\boldsymbol{\mathcal{L}}$ is negative except for the unique zero eigenvalue.

It is important to point out that discretizations of the continuum operators do not necessarily satisfy the discrete fluctuation-dissipation condition (32). One way to ensure the condition is satisfied is to discretize the diffusive components of the generator $\boldsymbol{L}_D = \boldsymbol{D}\boldsymbol{C}\boldsymbol{G}$ and the filter $\boldsymbol{K} = \boldsymbol{D}\boldsymbol{E}$ using a discrete divergence $\boldsymbol{D}$ and discrete gradient $\boldsymbol{G}$ so that the discrete fluctuation-dissipation balance condition $\boldsymbol{L}_D + \boldsymbol{L}_D^\star = -\boldsymbol{K}\boldsymbol{K}^\star$ holds. If, however, the discretization of the advective component of the generator $\boldsymbol{L}_A = -\boldsymbol{D}\boldsymbol{A}$ is not skew-adjoint, this can perturb the balance (31). Notably, various upwinding methods lead to discretizations that are not skew-adjoint. The correction to the structure factor $\boldsymbol{S}_k^{(0)} = \boldsymbol{I} + \Delta\boldsymbol{S}_k^{(0)}$ due to a nonzero $\Delta\boldsymbol{L}_A = (\boldsymbol{L}_A + \boldsymbol{L}_A^\star)/2$ can easily be obtained from (30), and in one dimension the result is simply

$$\Delta S_k^{(0)} = -\frac{\Delta L_k^{(A)}}{L_k^{(D)} + \Delta L_k^{(A)}}. \tag{37}$$

We will use centered differences for the advective generator in this work, which ensures a skew-adjoint $\boldsymbol{L}_A$, and our focus will therefore be on satisfying the discrete fluctuation-dissipation balance between the diffusive and stochastic terms.

4A. ***Finite-volume numerical schemes.*** We consider here rather general finite-volume methods for solving the linear SPDE (34) in one dimension,

$$\partial_t \boldsymbol{\mathcal{U}} = -\frac{\partial}{\partial x}[\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}) - \boldsymbol{\mathcal{Z}}] = -\frac{\partial}{\partial x}\left[\left(\boldsymbol{A} - \boldsymbol{C}\frac{\partial}{\partial x}\right)\boldsymbol{\mathcal{U}} - \boldsymbol{E}\boldsymbol{\mathcal{W}}\right] \tag{38}$$

with periodic boundaries, where we have denoted the stochastic flux with $\mathcal{Z} = \boldsymbol{E}\mathcal{W}$. As for classical finite-volume methods for the deterministic case, we start from the PDE and integrate the left and right sides over a given cell j over a given time step Δt, and use integration by parts to obtain the formally exact

$$\boldsymbol{U}_j^{n+1} = \boldsymbol{U}_j^n - \frac{\Delta t}{\Delta x}(\boldsymbol{F}_{j+1/2} - \boldsymbol{F}_{j-1/2}) + \frac{\Delta t}{\Delta x}\left(\frac{1}{\sqrt{\Delta x\,\Delta t}}\right)(\boldsymbol{Z}_{j+1/2} - \boldsymbol{Z}_{j-1/2}), \quad (39)$$

where the *deterministic discrete fluxes* $\boldsymbol{F}$ and *stochastic discrete fluxes* $\boldsymbol{Z}$ are calculated on the boundaries of the cells (points in one dimension, edges in two dimensions, and faces in three dimensions), indexed here with half-integers. These fluxes represent the total rate of transport through the interface between two cells over a given finite time interval Δt, and (39) is nothing more than a restatement of conservation. The classical interpretation of pointwise evaluation of the fluxes is not appropriate because white noise forcing lacks the regularity of classical smooth forcing and cannot be represented in a finite basis. Instead, just as we projected the fluctuating fields using finite-volume averaging, we ought to project the stochastic fluxes $\mathcal{Z}$ to a finite representation $\overline{\mathcal{Z}} = (\Delta x\,\Delta t)^{-1/2}\boldsymbol{Z}$ through spatio-temporal averaging, as done in (16) and (18). For the purposes of our analysis, one can simply think of the discrete fluxes as an approximation that has the same spectral properties as the corresponding continuum Gaussian fields over the wavevectors and frequencies represented by the finite discretization.

The goal of numerical methods is to approximate the fluxes as best as possible. In general, within each time step of a scheme there may be N_{st} stages or substeps; for example, in the classic MacCormack method there is a predictor and a corrector stage ($N_{st} = 2$), and in the three-stage Runge–Kutta method of Williams et al. [13], there are three stages ($N_{st} = 3$). Each stage $0 < s \le N_{st}$ is of the conservative form (39):

$$\boldsymbol{U}_j^{n+s/N_{st}} = \sum_{s'=0}^{s-1} \alpha_{s'}^{(s)} \boldsymbol{U}_j^{n+s'/N_{st}} - \frac{\Delta t}{\Delta x}(\boldsymbol{F}_{j+1/2}^{(s)} - \boldsymbol{F}_{j-1/2}^{(s)}) + \frac{\Delta t^{1/2}}{\Delta x^{3/2}}(\boldsymbol{Z}_{j+1/2}^{(s)} - \boldsymbol{Z}_{j-1/2}^{(s)}), \quad (40)$$

where the α's are some coefficients, $\sum_{s'=0}^{s-1} \alpha_{s'}^{(s)} = 1$, and each of the stage fluxes are partial approximations of the continuum flux. For the stochastic integrators we discuss here, the deterministic fluxes are calculated the same way as they would be in the corresponding deterministic scheme. In general, the stochastic fluxes $\boldsymbol{Z}_{j+1/2}$ can be expressed in terms of independent unit normal variates $\boldsymbol{W}_{j+1/2}$ that are sampled using a random number generator. The stochastic fluxes in each stage may be the same, may be completely independent, or they may have nontrivial correlations between stages.

Note that it is possible to avoid noninteger indices by reindexing the fluxes in (39) and writing it in a form consistent with (23):

$$\boldsymbol{U}_j^{n+1} = \boldsymbol{U}_j^n - \frac{\Delta t}{\Delta x}(\boldsymbol{F}_j - \boldsymbol{F}_{j-1}) + \frac{\Delta t^{1/2}}{\Delta x^{3/2}}(\boldsymbol{Z}_j - \boldsymbol{Z}_{j-1}). \tag{41}$$

However, when considering the order of accuracy of the stencils and also fluctuation-dissipation balance in higher dimensions, it will become important to keep in mind that the fluxes are evaluated on the faces (edges or half-grid points) of the grid, and therefore we will keep the half-integer indices. Note that for face-centered values, such as fluxes, it is best to add a phase factor $\exp(i\Delta k/2)$ in the definition of the Fourier transform, even though such pure phase shifts will not affect the correlation functions and structure factors.

Before we analyze schemes for the complex LLNS equations, we present an illustrative explicit calculation for the one-dimensional stochastic heat equation.

5. Example: stochastic heat equation

We now illustrate the general formalism presented in Section 4 for the simple case of an Euler and predictor-corrector scheme for solving the stochastic heat equation in one dimension,

$$\upsilon_t = \mu \upsilon_{xx} + \sqrt{2\mu}\mathcal{W}_x, \tag{42}$$

where $\upsilon(x,t) \equiv \mathcal{U}(x,t)$ is a scalar field and μ is the mass or heat diffusion coefficient. The solution in the Fourier domain is trivial, giving

$$S(k,\omega) = \frac{2\mu k^2}{\omega^2 + \mu^2 k^4} \quad \text{and} \quad S(k) = 1. \tag{43}$$

5A. ***Static structure factor.*** We first study a simple second-order spatial discretization of the dissipative fluxes

$$F_{j+1/2} = \frac{\mu}{\Delta x}(u_{j+1} - u_j),$$

combined with an Euler integration in time, to give a simple numerical method for solving the SPDE (42):

$$u_j^{n+1} = u_j^n + \frac{\mu\Delta t}{\Delta x^2}(u_{j-1}^n - 2u_j^n + u_{j+1}^n) + \sqrt{2\mu}\frac{\Delta t^{1/2}}{\Delta x^{3/2}}(W_{j+1/2}^n - W_{j-1/2}^n), \tag{44}$$

where $u \equiv U$ and the W's are independent unit normal random numbers with zero mean generated anew at every time step (here $N_s = N_{st} = 1$). From (44), we can extract the recursion coefficients appearing in (25),

$$M_k = 1 + \beta(e^{-i\Delta k} - 2 + e^{i\Delta k}) = 1 + 2\beta(\cos\Delta k - 1),$$

$$N_k = \sqrt{2\mu}\frac{\Delta t^{1/2}}{\Delta x^{3/2}}(e^{i\Delta k/2} - e^{-i\Delta k/2}),$$

where

$$\beta = \frac{\mu \Delta t}{\Delta x^2}$$

denotes a dimensionless diffusive time step (ratio of the time step to the diffusive CFL limit). Together with $C_W = 1$, Equation (27) becomes a scalar equation for the discrete structure factor

$$(M_k M_k^\star - 1) S_k = -\Delta x N_k N_k^\star,$$

with dimensionless solution

$$S_k = \frac{4\beta(1-\cos\Delta k)}{(1-M_k^2)} = [1+\beta(\cos\Delta k - 1)]^{-1}. \tag{45}$$

The time-dependent result can also easily be derived from (26):

$$S_k^n = (1 - e^{-t/\tau}) S_k, \qquad \text{where } t = n\Delta t,$$

and $\tau^{-1} = 4\mu\,(\cos\Delta k - 1)\,/\Delta x^2 \approx 2\mu k^2$ is the familiar relaxation time for wavenumber k, showing that the smallest wavenumbers take a long time to reach the equilibrium distribution.

Equation (45) is a vivid illustration of the typical result for schemes for stochastic transport equations based on finite difference stencils, also shown in Figure 1. Firstly, we see that for small k we have that $S_k \approx 1 + \beta\Delta k^2/2$, showing that the smallest wavenumbers are correctly handled by the discretization for any time step. Also, this shows that the error in the structure factor is of order β, that is, of order Δt, as expected for the Euler scheme, whose weak order of convergence is one for SODEs. Finally, it shows that the error grows quadratically with k (from symmetry arguments, only even powers will appear). By looking at the largest wavenumber, $\Delta k_{\max} = \pi$, we see that $S_{k_{\max}} = (1-2\beta)^{-1}$, from which we instantly see the CFL stability condition $\beta < 1/2$, which guarantees that the structure factor is finite and positive for all $0 \le k \le \pi$. Furthermore, we see that for $\beta \ll 1$, the structure factor is approximately unity for all wavenumbers. That is, a sufficiently small step will indeed reproduce the proper equilibrium distribution.

By contrast, a two-stage predictor-corrector scheme for the diffusion equation,

$$\begin{aligned}
\tilde{u}_j^n &= u_j^n + \frac{\mu\Delta t}{\Delta x^2}(u_{j-1}^n - 2u_j^n + u_{j+1}^n) + \sqrt{2\mu}\frac{\Delta t^{1/2}}{\Delta x^{3/2}}(W_{j+1/2}^n - W_{j-1/2}^n),\\
u_j^{n+1} &= \frac{1}{2}\Big[u_j^n + \tilde{u}_j^n + \frac{\mu\Delta t}{\Delta x^2}(\tilde{u}_{j-1}^n - 2\tilde{u}_j^n + \tilde{u}_{j+1}^n) + \sqrt{2\mu}\frac{\Delta t^{1/2}}{\Delta x^{3/2}}(W_{j+1/2}^n - W_{j-1/2}^n)\Big].
\end{aligned} \tag{46}$$

achieves much higher accuracy, namely, a structure factor that deviates from unity by a higher order in both Δt and k,

$$\text{PC-1RNG: } S_k \approx 1 - \tfrac{1}{4}\beta^2\Delta k^4,$$

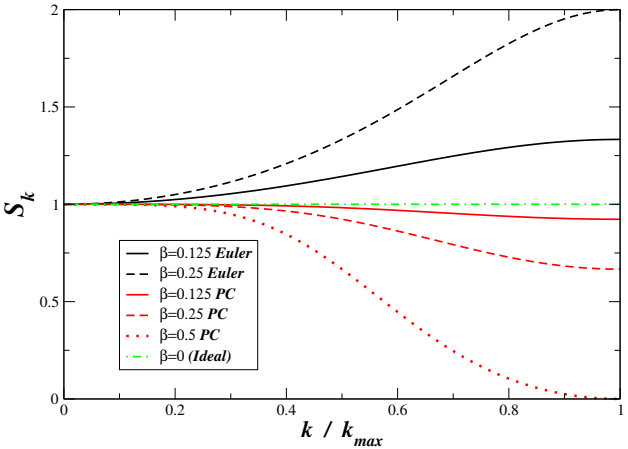


Figure 1. An illustration of the discrete structure factor S_k for the Euler (44) and predictor-corrector (46) schemes for the stochastic heat equation (42).

as illustrated in Figure 1. We can also use different stochastic fluxes in the predictor and the corrector stages (i.e., use $N_s = 2$ random numbers per cell per stage), with an added prefactor of $\sqrt{2}$ to compensate for the variance reduction of the averaging between the two stages,

$$\begin{aligned}
\tilde{u}_j^n &= u_j^n + \frac{\mu\Delta t}{\Delta x^2}(u_{j-1}^n - 2u_j^n + u_{j+1}^n) + 2\sqrt{\mu}\frac{\Delta t^{1/2}}{\Delta x^{3/2}}\big(W_{j+1/2}^{(n,P)} - W_{j-1/2}^{(n,P)}\big),\\
u_j^{n+1} &= \frac{1}{2}\Big[u_j^n + \tilde{u}_j^n + \frac{\mu\Delta t}{\Delta x^2}(\tilde{u}_{j-1}^n - 2\tilde{u}_j^n + \tilde{u}_{j+1}^n) + 2\sqrt{\mu}\frac{\Delta t^{1/2}}{\Delta x^{3/2}}(W_{j+1/2}^{(n,C)} - W_{j-1/2}^{(n,C)})\Big].
\end{aligned} \tag{47}$$

For the scheme (47) the analysis reveals an even greater spatiotemporal accuracy of the static structure factors, namely, third order temporal accuracy:

$$\text{PC-2RNG: } S_k \approx 1 + \tfrac{1}{8}\beta^3\Delta k^6.$$

This illustrates the importance of the handling of the stochastic fluxes in multi-stage algorithms, as we will come back to shortly. Note, however, that the PC-1RNG method (46) may be preferred in practice over the PC-2RNG method (47) even though using two random numbers per step gives greater accuracy for small wavenumbers for small time steps. This is not only because of the computational savings of generating half the random numbers, but also because PC-1RNG is better-behaved (more stable) at large wavenumbers for large time steps. Specifically, the structure factor can become rather large for $\Delta k = \pi$ for PC-2RNG for $\beta > 0.1$.

The analysis we presented here for explicit methods can easily be extended to implicit and semi-implicit schemes as well, as illustrated in the Appendix for the Crank–Nicolson method for the stochastic heat equation.

Previous studies [13; 29] have measured the accuracy of numerical schemes through the *variance* of the fields in real space, which, by Parseval's theorem, is related to the integral of the structure factor over all wavenumbers. For the Euler scheme (44) for the stochastic heat equation this can be calculated analytically,

$$\sigma_u^2 = \langle u_j^2 \rangle - \langle u_j \rangle^2 = \Delta x^{-1}(1-2\beta)^{-1/2} \approx \Delta x^{-1}(1+\beta),$$

showing first-order temporal accuracy (in the weak sense). For the predictor-corrector scheme (46), on the other hand,

$$(\sigma_u^{PC})^2 \approx \Delta x^{-1}(1-3\beta^2/2).$$

It is important to note, however, that using the variance as a measure of accuracy of stochastic real-space integrators is both too rough and also too stringent of a test. It does not give insights into how well the equipartition is satisfied for the different modes, and, at the same time, it requires that the structure factor be good even for the highest wavenumbers, which is unreasonable to ask from a finite-stencil scheme.

For pseudospectral methods, as studied for the incompressible fluctuating Navier–Stokes equation in [8; 43], one can modify the spectrum of the stochastic forcing so as to balance the numerical stencil artifacts, and one can also use an (exact) exponential temporal integrator in Fourier space to avoid the artifacts of time stepping. However, for finite-volume schemes, a more reasonable approach is to keep the stochastic fluxes uncorrelated between disjoint cells (which is actually physical), and instead of looking at the variance, focus on the accuracy of the static structure factor for small wavenumbers. Specifically, basic schemes will typically have $S_k - 1 = O(\Delta t k^2)$, while multistep schemes will typically achieve $S_k - 1 = O(\Delta t^2 k^2)$ or higher temporal order, or even $S_k - 1 = O(\Delta t^2 k^4)$.

5B. ***Dynamic structure factor.*** It is also constructive to study the full dynamic structure factor for a given numerical scheme, especially for small wavenumbers and low frequencies. This is significantly more involved in terms of analytical calculations and the results are algebraically more complicated, especially for multistage methods and more complex equations. For the Euler scheme (44) the solution to (28) is

$$S_{k,\omega} = \frac{2\chi_1\chi_2^{-1}\mu k^2}{2\Delta t^{-2}(1-\cos\Delta\omega) + \chi_1^2\chi_2^{-1}\mu^2 k^4},$$

where $\chi_1 = 2(1-\cos\Delta k)/\Delta k^2$ and $\chi_2 = 1+2\beta(\cos\Delta k - 1)$. This shows that the dynamic structure factor does not converge to the correct answer for all wavenumbers

even in the limit $\Delta t \to 0$, namely,

$$\lim_{\beta\to 0} S_{k,\omega} = \frac{2\chi_1 \mu k^2}{\omega^2 + \chi_1^2 \mu^2 k^4}. \tag{48}$$

For small Δk, $\chi_1 \approx 1 - \Delta k^2/6$, and the numerical result closely matches the theoretical result (43). However, for finite wavenumbers the effective diffusion coefficient is multiplied by a prefactor χ_1, which represents the spatial truncation error in the second-order approximation to the Laplacian. For all of the time-integration schemes for the stochastic heat equation discussed above, one can reduce the discrete dynamic structure factor to a form

$$S_{k,\omega} = \frac{2\chi_{\text{stoch}} \mu k^2}{2\Delta t^{-2}(1 - \cos\Delta\omega) + \chi_{\text{det}}^2 \mu^2 k^4},$$

where χ_{stoch} and χ_{det} depend on β and Δk and can be used to judge the accuracy of the scheme.

In this paper we focus on the static structure factors in order to optimize the numerical schemes and then simply check numerically that they also produce reasonably accurate results for the dynamic structure factors for small and intermediate wavenumbers and frequencies.

5C. ***Higher-order differencing.*** Another interesting question is whether using a higher-order differencing formula for the viscous fluxes improves upon the second-order formula in the basic Euler scheme (44). For example, a standard fourth order in space finite difference yields the modified Euler scheme

$$\begin{aligned} u_j^{n+1} = u_j^n &+ \frac{\mu\Delta t}{12\Delta x^2}(-u_{j-2}^n + 16u_{j-1}^n - 30u_j^n + 16u_{j+1}^n - u_{j+2}^n) \\ &+ \sqrt{2\mu}\frac{\Delta t^{1/2}}{\Delta x^{3/2}}(W_{j+1/2} - W_{j-1/2}). \end{aligned} \tag{49}$$

Repeating the previous calculation shows that

$$\lim_{\beta\to 0} S_k = 6\,[7 - \cos\Delta k]^{-1}, \tag{50}$$

demonstrating that the fluctuation-dissipation theorem is not satisfied for this scheme at the discrete level even for infinitesimal time steps. This is because the spatial discretization operators in (49) do not satisfy the discrete fluctuation dissipation balance.

In order to obtain higher-order divergence and Laplacian stencils that satisfy (31) we can start from a higher order divergence discretization $\boldsymbol{D}$ and then simply calculate the resulting discrete Laplacian $\boldsymbol{L} = -\boldsymbol{D}\boldsymbol{D}^\star$. Here $\boldsymbol{D}$ should be a fourth-order (or higher) difference formula that combines four face-centered values, two

on each side of a given cell, into an approximation to the derivative at the cell center. Conversely, $\boldsymbol{D}^\star$ combines the values from four cells, two on each side of a given face, into an approximation to the derivative at the face center. A standard fourth-order finite-difference stencil for $\boldsymbol{D}$ produces the *higher-order Euler scheme*

$$u_j^{n+1} = u_j^n + \frac{\mu\Delta t}{\Delta x^2}\Big(\frac{1}{576}u_{j-3}^n - \frac{3}{32}u_{j-2}^n + \frac{87}{64}u_{j-1}^n - \frac{365}{144}u_j^n + \frac{87}{64}u_{j+1}^n - \frac{3}{32}u_{j+2}^n + \frac{1}{576}u_{j+3}^n\Big) + \sqrt{2\mu}\frac{\Delta t^{1/2}}{\Delta x^{3/2}}\Big(\frac{1}{24}W_{j-3/2} - \frac{9}{8}W_{j-1/2} + \frac{9}{8}W_{j+1/2} - \frac{1}{24}W_{j+3/2}\Big), \tag{51}$$

for which $S_k \approx 1 + \beta\Delta k^2/2$, which is the same leading-order error as the basic Euler scheme (44). On the other hand, the dynamic structure factor for small time steps is as in (48) but now

$$\chi_1 = (1-\cos\Delta k)(13-\cos\Delta k)/\left(72\Delta k^2\right) \approx 1 - \tfrac{3}{320}\Delta k^4,$$

which shows the higher spatial order of the scheme.

Note that in (51) both the discretization of the Laplacian and of the gradient are of higher spatial order than in (44), however, the Laplacian operator is not of the highest order possible for the given stencil width. We will not use higher-order differencing for the diffusive fluxes in this work in order to avoid large Laplacian stencils like the one above. Rather, we will use the traditional second-order discretization and focus on the time integration of the resulting system.

5D. ***Handling of advection.*** The analysis we illustrated here for the stochastic heat equation can be directly applied to the scalar advection-diffusion equation (35) in one dimension:

$$\upsilon_t = -a\upsilon_x + \mu\upsilon_{xx} + \sqrt{2\mu}\mathcal{W}_x. \tag{52}$$

For example, a second-order centered difference discretization of the advective term $-a\upsilon_x$ leads to the following explicit Euler scheme

$$u_j^{n+1} = u_j^n - \frac{\alpha}{2}(u_{j+1}^n - u_{j-1}^n) + \beta(u_{j-1}^n - 2u_j^n + u_{j+1}^n) + \sqrt{2\mu}\frac{\Delta t^{1/2}}{\Delta x^{3/2}}(W_{j+1/2}^n - W_{j-1/2}^n), \tag{53}$$

where the dimensionless advective CFL number is

$$\alpha = \frac{a\Delta t}{\Delta x} = \beta r,$$

and $r = a\Delta x/\mu$ is the so-called cell Reynolds number and measures the relative importance of advective and diffusive terms at the grid scale. Note that this scheme is unconditionally unstable when $\mu = 0$, specifically, the stability condition is $\alpha^2/2 \le \beta \le 1/2$.

For the Euler method (53) the analysis yields a structure factor

$$S_k \approx \frac{1}{1-\alpha r/2} + \frac{(1-r^2/4)}{2(1-\alpha r/2)^2}\beta\Delta k^2,$$

showing that even the smallest wavenumbers have the wrong spectrum for a finite time step when $|r| > 0$, which is unacceptable in practice since it means that even the slowly evolving large-scale fluctuations are not handled correctly. Adding an artificial diffusion $\Delta\mu = \mu\,|r|\,/2$ to μ leads to an improved leading order error:

$$S_k \approx 1 + \tfrac{1}{2}(1-r^2/4)\beta\Delta k^2 + O(\Delta t^2 \Delta k^2).$$

It is well known that adding such an artificial diffusion is equivalent to upwinding the advective term and leads to much improved stability for large r as well.[1]

The second-order predictor-corrector time stepping scheme can be applied when advection is included as well. If $|r| > 0$, the leading order errors are

$$\text{PC-1RNG: } S_k \approx 1 - \tfrac{1}{4}\alpha^2\big(1 - \tfrac{1}{2}r\alpha\big)\Delta k^2, \tag{54}$$

$$\text{PC-2RNG: } S_k \approx 1 - \tfrac{1}{8}r\alpha^3\Delta k^2, \tag{55}$$

showing that PC-2RNG gives a more accurate discrete structure factor than PC-1RNG for small wavenumbers and time steps. Note that the predictor-corrector method is unconditionally unstable when $\mu = 0$. In Section 6A we analyze a three-stage Runge–Kutta scheme that has a small leading order error in S_k but is also stable when $\alpha < 1$ even if $\mu = 0$.

6. LLNS equations in one dimension

In this section, we will consider the linearized LLNS system (4) for a monoatomic ideal gas in one spatial dimension, that is, where symmetry dictates variability along only the x axis. As explained in the Introduction, focusing on an ideal gas simply fixes the values of certain coefficients and thus simplifies the algebra, without limiting the generality of our analysis. We will arbitrarily choose the number of degrees of freedom per particle to be $d_f = 1$, even though in most cases of physical interest $d_f = 3$ is appropriate; this merely changes some of the constant coefficients and does not affect our discussion. Explicitly, the one-dimensional linearized LLNS

[1] Note that for this particular type of upwinding the denominator in (37) vanishes identically and it can be shown that the correct solution is $\Delta S_k^{(0)} = 0$; however, this is not necessarily true for other, higher order, upwind discretizations of advection.

equations are

$$\begin{bmatrix}\partial_t \rho\\ \partial_t v\\ \partial_t T\end{bmatrix} = -\frac{\partial}{\partial x}\begin{bmatrix}\rho_0 v + \rho v_0\\ c_0^2\rho_0^{-1}\rho + c_0^2 T_0^{-1} T + v_0 v\\ c_0^2 c_v^{-1} v + T v_0\end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix}0\\ \rho_0^{-1}\eta_0 v_x\\ \rho_0^{-1}c_v^{-1}\mu_0 T_x\end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix}0\\ \rho_0^{-1}\Sigma\\ \rho_0^{-1}c_v^{-1}\Xi\end{bmatrix}, \tag{56}$$

where the covariance matrices of the stochastic fluxes are $C_\Sigma = 2\eta_0 k_B T_0$ and $C_\Xi = 2\mu_0 k_B T_0^2$. In Fourier space the flux becomes

$$\widehat{\boldsymbol{F}} = \begin{bmatrix} v_0 & \rho_0 & 0\\ \rho_0^{-1}c_0^2 & (v_0 - ik\rho_0^{-1}\eta_0) & T_0^{-1}c_0^2\\ 0 & c_0^2 c_v^{-1} & (v_0 - ik\rho_0^{-1}c_v^{-1}\mu_0)\end{bmatrix},$$

which through Equations (13) and (14) (or, equivalently, (30)) gives static structure factors that are independent of k:

$$\boldsymbol{S}(k) = \begin{bmatrix}\rho_0 c_0^{-2} k_B T_0 & 0 & 0\\ 0 & \rho_0^{-1} k_B T_0 & 0\\ 0 & 0 & \rho_0^{-1} c_v^{-1} k_B T_0^2\end{bmatrix}. \tag{57}$$

Therefore, the invariant distribution for the fluctuating fields is spatially-white, with no correlations among the different primitive variables, and with variances given in (57). This is in agreement with predictions of statistical mechanics, and how Landau and Lifshitz obtained the form of the stochastic fluxes. Note that in the incompressible limit, $c_0 \to \infty$, the density fluctuations diminish, but the velocity and temperature fluctuations are independent of c_0.

In this section we will calculate the discrete structure factor for several finite-volume approximations to (56). From the diagonal elements of $\boldsymbol{S}_k$ we can directly obtain the nondimensionalized static structure factors for the three primitive variables, for example,

$$S_k^{(\rho)} = \frac{V}{\rho_0 c_0^{-2} k_B T_0}\left\langle \hat{\rho}_k \hat{\rho}_k^\star \right\rangle,$$

which for a perfect scheme would be unity for all wavevectors. Similarly, the off-diagonal or cross elements, such as, for example,

$$S_k^{(\rho,v)} = \frac{V}{\sqrt{(\rho_0 c_0^{-2} k_B T_0)(\rho_0^{-1} k_B T_0)}}\left\langle \hat{\rho}_k \hat{v}_k^\star \right\rangle,$$

would all vanish for all wavevectors for a perfect scheme. Our goal will be to quantify the deviations from "perfect" for several methods, as a function of the discretization parameters Δx and Δt.

6A. ***Third-order Runge–Kutta (RK3) scheme.*** When designing numerical schemes to integrate the full LLNS system, it seems most appropriate to base the scheme on well known robust deterministic methods, and modify the deterministic methods by simply adding a stochastic component to the fluxes, in addition to the usual deterministic component. With such an approach, at least we can be confident that in the case of weak noise the solver will be robust and thus we will not compromise the fluid solver just to accommodate the fluctuations.

A well known approach to solving PDEs in conservation form

$$\partial_t \boldsymbol{\mathcal{U}} = -\boldsymbol{\nabla} \cdot [\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}})] = -\boldsymbol{\nabla} \cdot [\boldsymbol{\mathcal{F}}_H(\boldsymbol{\mathcal{U}}) + \boldsymbol{\mathcal{F}}_D(\boldsymbol{\nabla}\boldsymbol{\mathcal{U}})]$$

is to use the *method of lines* to decouple the spatial and temporal discretizations. We will focus on one dimension first for notational simplicity. In the method of lines, a finite-volume spatial discretization is applied to the obtain a system of differential equations for the discretized fields

$$\begin{aligned} \frac{d\boldsymbol{U}_j}{dt} &= -\Delta x^{-1}[\boldsymbol{F}_{j+1/2}(\boldsymbol{U}) - \boldsymbol{F}_{j-1/2}(\boldsymbol{U})] \\ &= -\Delta x^{-1}[\boldsymbol{F}_H(\boldsymbol{U}_{j+1/2}) - \boldsymbol{F}_H(\boldsymbol{U}_{j-1/2})] \\ &\quad -\Delta x^{-1}[\boldsymbol{F}_D(\boldsymbol{\nabla}_{j+1/2}\boldsymbol{U}) - \boldsymbol{F}_D(\boldsymbol{\nabla}_{j-1/2}\boldsymbol{U})], \end{aligned} \tag{58}$$

where $\boldsymbol{U}_{j+1/2}$ are face-centered values of the fields that are calculated from the cell-centered values $\boldsymbol{U}_j$, and $\boldsymbol{\nabla}_{j+1/2}$ is a cell-to-face discretization of the gradient operator. Any classical temporal integrator can be applied to the resulting system of semidiscrete system. It is well known that the Euler and Heun (two-step second-order Runge–Kutta) methods are unconditionally unstable for hyperbolic equations. In [13], an algorithm for the solution of the LLNS system of equations (1) was proposed, which is based on the three-stage, low-storage TVD Runge–Kutta (RK3) scheme of Gottlieb and Shu [37]. The RK3 scheme is the simplest TVD RK discretization for the deterministic compressible Navier–Stokes equations that is stable even in the inviscid limit, with the omission of slope-limiting. Here we adopt the same basic scheme and investigate optimal ways of evaluating the stochastic flux.

In the RK3 scheme, the hyperbolic component of the face flux $\boldsymbol{F}_H$ is calculated by a cubic interpolation of $\boldsymbol{U}$ from the cell centers to the faces using an interpolation formula borrowed from PPM (piecewise parabolic method), [18],

$$\boldsymbol{U}_{j+1/2} = \tfrac{7}{12}(\boldsymbol{U}_j + \boldsymbol{U}_{j+1}) - \tfrac{1}{12}(\boldsymbol{U}_{j-1} + \boldsymbol{U}_{j+2}), \tag{59}$$

and then directly evaluating the hyperbolic flux from the interpolated values. In [13; 10] a modified interpolation is proposed that preserves variances; however, our analytical calculations indicate that this type of interpolation artificially increases the structure factor for intermediate wavenumbers in order to compensate for the errors at larger wavenumbers. Note that for the full nonlinear equations, either the conserved or the primitive quantities can be interpolated. For the linearized equations it does not matter and it is simpler to work exclusively with primitive variables.

In the RK3 method, the diffusive components of the fluxes $\boldsymbol{F}_D$ are calculated using classical face-centered second-order centered stencils to evaluate the gradients of the fields at the cell faces. Stochastic fluxes $\boldsymbol{Z}_{j+1/2}$ are also generated at the faces of the grid using a standard random number generator (RNG). These stochastic fluxes are generated independently for velocity and temperature, and are zero for density,

$$\boldsymbol{Z}^{(RNG)}_{j+1/2} = \begin{bmatrix} 0 \\ \rho_0^{-1}\,(2\eta_0 k_B T_0)^{1/2}\, W^{(1)}_{j+1/2} \\ \rho_0^{-1} c_v^{-1} (2\mu_0 k_B T_0^2)^{1/2} W^{(2)}_{j+1/2} \end{bmatrix},$$

where $W^{(1/2)}_{j+1/2}$ denotes a normal variate with zero mean and unit variance.

For each stage of the RK3 scheme, a total cell increment is calculated as

$$\Delta \boldsymbol{U}_j(\boldsymbol{U}, \boldsymbol{W}) = -\frac{\Delta t}{\Delta x}[\boldsymbol{F}_{j+1/2}(\boldsymbol{U}) - \boldsymbol{F}_{j-1/2}(\boldsymbol{U})] + \frac{\Delta t^{1/2}}{\Delta x^{3/2}}(\boldsymbol{Z}_{j+1/2} - \boldsymbol{Z}_{j-1/2}).$$

Each time step of the RK3 algorithm is composed of three stages

$$\begin{aligned} \boldsymbol{U}_j^{n+1/3} &= \boldsymbol{U}_j^n + \Delta \boldsymbol{U}_j(\boldsymbol{U}^n, \boldsymbol{W}_1) && \text{(estimate at } t = (n+1)\Delta t), \\ \boldsymbol{U}_j^{n+2/3} &= \tfrac{3}{4}\boldsymbol{U}_j^n + \tfrac{1}{4}[\boldsymbol{U}_j^{n+1/3} + \Delta \boldsymbol{U}_j(\boldsymbol{U}_j^{n+1/3}, \boldsymbol{W}_2)] && \text{(estimate at } t = (n+\tfrac{1}{2})\Delta t), \\ \boldsymbol{U}_j^{n+1} &= \tfrac{1}{3}\boldsymbol{U}_j^n + \tfrac{2}{3}[\boldsymbol{U}_j^{n+2/3} + \Delta \boldsymbol{U}_j(\boldsymbol{U}^{n+2/3}, \boldsymbol{W}_3)], \end{aligned} \tag{60}$$

where for now we have not assumed anything about how the stochastic fluxes between different stages, $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ and $\boldsymbol{W}_3$, are related to each other. The relevant dimensionless parameters that measure the ratio of the time step to the CFL stability limits are

$$\alpha = \frac{c_0 \Delta t}{\Delta x}, \quad \beta = \frac{\eta_0 \Delta t}{\rho_0 \Delta x^2} = \frac{\alpha}{r}, \quad \beta_T = \frac{\mu_0 \Delta t}{\rho_0 c_v \Delta x^2} = \frac{1}{\mathrm{Pr}}\frac{\alpha}{r} = \frac{\alpha}{p},$$

where $r = c_0 \rho_0 \Delta x / \eta_0$ is the cell Reynolds number (we have assumed a low Mach number flow, that is, $|v_0| \ll c_0$), and $\mathrm{Pr} = \eta_0 c_v / \mu_0$ is the Prandtl number of the fluid. For low-density gases, r and $p = r\mathrm{Pr}$ can be close to or smaller than one; however, for dense fluids sound dominates and $r > 1$ and $p > 1$ for all reasonable

Δx (essentially, $\Delta x > \lambda$, where λ is the mean free path). In practice, in order to fully resolve viscous scales, one should keep both r and p reasonably small.

6B. ***Evaluation of the stochastic fluxes.*** In the original RK3 algorithm [13], a different stochastic flux is generated in each stage, that is, $\boldsymbol{W}_s = \sqrt{2}\boldsymbol{W}_{RNG}^{(s)}$, $s = 1, 2, 3$. The additional prefactor $\sqrt{2}$ is added because the averaging between the three stages reduces the variance of the overall stochastic flux. One can also use different weights for each of the three stochastic fluxes, that is, $\boldsymbol{W}_s = w_s \boldsymbol{W}_{RNG}^{(s)}$. Another option is to simply use the same stochastic flux $\boldsymbol{W}_{RNG}^{(0)}$ in all three stages, that is, $\boldsymbol{W}_s = \boldsymbol{W}_{RNG}^{(0)}$. A further option is to use the same random flux $\boldsymbol{W}_{RNG}^{(0)}$ in all three stages, but put in different weights in each stage, that is, $\boldsymbol{W}_s = w_s \boldsymbol{W}_{RNG}^{(0)}$. Our goal is to find out which approach is optimal. For this purpose, we can generally assume that the three random fluxes are different, to obtain a total of six random numbers per cell per step, and use the formalism developed in Section 3 with $N_s = 6$ to express the structure factor in terms of the 6×6 covariance matrix of the random variates. This calculation is too tedious even for a computer algebra system, and we therefore first study the simple advection-diffusion Equation (35) in order to gain some insight.

6B1. *Advection-diffusion equation.* The RK3 method can be directly applied to the scalar advection-diffusion equation in one dimension (52). Experience with deterministic solvers suggests that a numerical scheme that performs well on this type of model equation is likely to perform well on the full system (1) when viscous effects are fully resolved. Here we use PPM-interpolation based discretization of the hyperbolic flux given in (59), which leads to a standard fourth-order centered difference approximation to the first derivative v_x [9], and thus justifies our choice for the interpolation. We discretize the gradient used in calculating the diffusive fluxes using the second-order centered difference

$$\nabla_{j+1/2} u = \frac{u_{j+1} - u_j}{\Delta x},$$

which leads to the standard second-order centered difference approximation to the second derivative v_{xx} (the challenges with using the standard fourth-order centered difference approximation to v_{xx} [9] are discussed in Section 5C). The stencil widths in (23) are $w_D = 6$ (three stages with stencil width two each) and $w_S = 4$, and there are $N_s = 3$ random numbers per cell per step (one per stage), with a general 3×3 covariance matrix $\boldsymbol{C_W}$. Equation (27) can then be solved to obtain the static structure factor for any wavenumber, however, these expressions are too complex to be useful for analysis. Instead, we perform an expansion of both sides of (27) for small k and thus focus on the behavior of the static structure factors for small wavenumbers and small time steps.

As a first condition on $\boldsymbol{C_W}$, we have the weak consistency requirement $S_{k=0} = 1$. With this condition satisfied, the method satisfies the discrete fluctuation-dissipation

balance in the limit $\Delta t \to 0$ since the discretization of the divergence is the negative adjoint of the discretization of the gradient. A second condition is obtained by equating the coefficient in front of the leading-order error term in S_k, of order $\alpha \Delta k^2$, to zero; where the advective dimensionless CFL number is $\alpha = a\Delta t/\Delta x$. It turns out that this also makes the term of order $\alpha \Delta k^4$ vanish. A third condition is obtained by equating the coefficient in front of the next-order error term of order $\alpha^2 \Delta k^2$ to zero. Finally, a fourth condition equates the coefficient in front of $\alpha^2 \Delta k^4$ to zero. For this three-stage method, it is not possible to make the terms with higher powers of α vanish identically for any choice of $\boldsymbol{C_W}$. No additional conditions are obtained by looking at terms with powers of the diffusive CFL number $\beta = \mu \Delta t/\Delta x^2$ since, as it turns out, the accuracy is always limited by the hyperbolic fluxes.

The various ways of generating the stochastic fluxes can now be compared by investigating how many of these conditions are satisfied. It turns out that only the first condition is satisfied if we use a different independently generated stochastic flux in each stage (one can satisfy one more condition by using different weights for the three independent stochastic fluxes). The second condition is satisfied if we use the same stochastic flux in all stages with a unit weight, that is, $\boldsymbol{W}_s = w_s \boldsymbol{W}^{(0)}_{RNG}$ with $w_1 = w_2 = w_3 = 1$. Armed with the freedom to put a different weight for this flux in each of the stages, we can satisfy the third condition as well if we use

$$w_1 = \tfrac{3}{4}, \quad w_2 = \tfrac{3}{2}, \quad w_3 = \tfrac{15}{16}, \tag{61}$$

which gives a structure factor

$$S_k = 1 - \frac{r}{24}\alpha^3 \Delta k^2 - \frac{1}{6r^2}\alpha^2 \Delta k^4 + \text{h.o.t.}$$

If we are willing to increase the cost of each step and generate two random numbers per cell per step, we can satisfy the fourth condition as well. For this purpose, we look for a covariance matrix $\boldsymbol{C_W}$ that satisfies the four conditions and is also positive semidefinite and has a rank of two, that is, has a smallest eigenvalue of zero. A solution to these equations gives the following method for evaluating the stochastic fluxes in the three stages

$$\boldsymbol{W}_1 = \boldsymbol{W}^{(A)}_{RNG} - \sqrt{3}\boldsymbol{W}^{(B)}_{RNG}, \quad \boldsymbol{W}_2 = \boldsymbol{W}^{(A)}_{RNG} + \sqrt{3}\boldsymbol{W}^{(B)}_{RNG}, \quad \boldsymbol{W}_3 = \boldsymbol{W}^{(A)}_{RNG}, \tag{62}$$

where $\boldsymbol{W}^{(A)}_{RNG}$ and $\boldsymbol{W}^{(B)}_{RNG}$ are two independent random vectors that need to be generated and stored during each RK3 step. This approach produces a structure factor

$$S_k = 1 - \frac{r}{24}\alpha^3 \Delta k^2 - \frac{24 + r^2}{288r}\alpha^3 \Delta k^4 + \text{h.o.t.}$$

We will refer to the RK3 scheme that uses one random flux per step and the weights in (61) as the *RK3-1RNG scheme*, and to the RK3 scheme with two random fluxes per step as given in (62) as the *RK3-2RNG scheme.*

It is important to point out that for the MacCormack method, which is equivalent to the Lax–Wendroff method for the advection-diffusion equation, the leading-order errors are of order $\alpha\Delta k^2$. This is much worse than for the stochastic heat equation (see Section 5A) even though the MacCormack scheme is a predictor-corrector method. This is because of the low-order handling of advective fluxes used in the MacCormack method to stabilize the two-stage Runge–Kutta time integrator.

6C. ***Results for LLNS equations in one dimension.*** We can now theoretically study the behavior of the RK3-1RNG and RK3-2RNG schemes on the full linearized system (56), specializing to the case of zero background flow, $v_0 = 0$. As expected, we find that the behavior is very similar to the one observed for the advection-diffusion equation; in particular, the leading order terms have the same basic form. Specifically, the expansions of the diagonal and off-diagonal components of the structure factor $\boldsymbol{S}_k$ for the RK3-1RNG method are

$$\begin{aligned} S_k^{(\rho)} &\approx S_k^{(T)} \approx 1 + \frac{S_k^{(u)} - 1}{3} \approx 1 + \varepsilon(\alpha)\Delta k^2, \\ S_k^{(\rho,u)} &\approx \frac{i}{12r}\alpha^2 \Delta k^3, \\ S_k^{(\rho,T)} &\approx 2\varepsilon(\alpha)\Delta k^2, \\ S_k^{(u,T)} &\approx i\frac{r-p}{6pr}\alpha^2 \Delta k^3, \end{aligned} \tag{63}$$

where

$$\varepsilon(\alpha) = -\frac{3\alpha^3 pr}{4(3p+2r)}.$$

These structure factors are shown in Figure 2 for sample discretization parameters, along with the corresponding results for RK3-2RNG. We see from these expressions that as the speed of sound dominates the stability restrictions on the time step more and more, namely, as p or r become larger and larger, a smaller α is required to reach the same level of accuracy, that is, a smaller time step relative to the acoustic CFL stability limit is required.

Similar results to Equation (63) hold also for the isothermal LLNS equations (in which the there is no energy equation), for which the calculations are simpler. For linearization around a constant background flow of speed $v_0 = c_0\mathrm{Ma}$, where Ma is the reference Mach number, the analysis for the isothermal LLNS equations shows that the error grows with the Mach number as

$$S_k^{(\rho)} \approx 1 + \varepsilon(\alpha)[1 + 6\mathrm{Ma}^2 + \mathrm{Ma}^4]\Delta k^2.$$

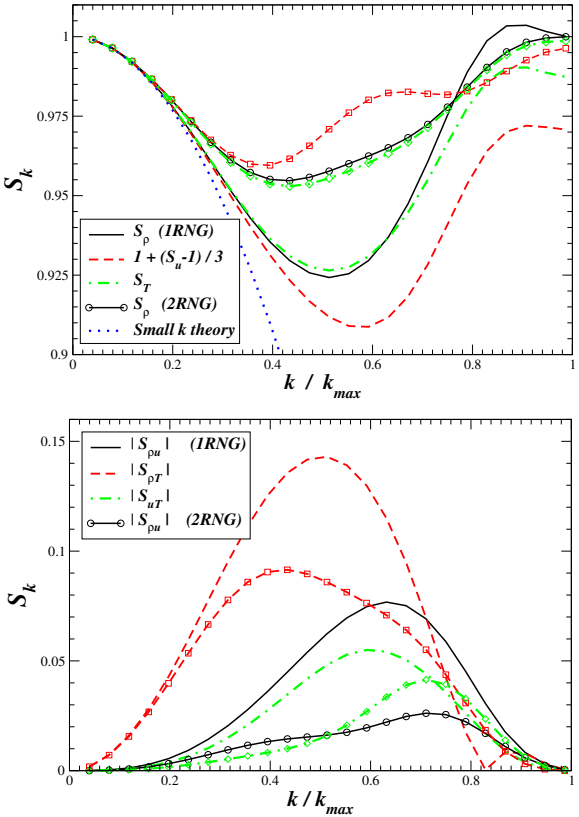


Figure 2. Discrete structure factor $\boldsymbol{S}_k$ for the LLNS equation under the RK3-1RNG (lines) and RK3-2RNG (same style of lines with added symbols) schemes, as calculated by numerical solution of (27) for an ideal one-dimensional gas, for $\alpha = 0.5$, $\beta = 0.2$ and $\beta_T = 0.1$. Left: diagonal (self) structure factors, which should ideally be identically unity. Also shown is the leading order error term $1+\varepsilon(\alpha)\Delta k^2$ (dotted line), which is the same for both schemes. Right: off-diagonal (cross) structure factors, which should ideally be identically zero.

