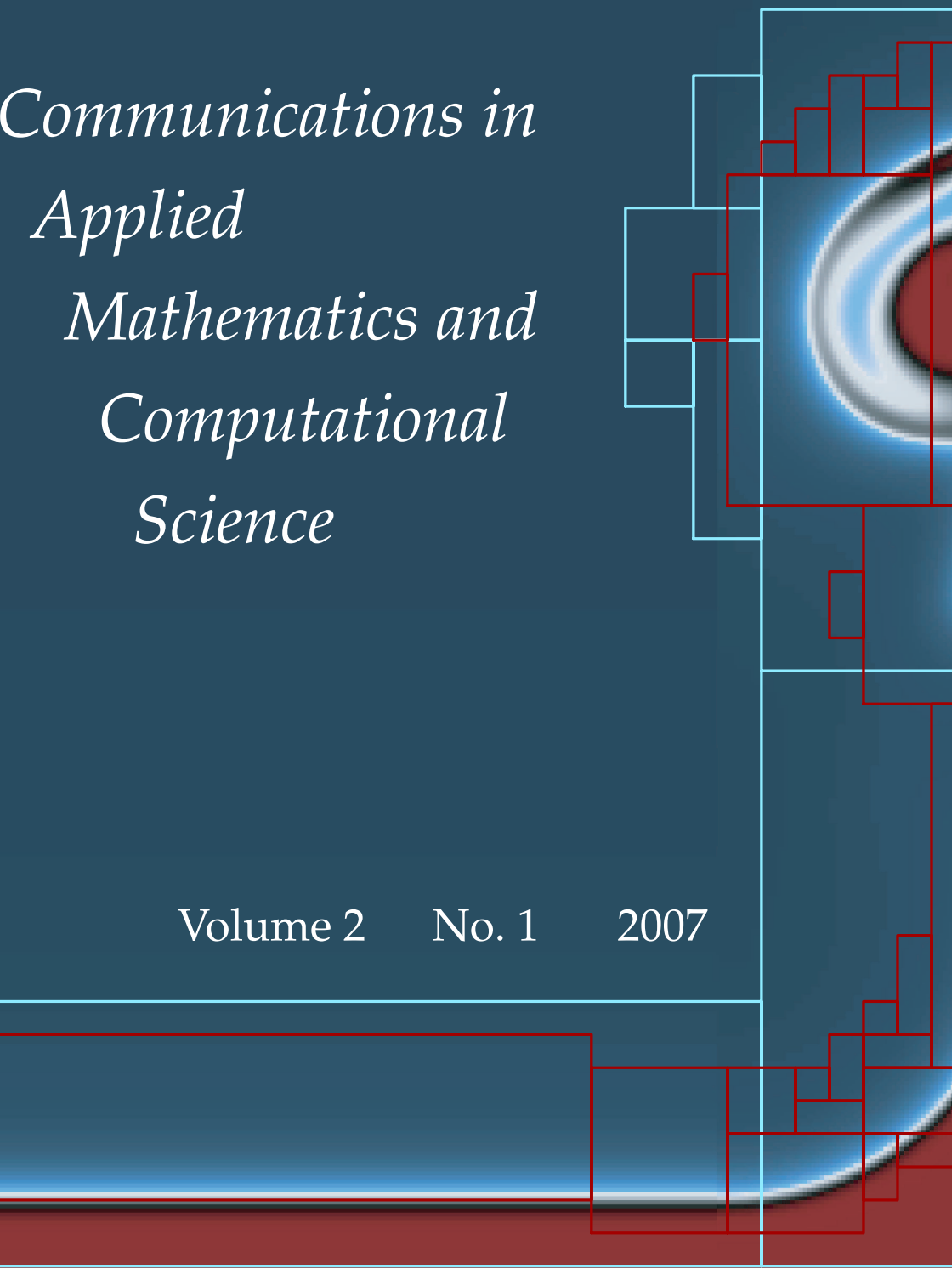
Communications in Applied Mathematics and Computational Science



Volume 2 No. 1 2007

mathematical sciences publishers

COMM. APP. MATH. AND COMP. SCI.
Vol. 2, No. 1, 2007

IMPLICATIONS OF THE CHOICE OF PREDICTORS FOR SEMI-IMPLICIT PICARD INTEGRAL DEFERRED CORRECTION METHODS

ANITA T. LAYTON AND MICHAEL L. MINION

High-order semi-implicit Picard integral deferred correction (SIPIDC) methods have previously been proposed for the time-integration of partial differential equations with two or more disparate time scales. The SIPIDC methods studied to date compute a high-order approximation by first computing a provisional solution with a first-order semi-implicit method and then using a similar semi-implicit method to solve a series of correction equations, each of which raises the order of accuracy of the solution by one. This study assesses the efficiency of SIPIDC methods that instead use standard semi-implicit methods with orders two through four to compute the provisional solution. Numerical results indicate that using a method with more than first-order accuracy in the computation of the provisional solution increases the efficiency of SIPIDC methods in some cases. First-order PIDC corrections can improve the efficiency of semi-implicit integration methods based on backward difference formulae (BDF) or Runge–Kutta methods while maintaining desirable stability properties. Finally, the phenomenon of order reduction, which may be encountered in the integration of stiff problems, can be partially alleviated by the use of BDF methods in the computation of the provisional solution.

1. Introduction

The dynamics of many physical and biological systems of interest today involve processes with two or more characteristic time scales. When the time scales of the physical processes vary widely, efficient time-marching of the partial differential equations (PDEs) that describe the dynamics may require specialized numerical methods, particularly when one wishes to accurately resolve processes at each time

MSC2000: primary 65B05; secondary 65L20.
Keywords: semi-implicit methods, deferred correction methods, order reduction.
A. T. Layton was supported in part by the National Science Foundation, grant DMS-0340654. M. L. Minion was supported in part under contract DE-AC03-76SF00098 by the Director, Department of Energy (DOE) Office of Science; Office of Advanced Scientific Computing Research; Office of Mathematics, Information, and Computational Sciences; Applied Mathematics Sciences Program; and by the Alexander von Humboldt Foundation.

scale. For example, following the method-of-lines approach, when the PDEs are discretized in space, the resulting system of coupled ordinary differential equations (ODEs) typically contains both stiff and nonstiff terms. When the stiffness of one of these terms corresponds to eigenvalues with a large negative real part (for example, from the discretization of a diffusive term), an implicit treatment of this term can allow a much larger stable time step (without significantly sacrificing accuracy) than an explicit treatment. Hence, the use of semi-implicit methods for such systems, that is, methods that treat only the stiff terms implicitly, can result in a considerable improvement in efficiency compared to fully implicit methods, particularly when other nonstiff terms in the equations are computationally expensive to treat implicitly. Provided that a sufficiently high level of accuracy is desired, and/or the temporal interval is sufficiently long, high-order methods for ODEs are more efficient than low-order methods in that less computational cost is required by high-order methods to achieve a given, sufficiently stringent error tolerance. Hence the construction of stable and efficient higher-order semi-implicit methods for ODEs is desirable.

Indeed, semi-implicit (also known as implicit-explicit or IMEX) versions of popular time-integration methods such as Runge–Kutta (RK), linear multistep, or backward difference formulae (BDF) methods have been developed to efficiently integrate ODEs with both nonstiff and stiff components. Semi-implicit RK methods have been proposed and tested by a number of authors [Ascher et al. 1997; Kennedy and Carpenter 2003; Pareschi and Russo 2001; Shen and Zhong 1996; Calvo et al. 2001]; however, owing in part to the complexity of deriving such schemes, only semi-implicit RK methods with order up to five have so far been developed. Similarly, several papers have analyzed the stability and accuracy of semi-implicit methods derived from linear multistep methods [Akrivis et al. 1999; Ascher et al. 1995; Frank et al. 1997; in't Hout 2002]. In this case, stable schemes up to order six are easily constructed, although higher-order versions have the disadvantages that they require multiple starting values, require care when used with variable time stepping schemes, and, as further discussed in Section 3, have less satisfactory stability characteristics.

In a series of studies [Minion 2003; Minion 2004; Layton and Minion 2005], we developed and analyzed a new class of semi-implicit methods for integrating ODEs that arise from a method-of-lines discretization of PDEs involving time-scale disparity. The methods are based on a semi-implicit Picard integral deferred correction (SIPIDC) approach, which is a generalization of the explicit and implicit spectral deferred correction (SDC) methods introduced in [Dutt et al. 2000]. SDC methods use a low-order numerical method to compute an approximate solution with an arbitrarily high order of accuracy. This is achieved by using the low-order numerical method to solve a series of correction equations, each of which increases the order of accuracy of the approximation.

The SDC methods introduced in [Dutt et al. 2000] and the SIPIDC methods described in [Layton and Minion 2005], as well as most of the SISDC methods described in [Minion 2003], use a first-order method both to compute the provisional solution and to approximate the correction equations. It has previously been demonstrated that higher-order versions of these methods are more efficient than lower-order methods, and that the stability properties of the methods with very high order remain similar to those with lower order [Dutt et al. 2000; Minion 2003]. A reasonable question to ask is whether the efficiency of SIPIDC methods can be improved by using a semi-implicit method with higher than first-order accuracy to compute the provisional solution. (We will refer to the standard method used to compute the provisional solution in a particular PIDC method as the *predictor*.) Hence we wish to investigate whether using a semi-implicit BDF or RK method as the predictor in a PIDC method improves the overall efficiency of the PIDC method. PIDC methods using a predictor with higher than first-order accuracy require fewer iterations of the correction equation to achieve the same overall order of accuracy relative to methods using a first-order predictor. However, it is not immediately clear if the lower computational cost comes at the expense of a loss in accuracy, or if using such a predictor negatively affects the stability of PIDC methods. Another relevant question addressed here is whether performing a series of SIPIDC corrections on a solution generated from a semi-implicit BDF or RK method results in a SIPIDC method with greater numerical efficiency than that afforded by simply using the base methods alone. The primary goal of this paper is to address these questions using the linear stability analysis in Section 3 and numerical tests in Section 4.

A further issue addressed here concerns order reduction for stiff problems, something observed in connection with both SIPIDC methods and semi-implicit RK methods [Minion 2003; Layton and Minion 2005; Kennedy and Carpenter 2003]. In [Layton and Minion 2005] the effect of the choice of quadrature nodes on the extent and character of order reduction of SIPIDC methods on stiff problems is investigated. Both analytical and numerical results in [Layton and Minion 2005] show that, for a sufficiently stiff problem, SIPIDC methods using a first-order forward-backward Euler predictor exhibit a reduction of order for a range of time steps, and the extent and character of the reduction depends on the choice of quadrature rule used in the method. The results presented in Section 4.3 show that the extent and character of order reduction also depend critically on the predictor. Specifically, when a kth-order IMEX BDF predictor is used with uniform quadrature nodes, the convergence rate in the region of order reduction is $k-1$, compared to a reduction to first-order when an IMEX RK predictor (regardless of order) is used.

2. SIPIDC methods

Below is a short description of SIPIDC methods. A detailed derivation of the SIPIDC methods for ODEs and for an advection-diffusion-reaction equation can be found in [Minion 2003] and [Bourlioux et al. 2003]. The target ODE takes the form

$$\begin{aligned} u'(t) &= F_E\big(t, u(t)\big) + F_I\big(t, u(t)\big), \qquad t \in [a, b] \\ u(a) &= u_a, \end{aligned} \tag{1}$$

where F_I is assumed to be significantly stiffer than F_E. Thus, SIPIDC methods compute $u(t)$ by integrating F_E explicitly and F_I implicitly.

Without loss of generality, a uniform time step $\Delta t = (b-a)/N_T$, for some positive integer N_T, is assumed in the numerical discretization. Let $t_n = n\Delta t$, for $n = 0, 1, 2, \ldots, N_T$, be the n-th time-level. In the integration of the solution from t_n to t_{n+1}, the time interval $[t_n, t_{n+1}]$ is divided into P subintervals by choosing points $t_{n,m}$ for $m = 0, 1, \ldots, P$ such that $t_n = t_{n,0} < t_{n,1} < \cdots < t_{n,m} < \cdots < t_{n,P} \leq t_{n+1}$. For notational simplicity, the subscript n in $t_{n,m}$ is omitted and $t_{n,m}$ is written as t_m. Let $\Delta t_m \equiv t_{m+1} - t_m$; the interval $[t_m, t_{m+1}]$ is referred to as a *substep*.