7. Higher dimensions

Much of what we already described for one dimension applies directly to higher dimensions [13; 10]. However, there is a peculiarity with the LLNS equations in three dimensions that does not appear in one dimension, and also does not appear for the scalar diffusion equation [6]. In one dimension the velocity component of the LLNS system of equations is essentially an advection-diffusion equation. In higher dimensions, however, there is an important difference: namely, the dissipation

operator is a *modified* Laplacian $\mathcal{L}_m$. By neglecting the hyperbolic coupling between velocity and the other variables in the linearized LLNS equations, we obtain the *stochastic diffusion equation*

$$\begin{aligned}\vartheta_t &= \eta \nabla \cdot [\boldsymbol{C}(\nabla \vartheta)] + \sqrt{2\eta}\nabla \cdot [\boldsymbol{C}^{1/2}\boldsymbol{\mathcal{W}}] \\ &= \eta\,(\boldsymbol{\mathcal{D}}\boldsymbol{C}\boldsymbol{\mathcal{G}})\,\vartheta + \sqrt{2\eta}\boldsymbol{\mathcal{D}}\boldsymbol{C}^{1/2}\boldsymbol{\mathcal{W}} = \eta\mathcal{L}_m\vartheta + \sqrt{2\eta}\boldsymbol{\mathcal{W}}_m,\end{aligned} \tag{64}$$

where $\boldsymbol{C}$ is the linear operator that transforms the velocity gradient into a traceless symmetric stress tensor

$$\boldsymbol{C}(\nabla\vartheta) = 2\big[\tfrac{1}{2}(\nabla\vartheta + \nabla\vartheta^T) - \tfrac{1}{3}\boldsymbol{I}\,(\nabla\cdot\vartheta)\big], \tag{65}$$

and we have denoted the continuum velocity field by $\vartheta \equiv \boldsymbol{\mathcal{U}}$ in order to distinguish from the discretized velocities $\boldsymbol{v} \equiv \boldsymbol{U}$. Here we will focus on two-dimensional flows, $\vartheta = [\vartheta_x, \vartheta_y]$, however, identical considerations apply to the fully three-dimensional case.

If we arrange the components of the velocity gradient as a vector with four components, $\nabla\vartheta = [\partial_x\vartheta_x, \partial_x\vartheta_y, \partial_y\vartheta_x, \partial_y\vartheta_y]^T$, the linear operator $\boldsymbol{C}$ in (65) becomes the matrix

$$\boldsymbol{C} = \begin{bmatrix} \frac{4}{3} & 0 & 0 & -\frac{2}{3} \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ -\frac{2}{3} & 0 & 0 & \frac{4}{3} \end{bmatrix}, \tag{66}$$

which is not diagonal. This means that the components of the stochastic stress $\boldsymbol{C}^{1/2}\boldsymbol{W}$ would need to have nontrivial correlations between the x fluxes for v_x and y fluxes for v_y, as well as between the x fluxes for v_y and y fluxes for v_x. These correlations essentially amount to the requirement that the stochastic stress be a traceless symmetric tensor, at least at the level of its covariance matrix. Numerically, one generates independent random variates for the upper triangular portion of the stochastic stress tensor for each cell, then makes the tensor traceless and symmetric [28]. Note that one can save one random number by using only $d-1$ variates to generate the diagonal elements.

However, it is important to point out that an *equivalent* formulation is obtained by using the operator

$$\boldsymbol{C} = \begin{bmatrix} \frac{4}{3} & 0 & 0 & \frac{1}{3} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{4}{3} \end{bmatrix} = \boldsymbol{I} + \begin{bmatrix} \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} \end{bmatrix}, \tag{67}$$

where there is nontrivial cross correlations only between the x fluxes for v_x and y fluxes for v_y. The splitting of the operator $\boldsymbol{C}$ in (67) corresponds to rewriting the

stochastic diffusion Equation (64) in the equivalent but suggestive form

$$\begin{aligned}\vartheta_t &= -\eta\left[\nabla^2\vartheta + \tfrac{1}{3}\nabla\left(\nabla\cdot\vartheta\right)\right] + \sqrt{2\eta}\left[\left(\nabla\cdot\boldsymbol{\mathcal{W}}_T\right) + \sqrt{\tfrac{1}{3}}\nabla\mathcal{W}_V\right] \\ &-= \eta\left(\boldsymbol{\mathcal{D}}_T\boldsymbol{\mathcal{G}}_T + \tfrac{1}{3}\boldsymbol{\mathcal{G}}_V\boldsymbol{\mathcal{D}}_V\right)\vartheta + \sqrt{2\eta}\left(\boldsymbol{\mathcal{D}}_T\boldsymbol{\mathcal{W}}_T + \sqrt{\tfrac{1}{3}}\boldsymbol{\mathcal{G}}_V\mathcal{W}_V\right),\end{aligned} \tag{68}$$

where we have now distinguished between the *tensorial* divergence $\boldsymbol{\mathcal{D}}_T$ and gradient operators $\boldsymbol{\mathcal{G}}_T = -\boldsymbol{\mathcal{D}}_T^\star$, which map from tensor to vector fields and vector to tensor fields, respectively, and the *vectorial* divergence $\boldsymbol{\mathcal{D}}_V$ and gradient operators $\boldsymbol{\mathcal{G}}_V = -\boldsymbol{\mathcal{D}}_V^\star$, which map from vector to scalar fields and scalar to vector fields, respectively. Corresponding to the splitting of the modified Laplacian $\boldsymbol{\mathcal{L}}_m = \boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{L}}_T + \boldsymbol{\mathcal{L}}_V$ into the tensorial Laplacian operator $\boldsymbol{\mathcal{L}}_T = \boldsymbol{\mathcal{D}}_T\boldsymbol{\mathcal{G}}_T$ and the vectorial component $\boldsymbol{\mathcal{L}}_V = \boldsymbol{\mathcal{G}}_V\boldsymbol{\mathcal{D}}_V/3$, in (68) we have split the stochastic stress into a tensor white-noise field $\boldsymbol{\mathcal{W}}_T$ in which all components are uncorrelated, and a scalar white-noise field $\mathcal{W}_V$, which we will call the stochastic *divergence stress*. This representation is perhaps more physically intuitive than the standard formulation in which the stochastic stress has unexpected exact symmetry and is exactly traceless. Note that in the more general case where the diffusion coefficient is spatially dependent and there is nonzero bulk viscosity η_B, the dissipative term in (68) becomes

$$\nabla\cdot[\eta(\nabla\vartheta)] + \nabla[(\eta/3+\eta_B)\nabla\cdot\vartheta],$$

with an equivalent change in the stochastic term. Also note that for the fluctuating incompressible Navier–Stokes equation the term with the velocity divergence disappears and the dissipation operator is a projected traditional Laplacian [8; 5], while the stochastic flux is simply a projected tensor white-noise field.

7A. ***Discrete fluctuation dissipation balance.*** Our ultimate goal is to find a scheme that satisfies the discrete fluctuation dissipation theorem, that is, find a discrete modified Laplacian $\boldsymbol{L}_m$ that is a consistent approximation to the continuum modified Laplacian $\boldsymbol{\mathcal{L}}_m(\boldsymbol{k})\widehat{\vartheta} = \boldsymbol{k}\cdot[\boldsymbol{C}(\boldsymbol{k}\widehat{\vartheta}^T)]$ for small k, and a way to efficiently generate random increments $\boldsymbol{W}_m$ that discretize $\boldsymbol{\mathcal{W}}_m$ and whose covariance is $\langle \boldsymbol{W}_m\boldsymbol{W}_m^\star\rangle = \boldsymbol{L}_m$. This task is nontrivial in general, and completing it requires some ingenuity and insight, as illustrated in the work of Atzberger [6] on multigrid methods for the scalar stochastic diffusion equation. We illustrate two different approaches next, the first corresponding to attempting to directly discretize the modified Laplacian $\boldsymbol{\mathcal{L}}_m$, and the second corresponding to discretizing the split Laplacian $\boldsymbol{\mathcal{L}}_T + \boldsymbol{\mathcal{L}}_V/3$. In the continuum context these are, of course, equivalent, but this is not the case in the discrete context. Namely, in the continuum formulation, $\boldsymbol{\mathcal{C}}$ maps from gradients to stresses, the divergence operator $\boldsymbol{\mathcal{D}}$ maps from fluxes to fields, and the gradient $\boldsymbol{\mathcal{G}}$ maps from fields to gradients. In the continuum context, stresses, gradients and fluxes are all tensor fields and thus in the same Hilbert space. In the discrete

context, however, stresses, gradients and fluxes may be discretized differently and thus belong to different spaces.

7A1. *The modified Laplacian approach.* One approach to the problem of constructing discrete operators that satisfy the discrete fluctuation-dissipation balance is to find a discretization of the divergence $\boldsymbol{D}$ and gradient $\boldsymbol{G}$ operators that are skew-adjoint and then form the modified Laplacian $\boldsymbol{L}_m = \boldsymbol{DCG} = -\boldsymbol{DCD}^\star$, and generate the stochastic increments as $\boldsymbol{W}_m = \boldsymbol{DC}^{1/2}\boldsymbol{W}$. As discussed above, for the meaning of $\boldsymbol{C}^{1/2}$ to be clear, stresses and gradients must belong to the same space. Furthermore, it is required that the discrete operators $\boldsymbol{D}$ and $\boldsymbol{G}$ be skew adjoint so that the discrete fluctuation dissipation balance condition (31) is satisfied.

The issue of how to define skew adjoint $\boldsymbol{D}$ and $\boldsymbol{G}$ operators also arose in the historical development of projection algorithms for incompressible flow. The incompressible flow literature suggests two approaches that discretize both gradients and stresses by representing them with tensors at the same grid of points. The first approach corresponds to fully cell-centered discretization originally proposed by Chorin [17], which uses centered differences to define a skew-adjoint gradient and divergence operators. The second approach corresponds to a finite element-based discretization developed by Fortin [31] and later used in the projection algorithm of Bell et al. [11].

In the Fortin approach both stresses and gradients are represented as $d \times d$ tensors at the corners of a regular grid, where d is the spatial dimension. The divergence operator $\boldsymbol{D}$ combines the values of the stresses at the $2d$ corners of a cell to produce a value at the center of the cell. The gradient $\boldsymbol{G} = -\boldsymbol{D}^\star$ combines the values of the fields at the centers of the $2d$ cells that share a corner into a gradient at that corner. In this scheme, the stochastic stresses also live at the corners of the grid. They are generated to have the required covariance, for example, (66). Unfortunately, the discrete Fortin Laplacian $\boldsymbol{L} = \boldsymbol{DG}$ suffers from a serious drawback: it has a nontrivial null space. For example, for the scalar heat equation on a uniform grid in two dimensions, the Laplacian stencil obtained from the Fortin discretization is

$$(L^{(F)}u)_{i,j} = \Delta x^{-2}\left[\tfrac{1}{2}(u_{i+1,j+1} + u_{i-1,j+1} + u_{i-1,j-1} + u_{i+1,j-1}) - 2u_{i,j}\right],$$

for which the odd ($i+j$ odd) and even ($i+j$ even) points on the grid are completely decoupled. In Fourier space the above Laplacian is $-2[1-\cos(\Delta k_x)\cos(\Delta k_y)]$ and thus vanishes for the largest wavevectors, $|\Delta k_x| = \pi$, $|\Delta k_y| = \pi$, which correspond to checker board zero eigenmodes.

It can easily be verified that the same type of checker board zero eigenmodes also exist for the modified Fortin Laplacian $\boldsymbol{L}_m = \boldsymbol{DCG}$. In three dimensions, there are $O(N)$ zero eigenmodes for a grid of size N^3. Issues arising when using these types of stencils in the deterministic context are discussed in Almgren et al. [3]. Our theory for the structure factor implicitly relies on the definiteness of the

discrete generator, and in fact, in the general nonlinear setting the zero modes lead to instabilities of the solution of the full LLNS system of equations. We therefore abandon the Fortin corner-centered discretization of the fluxes.

Fully cell-centered approximations to $\boldsymbol{D}$ and $\boldsymbol{G}$ based on second-order centered differences, previously studied in the context of projection methods for incompressible flows by Chorin [17], lead to a discrete Laplacian that also has a nontrivial null space and suffers similar shortcomings as the Fortin Laplacian. Specifically, even in one dimension one obtains a Laplacian stencil

$$(L^{(C)}u)_i = \frac{1}{4\Delta x^2}[u_{i-2} - 2u_i + u_{i+2}],$$

where the odd-even decoupling is evident. Here we develop a cell-centered (collocated) discretization that preserves the null space of the continuum Laplacian.

7A2. *The split Laplacian approach.* An alternative to trying to form a discrete modified Laplacian $\boldsymbol{L}_m = \boldsymbol{L}_T + \boldsymbol{L}_V$ directly is to use the splitting in (68) and form the discrete tensorial $\boldsymbol{L}_T = \boldsymbol{D}_T\boldsymbol{G}_T$ and vectorial $\boldsymbol{L}_V = \boldsymbol{G}_V\boldsymbol{D}_V/3$ components separately from discretizations of the tensorial and vectorial divergence and gradient operators that are skew-adjoint, $\boldsymbol{G}_T = -\boldsymbol{D}_T^\star$ and $\boldsymbol{G}_V = -\boldsymbol{D}_V^\star$. The stochastic increments would simply be generated as $\boldsymbol{D}_T\boldsymbol{W}_T + \boldsymbol{G}_V W_V/\sqrt{3}$, where W_V and the components of $\boldsymbol{W}_T$ are independent normal variates.

A popular approach to discretizing the tensorial divergence and gradient operators, commonly referred to as a MAC discretization in projection algorithms for incompressible flow [38], defines a divergence at cells centers from normal fluxes on edges, with a corresponding gradient that gives normal derivatives at cell edges from cell-centered values:

$$\begin{aligned}(\boldsymbol{D}\boldsymbol{Z})_{i,j} &= \Delta x^{-1}(\boldsymbol{Z}^{(x)}_{i+1/2,j} - \boldsymbol{Z}^{(x)}_{i-1/2,j}) + \Delta y^{-1}(\boldsymbol{Z}^{(y)}_{i,j+1/2} - \boldsymbol{Z}^{(y)}_{i,j-1/2}) \to \nabla\cdot\boldsymbol{Z},\\ -(\boldsymbol{D}^\star\boldsymbol{v})_{i+1/2,j} &= \Delta x^{-1}(\boldsymbol{v}_{i+1,j} - \boldsymbol{v}_{i,j}) \to \partial\boldsymbol{v}/\partial x,\\ -(\boldsymbol{D}^\star\boldsymbol{v})_{i,j+1/2} &= \Delta y^{-1}(\boldsymbol{v}_{i,j+1} - \boldsymbol{v}_{i,j}) \to \partial\boldsymbol{v}/\partial y.\end{aligned} \tag{69}$$

In this discretization, the tensor field

$$\boldsymbol{Z} = [\boldsymbol{Z}^{(x)}; \boldsymbol{Z}^{(y)}] = [Z^{(x)}_{v_x}, Z^{(x)}_{v_y}; Z^{(y)}_{v_x}, Z^{(y)}_{v_y}]$$

is strictly divided into an x vector $\boldsymbol{Z}^{(x)}$, which is represented on the x faces of the grid, and a y vector $\boldsymbol{Z}^{(y)}$, represented on the y faces of the grid. The MAC discretization, which we used in the earlier one-dimensional examples, leads to a standard 5 point discrete Laplacian in two dimensions (3 point in one dimension, 7 point in three dimensions),

$$(L^{(\mathrm{MAC})}u)_{i,j} = [\Delta x^{-2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + \Delta y^{-2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1})].$$

In Fourier space the MAC Laplacian is $2\cos(\Delta k_x) + 2\cos(\Delta k_y) - 4$ and is strictly negative for all nonzero wavevectors, and thus does not suffer from the instabilities of the Chorin and Fortin discrete Laplacians, discussed in Section 7A1.

The vectorial divergence and gradient operators cannot be discretized using the MAC framework. Namely, $\boldsymbol{D}_V$ must operate on a cell-centered vector field $\boldsymbol{v}$, whereas the MAC-type discretization operates on face-centered values. Instead, for the vectorial component we can use either the Chorin discretization [17], in which both scalar and vector fields are cell-centered, or the Fortin discretization [31], in which scalar fields are represented at corners and vector fields are cell-centered. Here we choose the Fortin discretization and calculate a (scalar-valued) velocity divergence and the corresponding divergence stress at the corners of the grid, and also generate a (scalar) random divergence stress at each corner. The deterministic and random components are added to form the total corner-centered divergence stress, and the velocity increment is calculated from the (vector-valued) cell-centered gradient of the divergence stresses. Note that the nontrivial nullspace of $\boldsymbol{L}_V$ does not pose a problem since $\boldsymbol{L}_T$ and thus also $\boldsymbol{L}_m = \boldsymbol{L}_T + \boldsymbol{L}_V$ has a trivial nullspace.

The discrete modified Laplacian that is obtained by this mixed MAC/Fortin discretization can be represented in terms of second-order centered-difference stencils. The first (i.e., the v_x) component of this Laplacian can be represented as a linear combination of the velocities in the 9 neighboring cells:

$$(\boldsymbol{L}_m \boldsymbol{v})_{jk}^{(v_x)} = \sum_{l,m=-1}^{1} \Big(\frac{1}{\Delta x^2} L^{(\mathrm{MAC},x)}_{2-m,2+l} v^{(x)}_{j+l,k+m} + \frac{1}{\Delta y^2} L^{(\mathrm{MAC},y)}_{2-m,2+l} v^{(x)}_{j+l,k+m} + \frac{1}{3\Delta x^2} L^{(F,x)}_{2-m,2+l} v^{(x)}_{j+l,k+m} + \frac{1}{3\Delta x \Delta y} L^{(F,xy)}_{2-m,2+l} v^{(y)}_{j+l,k+m} \Big), \quad (70)$$

where $\boldsymbol{L}^{(\mathrm{MAC},x/y)}$ and $\boldsymbol{L}^{(F,x/y)}$ correspond to a second-order MAC and Fortin discretizations of the terms $\partial_{xx}\vartheta_x$ and $\partial_{yy}\vartheta_y$ respectively, and $\boldsymbol{L}^{(F,xy)}$ discretizes $\partial_{xy}\vartheta_y$. The same stencils apply to the second (i.e., the v_y) component of the Laplacian as well, by symmetry:

$$(\boldsymbol{L}_m \boldsymbol{v})_{jk}^{(v_y)} = \sum_{l,m=-1}^{1} \Big(\frac{1}{\Delta x^2} L^{(\mathrm{MAC},x)}_{2-m,2+l} v^{(y)}_{j+l,k+m} + \frac{1}{\Delta y^2} L^{(\mathrm{MAC},y)}_{2-m,2+l} v^{(y)}_{j+l,k+m} + \frac{1}{3\Delta y^2} L^{(F,y)}_{2-m,2+l} v^{(y)}_{j+m,k+l} + \frac{1}{3\Delta x \Delta y} L^{(F,xy)}_{2-m,2+l} v^{(x)}_{j+m,k+l} \Big). \quad (71)$$

Note that we chose the peculiar indexing of the stencils so that when printed on paper they correspond to the usual Cartesian representation of the x-y grid. The

coefficients of the MAC stencil (70) are

$$\boldsymbol{L}^{(\mathrm{MAC},x)} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{L}^{(\mathrm{MAC},y)} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \tag{72}$$

while the Fortin stencils are

$$\boldsymbol{L}^{(F,x)} = \begin{bmatrix} \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & -1 & \frac{1}{2} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \end{bmatrix}, \quad \boldsymbol{L}^{(F,y)} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ -\frac{1}{2} & -1 & -\frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix},$$
$$\boldsymbol{L}^{(F,xy)} = \begin{bmatrix} -\frac{1}{4} & 0 & \frac{1}{4} \\ 0 & 0 & 0 \\ \frac{1}{4} & 0 & -\frac{1}{4} \end{bmatrix}. \tag{73}$$

7B. ***Results in three dimensions.*** Our theoretical calculations have helped in formulating a complete three-stage Runge–Kutta scheme for solving the full LLNS system in one, two or three spatial dimensions. We have discussed how to generate stochastic fluxes in each stage, including the required correlations among the components of the stochastic stress, and have also discussed how to relate the stochastic fluxes in each stage. Since theoretical calculation of the three-dimensional structure factors is out of reach, we present some numerical results for the RK3-2RNG method in three dimensions with the mixed MAC/Fortin handling of the split Laplacian as given in Equations (70) and (71), hereafter termed the *RK3D-2RNG algorithm*.

We note in passing that it is also possible to discretize the modified Laplacian (see Section 7A1) using a MAC-like discretization of the viscous and stochastic stresses that avoids the use of the Fortin corner-based discretization of the divergence stress. This saves one random number per cell per stochastic flux, however, it requires the use of a nonstandard randomized cell-to-face projection (splitting) of the stochastic stresses that complicates the analysis and handling of physical boundaries and makes parallelization more difficult. We therefore do not describe this approach here, and only note that it produces very similar structure factors to those reported here.

We focus on the behavior of the scheme in global equilibrium with periodic boundary conditions. We have implemented the full nonlinear fluxes as proposed in [13; 10], using the interpolation in (59) for the hyperbolic fluxes and simple interpolation of the spatially varying viscosity and thermal conductivity in the handling of the viscous and stochastic fluxes. However, in the tests reported here we have made the magnitude of the fluctuations small compared to the means to ensure that the behavior is very similar to the linearized LLNS equations. Including the full

nonlinear system guarantees conservation and ensures that there are no nonlinearly unstable modes. More careful study of the proper handling of nonlinearity in the LLNS equations themselves and the associated numerical solvers is deferred to future publications; here, we focus on verification that the nonlinear scheme produces behavior consistent with the linearized analysis. We note that we have implemented the new RK3D algorithm also for the LLNS equations for a mixture of two ideal gases, closely following the original scheme described in [10]. We find that the spatial discretization satisfies the discrete fluctuation-dissipation balance even in the presence of concentration as an additional primitive variable and that the RK3D-2RNG method performs very well with reasonably large time steps.

7B1. *Static structure factors.* Examples of static structure factor $\boldsymbol{S}_{\boldsymbol{k}}$ for the RK3D-2RNG scheme are shown in Figure 3, showing that the diagonal components $S_{\boldsymbol{k}}^{(\rho)}$, $S_{\boldsymbol{k}}^{(v_x)}$, and $S_{\boldsymbol{k}}^{(T)}$ are close to unity, while the off-diagonal components $S_{\boldsymbol{k}}^{(\rho,v_x)}$, $S_{\boldsymbol{k}}^{(v_x,v_y)}$, and $S_{\boldsymbol{k}}^{(\rho,T)}$ are close to zero (similar results hold for $S_{\boldsymbol{k}}^{(v_x,T)}$, not shown), even for a large time step (half of the stability limit). Note that the static structure factor is difficult to obtain accurately for the smallest wavenumbers (slowest modes) and therefore the values near the centers of the $\boldsymbol{k}$-grid should be ignored.

It is seen in the figures that the diagonal components of $\boldsymbol{S}_{\boldsymbol{k}}$ are quite close to unity for the largest wavevectors, which is somewhat surprising, and the largest error is actually seen for intermediate wavenumbers, consistent with the one-dimensional results shown in Figure 2. We have tested the method on several cell Reynolds numbers r and found that the results are worse as r increases, consistent with the previous analysis, however, the higher order of temporal accuracy allows for increasing the time step to be a reasonable fraction of the stability limit even for large r.

These results represent a significant improvement over the results obtained for the original RK3 scheme presented in Bell et al. [13; 10]. Results with the original scheme were sensitive to time steps, requiring small time steps to obtain satisfactory results; the new scheme produces satisfactory results for time steps near the stability limit. Also, through the use of the mixed MAC and Fortin discretization, the new scheme eliminates a weak but spurious correlation $S_{\boldsymbol{k}}^{(v_x,v_y)}$ present in the original scheme for small wavenumbers even in the limit of small time steps.

7B2. *Dynamic structure factors.* Examples of dynamic structure factors $\boldsymbol{S}_{\boldsymbol{k},\omega}$ for the RK3D-2RNG scheme are shown in Figure 4 as a function of ω for two relatively large wavevectors, along with the correct continuum result obtained by solving the system (4) through a space-time Fourier transform (we did not make any of the usual approximations made in analytical calculations of $\boldsymbol{S}_{\boldsymbol{k},\omega}$ [20], and instead used Maple's numerical linear algebra). It is well known that $\boldsymbol{S}_{\boldsymbol{k},\omega}^{(\rho)}$ and $\boldsymbol{S}_{\boldsymbol{k},\omega}^{(T)}$ exhibit three peaks for a given $\boldsymbol{k}$ [20], one central Rayleigh peak at $\omega = 0$ similar to the

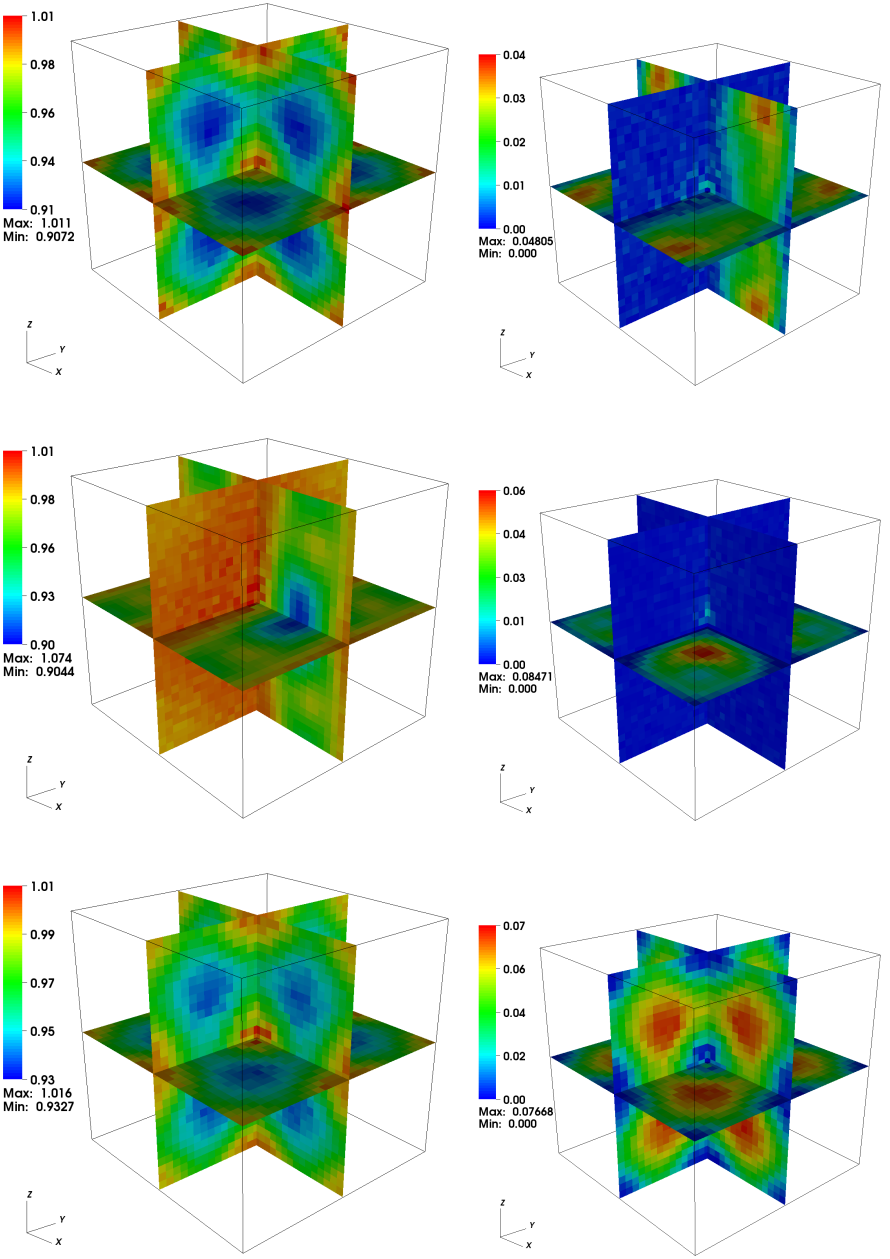


Figure 3. Left: $S_{\boldsymbol{k}}^{(\rho)}$, $S_{\boldsymbol{k}}^{(v_x)}$, and $S_{\boldsymbol{k}}^{(T)}$ (top to bottom). Right: $|S_{\boldsymbol{k}}^{(\rho,v_x)}|$, $|S_{\boldsymbol{k}}^{(v_x,v_y)}|$ and $|S_{\boldsymbol{k}}^{(\rho,T)}|$ (top to bottom) for RK3D-2RNG (random direction), with the time step $\alpha = 0.5$, $\beta = 3\beta_T/2 = 0.1$, periodic boundary conditions with 30^3 cells, and averaging over 10^6 time steps.

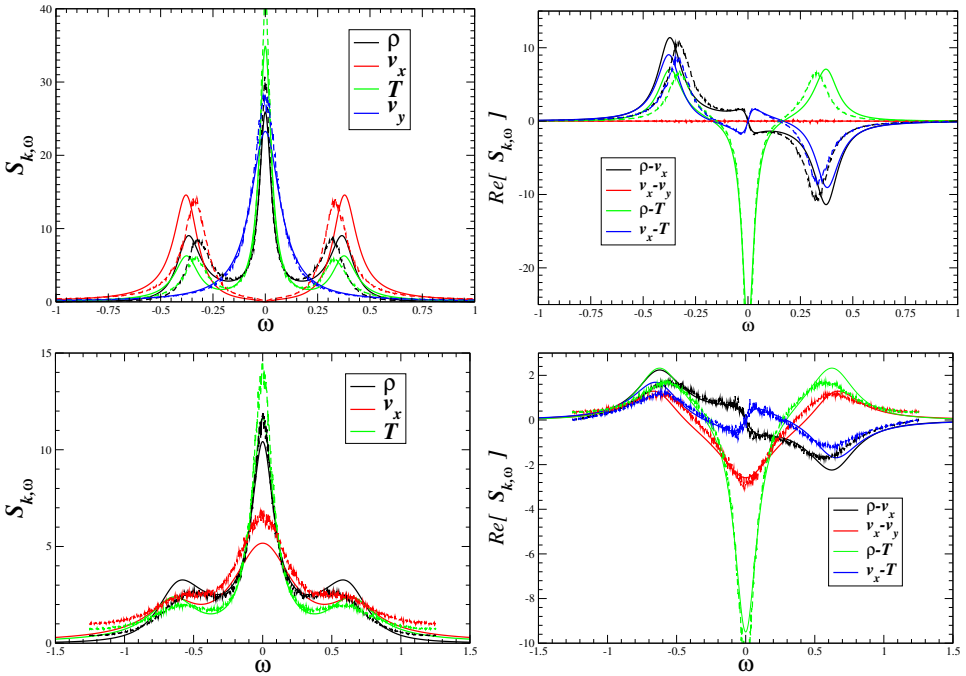


Figure 4. Diagonal (left) and the real part of the off-diagonal (right) components of the dynamic structure factor $\boldsymbol{S}_{\boldsymbol{k},\omega}$ for RK3D-2RNG (dashed lines) for the same parameters as in Figure 3. For comparison, the analytical solution of the LLNS equations in Fourier space are also shown (solid lines). The imaginary part of the off-diagonal components is less than 0.1 and it vanishes in the theory. The top part shows the wavevector $\boldsymbol{k} = (k_{\max}/2, 0, 0)$ and the bottom shows $\boldsymbol{k} = (k_{\max}/2, k_{\max}/2, k_{\max}/2)$.

peak for the diffusion equation given in (43), and two symmetric Brillouin peaks at $\omega \approx c_s k$, where c_s is the adiabatic speed of sound, $c_s = c_T\sqrt{1+2/d_f}$ for an ideal gas. For the velocity components, the transverse components $\boldsymbol{S}^{(v_\perp)}_{\boldsymbol{k},\omega}$ exhibit all three peaks, while the longitudinal component $S^{(v_\parallel)}_{\boldsymbol{k},\omega}$ lacks the central peak, as seen in the figure. Note that as the fluid becomes less compressible (i.e., the speed of sound increases), there is an increasing separation of time-scales between the side and central spectral peaks, showing the familiar numerical stiffness of the full compressible Navier–Stokes equations.

We have verified that for small wavevectors the numerical dynamic structure factors are in excellent agreement with the analytical predictions, even for such large time steps. For wavevectors that are not small compared to the discretization limits we do not expect a perfect dynamic structure factor, even for very small

time steps. It is important, however, that the discretization behave reasonably for all wavevectors (e.g., there should be no spurious maxima), and be somewhat accurate for intermediate wavevectors, even for large time steps. As seen in Figure 4, the RK3D-2RNG algorithm seems to perform well even with a large time step. Improving the accuracy at larger wavevectors requires using higher-order spatial differencing [50] (see discussion in Section 5C), compact stencils (linear solvers) [48], or pseudospectral methods [30], each of which has certain advantages but also significant disadvantages over the finite-volume approach in a more general nonlinear nonequilibrium context.

8. Summary and concluding remarks

We analyzed finite-volume schemes for the linearized Landau–Lifshitz Navier–Stokes (LLNS) system (4) and related SPDEs such as the stochastic advection-diffusion Equation (35). Our approach to studying the accuracy of these explicit schemes is based on evaluating the discrete static and dynamic structure factors, focusing on the accuracy at small wavenumber $\Delta k = k\Delta x$ and wavefrequency $\Delta\omega = \omega\Delta t$. The methodology for formulating the structure factor for numerical schemes is developed in Section 3, and then specialized to stochastic conservation laws in Section 4. Applying this analysis to the stochastic heat Equation (42) in Section 5 we find the truncation error for the Euler method to be $O(\Delta t k^2)$; the error for a standard predictor-corrector scheme is $O(\Delta t^2 k^4)$ using the same random numbers in the predictor and corrector stages but $O(\Delta t^3 k^6)$ using independent random numbers at each stage. Section 6 extends this analysis to the third-order Runge–Kutta scheme of Bell et al. [13; 10] for the one-dimensional advection-diffusion SPDE. We find the best accuracy when the stochastic fluxes at the three stages are generated from two sets of random numbers, as given by (62); using this version, called RK3-2RNG, for the LLNS equations gives good results, even when nonlinear effects are included (see Figures 2–4). Finally, Section 7 explains why the cross-correlations in the stress tensor in the three-dimensional LLNS require special treatment and proposes a mixed MAC/Fortin discretization as a way to obtain the desired discrete fluctuation-dissipation balance.

Here we have investigated linearized PDEs with stochastic fluxes where the noise is additive. As such, the stability properties of the numerical schemes are the same as for the deterministic case. Yet in practice one would like to implement these schemes for the nonlinear stochastic PDEs with state-dependent stochastic fluxes. While in the limit of small fluctuations the behavior of the schemes is expected to be similar to the linearized case, the proper mathematical foundation and even formulation of the nonlinear fluctuating equations has yet to be laid out. Furthermore, the stability properties of numerical schemes for the nonlinear

LLNS system are not well understood and the whole notion of stability is different than it is for deterministic schemes. For example, even at equilibrium, a rare fluctuation can cause a thermodynamic instability (e.g., a negative temperature which implies a complex sound speed) or a mechanical instability (e.g., a negative mass density). Capping the noises in the stochastic flux terms will not necessarily solve the problem because the hydrodynamic variables are time-correlated so the numerical instability may not appear on a single step but rather as an accumulated effect. We are investigating these issues and will discuss strategies to address this type of stability issue in future publications.

One of the advantages of finite volume solvers over spectral methods is the ability to implement realistic, complex geometries for fluid simulations. In this paper we only consider periodic boundaries but many other boundary conditions are of interest, notably, impenetrable flat hard walls with stick and slip conditions for the velocities and either adiabatic (zero temperature gradient) or thermal (constant temperature) conditions for the temperature. Equilibrium statistical mechanics requires that the static structure factor be oblivious to the presence of walls, even though the dynamic structure factors typically exhibit additional peaks due to the reflections of fluctuations from the boundaries [25]. Therefore, the numerical discretization of the Laplacian operator $\boldsymbol{L}$, the divergence operator $\boldsymbol{D}$ and the covariance of the stochastic fluxes $\boldsymbol{C}$ should continue to satisfy the discrete fluctuation-dissipation balance condition $\boldsymbol{L} + \boldsymbol{L}^\star = -2\boldsymbol{D}\boldsymbol{C}\boldsymbol{D}^\star$ and be consistent, even in the presence of boundaries. Standard treatments of boundary conditions used in deterministic schemes can easily be implemented in the stochastic setting [13; 6], however, satisfying the discrete fluctuation-dissipation balance is not trivial and requires modifying the stochastic fluxes and possibly also the finite-difference stencils near the boundaries [6], as briefly discussed in the Appendix to [25]. In particular, the case of Dirichlet boundary conditions is more complicated, especially in the case of the mixed MAC and Fortin discretization of the compressible Navier–Stokes equations. Complex boundaries present further challenges even in the deterministic setting. We will explore the issues associated will fluctuations at physical boundaries in future publications.

One motivation for the development of numerical methods for the LLNS equations is for their use in multialgorithm hybrids. One emerging paradigm in the modeling and simulation of multiscale problems is multialgorithm refinement (MAR). MAR is a general simulation approach that combines two or more algorithms, each of which is appropriate for a different scale regime. MAR schemes typically couple structurally different computational schemes such as particle-based molecular simulations with continuum partial differential equation (PDE) solvers. The general idea is to perform detailed calculations using an accurate but expensive algorithm in a small region (or for a short time), and couple this computation to

a simpler, less expensive method applied to the rest. The major difficulty is in constructing hybrid is that particle and continuum methods treat thermal noise (fluctuations) in completely different ways. The challenge is to ensure that the numerical coupling of the particle and continuum computations is self-consistent, stable, and most importantly, does not adversely impact the underlying physics. These problems become particularly acute when one wants to accurately capture the physical fluctuations at micro- and mesoscopic scales. The correct treatment of boundary conditions in stochastic PDE schemes is particularly difficult yet crucial in hybrid schemes since the coupling of the two algorithms is essentially a dynamic, two-way boundary condition. Recent work by Tysanner et al. [62], Foo et al. [12], Williams et al. [64] and Donev et al. [25] has demonstrated the need to model fluctuations at the continuum level in hybrid continuum / particle approaches, however, a seamless coupling has yet to be developed.

In this paper we consider the fully compressible LLNS system, for many of the phenomena of interest the fluid flow aspects occur at very low Mach numbers. Another topic of future work for stochastic PDE schemes is to construct a low Mach number fluctuating hydrodynamics algorithm. A number of researchers have considered extended versions of the incompressible Navier–Stokes equations that include a stochastic stress tensor [56; 61; 8]. This type of model does introduce fluctuations into the Navier–Stokes equations and is applicable in some settings, such as in modeling simple Brownian motion. However, as pointed out by Zaitsev and Shliomis [66], the incompressible approximation introduces fictitious correlations between the velocity components of the fluid. Furthermore, this type of approach does not capture the full range of fluctuations in the compressible equations. In particular, adding a stochastic stress into the incompressible Navier–Stokes equations creates fluctuations in velocity but does not reproduce the large scale and slow fluctuations in density and temperature, which persist even in the incompressible limit. We plan to investigate alternative formulations that can capture more of the features of the fluctuating hydrodynamics while still exploiting the separation of scales inherent in low Mach number flows. We also note that although the theoretical importance of distinguishing between the incompressible approximation and the low Mach number limit is well established for fluctuating hydrodynamics [14; 67], numerical algorithms for the latter have yet to be developed.

Appendix: Semi-implicit Crank–Nicolson method

When sound is included in the fluctuating hydrodynamic equations implicit methods are not really beneficial since the large sound speed limits the time step. However, for the pure stochastic diffusion/heat equation or advection-diffusion equations with a small advection speed the time step may become strongly limited by the

diffusive CFL limit, especially for small cells. In such cases an implicit method can be used to lift the diffusive stability restriction on the time step. For example, the second-order (in both space and time) Crank–Nicolson semi-implicit scheme for the stochastic heat equation entails solving the linear system

$$u_j^{n+1} - \frac{\mu\Delta t}{2\Delta x^2}(u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}) = u_j^n + \frac{\mu\Delta t}{2\Delta x^2}(u_{j-1}^n - 2u_j^n + u_{j+1}^n) + \sqrt{2\mu}\frac{\Delta t^{1/2}}{\Delta x^{3/2}}(W_{j+1/2}^n - W_{j-1/2}^n), \quad \text{(A.1)}$$

which is tridiagonal except at periodic boundaries.

The analysis carried out above for explicit schemes can easily be extended to implicit methods since in Fourier space different wavevectors again decouple and the above iteration becomes a scalar linear equation for $\hat{u}_k^{n+1}$ that can trivially be solved. Firstly, it is observed that the small time step limit is the same regardless of the semi-implicit treatment, specifically, the same discrete fluctuation-dissipation condition (31) applies. Remarkably, for the Crank–Nicolson iteration (A.1) it is found that the discrete static structure factor is independent of the time step, $S_k = 1$ for all β. The dynamic structure factor, however, has the same spatial discretization errors (48) as for the Euler scheme even in the limit $\beta \to 0$. Furthermore, as expected, the dynamics is not accurate for large β and the time step cannot be enlarged much beyond the diffusive stability limit related to the smallest length-scale at which one wishes to correctly resolve the dynamics of the fluctuations.

If advection is included as well and also discretized semi-implicitly, the method again gives perfect structure factors, $S_k = 1$ identically, and is unconditionally stable. If only diffusion is handled semi-implicitly but advection is handled with a predictor-corrector approach, then it turns out that the optimal method is to not include a stochastic flux in the predictor step, giving the same leading-order error term as PC-2RNG in (55) when $|r| > 0$, but giving a perfect $S_k = 1$ when $r = 0$.

Acknowledgments

The authors thank Berni Alder and Jonathan Goodman for helpful discussions, and Paul Atzberger for inspiring perspectives on the discrete fluctuation dissipation relation and a critical reading of this paper.

References

[1] R. Adhikari, K. Stratford, M. E. Cates, and A. J. Wagner, *Fluctuating lattice boltzmann*, Europhysics Letters **71** (2005), 473–479.

[2] F. J. Alexander and A. L. Garcia, *The direct simulation Monte Carlo method*, Computers in Physics **11** (1997), no. 6, 588–593.

[3] A. S. Almgren, J. B. Bell, and W. G. Szymczak, *A numerical method for the incompressible Navier–Stokes equations based on an approximate projection*, SIAM J. Sci. Comput. **17** (1996), no. 2, 358–369. MR 96j:76104 Zbl 0845.76055

[4] R. D. Astumian and P. Hanggi, *Brownian motors*, Physics Today (2002), 33–39.

[5] P. J. Atzberger, *Stochastic eulerian-lagrangian methods for fluid-structure interactions with thermal fluctuations and shear boundary conditions*, Preprint. arXiv 0910.5739

[6] ———, *Spatially adaptive stochastic numerical methods for intrinsic fluctuations in reaction-diffusion systems*, Journal of Computational Physics **229** (2010), no. 9, 3474 – 3501.

[7] P. J. Atzberger, P. R. Kramer, and C. S. Peskin, *A stochastic immersed boundary method for fluid-structure dynamics at microscopic length scales*, J. Comput. Phys. **224** (2007), no. 2, 1255–1292. MR 2008g:74031 Zbl 1124.74052

[8] ———, *A stochastic immersed boundary method for fluid-structure dynamics at microscopic length scales*, J. Comput. Phys. **224** (2007), no. 2, 1255–1292. MR 2008g:74031 Zbl 1124.74052

[9] W. Bao and S. Jin, *High-order I-stable centered difference schemes for viscous compressible flows*, J. Comput. Math. **21** (2003), no. 1, 101–112. MR 2004c:65084 Zbl 1086.76052

[10] J. B. Bell, A. Garcia, and S. Williams, *Computational fluctuating fluid dynamics*, ESAIM: Mathematical Modelling and Numerical Analysis (2010), To appear.

[11] 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

[12] J. B. Bell, J. Foo, and A. L. Garcia, *Algorithm refinement for the stochastic Burgers' equation*, J. Comput. Phys. **223** (2007), no. 1, 451–468. MR 2008a:65010

[13] J. B. Bell, A. L. Garcia, and S. A. Williams, *Numerical methods for the stochastic Landau–Lifshitz Navier–Stokes equations*, Phys. Rev. E (3) **76** (2007), no. 1, 016708, 12. MR 2008i:76137

[14] I. Bena, F. Baras, and M. M. Mansour, *Hydrodynamic fluctuations in the Kolmogorov flow: Nonlinear regime*, Phys. Rev. E **62** (2000), no. 5, 6560–6570.

[15] I. Bena, M. Malek Mansour, and F. Baras, *Hydrodynamic fluctuations in the Kolmogorov flow: linear regime*, Phys. Rev. E (3) **59** (1999), no. 5, part B, 5503–5510. MR 1690930

[16] J. Borrill and M. Gleiser, *Matching numerical simulations to continuum field theories: A lattice renormalization study*, Nuclear Phys. B **483** (1997), 416–428.

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

[18] P. Collela and P. R. Woodward, *The piecewise parabolic method (ppm) for gas dynamics calculations*, J. Comp. Phys **54** (1984), 174.

[19] G. Da Prato, *Kolmogorov equations for stochastic PDEs*, Birkhäuser, Basel, 2004. MR : MR 2005m:60002 Zbl 1066.60061

[20] J. M. O. de Zarate and J. V. Sengers, *Hydrodynamic fluctuations in fluids and fluid mixtures*, Elsevier Science, 2007.

[21] A. Debussche and J. Printems, *Weak order for the discretization of the stochastic heat equation*, Math. Comp. **78** (2009), no. 266, 845–863. MR 2010f:60192

[22] R. Delgado-Buscalioni and A. Dejoan, *Nonreflecting boundaries for ultrasound in fluctuating hydrodynamics of open systems*, Phys. Rev. E **78** (2008), no. 4, 046708.

[23] R. Delgado-Buscalioni and G. D. Fabritiis, *Embedding molecular dynamics within fluctuating hydrodynamics in multiscale simulations of liquids*, Phys. Rev. E **76** (2007), no. 3, 036709.

[24] C. V. den Broeck, R. Kawai, and P. Meurs, *Exorcising a Maxwell demon*, Phys. Rev. Lett. **93** (2004), 090601.

[25] A. Donev, J. B. Bell, A. L. Garcia, and B. J. Alder, *A hybrid particle-continuum method for hydrodynamics of complex fluids*, SIAM J. Multiscale Modeling and Simulation **8** (2010), no. 3, 871–911.

[26] B. Dünweg, U. D. Schiller, and A. J. C. Ladd, *Statistical mechanics of the fluctuating lattice Boltzmann equation*, Phys. Rev. E (3) **76** (2007), no. 3, 036704, 10. MR 2008i:82096

[27] J. Eggers, *Dynamics of liquid nanojets*, Phys. Rev. Lett. **89** (2002), no. 8, 084502.

[28] P. Español, *Stochastic differential equations for non-linear hydrodynamics*, Physica A **248** (1998), no. 1-2, 77–96.

[29] G. D. Fabritiis, M. Serrano, R. Delgado-Buscalioni, and P. V. Coveney, *Fluctuating hydrodynamic modeling of fluids at the nanoscale*, Phys. Rev. E **75** (2007), no. 2, 026307.

[30] B. Fornberg, *A practical guide to pseudospectral methods*, Cambridge Monographs on Applied and Computational Mathematics, no. 1, Cambridge University Press, Cambridge, 1996. MR 97g:65001 Zbl 0844.65084

[31] M. Fortin, *Numerical solution of the steady state Navier–Stokes equations*, Numerical Methods in Fluid Dynamics (J. J. Smolderen, ed.), Technical Editing and Reproduction Ltd., London, 1972, pp. 5.1–5.7 AGARD–LS–48.

[32] A. L. Garcia, M. M. Mansour, G. Lie, and E. Clementi, *Numerical integration of the fluctuating hydrodynamic equations*, J. Stat. Phys. **47** (1987), 209.