For an arbitrary function $\psi(t)$, let ψ^k and ψ_m^k denote numerical approximations to $\psi(t)$ and $\psi(t_m)$, respectively, after k iterations. To advance the solution from t_n to t_{n+1}, a SIPIDC method first computes a provisional solution $\tilde{u}(t_m) \equiv u_m^0$, for $m = 0, 1, \ldots, P$, by means of a semi-implicit method that we refer to as the predictor. Presumably any method could be chosen to compute the provisional solution, and the main point of this paper is to investigate the relative performance of SIPIDC methods using different predictors.

Given a provisional solution $\tilde{u}(t)$, the accuracy of that solution can be improved using an estimate of its error (or correction): $u(t) - \tilde{u}(t)$, denoted by $\delta(t)$. Using the Picard integral form of the solution to Equation (1), one can express $\delta(t)$ as the solution to the integral equation

$$\delta(t) = \int_a^t \big(F_E(\tau, \tilde{u}+\delta) - F_E(\tau, \tilde{u}) + F_I(\tau, \tilde{u}+\delta) - F_I(\tau, \tilde{u})\big)\, d\tau + E(t, \tilde{u}), \tag{2}$$

where E is the residual function given by

$$E(t, \tilde{u}) = u_0 + \int_a^t F_E(\tau, \tilde{u}) + F_I(\tau, \tilde{u})\, d\tau - \tilde{u}(t).$$

We have suppressed the time dependence of $\tilde{u}(t)$ and $\delta(t)$ in the integrands to avoid clutter. A detailed derivation of Equation (2) is given in [Minion 2003].

In SIPIDC methods, a semi-implicit discretization of Equation (2) is used to iteratively increase the order of accuracy of the provisional solution, that is,

$$u_m^{k+1} = u_m^k + \delta_m^k.$$

Specifically, a forward-backward Euler method for computing δ_m^k is given by

$$\delta_{m+1}^k = \delta_m^k + \Delta t_m \big(F_E(u_m^{k+1}) - F_E(u_m^k) + F_I(u_{m+1}^{k+1}) - F_I(u_{m+1}^k)\big) + E_{m+1}(u^k) - E_m(u^k), \quad (3)$$

where the terms $E_m(u^k)$ are approximated with numerical quadrature. Let

$$\mathfrak{Q}_m^{m+1}(F)$$

be a Pth-order numerical quadrature approximation to $\int_{t_m}^{t_{m+1}} F(\tau)d\tau$, that is,

$$\mathfrak{Q}_m^{m+1}(F) = \Delta t_m \sum_{l=0}^{p} q_l^m F_l = \int_{t_m}^{t_{m+1}} F(\tau)d\tau + \mathcal{O}(\Delta t^{P+1}). \quad (4)$$

By adding u^k to both sides of (3), one obtains a direct update equation that can be used to improve the accuracy of u^k:

$$u_{m+1}^{k+1} = u_m^{k+1} + \Delta t_m \big(F_E(u_m^{k+1}) - F_E(u_m^k) + F_I(u_{m+1}^{k+1}) - F_I(u_{m+1}^k)\big) + \mathfrak{Q}_m^{m+1}\big(F_E(u^k) + F_I(u^k)\big). \quad (5)$$

Equation (5) is solved at the k-th iteration, referred to as a deferred correction iteration; see [Minion 2003] for details. The quadrature $\mathfrak{Q}$ should have at least the same order of accuracy as the updated approximation u^{k+1}. As in [Bourlioux et al. 2003; Minion 2003], the quadrature $\mathfrak{Q}_m^{m+1}$ is computed as the integral of an interpolating polynomial over the subinterval $[t_m, t_{m+1}]$ (see further discussion below).

In the SDC methods presented in [Dutt et al. 2000], the points t_m are chosen to be the Gaussian quadrature nodes of the interval $[t_n, t_{n+1}]$, and the solution is only integrated at these nodes on the interior of the interval. In [Bourlioux et al. 2003; Minion 2003] the points t_m are chosen to be Gauss–Lobatto quadrature nodes, which are more convenient in that the solution is directly computed at both endpoints of the time step interval. However, because Gauss–Lobatto nodes are not evenly spaced for orders of accuracy >2, predictors with higher than first order are less convenient to implement (nonetheless, it can be done; see [Minion 2003]). This is particularly true if the SIPIDC method is used for the temporal integration of PDEs in which block structured adaptive mesh refinement is used (see [Berger and Oliger 1984]). In this instance (currently a topic of research by the authors), the colocation of coarse and fine grid data requires the use of uniform substeps. Other examples in which it is either convenient or necessary to choose substeps that are uniformly spaced have been discussed in [Layton and Minion 2005]. For these reasons, SIPIDC methods presented here use uniform nodes such that $\Delta t_1 = \cdots = \Delta t_m \cdots \equiv \Delta t_s$.

The form of the quadrature rule also has a significant effect on the stability and accuracy of the SIPIDC method for stiff equations. The methods in [Layton and Minion 2005] actually use two separate quadrature rules for the two terms in $\mathbb{Q}_m^{m+1}\big(F_E(u^k)+F_I(u^k)\big)$ in Equation (4). It is shown in [Layton and Minion 2005] that, when function values at the left endpoint t_n are omitted in the quadrature rule associated with the stiff component (that is, $q_0^m = 0$ for all m), the resulting SIPIDC method is $L(\alpha)$-stable. Also, by including the left endpoint in the nonstiff quadrature rule, the accuracy of the quadrature associated with the explicit piece is improved. This choice of quadrature rules, denoted *LR* in [Layton and Minion 2005], is adopted in this study. To construct a method with Kth-order accuracy, the quadrature $\mathbb{Q}$ should also have at least Kth-order accuracy. If $P+1$ nodes (or P substeps) are used in the interval $[t_n, t_{n+1}]$, uniform integration nodes yield order P accuracy for the integral $\mathbb{Q}_m^{m+1}$ over the subinterval $[t_m, t_{m+1}]$. Thus, to construct a Kth-order SIPIDC method with uniform nodes that uses the LR quadrature rule (which excludes the left endpoints in the stiff quadrature rule), $K+1$ nodes or K substeps are required.

2.1. ***Moderate-order predictors.*** The SDC methods presented in [Bourlioux et al. 2003; Dutt et al. 2000] are based on forward-backward Euler methods; that is, the prediction and correction steps are first order. Because higher-order methods are generally more efficient than lower-order methods, we investigate SIPIDC methods that are based on second- through fourth-order semi-implicit methods in the prediction step. We refer to these predictors as *moderate-order* predictors. The methods that we use for computing the provisional solution are based on Euler methods, IMEX BDF [Ascher et al. 1995], IMEX RK methods [Ascher et al. 1997; Kennedy and Carpenter 2003], and classical Adams-type multistep methods. These methods, chosen either for their popularity or known stability, are described below.

IMEX BDF methods. BDF methods are a class of linear multistep methods specifically developed for the solution of stiff ODEs. Hence it is natural when constructing semi-implicit generalizations of linear multistep methods to base the treatment of the stiff piece of the equation on BDF methods. IMEX BDF methods have been previously studied [Ascher et al. 1995; Akrivis et al. 1998; in't Hout 2002]. Here we use second- through fourth-order semi-implicit BDF methods from [Ascher et al. 1995] (denoted BDF2, BDF3, and BDF4) in the provisional step. Forward-backward Euler methods, which can be considered as either a first-order IMEX BDF or IMEX RK method, are included here. For brevity, IMEX BDF will be referred to simply as BDF. The specific formulae are

$$\text{Euler: } u_{m+1}^0 = u_m^0 + \Delta t_s\big(F_E(u_m^0) + F_I(u_{m+1}^0)\big), \tag{6}$$

$$\text{BDF2: } \tfrac{3}{2}u^0_{m+1} = 2u^0_m - \tfrac{1}{2}u^0_{m-1} + \Delta t_s\big(2F_E(u^0_m) - F_E(u^0_{m-1}) + F_I(u^0_{m+1})\big), \quad (7)$$

$$\text{BDF3: } \tfrac{11}{6}u^0_{m+1} = 3u^0_m - \tfrac{3}{2}u^0_{m-1} + \tfrac{1}{3}u^0_{m-2} + \Delta t_s\big(3F_E(u^0_m) - 3F_E(u^0_{m-1}) + F_E(u^0_{m-2}) + F_I(u^0_{m+1})\big), \quad (8)$$

$$\text{BDF4: } \tfrac{25}{12}u^0_{m+1} = 4u^0_m - 3u^0_{m-1} + \tfrac{4}{3}u^0_{m-2} - \tfrac{1}{4}u^0_{m-3} + \Delta t_s\big(4F_E(u^0_m) - 6F_E(u^0_{m-1}) + 4F_E(u^0_{m-2}) - F_E(u^0_{m-3}) + F_I(u^0_{m+1})\big). \quad (9)$$

IMEX RK methods. There are several different implementations of second-order IMEX RK methods. The one used in this study is a two-stage L-stable RK2 method described by Ascher et al [Ascher et al. 1997]. This particular implementation of IMEX RK2 is chosen owing to its L-stability, even though it requires two stages and is thus more costly than some alternative implementations (for example, the IMEX midpoint [Ascher et al. 1997]). The L-stable IMEX RK2 method, which we refer to as RK2 for brevity, generates a provisional solution as follows:

$$\begin{aligned}
\text{RK2: } \phi^{(1)}_{m+c_1} &= u^0_m + c_1\Delta t_s\big(F_E(u^0_m) + F_I(\phi^{(1)}_{m+c_1})\big),\\
\phi^{(2)}_{m+1} &= u^0_m + \Delta t_s\Big(c_2F_E(u^0_m) + (1-c_2)F_E(\phi^{(1)}_{m+c_1})\\
&\qquad + (1-c_1)F_I(\phi^{(1)}_{m+c_1}) + c_1F_I(\phi^{(2)}_{m+1})\Big),\\
u^0_{m+1} &= u^0_m + \Delta t_s\Big((1-c_1)\big(F_E(\phi^{(1)}_{m+c_1}) + F_I(\phi^{(1)}_{m+c_1})\big)\\
&\qquad + c_1\big(F_E(\phi^{(2)}_{m+1}) + F_I(\phi^{(2)}_{m+1})\big)\Big),
\end{aligned}$$

where $c_1 = 1 - \sqrt{2}/2$, $c_2 = -2\sqrt{2}/3$.

The third- and fourth-order IMEX RK methods used here are based on the Additive RK methods developed by Kennedy and Carpenter [Kennedy and Carpenter 2003], specifically, the ARK3(2)4L[2]SA-ERK and ARK4(3)6L[2]SA methods. These methods, which we refer to simply as ARK3 and ARK4, involve three and five stages, respectively, and we refer interested reader to [Kennedy and Carpenter 2003] for the relevant coefficients. There is a fifth-order ARK method introduced in [Kennedy and Carpenter 2003], but Kennedy and Carpenter determine that it is not competitive with the fourth-order methods, and hence we do not study it here. We know of no IMEX RK methods of order greater than five in the literature (although it is possible to construct such methods).

Multistep methods. We also investigate the well known second- and third-order multistep methods: Crank–Nicolson/Adam–Bashforth (CNAB) and Adam–Bashforth/Adam–Moulton (ABAM) methods. When these methods are used, the provisional solution is given by:

$$\begin{aligned}
\text{CNAB: } u^0_{m+1} &= u^0_m + \Delta t_s \Big(\tfrac{3}{2} F_E(u^0_m) \\
&\qquad - \tfrac{1}{2} F_E(u^0_{m-1}) + \tfrac{1}{2} F_I(u^0_{m+1}) + \tfrac{1}{2} F_I(u^0_m) \Big), \\
\text{ABAM: } u^0_{m+1} &= u^0_m + \tfrac{\Delta t_s}{12} \Big(23 F_E(u^0_m) - 16 F_E(u^0_{m-1}) + 5 F_E(u^0_{m-2}) \\
&\qquad + 5 F_I(u^0_{m+1}) + 8 F_I(u^0_m) - F_I(u^0_{m-1}) \Big). \qquad (10)
\end{aligned}$$

Both BDF and multistep methods require function values from multiple previous time-levels, values that are not available at the initial substeps of the first time step $[t_0, t_1]$. To generate these starting values for a Kth-order SIPIDC method, initial conditions at t_0 are advanced to t_1 using one time step (or K substeps) of the Kth-order SIPIDC method that uses the forward-backward Euler method in the prediction step. The substep values from this first step are then used as starting values for the subsequent time steps.

We use the notation SIPIDCK[P$_{\text{name}}$] to denote a Kth-order SIPIDC method using P$_{\text{name}}$ as the predictor, where P$_{\text{name}}$ is one of the methods described above. The forward-backward Euler method in (3) is used in the correction steps, hence, if a pth-order predictor is used to construct a Kth-order SIPIDC method, then the correction equation must be iterated $K - p$ times.