[33] A. L. Garcia and C. Penland, *Fluctuating hydrodynamics and principal oscillation pattern analysis*, J. Stat. Phys. **64** (1991), 1121.

[34] C. W. Gardiner, *Handbook of stochastic methods for physics, chemistry and the natural sciences*, 3rd ed., Springer Series in Synergetics, no. 13, Springer, Berlin, 2004. MR 2004m:00008 Zbl 1143.60001

[35] G. Giupponi, G. D. Fabritiis, and P. V. Coveney, *Hybrid method coupling fluctuating hydrodynamics and molecular dynamics for the simulation of macromolecules*, J. Chem. Phys. **126** (2007), no. 15, 154903.

[36] J. Goodman and A. D. Sokal, *Multigrid Monte Carlo method: Conceptual foundations*, Phys. Rev. D **40** (1989), no. 6, 2035–2071.

[37] S. Gottlieb and C.-W. Shu, *Total variation diminishing Runge–Kutta schemes*, Math. Comp. **67** (1998), no. 221, 73–85. MR 98c:65122 Zbl 0897.65058

[38] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluids with free surfaces*, Physics of Fluids **8** (1965), 2182–2189.

[39] A. Jentzen and P. E. Kloeden, *The numerical approximation of stochastic partial differential equations*, Milan J. Math. **77** (2009), 205–244. MR 2578878 Zbl 1186.65011

[40] K. Kadau, C. Rosenblatt, J. L. Barber, T. C. Germann, Z. Huang, P. Carles, and B. J. Alder, *The importance of fluctuations in fluid mixing*, PNAS **104** (2007), no. 19, 7741–7745.

[41] K. Kadau, T. C. Germann, N. G. Hadjiconstantinou, P. S. Lomdahl, G. Dimonte, B. L. Holian, and B. J. Alder, *Nanohydrodynamics simulations: an atomistic view of the Rayleigh–Taylor instability*, Proc. Natl. Acad. Sci. USA **101** (2004), no. 16, 5851–5855. MR 2004m:76077 Zbl 1063.76029

[42] W. Kang and U. Landman, *Universality crossover of the pinch-off shape profiles of collapsing liquid nanobridges in vacuum and gaseous environments*, Phys. Rev. Lett. **98** (2007), no. 6, 064504.

[43] P. R. Kramer, C. S. Peskin, and P. J. Atzberger, *On the foundations of the stochastic immersed boundary method*, Comput. Methods Appl. Mech. Engrg. **197** (2008), no. 25-28, 2232–2249. MR 2009d:60269 Zbl 1158.76420

[44] R. Kubo, *The fluctuation-dissipation theorem*, Reports on Progress in Physics **29** (1966), no. 1, 255–284.

[45] A. J. C. Ladd, *Short-time motion of colloidal particles: Numerical simulation via a fluctuating lattice-Boltzmann equation*, Phys. Rev. Lett. **70** (1993), no. 9, 1339–1342.

[46] L. D. Landau and E. M. Lifshitz, *Fluid mechanics*, Course of Theoretical Physics, no. 6, Pergamon Press, London, 1959. MR 21 #6839 Zbl 0146.22405

[47] J. L. Lebowitz, E. Presutti, and H. Spohn, *Microscopic models of hydrodynamic behavior*, J. Statist. Phys. **51** (1988), no. 5-6, 841–862. MR 90c:60072 Zbl 1086.60531

[48] S. K. Lele, *Compact finite difference schemes with spectral-like resolution*, J. Comput. Phys. **103** (1992), no. 1, 16–42. MR 93g:76086 Zbl 0759.65006

[49] A. Lemarchand and B. Nowakowski., *Fluctuation-induced and nonequilibrium-induced bifurcations in a thermochemical system*, Molecular Simulation **30** (2004), no. 11-12, 773–780.

[50] K. Mahesh, *A family of high order finite difference schemes with good spectral resolution*, J. Comput. Phys. **145** (1998), no. 1, 332–358. MR 99d:76068 Zbl 0926.76081

[51] M. M. Mansour, A. L. Garcia, G. C. Lie, and E. Clementi, *Fluctuating hydrodynamics in a dilute gas*, Phys. Rev. Lett. **58** (1987), 874–877.

[52] M. M. Mansour, C. V. den Broeck, I. Bena, and F. Baras, *Spurious diffusion in particle simulations of the Kolmogorov flow*, Europhysics Letters **47** (1999), no. 1, 8–13.

[53] M. Mareschal, M. M. Mansour, G. Sonnino, and E. Kestemont, *Dynamic structure factor in a nonequilibrium fluid: A molecular-dynamics approach*, Phys. Rev. A **45** (1992), 7180–7183.

[54] P. Meurs, C. V. den Broeck, and A. L. Garcia, *Rectification of thermal fluctuations in ideal gases*, Phys. Rev. Lett. E **70** (2004), 051109.

[55] E. Moro, *Hybrid method for simulating front propagation in reaction-diffusion systems*, Phys. Rev. E **69** (2004), no. 6, 060101.

[56] M. Moseler and U. Landman, *Formation, stability, and breakup of nanojets*, Science **289** (2000), no. 5482, 1165–1169.

[57] B. Nowakowski and A. Lemarchand, *Sensitivity of explosion to departure from partial equilibrium*, Phys. Rev. E **68** (2003), 031105.

[58] G. Oster, *Darwin's motors*, Nature **417** (2002), 25.

[59] Y. Pomeau and P. Résibois, *Time dependent correlation functions and mode-mode coupling theories*, Phys. Rep. **19** (1975), 63–139.

[60] G. Quentin and I. Rehberg, *Direct measurement of hydrodynamic fluctuations in a binary mixture*, Phys. Rev. Lett. **74** (1995), no. 9, 1578–1581.

[61] N. Sharma and N. A. Patankar, *Direct numerical simulation of the Brownian motion of particles by using fluctuating hydrodynamic equations*, J. Comput. Phys. **201** (2004), 466–486.

[62] M. Tysanner and A. L. Garcia, *Non-equilibrium behavior of equilibrium reservoirs in molecular simulations*, Int. J. Numer. Meth. Fluids **48** (2005), 1337–1349.

[63] N. K. Voulgarakis and J.-W. Chu, *Bridging fluctuating hydrodynamics and molecular dynamics simulations of fluids*, J. Chem. Phys. **130** (2009), no. 13, 134111.

[64] S. A. Williams, J. B. Bell, and A. L. Garcia, *Algorithm refinement for fluctuating hydrodynamics*, Multiscale Model. Simul. **6** (2007), no. 4, 1256–1280. MR 2009c:76059 Zbl 05381404

[65] M. Wu, G. Ahlers, and D. S. Cannell, *Thermally induced fluctuations below the onset of Rayleigh–Bénard convection*, Phys. Rev. Lett. **75** (1995), no. 9, 1743–1746.

[66] V. M. Zaitsev and M. I. Shliomis, *Hydrodynamic fluctuations near convection threshold*, Sov. Phys. JETP **32** (1971), 866.

[67] R. Zwanzig and M. Bixon, *Compressibility effects in the hydrodynamic theory of Brownian motion*, Journal of Fluid Mechanics **69** (1975), 21–25.

Received June 12, 2009. Revised December 18, 2009.

ALEKSANDAR DONEV: aleks.donev@gmail.com
Lawrence Berkeley National Laboratory, Center for Computational Sciences and Engineering, MS 50A-1148, LBL, 1 Cyclotron Rd., Berkeley 94720, United States
http://cims.nyu.edu/~donev

ERIC VANDEN-EIJNDEN: eve2@cims.nyu.edu
New York University, Courant Institute of Mathematical Sciences, New York 10012, United States

ALEJANDRO GARCIA: algarcia@algarcia.org
San Jose State University, Department of Physics and Astronomy, San Jose, CA 95192, United States

JOHN BELL: jbbell@lbl.gov
Lawrence Berkeley National Laboratory, Center for Computational Science and Engineering, Berkeley, CA 94720, United States

COMM. APP. MATH. AND COMP. SCI.
Vol. 5, No. 2, 2010

A VOLUME-OF-FLUID INTERFACE RECONSTRUCTION ALGORITHM THAT IS SECOND-ORDER ACCURATE IN THE MAX NORM

ELBRIDGE GERRY PUCKETT

In an article recently published in this journal the author proved there exists a two-dimensional, volume-of-fluid interface reconstruction method that is second-order accurate in the max norm. However, that article did not include an example of such an algorithm. This article contains a description of a two-dimensional, volume-of-fluid interface reconstruction method that is second-order accurate in the max norm, provided the curve that one is reconstructing is two times continuously differentiable and the length of the sides of the square grid cells is less than a constant divided by the maximum of the absolute value of the curvature of the interface. A computation made with this algorithm is presented that demonstrates the convergence rate is second-order, as expected.

1. Introduction

Let $\Omega \in R^2$ denote a two-dimensional computational domain and take an oriented curve in Ω parametrized by $z(s) = (x(s), y(s))$, where $0 \leq s \leq s_{\text{end}}$ is arc length. Let L be a characteristic length of the computational domain Ω. Cover Ω with a grid consisting of square cells each of side $\Delta x \leq L$ and let

$$h = \frac{\Delta x}{L} \tag{1}$$

be a dimensionless parameter that represents the size of a grid cell. Note that h is bounded above by 1. For the remainder of this article it is to be understood that quantities such as the arc length s are also nondimensional quantities that have been obtained by division by L as in (1), and that the curvature κ has been nondimensionalized by dividing by $1/L$.

MSC2000: primary 76-04, 65M06, 65M15, 76M20, 76M25; secondary 74A50, 94A08, 74S99, 76T99.

Keywords: volume-of-fluid, interface reconstruction, front reconstruction, piecewise linear interface reconstruction, fronts, two-phase flow, multiphase systems, adaptive mesh refinement, computational fluid dynamics.

Sponsored by the U.S. Department of Energy Mathematical, Information, and Computing Sciences Division contracts DE-FC02-01ER25473 and DE-FG02-03ER25579.

The *volume-of-fluid interface reconstruction problem* is to compute an approximation $\tilde{z}(s)$ to $z(s)$ in Ω using only the volume fractions Λ_{ij} associated with the curve z on the grid. In this paper the convention will be that the volume fraction Λ_{ij} in the ij-th cell is the fraction of material that lies in the ij-th cell on the "outside" of $z(s)$ — that is, on the side towards which the outward pointing unit normal vector $\boldsymbol{n}$ to the interface $z(s)$ points.

The purpose of this article is to describe a new volume-of-fluid method for reconstructing the interface $z(s) = (x(s), y(s))$ between two materials that is second-order accurate in the max norm. The main result in [24; 25] is a proof that

$$\left| g(x) - \tilde{g}_{ij}(x) \right| \leq \left(\frac{50}{3} \kappa_{\max} + C_S \right) h^2 \quad \text{for all } x \in [x_i, x_{i+1}], \tag{2}$$

where $(x, \tilde{g}_{ij}(x))$ is the volume-of-fluid approximation to the interface[1] $(x, g(x))$ described in this paper; $\kappa_{\max}$ is the maximum of the absolute value of the curvature of the interface in the 3×3 block of cells B_{ij} centered on the cell C_{ij} in which one wishes to reconstruct the interface; $x = x_i$ and $x = x_{i+1}$ are the x-coordinates of the left and right edges of the cell $C_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ in which one wishes to reconstruct the interface; $h = \Delta x / L$ is the length defined in (1) of the edges of the square grid cells; and

$$C_S = \frac{\sqrt{3}}{2} \left\{ (2\sqrt{2} - 1) 4 \sqrt{\kappa_{\max}} + \left(1 - \frac{7\,(1+\sqrt{2})}{20} \right) \frac{32}{3} (\sqrt{\kappa_{\max}})^3 \right\}^2. \tag{3}$$

The bound in (2) holds whenever the grid size h and the maximum $\kappa_{\max}$ of the absolute value of the curvature $\kappa(s)$ of the interface $z(s)$

$$\kappa_{\max} = \max_s |\kappa(s)| \tag{4}$$

satisfies[2]

$$h \leq \frac{C_h}{\kappa_{\max}}, \tag{5}$$

where

$$C_h = \frac{1}{25}. \tag{6}$$

[1] As explained in the following paragraph, it is proven in [24] that the constraint on h in terms of $\kappa_{\max}$ in (5) ensures that one can reparametrize the interface as $y = g(x)$ or $x = G(y)$ locally about the cell $C_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ in which one wishes to reconstruct the interface. For convenience, in this article the interface will frequently be written as $y = g(x)$ instead of $z(s)$, it being understood that in some cells the parametrizations has to be $x = G(y)$.

[2] It is only necessary that this condition be satisfied in a neighborhood of the cell C_{ij}; for example, in the 3×3 block of cells B_{ij} centered on the cell C_{ij}.

Section 2 of [24] contains a proof that the constraint in (5) on h is sufficient to ensure that the interface $z(s) = (x(s), y(s))$ can be written as a single valued function $y = g(x)$ or $x = G(y)$ on any 3×3 block of cells B_{ij} centered on a cell C_{ij} which contains a portion of the interface.[3]

In the algorithm described in this article one uses the row of three cells above and below B_{ij} to determine which columns to use in the approximation to the slope m_{ij} of the piecewise linear approximation

$$\tilde{g}_{ij}(x) = m_{ij}\, x + b_{ij} \tag{7}$$

to the interface $y = g(x)$. For this reason one must consider the 5×5 block of cells $\tilde{B}_{ij}$ centered on the cell C_{ij}. However, as shown in Section 5, once one has rotated the block $\tilde{B}_{ij}$ so that the interface only enters (resp., exits) the 3×3 subblock B_{ij} across its left or top edge (resp., top or right edge); it is not necessary to use the first and last columns of the larger block of cells $\tilde{B}_{ij}$.

The articles [24; 25] consist solely of a collection of proofs showing that *there exists* a volume-of-fluid interface reconstruction algorithm that is second-order accurate in the max norm; that is, *they do not contain an example of such an algorithm.* However, the proofs in [24; 25] are constructive, and the algorithm described here is based on those proofs. To date, no other volume-of-fluid interface reconstruction algorithms have been proven to be second-order accurate in the max norm. Section 6 contains a computational example to demonstrate that this algorithm produces an approximation to $\cos x$ for $0 \le x \le \pi$ that is a second-order accurate in the max norm.

A detailed statement of the problem. Suppose that one is given a simply connected computational domain $\Omega \in R^2$ that is divided into two distinct, disjoint regions Ω_1 and Ω_2 such that $\Omega_1 \cup \Omega_2 = \Omega$. Let Ω_1 be referred to as material 1 and Ω_2 as material 2. (Although these algorithms have historically been known as "volume-of-fluid" methods, they are frequently used to model the interface between any two materials, including gases, liquids, solids and any combination thereof; for example, see [5; 14; 15; 16].)

Let $z(s) = (x(s), y(s))$, where s is arc length, denote the *interface* between these two materials and assume that the interface has been oriented so that as one traverses the interface with increasing arc length material 1 lies to the right. Cover Ω with a uniform square grid of cells, each with side h, and let Λ_{ij} denote the fraction of material 1 in the ij-th cell.

[3] In that article and this one interfaces that are undergoing topological changes, such as when a droplet separates into two droplets, are not considered. In order for this algorithm to achieve second-order accuracy, the interface must be a *single-valued* C^2 function in the 3×3 block B_{ij} surrounding the cell C_{ij}.

Each number Λ_{ij} satisfies $0 \leq \Lambda_{ij} \leq 1$ and is called the *volume fraction* (of material 1) in the ij-th cell.[4] Note that

$$0 < \Lambda_{ij} < 1 \tag{8}$$

if and only if a portion of the interface $z(s)$ lies in the ij-th cell. Similarly, $\Lambda_{ij} = 1$ means the ij-th cell only contains material 1, and $\Lambda_{ij} = 0$ means it only contains material 2.

Consider the following problem. Given only the collection of volume fractions Λ_{ij} in the grid covering Ω, *reconstruct* $z(s)$; in other words, find a piecewise linear approximation $\tilde{z}$ to z in each cell C_{ij} that contains a portion of the interface $z(s)$. Furthermore, the approximate interface $\tilde{z}$ must have the property that the volume fractions $\tilde{\Lambda}_{ij}$ due to $\tilde{z}$ are identical to the original volume fractions Λ_{ij}; that is,

$$\tilde{\Lambda}_{ij} = \Lambda_{ij} \quad \text{for all cells } C_{ij}. \tag{9}$$

An algorithm for finding such an approximation is known as a *volume-of-fluid interface reconstruction method*.

The property that $\tilde{\Lambda}_{ij} = \Lambda_{ij}$ is the principal feature that distinguishes volume-of-fluid interface reconstruction methods from other interface reconstruction methods. This ensures that the computational value of the total volume of each material is conserved. In other words, all volume-of-fluid interface reconstruction methods are conservative in that they conserve the volume of each material in the computation. When the underlying numerical method is a conservative finite difference method that one is using to model a system of hyperbolic conservation laws (e.g., gas dynamics) this can be essential since, for example, in order to obtain the correct shock speed it is necessary for all of the conserved quantities to be conserved by the underlying numerical method [13]. More generally, a necessary condition for the numerical method to converge to the correct weak solution of a system of hyperbolic conservation laws is that all of the quantities that are conserved in the underlying partial differential equation must be conserved by the numerical method [12].

Volume-of-fluid methods have been used by researchers to track material interfaces since at least the mid 1970s [18; 19]. Researchers have developed a variety of volume-of-fluid algorithms for modeling everything from flame propagation [3] to curvature and solidification [4]. In particular, the problem of developing high-order accurate volume-of-fluid methods for modeling the curvature and surface tension of an interface has received a lot of attention [1; 2; 4; 7; 32; 22]. Volume-of-fluid methods were among the first interface tracking algorithms to be implemented in codes originally developed at the U.S. National Laboratories, and subsequently

[4]Even though in two dimensions Λ_{ij} is technically an area fraction, the convention is to refer to it as a *volume fraction*.

released to the general public, that were capable of tracking material interfaces in a variety of complex material flow problems [6; 9; 17; 30; 31]. They continue to be widely used at these institutions, as well as by the general scientific community.

This article does not contain work concerning the related problem of approximating the movement of the interface in time, for which one would use a *volume-of-fluid advection algorithm*. The interested reader may wish to consult [21; 27; 28] for a detailed description and analysis of several such algorithms. In this article only the accuracy that one can obtain when using a volume-of-fluid interface reconstruction algorithm to approximate a given *stationary* interface $z(s)$ is considered.

2. Assumptions and definitions

Notation. The *center cell* $C_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ is the square grid cell with side h that contains a portion of the interface $z(s) = (x(s), y(s))$ for s in some interval, say $s \in (s_l, s_r)$, in which one wishes to reconstruct the interface. This is equivalent to saying that $0 < \Lambda_{ij} < 1$. In what follows the 5×5 block of square cells — each with side h — centered on the center cell C_{ij}, as shown, for example, in Figure 1, will be denoted $\tilde{B}_{ij} = [x_{i-2}, x_{i+3}] \times [y_{j-2}, y_{j+3}]$. In addition, the subblock of $\tilde{B}_{ij}$ which consists of the 3×3 subblock of cells centered on the cell

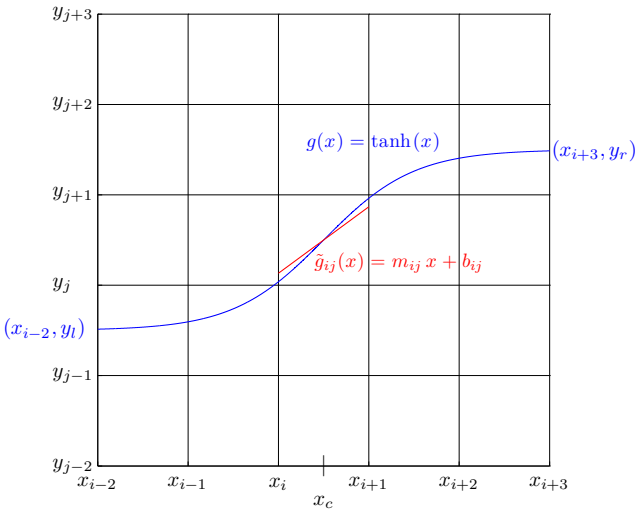


Figure 1. In this example the interface is $g(x) = \tanh(x)$ and material 1 lies *below* the curve. Note that all three of the column sums are *exact*, but that for the inverse function $x = g^{-1}(y)$, only the (horizontal) center column sum is exact. (Exact column sums are defined in Section 4 below.) The cell in which one wishes to reconstruct the interface is C_{ij}, the 3×3 block of cells centered on C_{ij} is B_{ij}, and the 5×5 block of cells centered on C_{ij} is $\tilde{B}_{ij}$.

C_{ij} will be denoted $B_{ij} = [x_{i-1}, x_{i+2}] \times [y_{j-1}, y_{j+2}]$, and the 3×5 subblock of $\tilde{B}_{ij}$, which is B_{ij} together with the row of 3 cells $C_{i-1,j-2}$, C_{ij-2} and $C_{i+1,j-2}$ added to its bottom and the row of 3 cells $C_{i-1,j+2}$, $C_{i,j+2}$ and $C_{i+1,j+2}$ added to its top, will be denoted $\hat{B}_{ij} = [x_{i-1}, x_{i+2}] \times [y_{j-2}, y_{j+3}]$. The coordinates of the vertical edges of the cells in $\tilde{B}_{ij}$ are denoted x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2} and x_{i+3} and the horizontal edges by y_{j-2}, y_{j-1}, y_j, y_{j+1}, y_{j+2} and y_{j+3} as shown, for example, in Figure 1.[5] It will always be the case that $x_{i+1} - x_i = h$, $y_{j+1} - y_j = h$, etc.

Assumptions concerning the interface. In this article the exact interface

$$z(s) = ((x(s), y(s))$$

is assumed to satisfy the following conditions:

I. The interface z is two times continuously differentiable; in other words,

$$z(s) \in C^2(\Omega). \tag{10}$$

II. The grid size h and the maximum value

$$\kappa_{\max} = \max_s |\kappa(s)|$$

of the absolute value of the curvature $\kappa(s)$ of the interface satisfy the following constraint in terms of each other:[6]

$$h \leq \frac{C_h}{\kappa_{\max}}, \tag{11}$$

where

$$C_h = \frac{1}{25}.$$

Remark. One can show that the constraint in (11) prevents configurations in which the interface enters the center cell C_{ij}, exits it, and then enters it again, before exiting the 5×5 block of cells $\tilde{B}_{ij}$, as shown in Figure 2. In particular, one can use the constraint in (11) to show that the interface does not have hairpin turns which are on the order of a grid cell. See [24] for a proof of these facts.

Note that it may be possible for the interface to pass through the center cell, then exit the 3×3 block B_{ij}, "wander around the computational domain", and then reenter the 3×3 block B_{ij} and the center cell C_{ij} again. The constraint in (11) simply guarantees that given a point on the interface that lies in the center cell C_{ij}, one can find an orientation of the 3×3 block B_{ij} such that *locally* the interface can be written as a single-valued function on the interval $[x_{i-1}, x_{i+2}]$ such that the

[5] Whenever possible, the same notation is used in this article as in [24; 25]. One significant change, however, is that the lower left corner of the center cell C_{ij} is now (x_i, y_j) rather than (x_{i-1}, y_{j-1}) as it was denoted in [24].

[6] See footnote 2.

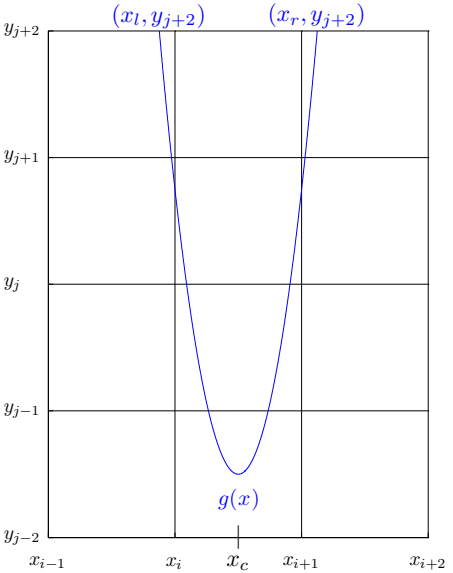


Figure 2. In this example, $h = 1$ and the interface is the parabola $g(x) = a(x - x_c)^2 - \frac{1}{2}$ with $a = 9$. Consequently, the maximum curvature of the interface is

$$\kappa_{\max} = 2a = 18 > C_h\, h^{-1} = \frac{1}{25},$$

and hence the constraint on the cell size h in (5) is not satisfied. As one can see from the figure, the interface enters the 3×3 block of cells B_{ij} through the top edge of the left column, passes through the center cell C_{ij}, exits the 3×3 block of cells B_{ij} through the bottom edge of the center column (i.e., the line $y = y_{j-1}$), and then passes through B_{ij} again; the second path being a reflection about the line $x = x_c$ of the first. The constraint on h with respect to the maximum curvature $\kappa_{\max}$ in (5) ensures that the interface does not have sharp or "hairpin" turns that are on the scale of the 3×3 block of cells B_{ij}, such as the one illustrated here. A finer grid (i.e., smaller h) is required in order to resolve curves such as this one.

curve enters the 3×3 block B_{ij}, passes through the center cell once — and only once — and then exits B_{ij}. It does not prevent the curve from eventually reentering the center cell after traversing the domain for a large number of cell lengths. In this article it is assumed that this latter case does not occur.

3. Rotation of the 5 by 5 block

Given a cell C_{ij} that contains a portion $y = g(x)$ of the interface, or equivalently, a cell C_{ij} in which $0 < \Lambda_{ij} < 1$, assume that the 5×5 block of cells $\tilde{B}_{ij}$ centered on C_{ij} has been rotated so that *in the rotated coordinate frame* the bottom row of cells in the 3×5 subblock of cells $\hat{B}_{ij}$ satisfy

$$\Lambda_{i-1,j-2} = 1, \quad \Lambda_{ij-2} = 1, \quad \Lambda_{i+1,j-2} = 1.$$

This ensures that the interface does not exit the 3×5 subblock of cells $\hat{B}_{ij}$ across its bottom edge. Not only does this reduce the number of cases that one must consider in the description of the algorithm, but it also reduces the number of cases one must consider in the implementation of the algorithm. This is because the *Symmetry Lemma* of [24, page 119] states that all configurations of the interface with respect to the 3×3 block B_{ij} are equivalent to the following two cases:

Configuration A: The interface enters B_{ij} across the left edge of B_{ij} and exits across the right edge of B_{ij} (as shown, for example, in Figure 1).

Configuration B: The interface enters B_{ij} across the left edge of B_{ij} and exits across the top edge of B_{ij} (as shown, for example, in Figure 4).

Provided only that the interface satisfies the conditions in (5) and (10), these two cases are equivalent to all of the other ways in which the interface can enter the 3×3 block of cells B_{ij}, pass through the center cell C_{ij}, and exit the block B_{ij}. In other words, rotating the block B_{ij} by 0, 90, 180 or 270 degrees and/or interchanging the direction traversed by the arc length parameter $s \to -s$, one can arrive at a configuration that is identical to one of the two configurations listed above. This is a consequence of the Symmetry Lemma cited above.

In the implementation of this algorithm, for the purposes of producing the results shown in Section 6, the case in which — after the 5×5 block of cells $\tilde{B}_{ij}$ has been rotated — the interface enters B_{ij} across the top edge and exits it across the right edge is also included. In other words, the symmetric image of the example shown in Figure 4 is also included in this implementation of the algorithm, although according to the Symmetry Lemma this is not strictly necessary.

During the course of proving the results in [24; 25], or in developing a second-order accurate volume-of-fluid interface reconstruction method such as the one described here, it is often necessary to rotate the 5×5 block of cells $\tilde{B}_{ij}$ centered on C_{ij} by 90, 180, or 270 degrees and/or reflect the coordinates about one of the coordinate axes: $x \to -x$ or $y \to -y$. No other coordinate transformations besides one of these three rotations and a possible reversal of one or both of the variables $x \to -x$ and/or $y \to -y$ are required in order for the algorithm studied in this article and the articles in [24; 25] to converge to the exact interface as $h \to 0$. Furthermore,

these coordinate transformations are only used to determine a first-order accurate approximation m_{ij} to $g'(x_c)$ in the center cell. The grid covering the domain Ω always remains the same.

Thus, if one is using the interface reconstruction algorithm as part of a numerical method to solve a more complex problem than the one posed here (e.g., the movement of a fluid interface where the underlying fluid flow is a solution of the Euler or Navier–Stokes equations), it is not necessary to perform these coordinate transformations on the underlying numerical fluid flow solver. Therefore, unless noted otherwise, in what follows the interface will always be written $y = g(x)$ and the coordinates of the edges of the cells in the 3×3 block B_{ij} will be denoted by x_{i-1}, x_i, x_{i+1}, x_{i+2} and y_{j-1}, y_j, y_{j+1}, y_{j+2}, it being implicitly understood that a transformation of the coordinate system as described above may have been performed in order for this representation of the interface to be valid, and that the names of the variables x and y might have been interchanged in order to write the interface as $y = g(x)$.

4. Column sums

Let S_{i-1}, S_i and S_{i+1} represent the left, center and right *column sums* respectively in the 3×3 subblock of cells B_{ij} centered on C_{ij}:

$$S_{i-1} = \sum_{j'=j-1}^{j+1} \Lambda_{i-1,j'}, \quad S_i = \sum_{j'=j-1}^{j+1} \Lambda_{ij'}, \quad S_{i+1} = \sum_{j'=j-1}^{j+1} \Lambda_{i+1,j'}. \tag{12}$$

The volume fraction Λ_{ij} in the ij-th cell C_{ij} is a nondimensional way of storing the volume of material 1 in that cell, while the i-th *column sum* S_i defined above is a nondimensional way of storing the total volume of material 1 in the column of three cells centered on the ij-th cell, and similarly for S_{i-1} and S_{i+1}.

Now consider an arbitrary column consisting of three cells with left edge $x = x_i$ and right edge $x = x_{i+1}$. Furthermore, assume that the interface can be written as a function $y = g(x)$ on the interval $[x_i, x_{i+1}]$. Assume also that the interface enters the column through its left edge, exits the column through its right edge and does not cross the top or bottom edges of the column, as is the case with each of the columns S_{i-1}, S_i and S_{i+1} in the 3×3 subblock of cells B_{ij} centered on the cell of interest C_{ij} shown in the example in Figure 1. Then, in particular, the total volume of material 1 that occupies the three cells of the center column and lies below the interface $g(x)$ is equal to the integral of $(g(x) - y_{j-1})$ over the interval $[x_i, x_{i+1}]$. This leads to the following relationship between the column sum and the normalized volume of material 1 in the column:

$$S_i = \sum_{j'=j-1}^{j+1} \Lambda_{ij'} = \frac{1}{h^2} \int_{x_i}^{x_{i+1}} (g(x) - y_{j-1})\, dx. \tag{13}$$

This in turn leads to the following definition.

Definition. Assume that the interface $y = g(x)$ enters the i-th column through its left edge and exits the i-th column through its right edge and does not cross the top or bottom edges of the column. Then the column sum S_i is *exact* whenever (13) holds. Integrals such as the one on the right in (13) will be referred to as *the normalized integral of* g in the i-th column.[7]

Given the 3×3 block of cells B_{ij} surrounding a cell C_{ij} that contains a portion $y = g(x)$ of the interface, the accuracy of the algorithm described in this paper is based on how well the column sums S_{i-1}, S_i and S_{i+1} approximate the normalized integral of g in the $(i-1)$-st, i-th, and $(i+1)$-st column. This is because if two of the column sums $S_{i+\alpha}$ and $S_{i+\beta}$ with $\alpha, \beta = 1, 0, -1$ and $\alpha \neq \beta$ are exact, then the slope

$$m_{ij} = \frac{S_{i+\beta} - S_{i+\alpha}}{\beta - \alpha} \tag{14}$$

will be a first-order accurate approximation to $g'(x_c)$, where $x_c = \frac{1}{2}(x_{i+1} - x_i)$, as shown in (18) below. (This is Theorem 23 in [24].) It then follows that the piecewise linear approximation

$$\tilde{g}_{ij}(x) = m_{ij}\, x + b_{ij} \tag{15}$$

to the portion of the interface $g(x)$ in C_{ij} is pointwise second-order accurate, as shown in (19) below. (This is Theorem 24 in [24].) Therefore, one should use one of the following three slopes for m_{ij} in (15):

$$m^l_{ij} = (S_i - S_{i-1}), \quad m^c_{ij} = \tfrac{1}{2}(S_{i+1} - S_{i-1}), \quad m^r_{ij} = (S_{i+1} - S_i). \tag{16}$$

Example 1. In order to see why one of the three slopes in (16) will be the best choice for m_{ij}, consider the case when the interface is a line $g(x) = m\,x + b$. In this case the 3×3 subblock of cells B_{ij} has two exact column sums as shown in Figure 3. Note that in this particular orientation of B_{ij}, g has two exact column sums; namely the sums in the first and second columns. It is easy to check that

$$\begin{aligned} m &= \frac{1}{h^2}\int_{x_{i-1}}^{x_i} (g(x) - y_{j-1})\,dx - \frac{1}{h^2}\int_{x_{i-2}}^{x_{i-1}} (g(x) - y_{j-1})\,dx \\ &= (S_i - S_{i-1}) = m^l_{ij}. \end{aligned}$$

[7] In [24] an *exact column sum* was mistakenly defined as

$$S_i \equiv \sum_{j'=j-1}^{j+1} \Lambda_{ij'} = \frac{1}{h^2}\int_{x_i}^{x_{i+1}} (g(x) - y_{j-1}h)\,dx;$$

that is, $y_{j-1}h$ appears in the integrand, rather than just y_{j-1}. The correct definition appears here.

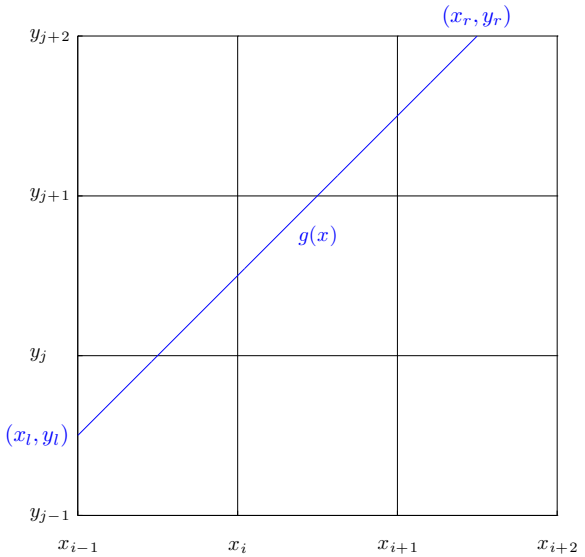


Figure 3. Here the interface g is a line $g(x) = m\,x + b$ that has two exact column sums; namely the sums in the first and second columns. In this case the slope m_{ij}^l from (16) is *exactly* equal to the slope m of the interface: $m_{ij}^l = m$. It is always the case that when the exact interface is a line on a grid of square cells one can find an orientation of the 3×3 block of cells B_{ij} such that at least one of the divided differences of the column sums in (16) is exact.

In this example the divided difference m_{ij}^l of the column sums S_{i-1} and S_i is *exactly* equal to the slope m of the exact interface. In fact, it is *always* the case that when the exact interface is a line and the grid consists of square cells, one can find an orientation of the 3×3 subblock of cells B_{ij} such that at least one of the divided differences of the column sums in (16) is exact. For example, note that in the case shown in Figure 3 one could rotate the 3×3 block of cells 90 degrees clockwise and then the correct slope to use when forming the piecewise linear approximation $\tilde{g}_{ij}(x) = m_{ij}\,x + b_{ij}$ would be $m_{ij} = m_{ij}^r$ *in the rotated coordinate frame*. One can easily check that this choice for m_{ij} would again be exactly equal to the slope m of the exact interface (in the rotated coordinate frame).

Example 2. However, as demonstrated in Example 2 of [25], there are instances in which the interface satisfies (5) but the center column sum S_i is not exact. An example was shown in Figure 4. All of the work in [25] is devoted to showing that when the interface satisfies (5), the center column sum S_i will still be exact to

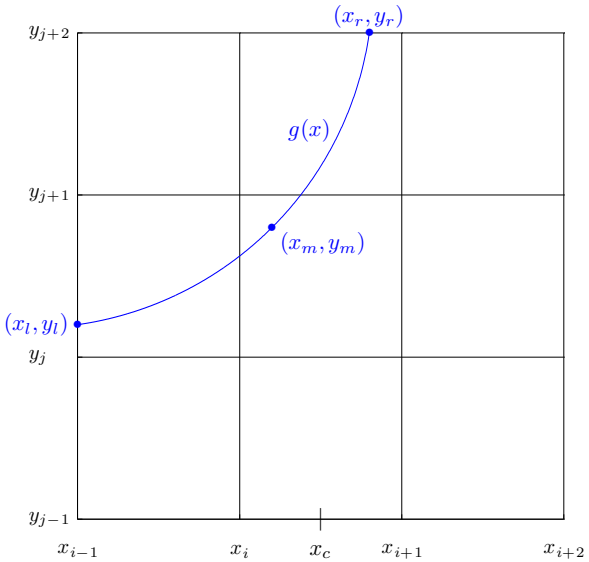


Figure 4. An example of a circular interface $g(x)$ that satisfies (5), but for which the center column sum is not exact in any of the four standard orientations of the grid. Consequently, any approximation m_{ij} to the slope $g'(x_c)$ of the form (14) must have a center column sum S_i that is not exact. (See [25, Example 2] for more details.) Theorem 4 of [25] states that if (5) is satisfied, then the error between the column sum S_i and the normalized integral of g over the center column is $O(h)$; that is, (5) implies that (17) holds. This suffices to ensure that (18) is true, and hence that (19) is true.

$O(h)$:[8]

$$\left| S_i - \frac{1}{h^2} \int_{x_i}^{x_{i+1}} (g(x) - y_{j-1})\, dx \right| \leq C_S h, \tag{17}$$

where C_S is defined in (3). This is sufficient for either the left- or the right-sided difference in (16) to satisfy

$$|m_{ij} - g'(x_c)| \leq \left(\frac{26}{3} \kappa_{\max} + C_S \right) h, \tag{18}$$

where $\kappa_{\max}$ is defined in (4). This, in turn, is sufficient for the piecewise linear volume-of-fluid approximation $\tilde{g}_{ij} = m_{ij}\, x + b_{ij}$ to still be second-order accurate

[8] In [24] the definition that the *center column sum is exact to O(h)* was mistakenly defined as

$$\left| S_i - \frac{1}{h^2} \int_{x_i}^{x_{i+1}} (g(x) - y_{j-1} h) dx \right| \leq C_S h,$$

that is, $y_{j-1} h$ appears in the integrand, rather than just y_{j-1}. The correct definition appears here.

in the max norm:

$$|\, g(x) - \tilde{g}_{ij}(x)| \leq \left(\frac{50}{3}\kappa_{\max} + C_S\right)h^2 \quad \text{for all } x \in [x_i, x_{i+1}]. \tag{19}$$

See Section 4 of [24] for proofs of (18) and (19).

To summarize, once the 5×5 block of cells $\tilde{B}_{ij}$ centered on the cell C_{ij} in which one wishes to reconstruct the interface has been rotated as described in Section 3, the interface reconstruction algorithm is based on choosing the slope m_{ij} of the piecewise linear approximation $\tilde{g}_{ij} = m_{ij}\, x + b_{ij}$ to the interface $g(x)$ to be one of the three divided differences of the column sums S_{i-1}, S_i and S_{i+1} from the 3×3 subblock B_{ij} centered on the cell C_{ij} as shown in (16). The best choice is when both column sums are exact, which — provided that the condition in (5) is satisfied — *is true in all but one case.*

This one case is the one in which the interface g satisfies (5) yet exits the i-th column S_i across its top edge as shown in Figure 4. (In this particular case the interface is *always monotonically increasing*.) Example 2 of [25] demonstrates that this case can occur for any value of h, no matter how small. However, in [25] it is proven that when this case occurs, one of the two divided differences of column sums, m^l_{ij} or m^r_{ij}, will still satisfy (18). Thus, choosing this quantity for the slope m_{ij} in $\tilde{g}_{ij} = m_{ij}\, x + b_{ij}$ still yields a pointwise second-order accurate approximation to g; that is, the bound in (19) remains true. The following section contains a description of an algorithm for determining which of these cases is present, and hence which of the slopes in (16) — the first or the third — will yield a second-order accurate approximation to the interface in C_{ij}.

5. A description of the algorithm

Before proceeding one should note that there are a variety of ways to implement this algorithm. In particular, one can implement it so that it is not necessary rotate the 5×5 block $\tilde{B}_{ij}$. The description given here was chosen because it seems to be the easiest one to follow. Furthermore this is the way in which the algorithm was implemented in order to produce the computational results shown in Section 6.

There are two steps involved in computing the approximation $\tilde{g}_{ij}(x)$ to the interface $g(x)$ in a given cell C_{ij}.

I. Determine the slope m_{ij} of the piecewise linear approximation

$$\tilde{g}_{ij}(x) = m_{ij}\, x + b_{ij}$$

to the interface $g(x)$ in the cell C_{ij}.

II. Determine the y-intercept b_{ij} of $\tilde{g}_{ij}(x)$.

Step I. Given that the 5×5 block $\tilde{B}_{ij}$ has been placed in the configuration described in Section 3 above, one need only consider the five cases listed below; namely Cases 1(a), 1(b) and Cases 2–4. From the discussion on column sums in Section 4 it is apparent that one needs two column sums that are either exact, or a left (resp., right) column sum that is exact and a center column sum that is exact to $O(h)$ as shown, for example, in Figure 4 (page 210).

The constraint in (5) on the interface $z(s)$ ensures that, once the 5×5 block of cells $\tilde{B}_{ij}$ has been rotated as described above, only the following cases can occur:

(1) The center column sum S_i is *exact to* $O(h)$, and hence satisfies (17), as shown, for example, in Figure 4, and one of the following two cases hold:

   (a) The left column sum S_{i-1} is exact, in which case one uses the left-sided divided difference:

$$m_{ij} = m_{ij}^{l} = S_i - S_{i-1}. \tag{20}$$

   (b) The right column sum S_{i-1} is exact, in which case one uses the right-sided divided difference:

$$m_{ij} = m_{ij}^{r} = S_{i+1} - S_i. \tag{21}$$

(2) All three column sums S_{i-1}, S_i and S_{i+1} are exact as shown, for example, in Figure 1. In this case one uses the centered difference of the left and right column sums:

$$m_{ij} = m_{ij}^{c} = \tfrac{1}{2}(S_{i+1} - S_{i-1}). \tag{22}$$

(3) The left column sum S_{i-1} and center column sum S_i are exact. In this case one uses the left-sided divided difference:

$$m_{ij} = m_{ij}^{l} = S_i - S_{i-1}. \tag{23}$$

(4) The center column S_i and right column sum S_{i+1} are exact. In this case one uses the right-sided divided difference:

$$m_{ij} = m_{ij}^{r} = S_{i+1} - S_i. \tag{24}$$

The various theorems and lemmas in [24; 25] prove that in each of the above cases the formulas in (20)–(24) result in a first-order accurate approximation m_{ij} to the first derivative $g'(x_c)$ of the interface at the point $x_c = \frac{1}{2}(x_i + x_{i+1})$, as shown in Equation (18).

Next is a description of the algorithm that one uses to obtain the correct slope given only the volume fraction information in the 3×5 block of cells $\hat{B}_{ij}$.

The algorithm to choose the slopes. Once the 5×5 block $\tilde{B}_{ij}$ has been rotated as described in Section 3, one only uses the 3×5 portion $\hat{B}_{ij}$ of $\tilde{B}_{ij}$ to make the decision as to which of the cases listed above one uses for that particular cell C_{ij}. The algorithm for selecting the slope is as follows.

Case 1: (The center column sum S_i is not exact, but is exact to $O(h)$.) First one checks the cell C_{ij+2}. If $\Lambda_{ij+2} > 0$, then the cell *above* the center column contains some material 1 and hence the center column sum S_i is not exact. However, by Theorem 4 of [25] the condition on h in (5) ensures that S_i is exact to $O(h)$. Therefore, one next checks the left column sum S_{i-1} and right column sum S_{i+1} to determine which column sum is exact, and hence which difference one will use; that is, Case 1(a) or 1(b) from the list above. (The constraint in (5) will ensure that the column sums S_{i-1} and S_{i+1} are not both exact, but one of them will be.)

Case 1(a): If $\Lambda_{i-1,j+2} = 0$, the left column sum S_{i-1} is exact, and hence one uses the left-sided difference:

$$m_{ij} = m_{ij}^l = S_i - S_{i-1}.$$

Case 1(b): Otherwise, it must be the case that $\Lambda_{i+1,j+2} = 0$, and hence the right column sum is exact. Therefore, one uses the right-sided difference:

$$m_{ij} = m_{ij}^r = S_{i+1} - S_i.$$

Case 2: (The center column sum must be exact.) Otherwise, $\Lambda_{i,j+2} = 0$, and hence the center column sum S_i is exact. In this case one first checks to see which of the left and right column sums are exact. If both are exact, then one uses a centered difference. Otherwise one uses a one-sided difference with the center column and whichever of the left or right column sums is exact.

Case 2(a): If $\Lambda_{i-1,j+2} = 0$ and $\Lambda_{i+1,j+2} = 0$, then both the left column sum S_{i-1} and the right column sum S_{i+1} are exact. Therefore, one uses a centered difference, since it is one order more accurate than a one sided difference:

$$m_{ij} = m_{ij}^c = \tfrac{1}{2}(S_{i+1} - S_{i-1}).$$

Case 2(b): If only $\Lambda_{i-1,j+2} = 0$, and hence the right column sum S_{i+1} is not exact, then one uses the left-sided difference:

$$m_{ij} = m_{ij}^l = S_i - S_{i-1}.$$

Case 2(c): Otherwise, if only $\Lambda_{i+1,j+2} = 0$, and hence the left column sum S_{i-1} is not exact, then one uses the right-sided difference:

$$m_{ij} = m_{ij}^r = S_{i+1} - S_i.$$

Step II. Once the slope m_{ij} has been found the constraint

$$\tilde{\Lambda}_{ij}(\tilde{g}) = \Lambda_{ij}(g),$$

where $\tilde{\Lambda}_{ij}(\tilde{g})$ denotes the fraction of material 1 in the ij-th cell due to $\tilde{g}$, immediately determines b_{ij}. In other words, once the 5×5 block of cells $\tilde{B}_{ij}$ has been appropriately rotated, b_{ij} is a single valued function of $\tilde{\Lambda}_{ij}(\tilde{g})$ and m_{ij}:

$$b_{ij} = b_{ij}(\tilde{\Lambda}_{ij}(\tilde{g}), m_{ij}).$$

There are a variety of formulas one can employ to determine b_{ij} given m_{ij}. For example, there is an approach that is based on representing the boundary of the portion of the cell C_{ij} that contains material 1 by directed line segments and using the divergence theorem to compute the volume fraction Λ_{ij} developed by S. G. Roberts and used in [26]. There is the approach developed by J. S. Saltzman and used in [23] that is based on employing a coordinate system in which the approximate interface $\tilde{g}_{ij}(x)$ is given by

$$n_{ij}^x\, x + n_{ij}^y\, y = \sigma,$$

where $\boldsymbol{n}_{ij} = (n_{ij}^x\,, n_{ij}^y)$ is the unit normal to $\tilde{g}_{ij}(x)$ that points away from the material 1 and σ is the distance from $\tilde{g}_{ij}$ to (x_i, y_j), the lower left hand corner of the cell C_{ij}. In work with Kothe et al. [8; 10; 32; 11], M. W. Williams developed algorithms for working on three-dimensional hexahedral and other unstructured meshes. Details of this work may also be found in Williams' Ph.D. thesis [33]. Finally, an article by Scardovelli and Zaleski [29] describes a collection of formulas one may use on two and three dimensional rectangular grids.