3. Linear stability analysis

One of the motivations for the development of high-order SIPIDC methods is that stable methods with very high order of accuracy can be easily constructed. This is in contrast to BDF methods, the stability of which degrades significantly as the order increases, and to IMEX RK methods, where no methods of order greater than five are known. When using either of these methods as predictors in a SIPIDC method, it is important to understand the effect these predictors have on the stability of the overall method.

Hence, the linear stability of SIPIDC methods using BDF or RK predictors is studied in this section. Traditionally, the stability of single-step implicit or explicit methods is studied by considering the model problem

$$\begin{aligned}
u'(t) &= \lambda u(t), \\
u(0) &= 1, \qquad (11)
\end{aligned}$$

for some complex constant λ. By applying a numerical method to this problem, one can derive an amplification factor $\rho(\lambda\Delta t)$, such that

$$u_{n+1} = \rho(\lambda\Delta t)u_n,$$

where u_n is the numerical solution at the nth time step.

When studying the linear stability of semi-implicit methods, one must specify how the standard model problem Equation (11) is decomposed into explicit and implicit parts. Numerous choices of the splitting have appeared in the literature [Frank et al. 1997; Ascher et al. 1995; Pareschi and Russo 2001; Zhong 1996; Pareschi and Russo 2005]. The most general approach is to decompose the problem into explicit and implicit terms by

$$u'(t) = \lambda_E u + \lambda_I u,$$
$$u(0) = 1,$$

where λ_E and λ_I are complex constants [Frank et al. 1997; Pareschi and Russo 2001; Liotta et al. 2000; Pareschi and Russo 2005; Zhong 1996]. Then additional constraints are made to define a stability region which depends only on a single complex number. For example in [Frank et al. 1997] the stability region is defined as the set of λ_I such that the method is stable for all λ_E in the stability region of the explicit method. This approach is also used in [Layton and Minion 2005] but is not used in the following comparisons, since by this definition a method could have a very large stability region despite a severe restriction on the step size due to the properties of the explicit method. Instead, the procedure used in [Ascher et al. 1995; Ascher et al. 1997; Minion 2003] is followed, wherein the imaginary part of the right side of Equation (11) is associated with the nonstiff process and treated explicitly, while the real part is associated with the stiff process and treated implicitly. It should be noted that, regardless of the choice of splitting, the scalar stability analysis only carries over to linear systems when the matrices which define the explicit and implicit terms are simultaneously diagonalizable.

SIPIDC methods using a p-step method in the prediction step advance $u(t_n)$ to $u(t_{n+1})$ using p starting values $u_{n-1,P}(\equiv u_n)$, $u_{n-1,P-1}, \ldots, u_{n-1,P-p+1}$, where P denotes the number of substeps. Let $\vec{u}_n$ denote the vector $(u_{n-1,0}, u_{n-1,1}, \ldots, u_{n-1,P})$. Then the procedure for advancing u_n to u_{n+1} can be written in matrix form:

$$M(\lambda\Delta t)\vec{u}_n = \vec{u}_{n+1},$$

where $M \in \Re^{P\times P}$ and depends on the product $\lambda\Delta t$. To define the stability region for this method, set $\Delta t = 1$ and denote by $\rho(\lambda)$, the maximum magnitude of the eigenvalues $M(\lambda)$. The stability region is then the set of λ such that $\rho(\lambda) \leq 1$. For SIPIDC methods with single-step predictors, this definition reduces to the usual

definition of the amplification factor of a method. In the following, the stability regions for SIPIDC methods with multistep predictors are numerically computed by setting $\vec{u}_n$ to be e_j for $j = 1, \ldots, P$, where the i-th entry of e_j is given by

$$(e_j)_i = \begin{cases} 0, & i \neq j, \\ 1, & i = j. \end{cases}$$

For each λ, the resulting P vectors $\vec{u}_{n+1}$ form the P columns of $M(\lambda)$. MATLAB is used to compute the maximum of the magnitude of eigenvalues of $M(\lambda)$ at a regular array of points in the complex plane. The condition number of the eigenvalues are also monitored to check for degenerate eigenvalues with magnitude near 1, but none were found. The standard definition of $A(\alpha)$-stability [Widlund 1967] is easily extended to the semi-implicit case by defining a method to be $A(\alpha)$-stable for some $\alpha > 0$, if the defined stability region contains the region $\lambda = re^{i\theta}$, for all $\theta \in [\pi - \alpha, \pi + \alpha]$.

It is well known that the size of the stability region for implicit BDF methods decreases as the order increases; indeed, BDF methods with order above six are not acceptable [Gear 1971]. However, the stability properties of IMEX versions of these methods are not as well known and hence are investigated here. The numerically computed stability diagrams for IMEX BDF methods of orders 2, 3, 4, 6, and 7 are displayed in Figure 1. In this and all other figures in this section, the axes are scaled cubically to show both detail near the origin and the general

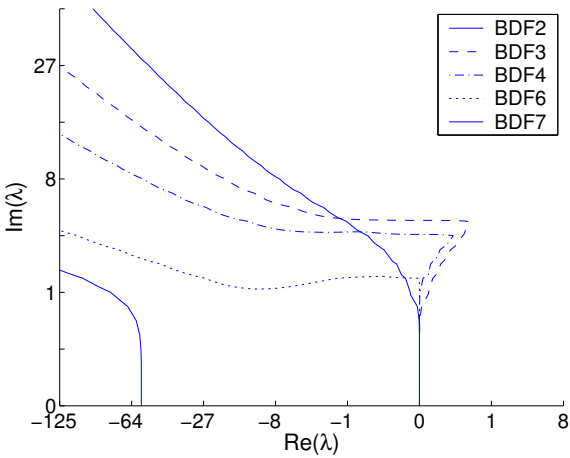


Figure 1. Stability diagrams for second-, third-, fourth-, sixth-, and seventh-order IMEX BDF. Stability regions for IMEX BDF decrease significantly as the order increases, and BDF7 is not stable near the origin.

shape of the stability region in the left half of the complex plane. Figure 1 shows that, as with fully implicit methods, the size of the stability regions of IMEX BDF methods decreases significantly as the order of the method increases, and that the seventh-order method is not stable near the origin. In particular, each method is $A(\alpha)$-stable with α decreasing with increasing order. The stability of certain IMEX RK methods of orders up to three has been studied previously (e.g. [Ascher et al. 1997]). For completeness, we include a plot of the stability regions of the ARK methods of orders 3 and 4 used in this study in Figure 2, which demonstrates that both methods are $A(\alpha)$-stable with similar stability regions. As noted previously, we do not consider fifth-order ARK as it has been deemed to be not competitive [Kennedy and Carpenter 2003], and we are not aware of sixth- or higher-order IMEX RK methods.

Extending the standard definition of L-stability [Ehle 1969] requires care since

$$\lim_{\Re(\lambda)\to-\infty} \rho(\lambda) \tag{12}$$

will in general depend on how the limit is taken. Here we define a method to be $L(\alpha)$-stable if it is $A(\alpha)$-stable and the limit in Equation (12) is zero whenever the imaginary part of λ is fixed in the limit, that is,

$$\lim_{\substack{\Re(\lambda)\to-\infty \\ \Im(\lambda)\equiv c}} \rho(\lambda) = 0, \tag{13}$$

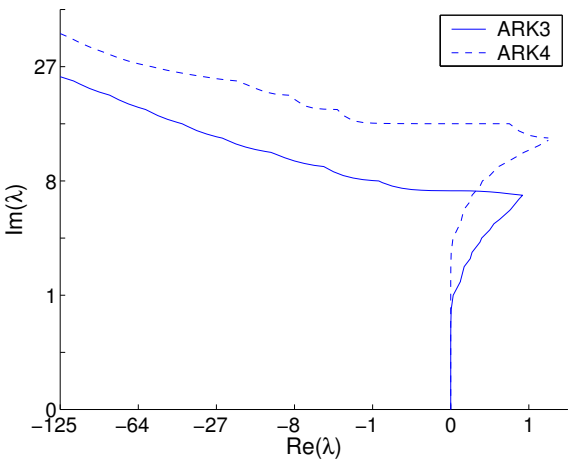


Figure 2. Stability diagrams for third- and fourth-order ARK methods.

for all $c \in \Re$. This, for example, would be the relevant infinitely diffusive stability limit of an approximation to an advection-diffusion equation based on finite differences and the method of lines.

It is shown in [Layton and Minion 2005] that $A(\alpha)$-stable SIPIDC methods can be constructed using forward-backward Euler methods, and that those methods using LR quadrature rules are also $L(\alpha)$-stable. Given an SIPIDC method for which the corrector is based on the forward-backward Euler method and for which the quadrature rule for the implicit piece does not include the left endpoint, one can show that if the predictor satisfies Equation (13), then the overall scheme will also (see [Layton and Minion 2005, Theorems 3.1–3.3].) Hence, since the IMEX BDF and RK methods that are used as predictors in this paper are $L(\alpha)$-stable, $A(\alpha)$-stability for the SIPIDC methods in this paper implies $L(\alpha)$-stability. As an example, stability regions for the SIPIDC6[ARK3] method are shown in Figure 3. In this figure, stability curves corresponding to $\rho(\lambda) = 0.001$ 0.01, 0.1, and 1 are shown to demonstrate that the method is $L(\alpha)$-stable.

An $L(\alpha)$-stable method can also be constructed using BDF3 in the prediction step (not shown). Note also that the stability region of the SIPIDC6[ARK3] method corresponding to $\rho = 1$ in Figure 3 is significantly larger than the stability region of IMEX BDF6 (see Figure 1). However, the SIPIDC6[ARK3] method is also computationally more expensive than IMEX BDF6, owing to the deferred correction iterations and the multiple stages. Thus, it is not immediately clear that for a given

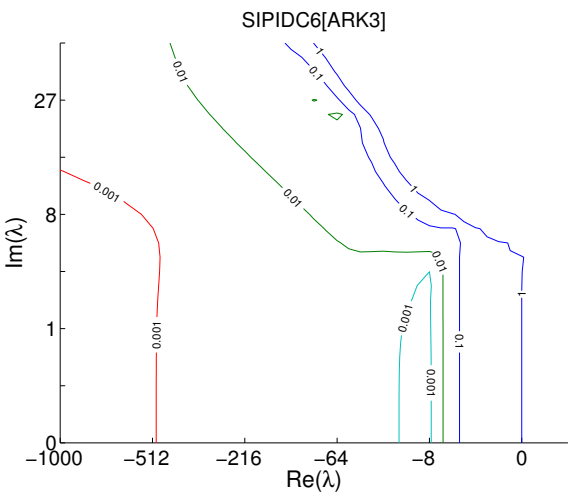


Figure 3. Contours of the amplification factor $\rho(\lambda)$ for the SIPIDC6 method using ARK3 in the provisional step. SIPIDC6[BDF3] is $L(\alpha)$-stable.

computational cost, the SIPIDC[ARK3] has a larger stability region than IMEX BDF6. This issue is further investigated below using *scaled* stability diagrams.

We will now use three different examples to demonstrate the main point of this section, namely that higher-order SIPIDC methods using moderate-order predictors have similar stability regions as the predictors. A corollary to this is that combining moderate-order IMEX BDF or ARK methods with SIPIDC corrections results in a higher-order method with better stability characteristics than the corresponding higher-order IMEX BDF or RK methods. We will demonstrate these points with three separate comparisons:

(1) a comparison of SIPIDC methods of a fixed order using different types of predictors of the same order (that is, IMEX BDF, RK, or multistep);

(2) a comparison of SIPIDC methods of a fixed order using one specific type of predictor with differing orders;

(3) a comparison of SIPIDC methods of differing order using one specific type of predictor with fixed order.

In the first example, we consider the effect of applying SIPIDC corrections on the stability region of different third-order predictors. To this end, we obtain stability diagrams (contour curves of $|\rho| = 1$) for IMEX BDF3, ARK3, and ABAM. These stability diagrams, shown in Figure 4A, indicate that BDF3 and ARK3 are $A(\alpha)$-stable, whereas ABAM is not. SIPIDC methods using the above three methods as predictors (not shown) exhibit similar stability properties as the predictors, that is, SIPIDC6[BDF3] and SIPIDC6[ARK3] are $A(\alpha)$-stable, but SIPIDC6[ABAM] is not.