The algorithm that was used to compute the computational example shown in Section 6 of this article is based on determining which of the three polygons the approximate interface $\tilde{g}_{ij}(x)$ forms when it passes through the cell C_{ij}:

(1) a triangle,

(2) the complement of a triangle in a square, and

(3) a trapezoid.

In other words, material 1 is contained in a region that has the shape of one of the three polygons listed above.

Given the volume fraction Λ_{ij} — and hence the volume[9] V_{ij} of material 1 in the ij-th cell — and the slope m_{ij}, one can write down algebraic formulas for each of these polygons. In this way the polygon with the correct volume is readily identified, and with it the point of intersection of the approximate interface $\tilde{g}_{ij}(x)$ with two of

[9] As previously noted, strictly speaking one is given the *area* of material 1 in the ij-th cell. By convention this area is referred to as *the volume* of material 1 in the ij-th cell.

the four grid lines: $x = x_i$, $x = x_{i+1}$, $y = y_j$ and $y = y_{j+1}$ that form the edges of the cell C_{ij}. Given this information, the y-intercept b_{ij} is easily found.

Note that several of the other algorithms for representing the approximate interface listed above allow one to design the method so that it is not necessary to rotate the 5×5 block of cells $\tilde{B}_{ij}$. However, the implementation of such an algorithm may be more complex than the one described here.

6. A computational example

Table 1 contains the max norm error from a computation in which the interface reconstruction algorithm described in this paper is used to approximate $\cos x$ for $0 \le x \le \pi$ on square grids with cell sizes varying from 32^{-1} to 4096^{-1}. The error reported in the table is computed according to the formula

$$l^\infty \text{error} = \max_{C_{ij}} \left\{ \max_{x \in [x_i, x_{i+1}]} \left| \, g(x) - \tilde{g}_{ij}(x) \, \right| \right\}, \tag{25}$$

where $\max_{x \in [x_i, x_{i+1}]} \left| \, g(x) - \tilde{g}_{ij}(x) \, \right|$ is computed at 1000 points between the endpoints x_l (resp., x_r) at which the curve $g(x)$ enters (resp., exits) the cell C_{ij} and the outer maximum in (25) is taken over all cells C_{ij} that satisfy $0 < \Lambda_{ij} < 1$.

The third column contains the error (25) for the cell size reported in the second column. The fourth column contains the theoretical error bound from [25], which is quoted in Equation (2) (and also in Equation (26) below). Note that for all values of Δx the actual error is two orders of magnitude *less than* the theoretical error bound.

The last column of Table 1 contains the *convergence rate*. For a particular value of $\Delta x = 2^{-k}$, the convergence rate is defined to be the rate at which the error would have to decrease in going from $\Delta x = 2^{-(k-1)}$ to $\Delta x = 2^{-k}$ in order to achieve the

k	cell size $\Delta x = 2^{-k}$	l^∞ error	theoretical error bound	convergence rate
05	32^{-1}	$1.07 \cdot 10^{-4}$	$8.43 \cdot 10^{-2}$	2.19
06	64^{-1}	$2.67 \cdot 10^{-5}$	$2.11 \cdot 10^{-2}$	2.00
07	128^{-1}	$7.26 \cdot 10^{-6}$	$5.27 \cdot 10^{-3}$	1.88
08	256^{-1}	$1.80 \cdot 10^{-6}$	$1.32 \cdot 10^{-3}$	2.02
09	512^{-1}	$4.45 \cdot 10^{-7}$	$3.29 \cdot 10^{-4}$	2.01
10	1024^{-1}	$1.12 \cdot 10^{-7}$	$5.95 \cdot 10^{-5}$	1.99
11	2048^{-1}	$2.83 \cdot 10^{-8}$	$2.06 \cdot 10^{-5}$	1.99
12	4096^{-1}	$7.13 \cdot 10^{-9}$	$5.14 \cdot 10^{-6}$	1.99

Table 1. The max norm of the error from the computation of $\cos x$ for $0 \le x \le \pi$.

error that is shown for $\Delta x = 2^{-k}$. In other words,

$$\text{convergence rate in the row with } \Delta x = 2^{-k} \;:=\; \log_2\left(\frac{\text{error}(2^{-(k-1)})}{\text{error}(2^{-k})}\right),$$

where error(2^{-k}) denotes the error in the max norm when $\Delta x = 2^{-k}$. It is apparent that the error in the max norm decreases at a rate commensurate with a method that is second-order accurate in the max norm as claimed.

7. Conclusions

The main result of [24; 25] is a proof that

$$|g(x) - \tilde{g}_{ij}(x)| \le \left(\frac{50}{3}\kappa_{\max} + C_S\right)h^2 \quad \text{for all } x \in [x_i, x_{i+1}], \tag{26}$$

where $\tilde{g}_{ij}(x)$ is the volume-of-fluid approximation to the interface $g(x)$ in the cell C_{ij} that is described in this paper, $\kappa_{\max}$ is the maximum curvature of the interface as defined in (4), x_i and x_{i+1} denote the left and right edges respectively of the cell C_{ij}, h is the length of each side of the square grid cell C_{ij} and

$$C_S = \frac{\sqrt{3}}{2}\left\{(2\sqrt{2}-1)4\sqrt{\kappa_{\max}} + \left(1 - \frac{7\,(1+\sqrt{2})}{20}\right)\frac{32}{3}\,(\sqrt{\kappa_{\max}})^3\right\}^2.$$

The bound in (26) holds whenever the grid size h and the maximum value

$$\kappa_{\max} = \max_s |\kappa(s)|$$

of the curvature $\kappa(s)$ of the interface $z(s)$ in the 3×3 block of cells B_{ij} satisfies

$$h \le \frac{C_h}{\kappa_{\max}}, \quad \text{where } C_h = \frac{1}{25}. \tag{27}$$

However, in [24; 25] there are no examples of algorithms for finding the volume-of-fluid approximation $\tilde{g}_{ij}(x)$ in each cell C_{ij} which contains a portion of the interface. This article contains a description of one such algorithm. This algorithm is new and has not appeared previously in the scientific literature. As shown in Table 1 the computations to approximate $\cos x$ on the interval $[0, \pi]$ shown in Section 6 are consistent with the theoretical error bounds in [24; 25]. In other words, the computational approximation of $\cos x$ made with this new algorithm is consistent with the claim that it is second order accurate in the max norm provided that the interface $g \in C^2$ and (27) is satisfied.

It may be possible for the interface to pass through the center cell, then exit the 3×3 block B_{ij}, wander around the computational domain, and reenter the 3×3 block B_{ij} and the center cell C_{ij} again. For the purposes of this paper it is assumed that this does not happen. When implementing the algorithm described in this paper, one can design the code to automatically check for such cases and flag the 3×3

block B_{ij} for grid refinement, so that no cell contains two instances of the interface in a configuration such as the one just described.

In [21] J. E. Pilliod and Puckett described two volume-of-fluid interface reconstruction algorithms they had developed, and which they named the *Least Squares Volume-of-Fluid Interface Reconstruction Algorithm* (LVIRA) and the *Efficient Least Squares Volume-of-Fluid Interface Reconstruction Algorithm* (ELVIRA). They then presented computations with these algorithms on both C^2 and C^0 interfaces. When the underlying exact solution was a circle, the LVIRA and ELVIRA algorithms were shown to be second-order accurate in the max norm.

In all of the other computations they computed the errors in the *averaged* l^1 norm; that is, the l^1 norm averaged over 1000 random perturbations of the problem. For example, the l^1 error reported in approximating a circle was an average of the errors obtained when approximating 1000 different unit circles in which the center of the circle was chosen at random. They then compared the errors with errors obtained when they used several other widely used volume-of-fluid interface reconstruction algorithms such as SLIC [19] and the method developed by Parker and Youngs [20]. The only other algorithm that was close to being second-order accurate in the averaged l^1 norm consisted of taking the centered difference for the slope — that is, $m_{ij} = m^c_{ij}$, where m^c_{ij} is defined in (16).

It is not clear whether the proofs in [24; 25] apply to the LVIRA and ELVIRA algorithms. Hence, it is not clear whether LVIRA and ELVIRA are second-order accurate in the max norm, or in the l^1 and l^2 norms when the errors are not averaged over many computations. Future work should include a study of this issue and a direct comparison between the algorithm presented here and the LVIRA and ELVIRA algorithms.

One should also note that both the LVIRA and ELVIRA algorithms, as well as the algorithm presented here, reconstruct lines *exactly*. It is an open problem to prove whether or not this is a sufficient condition for the algorithm to be second-order accurate when the underlying interface is C^2.

Corollary 22 in [24] states that the algorithm presented in this paper with slope $m_{ij} = 0$ (i.e., the piecewise constant or "stair-step" volume-of-fluid interface reconstruction algorithm) will be first-order accurate whenever the interface is C^1 rather than C^2. (See footnote 10 in [24] regarding how smooth the interface must be in order to prove Corollary 22.) Future work should include an exploration of how well the algorithm presented here approximates interfaces that are less than C^2.

Finally, when the interface reconstruction algorithm is coupled to an adaptive mesh refinement algorithm, the parameter

$$H_{\max} = C_h(\kappa_{\max})^{-1},$$

where $\kappa_{\max}$ is the maximum curvature of the interface over the 3×3 block of cells

B_{ij} centered on a given cell C_{ij}, can be used to develop a criterion for determining when to increase the resolution of the grid. Namely, the computation of the interface in C_{ij} is *under-resolved* whenever

$$h > H_{\max},$$

and hence the grid needs to be refined in a neighborhood of the block B_{ij}.

References

[1] I. Aleinov and E. G. Puckett, *Computing surface tension with high-order kernels*, Proceedings of the 6th International Symposium on Computational Fluid Dynamics (Lake Tahoe, CA) (K. Oshima, ed.), 1995, pp. 6–13.

[2] J. U. Brackbill, D. B. Kothe, and C. Zemach, *A continuum method for modeling surface tension*, J. Comput. Phys. **100** (1992), no. 2, 335–354. MR 93c:76008 Zbl 0775.76110

[3] A. J. Chorin, *Flame advection and propagation algorithms*, J. Comput. Phys. **35** (1980), no. 1, 1–11. MR 81d:76061 Zbl 0425.76086

[4] _____, *Curvature and solidification*, J. Comput. Phys. **57** (1985), no. 3, 472–490. MR 86d:80001 Zbl 0555.65085

[5] L. F. Henderson, P. Colella, and E. G. Puckett, *On the refraction of shock waves at a slow-fast gas interface*, J. Fluid Mech. **224** (1991), 1–27.

[6] C. W. Hirt and B. D. Nichols, *Volume of fluid (VOF) method for the dynamics of free boundaries*, J. Comput. Phys. **39** (1981), 201–225.

[7] R. M. Hurst, *Numerical approximations to the curvature and normal of a smooth interface using high-order kernels*, MS Thesis, Department of Mathematics, University of California, Davis, December 1995, Shields Library Special Collections LD781.D5j 1995 H873.

[8] D. R. Korzekwa, D. B. Kothe, K. L. Lam, E. G. Puckett, P. K. Tubesing, and M. W. Williams, *A second-order accurate, linearity-preserving volume tracking algorithm for free surface flows on 3-d unstructured meshes*, Proceedings of the 3rd ASME /JSME Joint Fluids Engineering Conference (San Francisco, CA), FEDSM99-7109, American Society of Mechanical Engineers, 1999.

[9] D. B. Kothe, J. R. Baumgardner, S. T. Bennion, J. H. Cerutti, B. J. Daly, K. S. Holian, E. M. Kober, S. J. Mosso, J. W. Painter, R. D. Smith, and M. D. Torrey, *PAGOSA: A massively-parallel, multi-material hydro-dynamics model for three-dimensional high-speed flow and high-rate deformation*, Technical Report LA-UR-92-4306, Los Alamos National Laboratory, 1992.

[10] D. B. Kothe, E. G. Puckett, and M. W. Williams, *Approximating interface topologies with applications to interface tracking algorithms*, Proceedings of the 37th AIAA Aerospace Sciences Meetings (Reno, NV), American Institute of Aeronautics and Astronautics, 1999, pp. 1–9.

[11] _____, *Robust finite volume modeling of 3-d free surface flows on unstructured meshes*, Proceedings of the 14th AIAA Computational Fluid Dynamics Conference (Norfolk, VA), American Institute of Aeronautics and Astronautics, 1999, pp. 1–6.

[12] P. Lax and B. Wendroff, *Systems of conservation laws*, Comm. Pure Appl. Math. **13** (1960), 217–237. MR 22 #11523 Zbl 0152.44802

[13] R. J. LeVeque, *Numerical methods for conservation laws*, Lectures in Math. ETH Zürich, Birkhäuser, Basel, 1990. MR 91j:65142 Zbl 0723.65067

[14] 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

[15] G. H. Miller and E. G. Puckett, *Edge effects in molybdenum-encapsulated molten silicate shock wave targets*, J. Appl. Phys. **75** (1994), no. 3, 1426–1434.

[16] ———, *A high-order Godunov method for multiple condensed phases*, J. Comput. Phys. **128** (1996), no. 1, 134–164.

[17] B. D. Nichols, C. W. Hirt, and R. S. Hotchkiss, *SOLA-VOF: A solution algorithm for transient fluid flow with multiple free boundaries*, Technical Report LA-8355, Los Alamos National Laboratory, August 1980.

[18] W. F. Noh and P. R. Woodward, *SLIC* (*Simple Line Interface Calculation*), Technical Report UCRL-77651, Lawrence Livermore National Laboratory, August 23 1976.

[19] ———, *SLIC (Simple Line Interface Calculation)*, Lecture Notes in Physics (A. I. van der Vooren and P. J. Zandbergen, eds.), vol. 59, Springer, New York, 1976, pp. 330–340.

[20] B. J. Parker and D. L. Youngs, *Two and three dimensional Eulerian simulation of fluid flow with material interfaces*, Technical Report 01/92, UK Atomic Weapons Establishment, Aldermaston, Berkshire, Feb 1992.

[21] J. E. Pilliod, Jr. and E. G. Puckett, *Second-order accurate volume-of-fluid algorithms for tracking material interfaces*, J. Comput. Phys. **199** (2004), no. 2, 465–502. MR 2005d:65145 Zbl 1126.76347

[22] S. Popinet and S. Zaleski, *A front-tracking algorithm for accurate representation of surface tension*, Int. J. for Numer. Methods in Fluids **30** (1999), no. 6, 775–793. Zbl 0940.76047

[23] E. G. Puckett and J. S. Saltzman, *A 3D adaptive mesh refinement algorithm for multimaterial gas dynamics*, Phys. D **60** (1992), no. 1-4, 84–93. MR 93i:76062 Zbl 0779.76059

[24] E. G. Puckett, *On the second-order accuracy of volume-of-fluid interface reconstruction algorithms: Convergence in the max norm*, Commun. Appl. Math. Comput. Sci. **5** (2010), 99–148. MR 2600824 Zbl 05709095

[25] E. G. Puckett, *On the second-order accuracy of volume-of-fluid interface reconstruction algorithms II: An improved constraint on the cell size*, CAMCoS (2010), Submitted for publication.

[26] E. G. Puckett and S. G. Roberts, *A volume-of-fluid method for modeling flow by mean curvature*, 1990, work performed at the Australian National University.

[27] W. J. Rider and D. B. Kothe, *Reconstructing volume tracking*, J. Comput. Phys. **141** (1998), no. 2, 112–152. MR 99a:65200 Zbl 0933.76069

[28] R. Scardovelli and S. Zaleski, *Direct numerical simulation of free-surface and interfacial flow*, Annual review of fluid mechanics (C. Cambon and J. F. Scott, eds.), vol. 31, Annual Reviews, Palo Alto, CA, 1999, pp. 567–603. MR 99m:76002

[29] R. Scardovelli and S. Zaleski, *Analytical relations connecting linear interfaces and volume fractions in rectangular grids*, J. Comput. Phys. **164** (2000), no. 1, 228–237. MR 1786246 Zbl 0993.76067

[30] M. D. Torrey, L. D. Cloutman, R. C. Mjolsness, and C. W. Hirt, *NASA-VOF2D: A computer program for incompressible flows with free surfaces*, Technical Report LA-10612-MS, Los Alamos National Laboratory, December 1985.

[31] M. D. Torrey, R. C. Mjolsness, and L. R. Stein, *NASA-VOF3D: A three-dimensonal computer program for incompressible flows with free surfaces*, Technical Report LA-11009-MS, Los Alamos National Laboratory, July 1987.

[32] M. W. Williams, D. B. Kothe, and E. G. Puckett, *Accuracy and convergence of continuum surface-tension models*, Fluid dynamics at interfaces (W. Shyy, ed.), Cambridge Univ. Press, 1999, pp. 294–305. MR 1719592 Zbl 0979.76014

[33] M. W. Williams, *Numerical methods for tracking interfaces with surface tension in 3-d mold-filling processes*, Ph.D. thesis, University of California, Davis, 2000.

Received February 23, 2010.

ELBRIDGE GERRY PUCKETT: egpuckett@ucdavis.edu
University of California, Davis, Department of Mathematics, One Shields Avenue, Davis, CA 95616, United States
http://www.math.ucdavis.edu/

COMM. APP. MATH. AND COMP. SCI.
Vol. 5, No. 2, 2010

IMPLICIT PARTICLE FILTERS FOR DATA ASSIMILATION

ALEXANDRE CHORIN, MATTHIAS MORZFELD AND XUEMIN TU

Implicit particle filters for data assimilation update the particles by first choosing probabilities and then looking for particle locations that assume them, guiding the particles one by one to the high probability domain. We provide a detailed description of these filters, with illustrative examples, together with new, more general, methods for solving the algebraic equations and with a new algorithm for parameter identification.

1. Introduction

There are many problems in science, for example in meteorology and economics, in which the state of a system must be identified from an uncertain equation supplemented by noisy data (see, for instance, [9; 22]). A natural model of this situation consists of an Ito stochastic differential equation (SDE):

$$dx = f(x,t)\,dt + g(x,t)\,dw, \tag{1}$$

where $x = (x_1, x_2, \ldots, x_m)$ is an m-dimensional vector, f is an m-dimensional vector function, $g(x,t)$ is an m by m matrix, and w is Brownian motion which encapsulates all the uncertainty in the model. In the present paper we assume for simplicity that the matrix $g(x,t)$ is diagonal. The initial state $x(0)$ is given and may be random as well.

The SDE is supplemented by measurements b^n at times t^n, $n = 0, 1, \ldots$. The measurements are related to the state $x(t)$ by

$$b^n = h(x^n) + GW^n, \tag{2}$$

where h is a k-dimensional, generally nonlinear, vector function with $k \le m$, G is a matrix, $x^n = x(t^n)$, and W^n is a vector whose components are independent Gaussian variables of mean 0 and variance 1, independent also of the Brownian motion in (1). The independence requirements can be greatly relaxed but will be

MSC2000: 60G35, 62M20.

Keywords: implicit sampling, data assimilation, particle filter.

This work was supported in part by the Director, Office of Science, Computational and Technology Research, United States Department of Energy under Contract no. DE-AC02-05CH11231, and by the National Science Foundation under grants DMS-0705910 and OCE-0934298.

observed in the present paper. The task of a filter is to assimilate the data, that is, estimate x on the basis of both (1) and the observations (2).

If the system (1) and the function h in (2) are linear and the data are Gaussian, the solution can be found in principle via the Kalman–Bucy filter [19]. In the general case, one often estimates x as a statistic (often the mean) of a probability density function (pdf) evolving under the combined effect of (1) and (2). The initial state x^0 being known, all one has to do is evaluate sequentially the pdfs P_{n+1} of the variables x^{n+1} given the equations and the data. In a "particle" filter this is done by following "particles" (replicas of the system) whose empirical distribution at time t^n approximates P_n. One may for example [1; 9; 7; 8; 19] use the pdf P_n and (1) to generate a prior density (in the sense of Bayes) , and then use the data b^{n+1} to generate sampling weights which define a posterior density P_{n+1}. This can be very expensive because in most weighting schemes, most of the weights tend to zero fast and the number of particles needed can grow catastrophically [21; 2]; various strategies have been proposed to ameliorate this problem.

Our remedy is implicit sampling [4; 5]. The number of particles needed in a filter remains moderate if one can find high probability particles; to this end, implicit sampling works by first picking probabilities and then looking for particles that assume them, so that the particles are guided efficiently to the high probability region one at a time, without needing a global guess of the target density. In the present paper we provide an expository account of particle filters, separating clearly the general principles from details of implementation; we provide general solution algorithms for the resulting algebraic equations, in particular for nonconvex cases which we had not considered in our previous publications, as well as a new algorithm for parameter identification based on an implicit filter. We also provide examples, in particular of nonconvex problems.

Implicit filters are a special case of chainless sampling methods [3]; a key connection was made in [23; 24], where it was observed that in the sampling of stochastic differential equations, the marginals needed in Markov field sampling can be read off the equations and need not be estimated numerically.

2. The mathematical framework

The first thing we do is discretize the SDE (1) by a difference scheme, so that the equation becomes a discrete recurrence, and assume temporarily that the time step in the dynamics equals the fixed time δ between observations. For simplicity, in this section we assume the scheme is the Euler scheme

$$x^{n+1} = x^n + \delta f(x^n, n\delta) + V, \tag{3}$$

where V is a Gaussian of mean zero and variance $g^2(x^n, n\delta)\delta$. Higher-order schemes

are discussed in Section 4.

The conditional probability densities $P_n(x)$ at times t^n, determined by the discretized SDE (3) given the observations (2), satisfy the recurrence relation [9, p. 6]

$$P(x^{n+1}) = P(x^n)P(x^{n+1}|x^n)P(b^{n+1}|x^{n+1})/Z, \tag{4}$$

where $P(x^n) = P(x^n|b^1, b^2, \ldots, b^n)$ is the probability density at time $n\delta$ given the data up to time $n\delta$, $P(x^{n+1}|x^n)$ is the probability density of x^{n+1} given x^n as it is determined by the dynamics, $P(b^{n+1}|x^{n+1})$ is the probability of the next observation given the new position, as per the observation equation, and Z is a normalization constant.

We estimate P_{n+1} with the help of M particles, with positions X_i^n at time t^n and X_i^{n+1} at time t^{n+1} ($i = 1, \ldots, M$), which define empirical densities $\hat{P}_n$, $\hat{P}_{n+1}$ that approximate P_n, P_{n+1}. We do this by requiring that, when a particle moves from X_i^n to X_i^{n+1}, the probability of X_i^{n+1} given b^k for $k \le n+1$ be given by

$$P(X_i^{n+1}) = P(X_i^n)P(X_i^{n+1}|X_i^n)P(b^{n+1}|X_i^{n+1})/Z_0, \tag{5}$$

where the hats have been omitted as they will be from now on, $P(X_i^n)$, the probability of X_i^n given b^k for $k \le n$, is assumed given, $P(X_i^{n+1}|X_i^n)$, the probability of X_i^{n+1} given X_i^n, is determined by the discretized SDE (3), $P(b^{n+1}|X_i^{n+1})$, the probability of the observations b^{n+1} given the new positions X_i^{n+1}, is determined by the observation equaiton (2), and Z_0 is an unknown normalization constant. We shall see below that one can set $P(X_i^n) = 1$ without loss of generality.

Equation (5) defines the pdf we now need to sample for each particle. One way to do this is to pick a position X_i^{n+1} according to some prior guess of P_{n+1}, and then use weights to get the resulting pdf to agree with the true P_{n+1} (the "posterior" density); in general many of the new positions will have low probability and therefore small weights. The idea in implicit sampling is to define probabilities first, and then look for particles that assume them; this is done by choosing once and for all a fixed reference random variable, say ξ, with a given pdf, say a Gaussian $\exp(-\xi^T\xi/2)/(2\pi)^{m/2}$, which one knows how to sample, and then making X_i^{n+1} a function of ξ, a different function of each particle and each step, each function designed so that the map $\xi \to X_i^{n+1}$ connects highly probable values of ξ to highly probable values of X_i^{n+1}. To that end, write

$$P(X_i^{n+1}|X_i^n)P(b^{n+1}|X_i^{n+1}) = \exp(-F_i(X)),$$

where on the right-hand side X is a shorthand for X_i^{n+1} and all the other arguments are omitted. This defines a function F_i for each particle i and each time t^n. For each i and n, F_i is an explicitly known function of $X = X_i^{n+1}$. Then solve the equation

$$F_i(X) - \phi_i = \xi^T\xi/2, \tag{6}$$

where ξ is a sample of the fixed reference variable and ϕ_i is an additive factor needed to make the equation solvable. The need for ϕ_i becomes obvious if one considers the case of a linear observation function h in (2), so that the right side of (6) is quadratic but the left is a quadratic plus a constant. It is clear that setting $\phi_i = \min F_i$ will do the job, but it is sometimes convenient to perturb this choice by a small amount (see below). In addition, and most important, with our choice of reference variable ξ, the most likely choice of ξ is in the neighborhood of 0; if the mapping $\xi \to X$ satisfies (6), this likely choice of ξ produces an X near the minimum of F_j, hence a high probability position for the particle. We also require that for each particle, the function $X_i^{n+1} = X = X(\xi)$ defined by (6) be one-to-one so that the correct pdf is sampled, in particular, it must have distinct branches for positive values and negative values of each component of ξ. The solution of (6) is discussed in the next section. From now on we omit the index i in both F and ϕ, but it should not be forgotten that these functions vary from particle to particle and from one time step to the next.

Once the function $X = X(\xi)$ is determined, each value of $X^{n+1} = X$ (the subscript i is omitted) appears with probability $\exp(-\xi^T\xi/2)J^{-1}/(2\pi)^{m/2}$, where J is the Jacobian of the map $X = X(\xi)$, while the product $P(X^{n+1}|X^n)P(b^{n+1}|X^{n+1})$ evaluated at X^{n+1} equals $\exp(-\xi^T\xi/2)\exp(-\phi)$. The sampling weight for the particle is therefore $\exp(-\phi)J(2\pi)^{m/2}$. If the map $\xi \to X$ is smooth near $\xi = 0$, so that ϕ and J do not vary rapidly from particle to particle, and if there is an easy way to compute J (see the next section), then we have an effective way to sample P_{n+1} given P_n. It is important to note that though the functions F and ϕ vary from particle to particle, the probabilities of the various samples are expressed in terms of the fixed reference pdf, so that they can be compared with each other.

The weights can be eliminated by resampling. A standard resampling algorithm goes as follows [9]: let the weight of the i-th particle be W_i, $i = 1, \dots, M$. Define $A = \sum W_i$; for each of M random numbers θ_k, $k = 1, \dots, M$, drawn from the uniform distribution on [0, 1], choose a new $\widehat{X}_k^{n+1} = X_i^{n+1}$ such that $A^{-1}\sum_{j=1}^{i-1} W_j < \theta_k \le A^{-1}\sum_{j=1}^{i} W_j$, and then suppress the hat. This justifies the statement following (5) that one can set $P(X_n) = 1$.

To see what has been gained, compare our construction with the usual "Bayesian" particle filter, where one samples $P(X^{n+1}|X^n)P(b^{n+1}|X^{n+1})$ by first finding a "prior" density $Q(X^{n+1})$ (omitting all arguments other than X^{n+1}), such that the ratio $W = P(X^{n+1})/Q(X^{n+1})$ is close to a constant, and then assigning to the i-th particle the importance weight $W = W_i$ evaluated at the location of the particle. The pdf defined by the set of positions and weights is the density P_{n+1} we are looking for. An important special case is the choice $Q(X^{n+1}) = P(X^{n+1}|X^n)$; the prior is then defined by the equation of motion alone and the posterior is obtained by using the observations to weight the particles. We shall refer to this special case as

"standard importance sampling" or "standard filter". Of course, once the positions and the weights of the particles have been determined, one should resample as above.

The catch in these earlier constructions is that the prior density Q and the desired posterior can become nearly mutually singular, and the number of particles needed may become catastrophically large, especially when the number of variables m is large [2; 21]. To avoid this catch one has to make a good guess for the pdf Q, which may not be easy because Q should approximate the unknown density P_{n+1} one is looking for — this is the basic conundrum of Monte Carlo methods, in which one needs a good estimate to get a good estimate. In contrast, in implicit sampling one does a separate calculation for each sample and there is no need for prior global information. One can of course still identify the pdf defined by the positions of the particles at time t^{n+1} as a "prior" and the pdf defined by both the positions and the weights as a "posterior" density.

Note that one can recover standard importance sampling within our framework by setting $\phi = -\log P(b^{n+1}|X^{n+1})$, but this choice of course violates our rule for choosing ϕ.

Finally, implicit sampling can be viewed as an implicit Monte Carlo scheme for solving the Zakai equation [25], which describes the evolution of the unnormalized conditional distribution for an SDE conditioned by observations. This should be contrasted with the procedure in the popular ensemble Kalman filter [11], where a Gaussian approximation of the pdf defined by the SDE is extracted from a Monte Carlo solution of the corresponding Fokker–Planck equation, a Gaussian approximation is made for the pdf $P(b^{n+1}|x^{n+1})$ [17], and new particle positions are obtained by a Kalman step. Our replacement of the Fokker–Planck equation that corresponds to the SDE alone by a Zakai equation that describes the evolution of the unnormalized conditional distribution does away with the need for the approximate and expensive extraction of Gaussians and consequent Kalman step.

3. Solution of the algebraic equation that defines a new sample

We now explain how to solve (6), $F(X) - \phi = \xi^T \xi/2$, under several sets of assumptions which are met in practice. This is a well-defined, deterministic, algebraic equation for each particle. Note the great latitude that it provides in linking the ξ variables to the X variables: it is a single equation that connects $2m$ variables (the m components of ξ and the m components of X) and can be satisfied by many maps $\xi \to X$ as long as (i) they are one-to-one, (ii) they map the neighborhood of 0 into a set that contains the minimum of F, (iii) they are smooth near $\xi = 0$ so that the weights $\exp(-\phi)J$ not vary unduly from particle to particle in the target area, and (iv) they allow the Jacobian J to be calculated easily. The solution methods

presented here are far from exhaustive; further examples will be presented in the context of specific applications.

Algorithm A (presented in [4; 5]). Assume the function F is convex upwards and h is not too different from a linear function. For each particle, we set up an iteration, with iterates $X^{n+1,j}$, $j = 0, 1, \ldots$, (X^j for brevity), with $X^0 = 0$, that converge to the next position X^{n+1} of that particle. The index i that identifies the particle is omitted again. We write the equations as if the system were one-dimensional; the multidimensional case was presented in detail in [5]. First we sample the reference variable ξ. The iteration is defined when one knows how to find X^{j+1} given X^j.

Expand the observation function h in (2) around X^j:

$$h(X^{j+1}) = h(X^j) + (Dh)^j (X^{j+1} - X^j), \tag{7}$$

where $(Dh)^j$ is the derivative of h evaluated at X^j. The observation function in (2) is now approximated as a linear function of X^{j+1}, and the function F is the sum of two Gaussians in X^{j+1}. Completing a square yields a single Gaussian with a remainder ϕ, i.e., $F(X) = (x - \bar{a})^2/(2\bar{v}) + \phi(X^j)$, where the parameters ϕ, $\bar{a}$, $\bar{v}$ are functions of X^j (this is what we called in [4] a "pseudo-Gaussian"). The next iterate is now $X^{j+1} = \bar{a} + \sqrt{\bar{v}}\xi$. In the multidimensional case, when each component of the function h in (2) depends on more than one variable, finding X^{j+1} as a function of ξ may require the solution of an equation of the form $(X^{j+1})^T A X^{j+1} = \xi^T \xi/2$, where A is positive definite and symmetric. This is, as expected, a single equation for several variables, so that the solution is not unique. We may choose, as in [5], to connect ξ to X^{j+1} by performing a Choleski decomposition, $A = LL^T$, where L is lower triangular, and then solving $\sqrt{2}L^T X^{j+1} = \xi$. A different connection was presented in [4]. If the iteration converges, it converges to the exact solution of (6), with ϕ the limit of the $\phi(X^j)$. Its convergence can be accelerated by Aitken's extrapolation [13]. The Jacobian J can be evaluated either by an implicit differentiation of (6) or numerically, by perturbing ξ in (6) and solving the perturbed equation (which should not require more than a single additional iteration step). It is easy to see that this iteration, when it converges, produces a mapping $\xi \to X$ that is one to one and onto.

An important special case occurs when the observation function h is linear in X and there are observations at every step (see section 5 for the case of sparse observations). It is immaterial then whether the SDE (1) is linear. In this case the iteration converges in one step; the Jacobian J is easy to find; if in addition the function $g(x, t)$ in (1) is independent of x, then J is independent of the particle and need not be evaluated; the additive term ϕ can be written explicitly as a function of the previous position X^n of the particle and of the observation b^{n+1}. We recover an easy implementation of optimal sequential importance sampling [1; 9; 8].

This iteration has been used in [5]. It may fail to converge if the function F is not convex, as happens in particular when the observation function h is highly nonlinear. In the latter case the value of ϕ it produces may also be far from the minimum of F. If h is strongly nonlinear, the next iteration is preferable.

Algorithm B. Assume the function F is U-shaped, i.e., in the scalar case, it is at least piecewise differentiable, F' vanishes at a single point which is a minimum, F is strictly decreasing on one side of the minimum and strictly increasing on the other, with $F(X) = \infty$ when $X = \pm\infty$. In the m-dimensional case, assume that F has a single minimum and that each intersection of the graph of the function $y = F(X)$ with a vertical plane through the minimum is U-shaped in the scalar sense (a function may be U-shaped without being convex).

Find z, the minimum of F (this is the minimum of a given real valued function, not a minimum of a possibly multimodal pdf generated by the SDE; finding this minimum is not equivalent to the difficult problem of finding a maximum likelihood estimate of the state of the system). The minimum z can be found by standard minimization algorithms.

Again we are solving the equations by finding iterates X^j that converge to X^{n+1}. In the scalar case, given a sample of the reference variable ξ, find first X^0 such that $X^0 - z$ has the sign of ξ, and then find the next iterates X^j by standard tools (e.g., by Newton iteration), modified so that the X^j are prevented from leaping over z.

In the vector case, if the observation function is diagonal — i.e., each component of the observation is a function of a single component of the solution X — then the scalar algorithm can be used component by component. In more complicated situations one can take advantage of the freedom in connecting ξ to X.

Here is an interesting example of the use of this freedom, which we present in the case of a multidimensional problem where the observation function is linear but need not be diagonal. Set $\phi = \min F$. The function $F(X) - \phi$ can now be written as $(X-a)^T A(X-a)/2$, where a is a known vector, T denotes a transpose as before, and A is a positive definite symmetric matrix. Write further $y = X - a$. Equation (6) becomes

$$y^T A y = |\xi|^2, \tag{8}$$

where $|\xi|$ is the length of the vector ξ. Make the ansatz

$$y = \lambda\eta,$$

where λ is a scalar, $\eta = \xi/|\xi|$ is a random unit vector and ξ is a sample of the reference density. Substitution into (8) yields

$$\lambda^2(\eta^T A \eta) = |\xi|^2. \tag{9}$$

It is easy to see that $E[\eta_i \eta_j] = \delta_{ij}/m$, where $E[\,\cdot\,]$ denotes an expected value, the η_i are the components of η, m is the number of variables, and δ_{ij} is the Kronecker delta, and hence

$$E[\eta^T A \eta] = \text{trace}(A)/m.$$

Replace (9) by

$$\lambda^2 \Lambda = |\xi|^2. \tag{10}$$

where $\Lambda = \text{trace}(A)/m$. This equation has the solution $\lambda = |\xi|/\sqrt{\Lambda}$, and substitution into the ansatz leads to $y_i = \xi_i/\sqrt{\Lambda}$, an easily implementable transformation with Jacobian $J = \Lambda^{-m/2}$. The difference between (9) and (10) can be compensated for by adding to ϕ the term $\lambda^2[(\eta^T A \eta) - \Lambda]$. Note that as $m \to \infty$, $(\eta^T A \eta) \to \Lambda$ provided A satisfies some minor requirements, so that when the number of variables is sufficiently large, the perturbation one has to compensate for becomes negligible. Detailed implementations and generalizations of this construction will be given elsewhere in the context of specific applications.

One can readily devise algorithms also for cases where F is not U-shaped, for example, by dividing F into monotonic pieces and sampling each of these pieces with its predetermined probability. An alternative that is usually easier is to replace the non-U-shaped function F by a suitable U-shaped function F_0 and make up for the bias by adding $F(X) - F_0(X)$ to the ϕ in the weights $\exp(-\phi)J$ so that (6) is still satisfied. One also has to make sure that the small ξ region is still mapped on the high probability region for X; see the examples below.

More generally, even in convex cases, one can often change F in (6) to make the algebraic problem easy without reducing the quality of the samples; examples will be given in the next two sections.

4. Examples

We now present examples that illustrate the algorithms we have just described. For more examples, see [4; 5]. For the sake of clarity, in this section we continue to rely on an Euler discretization of the SDE, as in (3).

We begin with a response to a comment we have often heard: "This is nice, but the construction will fail the moment you are faced with potentials with multiple wells". This is not so: the function F depends on the nature of the noise in the SDE and on the function $h = h(x)$ in the observation (2), but not on the potential. Consider, for example, a one dimensional particle moving in the potential $V(x) = 2.5(x^2 - 0.5)^2$ (see Figure 1) with the force $f(x) = -\nabla V = -10x(x^2 - 1)$ and the resulting SDE $dx = f(x)dt + \sigma dw$, where $\sigma = 0.1$ and w is Brownian motion with unit variance; with this choice of parameters the SDE has an invariant density concentrated in the neighborhoods of $x = \pm\sqrt{1/2}$. We consider linear observations $b^n = x(t^n) + W$, where W is a Gaussian variable with mean zero and variance $s = 0.025$. We

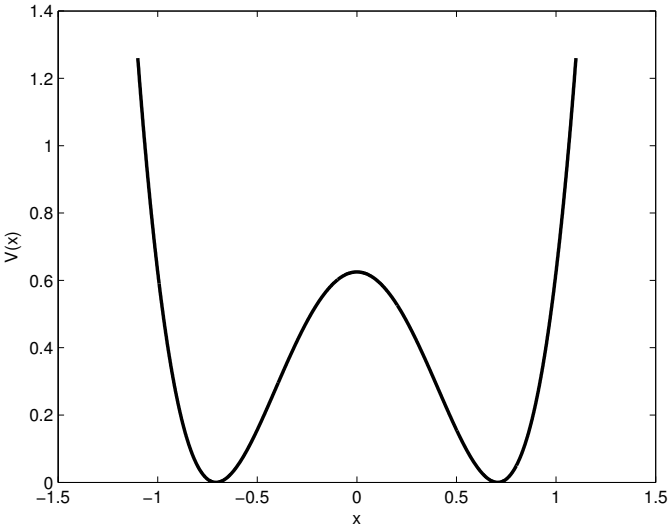


Figure 1. The potential in the first example.

approximate the SDE by an Euler scheme [16] with time step $\delta = 0.01$, and assume observations are available at all the points $n\delta$. The particles all start at $x = 0$. We produce data b^n by running a single particle and adding to its positions errors drawn from the assumed error density in (2), and then attempt to reconstruct this path with our filter. For the i-th particle located at time $n\delta$ at X_i^n the function $F(X)$ is

$$F(X) = \frac{(X - X_i^n)^2}{2\sigma\delta} + \frac{(X - b^{n+1})^2}{2s},$$

which is always convex. A completion of the square yields

$$\min F = \phi = \frac{(X_i^n - b^{n+1})^2}{2(\sigma\delta + s)};$$

the Jacobian J is independent of the particle and need not be evaluated. In Figure 2 we display a particle run used to generate data and its reconstruction by our filter with 50 particles.

This figure is included for completeness but both of these paths are random, their difference varies from realization to realization, and may be large or small by accident. To get a quantitative estimate of the performance of the filter, we repeated this calculation 10^4 times and computed the mean and the variance of the difference Δ between the run that generated the data and its reconstruction at time $t = 1$; see Table 1. This table shows that the filter is unbiased and that the variance of Δ is comparable to the variance of the error in the observations $s = 0.025$. Even with one single particle (and therefore no resampling) the results are still acceptable.

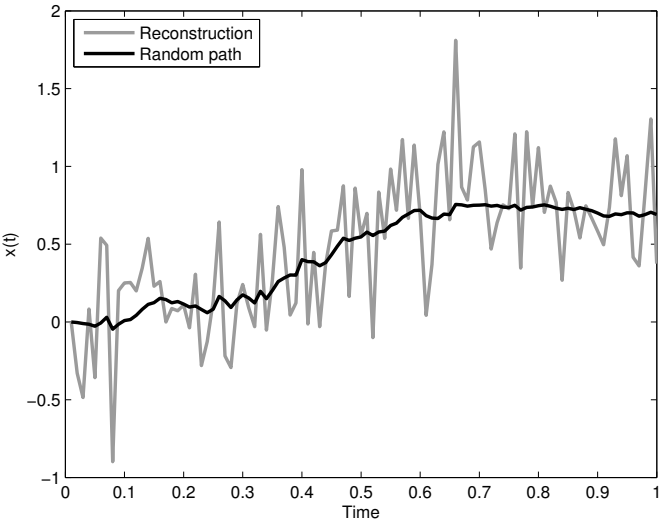


Figure 2. A random path (black) and its reconstruction by our filter (gray).

We now discuss the relation between the posterior we wish to sample and the prior in several special cases, including nonconvex situations. We want to produce samples of the pdf $P(x) = \exp(-F(x))/Z$, where Z is a normalization constant and

$$F(x) = \frac{x^2}{2\sigma} + \frac{(h(x)-b)^2}{2s}. \tag{11}$$

Here $h(x)$ is a given function of x as in (2), and σ, s, b are given parameters. This can be viewed as a first step in time for a filtering problem where all the particles start from the same point so that $\exp(-F(x))/Z = P_1$, or as an analysis of the sampling for one particular particle in a general filtering problem, or as an instance of the more general problem of sampling a given pdf when the important events may be rare. In standard Bayesian sampling one samples the variable with pdf

$$\frac{\exp(-x^2/(2\sigma))}{\sqrt{2\pi\sigma}}$$

M	mean	variance	M	mean	variance
100	−0.0001	0.021	10	0.0001	0.024
50	−0.0001	0.022	5	−0.0001	0.027
20	−0.0001	0.023	1	−0.0001	0.038

Table 1. Mean and variance of the discrepancy between the observed path and the reconstructed path in Example 1 as a function of the number of particles M, with $s = 0.025$.

and then one attaches to the sample at x the weight $\exp(-(h(x)-b)^2/(2s))$; in an implicit sampler one finds a sample x by solving $F(x)-\phi=\xi^2/2$ for a suitable ϕ and ξ and attaching to the sample the weight $\exp(-\phi)J$. For given σ, s, the problem becomes more challenging as $|b|$ increases.

In both the standard and the implicit filters one can view the empirical pdf generated by the unweighted samples as a "prior" and the one generated by the weighted samples as the "posterior". The difficulty with standard filters is that the prior and posterior densities may approach being mutually singular, so it is of interest to estimate the Radon–Nikodým derivative of one of these with respect to the other. If that derivative is a constant, we have achieved perfect importance sampling, as every neighborhood in the sample space is visited with a frequency proportional to its density. We estimate the Radon–Nikodým derivative of the prior with respect to the posterior as follows. In this simple problem one can evaluate the probability of any interval with respect to the posterior we wish to sample by quadratures. We divide the interval $[0, 1]$ into K pieces of equal lengths $1/K$, then find numerically points $Y_1, Y_2, \ldots, Y_{K-1}$, with $Y_K=+\infty$, such that the posterior probability of the interval $[-\infty, Y_k]$ is k/K for $k=1, 2, \ldots, K$. We then find $L=10^5$ samples of the prior and plot of a histogram of the frequencies with which these samples fall into the posterior equal probability intervals (Y_{k-1}, Y_k). The more this histogram departs from being a constant independent of k, the more samples are needed to calculate the statistics of the posterior.

If $h(x)$ is linear, the weights in the implicit filter are all equal and the histogram is constant for all values of b. This remains true for all values of b, i.e., however far the observation b is from what one may expect from the SDE alone. This is not the case with a standard Bayesian filter, where some parts of the sample space that have nonzero probability are visited very rarely.

In Table 2 we list the histogram of frequencies for a linear observation function $h(x)=x$ and $b=2$ in a standard Bayesian filter, with $K=10$. We used 10^4 samples; the fluctuations in the implicit case measure only the accuracy with which

k	standard	implicit	k	standard	implicit
1	0.987	0.099	6	0.003	0.099
2	0.006	0.108	7	0.001	0.101
3	0.002	0.097	8	0.001	0.101
4	0.001	0.099	9	0.000	0.102
5	0.004	0.101	10	0.000	0.093

Table 2. Histogram of the Radon–Nikodým derivative of the prior with respect to the posterior, standard Bayesian filter versus the implicit filter, 10000 particles, $b=2$, $\sigma=s=0.1$, $h(x)=x$.

b	exact	standard	implicit
0	0	−0.05	0.02
0.5	0.25	0.10	0.27
1	0.5	0.18	0.51
1.5	0.75	0.23	0.76
2	1	0.26	1.01

Table 3. Comparison of the estimates of the means, implicit vs. standard filter, 30 particles, together with the exact results, linear case, as explained in the text.

the histogram is computed with this number of samples. As a consequence, estimates obtained with the implicit filter are much more reliable than the ones obtained with the standard Bayesian filter.

In Table 3 we list the estimates of the mean position of the linear problem as a function of b, with 30 particles, $\sigma = s = 0.1$, for the standard Bayesian and the implicit filters, compared with the exact result. The standard deviations are not displayed; they are all near 0.01.

The results in this one-dimensional problem mirror the situation with the example of Bickel et al. [2; 21], designed to display the breakdown of the standard Bayesian filter when the number of dimension is large; what happens there is that one particle hogs almost the whole weight, so that the number of particles needed grows catastrophically; in contrast, the implicit filter assigns equal weights to all the particles in any number of dimensions, so that the number of particles needed is independent of dimension; see also [5].

We now turn to nonlinear and nonconvex examples. Let the observation function h be strongly nonlinear: $h(x) = x^3$. With $\sigma = s = 0.1$, the pdf (11) becomes non-U-shaped for $|b| \geq 0.77$. In Figure 3 we display the function F for $b = 1$ (the solid curve). To use the algorithms above we need a substitute function F_0 that is U-shaped; we also display in Figure 3 (the broken line) the function F_0 we used; the recipe here is to link a point above the local minimum on the left to the absolute minimum on the right by a straight line. It is important to make F_0 and F have the same minimum. Many other constructions are possible (see in particular the next section). As described above, we solve

$$F_0(x) - \phi = \xi^T \xi/2$$

with $\phi = \min F_0$, and once x has been determined, add the difference $F(x) - F_0(x)$ to ϕ in the weight $\exp(-\phi)J$. This construction does not introduce any bias. The function F_0 constructed in this way is U-shaped but need not be convex, so that one

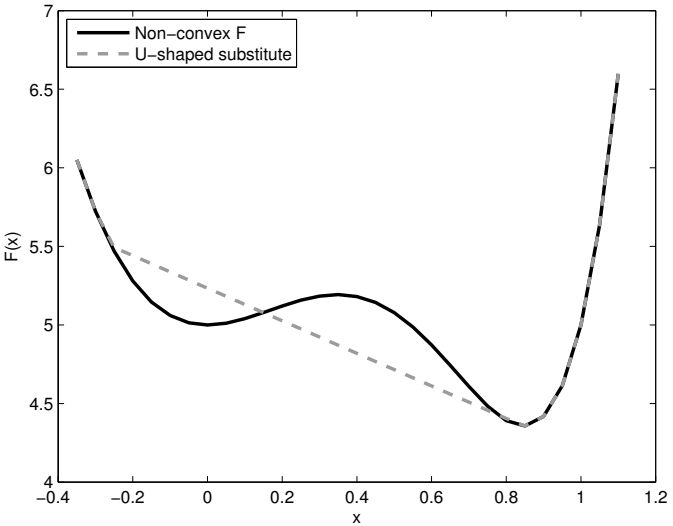


Figure 3. A nonconvex function F (solid line) and a U-shaped substitute (broken line).

needs Algorithm B above. In Table 4 we compare the Radon–Nikodým derivatives of the prior with respect to the posterior for the resulting implicit sampling and for standard Bayesian sampling with $\sigma = s = 0.1$, $b = 1.5$.

The histogram for the implicit filter is no longer perfectly balanced. The asymmetry in the histogram reflects that of F_0 and can be eliminated by biasing ξ, but there is no reason to do so; there is enough importance sampling without this extra step.

k	standard	explicit
1	0.9948	0.0899
2	0.0028	0.0537
3	0.0011	0.0502
4	0.0004	0.0563
5	0.0003	0.0696
6	0.0002	0.1860
7	0.0001	0.1107
8	0.0001	0.1194
9	0.0001	0.1196
10	0	0.1446

Table 4. Radon–Nikodým derivatives of the prior with respect to the posterior, $h(x) = x^3$, $\sigma = s = 0.1$, $b = 1.5$, 10000 samples, F_0 as in the text.

b	exact	standard	implicit
0.	0.	-0.00 ± 0.01	-0.00 ± 0.01
0.5	0.109	0.109 ± 0.01	0.109 ± 0.01
1.0	0.442	0.394 ± 0.04	0.451 ± 0.02
1.5	0.995	0.775 ± 0.09	0.995 ± 0.01
2.0	1.18	0.875 ± 0.05	1.18 ± 0.01
2.5	1.30	0.895 ± 0.02	1.29 ± 0.02

Table 5. Comparison of the estimates of the means, implicit vs. standard filter, 1000 particles, together with the exact result, when $h(x) = x^3$, as explained in the text.

In Table 5 we display the estimates of the means of the density for the two filters with 1000 particles for various values of b, compared with the exact results (the number of particles is relatively large because with $h(x) = x^3$ and our parameter choices the variance of the conditional density is significant, and this number of particles is needed for meaningful comparisons of either algorithm with the exact result).

As mentioned in the previous section, there are alternatives to the replacement of F by F_0; the point is that for each particle the function F is an explicitly known nonrandom function, and this fact can be used in multiple ways.

5. Sparse observations and higher-order difference approximations

We now discuss what happens when the observations are sparse, so that there are data only every $r > 1$ time steps, and how to sample when the difference approximation is more elaborate than the Euler scheme used so far. Along the way, we suggest additional ways to solve the algebraic equations.

Consider again the discrete SDE (3), with observations available only at times $r\delta$, $r > 1$. To simplify the presentation, let $g(x, t) = 1$ and assume the equation is scalar. Write the scheme in the form $x^{n+1} = q(x^n) + \delta V$, where V is a Gaussian with mean zero and variance one. We have data at the points $r\delta$, $r > 1$, where $b^{n+r} = h(x^{n+r}) + \sqrt{s}W$ and W is a Gaussian of mean zero and variance one and s is a constant. The probability of the particle path $(X_i^{n+1}, X_i^{n+1}, \dots, X_i^{n+r})$ (from now on we will suppress the index i) is

$$P(X^{n+1}, \dots, X^{n+r}) = \exp(-F(X^{n+1}, \dots, X^{n+r}))/Z, \tag{12}$$

where Z is a normalization constant and

$$F(X^{n+1}, \dots, X^{n+r}) = \frac{(h(X^{n+r}) - b^{n+r})^2}{2s} + \frac{1}{2\delta}\sum_{i=1}^{r}(X^{n+i} - q(X^{n+i-1}))^2.$$

The task at hand is to solve

$$F(X^{n+1},\dots,X^{n+r}) - \min F(X^{n+1},\dots,X^{n+r}) = \xi^T\xi/2, \tag{13}$$

where ξ is a sample of a r-dimensional Gaussian reference variable. This can be done by the methods presented above, but we use this opportunity to present some variants.

First, we find the minimum of F. If F is convex, this can be easily done by Newton's method (note that the matrices one gets are sparse). If F is not convex, one can try the following device: add to F the quantity αG, where $\alpha > 0$ is a parameter and G is the convex function

$$G = (X^{n+r} - b^{n+r})^2 + \sum_{j=i+1}^{j=i+r} (X^j - X^{j-1})^2.$$

Then minimize $F + \alpha G$ for a suitable sequence $\alpha_n \to 0$. (This device was inspired by the rubber band construction of computational chemistry [12]. More generally, it is useful to note the resemblance of the problem to the study of rare transitions in computational chemistry [18; 10]). A minimization by a Newton's method also yields the Hessian H of F at the minimizer z of F.

Define $F_0 = \phi + (1/2)(X-z)^T H(X-z)$, where $\phi = \min F = F(z)$ and X is the vector $(X^{n+1},\dots,X^{n+r})$. Solve the equation $F_0(X) - \phi = \xi^T\xi/2$ and obtain X. This is a linear problem and Choleski construction works fine and also yields the Jacobian J. Use as weight for the resulting sample X the quantity $\exp(-\phi_0)J$, with $\phi_0 = \phi + F(X) - F_0(x)$ so that (13) is still satisfied and there is no bias. This is still a high-probability sample, because the neighborhood of $\xi = 0$ is still mapped on the neighborhood of the minimum of F.

As an example, consider the SDE $dx = \cos(5x)dt + \sigma dW$, with $\sigma = 0.1$, discretized by Euler's method with time step 0.01; the observations $b^n = x^n + \eta$ are available every 20 steps (a time interval of 0.2) and η is a Gaussian variable of mean zero and variance 10^{-3}. The data are generated by running the equation once and observing its path. We used 4 particles. In Figure 4 we display the run that generated the data path and the reconstruction; the data are used in the reconstruction only when $t = 0.2$ and $t = 0.4$. Observe that between data the discrepancy can be quite significant, as is indeed unavoidable.

This last example should make it plain what one should do when one uses a higher-order discretization of the SDE. For example, suppose one is integrating the SDE $dx = f(x)dt + dW$ using the second-order Klauder–Petersen scheme [15]:

$$x^{n+1,*} = x^n + \delta f(x^n) + \eta_1, \tag{14}$$

$$x^{n+1} = x^n + (\delta/2)\left(f(x^n) + f(x^{n+1,*})\right) + \eta_2, \tag{15}$$

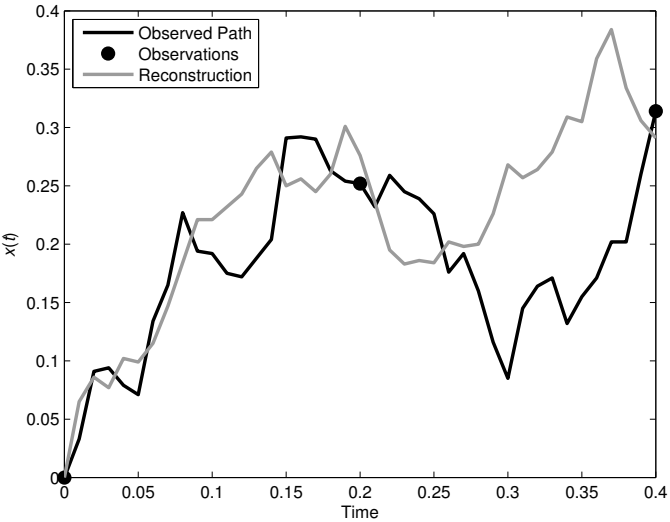


Figure 4. Reconstruction with sparse data.

where η_1, η_2 are Gaussians with mean zero and variance δ. Observations $b^n = h(x^n) + \eta_3$, where η_3 has mean zero and variance s, are assumed available at every step. The probability of the pair $(x^{n+1,*}, x^{n+1})$ is $\exp(-F)$, with

$$F = \frac{(x^{n+1,*} - x^n - \delta f(x^n))^2}{2\delta} + \frac{\left(x^{n+1} - x^n - \frac{\delta}{2}(f(x^n) + f(x^{n+1,*}))\right)^2}{2\delta} + \frac{\left(h(x^{n+1}) - b\right)^2}{2s}.$$

All one has to do then is solve $F - \min F = \xi^T \xi/2$ for a sample ξ of a two-dimensional Gaussian reference variable, along the lines suggested above.

6. Parameter identification

One important application of particle filters is to parameter identification, where the SDE contains an unknown parameter and the data are used to find this parameter's value. One of the standard ways of doing this [9; 14] is system augmentation: one adds to the SDE the equation $d\sigma = 0$ for the unknown parameter σ, one offers σ a gamut of possible values, and one relies on the resampling process that eliminates the values that do not fit the data. With the implicit filter this procedure fails, because the particles are not eliminated fast enough. One alternative is finding the unknown parameter σ by stochastic approximation. Specifically, find a statistic T of the output of the filter which is a function of σ, such that the expected value $E[T]$ vanishes when σ has the right value σ^*, and then solve the equation $E[T] = E[T(\sigma)] = 0$

by the Robbins–Monro iteration [20]:

$$\sigma_{n+1} = \sigma_n - \alpha_n T(\sigma_n), \tag{16}$$

which converges when the coefficients α_n are such that $\sum \alpha_n \to \infty$ while $\sum \alpha_n^2$ remains bounded (for example, $\alpha_n = 1/n^q$ with $0.5 < q \leq 1$). Related ideas can be found, for example, in [6].

As a concrete example, consider the SDE $dx = dW$, where W is Brownian motion with variance σ, discretized with time steps δ, with observations $b^n = x^n + \eta$, where η is a Gaussian with mean zero and variance s. Data are generated by running the SDE once with the true value σ^* of σ, adding the appropriate noise, and registering the result at time $n\delta$ as b^n for $n = 1, 2, \ldots, N$. For the functional T we chose

$$T(\sigma) = C \frac{\sum(\Delta_i \Delta_{i-1})}{\sqrt{\left(\sum \Delta_i^2\right)\left(\sum \Delta_{i-1}^2\right)}}, \tag{17}$$

where the summations are over i between 2 and N, Δ_i is the estimate of the increment of x in the i-th step and C is a scaling constant. Clearly if the σ used in the filtering equals σ^* then by construction the successive values of Δ_i are independent and $E[T] = 0$. We picked the parameters $N = 100$, $\sigma = 10^{-2}$, $s = 10^{-4}$, $\delta = 0.01$ (so that the increment of W in one step has variance 10^{-4}).

Our algorithm is as follows: We make a guess σ_1, run the filter for N steps, evaluate T, and make a new guess for σ using (16) and $\alpha_1 = 1$, rerun the filter, etc., with the α_n, the coefficient in (16) at the n-th step, equal to $1/n$. The scaling factor in (17) was found by trial and error: if it is too large the iteration becomes unstable, if it is too small the convergence is slow; we settled on $C = 4$.

This algorithm requires that the filter be run without resampling, because resampling introduces correlations between successive values of the Δ_i and bias the values of T. In a long run, in particular in a strongly nonlinear setting, one may need resampling for the filter to stay on track, and this can be done by segmentation: divide the run of the filter into segments of some moderate length L, perform the summations in the definition of T over that segment, then go back and run that segment with resampling, then proceed to the next segment, etc.

The first question is, how well is it possible in principle to reconstruct an unknown value of σ from N observations; this issue was already discussed in [4]. Given 100 samples of a Gaussian variable of mean 0 and variance σ, the variance reconstructed from the observations is a random variable of mean σ and variance $0.16 \cdot \sigma$; 100 observations do not contain enough information to reconstruct σ perfectly. A good way to estimate the best result that can be achieved is to run the algorithm with the guess σ_1 equal to the exact value σ^* with which the data were generated. When

Iteration	0	1	2	3	4	5	6	7	8	9
new estimate σ/σ^*	10	0.819	0.943	1.02	1.05	1.08	1.10	1.13	1.15	1.16

Table 6. Convergence of the parameter identification algorithm.

this was done, the estimate of σ was $1.27\sigma^*$. This result indicates the order of magnitude of the accuracy that can be achieved.

In Table 6 we display the result of our algorithm, run with 50 particles and starting value $\sigma_1 = 10\sigma^*$. Each iteration requires running the filter once.

7. Conclusions

We have presented the implicit filter for data assimilation, together with several algorithms for the solution of its algebraic equations, including cases with nonconvex functions F, as well as an algorithm for parameter identification. The key idea in implicit sampling is to solve an algebraic equation of the form

$$F(X) - \phi = \xi^T \xi/2$$

for every particle, where the function F is explicitly known, X is the new position of the particle, ϕ is an additive factor, and ξ is a sample of a fixed reference pdf; F varies from particle to particle and step to step. This construction makes it possible to guide the particles to the high-probability area one by one under a wide variety of circumstances. It is important to note that the equation that links ξ to X is underdetermined and its solution can be adapted for each particular problem. The effectiveness of implicit sampling depends on one's ability to design maps $\xi \to X$ that satisfy the criteria above and are computationally efficient. The design of such maps is problem dependent and we will present examples in the context of specific applications.

Acknowledgements

We would like to thank Prof. Jonathan Weare for asking penetrating questions and for making very useful suggestions, Prof. Robert Miller for his advice and encouragement, and Mr. G. Zehavi for performing some of the preliminary computations.

References

[1] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, *A tutorial on particle filters for online nonlinear/nongaussian Bayesian tracking*, IEEE Trans. Sig. Proc. **50** (2002), 174–188.

[2] P. Bickel, B. Li, and T. Bengtsson, *Sharp failure rates for the bootstrap particle filter in high dimensions*, Pushing the limits of contemporary statistics: contributions in honor of Jayanta K. Ghosh (B. Clarke and S. Ghosal, eds.), IMA Collections, no. 3, Inst. Math. Statist., Beachwood, OH, 2008, pp. 318–329. MR 2010c:93107

[3] A. J. Chorin, *Monte Carlo without chains*, Commun. Appl. Math. Comput. Sci. **3** (2008), 77–93. MR 2425547 Zbl 1165.65302

[4] A. J. Chorin and X. Tu, *Implicit sampling for particle filters*, Proc. Nat. Acad. Sc. USA **106** (2009), 17249–17254.

[5] ———, *An iterative implementation of the implicit nonlinear filter*, 2010, submitted to *Math. Mod. Num. Anal.*

[6] D. P. Dee, *On-line estimation of error covariance parameters for atmospheric data assimilation*, Mon. Wea. Rev. **123** (1995), 1128–1145.

[7] P. Del Moral, *Feynman–Kac formulae: Genealogical and interacting particle systems with applications*, Springer, New York, 2004. MR 2005f:60003 Zbl 1130.60003

[8] A. Doucet, S. Godsill, and C. Andrieu, *On sequential Monte Carlo sampling methods for Bayesian filtering*, Stat. Comp. **10** (2000), 197–208.

[9] A. Doucet, N. de Freitas, and N. Gordon (eds.), *Sequential Monte Carlo methods in practice*, Springer, New York, 2001. MR 2003h:65007

[10] W. E, *Principles of multiscale modeling*, Cambridge University Press, New York, To appear.

[11] G. Evensen, *Data assimilation: the Ensemble Kalman Filter*, 2nd ed., Springer, Berlin, 2009. MR 2555209 Zbl 1157.86001

[12] R. Gillian and K. Wilson, *Shadowing, real events, and rubber bands. a variational Verlet algorithm for molecular dynamics*, J. Chem. Phys. **97** (1992), 1757–1772.

[13] E. Isaacson and H. B. Keller, *Analysis of numerical methods*, John Wiley & Sons, New York, 1966. MR 34 #924 Zbl 0168.13101

[14] G. Kitagawa, *A self-organizing state-space model*, J. Am. Stat. Ass. **93** (1998), 1203–1215.

[15] J. R. Klauder and W. P. Petersen, *Numerical integration of multiplicative-noise stochastic differential equations*, SIAM J. Numer. Anal. **22** (1985), no. 6, 1153–1166. MR 87a:34064 Zbl 0583.65098

[16] P. E. Kloeden and E. Platen, *Numerical solution of stochastic differential equations*, Appl. Math., no. 23, Springer, Berlin, 1992. MR 94b:60069 Zbl 0752.60043

[17] I. N. M. Jardak and M. Zupanski, *Comparison of ensemble data assimilation for the shallow water equations model in the presence of nonlinear observation operators*, J. Geophys. Res. (2010), in press.

[18] P. Metzner, C. Schuette, and E. Vanden-Eijnden, *Illustration of transition path theory on a collection of simple examples*, J. Chem. Phys. **125** (2006), 084110.

[19] R. Miller, E. Carter, and S. Blue, *Data assimilation into nonlinear stochastic systems*, Tellus **51A** (1999), 167–194.

[20] H. Robbins and S. Monro, *A stochastic approximation method*, Ann. Math. Statistics **22** (1951), 400–407. MR 13,144j Zbl 0054.05901

[21] C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson, *Obstacles to high-dimensional particle filtering*, Mon. Wea. Rev. **136** (2008), 4629–4640.

[22] A. M. Stuart, *Inverse problems: a Bayesian perspective*, Acta Numer. **19** (2010), 451–559. MR 2652785

[23] J. Weare, *Efficient Monte Carlo sampling by parallel marginalization*, Proc. Nat. Acad. Sc. USA **104** (2007), 12657–12662.

[24] J. Weare, *Particle filtering with path sampling and an application to a bimodal ocean current model*, J. Comput. Phys. **228** (2009), no. 12, 4312–4331. MR 2010g:86010 Zbl 1165.76045

[25] M. Zakai, *On the optimal filtering of diffusion processes*, Z. Wahrscheinlichkeitstheorie und Verw. Gebiete **11** (1969), 230–243. MR 39 #3883 Zbl 0164.19201

Received May 24, 2010.

ALEXANDRE CHORIN: chorin@math.berkeley.edu
Department of Mathematics, University of California, Berkeley, Berkeley, CA 94720, United States
http://math.berkeley.edu/∼chorin

MATTHIAS MORZFELD: mmo@berkeley.edu
Department of Mechanical Engineering, University of California, Berkeley, CA 94720, United States

XUEMIN TU: xuemin@math.berkeley.edu
Department of Mathematics, University of Kansas, 1460 Jayhawk Boulevard, Lawrence, KS 66045, United States
http://math.ku.edu/∼xtu

COMM. APP. MATH. AND COMP. SCI.
Vol. 5, No. 2, 2010

PARALLEL IN TIME ALGORITHMS WITH REDUCTION METHODS FOR SOLVING CHEMICAL KINETICS

ADEL BLOUZA, LAURENT BOUDIN AND SIDI MAHMOUD KABER

We design suitable parallel in time algorithms coupled with reduction methods for the stiff differential systems integration arising in chemical kinetics. We consider linear as well as nonlinear systems. The numerical efficiency of our approach is illustrated by a realistic ozone production model.

1. Introduction

Parareal algorithms were first introduced in [18] to solve evolution problems in real time. The principle is the following. One first approximates the solution on a coarse time grid, and then locally solves the equations on fine time subgrids on parallel computers. One can prove that the associated iterative procedure ensures an accuracy which is of same order as a sequential algorithm on a global fine time grid. Mathematical properties of these algorithms have been recently investigated; see [2; 25; 16; 15; 13; 14; 12], for example. They have been applied in various fields, such as financial mathematics [3], fluid mechanics and fluid-structure interaction [9; 11; 10], oceanography [19], chemistry [21] and quantum chemistry [22].

The present work is dedicated to standard chemistry. We study monomolecular chemistry, as in [21], for which we carry out a new modified parareal algorithm preserving stoichiometric invariants. We also investigate the nonlinear chemistry case.

When the reaction scheme is monomolecular, the kinetic equations describing species evolutions are linear but may be stiff. In this context, we consider the thyroid reaction scheme given in [23]. An efficient reduction algorithm is described in [5], and applied to this biochemical model. It is an inductive procedure, based on linear algebraic techniques. The reduction process, applied to the initial kinetic system, eliminates the fastest dynamics, and no change of coordinates is required. This process is systematic and does not rely on conventional chemical assumptions (see [27] for a large survey of these techniques). Applied to chemical kinetic systems with kinetic constants in different scales, the algorithm eliminates reactants

MSC2000: 65L05, 65L80, 68W10, 80A32.
Keywords: chemical kinetics, reduction, parareal algorithm.
Funded by the ANR-06-CIS6-007-01 project PITAC headed by Y. Maday.

arising in some of the fastest reactions. The reduced system then provides an accurate approximation for the slow dynamics.

The ozone model we study here is a typical nonlinear, realistic model for ozone production in the troposphere. The issue of ozone pollution is one of the most important environmental problems we have faced for the last three decades (see [24] and the references therein). The massive presence in the troposphere, mainly above urban areas, of nitrogen dioxide NO_2, coupled with one of various hydrocarbons, induces a preferential chain of reactions which produces ozone O_3. This chain is really favored by a large amount of ultraviolet rays, basically during sunny summers. The ozone concentration then reaches a level that may be dangerous for both human health and ecosystems.

The ozone model of [1; 4] describes the evolution of the main species concentrations at stake. Numerical simulations of reactive flows can often be really difficult to tackle, mainly because of the intricate chemical mechanisms that must be taken into account. That is the case, for instance, with the air quality issue: we do not focus on a simple description of the chemical kinetics of reactions of nitrogen oxides and ozone. We need to take into account more reactions including pollutants themselves to model more faithfully the pollution in the atmosphere.

We aim to compute numerically the evolution of the chemical species in the atmosphere, including the pollutants, within a reasonable computational time. Some of the phenomena are really stiff and have to be discretized with a very small time step. To get around this major numerical difficulty, we use a suitable parareal algorithm where the associated coarse propagator is applied to the reduced system in order to minimize its cost. We also need an accurate description of both the physics (convection, diffusion, source or well of pollution) and the chemistry (reactions). Nevertheless, it is quite clear that our PDE system is large, and most of the nonlinearity comes from the chemical part. We here assume that the chemistry also governs the coupling between our equations. Therefore, the chemical kinetics naturally stand as the key point of our study of the ozone model in the troposphere.

We first focus on the kinetics of the reactions producing ozone, and drop the dependence on the space variable. Those reactions lead to an ODE system, where several problems have to be taken care of.

- There is a large variety of characteristic time scales for the species involved.
- Dozens or even hundreds of chemical reactions and species may be concerned.
- Most of the ODEs are nonlinear.

Once again, our approach consists in first using a reduced model: when possible, we approximate the full differential system by an algebraic-differential system where transitional fast states are neglected. This allows us to simultaneously lower the size of the system and to avoid or at least weaken its stiffness. In the nonlinear

case, which is the most common situation, reduction algorithms are not so easy to design. However, the quasisteady state (QSS) method seems to be an efficient compromise: some species are put at chemical equilibrium, with high rates of both production and consumption (the species is destroyed as soon as it is produced).

This work is organized as follows. In Section 2, we recall a convenient parareal algorithm for our models, which considers stoichiometric invariants. Section 3 is dedicated to the study of monomolecular chemistry, and Section 4 to the ozone model. In each case, we describe the chemical models, briefly discuss a reduction method fitted to the situations, and show numerical results that indicate the efficiency of the parareal algorithm.

2. Parareal algorithm

Let $m \geq 1$ and consider the following ordinary differential equation, where $y : \mathbb{R}_+ \to \mathbb{R}^m$ is the unknown:

$$y'(t) = f(t, y(t)), \tag{1}$$

with initial Cauchy condition $y(0) = y_0 \in \mathbb{R}^m$ and where $f : \mathbb{R}_+ \times \mathbb{R}^m \to \mathbb{R}^m$ is given. Section 3 corresponds to the linear version of (1), that is, with

$$f(t, y(t)) = Jy(t),$$

where J is a time-independent matrix.

We are interested in computing the solution u of (1) on an interval $[0, T]$, with $T > 0$. For any $N \geq 1$, we consider intermediate times $0 = T_0 < T_1 < \cdots < T_N = T$ and, for the sake of simplicity, a constant coarse time step $\Delta T = T_{n+1} - T_n$, where n denotes the coarse time index. Note that ΔT may not be constant, and that the associated algorithm would only be an adjustment of the one presented below.

Let k denote the parareal iteration index. The parareal scheme consists in designing a sequence $(y_n^k)_{k\in\mathbb{N}}$ at each coarse time step $[T_n, T_{n+1}]$ such that, for each n,

$$\lim_{k\to+\infty} y_n^k = \bar{y}_n,$$

where $\bar{y}_n$ is an approximation of $y(T_n)$, and the convergence, which of course depends on the accuracy of the coarse propagator, should be fast. Indeed, in many applications, $k \leq 5$ is enough to get a satisfying approximation.

The parareal algorithm uses two different schemes: a fine one $\mathcal{F}_{\delta t}$, based on a fine time step $\delta t > 0$, and a coarse one $\mathcal{C}_{\Delta T}$, based on coarse time step $\Delta T = s\delta t$, $s \in \mathbb{N}^*$. In most situations, we consider $s \gg 1$.

The coarse solver is applied for the evolution on $[0, T]$. The quantity

$$v_{n+1} = \mathcal{C}_{\Delta T}(T_{n+1}; T_n, v_n)$$

is an approximation, at time T_{n+1}, of the solution of (1) on $[T_n, T_{n+1}]$, with initial value v_n at time T_n. Note that v_{n+1} is the only value computed by the coarse solver on $]T_n, T_{n+1}]$. For simplicity and stability reasons, we use the implicit Euler scheme. In the linear case, we write $v_{n+1} = Cv_n$, where $C = (I - \Delta T\ J)^{-1}$.

The fine solver is applied for the evolution on each subinterval $[T_n, T_{n+1}]$. The quantity

$$w_{n+1} = \mathcal{F}_{\delta t}(T_{n+1}; T_n, w_n)$$

is an approximation, at time T_{n+1}, of the solution of (1) on $[T_n, T_{n+1}]$, with initial value w_n at time T_n. In the linear case, w_{n+1} can be written under the form $w_{n+1} = F^s w_n$, where F is a time-independent matrix. For instance, in the case of an explicit Euler scheme, we have $F = I + \delta t\ J$. We may also use the Runge–Kutta RK4 and the implicit Euler schemes.

The full parareal sequence $(y_n^k)_{n,k}$ is inductively defined, for any k, n, by

$$y_0^k = y_0, \tag{2}$$

$$y_{n+1}^0 = \mathcal{C}_{\Delta T}(T_{n+1}; T_n, y_n^0), \tag{3}$$

$$y_{n+1}^{k+1} = \mathcal{C}_{\Delta T}(T_{n+1}; T_n, y_n^{k+1}) + \mathcal{F}_{\delta t}(T_{n+1}; T_n, y_n^k) - \mathcal{C}_{\Delta T}(T_{n+1}; T_n, y_n^k). \tag{4}$$

Passing to the limit in (4) as k goes to $+\infty$, we get

$$\bar{y}_{n+1} = \mathcal{F}_{\delta t}(T_{n+1}; T_n, \bar{y}_n).$$

This relation means that $\bar{y}_{n+1}$ is obtained from $\bar{y}_n$ by use of the fine scheme $\mathcal{F}_{\delta t}$ on the interval $[T_n, T_{n+1}]$. That ensures that y_n^k may be a good approximation of the fine solution when k is not too small.

In order to reduce the CPU cost, we shall apply the coarse propagator $\mathcal{C}_{\Delta T}$ to a reduced system of ODEs, that is

$$\tilde{y}'(t) = \tilde{f}(t, \tilde{y}(t)) \tag{5}$$

where $\tilde{m} < m$ and $\tilde{f} : \mathbb{R}_+ \times \mathbb{R}^{\tilde{m}} \to \mathbb{R}^{\tilde{m}}$ can be computed from f. It is assumed that (5) is easier to solve than (1) and that $\tilde{f}$ describes a simpler but still faithful physics.

Remark 1. In a chemical kinetics context, when we apply algorithms (2)–(4) to integrate the differential system, we should pay special attention to preserving certain properties of the system, such as stoichiometric invariants and stationary points. We shall see below how to establish from (2)–(4) an ad hoc numerical scheme considering these invariants.

3. Linear chemistry

3.1. ***Framework.*** Consider N gaseous species $(\mathcal{A}_i)_{1\le i\le N}$ and denote by $(y_i)_{1\le i\le N}$ the corresponding concentrations. In the gaseous mixture, there can also be some species $(\mathcal{N}_l)_{1\le l\le L}$, like O_2 or N_2 in air, whose concentration variations are neglected. Consequently, we only take into account the variations of (y_i) of the limiting species $(\mathcal{A}_i)$. We assume that R chemical reactions take place simultaneously in the mixture, and that these R reactions are monomolecular in the species $(\mathcal{A}_i)$; that is, each reaction r can be written as

$$\mathcal{A}_{i_r} + \mathcal{N}_{i_r} \xrightarrow{k_r} \mathcal{A}_{j_r} + \mathcal{N}_{j_r},$$

with reaction rate $v_r = k_r y_{i_r}$, where k_r is a given positive kinetic constant. For the sake of simplicity, we assume that a pair $(\mathcal{A}_i, \mathcal{A}_j)$ appears in at most one reaction as reactant-product.

The time evolution of the concentration y_i, $1 \le i \le N$, is governed by the law of mass action

$$\frac{\mathrm{d}y_i}{\mathrm{d}t} = -\sum_{r,\ i_r=i} k_r y_{i_r} + \sum_{r,\ j_r=i} k_r y_{j_r} = -\sum_{r,\ i_r=i} v_r + \sum_{r,\ j_r=i} v_r.$$

The first sum deals with the reactions where $\mathcal{A}_i$ is a reactant, and the second one deals with the ones where $\mathcal{A}_i$ is a product. In other words, $y = (y_i)_{1\le i\le N}$ satisfies the stoichiometric system

$$\frac{\mathrm{d}y}{\mathrm{d}t} = Sv,$$

where $v = (v_r)_{1\le r\le R}$ is the vector of the reaction rates and $S \in \mathbb{R}^{N\times R}$ is the stoichiometric matrix, defined by $S_{i,r} = -1$ for $i = i_r$, $S_{i,r} = 1$ for $i = j_r$, and $S_{i,r} = 0$ otherwise. Since the chemical reactions are assumed to be monomolecular, v linearly depends on y. Hence, y solves the following differential system, for a given initial datum,

$$\frac{\mathrm{d}y}{\mathrm{d}t} = Jy, \quad t \ge 0, \tag{6}$$

where $J \in \mathbb{R}^{N\times N}$ is defined by $J_{jj} = -\sum_{r,\ i_r=j} k_r$, $J_{ij} = k_r$ if $i \ne j$ and there exists r such that $(i_r, j_r) = (j, i)$, and $J_{ij} = 0$ otherwise. By assumption on our system, if $i \ne j$ and if there exists a reaction r such that $(i_r, j_r) = (j, i)$, then r is unique. Note that J is a kinetic matrix, that is, it satisfies, for any j, $J_{jj} \le 0$, $J_{ij} \ge 0$ for all $i \ne j$, and $\sum_i J_{ij} = 0$.

The kinetic matrix J is semistable, that is, all its nonzero eigenvalues have negative real parts. Besides, 0 is an eigenvalue of J and its multiplicity indicates the number of stoichiometric invariants. That implies that any solution of (6) has a finite limit when t goes to $+\infty$. The reader may find more details in [7; 8; 26].

3.2. ***A reduction method.*** In [5], the authors introduced an algorithm to obtain, from (6), a reduced and nonstiff system which only involves a subset of the initial species, coupled with algebraic equations for the remaining species concentrations. This algebraic-differential system accurately approximates the full stiff system, after the exit time from the boundary layer. Let us briefly recall this reduction method and the associated error estimate.

At each reduction step $1 \le k \le N-1$, an index $i_k \in \{1, \dots, N\}$, a semistable matrix J^k, and real coefficients $(\beta_{i_k,j})_{j \in \mathcal{H}_k}$ are inductively built with

$$\mathcal{H}_k = \{1, \dots, N\} \backslash \{i_1, \dots, i_k\}.$$

Let us then fix a step $1 \le p \le N-1$ at which we decide to stop the reduction process. For any $y = (y_i)_{1 \le i \le N}$, we denote $\tilde{y} = (y_i)_{i \in \mathcal{H}_p}$. The reduced system associated to (6) up to step p is the following algebraic-differential system of unknown $z^p = (z_i^p)_{1 \le i \le N}$, defined for a given initial datum (at $t = T^*$) for the differential part of the system:

$$\frac{\mathrm{d}\tilde{z}^p}{\mathrm{d}t} = J^p \tilde{z}^p, \quad t \ge T^*, \tag{7}$$

$$z_{i_k}^p(t) = \sum_{j \in \mathcal{H}_k} \beta_{i_k j} z_j^p(t), \quad 1 \le k \le p, \quad t \ge T^*. \tag{8}$$

If p is suitably chosen, the reduced matrix J^p only contains the $N-p$ eigenvalues of J which have the lowest real parts. Note that, when the eigenvalues of J^p are small with respect to the p first eigenvalues of J, the differential system (7) is not stiff anymore (this idea was also used in [17; 20]). Here, the algebraic equations (8) approximate the fast species $(z_i)_{i \notin \mathcal{H}_p}$, and T^* is an exit time from the corresponding boundary layer. Real coefficients $\beta_{i_k j}$ are expressed in terms of left eigenvectors of matrices J^k, $k = 0, \dots, p$.

It is shown in [5] that the nonstiff problem (7)–(8) actually yields a relevant approximation of the solutions of (6) for $t \ge T^*$. More precisely, assume that the initial data for (7) is chosen such that

$$|z_j^p(T^*) - y_j(T^*)| \le ch, \quad \text{for all } j \in \mathcal{H}_p,$$

where h can be viewed as the numerical error of the underlying scheme at time T^* and c is a nonnegative constant which does not depend on h. Then there exists $\alpha \ge 1$ and $C \ge 0$, depending on $y(0)$, such that, for any $t \ge T^*$, with $T^* \ge \alpha\varepsilon \ln(1/\varepsilon)$,

$$|z_j^p(t) - y_j(t)| \le C(h + \varepsilon), \quad 1 \le j \le N.$$

In other words, provided that the errors due to the prescribing of the values of the slow species at the exit time T^* from the boundary layer are small, the errors with respect to the exact solution remain small at any further time $t \ge T^*$.

3.3. ***Numerical tests.*** We apply the previous algorithm to the dynamics of the thyroid hormones [23]. It was investigated in [5] for the reduction method, and in [21] for both reduction method and parareal algorithm, including a source term. The chemical network (see Figure 1) involves 8 species and 14 reactions.

The kinetic matrix of the set of reactions reads

$$J = \begin{pmatrix} -5.1 & 0.01 & 0. & 0. & 0.06 & 0. & 0. & 0. \\ 0. & -2.516 & 0. & 0. & 0. & 0.0008 & 0. & 0. \\ 0. & 0. & -1.3 & 0.001 & 0.0003 & 0. & 0. & 0. \\ 0. & 0. & 0. & -1.091 & 0. & 0.00008 & 0. & 0. \\ 5. & 0. & 1. & 0. & -0.0603 & 0. & 0. & 0. \\ 0. & 2.5 & 0. & 1. & 0. & -0.00088 & 0. & 0. \\ 0.1 & 0.006 & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0.3 & 0.09 & 0. & 0. & 0. & 0. \end{pmatrix}.$$

In the following subsections, we compare the computational behavior of reduced and/or parareal algorithms with respect to the fine algorithm. Let us note that for all computations, the speed-up is defined as the ratio

$$\frac{\text{CPU (fine scheme)}}{\text{CPU (current scheme)}},$$

where the CPU of the currently studied scheme takes into account the initialization step and neglects the communications between processors since we only simulate parareal implementation, and not perform actual parareal computations.

3.3.1. *Parareal algorithm vs. fine algorithm.* The computations are first performed up to final time $T = 3$ with the following parameters for the parareal algorithm. The coarse grid has $N = 50$ cells which constitute a regular subdivision of $[0, 3]$, so that the coarse time step is $\Delta T = 0.06$. Then each coarse cell is divided into a regular fine subdivision of $s = 500$ cells, so that the fine time step is $\delta t = 0.00012$. The numerical tolerance is set to 0.01. We use Runge–Kutta RK4 for the fine

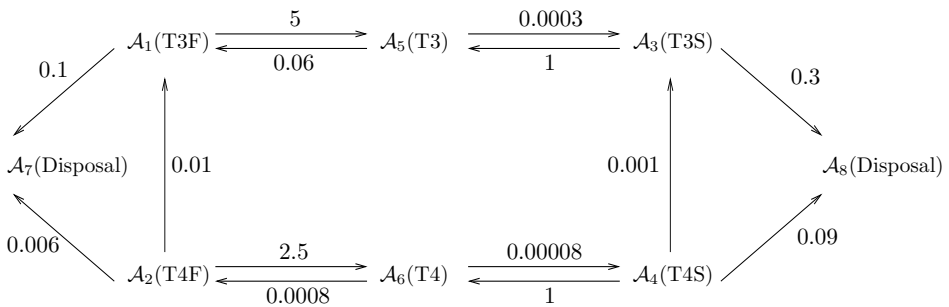


Figure 1. The thyroid reaction scheme.

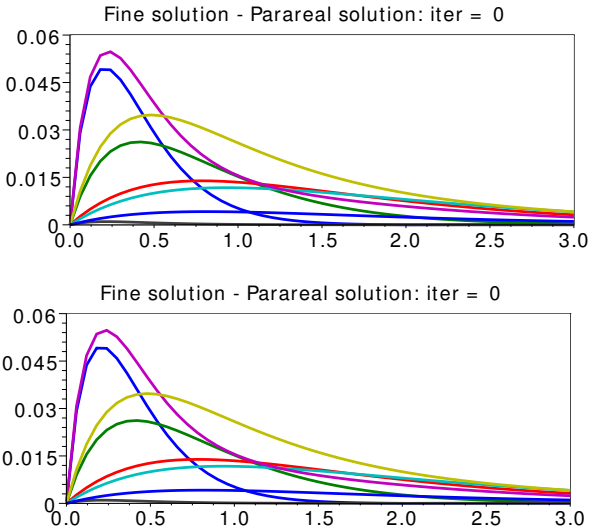


Figure 2. Thyroid: numerical error between the fine solution and (top) the coarse initialized solution, and (bottom) the parareal solution after 3 parareal iterations.

scheme and the implicit Euler for the coarse scheme. The speed-up we obtain is approximately 12.

Figure 2 shows the error at the initialization step (top) and the error between the fine and parareal solutions (bottom). We observe that the maximal error is divided by 1000 after three iterations of the parareal algorithm.

Moreover, we check numerically (Figure 3) that the two stoichiometric invariants of this problem still hold. The plots of both invariants are exactly superimposed, since both fine and coarse solvers in the parareal algorithm conserve the stoichiometric invariants.

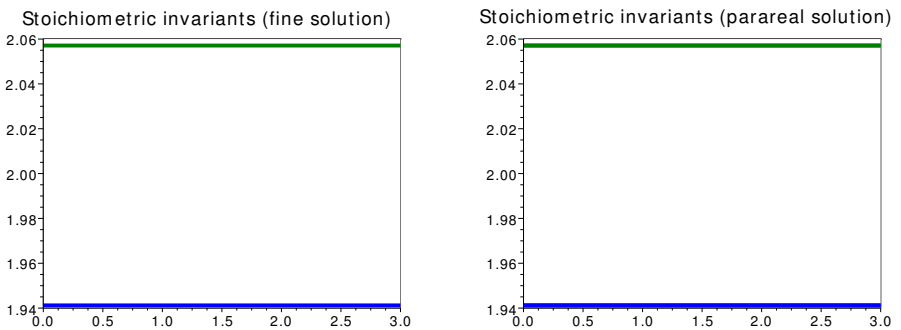


Figure 3. Thyroid: stoichiometric invariants.

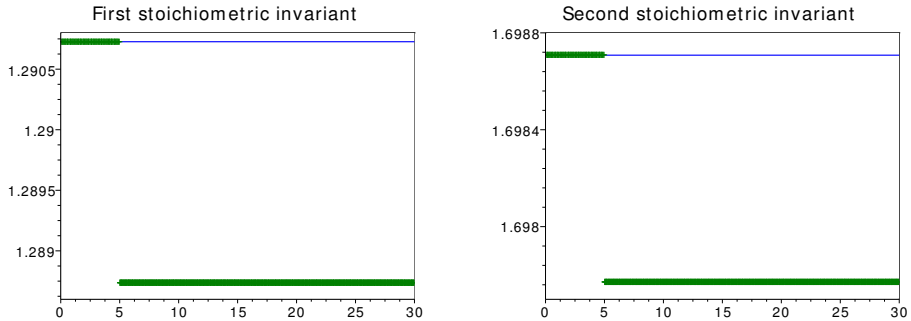


Figure 4. Thyroid: stoichiometric invariants with reduction.

3.3.2. *Reduction algorithm vs. fine algorithm.* We now use the reduction method described in Section 3.2. Matrix J has four dominant eigenvalues of the same magnitude, lying between -6 and -1. The remaining eigenvalues are either 0, with multiplicity 2, or very small with respect to 1. We compare the results obtained with the fine solver and by applying the reduction method. For the reduction, we compute the solution of the full system up to the characteristic time $T^* = 5$, then the solution of the reduced system up to final time $T = 30$, including the algebraic equations.

One can check on Figure 4 that the stoichiometric invariants are not conserved. However, the jumps in the stoichiometric invariants are small. We also note that computations with the reduction method are 2.5 times faster compared to those obtained by the fine solver.

3.3.3. *Parareal algorithm with reduction method.* We use the values $T^* = 10$ and $T = 30$ as in 3.3.2. For the parareal algorithm, the coarse solver is, on $[0, T^*]$ (respectively on $[T^*, T]$), the implicit Euler scheme for the full problem with a coarse time step $\Delta T_1 = 2$ (respectively for the reduced problem with a coarse time step $\Delta T_2 = 4$). The fine solver is still RK4, with a fine time step $\delta t_1 = 0.02$ on $[0, T^*]$, and $\delta t_2 = 0.04$ on $[T^*, T]$. The tolerance is set at 0.01. The method converges in three iterations and we obtain a speed-up of 4.3 for 10 processors, that is, an efficiency of 0.43.

On Figure 5, we can check that the error between the parareal algorithm and the reduction method decreases with the number of parareal iterations. Eventually, we note on Figure 6 that the stoichiometric invariants are not conserved. Hence, a basic parareal algorithm connected with the reduction method does not conserve the stoichiometric invariants.

3.3.4. *A modified parareal algorithm preserving stoichiometric invariants.* Recall that a stoichiometric invariant γ is a linear combination of concentrations (y_i) such that its time derivative is nil.

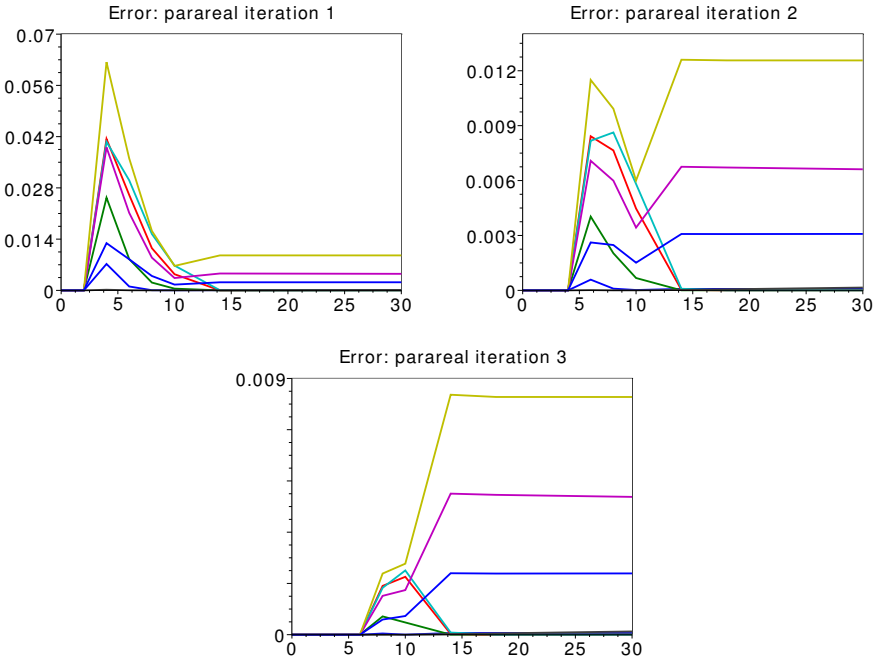


Figure 5. Thyroid: errors during the parareal iterations.

In other words, there exists a nonzero vector $w \in \ker(S^T)$ such that $\gamma = \langle w, y \rangle$ and

$$\frac{\mathrm{d}\gamma}{\mathrm{d}t} = \langle w, Sy(t) \rangle = \langle S^T w, y(t) \rangle = 0.$$

This shows that $y(t)$ belongs to the affine space $y(0) + \mathrm{Range}(S)$.

In order to conserve the stoichiometric invariants, we consider the modified parareal algorithm

$$y_{n+1}^{k+1} = \mathcal{F}_{\delta t}(y_n^k) + \Pi(\mathcal{C}_{\Delta T}(y_n^{k+1}) - \mathcal{C}_{\Delta T}(y_n^k)) \tag{9}$$

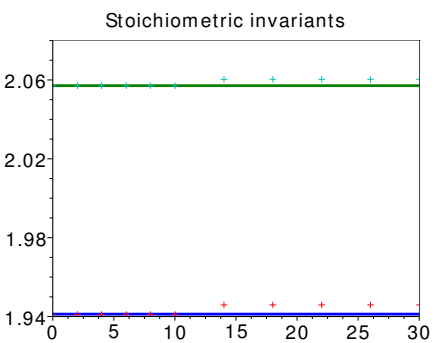


Figure 6. Thyroid: fine (solid) and parareal (+) stoichiometric invariants.