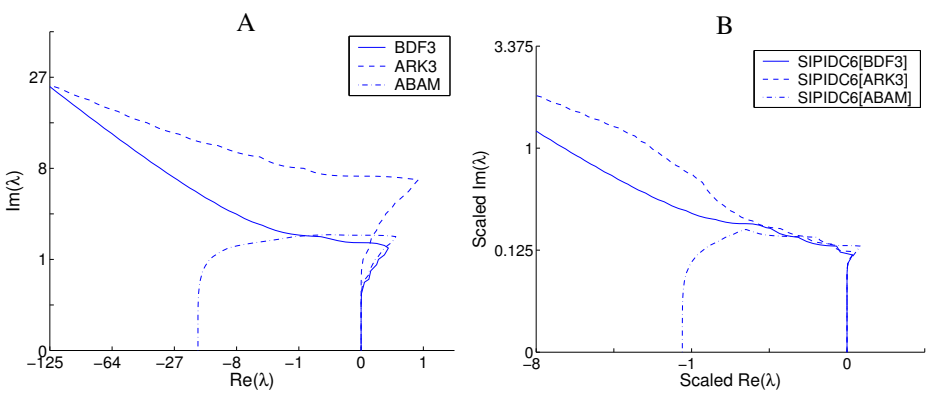


Figure 4. A: stability diagrams for three third-order ODE methods: IMEX BDF3, ARK3, and ABAM. B: scaled stability diagrams for SIPIDC6 methods using the methods in panel A as predictors.

Although the ARK3 method has the largest stability region of the three predictors above (see Figure 4A), ARK3 is also more computationally expensive owing to the multiple stages required. To take into account the additional computational costs, we show *scaled* stability diagrams for SIPIDC6 methods using the three predictors in Figure 4B. By assuming that the solution of the implicit part of the system is much more expensive than the explicit part (even though in the model problem (11), the solution of the implicit piece is a simple scalar division), the computational costs of SIPIDC methods are measured in terms of the numbers of implicit solves. To obtain the scaled stability diagrams, $\mathrm{Re}(\lambda)$ and $\mathrm{Im}(\lambda)$ are divided by the number of implicit function evaluations. These results show that even with computational costs taken into account, SIPIDC6[ARK3] still has the largest stability region.

We now present the second example to examine the effect on the stability of the overall SIPIDC method for a given type of predictor of differing orders. To this end, we compare the stability of SIPIDC6 methods implemented using first-order forward-backward Euler, IMEX BDF2, and IMEX BDF3 in the prediction step, and using forward-backward Euler methods in the correction steps. Figure 5 shows the scaled stability diagrams for the three SIPIDC6 methods, with the stability diagram for BDF6 included for comparison. (Note that the BDF6 method is applied to compute the solution at each substep of the SIPIDC methods as is done with the other BDF predictors, hence the stability region for BDF6 is scaled by a factor of 6 compared to Figure 1.) The unscaled stability diagrams for SIPIDC6[Euler], SIPIDC6[BDF2], SIPIDC6[BDF3], and BDF6 are qualitatively similar to those for the predictors (Figure 1); however, when computational costs are taken into account, the relative size of the stability diagrams change. The scaled stability region

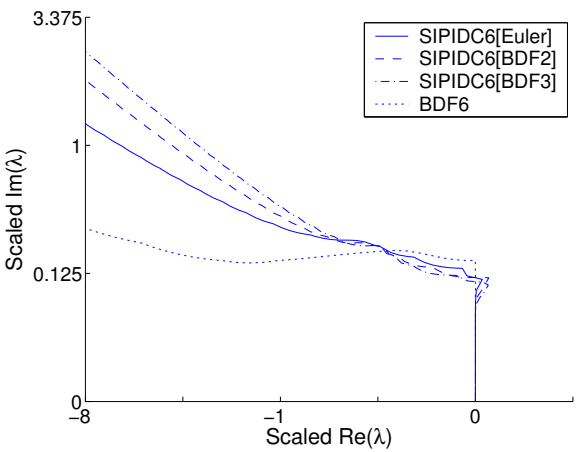


Figure 5. Scaled stability diagrams for SIPIDC6[Euler], SIPIDC6[BDF2], SIPIDC6[BDF3], and IMEX BDF6.

associated with the SIPIDC6[BDF3] is the largest, followed by SIPIDC6[BDF2], and by SIPIDC6[Euler]. Also noteworthy is that the stability regions of all three SIPIDC6 methods are substantially larger than that of the BDF6 method, even when the stability diagrams are scaled by the computational costs. This suggests that applying PIDC steps to a provisional solution computed by a BDF method generates an approximation with accuracy comparable to that computed by a high-order BDF method, without a decrease in the size of the stability region associated with the BDF6 scheme.

Finally, we consider the stability regions of SIPIDC schemes of varying order using the BDF3 scheme as a predictor. Figure 6 shows the scaled stability regions for SIPIDCk[BDF3] schemes for k ranging from 4 to 7, as well as that of the BDF3 method for comparison. Each method is $A(\alpha)$-stable with roughly the same α. Comparing Figure 6 with Figure 1 further demonstrates that higher-order SIPIDC methods do not suffer from a reduction in the size of the stability region as do the BDF methods. Note in particular that the stability region for SIPIDC7[BDF3] method is not significantly smaller than that of the moderate-order methods.

The accuracy and stability of SIPIDC methods using predictors of differing types and orders will be further assessed in Section 4 using more complex problems.

4. Numerical examples

In this section, numerical examples are used to further assess the stability and accuracy of SIPIDC methods. The first example is the *van der Pol's equation*, which is a popular nonlinear test problem for methods for stiff ODEs. The equation

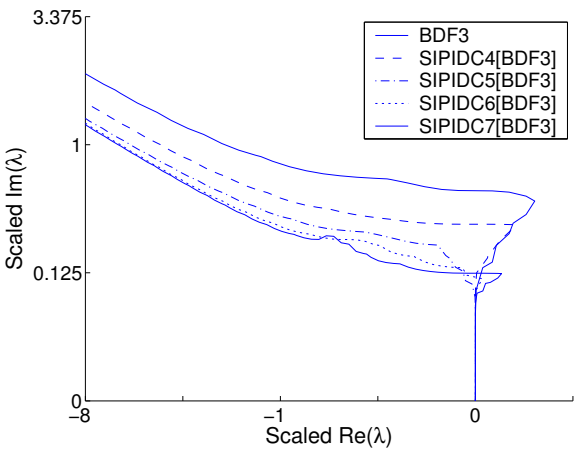


Figure 6. Scaled stability diagrams for IMEX BDF3 and SIPIDCk[BDF3] methods with $k = 4$ through 7.

prescribes the motion of a particle $x(t)$ governed by

$$x''(t) + \mu(1 - x(t)^2)x'(t) + x(t) = 0.$$

After applying the transformation $y_1(t) = x(t)$, $y_2(t) = \mu x'(t)$, and $t = t/\mu$, one obtains the system

$$y_1(t)' = y_2(t), \tag{14}$$

$$y_2(t)' = \tfrac{1}{\epsilon}\big(-y_1(t) + (1 - y_1(t)^2)y_2(t)\big), \tag{15}$$

where $\epsilon = 1/\mu^2$. As ϵ approaches zero, these equations become increasingly stiff. In the integration of (14) and (15), the first equation is treated explicitly, whereas the second equation is treated implicitly. Equations (14) and (15) are integrated for $t \in [0, 0.5]$ with the equilibrium initial conditions shown in Table 1. Because an exact solution is not known for this problem, errors are computed from a reference solution obtained using a 7th-order implicit PIDC[Euler] method and a very small time step, chosen so that the solutions computed with the PIDC method and the ARK4(3)6L[2]SA method in [Kennedy and Carpenter 2003] agree to 14 digits.

ϵ	$y_1(0)$	$y_2(0)$
10^{-3}	2	-0.66654321
10^{-4}	2	-0.666654321
10^{-5}	2	-0.6666654321
10^{-6}	2	-0.66666654321
10^{-7}	2	-0.666666654321

Table 1. Initial conditions for van der Pol's equation.

The second example is a linear system of four equations given by

$$y'(t) = Ay + By \tag{16}$$

where $B \in \Re^{4\times4}$ contains at least one eigenvalue with a large negative real part that scales as $1/\epsilon$, and $A \in \Re^{4\times4}$ has eigenvalues close to the origin. A and B are given

by

$$A = \begin{pmatrix} 0 & c_1 & 0 & 0 \\ -c_1 & 2a_1 & 0 & 0 \\ 0 & 0 & 0 & c_2 \\ 0 & 0 & -c_2 & 2a_2 \end{pmatrix}, \qquad D = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -10 & 0 & 0 \\ 0 & 0 & -10^2 & 0 \\ 0 & 0 & 0 & 1/\epsilon \end{pmatrix},$$

$$S = \begin{pmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & 0.5 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0.5 & 0 & 0 & 1 \end{pmatrix}, \qquad B = SDS^{-1}$$

where $a_1 = -0.2$, $b_1 = 5$, $a_2 = -0.4$, $b_2 = 12$, and $c_k = \sqrt{a_k + b_k}$ for $k = 1$ and 2. If ϵ is chosen carefully, then the sum $A + B$ contains one complex eigenvalue pair with small negative real part, and two negative real eigenvalues, one with magnitude of $\sim 1/\epsilon$. A and B do not commute, so the eigenvalues of $A + B$ do not correspond to the sum of eigenvalues of A and B. Equation (16) is integrated for $t \in [0.4, 2.4]$. The initial conditions are chosen to be the sum of the two normalized eigenvectors corresponding to the complex eigenvalues, so that transients are eliminated from the solution. We refer to this example as the *linear system test*.

The third example is the *cosine test*, which consists of the ODE

$$\begin{aligned} y(t)' &= -2\pi \sin(2\pi t) - \tfrac{1}{\epsilon}\big(y - \cos(2\pi t)\big), \\ y(0) &= 0, \end{aligned}$$

for $t \in [0, 10]$. The exact solution of is $y(t) = \cos(2\pi t)$, and as $\epsilon \to 0$, this equation becomes increasingly stiff. In this implementation, SIPIDC methods treat the term $-2\pi \sin(2\pi t)$ explicitly and the term $-(y - \cos(2\pi t))/\epsilon$ implicitly. A slightly more general problem was studied in [Prothero and Robinson 1974] and is considered here in Appendix A, since its simplicity allows an explicit examination of dominant error terms.

Because SIPIDC methods can be used to integrate ODEs arising from a method-of-lines discretization of PDEs, we include here a PDE example: the *Kuramoto–Silvashinsky (KS) equation*, which is used in [Akrivis and Smyrlis 2004] to study the accuracy of IMEX BDF methods. The inhomogeneous KS equation is given by

$$\begin{aligned} &u_t + uu_x + u_{xx} + \nu u_{xxxx} = f(x, t), \\ &u(x, 0) = g(x), \end{aligned} \tag{17}$$

for $x \in [0, 2\pi]$ and $t \in [0, T]$ and periodic boundary conditions $u(x + 2\pi, t) = u(x, t)$. As in [Akrivis and Smyrlis 2004], the functions $f(x, t)$ and $g(x)$ are constructed so that the exact solution is $u(x, t) = \sin(x + t)$; T and μ are taken to be 1 and 0.5, respectively. Equation (17) is first discretized in space using a pseudo-spectral

method as in [Akrivis and Smyrlis 2004], and then integrated in time using SIPIDC methods.

Note that the use of periodic boundary conditions for this problem avoids the issue of how to correctly impose boundary conditions for the provisional solutions in SIPIDC methods applied to PDEs with time-dependent boundary conditions. It is now well established that a naive imposition of the exact boundary conditions for PDEs within a RK method often results in a reduction of order of accuracy in the solution [Sanz-Serna et al. 1987; Carpenter et al. 1995], and a similar problem exists for PIDC methods. Strategies for addressing this problem have been proposed for RK methods for certain classes of problems (see [Abarbanel et al. 1996; Pathria 1997; Calvo and Palencia 2002; Alonso-Mallo 2002b; Alonso-Mallo 2002a; Portero et al. 2004]). Results in this direction for PIDC methods will be reported in future works.

All calculations reported below were performed using MATLAB programs. For brevity, we report results of only one or two examples for each study. Unless otherwise stated, qualitatively similar results were also obtained using other examples. For the ODE problems, the error reported is the discrete L_2 norm of the error in time of the computed solution $y(t_n)$ at each time step. For the KS equation, the error reported is the discrete L_2 norm of the error at the final time.

4.1. ***Efficiency improvement due to deferred corrections.*** We first assess the effect on the accuracy and stability of solutions computed by IMEX BDF and ARK methods after SIPIDC correction steps have been applied to those solutions. To this end, we compare the efficiency of IMEX BDF and ARK methods with SIPIDC methods that use these BDF and ARK methods in the prediction step. The SIPIDC methods use the first-order Euler method in the correction steps to improve the accuracy of the intermediate approximations. As noted previously, the solution of the implicit part of the ODEs is assumed to be much more expensive than the explicit part. For simplicity, we further assume that the implicit solves in all methods have similar computational costs. With these assumptions, we measure computational costs in terms of the numbers of implicit solves. Recall that starting values required for IMEX BDF and multistep methods are generated using a SIPIDC[Euler] method to advance the initial solution to t_1. The computational cost associated with this initial step is included in the total cost.