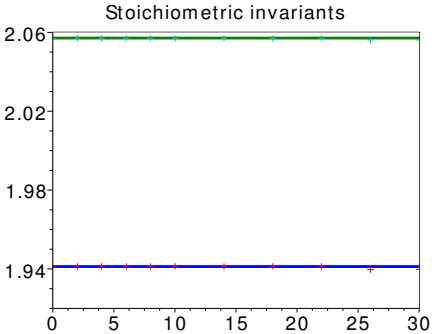


Figure 7. Thyroid: fine (solid) and new parareal (+) stoichiometric invariants with the improved parareal algorithm.

where Π is the orthogonal projection on the range of J. Nonorthogonal projections can be used too.

Of course, one can then check on Figure 7 that the stoichiometric invariants are now conserved. Moreover, the algorithm is still a good approximation of the fine solution with the same number of iterations and the same speed-up.

4. Example of nonlinear chemistry: an ozone model

4.1. *Description of the model.*

4.1.1. *Ozone model.* The ozone model we investigate here is described in detail in [1]. It involves 16 species and 12 reactions, that makes it a rather simple model, though realistic. The kinetic constant of reaction r is denoted k_r, $1 \leq r \leq 12$. The chemical reactions are

$$\begin{aligned}
\mathrm{OD} + \mathrm{air} + \mathrm{O_2} &\xrightarrow{k_1} \mathrm{O_3} + \mathrm{air} + \mathrm{O_2}, & \mathrm{RCHO} + \mathrm{OH} &\xrightarrow{k_7} \mathrm{RCO_3},\\
\mathrm{O_3} + \mathrm{NO} &\xrightarrow{k_2} \mathrm{NO_2}, & \mathrm{RCHO} &\xrightarrow{k_8} \mathrm{RO_2} + \mathrm{CO} + \mathrm{HO_2},\\
\mathrm{NO} + \mathrm{HO_2} &\xrightarrow{k_3} \mathrm{NO_2} + \mathrm{OH}, & \mathrm{NO} + \mathrm{RO_2} &\xrightarrow{k_9} \mathrm{NO_2} + \mathrm{RCHO} + \mathrm{HO_2},\\
\mathrm{OH} + \mathrm{NO_2} &\xrightarrow{k_4} \mathrm{NHO_3}, & \mathrm{NO} + \mathrm{RCO_3} &\xrightarrow{k_{10}} \mathrm{NO_2} + \mathrm{RO_2} + \mathrm{CO_2},\\
\mathrm{NO_2} &\xrightarrow{k_5} \mathrm{NO} + \mathrm{OD}, & \mathrm{RCO_3} + \mathrm{NO_2} &\xrightarrow{k_{11}} \mathrm{RCO_3NO_2},\\
\mathrm{RH} + \mathrm{OH} &\xrightarrow{k_6} \mathrm{RO_2}, & \mathrm{RCO_3NO_2} &\xrightarrow{k_{12}} \mathrm{RCO_3} + \mathrm{NO_2}.
\end{aligned}$$

The previous scheme is already somehow reduced, since the concentrations of air, O_2 and H_2O (which does not appear in this system, but is necessary) are considered as very high constants. Therefore, some reactions do not seem to be balanced. That only means that we may not take into account those three species in

the previous reactions. Note also that CO_2, NHO_3 and CO are not reactants. The values of the kinetic constants are the following:

$$\begin{array}{llllll} k_1 = 10^{-33}, & k_5 = 8.9\,10^{-3}, & k_9 = 7.6\,10^{-12}, \\ k_2 = 2\,10^{-14}, & k_6 = 2.6\,10^{-12}, & k_{10} = 7.6\,10^{-12}, \\ k_3 = 8.2\,10^{-12}, & k_7 = 1.6\,10^{-11}, & k_{11} = 4.7\,10^{-12}, \\ k_4 = 1.1\,10^{-11}, & k_8 = 3.2\,10^{-6}, & k_{12} = 4\,10^{-4}. \end{array}$$

Each species is denoted by an integer index, as follows:

index	1	2	3	4	5	6	7	8
species	air	O_2	CO_2	NHO_3	RH	CO	NO	NO_2
index	9	10	11	12	13	14	15	16
species	RCO_3NO_2	RCHO	O_3	OH	HO_2	RCO_3	RO_2	OD

The vector $y = (y_i)_{1\le i\le 16} \in \mathbb{R}^{16}$, whose coordinates y_i are the concentrations of the species represented in the previous table solves the following differential system

$$\begin{aligned} y_1' &= 0, & y_9' &= v_{11} - v_{12}, \\ y_2' &= 0, & y_{10}' &= -v_7 - v_8 + v_9, \\ y_3' &= v_{10}, & y_{11}' &= v_1 - v_2, \\ y_4' &= v_4, & y_{12}' &= v_3 - v_4 - v_6 - v_7, \\ y_5' &= -v_6, & y_{13}' &= -v_3 + v_8 + v_9, \\ y_6' &= v_8, & y_{14}' &= v_7 - v_{10} - v_{11} + v_{12}, \\ y_7' &= -v_2 - v_3 + v_5 - v_9 - v_{10}, & y_{15}' &= v_6 + v_8 - v_9 + v_{10}, \\ y_8' &= v_2 + v_3 - v_4 - v_5 + v_9 + v_{10} - v_{11} + v_{12}, & y_{16}' &= -v_1 + v_5, \end{aligned}$$

where $v = (v_r)_{1\le r\le 12}$ denotes the reaction rate vector, depending on y.

More precisely, we have

$$\begin{aligned} v_1 &= k_1 y_1 y_2 y_{16}, & v_2 &= k_2 y_7 y_{11}, \\ v_3 &= k_3 y_7 y_{13}, & v_4 &= k_4 y_8 y_{12}, \\ v_5 &= k_5 y_8, & v_6 &= k_6 y_5 y_{12}, \\ v_7 &= k_7 y_{10} y_{12}, & v_8 &= k_8 y_{10}, \\ v_9 &= k_9 y_7 y_{15}, & v_{10} &= k_{10} y_7 y_{14}, \\ v_{11} &= k_{11} y_8 y_{14}, & v_{12} &= k_{12} y_9. \end{aligned}$$

The differential system can be rewritten under the form

$$y' = Sv, \tag{10}$$

where $S = (S_{i,r})_{1\le i\le 16, 1\le r\le 12}$ is the stoichiometric matrix. Equation (10) is clearly nonlinear, since each v_r nonlinearly depends on y.

Figures 8–9 show the evolution of the concentrations of all the species involved in the model, directly computed with the software Scilab, within two time scales,

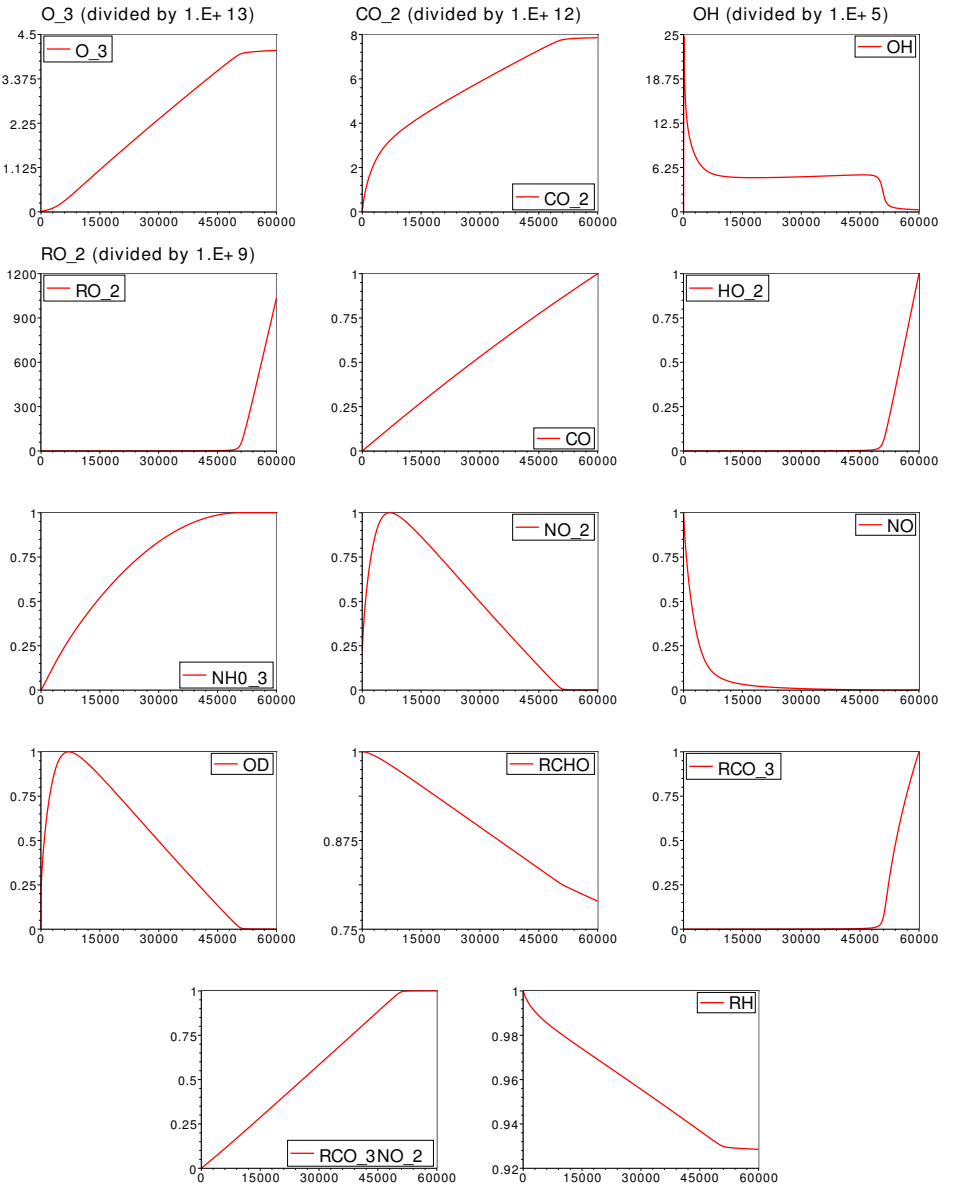


Figure 8. Ozone: Normalized concentrations of all species (final time 16 h 40 min).

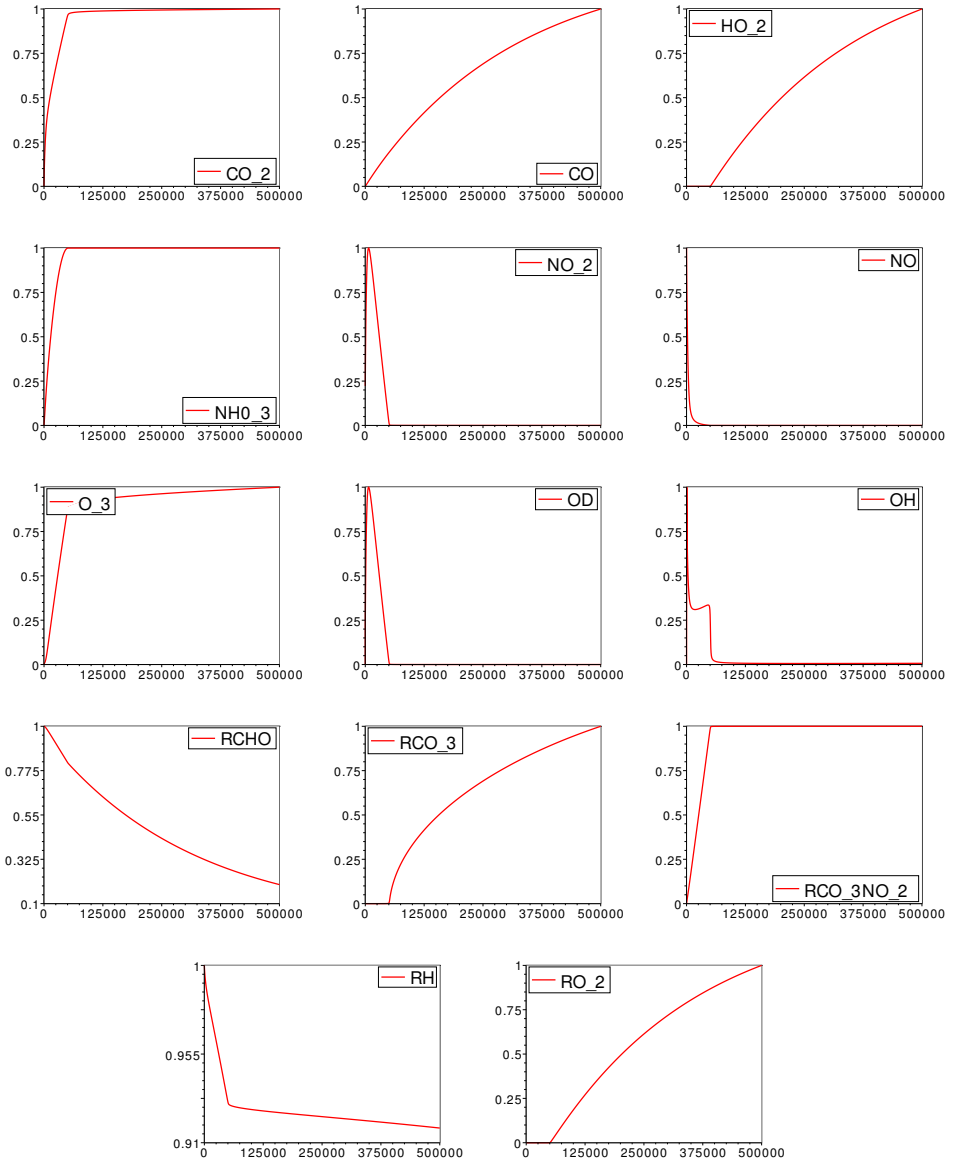


Figure 9. Ozone: Normalized concentrations of all the species (final time 5 days and 19 h).

with respective final times 16 h 40 min, and 5 days and 19 h. (Note that the model may not hold anymore at the second time scale: this is discussed in [1; 4].) The concentrations of air and O_2 are not plotted, since they remain constant. The initial values of the concentrations are:

air	O_2	CO_2	NHO_3	RH	CO	NO	NO_2
$2.45\,10^{19}$	$4.18\,10^{18}$	100	100	$5\,10^{13}$	100	$1.23\,10^{13}$	$2.5\,10^{12}$
RCO_3NO_2	RCHO	O_3	OH	HO_2	RCO_3	RO_2	OD
100	$5\,10^{13}$	100	200	100	300	200	100

We obviously recover the results from [1] with the same set of initial data.

4.1.2. *Stoichiometric invariants.* Since the rank of matrix S is 10, there are six stoichiometric invariants in the model, which can be chosen as follows [1]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\, y_1 = 0, \tag{11}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\, y_2 = 0, \tag{12}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(y_4 + y_7 + y_8 + y_9) = 0, \tag{13}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(y_5 + y_9 + y_{10} + y_{14} + y_{15}) = 0, \tag{14}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(y_4 - 2y_6 + y_9 + y_{12} + y_{13} + y_{14} + y_{15}) = 0, \tag{15}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(-3y_3 - 3y_6 - y_7 - 2y_9 - 2y_{10} + y_{11} + y_{13} - 2y_{14} + y_{16}) = 0. \tag{16}$$

The first two invariants are immediate, and equations (13)–(14) come from the conservation of species involving N and R radicals. Invariants (15)–(16) are numerically controlled with the same computation. More precisely, we can see in Figure 10 that the invariant (15) is not as well conserved as invariant (16). This is a consequence of the computation. Indeed, some concentrations involved in (15)–(16) are of very high order of magnitude ($\sim 10^{14}$). The numerical variations of a

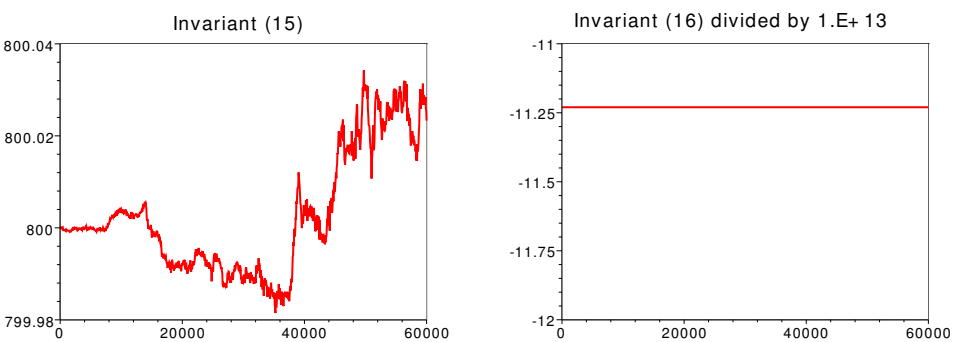


Figure 10. Ozone: conservation of invariants (15) and (16).

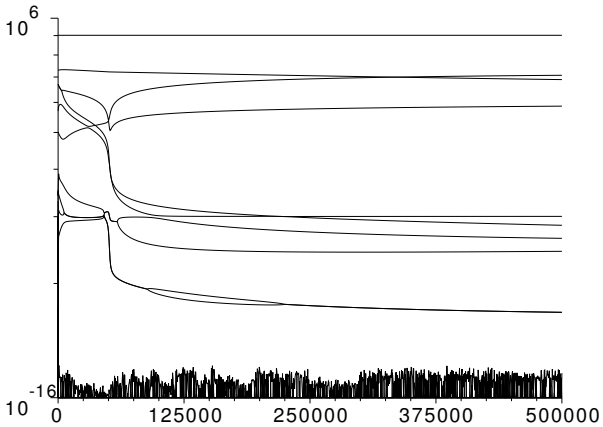


Figure 11. Ozone: evolution of the spectrum of the Jacobian matrix.

linear combination of such quantities, whose value is approximately 10^3, clearly involve unavoidable numerical errors.

4.1.3. *Stiffness.* We recall that the differential system (10) is stiff if there are some eigenvalues of its Jacobian matrix $(\partial_{y_i}(Sv)_j)_{1\leq i,j\leq 16}$ whose real parts are not of the same order of magnitude with respect to the other eigenvalues. We can check on Figure 11 that there are mainly three orders of magnitude for the eigenvalues. Note that there are a lot of oscillations for the smaller eigenvalues, again due to numerical errors.

4.2. ***A reduction method: the quasisteady states.*** There is no systematic method to obtain, from a nonlinear stiff differential system, a reduced model giving fine numerical approximations at any time. One solution consists in linearizing the system in a neighborhood of a given stationary point [6]. Other possibilities exist, such as the quasisteady state assumption on some species, or the partial equilibrium assumption for some reactions [27].

In the section, we focus on the quasisteady state method. Before applying it to the ozone model, let us briefly recall its mechanism. Denote by A an intermediary compound in a given chain of reactions. The evolution of its concentration is governed by

$$\frac{\mathrm{d}y_A}{\mathrm{d}t} = p - c,$$

where p and c are respectively the production and consumption of A. The species A is in a quasisteady state when its production rate is very close to its destruction rate, more precisely, if the quasistationary index, defined by

$$\mathcal{I}_A = \frac{|p-c|}{p+c},$$

is such that $\mathcal{I}_A \ll 1$. This index only gives an a posteriori criterion to select the quasisteady state species. Moreover, the quasisteady state of the species A does not mean that the concentration of A is constant.

In [1], using the a posteriori criterion defined above, one gets five quasistationary species in the ozone model: OH, HO_2, RCO_3, RO_2 and OD, for times smaller than 16 h and 40 min. In that situation, the concentration of OD, which is now denoted z_{16}, can be directly computed in terms of $(y_i)_{1\le i\le 11}$:

$$z_{16} = \frac{k_5\ y_8}{k_1\ y_1\ y_2}.$$

The concentrations of the other quasistationary species, also denoted $(z_i)_{12\le i\le 15}$, depend on each other. In fact, $(z_i)_{13\le i\le 15}$ can be written in terms of z_{12}, more precisely, we have

$$z_{15} = \frac{(k_4\ y_8 + k_6\ y_5 + k_7\ y_{10})z_{12} - k_8\ y_{10}}{k_9\ y_7}, \quad z_{14} = \frac{(k_4\ y_8 + k_7\ y_{10})z_{12} - 2k_8\ y_{10}}{k_{10}\ y_7},$$

$$z_{13} = \frac{(k_4\ y_8 + k_6\ y_5 + k_7\ y_{10})z_{12}}{k_3\ y_7},$$

with

$$z_{12} = \frac{2k_8\ y_{10}(k_{10}\ y_7 + k_{11}\ y_8) + k_{10}\ k_{12}\ y_7\ y_9}{(k_4\ y_8 + k_7\ y_{10})(k_{10}\ y_7 + k_{11}\ y_8) - k_7\ k_{10}\ y_7\ y_{10}}.$$

The five previous equalities come from (10), where we put $(Sv)_i = 0$ for $12 \le i \le 16$.

Note that, beyond 16 h and 40 min, the quasistationary species are not the same, and that the model itself does not hold anymore. Hence, in the sequel, we only present computations on times smaller than 16 h and 40 min, when we make the quasisteady state assumption.

4.3. ***Numerical tests.*** We apply the parareal algorithm (4) to solve the equations of the ozone model. We use the implicit Euler scheme for the fine and the coarse solvers, with different time steps. The parareal iterations are stopped as soon as the sum of the relative errors on each concentration (between parareal iterations k and $k+1$) is smaller than the numerical tolerance, which is set to 0.05. In the sequel, we only focus on the concentrations of four species: O_3, CO_2, OH and RO_2, for the sake of simplicity. Of course, the behaviors of the remaining species concentrations have been checked too.

4.3.1. *Parareal algorithm vs. fine algorithm: Test 1.* We compute the solution in the interval [0, 16 h 40 min]. One processor was used to solve the problem in the tiny interval [0, 0.01 h] to capture the first boundary layers, 9 processors for the computation in [0.01 h, 2 h 45 min], and 10 processors for the computation in [2 h 45 min, 16 h 40 min]. The parareal algorithm converges after only 1 iteration and the parareal computation is about 18 times faster than the fine computation

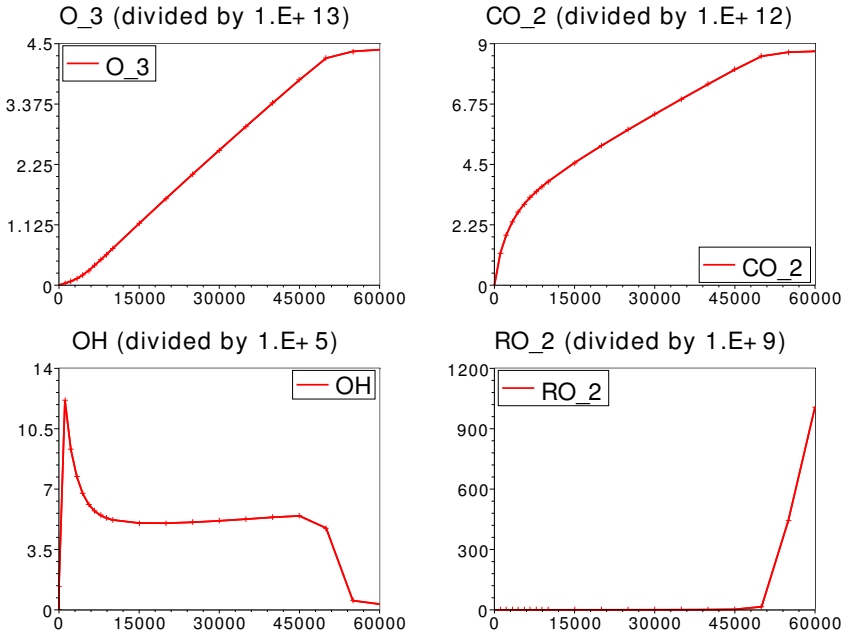


Figure 12. Ozone: fine (solid) and parareal (+) solutions on [0, 16 h 40 min].

obtained with the fine solver on the whole interval [0, 16 h 40 min]. We display the evolution of the species of interest on Figure 12.

4.3.2. *Parareal algorithm vs. fine algorithm: test 2.* We compute the solution over a long period of time [0, 5d19h]. The parareal computation converges after 2 iterations. It is 31 times faster than the fine computation on the whole interval [0, 5d19h]. Let us precise that we have used 170 processors for these computations: only one processor on [0, 0.01], 9 on [0.01, 1], 10 on [1, 10] and 150 processors on [10, 5d19h]. The concentrations of some species are shown on Figure 13.

4.3.3. *Parareal algorithm coupled with reduction vs. fine algorithm.* As already noted in Section 4.2, we focus on the time interval [0, 16 h 40 min]. The computational parameters are the same as in Section 4.3.1. The parareal algorithm is coupled with the QSS reduction, that is, the coarse solver uses the full system up to 2h45min, and the reduced system beyond that time, whereas the fine solver remains the same. The algorithm converges after 3 iterations. The parareal computation is about 4 times faster than the fine computation obtained with the fine scheme on the whole interval [0, 16 h 40 min]. The evolution of some species is shown on Figure 14 and the stoichiometric invariants are quite well conserved, see Figure 15. Note that the relative errors are at most of order 10^{-4}.

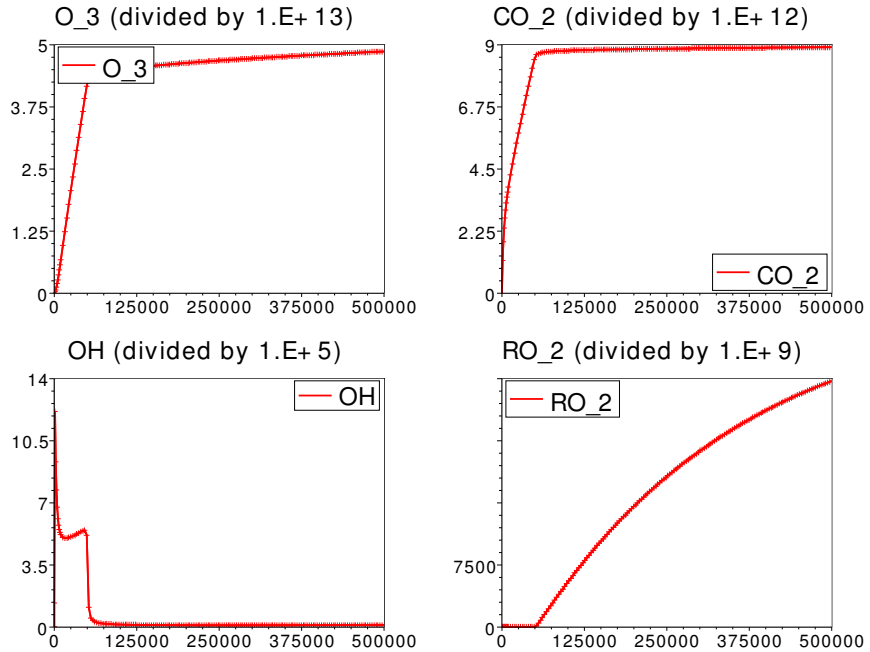


Figure 13. Ozone: fine (solid) and parareal (+) solutions on [0, 5 d 19 h].

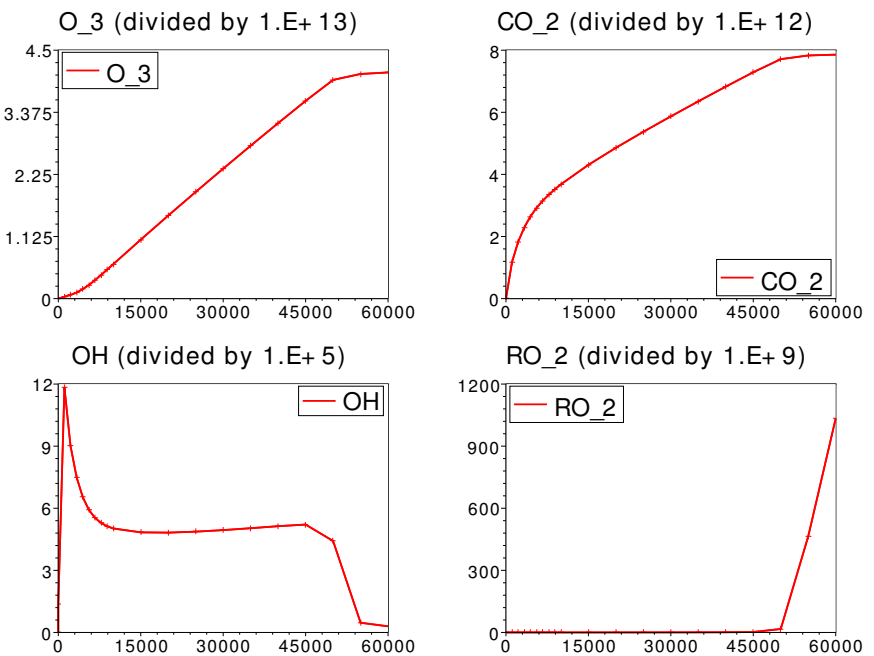


Figure 14. Ozone: fine (solid) and reduced parareal (+) solutions on [0, 16 h 40 min].

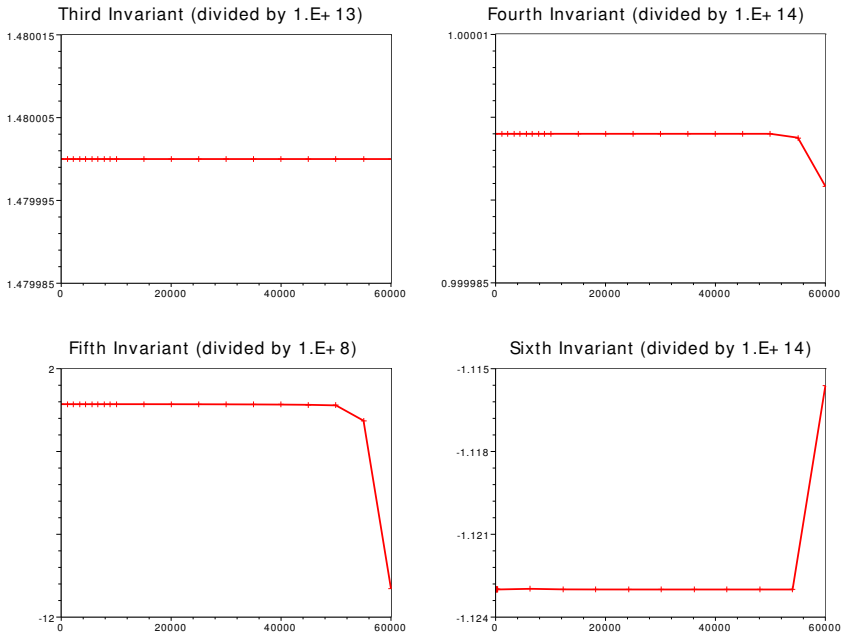


Figure 15. Ozone: fine (solid) and reduced parareal (+) stoichiometric invariants on [0, 16 h 40 min].

In order to decrease the numerical errors on Figure 15, we proceed in the same way as in Section 3.3.4 to modify the parareal algorithm, that is, we add an orthogonal projection to ensure a better conservation of the stoichiometric invariants; see Figure 16.

5. Concluding remarks

In chemical kinetics, the parareal algorithms coupled with reduction methods provide an essential tool to solve stiff differential systems with accuracy. Numerical tests point out the efficiency of the parareal approach in both linear and nonlinear cases. Let us note that one can ensure a better numerical conservation of the stoichiometric invariants by adding a projection in the standard parareal algorithm.

Acknowledgement

The authors thank the referees for suggestions resulting in a significant improvement to some sections, and the IFP (Institut Français du Pétrole — mainly people from its Applied Mathematics Department), for providing very helpful reports about the ozone model. They are also grateful to Yvon Maday, who suggested this work.

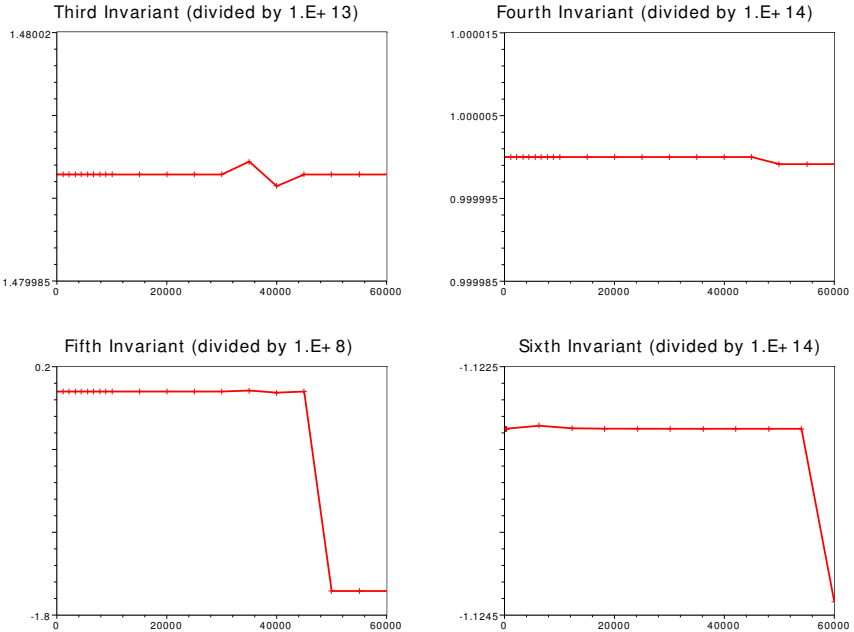


Figure 16. Ozone: fine (solid) and reduced parareal with projection (+) stoichiometric invariants on [0, 16 h 36 min].

References

[1] P. Ayoub, A. Bamberger, and Z. Benjelloun-Dabaghi, *Étude mathématique et numérique complète pour la réduction d'un schéma cinétique de production d'ozone*, technical report, Institut Français du Pétrole, 1994.

[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

[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. Pavarino and A. Toselli, eds.), Lect. Notes Comput. Sci. Eng., no. 23, Springer, Berlin, 2002, pp. 189–202. MR 1962689

[4] A. Bamberger and Z. Benjelloun-Dabaghi, *Étude mathématique d'un schéma cinétique de la production de l'ozone*, technical report, Institut Français du Pétrole, 1994.

[5] A. Blouza, F. Coquel, and F. Hamel, *Reduction of linear kinetic systems with multiple scales*, Combust. Theory Model. **4** (2000), no. 3, 339–362. MR 2001h:80009

[6] V. I. Bykov, V. I. Dimitri, and A. N. Gorban, *Marcelin-de Donder kinetics near equilibrium*, React. Kin. Catal. Lett. **12** (1979), no. 1, 19–23.

[7] S. L. Campbell and N. J. Rose, *Singular perturbation of autonomous linear systems*, SIAM J. Math. Anal. **10** (1979), no. 3, 542–551. MR 80i:34089a

[8] M. Coderch, A. S. Willsky, S. S. Sastry, and D. A. Castañon, *Hierarchical aggregation of linear systems with multiple time scales*, IEEE Trans. Automat. Control **28** (1983), no. 11, 1017–1030. MR 84k:93007

[9] 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

[10] C. Farhat, J. Cortial, C. Dastillung, and H. Bavestrello, *Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses*, Internat. J. Numer. Methods Engrg. **67** (2006), no. 5, 697–724. MR 2007a:74041

[11] 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

[12] M. Gander and M. Petcu, *Analysis of a Krylov subspace enhanced parareal algorithm for linear problems*, Paris–Sud Working Group on Modelling and Scientific Computing 2007–2008 (E. Cancés et al., eds.), ESAIM Proc., no. 25, EDP Sci., Les Ulis, 2008, pp. 114–129. MR 2010i:65119

[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] M. J. Gander and E. Hairer, *Nonlinear convergence analysis for the parareal algorithm*, Domain decomposition methods in science and engineering XVII (M. Bercovier, ed.), Lect. Notes Comput. Sci. Eng., no. 60, Springer, Berlin, 2008, pp. 45–56. MR 2009j:65165

[15] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput. **29** (2007), no. 2, 556–578. MR 2008c:65386

[16] ———, *On the superlinear and linear convergence of the parareal algorithm*, Domain decomposition methods in science and engineering XVI (O. Widlund and D. Keyes, eds.), Lect. Notes Comput. Sci. Eng., no. 55, Springer, Berlin, 2007, pp. 291–298. MR 2008g:65103

[17] S. H. Lam and D. A. Goussis, *The CSP method for simplifying kinetics*, Int. J. Chem. Kinet. **26** (1994), 461–486.

[18] 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

[19] Y. Liu and J. Hu, *Modified propagators of parareal in time algorithm and application to Princeton ocean model*, Internat. J. Numer. Methods Fluids **57** (2008), no. 12, 1793–1804. MR 2009g:86009

[20] U. Maas and S. B. Pope, *Simplifying chemical kinetics: intrinsic low-dimensional manifolds in composition space*, Combust. flame **88** (1992), 239–264.

[21] Y. Maday, *Parareal in time algorithm for kinetic systems based on model reduction*, High-dimensional partial differential equations in science and engineering (A. Bandrauk et al., eds.), CRM Proc. Lecture Notes, no. 41, Amer. Math. Soc., Providence, RI, 2007, pp. 183–194. MR 2009b:65162

[22] Y. Maday, J. Salomon, and G. Turinici, *Monotonic parareal control for quantum systems*, SIAM J. Numer. Anal. **45** (2007), no. 6, 2468–2482. MR 2008k:81003

[23] H. Mak and J. J. DiStefano, *Optimal control policies for the prescription of thyroid hormones*, Math. Biosciences **42** (1978), 159–186.

[24] B. Sportisse, *Modélisation et simulation de la pollution atmosphérique*, habilitation à diriger des recherches, Université Pierre et Marie Curie, Paris, 2007.

[25] G. A. Staff and E. M. Rønquist, *Stability of the parareal algorithm*, Domain decomposition methods in science and engineering (R. Kornhuber et al., eds.), Lect. Notes Comput. Sci. Eng., no. 40, Springer, Berlin, 2005, pp. 449–456. MR 2235772

[26] R. S. Varga, *Matrix iterative analysis*, expanded ed., Springer Series in Computational Mathematics, no. 27, Springer, Berlin, 2000. MR 2001g:65002

[27] F. A. Williams, *Combustion theory: the fundamental theory of chemically reacting flow systems*, Benjamin-Cummings Publishing Company, 1985.

Received October 4, 2009. Revised July 17, 2010.

ADEL BLOUZA: adel.blouza@univ-rouen.fr
Laboratoire de Mathématiques Raphaël Salem, UMR 6085 CNRS–Université de Rouen, 76801 Saint-Etienne-du-Rouvray, France

LAURENT BOUDIN: laurent.boudin@upmc.fr
Laboratoire J.-L. Lions, Université Pierre et Marie Curie, 75252 Paris cedex 05, France

SIDI MAHMOUD KABER: sidi-mahmoud.kaber@upmc.fr
Laboratoire Jacques-Louis Lions, Université Pierre et Marie Curie, 75252 Paris cedex 05, France

COMM. APP. MATH. AND COMP. SCI.
Vol. 5, No. 2, 2010

msp

A HYBRID PARAREAL SPECTRAL DEFERRED CORRECTIONS METHOD

MICHAEL L. MINION

The parareal algorithm introduced in 2001 by Lions, Maday, and Turinici is an iterative method for the parallelization of the numerical solution of ordinary differential equations or partial differential equations discretized in the temporal direction. The temporal interval of interest is partitioned into successive domains which are assigned to separate processor units. Each iteration of the parareal algorithm consists of a high accuracy solution procedure performed in parallel on each domain using approximate initial conditions and a serial step which propagates a correction to the initial conditions through the entire time interval. The original method is designed to use classical single-step numerical methods for both of these steps. This paper investigates a variant of the parareal algorithm first outlined by Minion and Williams in 2008 that utilizes a deferred correction strategy within the parareal iterations. Here, the connections between parareal, parallel deferred corrections, and a hybrid parareal-spectral deferred correction method are further explored. The parallel speedup and efficiency of the hybrid methods are analyzed, and numerical results for ODEs and discretized PDEs are presented to demonstrate the performance of the hybrid approach.

1. Introduction

The prospect of parallelizing the numerical solution of ordinary differential equations (ODEs) in the temporal direction has been the topic of research dating back at least to the early work of Nievergelt [55] and Miranker and Liniger [54]. These early papers as well as the methods described here employ multiple processing units to compute the solution over multiple time intervals in parallel. Hence, in the classification used in [13], for example, these methods are categorized as employing parallelization *across the steps* as opposed to *across the method* or *across the problem*.

Examples of approaches to parallelization across the method include the computation of intermediate or stage values in Runge–Kutta and general linear methods simultaneously on multiple processors [38; 14]. These attempts to parallelize can

MSC2000: 65L99.

Keywords: parallel in time, parareal, ordinary differential equations, parallel computing, spectral deferred corrections.

only be efficient when the number of processors being used is no larger than the number of stage values and hence typically yield modest parallel speedup.

Methods that utilize parallelization *across the problem* rely on a splitting of the problem into subproblems that can be computed in parallel and an iterative procedure for coupling the subproblems so that the overall method converges to the solution of the full problem. A well known class of methods in this style is the parallel waveform-relaxation schemes [25; 61; 18]. Despite efforts to accelerate the rate of convergence of wave-form relaxation methods [29; 34], convergence can still be slow, especially for stiff problems.

In addition to the methods in [55; 54], methods based on multiple shooting [40] also employ parallelization across the steps. In 2001, a new approach similar in spirit to multiple shooting was introduced in [45]. The so called *parareal* method is appropriate for larger numbers of processors and has sparked many new papers devoted to the subject of time parallelization. The parareal method has been further analyzed and refined [47; 22; 7; 60; 49; 30; 28; 62; 48; 32; 27; 9] and implemented for different types of applications [8; 23; 46]. This paper details a new variant of the parareal method first outlined in [53] that utilizes an iterative ODE method based on deferred corrections within the parareal iteration. As outlined below, this approach can either be thought of as a way to parallelize a deferred correction approach to solving ODEs or as a way to increase the efficiency of the parareal algorithm by removing the requirement to solve the subproblems on each processor during each iteration with a full accuracy solver.

One of the justifications of the use of parareal methods is the scenario where a specific computation must be completed in a fixed amount of time and sufficient computational resources are available. In the context of the numerical solution of time dependent PDEs, although parallelization of methods in the spatial dimensions has seen a tremendous amount of successful research, for a fixed problem size, spatial parallel speedup will eventually saturate as more processors are employed. If additional processors are available, then additional parallelization in the temporal direction could reduce the overall parallel computational cost. The name *parareal* in fact is derived from *parallel* and *real time* and encapsulates the desire to complete a computation of a specific size faster in real time. This is in contrast to the common practice of reporting spatial parallel efficiency or speedup in the context of increasing problem size as the number of processors are increased. The current work is motivated by the desire to develop efficient methods for time-space parallelization for PDEs.

As discussed in detail in Section 3, it has become standard to describe the parareal algorithm in terms of two computational methods to approximate the temporal evolution of the equation over a fixed time interval. A fine, or accurate, method (denoted here by $\mathcal{F}$) computes an accurate (and hence more computationally

expensive) approximation to the solution. A coarse or less accurate method (denoted here by $\mathcal{G}$) is also defined, and is used in a serial fashion to propagate a correction through the time domain. We will refer to $\mathcal{G}$ and $\mathcal{F}$ as the coarse and fine propagators respectively. The parareal method alternates between the parallel application of $\mathcal{F}$ on multiple time domains using approximate initial conditions and a serial sweep using $\mathcal{G}$ that propagates a correction to the initial conditions throughout the time interval. Upon convergence, the accuracy of the parareal method is limited by what one would obtain if the $\mathcal{F}$ method was used in serial on each subdomain. Hence, as explained in detail in Section 5, in order for the parareal method to achieve parallel efficiency, $\mathcal{G}$ must be less expensive than $\mathcal{F}$. The reduced computational cost of $\mathcal{G}$ can be achieved by using a coarser time step for $\mathcal{G}$ than for $\mathcal{F}$, or a less expensive numerical method, or both. As has been pointed out [8; 7; 23; 26], for PDEs, it is also possible to use a coarser spatial discretization for $\mathcal{G}$. The parareal algorithm can, in principle, use any self-starting ODE method for $\mathcal{F}$ and $\mathcal{G}$ and hence can be used in a "black box" fashion.

As detailed in Section 5, the parallel efficiency of parareal is limited by the fact that during each iteration, the parallel application of $\mathcal{F}$ has the same computational cost as the serial algorithm applied to the subdomain. Hence, a significant parallel speed up (the ratio of serial to parallel cost) can only be achieved if the number of iterations required to converge to the serial solution to a given tolerance is significantly smaller than the number of subdomains. Similarly, the parallel efficiency (speedup divided by the number of processors), is bounded above by the reciprocal of the number of iterations regardless of the cost of $\mathcal{G}$. In most instances, the total computational cost of the parareal algorithm is dominated by the cost of the $\mathcal{F}$.

In [53], a new variant of the parareal method is presented that uses an iterative method for solving ODEs for the coarse and fine propagators rather than traditional methods like Runge–Kutta (RK) that are typically used in the literature. The key observation in [53] is that the $\mathcal{F}$ propagator in traditional parareal approaches makes no use of the previously computed solution on the same interval (a recent alternative approach to reusing information appears in [28]). It is shown how the use of an iterative method can be combined with parareal to improve the solution from the previous parareal iteration rather than computing a solution from scratch. The result is that the $\mathcal{F}$ propagator becomes much cheaper than a full accuracy solution on the interval, and in fact the dominant cost in the numerical tests in [53] becomes the $\mathcal{G}$ propagator.

The numerical method in [53] is based on the method of spectral deferred corrections (SDC) [21]. Since SDC methods converge to the solution of the Gaussian collocation formula, very accurate solutions can be obtained using a modest number of SDC substeps per time step. This accuracy is offset by the relatively high computational cost of SDC methods *per time step*. However, when one compares

the computational cost for a given (sufficiently small) error tolerance, SDC methods have been shown to compare favorably to RK schemes. This is especially true in the case of problems for which semi-implicit or implicit-explicit (IMEX) methods are appropriate since very higher-order IMEX RK methods have not been developed whereas IMEX SDC methods with arbitrarily high formal order of accuracy are easily constructed [51]. One main point of this paper is that the relatively high cost per time step of SDC methods can be effectively amortized when SDC methods are combined with the parareal algorithm since only one (or a few) SDC iterations are done during each parareal iteration. This means that the cost of the $\mathcal{F}$ propagator is similar to that of a modest number of steps of a low-order method rather than many steps of a higher-order method. Differences between the hybrid parareal/SDC approach and recent parallel deferred correction methods [33; 16] are discussed in Section 4.1.

The preliminary numerical results included in Section 6 suggest that the use of a single SDC iteration in lieu of a full-accuracy $\mathcal{F}$ propagator does not significantly affect the convergence behavior of the parareal iteration. Rather, the accuracy of $\mathcal{G}$ determines the rate of convergence (as was proven for the parareal method in [27]). Since using a single SDC iteration is markedly less expensive than a higher-order RK method with finer time steps, the dominant cost of the parareal/SDC hybrid method when many processors are used becomes the serial procedure for initializing the solution on each processor (typically done with $\mathcal{G}$). Hence for PDEs, the possibility of reducing the cost of $\mathcal{G}$ by using a coarser spatial discretization (already proposed in [8; 7; 23; 26]) is very attractive. This idea will be pursued in a sequel to this paper.