A comparison among SIPIDC5, SIPIDC7, BDF, and ARK methods using the van der Pol and cosine problems is shown in Figure 7. The comparison is obtained for the nonstiff case, with the stiffness parameter ϵ set to 10^{-1}. We first compare IMEX BDF methods with SIPIDC methods that use BDF as predictor. The left panels of Figure 7 show log-log plots of solution error versus the number of implicit function evaluations obtained using SIPIDC5[BDFk], SIPIDC7[BDFk], and BDFk methods,

for $k = 3$ and 4. The errors shown in Figure 7 for the van der Pol problem are for y_2; results for y_1 are similar. For a sufficiently high accuracy requirement, the method with the highest order, i.e., the SIPIDC7[BDFk] method, is the most efficient; and both SIPIDC5[BDFk] and SIPIDC7[BDFk] methods are more efficient than IMEX BDF3 and BDF4.

The comparisons between SIPIDC5[ARKk], SIPIDC7[ARKk], and ARKk, for $k = 3$ and 4, are similar. The results shown in the right-hand panels of Figure 7 indicate that, for a sufficiently high accuracy requirement, SIPIDC7[ARKk] is the most efficient, and that both SIPIDC methods are more efficient than the moderate-order ARK methods. Similar results (not shown) were also obtained for SIPIDC methods of order > 4, using a third- or fourth-order predictor and at least one correction step.

4.2. ***Comparison of predictors.*** In the next set of tests, we compare the efficiency of SIPIDC methods using different predictors. We first consider predictors of

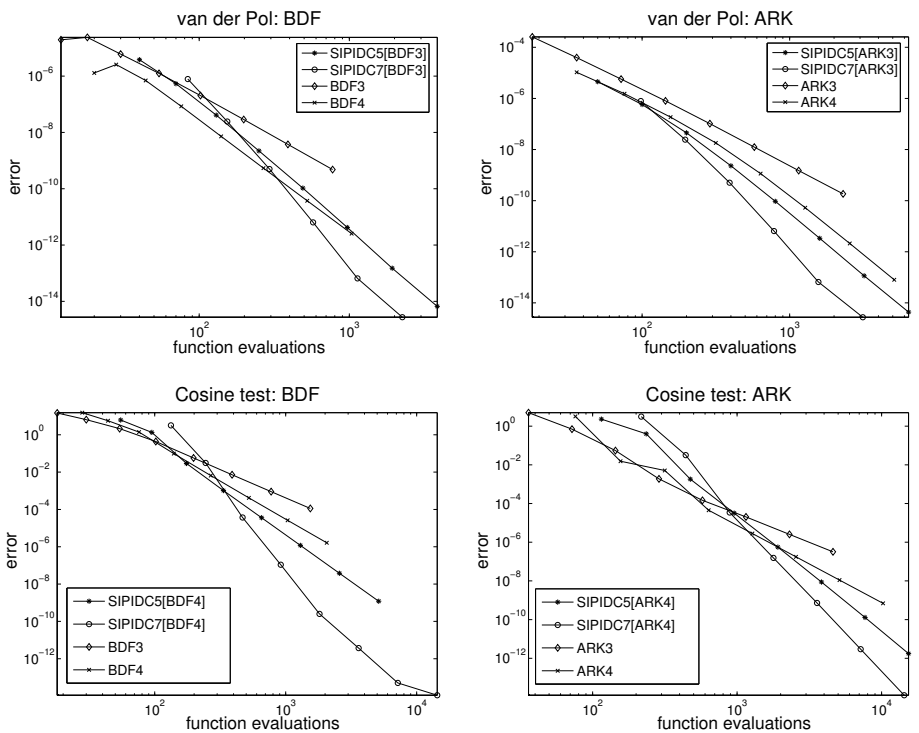


Figure 7. Efficiency comparison for SIPIDC5, SIPIDC7, IMEX BDF, and ARK methods using the van der Pol and cosine tests with $\epsilon = 10^{-1}$. Results show that correction steps improve efficiency of overall methods.

the same order but differing types (for example, BDF2 versus RK2). Results are obtained for the van der Pol problem with both nonstiff and stiff parameters. In the first set of experiments, we compare SIPIDC methods using second-order methods in the prediction step. Figure 8A1 and Figure 8A2 compare the efficiency of SIPIDC6 methods using three different second-order predictors—BDF2, RK2, and CNAB. Error curves obtained for IMEX BDF4 and ARK4 are also included for comparison. The stiffness parameters ϵ are 10^{-1} and 10^{-4} for results in Figure 8A1 and Figure 8A2, respectively. For the nonstiff problem (panel A1), the SIPIDC6[BDF2] and SIPIDC6[CNAB] methods, which have similar accuracy and the same computational costs, are the most efficient for sufficiently high accuracy requirement. (The two error curves approximately overlap.) These methods are more efficient than the SIPIDC6[RK2] method because of the lower computational costs in their prediction

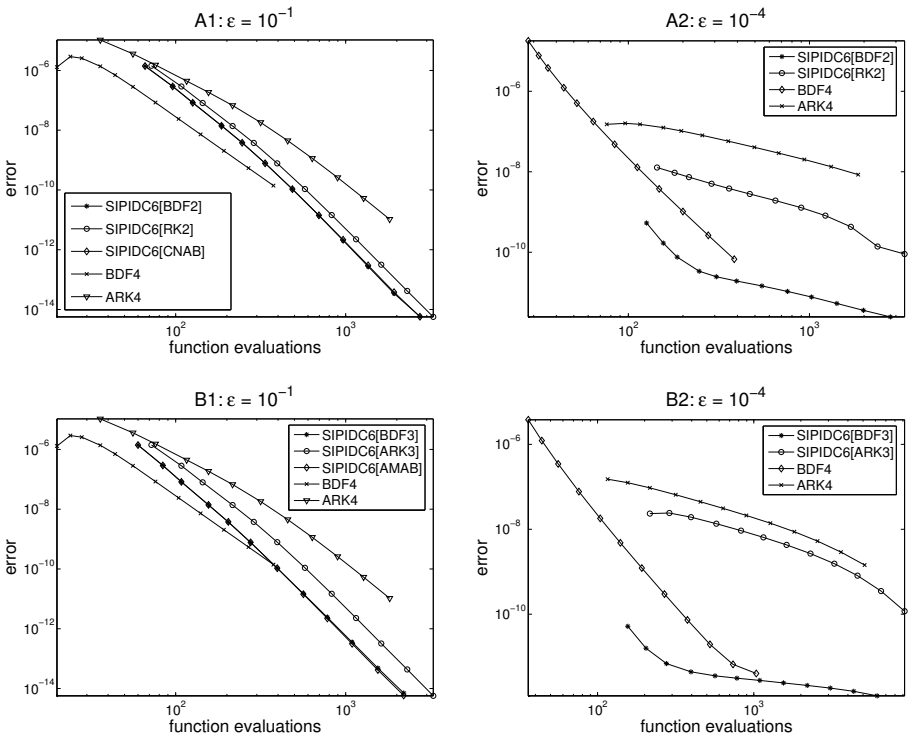


Figure 8. Efficiency comparison among SIPIDC6 methods using various predictors. Results for van der Pol problem. Errors in y_2 shown. A1, B1: $\epsilon = 10^{-1}$. A2, B2: $\epsilon = 10^{-4}$. A1, A2: SIPIDC6 methods with second-order predictors. B1, B2: SIPIDC6 methods with third-order predictors. Results for IMEX BDF4 and ARK4 are also included for comparison.

step. The IMEX BDF4 and ARK4 are less efficient, as expected, at sufficiently high accuracy requirement.

For the stiff case, the results shown in Figure 8A2 are markedly different. First, in this case SIPIDC6[CNAB] is unstable for sufficiently large Δt, owing to its lack of L-stability, and thus its error curve is not shown. Secondly, although approximations computed by BDF4 converge at fourth order, the SIPIDC6[RK2] and ARK4 methods appear to be converging at approximately a first-order rate in the range of Δt shown. Finally, the SIPIDC6[BDF2] method exhibits two regions of convergence: an approximately first-order convergence region at sufficiently small Δt and a higher-order region at larger Δt (although the latter region is too small for the order of convergence to be determined).

The results of the above tests are now presented using third-order predictors—IMEX BDF3, ARK3, and AMAB. The nonstiff results ($\epsilon = 10^{-1}$) are shown in

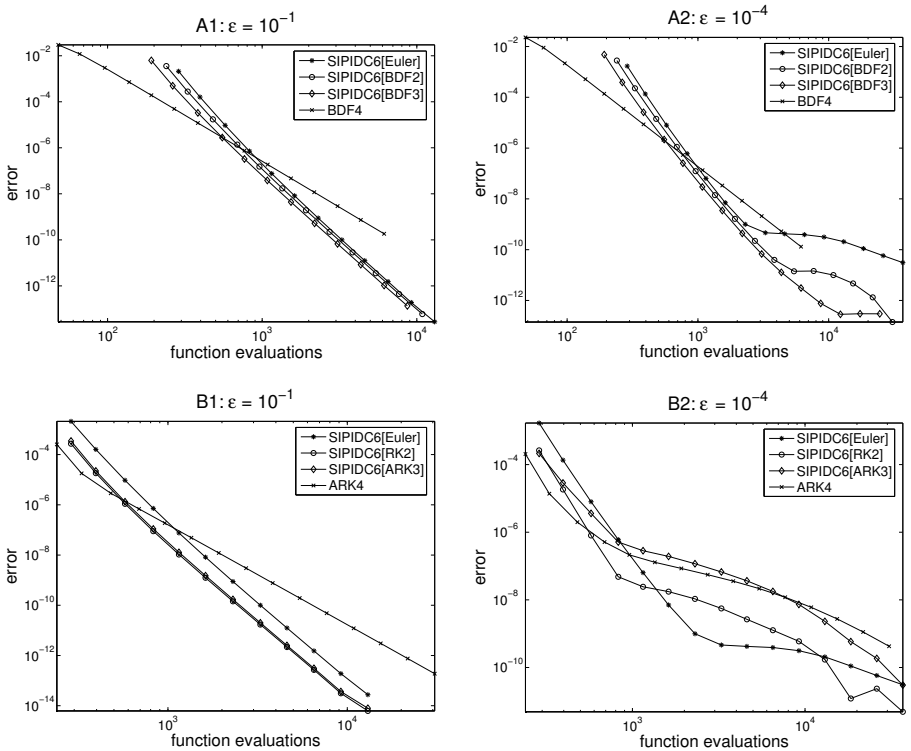


Figure 9. Efficiency comparison among SIPIDC6 methods using various predictors. Results for linear system problem. A1, B1: $\epsilon = 10^{-1}$. A2, B2: $\epsilon = 10^{-4}$. A1, A2: SIPIDC6 methods with Euler, BDF2, and BDF3 predictors, and BDF4. B1, B2: SIPIDC6 methods with Euler, RK2, and ARK3 predictors, and ARK4.

Figure 8B1 and are very similar to those for the second-order predictors in Figure 8B2. The SIPIDC6[BDF3] and SIPIDC6[AMAB] methods are more efficient than SIPIDC6[ARK3] as well as the BDF4 and ARK4 methods for a sufficiently high accuracy requirement.

For the stiff problem ($\epsilon = 10^{-4}$) the observed results shown in panel B2 are again different from the nonstiff results, although they are similar to the results for second-order methods in the stiff case shown in panel A2. The SIPIDC6[AMAB] method, with a predictor that is not L-stable, is unstable like the SIPIDC6[CNAB] above and is not shown. Both ARK4 and SIPIDC6[ARK3] appear to be converging at approximately a first-order rate in the range of Δt shown, while the BDF4 method converges at the proper order. Also, the solutions computed by SIPIDC6[BDF3], show two different convergence regimes, although the limits of machine precision make it difficult to determine the respective rates. The order reduction behavior of SIPIDC methods with BDF and RK predictors will be further investigated in Section 4.3.

We will now compare predictors of the same type but differing orders. For a SIPIDCK method that is based on a first-order method, K implicit solves are required (one for each of the K substeps) for the provisional step and for each of the $K-1$ correction steps. Thus, a total of K^2 implicit solves are required. On the other hand, an SIPIDCK method that uses a first-order corrector but a pth-order predictor requiring s implicit solves per substep will require $K-p$ correction iterations and thus a total of $(K-p+s)K$ implicit solves per time step. Hence, assuming the implicit solves require similar computational costs for all methods, the resulting SIPIDC methods have the same order but a smaller computational cost if $p > s$ (e.g BDF methods where $s = 1$). However, regardless of whether $p > s$, it is not clear that increasing the order of the predictor results in a more efficient method in terms of error per function evaluation.