2. Spectral deferred corrections

The spectral deferred correction method (SDC) is a variant of the traditional deferred and defect correction methods for ODEs introduced in the 1960s [63; 57; 58; 19]. The original methods never gained the popularity of Runge–Kutta or linear multistep methods, however, a series of papers beginning in 2000 has rekindled interest in using such methods for large scale physical simulations. The SDC method introduced in [21] couples a Picard integral formulation of the correction equation with spectral integration rules to achieve stable explicit and implicit methods with arbitrarily high formal order of accuracy.

SDC methods possess two characteristics that make them an attractive option for the temporal integration of complex physical applications. First, SDC methods with an arbitrarily high formal order of accuracy and good stability properties can easily be constructed. Second, the SDC framework provides the flexibility to apply different time-stepping procedures to different terms in an equation (as in

operator splitting methods) while maintaining the high formal order of accuracy. This second property has led to the development of semi- and multiimplicit methods for equations with disparate time scales [50; 52; 10; 42]. Since SDC methods are nonstandard, the original SDC method will be reviewed in the next section followed by a brief discussion in Section 2.2 of the semi-implicit variants that are used in the second numerical example in Section 6.2.

2.1. ***Original spectral deferred corrections.*** Consider the ODE initial value problem

$$\begin{aligned} u'(t) &= f(t, u(t)), \quad t \in [0, T], \\ u(0) &= u_0, \end{aligned}$$

where $u_0, u(t) \in \mathbb{C}^N$ and $f : \mathbb{R} \times \mathbb{C}^N \to \mathbb{C}^N$. This equation is equivalent to the Picard integral equation

$$u(t) = u_0 + \int_0^t f(\tau, u(\tau))\, d\tau, \tag{1}$$

and this latter form is used extensively in the discussion that follows.

As with traditional deferred correction methods, a single time step $[t_n, t_{n+1}]$ of size $\Delta t = t_{n+1} - t_n$ is divided into a set of intermediate substeps by defining $\boldsymbol{t}_n = [t_1, \ldots, t_J]$ with $t_n \le t_1 < \cdots < t_J \le t_{n+1}$; however, for SDC methods, $\boldsymbol{t}_n$ corresponds to Gaussian quadrature nodes. The intermediate times in $\boldsymbol{t}_n$ will be denoted by the subscript j with $j = 1 \ldots J$, and numerical approximations of quantities at time t_j will likewise carry the subscript j. Beginning with the initial condition for the time step $U_{n,1} \approx u(t_n)$, a provisional approximation $\boldsymbol{U}_n^0 = [U_{n,1}^0, \ldots, U_{n,J}^0]$ is computed at the intermediate points using a standard numerical method. The superscripts on numerical values (for example $\boldsymbol{U}_n^0$) denote here the iteration number in the SDC procedure. The continuous counterpart of $\boldsymbol{U}_n^0$ can be constructed by standard interpolation theory and is represented as $U_n^0(t)$. Using $U_n^0(t)$, an integral equation similar to (1) for the error $\delta(t) = u(t) - U_n^0(t)$ is then derived

$$\delta(t) = \int_{t_n}^t \left[f(\tau, U_n^0(\tau) + \delta(\tau)) - f(\tau, U_n^0(\tau)) \right] d\tau + \epsilon(t), \tag{2}$$

where

$$\epsilon(t) = U_n + \int_{t_n}^t f(\tau, U_n^0(\tau))\, d\tau - U_n^0(t). \tag{3}$$

Note that $\epsilon(t_j)$ can be accurately and stably approximated using spectral integration [31], since the provisional solution $U^0(t)$ is known at the Gaussian quadrature

nodes. An update form of (2) is

$$\delta(t_{j+1}) = \delta(t_j) + \int_{t_j}^{t_{j+1}} \left[f(\tau, U_n^0(\tau) + \delta(\tau)) - f(\tau, U_n^0(\tau)) \right] d\tau + \epsilon(t_{j+1}) - \epsilon(t_j). \quad (4)$$

In order to discretize (4), an approximation to the integral term in

$$\epsilon(t_{j+1}) - \epsilon(t_j) = \int_{t_j}^{t_{j+1}} f(\tau, U_n^0(\tau)) d\tau - U_n^0(t_{j+1}) + U_n^0(t_j). \quad (5)$$

must be constructed. This is done using spectral integration representing quadrature at the nodes $\boldsymbol{t}_n$. The spectral quadrature approximation is denoted by

$$S_j^{j+1} f(\boldsymbol{t}_n, \boldsymbol{U}_n^0) \approx \int_{t_j}^{t_{j+1}} f(\tau, U_n^0(\tau)) d\tau, \quad (6)$$

and the computation of the values $S_j^{j+1} f(\boldsymbol{t}_n, \boldsymbol{U}_n^0)$ is a matrix-vector multiplication using a precomputed integration matrix (see [35] for details).

A low-order method is then applied to approximate (2) at the points $\boldsymbol{t}_n$ resulting in a correction to the provisional solution. For example, an explicit time-stepping scheme similar to the forward Euler method is

$$\delta_{j+1}^0 = \delta_j^0 + \Delta t_j [f(t_j, U_{n,j}^0 + \delta_j^0) - f(t_j, U_{n,j}^0)] + S_j^{j+1} f(\boldsymbol{t}_n, \boldsymbol{U}_n^0) - U_{n,j+1}^0 + U_{n,j}^0, \quad (7)$$

where $\Delta t_j = t_{j+1} - t_j$ and again subscripts on numerical values denote approximations corresponding the t_j. Similarly, an implicit method similar to the backward Euler method is

$$\delta_{j+1}^0 = \delta_j^0 + \Delta t_j [f(t_{j+1}, U_{n,j+1}^0 + \delta_{j+1}^0) - f(t_{j+1}, U_{n,j+1}^0)] + S_j^{j+1} f(\boldsymbol{t}_n, \boldsymbol{U}_n^0) - U_{n,j+1}^0 + U_{n,j}^0. \quad (8)$$

The correction (2) can also be approximated by higher-order methods [41; 17].

The provisional numerical solution is then updated by adding to it the approximation of the correction, that is, $U_{n,j}^1 = U_{n,j}^0 + \delta_{n,j}^0$. The SDC method then proceeds iteratively, by recomputing the residuals, approximating a new correction, and setting $U_{n,j}^{k+1} = U_{n,j}^k + \delta_{n,j}^k$. Each SDC iteration raises the formal order of accuracy of the numerical solution by the order of the approximation to (4) provided the quadrature rule in (6) is sufficiently accurate. In the methods used in the numerical experiments presented here, (2) is approximated with a first-order method so that M total SDC sweeps (including the predictor) are needed for M-th order accuracy.

An alternative form of (8) for general k can be derived using $U_{n,j}^{k+1} = U_{n,j}^k + \delta_{n,j}^k$:

$$U_{n,j+1}^{k+1} = U_{n,j}^{k+1} + \Delta t_j \left(f(t_{j+1}, U_{n,j+1}^{k+1}) - f(t_{j+1}, U_{n,j+1}^k) \right) + S_j^{j+1} f(\boldsymbol{t}_n, \boldsymbol{U}_n^k), \quad (9)$$

This form of the update equation is compared below to the serial step in the parareal algorithm.

2.2. ***Semiimplicit methods.*** SDC methods are particularly well-suited for the temporal integration of ODEs which can be split into stiff and nonstiff components. When both stiff and nonstiff terms appear in the equation, it is often significantly more efficient to use methods that treat only the stiff terms implicitly and treat nonstiff terms explicitly. Such methods are usually referred to as semi-implicit or IMEX (implicit-explicit) methods. IMEX linear multistep methods that build on a backward difference formula treatment of the stiff term have been developed [5; 24; 1; 2; 3; 37], but like backward difference formula methods themselves, the stability of these methods deteriorates as the order increases, and methods above sixth-order are unstable (see the discussion in [44]). IMEX or additive Runge–Kutta methods have also been proposed [59; 64; 4; 15; 39; 56], but methods with order higher than five have not yet appeared.

To derive IMEX SDC methods, consider the ODE

$$u'(t) = f(t, u(t)) = f_E(t, u(t)) + f_I(t, u(t)), \quad t \in [0, T], \tag{10}$$

$$u(0) = u_0. \tag{11}$$

Here the right hand side of the equation is split into two terms, the first of which is assumed to be nonstiff (and hence treated explicitly), and the second of which is assumed to be stiff (and treated implicitly). A first-order semi-implicit method for computing an initial solution is simply

$$U^0_{n,j+1} = U^0_{n,j} + \Delta t_j \big(f_E(t_j, U^0_{n,j}) + f_I(t_{j+1}, U^0_{n,j+1})\big). \tag{12}$$

Following the same logic used to derive (2), one arrives at the correction equation

$$\delta(t) = \int_0^t \big[f_E(\tau, U^0(\tau)+\delta(\tau)) - f_E(\tau, U^0_n(\tau)) + f_I(\tau, U^0_n(\tau)+\delta(\tau)) - f_I(\tau, U^0_n(\tau))\big]\, d\tau + \epsilon(t), \tag{13}$$

where

$$\epsilon(t) = U_0 + \int_0^t f_E(\tau, U^0_n(\tau)) + f_I(\tau, U^0_n(\tau))d\tau - U^0_n(t). \tag{14}$$

A simple semi-implicit time-stepping method analogous to (9) is then

$$U^{k+1}_{n,j+1} = U^{k+1}_{n,j} + \Delta t_j \big(f_E(t_j, U^{k+1}_{n,j}) - f_E(t_j, U^k_{n,j}) + f_I(t_{j+1}, U^{k+1}_{n,j+1}) - f_I(t_{j+1}, U^k_{n,j+1})\big) + S^{j+1}_j f(\boldsymbol{t}_n, \boldsymbol{U}^k_n). \tag{15}$$

In [50; 52], such a semi-implicit version of SDC is used in combination with an auxiliary variable projection method approach for the Navier–Stokes equations that treats the viscous terms implicitly and the nonlinear advective terms explicitly.

These methods have been subsequently examined in more detail in [50; 44; 42; 41], and the second numerical example in Section 6.2 is based on this semi-implicit approach. Semi-implicit SDC methods have been extended to treat three or more terms (explicit or implicit) in (10), including modifications that allow different terms in the equation to be treated with different time steps [10; 42; 11].

2.3. ***Computational cost and storage.*** Like any numerical method, SDC has disadvantages as well as advantages. The price one pays to achieve the flexibility and high order of accuracy for SDC methods is primarily in the large computational cost *per time step*. SDC methods that use a first-order numerical method in the correction iterates require M total iterations per time step (provisional and correction) to achieve formal M-th order accuracy. Since each iteration sweep requires that the solution be computed at a number of substeps that is proportional to the order, the number of function evaluations per time step grows quadratically with the order. This makes the cost per time step appear quite high compared to linear multistep or Runge–Kutta methods.

However, relying on the number of function evaluations per time step as a measure of efficiency is very misleading. First, the stability region of SDC methods also increases roughly linearly with the order, so that larger substeps can be taken as the order increases [21; 43]. In addition, a more relevant measure of cost is in terms of computational effort versus error, and as is demonstrated in Section 6, SDC methods compare well with higher-order RK methods in this measure (see also comparisons in [51; 42]). For equations with both stiff and nonstiff terms, there are no semi- or multiimplicit methods based on RK or linear multistep methods with order of accuracy greater than six, so particularly when a small error is required, higher-order SDC method are very attractive (see Section 6.2). Additionally, techniques to reduce the computational cost of SDC methods by accelerating the convergence have also appeared [35].

In the current context, however, it is the parallel cost of the time integration method that is of interest. One of the main results of this paper is that, when combined with the parareal strategy, the high cost per time step of SDC methods due to the need to iterate the correction equation is amortized over the iterations that must be performed during the parareal methods. Hence the cost of SDC methods per parareal iteration is much smaller than for a noniterative method like RK.

As previously mentioned, SDC methods require that function values be stored at a set of substeps within a given time step, which correspond to quadrature nodes of the Picard integral discretization. If a semi- or multiimplicit operator splitting treatment is used, each split piece of the function must be stored separately. Since the number of substeps grows linearly with the order of the method, the storage costs are comparable to higher-order Runge–Kutta or linear multistep methods.

3. The parareal method

The parareal method was introduced in 2001 by Lions, Maday and Turinici [45] and has sparked renewed interest in the construction of time parallel methods. In this section, a short review of the parareal method is provided, and then comparisons between parareal and spectral deferred corrections will be explored.

3.1. ***Notation and method.*** The general strategy for the parareal method is to divide the time interval of interest $[0, T]$ into N intervals with each interval being assigned to a different processor. To simplify the discussion, assume that there are N processors $\boldsymbol{P}_0$ through $\boldsymbol{P}_{N-1}$, and that the time intervals are of uniform size $\Delta T = T/N$ so that the n-th processor computes the solution on the interval $[t_n, t_{n+1}]$ where $t_n = n\Delta T$. On each interval, the parareal method iteratively computes a succession of approximations $U_{n+1}^k \approx u(t_{n+1})$, where k denotes the iteration number.

It is becoming standard to describe the parareal algorithm in terms of two numerical approximation methods denoted here 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 (1) from t_n to t_{n+1}. For example, if $\mathcal{G}$ is defined by the forward Euler method applied to (1) , then

$$\mathcal{G}(t_{n+1}, t_n, U_n) = U_n + (t_{n+1} - t_n) f(t_n, U_n). \tag{16}$$

As discussed below, 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; hence, in practice, $\mathcal{G}$ is usually a low-order method. Note that $\mathcal{G}$ or $\mathcal{F}$ could be defined to be more than one step of a particular numerical method on the interval $[t_n, t_{n+1}]$. 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^0 for $n = 1 \ldots N$ often performed with the coarse propagator $\mathcal{G}$, that is,

$$U_{n+1}^0 = \mathcal{G}(t_{n+1}, t_n, U_n^0) \tag{17}$$

with $U_0^0 = u(0)$. Alternatively, one could use the parareal method with a coarser time step to compute the initial approximation [8; 7]. Once each processor $\boldsymbol{P}_n$ has a value U_n^0, the processors can in parallel compute the approximation $\mathcal{F}(t_{n+1}, t_n, U_n^0)$. This step is in spirit an accurate solution of the ODE on the interval $[t_n, t_{n+1}]$ using the approximate starting value U_n^0. Lastly, the parareal algorithm computes the serial correction step

$$U_{n+1}^{k+1} = \mathcal{G}(t_{n+1}, t_n, U_n^{k+1}) + \mathcal{F}(t_{n+1}, t_n, U_n^k) - \mathcal{G}(t_{n+1}, t_n, U_n^k), \tag{18}$$

for $n = 0 \ldots N-1$. The parareal method proceeds iteratively alternating between the parallel computation of $\mathcal{F}(t_{n+1}, t_n, U_n^k)$ and the serial computation of (18), which requires computing the $\mathcal{G}$ propagator. The calculation in (18) will be referred to as the $\mathcal{G}$ correction sweep.

Parareal is an iterative method and hence requires a stopping criteria. 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. Since each iteration of the parareal method requires the application of both $\mathcal{F}$ in parallel and $\mathcal{G}$ in serial (plus the cost of communication between processors), the parareal method can only provide parallel speedup compared to the serial $\mathcal{F}$ scheme if the number of iterations required to converge to the specified criteria (denoted here by K) is significantly less than N (see discussion in Section 5).

3.2. ***An examination of the parareal correction equation.*** Here we review the connection between deferred corrections and the parareal step defined by (18) first outlined in [53]. Both $\mathcal{F}$ and $\mathcal{G}$ are approximations to the exact update given by the Picard equation

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(\tau, u(\tau)) d\tau. \tag{19}$$

To highlight the approximation of $\mathcal{F}$ and $\mathcal{G}$ to the Picard Equation (19), we define

$$\mathcal{J}(t_{n+1}, t_n, U_n^k) = \mathcal{F}(t_{n+1}, t_n, U_n^k) - U_n^k, \tag{20}$$

$$\mathcal{Q}(t_{n+1}, t_n, U_n^k) = \mathcal{G}(t_{n+1}, t_n, U_n^k) - U_n^k, \tag{21}$$

so that

$$\mathcal{F}(t_{n+1}, t_n, U_n^k) = U_n^k + \mathcal{J}(t_{n+1}, t_n, U_n^k), \tag{22}$$

$$\mathcal{G}(t_{n+1}, t_n, U_n^k) = U_n^k + \mathcal{Q}(t_{n+1}, t_n, U_n^k). \tag{23}$$

Using these definitions, (18) can be rewritten

$$U_{n+1}^{k+1} = U_n^{k+1} + \mathcal{Q}(t_{n+1}, t_n, U_n^{k+1}) - \mathcal{Q}(t_{n+1}, t_n, U_n^k) + \mathcal{J}(t_{n+1}, t_n, U_n^k). \tag{24}$$

In the discussion leading to (9), (4) is discretized using a backward Euler type method to give a concrete example of a time stepping scheme. If $\mathcal{G}$ is similarly defined as a single step of backward Euler, then $\mathcal{Q}(t_{n+1}, t_n, U_n^k) = \Delta t f(t_{n+1}, U_{n+1}^k)$, and (24) becomes

$$U_{n+1}^{k+1} = U_n^{k+1} + \Delta t \big(f(t_{n+1}, U_{n+1}^{k+1}) - f(t_{n+1}, U_{n+1}^k) \big) + \mathcal{J}(t_{n+1}, t_n, U_n^k). \tag{25}$$

Note the similarities between this particular first-order incarnation of the parareal update and the first-order SDC substep given in (9). The two differences between

these equations are how the previous fine solution is used in the update and the fact that in the parareal update, only the solution at the end of the time interval is updated while the SDC sweep is done for each intermediate substep in the time interval. A hybrid parareal/SDC approach using deferred corrections for both the $\mathcal{F}$ propagator and the $\mathcal{G}$ correction sweep is described next.

4. Parallel and parareal SDC

4.1. ***Parallel SDC.*** In the description of SDC above, note that once the provisional solution has been computed at all the intermediate points t_j in a given time step, a provisional starting value for the next time step is available. This observation naturally leads to a time parallel version of SDC in which multiple processors are computing SDC iterations on a contiguous block of time steps simultaneously. Variations of this approach have been employed in [33; 16] to create parallel versions of deferred correction schemes. In practice, such an approach is only efficient when the number of processors is approximately the number of iterations of the serial deferred correction scheme. This then allows the first processor to finish the calculation of the first time step as the last processor is finishing the provisional solution. Then the first processor can receive a starting value from the last processor and continue forward in time. Hence, the parallel speedup in this type of parallel SDC method does not result from using an iterative strategy over the full time domain, but rather computing the necessary iterations of the SDC method on multiple processors simultaneously. In spirit, the hybrid parareal/SDC methods discussed below combine the parallel speedup obtained with parallel deferred corrections with that obtained with parareal.

4.2. ***Parareal using SDC.*** Since the parareal algorithm can in principle use any single-step ODE method for the $\mathcal{F}$ and $\mathcal{G}$ propagators, it would be straightforward to incorporate an SDC method into the parareal framework. However, in the standard parareal scheme, if the $\mathcal{F}$ propagator were an SDC method requiring M iterations in serial, then in the k-th parareal iteration, processor $\boldsymbol{P}_n$ would compute $\mathcal{F}$ using the initial value U_n^k by performing M total SDC iterations. When $k > 1$, however, it would be foolish to ignore the results of the $\mathcal{F}$ propagator from iteration $k-1$ when using SDC in iteration k.

Instead, the $\mathcal{F}$ propagator could perform one or several SDC sweeps on the solution from the previous parareal iteration (incorporating the initial condition in the first correction substep). In this approach, the fine solution is computed on each processor on the J Gaussian quadrature nodes within the time slice assigned to the processor. These values are stored from one parareal iteration to the next, and the $\mathcal{F}$ propagator is replaced by L SDC sweeps (e.g., the method described in (9) or a higher-order version thereof) over the J nodes.

As the parareal iterations converge, the fine solution on each processor converges to the high-accuracy SDC solution (in fact the spectral collocation solution [35]), but the cost of applying the $\mathcal{F}$ propagator during each iteration is that of a low-order method and less than a full step of the SDC method by a factor of M/L. Numerical experiments presented here suggest that $L = 1$ is sufficient for an efficient method. Hence, the cost of $\mathcal{F}$ is similar to J steps of a low-order method. In the numerical examples presented in Section 6, Gauss–Lobatto nodes are used for the fine solution with $J = 5$, 7 and 9.

In addition, following the argument in Section 3.2, the $\mathcal{G}$ corrector sweep can also be cast as a deferred correction sweep which both updates the initial condition for the following time step and provides a correction to the solution in the interval. However, since it is desirable to have $\mathcal{G}$ be as inexpensive as possible, the correction to the solution generated in the $\mathcal{G}$ correction sweep will generally not be available at all the fine SDC nodes. In [53] the $\mathcal{G}$ correction sweep is computed using one step of a third-order RK method applied to the correction equation, and the correction is interpolated from the 3 stage values to the SDC nodes of the fine solution. In general, $\mathcal{G}$ could be computed on a subset of the fine SDC nodes and the correction interpolated, and this approach is used in the examples presented here.

4.3. ***Parareal/SDC specifics.*** A detailed description of the parareal/SDC algorithm used in the numerical results is now presented. The algorithm in pseudocode appears in the Appendix. As in parareal, assume the time interval of interest $[0, T]$ is divided into N uniform intervals $[t_n, t_{n+1}]$ where $t_n = n\Delta T$ is assigned to $\boldsymbol{P}_n$. On each interval $[t_n, t_{n+1}]$ choose the J fine SDC nodes $\boldsymbol{t}_n$ corresponding to the Gauss–Lobatto nodes with $t_n = t_{n,1} < \cdots < t_{n,J} = t_{n+1}$. Likewise choose some subset $\tilde{\boldsymbol{t}}_n$ of size $\tilde{J}$ of $\boldsymbol{t}_n$ corresponding to the substeps for the coarse propagator $\mathcal{G}$. In the first numerical example $\tilde{J} = 3$, that is, $\tilde{\boldsymbol{t}}_n$ is the midpoint and endpoints of $[t_n, t_{n+1}]$. This situation is shown in Figure 1, where $J = 7$ and $\tilde{J} = 3$. The solution at the coarse nodes $\tilde{\boldsymbol{t}}_n$ on processor $\boldsymbol{P}_n$ during iteration k is denoted $\tilde{U}^k_{n,\tilde{j}}$, while the solution at the fine nodes is denoted $U^k_{n,j}$. One last piece of notational convention is that $f(\boldsymbol{t}_n, \boldsymbol{U}^k_n)$ refers to the set of function values $f(t_j, U^k_{n,j})$ at the nodes $\boldsymbol{t}_n$, with analogous notation using $\tilde{\boldsymbol{t}}_n$.

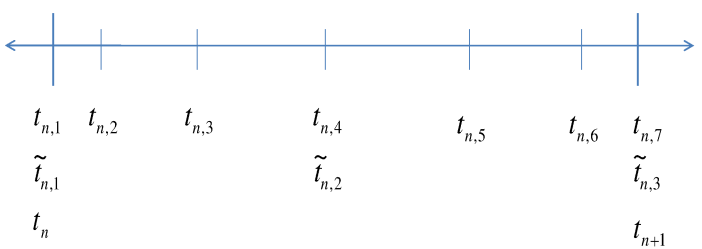


Figure 1. Notation for SDC substeps for the $\mathcal{F}$ and $\mathcal{G}$ propagators.

Predictor step. Starting with the initial data $\tilde{U}^0_{0,1} = u(0)$ on processor $\boldsymbol{P}_0$, compute the initial approximation $\tilde{U}^0_{n,1}$ on each processor in serial. Each processor (except $\boldsymbol{P}_0$) must wait to receive the value $\tilde{U}^0_{n,1} = \tilde{U}^0_{n-1,\tilde{J}}$ from processor $\boldsymbol{P}_{n-1}$. Then the values $\tilde{U}^0_{n,\tilde{j}}$ for $\tilde{j} = 1 \ldots \tilde{J}$ are computed using the numerical method of $\mathcal{G}$. The value $\tilde{U}^0_{n,\tilde{J}}$ is then passed to the processor $\boldsymbol{P}_{n+1}$ (if $n < N-1$), where it is received as $\tilde{U}^0_{n+1,1}$.

First parallel iteration $(k = 1)$**.** As soon as processor $\boldsymbol{P}_n$ is finished computing the last predictor value $\tilde{U}^0_{n,\tilde{J}}$ and sending it to $\boldsymbol{P}_{n+1}$ (if $n < N-1$), the following steps are taken in the first parallel iteration. Note that each of these steps can be done in parallel if the method is pipelined (see Section 5.2).

(1) Interpolate the values $\tilde{U}^0_{n,\tilde{j}}$ to all of the fine nodes $\boldsymbol{t}$ to form $U^0_{n,j}$ and compute $f(t_j, U^0_{n,j})$ for $j = 1 \ldots J$.

(2) Compute the values $S^{j+1}_j f(\boldsymbol{t}_n, \boldsymbol{U}^0_n)$ for $j = 1 \ldots J-1$ using the spectral quadrature rule as in (6).

(3) Perform one sweep ($J-1$ substeps) of the numerical method for the $\mathcal{F}$ propagator applied to the correction (2) using the values $f(t_j, U^0_{n,j})$ and $S^{j+1}_j f(\boldsymbol{t}_n, \boldsymbol{U}^0_n)$ just computed. This will yield updated values $U^1_{n,j}$.

(4) Compute the values $S^{\tilde{j}+1}_{\tilde{j}} f(\tilde{\boldsymbol{t}}_n, \boldsymbol{U}^1_n)$ for $\tilde{j} = 1 \ldots \tilde{J}-1$. Since the integral over a coarse time step is the sum of the integrals over the corresponding fine time steps, this is done by summing the corresponding values of the spectral integration rule $S^{j+1}_j f(\boldsymbol{t}_n, \boldsymbol{U}^1_n)$.

(5) Receive the new initial value $\tilde{U}^1_{n,1}$ from processor $\boldsymbol{P}_{n-1}$ (if $n > 0$). If $n = 0$, then $\tilde{U}^1_{0,1} = \tilde{U}^0_{0,1}$.

(6) Perform one sweep ($\tilde{J}-1$ substeps) of the numerical method for the $\mathcal{G}$ propagator applied to the correction (2) using the values $S^{\tilde{j}+1}_{\tilde{j}} f(\tilde{\boldsymbol{t}}_n, \boldsymbol{U}^0_n)$ just computed. This will yield updated values $\tilde{U}^1_{n,\tilde{j}}$ for $\tilde{j} = 1 \ldots \tilde{J}$.

(7) Pass the value $\tilde{U}^1_{n,\tilde{J}}$ to processor $\boldsymbol{P}_{n+1}$ (if $n < N-1$) which will be received as $\tilde{U}^2_{n+1,1}$

Each subsequent iteration. After the first iteration, the parareal/SDC algorithm proceeds in nearly the same way as in the first iteration except in the first two steps above. In the first iteration, the coarse values from the initial predictor must be interpolated to the fine nodes in step 1 above. In subsequent iterations, there are two alternatives regarding how the results from applying $\mathcal{G}$ in the previous parareal iteration are used. First, one could use no information from the values $U^{k-1}_{n,\tilde{j}}$ computed in the previous $\mathcal{G}$ step. This alternative is in the spirit of parareal where the $\mathcal{G}$ correction sweep only provides a new initial value for the $\mathcal{F}$ step on

the next processor. In this case step 1 above would be skipped and instead of using a quadrature rule in step 2, set $S_j^{j+1} f(\boldsymbol{t}, U_{n,j}^k) = S_j^{j+1} f(\boldsymbol{t}, \tilde{U}_{n,j}^{k-1})$.

Alternatively, the coarse values $U_{n,\tilde{j}}^{k-1}$ could also be used to improve the solution at the fine nodes as well. In the numerical tests included here, this is done simply by forming the difference $U_{n,\tilde{j}}^{k-1} - \tilde{U}_{n,\tilde{j}}^{k-1}$ at the coarse nodes, and then interpolating this difference to the points $\boldsymbol{t}_n$ to form $U_{n,j}^k$ in step 1. These values are then used to compute $S_j^{j+1} f(\boldsymbol{t}, \boldsymbol{U}_n^k)$ in step 2.

Note that in each iteration, two SDC sweeps are performed, one on the coarse nodes $\tilde{\boldsymbol{t}}_n$ corresponding to $\mathcal{G}$ and one on the fine nodes $\boldsymbol{t}_n$ corresponding to $\mathcal{F}$. However, data is only passed between processors once per iteration after $\mathcal{G}$ is completed. The use of the coarse nodes for $\mathcal{G}$ reduces the serial cost of computing the first predictor in contrast to the pipelined SDC methods from [33; 16] described in Section 4.1.

5. Parallel speedup and efficiency

In this section, an analysis of the theoretical parallel speedup and efficiency of the parareal and parareal/SDC methods is presented. First, the standard analysis of the parareal method is reviewed which shows that the parallel efficiency cannot exceed $1/K$, for K iterations of the method. Next, it is demonstrated that the parallel efficiency of the parareal/SDC method can be much closer to 1.

The application of the $\mathcal{G}$ corrector sweep described by (18) in the parareal method is generally regarded as a serial operation since $\boldsymbol{P}_n$ cannot apply the correction step described by (18) until the value U_n^{k+1} is computed on $\boldsymbol{P}_{n-1}$ and sent to $\boldsymbol{P}_n$. The $\mathcal{G}$ propagator is often assumed to be used for the initial prediction phase as well which is clearly a serial operation, hence parareal is often described as iteratively applying the $\mathcal{G}$ propagator in serial and the $\mathcal{F}$ propagator in parallel. The theoretical parallel speedup and efficiency of the parareal method from this point of view has been studied previously by [6; 7; 22] and this analysis is first reviewed below. Then, an analysis for a pipelined version of parareal will be discussed followed by a similar analysis for a parareal/SDC method.

5.1. ***Serial-parallel parareal.*** To begin, assume that each processor is identical and that the communication time between processors is negligible. Denote the time for a processor to compute one step of the numerical method used in the $\mathcal{G}$ propagator by τ_G. Likewise let τ_F denote the time for one processor to compute one step of the numerical method used as the $\mathcal{F}$ propagator. Since multiple steps of a numerical method can be used for either $\mathcal{G}$ or $\mathcal{F}$, denote the number of steps as N_G and N_F respectively, and the size of the steps by Δt and δt. Hence the total cost of $\mathcal{F}$ is $N_F \tau_F$, which is denoted Υ_F. We assume that the cost of applying the $\mathcal{G}$ correction in (18) is equal to $N_G \tau_G = \Upsilon_G$, that is, the cost of forming the difference

N	Number of processors
K	Number of parareal iterations
τ_G	Cost of the numerical method in $\mathcal{G}$
τ_F	Cost of the numerical method in $\mathcal{F}$
T	Length of time integration
ΔT	Time increment per processor (T/N)
Δt	Time increment for $\mathcal{G}$ propagator
δt	Time increment for $\mathcal{F}$ propagator
N_G	Number of $\mathcal{G}$ steps per processor $(\Delta T/\Delta t)$
N_F	Number of $\mathcal{F}$ steps per processor $(\Delta T/\delta t)$
Υ_G	Total cost $\mathcal{G}$ propagator $(N_G\tau_G)$
Υ_F	Total cost $\mathcal{F}$ propagator $(N_F\tau_F)$

Table 1. Notation for the parareal algorithm.

in (18) is negligible. Let N denote the number of processors used, and $\Delta T = T/N$ the time increment per processor. Table 1 summarizes these definitions.

Assume that the prediction step is done with the $\mathcal{G}$ propagator, then for N processors the cost of computing the predictor is $NN_G\tau_G = N\Upsilon_G$. Likewise the cost of each iteration of the parareal method is $NN_G\tau_G + N_F\tau_F = N\Upsilon_G + \Upsilon_F$ (assuming that the method ends with a correction step). A graphical description of the cost is shown in Figure 2. The dots in the figure indicate communication between two processors.

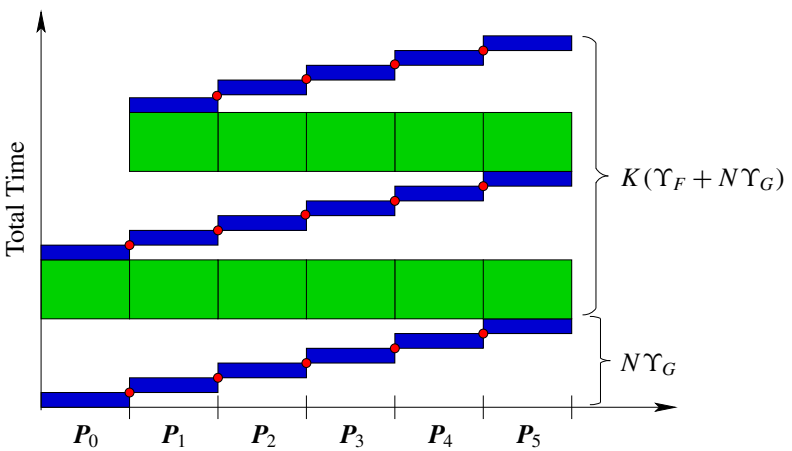


Figure 2. Cost of the serial-parallel version of the parareal method for $K = 2$ iterations and $N = 6$ processors. The dots indicate communication between two processors.

The total cost for the predictor and K parareal iterations as implemented in Figure 2 is hence

$$N N_G \tau_G + K(N N_G \tau_G + N_F \tau_F) = N\Upsilon_G + K(N\Upsilon_G + \Upsilon_F). \tag{26}$$

The cost of applying $\mathcal{F}$ serially is $N N_F \tau_F = N\Upsilon_F$; hence the speedup for parareal is

$$S = \frac{N\Upsilon_F}{N\Upsilon_G + K(N\Upsilon_G + \Upsilon_F)} = \frac{1}{\frac{\Upsilon_G}{\Upsilon_F} + K\left(\frac{\Upsilon_G}{\Upsilon_F} + \frac{1}{N}\right)}. \tag{27}$$

Denoting the ratio Υ_G/Υ_F by α we have

$$S = \frac{1}{\alpha + K(\alpha + 1/N)}. \tag{28}$$

In [6], some further assumptions are made to simplify the computation of a maximum possible speedup. Since we would like to compare the parareal solution with a serial fine solution with some fixed time step δt, N and N_F are related by

$$N_F = \Delta T/\delta t = \frac{T}{N\delta t}. \tag{29}$$

Assume further that the same method is used for both the fine and coarse propagator, and only one step is used for the coarse propagator; that is, $N_G = 1$ and $\tau_G = \tau_F$. Then

$$\alpha = \frac{1}{N_F} = \frac{N\delta t}{T}. \tag{30}$$

Under these additional assumptions, the speedup becomes

$$S = \frac{1}{(K+1)\frac{N\delta t}{T} + \frac{K}{N}}. \tag{31}$$

The maximum value of S in terms of N can hence be easily seen to occur when $N = N^*$, where

$$N^* = \sqrt{\frac{KT}{(K+1)\delta t}}. \tag{32}$$

This value of N^* gives a maximum speedup of S^* where

$$S^* = \frac{1}{2}\sqrt{\frac{T}{\delta t K(K+1)}}. \tag{33}$$

If one considers the parallel efficiency $E = S/N$, using (28) gives

$$E = \frac{1}{N\alpha(K+1) + K}. \tag{34}$$

Clearly the parallel efficiency is bounded by $E < 1/K$, which must be true since each parareal iteration is more expensive than computing the serial solution over the interval represented on each processor (i.e., ΔT). Since K is usually at least 2, it is often stated that the parallel efficiency of parareal is bounded by $\frac{1}{2}$. Using the value of N^* above, we see that the parallel efficiency at maximum speedup is $1/(2K)$.

5.2. ***Pipelined parareal.*** The version of the parareal algorithm just discussed is not the most efficient implementation possible on most current parallel architectures in which processors are able to perform different tasks at different times. Although the predictor step in parareal is certainly a serial operation, after $\boldsymbol{P}_n$ has computed the value U_n^0, it is free to immediately apply the $\mathcal{F}$ propagator rather than wait for the predictor to be computed on all processors. We refer to such an implementation where a processor computes a step in the algorithm as soon as possible as "pipelined". Likewise, if each processor begins computing the $\mathcal{G}$ propagator as soon as the initial condition is available, the parareal iterations can be pipelined so that the cost is of each parareal iteration is $N_F\tau_F + N_G\tau_G = \Upsilon_F + \Upsilon_G$ instead of $\Upsilon_F + N\Upsilon_G$. Figure 3 demonstrates graphically the cost of a pipelined parareal implementation. Note that this form of pipelining must be modified if the computation of the predictor is less expensive than applying the $\mathcal{G}$ corrector sweep.

The parallel speedup for pipelined parareal is

$$S = \frac{NN_F\tau_F}{NN_G\tau_G + K(N_G\tau_G + N_F\tau_F)} = \frac{N\Upsilon_F}{N\Upsilon_G + K(\Upsilon_G + \Upsilon_F)}. \tag{35}$$

Introducing $\alpha = \Upsilon_G/\Upsilon_F$ as above gives a parallel speedup of

$$S = \frac{1}{\alpha + K\left(\frac{\alpha}{N} + \frac{1}{N}\right)}. \tag{36}$$

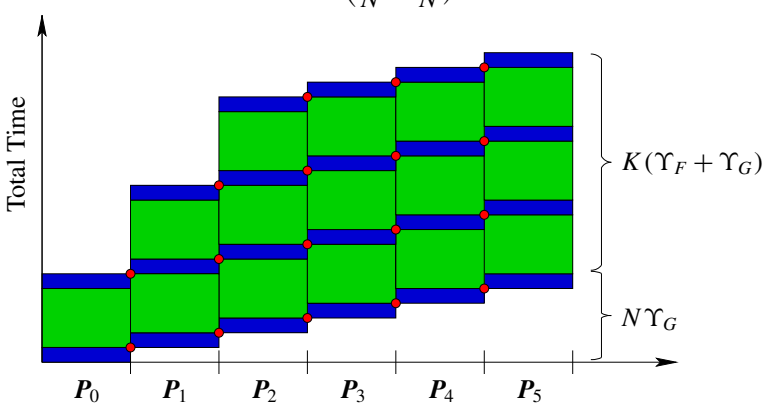


Figure 3. Cost of the pipelined version of the parareal method for $K = 3$ iterations and $N = 6$ processors. The dots indicate communication between two processors.

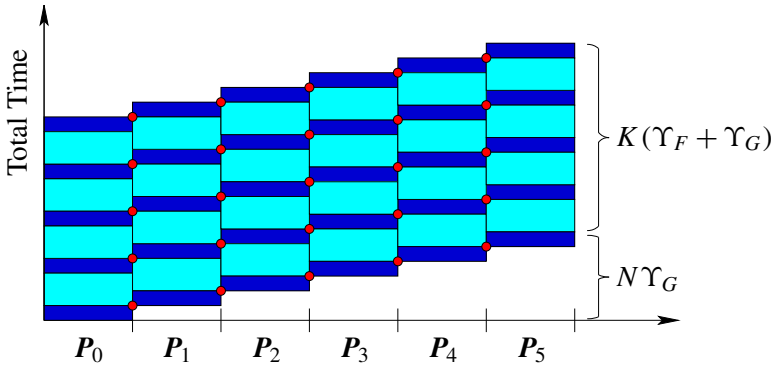


Figure 4. Cost of the hybrid parareal/SDC method for $K = 4$. The dots indicate communication between two processors.

Again making the simplifying assumptions from [6] of $N_G = 1$ and $\tau_G = \tau_F$, and hence $\alpha = N\delta t/T$, gives

$$S = \frac{1}{\frac{N\delta t}{T} + K\left(\frac{\delta t}{T} + \frac{1}{N}\right)}. \tag{37}$$

which is maximized by

$$N^* = \sqrt{KT/\delta t}. \tag{38}$$

This value is a factor of $\sqrt{K+1}$ larger than the value in (32), meaning that more processors can be used before the speedup saturates. This value of N^* gives a maximum speedup of

$$S^* = \frac{1}{2}\sqrt{\frac{T}{\delta t K}}\left(\frac{1}{1+\sqrt{K\delta t/T}}\right). \tag{39}$$

Comparing (33) and (39) shows that pipelining increases the maximum speedup by approximately a factor of $\sqrt{K+1}$ when $\delta t/T$ is small.

The parallel efficiency for pipelined parareal is

$$E = \frac{1}{N\alpha + K(\alpha+1)} = \frac{1}{\alpha(N+K)+K}. \tag{40}$$

Hence despite the lower cost of each parareal iteration, $E < 1/K$ since the cost of each iteration is still greater than the serial cost of computing the fine solution over the interval of length ΔT. In the first numerical tests presented in Section 6, for the tolerances used, K is larger than 10, which unfortunately gives a low parallel efficiency. Note that the bound $E < 1/K$ holds regardless of the assumptions made about the relative cost of τ_G because in the parareal iteration, $\mathcal{F}$ is computed during every iteration.

5.3. ***Parareal with SDC.*** In order to examine the speedup and efficiency of the parareal/SDC hybrid method, the analysis and notation above must be modified slightly. First we assume again that parareal/SDC has been implemented in a pipelined fashion as in Figure 3. We assume that the cost of computing one substep of SDC with $\mathcal{F}$ or $\mathcal{G}$ is the same as one step of $\mathcal{F}$ or $\mathcal{G}$ for an ODE (i.e., the cost of additional residual term in the correction equation is negligible). In the implementation used for the numerical results presented here, the parareal/SDC method will converge to an SDC method that uses one time step per processor. Of relevance to the cost is the number of substeps used in the fine SDC method, and this is denoted as in Section 4.3 as J (which is the number of SDC nodes used minus one). Let τ_F denote the cost of the method used for each substep in the correction step of SDC on the fine nodes. Likewise, let τ_G and $\tilde{J}$ be the corresponding constants for the $\mathcal{G}$ propagator. The total cost of the computing $\mathcal{F}$ is $J\tau_F$, which is again denoted Υ_F. The corresponding cost for $\mathcal{G}$ is $\Upsilon_F = \tilde{J}\tau_G$. Table 2 summarizes this notation.

N	Number of processors
K	Number of parareal iterations
M	Number of SDC iterations for a serial method
τ_G	Cost of the numerical method in $\mathcal{G}$
τ_F	Cost of the numerical method in $\mathcal{F}$
$\tilde{J}$	Number of substeps for coarse SDC sweep in $\mathcal{G}$
J	Number of substeps for fine SDC sweep in $\mathcal{F}$
Υ_G	Total cost $\mathcal{G}$ propagator ($\tilde{J}\tau_G$)
Υ_F	Total cost $\mathcal{F}$ propagator ($J\tau_F$)

Table 2. Notation for the parareal/SDC algorithm.

Assuming the method is pipelined, each parareal/SDC iteration will have a cost $J\tau_F + \tilde{J}\tau_G = \Upsilon_F + \Upsilon_G$. If the method used for the predictor also has cost per time step Υ_G, then the total cost for K iterations of parareal/SDC is $N\Upsilon_G + K(\Upsilon_F + \Upsilon_G)$.

Let M denote the number of SDC iterations needed to compute the solution to the desired accuracy on a single processor using the fine SDC nodes. Then the cost of the serial SDC method will be approximately $NM\Upsilon_F$. The parallel speedup S is hence

$$S = \frac{NM\Upsilon_F}{N\Upsilon_G + K(\Upsilon_G + \Upsilon_F)}. \tag{41}$$

Note that this is exactly M times greater than the value for the speedup for the pipelined parareal method in (35). Proceeding as in the pipelined parareal analysis above would lead to the same value for N^* and a value of S^* which is again M times bigger than before.

Setting $\alpha = \Upsilon_G/\Upsilon_F$ in (41) gives

$$S = \frac{M}{\alpha + (K/N)(\alpha+1)}, \tag{42}$$

and efficiency

$$E = \frac{M}{N\alpha + K(\alpha+1)} = \frac{M}{(N+K)\alpha + K}. \tag{43}$$

Again, the efficiency is exactly M times greater than the value for the pipelined parareal method in (40). This is due to the fact that the cost of doing M iterations of the SDC method has been amortized over the cost of the parareal iterations. In contrast to the standard parareal method, the efficiency is not automatically bounded above by $E < 1/K$. However, since M is fixed, for the efficiency to approach M/K, the product $(N+K)\alpha$ should be as small as possible. Unfortunately, for ODEs decreasing α is equivalent to decreasing the accuracy of $\mathcal{G}$, and this leads in general to an increase in K as is demonstrated in Section 6.

However for PDEs, if $\mathcal{G}$ could be computed on a coarser spatial mesh, the cost τ_G could be reduced significantly. This idea has in fact been suggested for the parareal method as well [8; 7; 23; 26]. In the parareal method however, reducing the cost of τ_G cannot increase the parallel efficiency beyond the bound $E < 1/K$ since $\mathcal{F}$ is still computed in every iteration.

5.4. ***Further discussion.*** Several comments can be made about the parallel cost and efficiency of parareal/SDC. First, it should be noted that achieving a large parallel speedup is only significant if the serial method to which one is comparing is efficient. Serial SDC methods have already been shown to be competitive in terms of cost per accuracy with higher-order Runge–Kutta and linear multistep methods [51; 43]. A comparison for explicit SDC methods and Runge–Kutta is also included in Section 6.

Secondly, in order for parareal/SDC to converge to the accuracy of the serial SDC method, K must be greater than $M/2$ since only $2K$ correction sweeps are done for K parareal iterations. In practice, since it is desirable to make α small (i.e., $\mathcal{G}$ much less expensive than $\mathcal{F}$), then K must be closer to M to achieve the accuracy of M serial iterations of SDC. If one examines the efficiency when $M = K$, then

$$E = \frac{1}{\left(\frac{N}{M}+1\right)\alpha + 1}. \tag{44}$$

Although this still seems to scale poorly as N grows larger, it should be noted that for the solution of PDEs (which motivates this work), temporal parallelization would be combined with spatial parallelization and hence N would be chosen so that the gain in efficiency from temporal parallelization exceeds the (presumably

saturated) spatial parallelization. Also, the processors in time parallelization can be used in a cyclical nature. Rather than dividing the entire time domain by the number of processors available, the processors are employed on the first N time-steps, and as the first processor finishes, it is employed to begin iterating on the $N+1$ time step as in the parallel deferred correction methods [33; 16].

As previously mentioned, a very promising approach is to use a coarser spatial grid for $\mathcal{G}$ as is discussed in [8; 7; 23; 26]. For three-dimensional calculations, even a factor of two reduction in grid spacing results in a factor of eight reduction in the number of spatial unknowns. In this scenario, the quantity α could be quite small, and the efficiency could theoretically exceed 1/2 in contrast to parareal where the efficiency is bounded by $1/K$ regardless of the cost of $\mathcal{G}$. Investigation of parareal/SDC using spatial coarsening for PDEs will be reported in a subsequent paper.

Finally, the above discussion ignores the reality that current massively parallel computers typically exhibit highly inhomogeneous communication costs between processors, and in the case of large scale grid computing, processors are not necessarily of comparable computational power. Even in a serial computation, when iterative methods are employed to solve the implicit equations associated with methods for stiff equations, the amount of work per time step can vary greatly. Furthermore, the issue of time-step adaptivity, which is critical for the efficient solution of many problems, causes further difficulties in analyzing the parallel efficiency of the time-parallel methods.