The linear system test is used to assess the extent to which the efficiency of a SIPIDC method is improved by using IMEX BDF and RK methods in the prediction step, first for a nonstiff problem with $\epsilon = 1$. For BDF methods, Figure 9A1 compares the efficiency of SIPIDC6 methods using BDF predictors of order one (Euler) through three. The error curve for BDF4 is included for comparison. In this case SIPIDC6[BDF3] requires the fewest correction steps and it is indeed the most efficient, albeit by a slight amount. Next we compare the efficiency of SIPIDC6 methods using IMEX RK-type predictors of differing orders (see Figure 9B1). For this comparison, the three SIPIDC6 methods have similar computational costs, but SIPIDC6[ARK3] and SIPIDC[RK2] appear more efficient than SIPIDC6[Euler]. Similar results were also obtained for SIPIDC methods of other overall orders.

The comparison is repeated for a stiff problem ($\epsilon = 10^{-4}$) in Figures 9A2 and 9B2. Results in Figure 9A2 show the advantage of using a moderate-order BDF method

in the provisional step. The error curve for SIPIDC6[Euler] shows three regions of convergence: for sufficiently large Δt (fewer than 2×10^3 function evaluations) and for sufficiently small Δt (more than 10^4 function evaluations, where convergence begins to increase), convergence is approximately sixth order; however, order reduction is observed for middle range Δt, where the curve is flat (i.e., zeroth-order convergence). Unlike SIPIDC6[Euler], the error curve corresponding to the order reduction region for SIPIDC6[BDF2] is less flat, although the region is too small for a reasonable estimate of the order of accuracy. Order reduction is not observed for BDF4, which is consistent with the analysis for fully implicit BDF methods (see [Hairer and Wanner 1991] Chapter V). Finally, the behavior of the SIPIDC6[BDF3] is difficult to determine due to machine precision. The extent of order reduction of SIPIDC methods using BDF predictors of differing orders is further investigated in Section 4.3

The stiff test is repeated for SIPIDC6[Euler], SIPIDC6[RK2], SIPIDC6[ARK3], and ARK4 and the results are shown in Figure 9B2. Three regions of convergence were obtained for each of these methods. As noted previously, the order reduction region for SIPIDC6[Euler] error curve is flat. In contrast, the order reduction region for the error curves associated with SIPIDC6[RK2], SIPIDC6[ARK3], and ARK4 appears to be first order. For sufficiently small Δt, the methods will again exhibit full order accuracy (in the absence of precision errors).

4.3. ***Order reduction.*** Numerical results in [Layton and Minion 2005] show that the characteristics of order reduction of SIPIDC methods depends critically on the choice of quadrature nodes: when uniform nodes are used and when the left endpoint is not used in the quadrature rule associated with the implicit piece (recall that such quadrature nodes are referred to as "LR" [Layton and Minion 2005]), an order reduction to $\mathbb{O}(\epsilon^2)$ is observed, compared to $\mathbb{O}(\epsilon\Delta t)$ for SIPIDC methods using nonuniform nodes (for example, Gauss quadrature nodes) or those including the left endpoint in the quadrature rules. SIPIDC methods studied in [Layton and Minion 2005] use Euler in both the provisional and correction steps. Below we examine convergence behavior and order reduction for stiff problems of SIPIDC methods using moderate-order methods in the provisional step, using first the van der Pol's problem and then the cosine problem.

To investigate the dependence of order reduction on the choice of predictor, we computed solutions for the van der Pol equation for increasing stiffness (for $\epsilon = 10^{-k}$, $k = 1, 3, 4, 5, 6$) by means of SIPIDC5 methods using different predictors (Euler, BDF2, BDF3, and ARK3). Log-log plots of errors for y_2 versus implicit function evaluations are shown in Figure 10. For sufficiently stiff parameters ($\epsilon < 10^{-3}$), the convergence rate drops to the zeroth order for SIPIDC5[Euler], to the first order for SIPIDC5[BDF2] and SIPIDC5[ARK3], and to the second

order for SIPIDC5[BDF3] in the order-reduction regime. We see that for the methods SIPIDC5[Euler], SIPIDC5[BDF2], and SIPIDC5[BDF3], the magnitude of the error in the regions of reduced convergence scales approximate as ϵ^2, and for SIPIDC5[ARK3] it scales approximately as ϵ. It is noteworthy that the order reduction results for SIPIDC5[ARK3] are similar to the SIPIDC[Euler] method using nonuniform points or using the left endpoint in the quadrature rules [Layton and Minion 2005]. Similar results are also shown for the cosine problem for increasingly stiff values of ϵ in Figure 11, shown as log-log plots of errors versus time step size Δt.

The above results for SIPIDC5[Euler] are consistent with those reported in [Layton and Minion 2005] for SIPIDC6[Euler] and SIPIDC7[Euler] using LR uniform nodes. The error formula derived in [Layton and Minion 2005] shows that the dominant error term for these methods, after one correction step, is $\mathbb{O}(\epsilon^2)$; thus, the region of reduced convergence is flat with magnitude that scales as ϵ^2.

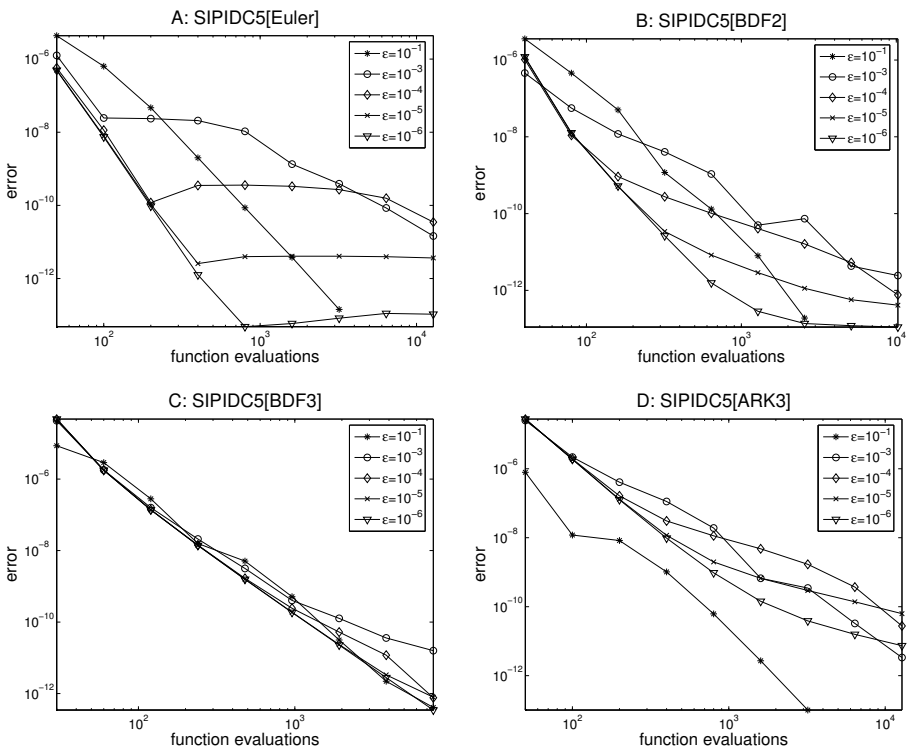


Figure 10. Error curves obtained for the van der Pol problem with a range of ϵ values, computed using the SIPIDC5 methods with differing predictors. The region of order reduction shows zeroth-order convergence in A, first-order in B and D, second-order in C.

Analogous error formulae are derived in the Appendix for SIPIDC[BDF2] and SIPIDC[BDF3]; these error formulae show that for SIPIDC[BDF2], the dominant error term is $\mathbb{O}(\epsilon^2\Delta t)$, and for SIPIDC[BDF3], it is $\mathbb{O}(\epsilon^2\Delta t^2)$, thereby explaining the shape of the error curves shown in Figure 10 and Figure 11.

4.4. ***A ladder approach.*** The approximations computed by the provisional step and by the initial correction steps have lower orders of accuracy than the final solution. A "ladder" approach makes use of this fact to reduce the computational cost of a SIPIDC method without compromising the overall order of the solution. This is achieved by allowing larger temporal or spatial errors in the initial PIDC iterations. One such ladder approach was implemented in [Minion 2003]. To obtain a Kth-order solution, the quadrature $\mathcal{Q}$ in (5) must be approximated to Kth order. When LR uniformly-spaced nodes are used, $K+1$ nodes or K substeps are required. In [Minion 2003], based on the observation that the kth correction equation

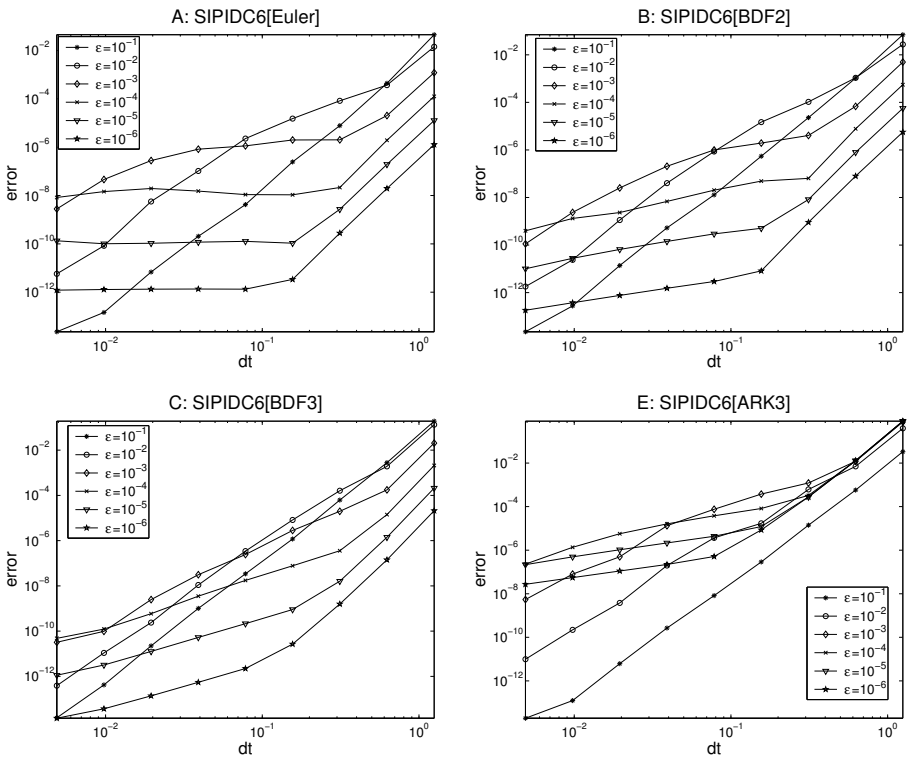


Figure 11. Error curves obtained for the cosine problem with a range of ϵ values, computed using the SIPIDC methods with differing predictors. The region of order reduction shows $\mathbb{O}(\epsilon^2)$ in A, $\mathbb{O}(\epsilon^2\Delta t)$ in B, $\mathbb{O}(\epsilon^2\Delta t)$ C, and $\mathbb{O}(\epsilon\Delta t)$ in D.

computes a globally $\mathbb{O}(\Delta t^{k+1})$ approximation (Euler is used in the predictor in [Minion 2003]), the number of substeps used to compute the solution during the initial PIDC iterations was reduced, i.e., fewer substeps were used when k is small. Although no significant improvement in efficiency was noted in [Minion 2003] when this approach was applied to a linear problem, the nonlinear KS equation (17) is used here to re-examine the effects of ladder approach, with a new focus on SIPIDC methods using BDF2 and BDF3 as predictors.

We compare the efficiency among SIPIDC6 methods using Euler, BDF2, and BDF3 methods in the provisional step, and with or without incorporating the ladder approach. Results are shown in Figure 12 . Consistent with results described previously, SIPIDC6 methods using moderate-order predictors are more efficient. Also, although the ladder approach reduces computational cost, it also increases error. These two competing effects result in a negligible improvement in efficiency. Qualitatively similar results were also obtained for SIPIDC methods of other orders.

5. Discussion

We have presented alternative implementations of SIPIDC methods for the temporal integration of ODEs with both nonstiff and stiff components corresponding to eigenvalues with large negative real part. In these implementations, various types of second- through fourth-order integration methods are used in the prediction step. The stability and efficiency of these SIPIDC methods are assessed and compared to traditional IMEX methods. High-order SIPIDC methods are proposed as alternatives

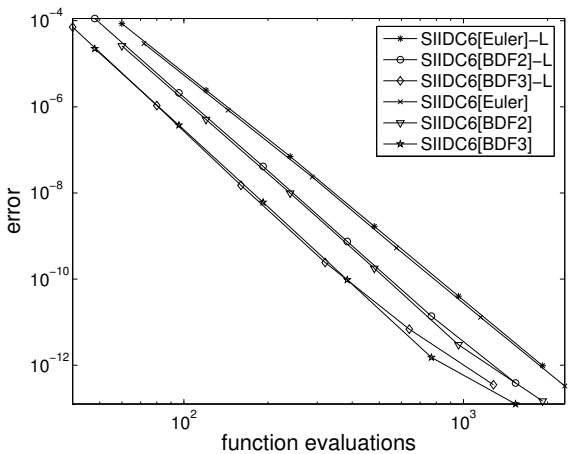


Figure 12. Error curves obtained for the KS equation using the SIPIDC6 methods. '-L' denotes methods using ladder approach.

to IMEX BDF or IMEX RK methods, which exhibit instability at high orders. In contrast, our stability analysis shows attractive stability properties for high-order SIPIDC methods using a moderate-order IMEX BDF or IMEX RK method in the prediction step.

Another goal of this study is to determine whether SIPIDC methods that use moderate-order predictors are more efficient. Numerical results suggest that using moderate-order IMEX BDF methods in the prediction step gives rise to SIPIDC methods that are more efficient and also stable for stiff problems. In contrast, moderate-order predictors based on IMEX RK do not significantly improve efficiency because of the multiple implicit solves required at the stages, and predictors based on multistep methods such as AMAB result in overall methods that are unstable when applied to stiff problems.

Although we only consider SIPIDC methods that use the forward-backward Euler method to solve the correction equations, moderate-order integration methods can no doubt be used in the correction steps. For example, the correction Equation (2) may be discretized by means of a second-order method, for example, CNAB, IMEX RK2, or IMEX BDF2. Such methods require fewer iterations of the correction equation to achieve the same overall order of accuracy relative to methods based on first-order methods, but each iteration of the correction equation may be more expensive. The behavior of various moderate-order correctors is the focus of an on-going project.

IMEX BDF predictors also change the extent and characteristics of order reduction of the SIPIDC methods when applied to stiff problems. The convergence rate in the region of order reduction is $k-1$ for a kth-order BDF predictor, with errors of $\mathbb{O}(\epsilon^2)$ magnitude, where ϵ is the stiffness parameter such that as $\epsilon \to 0$ the problem becomes increasingly stiff. In contrast, IMEX RK predictors give rise to first-order convergence in the region of order reduction, with $\mathbb{O}(\epsilon)$ errors. Thus, for stiff problems SIPIDC methods with IMEX BDF predictors likely generate solutions with higher accuracy than SIPIDC methods using non-BDF predictors.

A uniform time-step has been assumed throughout this work. However, SIPIDC methods are suitable candidates for adaptive time-marching: the correction term can be used to dynamically determine the appropriate time step size to meet certain accuracy requirements. However, when a BDF or multistep predictor is used, where solution values at k previous substeps are needed to advance the solution, care must be taken in computing the provisional solution at the first $k-1$ substeps, because the substep size may not be equal in $[t_{n-1}, t_n]$ and $[t_n, t_{n+1}]$. In this case, variable-step form of the methods can be used for the first $k-1$ iterations, where the coefficients in (7)–(9) and in (10) depend on the relative substep sizes.

The ultimate target applications for PIDC methods are PDEs with multiple stiff terms, such as the advection-diffusion-reaction equations. Indeed, in earlier studies

[Bourlioux et al. 2003; Layton and Minion 2004], we have proposed the multi-implicit PIDC (MIPIDC) methods (formerly MISDC methods) which decouple the stiff processes and integrate them separately, possibly using differing time steps. The MIPIDC methods developed so far are based on the forward/backward Euler methods and a first-order splitting. A project that develops and analyzes the performance of MIPIDC methods based on moderate-order IMEX BDF methods and on a moderate-order splitting is underway. It should be noted that no analysis of semi- and multi-implicit PIDC methods applied to PDEs with stiffness characterized by rapidly oscillatory modes (that is, corresponding to eigenvalues with large imaginary parts) has yet been attempted.

A ladder approach, which uses fewer substeps in the initial PIDC iterations, fails to significantly improve the efficiency of SIPIDC methods. Because of the extra effort involved in its implementation, the value of this ladder approach is not obvious. Alternatively, when integrating a PDE, one may use a less refined spatial grid during the initial PIDC iterations. This spatial ladder approach is likely to be particularly effective in higher spatial dimensions and warrants attention.

Appendix

In this Appendix we develop an analytical formulation for the truncation error for SIPIDC methods applied to the simple stiff equation analyzed in [Prothero and Robinson 1974].

Given a smooth function $p(t)$, consider the ODE with exact solution $y(t) = p(t)$ given by

$$\begin{aligned} y' &= p'(t) - \frac{1}{\epsilon}\big(y - p(t)\big), \\ y(0) &= p(0). \end{aligned} \tag{A.1}$$

Here ϵ is the stiffness parameter where the equation becomes more stiff as $\epsilon \to 0$. We integrate (A.1) by treating the first term explicitly and the second term implicitly. The following analysis applies to the stiff case where $\epsilon \ll \Delta t$.

We first consider a provisional solution computed using the BDF2 method given by (7). Let $p_m \equiv p(t_m)$ and $y_m^k \equiv y^k(t_m)$. Given a previously computed value y_m^0 with error $e_m^0 = y_m^0 - p_m$, one step of BDF2 applied to Equation (A.1) yields

$$y_{m+1}^0 = \frac{2y_m^0 - \frac{1}{2}y_{m-1}^0 + \Delta t_m(2p'_m - p'_{m-1} + \frac{1}{\epsilon}p_{m+1})}{\frac{3}{2} + \frac{\Delta t_m}{\epsilon}}. \tag{A.2}$$

When $\epsilon < \Delta t_m$, the quantity $1/(\frac{3}{2} + \frac{\Delta t_m}{\epsilon})$ can be expanded into the series

$$\frac{1}{\frac{3}{2} + \frac{\Delta t_m}{\epsilon}} = \frac{\epsilon}{\Delta t_m}\left(1 - \frac{3}{2}\frac{\epsilon}{\Delta t_m} + \left(\frac{3}{2}\frac{\epsilon}{\Delta t_m}\right)^2 - \dots\right). \tag{A.3}$$

Substituting (A.3) into (A.2) yields

$$y^0_{m+1} = p_{m+1} + \left(2p'_m - p'_{m-1} + \frac{2y^0_m - \frac{1}{2}y^0_{m-1} - \frac{3}{2}p_{m+1}}{\Delta t_m}\right) \times \left(\epsilon - \frac{3}{2}\frac{\epsilon^2}{\Delta t_m} + \left(\frac{3}{2}\frac{\epsilon}{\Delta t_m}\right)^2\epsilon - \dots\right) \quad \text{(A.4)}$$

To simplify (A.4), we make use of the following relations derived using Taylor's expansion

$$\begin{aligned} 2p'_m - p'_{m-1} &= p'_{m+1} - \Delta t_m^2 p^{(3)}_{m+1} + \Delta t_m^3 p^{(4)}_{m+1} + \mathbb{O}(\Delta t_m^4), \\ -\frac{3}{2\Delta t_m}p_{m+1} &= -p'_{m+1} - \frac{1}{\Delta t_m}\left(2p_m - \tfrac{1}{2}p_{m-1}\right) + \frac{\Delta t_m^2}{3}p^{(3)}_{m+1} \\ &\quad + \tfrac{3}{4}\Delta t_m^3 p^{(4)}_{m+1} + \mathbb{O}(\Delta t_m^4). \end{aligned} \quad \text{(A.5)}$$

From (A.5), one obtains

$$2p'_m - p'_{m-1} + \frac{2y_m - \frac{1}{2}y_{m-1} - \frac{3}{2}p_{m+1}}{\Delta t_m} = \frac{2e_m - \frac{1}{2}e_{m-1}}{\Delta t_m} - \tfrac{2}{3}\Delta t_m^2 p^{(3)}_{m+1} + \tfrac{3}{4}\Delta t_m^2 p^{(4)}_{m+1} \quad \text{(A.6)}$$

Thus,

$$y_{m+1} = p_{m+1} + \left(\frac{2e_m - \frac{1}{2}e_{m-1}}{\Delta t_m} - \tfrac{2}{3}\Delta t_m^2 p^{(3)}_{m+1} + \tfrac{3}{4}\Delta t_m^3 p^{(4)}_{m+1}\right) \times \left(\epsilon - \tfrac{3}{2}\frac{\epsilon^2}{\Delta t_m} + \left(\tfrac{3}{2}\frac{\epsilon}{\Delta t_m}\right)^2\epsilon - \dots\right). \quad \text{(A.7)}$$

Substituting (A.6) into (A.4) and making use of the definition of $e^0_m \equiv p_m - y^0_m$,

$$\begin{aligned} e^0_{m+1} &= \frac{2e^0_m - \frac{1}{2}e^0_{m-1}}{\Delta t_m}\left(\epsilon - \frac{3}{2}\frac{\epsilon^2}{\Delta t_m} + \left(\frac{3}{2}\frac{\epsilon}{\Delta t_m}\right)\epsilon\right) \\ &\quad -\tfrac{2}{3}\Delta t_m^2 p^{(3)}_{m+1}\left(\epsilon - \frac{3}{2}\frac{\epsilon^2}{\Delta t_m} + \left(\frac{3}{2}\frac{\epsilon}{\Delta t_m}\right)\epsilon\right) + \mathbb{O}(\epsilon\Delta t^3) + \mathbb{O}(\epsilon^2\Delta t^2) + \mathbb{O}(\epsilon^3). \end{aligned} \quad \text{(A.8)}$$

Now consider the correction equation given a provisional solution y^0_m. Note that $f(y^0_m, t_m) = p'_m - \frac{1}{\epsilon}e^0_m$. The direct form of the correction equation using

forward-backward Euler for (A.1) is

$$y^1_{m+1} = y^1_m + \Delta t_m\Big(p'_m - p'_m - \frac{1}{\epsilon}(y^1_{m+1} - p_{m+1}) + \frac{1}{\epsilon}(y^0_{m+1} + p_{m+1})\Big) + \mathcal{Q}_m^{m+1}(y^0)$$
$$= y^1_m + \Delta t_m\Big(-\frac{1}{\epsilon}(y^1_{m+1} - y^0_{m+1})\Big) + \mathcal{Q}_m^{m+1}(y^0).$$

Solving for y^1_{m+1} yields

$$y^1_{m+1} = \frac{y^1_m + \frac{\Delta t_m}{\epsilon} y^0_{m+1} + \mathcal{Q}_m^{m+1}\left(p'(t) - \frac{1}{\epsilon}e^0(t)\right)}{1 + \frac{\Delta t_m}{\epsilon}}. \tag{A.9}$$

To derive an error formula for y^1_{m+1}, we first consider the last quadrature term in the numerator. The integration rule given by Equation (4) defines

$$\mathcal{Q}_m^{m+1}\left(p'(t) - \tfrac{1}{\epsilon}e^0(t)\right) = \Delta t_m \sum_{l=0}^{p} q_m^l \left(p'_l - \tfrac{1}{\epsilon}e^0_l\right).$$

Since the integration rule is assumed to be $O(\Delta t^q)$, the first term can be integrated to give

$$\mathcal{Q}_m^{m+1}\left(p'(t) - \tfrac{1}{\epsilon}\tilde{e}(t)\right) = p_{m+1} - p_m + \mathbb{O}(\Delta t^q) - \frac{\Delta t_m}{\epsilon}\sum_{l=0}^{p} q_m^l e^0_l.$$

Substituting this expression into Equation (A.9) gives

$$y_{m+1} = \frac{y^1_m + p_{m+1} - p_m + \frac{\Delta t_m}{\epsilon}\left(y^0_{m+1} - \sum_{l=0}^{p} q_m^l e^0_l\right) + O(\Delta t^q)}{1 + \frac{\Delta t_m}{\epsilon}}.$$