6. Numerical examples

In this section, preliminary numerical results are presented to compare the efficiency of the parareal/SDC method to standard implementations of parareal using Runge–Kutta methods for both the $\mathcal{F}$ and $\mathcal{G}$ propagators. The problems considered are taken from recent papers on parareal and focus on the effect of using SDC sweeps for the $\mathcal{F}$ and $\mathcal{G}$ propagator on the convergence of the parareal iterations.

6.1. ***The Lorenz oscillator.*** Here the effectiveness of the hybrid parareal/SDC method is explored using the Lorenz equation test problem from [27]. Specifically, we consider

$$x' = \sigma(y - x), \quad y' = x(\rho - z) - y, \quad z' = xy - \beta z. \tag{45}$$

in $t \in [0, 10]$ with the usual choice of parameters $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$ and initial conditions $(x, y, z)(0) = (5, -5, 20)$. The resulting solution with these parameters is very sensitive to numerical error.

The particular implementation of parareal considered in [27] uses use $N = 180$ processors and the standard explicit fourth-order RK method for both $\mathcal{G}$ and $\mathcal{F}$, where

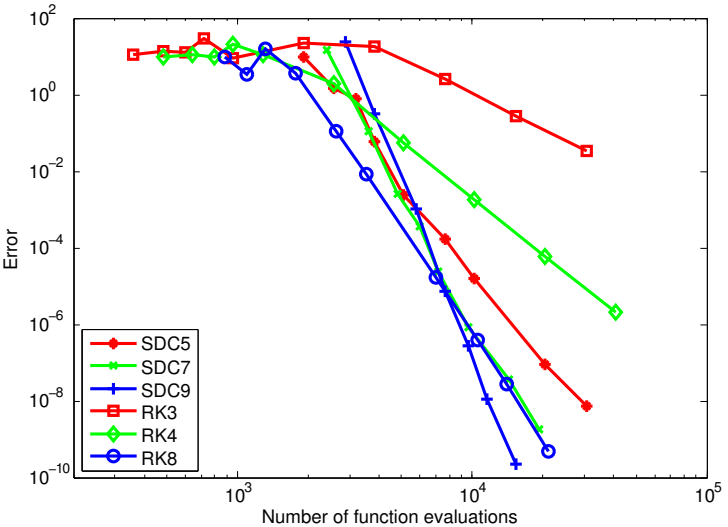


Figure 5. Error versus number of function evaluations: serial Runge–Kutta and SDC methods for the Lorenz equation.

$\mathcal{G}$ is implemented as a single step and $\mathcal{F}$ 80 steps. Here, additional implementations are considered using the standard explicit third-order RK method for $\mathcal{G}$ and explicit third-, fourth-, and eighth-order [20] RK methods (denoted here RK3, RK4, and RK8 respectively) as the $\mathcal{F}$ propagators. The convergence and efficiency of different combinations of RK methods will be compared to parareal/SDC methods using a single explicit correction sweep of SDC as the $\mathcal{F}$ propagator with five, seven, and nine Gauss–Lobatto nodes for each processor.

Before studying the parallel methods, the error for serial methods is first examined to understand the accuracy of different choices. Figure 5 compares the L_2 error of the solution at the final time $T = 10$ versus the number of function evaluations using serial Runge–Kutta and SDC methods. In the SDC methods, the RK4 method is used to compute the provisional solution at each node, and a second-order RK method is applied to the correction (2) during the deferred correction sweeps. Two, three, and four SDC sweeps respectively, are applied for the SDC methods using five, seven, and nine Lobatto nodes.

Note that the formal order of accuracy of the three SDC methods if the SDC iterations are fully converged to the collocation method would be 8, 12 and 16, but in this example, the number of SDC iterations used limits the formal order of accuracy to 8, 10, and 12. Here the computational cost of the SDC method ignores the cost of computing the numerical quadrature for the correction equations (which is done using a simple matrix-vector multiply). Figure 5 demonstrates that the numerical approximations for the SDC methods are more efficient than the third-

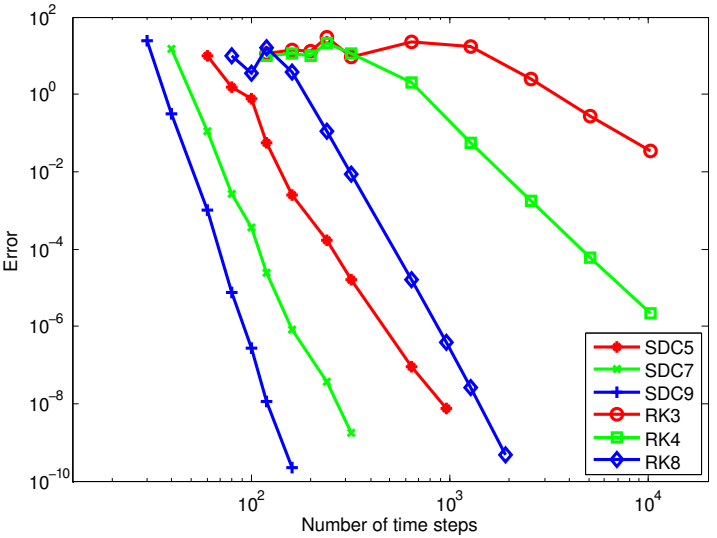


Figure 6. Error versus time step: serial Runge–Kutta and SDC methods for the Lorenz equation.

and fourth-order RK methods for a very modest error tolerance. For more stringent error tolerances, the efficiency of the SDC methods using seven and nine Lobatto nodes are similar to the eighth-order RK method.

In Figure 6 the error versus time step of the methods is compared. In this figure, the cost per time step of the methods is not relevant, and it is evident that the SDC methods are able to obtain the same level of accuracy as the RK methods with a much larger time step. This fact will allow a substantial increase in the efficiency of the parareal/SDC methods considered below since the increased cost per time step is amortized over parareal iterations.

Next the behavior of various implementations of the parareal and parareal/SDC methods in terms of the convergence of the parareal iterations is examined. First, the traditional parareal method using RK is examined. In Figure 7, the error at the final time is plotted versus parareal iteration for six different combinations of the $\mathcal{G}$ and $\mathcal{F}$ propagators. Two choices for $\mathcal{G}$ are used (one step of RK3 or RK4), while three choices for $\mathcal{F}$ are used (100 steps of RK3, 80 steps of RK4, or 8 steps of RK8). In each case the $\mathcal{G}$ propagator is used for the initial serial prediction step. Note that the convergence behavior for methods using the same $\mathcal{G}$ propagator are very similar. On the other hand, the overall error after convergence of parareal depends on the accuracy of the $\mathcal{F}$ propagator. Note that the absolute accuracy of the numerical method is used in the plots, rather than the difference between the parareal iterates and the result obtained from using $\mathcal{F}$ in serial. Hence the leveling off of the error in the convergence plots gives an indication of the accuracy of the serial $\mathcal{F}$ method.

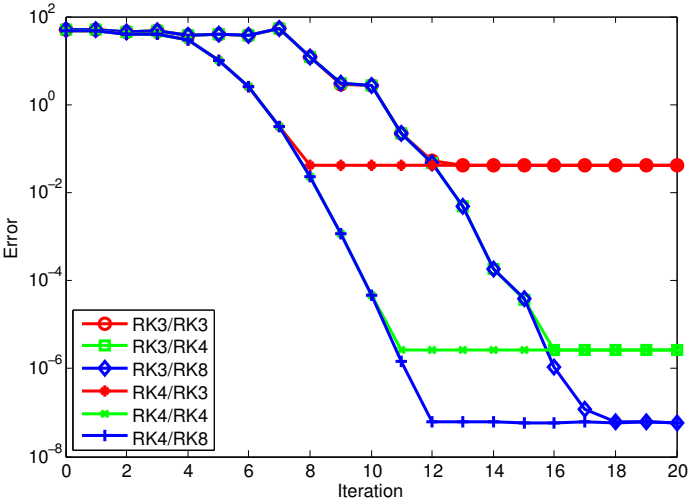


Figure 7. Convergence of parareal iterations for the Lorenz equation using different Runge–Kutta combinations for $\mathcal{G}$ and $\mathcal{F}$. In the legend, the combinations are listed as $\mathcal{G}/\mathcal{F}$.

The convergence results for parareal/SDC are presented in Figure 8. Again, six variations are presented. As before $\mathcal{G}$ is either RK3 or RK4, although these methods

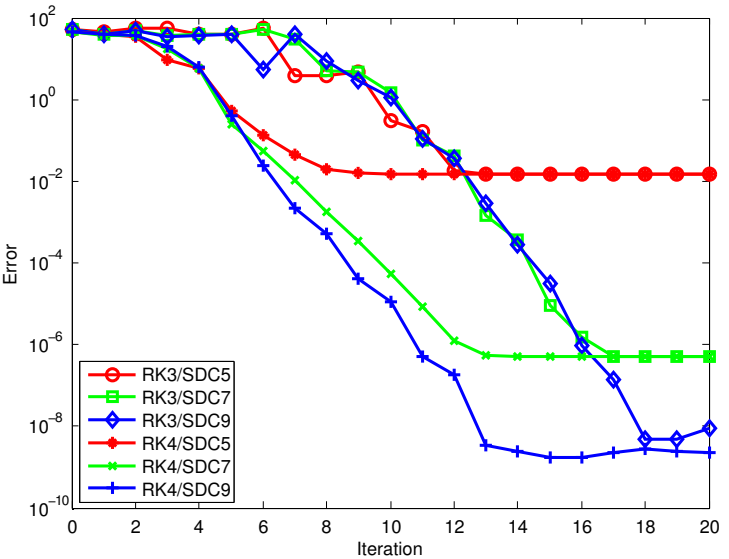


Figure 8. Convergence of parareal/SDC iterations for the Lorenz equation using different combinations of Runge–Kutta for $\mathcal{G}$ and a second-order SDC sweep for $\mathcal{F}$ with different number of nodes. In the legend, the combinations are listed as $\mathcal{G}/\mathcal{F}$.

are now applied to the correction (2). The $\mathcal{F}$ propagator is done using a single sweep of (2) using the first order method in (7) for each node. This means that $\mathcal{F}$ requires only 4, 6, and 8 function evaluations for the methods using 5, 7, and 9 Lobatto nodes respectively. As in Figure 7, the data in Figure 8 show that the number of parareal iterations required to converge depends on $\mathcal{G}$ while the overall accuracy depends on $\mathcal{F}$ (here the number of Lobatto nodes).

Next the convergence behavior of traditional parareal and parareal/SDC methods is compared in Figure 9. Of interest is whether the use of a low-order corrector for $\mathcal{F}$ in the parareal/SDC method adversely affects the number of iterations required for convergence. The left panel in Figure 9 shows the convergence for methods using RK3 in $\mathcal{G}$ and different choices of RK and SDC for $\mathcal{F}$. The right panel shows the corresponding data using RK4 for $\mathcal{G}$. The data demonstrate that replacing the full accuracy RK solve in $\mathcal{F}$ with a single SDC sweep does not significantly affect the convergence of the parareal iterates for this example. As observed above, the convergence behavior depends mainly on the accuracy of $\mathcal{G}$ for both methods and the number of iterations needed to converge to a given tolerance is very similar parareal and parareal/SDC methods.

Of greater interest here however is the total parallel computational cost of the parareal and parareal/SDC methods. Hence Figure 10 shows the convergence of parareal iterations versus total parallel cost for traditional parareal methods using

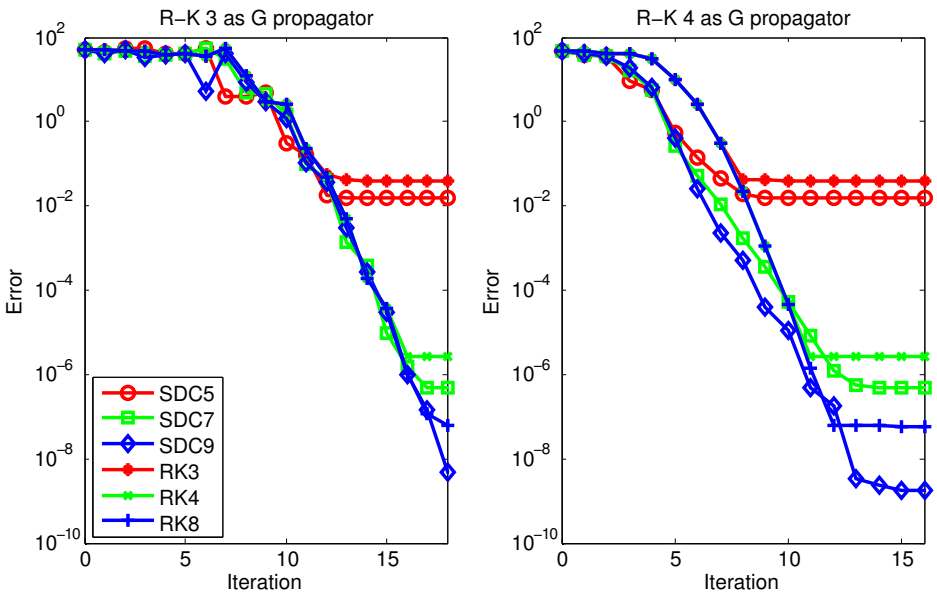


Figure 9. Convergence of parareal and parareal/SDC methods for the Lorenz equation. The methods in the left panel use explicit RK3 for $\mathcal{G}$, while those in the right panel use explicit RK4.

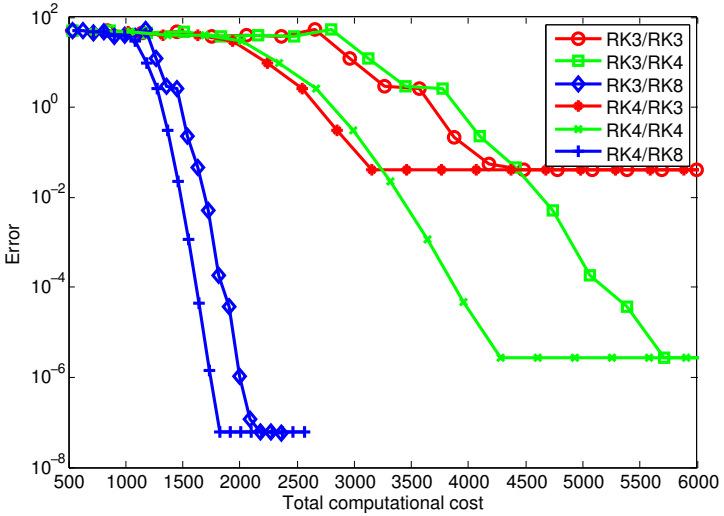


Figure 10. Error versus parallel computational cost of the parareal method using different combinations of Runge–Kutta for $\mathcal{G}$ and $\mathcal{F}$ (listed as $\mathcal{G}/\mathcal{F}$ in the legend) for the Lorenz equation.

RK methods for $\mathcal{G}$ and $\mathcal{F}$. The cost is computed using the assumption of a pipelined implementation as explained in Section 5.2 and is based on the number of explicit function evaluations: communication cost is ignored. Figure 10 shows that the total parallel cost is dominated by the cost of $\mathcal{F}$. Although using RK4 for $\mathcal{G}$ rather than RK3 reduces the number of iterations required somewhat, this is offset by the increased cost of RK4. Since the use of higher-order methods for $\mathcal{F}$ is more efficient in terms of accuracy per functions evaluations (see Figure 5), the total parallel cost for methods using RK8 for $\mathcal{F}$ is substantially less than those using RK3 and RK4.

Next consider the parallel cost for parareal/SDC shown in Figure 11. Again a pipelined implementation is assumed, and since only function evaluations are counted, the cost of the computing the numerical quadrature and interpolating the correction computed by $\mathcal{G}$ (both of which are simple matrix multiplications) is not included. The most notable difference in the data is that the total computational cost of the parareal/SDC method is dominated by the cost of the initial serial $\mathcal{G}$ sweep. As noted before, this fact suggests that using spatial coarsening for $\mathcal{G}$ for PDEs could increase the efficiency of parareal/SDC methods significantly.

Finally, the total parallel cost for both traditional parareal using RK and parareal/SDC is compared in Figure 12. Since the methods use the same predictor, the cost is identical at the end of the serial predictor step, but it is evident that the much reduced cost of using SDC for $\mathcal{F}$ greatly reduces the total computational cost. Specifically, the cost of $\mathcal{F}$ for the parareal/SDC method is either 4, 6, or 8 function

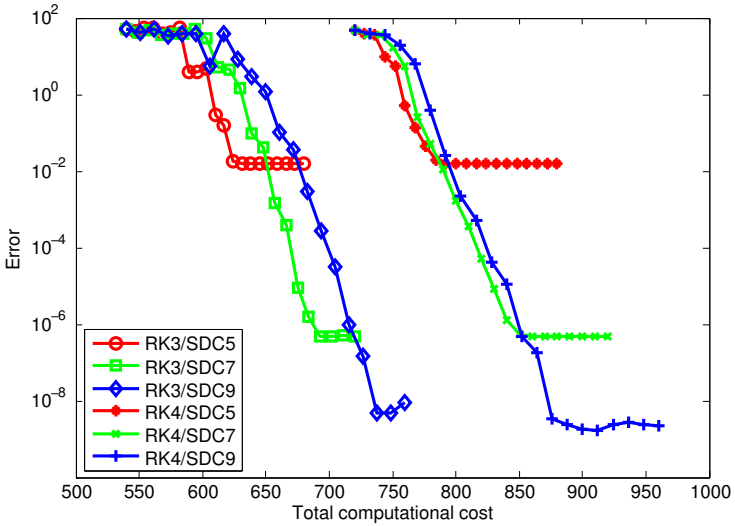


Figure 11. Error versus parallel computational cost of the parareal/SDC method using different combinations of $\mathcal{G}$ and $\mathcal{F}$ (listed as $\mathcal{G}/\mathcal{F}$ in the legend) for the Lorenz equation.

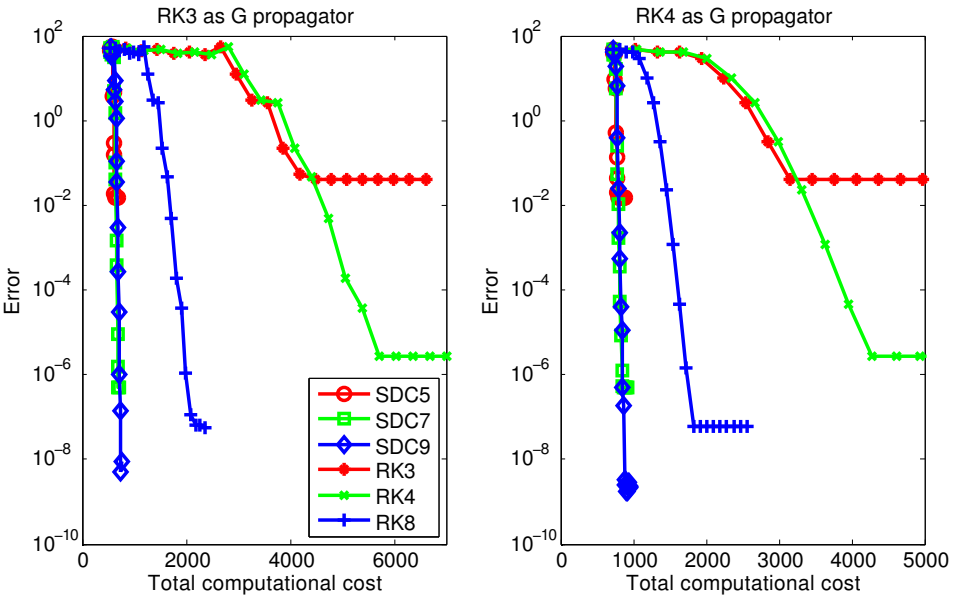


Figure 12. Comparison of total parallel cost for parareal and parareal/SDC methods for the Lorenz equation.

evaluations while for the RK based parareal method, it is 300, 320, or 88 function evaluations.

6.2. ***A PDE example.*** Next, the performance of the parareal/SDC method is demonstrated for a discretized partial differential equation, namely the viscous Burgers equation

$$u_t + uu_x = \nu u_{xx}, \quad u(x,0) = g(x).$$

The spatial domain for all experiment is the periodic interval [0, 1], with initial data given by $g(x) = \sin(2\pi x)$, and $\nu = 1/50$. This example is also considered in [27], although there a second-order finite difference spatial discretization with first-order backward Euler in time is used. Here, a semi-implicit or IMEX temporal discretization is used, and the finite difference discretization is replaced by a pseudo-spectral approach (with no de-aliasing). The semi-implicit time stepping requires only the diffusive piece to be treated implicitly; hence the implicit problem is linear (and here solved in spectral space).

For the parareal/SDC method, 7 Gauss–Lobatto nodes are used in the $\mathcal{F}$ propagator with a first-order semi-implicit corrector. For $\mathcal{G}$, either 1 or 2 steps of a first-order semi-implicit corrector is used. In each of the tests below, the error reported is the maximum of the error at the final time as compared to a highly resolved reference solution computed with Runge–Kutta and not the solution generated by using $\mathcal{F}$ in serial. Note that the reference solution is computing using the same number of spatial unknowns as the problem being run, so the solution to the PDE is not necessarily fully resolved in space.

In the first set of tests, 64 spatial grid points are used, and the convergence behavior for runs of different length of integration is compared. Specifically, the simulations are run to final time $T = 0.1$, $T = 0.5$, and $T = 1.0$. In all cases, the time interval for each processor is 1/100; hence 10, 50, and 100 processors are used. In Figure 13, the results are displayed in the left panel using one step of forward/backward Euler for $\mathcal{G}$ and in the right panel for two steps of forward/backward Euler. In these figures, one does not observe the exponential convergence of the error in the parareal iterations as is evident in the examples for the Lorenz equation. This is due to the fact that the convergence of the SDC iteration is slower in this example than the convergence of the parareal iterations for short time (Note in particular the nearly constant convergence rate for the $T = 0.1$ example). Note also in the right panel that the increase in the accuracy of $\mathcal{G}$ from using an additional step of forward/backward Euler increases the rate of convergence of the parareal/SDC iterations.

In the next set of tests, the integration time is fixed at $T = 1$ with 100 processors, but the number points used in the spatial discretization is varied. Figure 14 shows the convergence behavior for 64, 128, and 256 spatial points. The data show that the convergence behavior is completely unaffected by the number of spatial variables used.

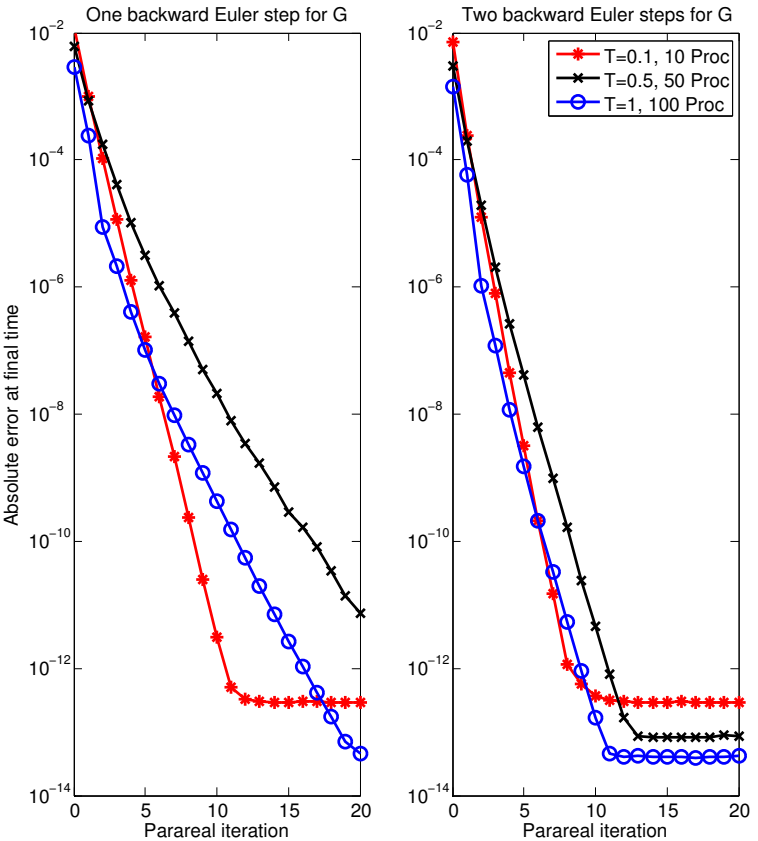


Figure 13. Convergence of the parareal/SDC method for Burgers equation using one step (left) and two steps (right) of forward/backward Euler for $\mathcal{G}$.

In the final set of tests, the integration time is fixed at $T = 1$ with a grid size of 64 spatial points, but the number of processors is varied. Figure 15 shows the convergence behavior for $N = 20$, 40 and 100 for 1 step of forward/backward Euler for $\mathcal{G}$ in the left panel, and 2 steps in the right panel. The fact that the parareal iterates are converging faster for a larger number of processors is again due to the increased accuracy of $\mathcal{G}$. As the number of processors is increased, the coarse time step gets smaller and hence $\mathcal{G}$ becomes more accurate. Hence the number of iterations needed to converge decreases.

A few words should be said regarding any comparison in cost to the method in [27]. First, the $\mathcal{F}$ propagator in [27] is based on 10 steps of backward Euler for the first set of test cases. Here, the $\mathcal{F}$ method is 6 substeps of a semi-implicit method applied in the SDC sweep. Depending on the efficiency of solving the nonlinear equation for the fully implicit method, the semi-implicit approach could

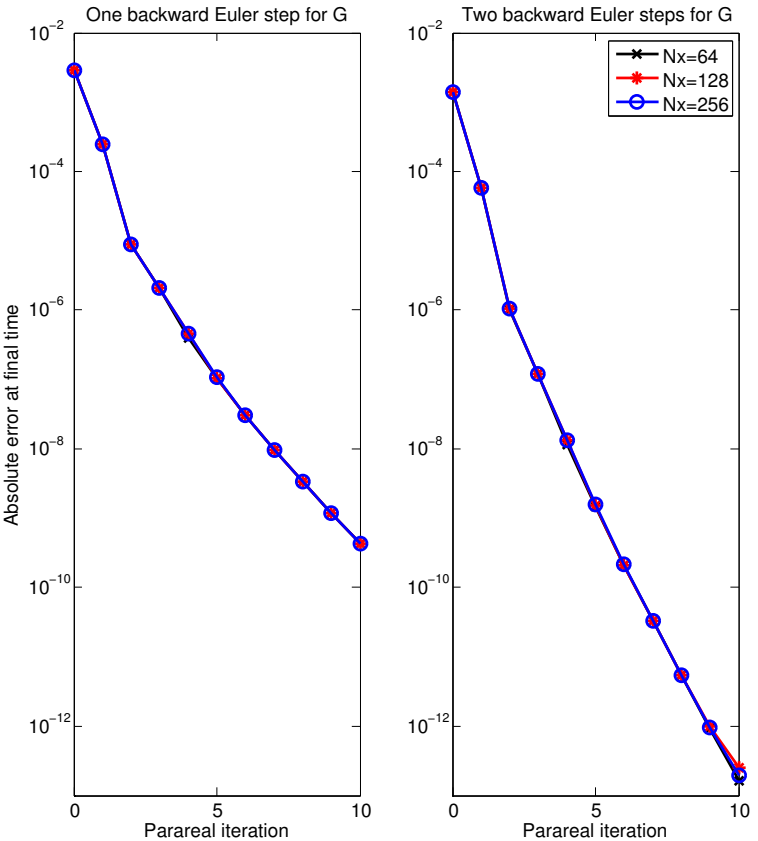


Figure 14. Parareal/SDC convergence for different spatial resolutions for Burgers equation. For all runs, the final time is $T = 1$, and 100 processors are used.

be substantially more efficient. The biggest difference between the two approaches is in the overall accuracy, which for the high-order SDC-based method used here is over ten orders of magnitude smaller than that achieved with the first-order temporal method used in [27].

7. Conclusions

A new strategy for combining deferred corrections and the parareal method for the temporal parallelization of ordinary differential equations first presented in [53] is further developed and evaluated. One can regard the parareal/SDC strategy as either a way to parallelize SDC methods in time or as a way to increase the efficiency of parareal methods by reducing the computational cost of the $\mathcal{F}$ propagator.

The motivation of this research is to develop parallel-in-time strategies to be combined with spatial parallelization for massively parallel computations of PDEs,

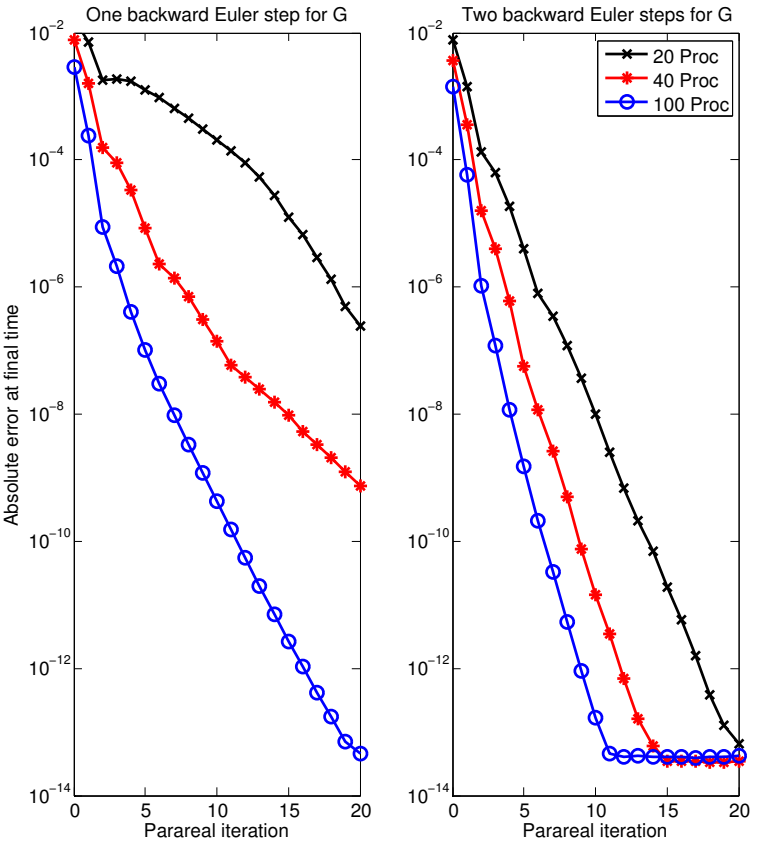


Figure 15. Parareal/SDC convergence for different numbers of processors. Each run is to the final time $T = 1$ with 64 spatial grid points.

particularly in computational fluid dynamics. One of the significant features of the parareal/SDC approach is that the greatly reduced cost of computing $\mathcal{F}$ means that reducing the cost of $\mathcal{G}$ by using a coarser spatial grid could increase the overall parallel efficiency significantly. Results in this direction will be reported in the future.

Other possibilities for further increasing the efficiency of the parareal/SDC approach include the use of Krylov methods to accelerate the convergence of the parareal/SDC iterations. Krylov acceleration methods have already been studied for serial SDC methods for both ODEs and DAEs [35; 36; 12], as well as for the traditional parareal method [28], although the effectiveness of these methods for large scale PDEs has not yet been demonstrated. Another possibility concerns the use of iterative solvers within implicit or semi-implicit temporal methods for PDEs. It seems reasonable that the error tolerance within implicit solvers could

be dynamically decreased as the parareal iterations progress, but this may affect the convergence of the parareal iterations. In the parareal/SDC approach, a very good initial guess for these implicit solves is available from the previous parareal iteration. In conclusion, despite the promising initial results reported here, avenues for further improvement need to be pursued.

Appendix: Pseudo-code of the parareal/SDC method

The following is the pseudocode for a semi-implicit parareal/SDC implementation using the first-order time-stepping method in (12) and (15).

Serial initialization:

FOR $n = 0 \ldots N-1$
  COMMENT: Get initial data
  IF $n = 0$
    $\tilde{U}^0_{1,1} = u(0)$
  ELSE
    Receive $\tilde{U}^0_{n-1,\tilde{J}}$ from $\boldsymbol{P}_{n-1}$
    Set $\tilde{U}^0_{n,1} = \tilde{U}^0_{n-1,\tilde{J}}$
  END IF
  COMMENT: Compute solution at coarse time nodes
  FOR $\tilde{j} = 0 \ldots \tilde{J}-1$
    $\tilde{U}^0_{n,\tilde{j}+1} = \tilde{U}^0_{n,\tilde{j}} + \Delta t_{\tilde{j}} \big(F_E(t_{\tilde{j}}, \tilde{U}^0_{n,\tilde{j}}) + F_I(t_{\tilde{j}+1}, \tilde{U}^0_{n,\tilde{j}+1}) \big)$
  END FOR
  COMMENT: Send data forward
  IF $n < N-1$
    Send $\tilde{U}^0_{n,\tilde{J}}$ to $\boldsymbol{P}_{n+1}$
  END IF
END FOR

Parallel iteration:

FOR $k = 1 \ldots K$
  DO in parallel on $\boldsymbol{P}_n$, $n = 0 \ldots N-1$
    COMMENT: Update values at fine time steps
    IF $k = 1$
      INTERPOLATE $\tilde{U}^k_{n,\tilde{j}}$ at coarse times to form $U^k_{n,j}$ at fine points
      COMPUTE $f(t_j, U^k_{n,j})$ for $j = 1 \ldots J$.
    ELSE
      INTERPOLATE $\tilde{U}^k_{n,\tilde{j}} - U^{k-1}_{n,\tilde{j}}$ at coarse times to form $U^k_{n,j}$ at fine points
    END IF

COMMENT: Compute time integral of fine solution at fine nodes
COMPUTE $S_j^{j+1} f(\boldsymbol{t}_n, \mathbf{U}_n^k)$ for $j = 0 \ldots J-1$ using spectral integration

COMMENT: Do a fine SDC sweep
FOR $j = 0 \ldots J-1$
$U_{n,j+1}^k = U_{n,j}^k + \Delta t_j \big(F_E(t_j, U_{n,j}^k) + F_I(t_{j+1}, U_{n,j+1}^k)\big) + S_j^{j+1} f(\boldsymbol{t}_n, \mathbf{U}_n^k)$
END FOR

COMMENT: Compute time integral of fine solution at coarse nodes
COMPUTE $S_{\tilde{j}}^{\tilde{j}+1} f(\tilde{\boldsymbol{t}}_n, \mathbf{U}_n^k)$ $\tilde{j} = 0 \ldots \tilde{J}-1$ by summing $S_j^{j+1} f(\boldsymbol{t}_n, \mathbf{U}_n^k)$

COMMENT: Get new initial data
IF $n = 0$
SET $U_{0,1}^k = \tilde{U}_{0,1}^{k-1}$,
SET $f(t_n, U_{0,1}^k) = f(t_n, \tilde{U}_{0,1}^{k-1})$
ELSE
RECEIVE $\tilde{U}_{n-1,\tilde{J}}^k$ from $\boldsymbol{P}_{n-1}$
SET $U_{n,1}^0 = \tilde{U}_{n-1,\tilde{J}}^k$
COMPUTE $f(t_n, U_{n,1}^k) = f(t_n, U_{n,1}^k)$
END IF

COMMENT: Do a coarse SDC sweep
FOR $\tilde{j} = 0 \ldots \tilde{J}-1$
$\tilde{U}_{n,\tilde{j}+1}^k = \tilde{U}_{n,\tilde{j}}^k + \Delta t_{\tilde{j}} \big(F_E(t_{\tilde{j}}, \tilde{U}_{n,\tilde{j}}^k) + F_I(t_{\tilde{j}+1}, \tilde{U}_{n,\tilde{j}+1}^k)\big) + S_{\tilde{j}}^{\tilde{j}+1} f(\tilde{\boldsymbol{t}}_n, \tilde{U}_n^k)$
END FOR

COMMENT: Send data forward
IF $n < P-1$
Send $U_{n,\tilde{J}}^k$ to $\boldsymbol{P}_{n+1}$
END IF
END DO
END FOR

References

[1] G. Akrivis, M. Crouzeix, and C. Makridakis, *Implicit-explicit multistep finite element methods for nonlinear parabolic problems*, Math. Comp. **67** (1998), no. 222, 457–477. MR 98g:65088 Zbl 0896.65066

[2] ———, *Implicit-explicit multistep methods for quasilinear parabolic equations*, Numer. Math. **82** (1999), no. 4, 521–541. MR 2000e:65075 Zbl 0936.65118

[3] G. Akrivis and Y.-S. Smyrlis, *Implicit-explicit BDF methods for the Kuramoto–Sivashinsky equation*, Appl. Numer. Math. **51** (2004), no. 2-3, 151–169. MR 2005h:65126 Zbl 1057.65069

[4] 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

[5] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Anal. **32** (1995), no. 3, 797–823. MR 96j:65076 Zbl 0841.65081

[6] G. Bal, *Parallelization in time of (stochastic) ordinary differential equations*, preprint, 2003, Available at http://www.columbia.edu/∼gb2030/PAPERS/paralleltime.pdf.

[7] 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

[8] 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. Pavarino and A. Toselli, eds.), Lect. Notes Comput. Sci. Eng., no. 23, Springer, Berlin, 2002, pp. 189–202. MR 1962689

[9] G. Bal and Q. Wu, *Symplectic parareal*, Domain decomposition methods in science and engineering XVII (U. Langer et al., eds.), Lect. Notes Comput. Sci. Eng., no. 60, Springer, Berlin, 2008, pp. 401–408. MR 2436107 Zbl 1140.65372

[10] 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

[11] E. L. Bouzarth, *Regularized singularities and spectral deferred correction methods: A mathematical study of numerically modeling Stokes fluid flow*, Ph.D. thesis, The University of North Carolina at Chapel Hill, 2008. MR 2711923

[12] S. Bu, J. Huang, and M. L. Minion, *Semi-implicit Krylov deferred correction methods for ordinary differential equations*, Proceedings of the 15th American Conference on Applied Mathematics, WSEAS, 2009, pp. 95–100.

[13] K. Burrage, *Parallel methods for ODEs*, Adv. Comput. Math. **7** (1997), no. 1-2, 1–31.

[14] J. C. Butcher, *Order and stability of parallel methods for stiff problems: Parallel methods for odes*, Adv. Comput. Math. **7** (1997), no. 1-2, 79–96. MR 98c:65105 Zbl 0884.65090

[15] M. P. Calvo, J. de Frutos, and J. Novo, *Linearly implicit Runge–Kutta methods for advection-reaction-diffusion equations*, Appl. Numer. Math. **37** (2001), no. 4, 535–549. MR 2002e:65114 Zbl 0983.65106

[16] A. Christlieb, C. Macdonald, and B. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput. **32** (2010), no. 2, 818–835.

[17] A. Christlieb, B. Ong, and J.-M. Qiu, *Integral deferred correction methods constructed with high order Runge–Kutta integrators*, Math. Comp. **79** (2010), no. 270, 761–783. MR 2600542 Zbl 05776244

[18] M. L. Crow and M. Ilic, *The parallel implementation of the waveform relaxation method for transient stability simulations*, IEEE Trans. Power Syst. **5** (1990), no. 3, 922–932.

[19] 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

[20] J. R. Dormand and P. J. Prince, *A family of embedded Runge–Kutta formulae*, J. Comput. Appl. Math. **6** (1980), no. 1, 19–26. MR 81g:65098 Zbl 0448.65045

[21] 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

[22] 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

[23] 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 (T. J. Barth et al., eds.), Lect. Notes Comput. Sci. Eng., no. 40, Springer, Berlin, 2005, pp. 433–440. MR 2235770 Zbl 02143574

[24] J. Frank, W. Hundsdorfer, and J. G. Verwer, *On the stability of implicit-explicit linear multistep methods*, Appl. Numer. Math. **25** (1997), no. 2-3, 193–205. MR 98m:65126 Zbl 0887.65094

[25] M. J. Gander, *A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations*, Numer. Linear Algebra Appl. **6** (1999), no. 2, 125–145. MR 2000m:65110 Zbl 0983.65107

[26] ———, *Analysis of the parareal algorithm applied to hyperbolic problems using characteristics*, Bol. Soc. Esp. Mat. Apl. SẽMA **42** (2008), 21–35. MR 2009b:65268

[27] M. J. Gander and E. Hairer, *Nonlinear convergence analysis for the parareal algorithm*, Domain decomposition methods in science and engineering XVII (U. Langer et al., eds.), Lect. Notes Comput. Sci. Eng., no. 60, Springer, Berlin, 2008, pp. 45–56. MR 2009j:65165 Zbl 1140.65336

[28] M. J. Gander and M. Petcu, *Analysis of a Krylov subspace enhanced parareal algorithm for linear problems*, Paris-Sud Working Group on Modelling and Scientific Computing 2007–2008 (E. Cancès et al., eds.), ESAIM Proc., no. 25, EDP Sci., Les Ulis, 2008, pp. 114–129. MR 2010i:65119 Zbl 1156.65322

[29] M. J. Gander and A. E. Ruehli, *Optimized waveform relaxation methods for RC type circuits*, IEEE Trans. Circuits Syst. I Regul. Pap. **51** (2004), no. 4, 755–768. MR 2005d:94228

[30] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput. **29** (2007), no. 2, 556–578. MR 2008c:65386 Zbl 1141.65064

[31] L. Greengard, *Spectral integration and two-point boundary value problems*, SIAM J. Numer. Anal. **28** (1991), no. 4, 1071–1080. MR 92h:65033 Zbl 0731.65064

[32] D. Guibert and D. Tromeur-Dervout, *Adaptive parareal for systems of ODEs*, Domain decomposition methods in science and engineering XVI (O. Widlund and D. Keyes, eds.), Lect. Notes Comput. Sci. Eng., no. 55, Springer, Berlin, 2007, pp. 587–594. MR 2334151

[33] ———, *Parallel deferred correction method for CFD problems*, Parallel computational fluid dynamics: parallel computing and its applications (J. H. Kwon, A. Ecer, N. Satofuka, J. Periaux, and P. Fox, eds.), Elsevier, Amsterdam, 2007, pp. 131–138.

[34] M. Hu, K. Jackson, J. Janssen, and S. Vandewalle, *Remarks on the optimal convolution kernel for CSOR waveform relaxation: Parallel methods for odes*, Adv. Comput. Math. **7** (1997), no. 1-2, 135–156. MR 98c:65107 Zbl 0889.65069

[35] 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

[36] ———, *Arbitrary order Krylov deferred correction methods for differential algebraic equations*, J. Comput. Phys. **221** (2007), no. 2, 739–760. MR 2008a:65134 Zbl 1110.65076

[37] K. J. in 't Hout, *On the contractivity of implicit-explicit linear multistep methods*, Appl. Numer. Math. **42** (2002), no. 1-3, 201–212. MR 2003j:65065 Zbl 1001.65090

[38] A. Iserles and S. P. Nørsett, *On the theory of parallel Runge–Kutta methods*, IMA J. Numer. Anal. **10** (1990), no. 4, 463–488. MR 91i:65127

[39] 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

[40] M. Kiehl, *Parallel multiple shooting for the solution of initial value problems*, Parallel Comput. **20** (1994), no. 3, 275–295. MR 95c:65099 Zbl 0798.65079

[41] 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

[42] 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

[43] ———, *Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations*, BIT **45** (2005), no. 2, 341–373. MR 2006h:65087 Zbl 1078.65552

[44] ———, *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

[45] 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

[46] Y. Liu and J. Hu, *Modified propagators of parareal in time algorithm and application to Princeton ocean model*, Internat. J. Numer. Methods Fluids **57** (2008), no. 12, 1793–1804. MR 2009g:86009 Zbl 05312577

[47] Y. Maday and G. Turinici, *Parallel in time algorithms for quantum control: Parareal time discretization scheme*, Int. J. Quantum Chem. **93** (2003), 223–228.

[48] Y. Maday, J. Salomon, and G. Turinici, *Monotonic parareal control for quantum systems*, SIAM J. Numer. Anal. **45** (2007), no. 6, 2468–2482. MR 2008k:81003 Zbl 1153.49005

[49] Y. Maday and G. Turinici, *The parareal in time iterative solver: a further direction to parallel implementation*, Domain decomposition methods in science and engineering (R. Kornhuber et al., eds.), Lect. Notes Comput. Sci. Eng., no. 40, Springer, Berlin, 2005, pp. 441–448. MR 2235771 Zbl 1067.65102

[50] M. L. Minion, *Higher-order semi-implicit projection methods*, Numerical simulations of incompressible flows (M. M. Hafez, ed.), World Sci. Publ., River Edge, NJ, 2003, pp. 126–140. MR 1984431 Zbl 1079.76056

[51] ———, *Semi-implicit spectral deferred correction methods for ordinary differential equations*, Commun. Math. Sci. **1** (2003), no. 3, 471–500. MR 2005f:65085 Zbl 1088.65556

[52] ———, *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

[53] M. L. Minion and S. A. Williams, *Parareal and spectral deferred corrections*, Numerical analysis and applied mathematics (T. E. Simos, ed.), AIP Conference Proceedings, no. 1048, AIP, 2008, pp. 388–391.

[54] W. L. Miranker and W. Liniger, *Parallel methods for the numerical integration of ordinary differential equations*, Math. Comp. **21** (1967), 303–320. MR 36 #6155 Zbl 0155.47204

[55] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Comm. ACM **7** (1964), 731–733. MR 31 #889 Zbl 0134.32804

[56] 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 Sci. Publ., Huntington, NY, 2001, pp. 269–288. MR 2005a:65065 Zbl 1018.65093

[57] 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

[58] _____, *Iterated deferred corrections for nonlinear operator equations*, Numer. Math. **10** (1967), 316–323. MR 36 #4812 Zbl 0258.65059

[59] 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, 1996, pp. 17–20.

[60] G. A. Staff and E. M. Rønquist, *Stability of the parareal algorithm*, Domain decomposition methods in science and engineering (R. Kornhuber et al., eds.), Lect. Notes Comput. Sci. Eng., no. 40, Springer, Berlin, 2005, pp. 449–456. MR 2235772

[61] S. Vandewalle and D. Roose, *The parallel waveform relaxation multigrid method*, Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing, Soc. Indust. Appl. Math., 1989, pp. 152–156.

[62] S. L. Wu, B. C. Shi, and C. M. Huang, *Parareal-Richardson algorithm for solving nonlinear ODEs and PDEs*, Commun. Comput. Phys. **6** (2009), no. 4, 883–902.

[63] P. E. 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 (G. Contopoulos, ed.), Proc. Int. Astron. Union Symp., no. 25, Academic Press, London, 1964.

[64] X. Zhong, *Additive semi-implicit Runge–Kutta methods for computing high-speed nonequilibrium reactive flows*, J. Comput. Phys. **128** (1996), no. 1, 19–31. MR 97e:80019 Zbl 0861.76057

Received January 18, 2010. Revised October 16, 2010.

MICHAEL L. MINION: minion@email.unc.edu
Department of Mathematics, University of North Carolina – Chapel Hill, Campus Box 3250, Chapel Hill, NC 27514-3250, United States
amath.unc.edu/Minion

Guidelines for Authors

Authors may submit manuscripts in PDF format on-line at the Submission page at pjm.math.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@mathscipub.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. 5 no. 2 2010