Applying the expansion (A.3), one obtains

$$y^1_{m+1} = \frac{\epsilon}{\Delta t_m}\left(y^1_m + p_{m+1} - p_m + \frac{\Delta t_m}{\epsilon}\left(y^0_{m+1} - \sum_{l=0}^{p} q_m^l e^0_l\right) + O(\Delta t^q)\right)$$
$$-\left(\frac{\epsilon}{\Delta t_m}\right)^2\left(y^1_m + p_{m+1} - p_m + \frac{\Delta t_m}{\epsilon}\left(y^0_{m+1} - \sum_{l=0}^{p} q_m^l e^0_l\right) + O(\Delta t^q)\right)$$
$$+\left(\frac{\epsilon}{\Delta t_m}\right)^3\left(y^1_m + p_{m+1} - p_m + \frac{\Delta t_m}{\epsilon}\left(y^0_{m+1} - \sum_{l=0}^{p} q_m^l e^0_l\right) + O(\Delta t^q)\right)\ldots,$$

hence

$$y^1_{m+1} = y^0_{m+1}\left(1 - \frac{\epsilon}{\Delta t_m} + \left(\frac{\epsilon}{\Delta t_m}\right)^2 \dots\right) + \left(y^1_m - p_m + p_{m+1}\right)\left(\frac{\epsilon}{\Delta t_m} - \left(\frac{\epsilon}{\Delta t_m}\right)^2 + \left(\frac{\epsilon}{\Delta t_m}\right)^3 \dots\right) - \left(\sum_{l=0}^{p} q^l_m e^0_l\right)\left(1 - \frac{\epsilon}{\Delta t_m} + \left(\frac{\epsilon}{\Delta t_m}\right)^2 \dots\right) + O(\epsilon \Delta t^{q-1}) + O(\epsilon^2 \Delta t^{q-2}) + O(\epsilon^3 \Delta t^{q-3}) \dots .$$

Finally, define the error in the updated solution $e^1_m = y^1_m - p_m$. Then subtracting p_{m+1} from both sides of the equation and manipulating yields

$$e^1_{m+1} = e^1_m\left(\frac{\epsilon}{\Delta t_m} - \left(\frac{\epsilon}{\Delta t_m}\right)^2 + \left(\frac{\epsilon}{\Delta t_m}\right)^3 \dots\right) + \left(e^0_{m+1} - \sum_{l=0}^{p} q^l_m e^0_l\right)\left(1 - \frac{\epsilon}{\Delta t_m} + \left(\frac{\epsilon}{\Delta t_m}\right)^2 \dots\right) + O(\epsilon \Delta t^{q-1}) + O(\epsilon^2 \Delta t^{q-2}) + O(\epsilon^3 \Delta t^{q-3}) + \dots . \quad \text{(A.10)}$$

Consider now the first time step of a SIPIDC method for Equation (A.1). Assume that the error at the beginning of the time step is given by e^0_0. The dominant error terms in the provisional solution Equation (A.8) are

$$e^0_{m+1} = -\frac{2}{3}\Delta t^2_m p^{(3)}_{m+1}\left(\epsilon - \frac{2}{3}\frac{\epsilon^2}{\Delta t_m}\right) + z_m, \quad \text{(A.11)}$$

where

$$z_m = \begin{cases} \frac{\epsilon}{\Delta t_m}\left(2e^0_0 - \frac{1}{2}e^0_{-1}\right), & m = 1, \\ -\frac{4}{3}p^{(3)}_1 \Delta t_m \epsilon^2 - \frac{\epsilon}{2\Delta t_m} e^0_0, & m = 2, \\ \left(\frac{4}{3}p^{(3)}_m - \frac{1}{3}p^{(3)}_{m-1}\right)\Delta t_m \epsilon^2, & m > 2. \end{cases}$$

In deriving the above expression, we made use of the assumption of uniform substep, i.e., $\dots = t_{m-2} = t_{m-1} = t_m = \dots$. Likewise, the dominant pieces of the correction equation error (A.10) comes from the term

$$e^1_{m+1} = e^0_{m+1} - \sum_{l=0}^{p} q^l_m e^0_l + e^1_m \frac{\epsilon}{\Delta t_m}. \quad \text{(A.12)}$$

Substituting the dominant provisional error (A.11) into the dominant correction error (A.12) gives

$$e^1_{m+1} = -\frac{2}{3}\Delta t_m^2 p^{(3)}_{m+1}\left(\epsilon - \frac{2}{3}\frac{\epsilon^2}{\Delta t_m}\right) + z_m - \sum_{l=1}^{p} q_m^l \left(-\frac{2}{3}\Delta t_m^2 p_l^{(3)}\left(\epsilon - \frac{2}{3}\frac{\epsilon^2}{\Delta t_m}\right) + z_l\right). \quad \text{(A.13)}$$

The summation term can be rewritten via a Taylor series expansion

$$\sum_{l=1}^{p} q_m^l \left(-\frac{2}{3}\Delta t_m^2 p_l^{(3)}\left(\epsilon - \frac{2}{3}\frac{\epsilon^2}{\Delta t_m}\right) + z_l\right) = -\frac{2}{3}\Delta t_m^2 p^{(3)}_{m+1}\left(\epsilon - \frac{2}{3}\frac{\epsilon^2}{\Delta t_m}\right) + \mathbb{O}(\epsilon \Delta t_m^3 + \epsilon^2 \Delta t_m^2) + \sum_{l=1}^{p} q_m^l z_l. \quad \text{(A.14)}$$

Substituting (A.14) into (A.13) and simplifying gives

$$e^1_{m+1} = z_m - \sum_{l=1}^{p} q_m^l z_l + \mathbb{O}(\Delta t^q + \epsilon \Delta t^3 + \epsilon^2 \Delta t^2) \quad \text{(A.15)}$$

owing to the mismatch between z_m for $m = 1$ and 2 and for $m > 2$, $z_m - \sum q_m^l z_l = \mathbb{O}(\epsilon^2 \Delta t_m)$. Thus, $e^1_{m+1} = \mathbb{O}(\Delta t^q + \epsilon^2 \Delta t)$.

Following similar procedures, an error formula for the correction step of a SIPIDC method using uniform quadrature nodes and BDF3 in the predictor step can be shown to be $\mathbb{O}(\Delta t^p + \epsilon^2 \Delta t^2)$.

References

[Abarbanel et al. 1996] S. Abarbanel, D. Gottlieb, and M. H. Carpenter, "On the removal of boundary errors caused by Runge–Kutta integration of nonlinear partial differential equations", *SIAM J. Sci. Comput.* **17** (1996), 777–782.

[Akrivis and Smyrlis 2004] G. Akrivis and Y.-S. Smyrlis, "Implicit-explicit BDF methods for the Kuramoto–Silvashinsky equation", *Appl. Numer. Math.* **51** (2004), 151–169.

[Akrivis et al. 1998] G. Akrivis, M. Crouzeix, and C. Makridakis, "Implicit-explicit multistep finite element methods for nonlinear parabolic equations", *Math. Comp.* **67** (1998), 457–477.

[Akrivis et al. 1999] G. Akrivis, M. Crouzeix, and C. Makridakis, "Implicit-explicit multistep methods for quasilinear parabolic equations", *Numer. Math.* **82** (1999), 521–541.

[Alonso-Mallo 2002a] B. Alonso-Mallo, I. and Cano, "Spectral/Rosenbrock discretizations without order reduction for linear parabolic problems", *Appl. Numer. Math.* **41** (2002), 247–268.

[Alonso-Mallo 2002b] I. Alonso-Mallo, "Runge–Kutta methods without order reduction for intial boundary value problems", *Numer. Math.* **91** (2002), 577–603.

[Ascher et al. 1995] 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**:3 (1995), 797–823. MR 96j:65076

[Ascher et al. 1997] 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), 151–167.

[Berger and Oliger 1984] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations", *J. Comput. Phys.* **53**:3 (1984), 484–512. MR 85h:65211

[Bourlioux et al. 2003] A. Bourlioux, A. T. Layton, and M. L. Minion, "Higher-order multi-implicit spectral deferred correction methods for problems of reacting flow", *J. Comput. Phys.* 189 (2003), 351–376.

[Calvo and Palencia 2002] M. P. Calvo and C. Palencia, "Avoiding the order reduction of Runge–Kutta methods for linear initial boundary value problems", *Math. Comp.* **71**:240 (2002), 1529–1543. MR 2003h:65091

[Calvo et al. 2001] M. P. Calvo, J. de Frutos, and J. Novo, "Linearly implicit Runge-Kutta methods for advection-reaction-diffusion equations", *Appl. Numer. Math.* **37**:4 (2001), 535–549. MR 2002e:65114

[Carpenter et al. 1995] M. H. Carpenter, D. Gottlieb, S. Abarbanel, and W. S. Don, "The theoretical accuracy of Runge–Kutta time discretizations for the initial-boundary value problem: a study of the boundary error", *SIAM J. Sci. Comput.* **16**:6 (1995), 1241–1252. MR 96h:65088

[Dutt et al. 2000] A. Dutt, L. Greengard, and V. Rokhlin, "Spectral deferred correction methods for ordinary differential equations", *BIT* **40**:2 (2000), 241–266. MR 2001e:65104

[Ehle 1969] B. L. Ehle, *On Padé approximations to the exponential function and A-stable methods for the numerical solution of initial value problems*, Dept. Applied Analysis and Computer Sci., Research Rpt. CSRR 2010, University of Waterloo, 1969.

[Frank et al. 1997] J. Frank, W. Hundsdorfer, and J. G. Verwer, "On the stability of implicit-explicit linear multistep methods", *Appl. Numer. Math.* **25**:2-3 (1997), 193–205. Special issue on time integration (Amsterdam, 1996). MR 98m:65126

[Gear 1971] C. W. Gear, *Numerical initial value problems in ordinary differential equations*, Prentice-Hall, Englewood Cliffs, N.J., 1971. MR 47 #4447

[Hairer and Wanner 1991] E. Hairer and G. Wanner, *Solving ordinary differential equations. II*, Springer Series in Computational Mathematics **14**, Springer, Berlin, 1991. Stiff and differential-algebraic problems. MR 92a:65016

[in't Hout 2002] K. J. in't Hout, "On the contractivity of implicit-explcit linear multistep methods", *Appl. Numer. Math.* **42** (2002), 201–212.

[Kennedy and Carpenter 2003] C. A. Kennedy and M. H. Carpenter, "Additive Runge–Kutta schemes for convection-diffusion-reaction equations", *Appl. Numer. Math.* **44**:1-2 (2003), 139–181. MR 2003m:65111

[Layton and Minion 2004] A. T. Layton and M. L. Minion, "Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics", *J. Comput. Phys.* **194**:2 (2004), 697–715. MR 2004k:76089

[Layton and Minion 2005] A. T. Layton and M. L. Minion, "Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations", *BIT* **45**:2 (2005), 341–373. MR 2006h:65087

[Liotta et al. 2000] S. F. Liotta, V. Romano, and G. Russo, "Central schemes for balance laws of relaxation type", *SIAM J. Numer. Anal.* **38**:4 (2000), 1337–1356. MR 2001j:65128

[Minion 2003] M. L. Minion, "Semi-implicit spectral deferred correction methods for ordinary differential equations", *Commun. Math. Sci.* **1**:3 (2003), 471–500. MR 2005f:65085

[Minion 2004] M. L. Minion, "Semi-implicit projection methods for incompressible flow based on spectral deferred corrections", *Appl. Numer. Math.* **48**:3-4 (2004), 369–387. Workshop on Innovative Time Integrators for PDEs. MR MR2056924

[Pareschi and Russo 2001] L. Pareschi and G. Russo, "Implicit-explicit Runge–Kutta schemes for stiff systems of differential equations", pp. 269–288 in *Recent trends in numerical analysis*, Adv. Theory Comput. Math. **3**, Nova Sci. Publ., Huntington, NY, 2001. MR 2005a:65065

[Pareschi and Russo 2005] L. Pareschi and G. Russo, "Implicit-explicit Runge–Kutta schemes and applications to hyperbolic systems with relaxation", *J. Sci. Comput.* **25**:1-2 (2005), 129–155. MR MR2231946

[Pathria 1997] D. Pathria, "The correct formulation of intermediate boundary conditions for Runge–Kutta time integration of initial-boundary value problems", *SIAM J. Sci. Comput.* **18**:5 (1997), 1255–1266. MR 98d:65100

[Portero et al. 2004] L. Portero, J. C. Jorge, and B. Bujanda, "Avoiding order reduction of fractional step Runge–Kutta discretizations for linear time dependent coefficient parabolic problems", *Appl. Numer. Math.* **48**:3-4 (2004), 409–424. Workshop on Innovative Time Integrators for PDEs. MR 2005d:65153

[Prothero and Robinson 1974] A. Prothero and A. Robinson, "On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations", *Math. Comp.* **28** (1974), 145–162. MR 48 #10125

[Sanz-Serna et al. 1987] J. M. Sanz-Serna, J. G. Verwer, and W. H. Hundsdorfer, "Convergence and order reduction of Runge–Kutta schemes applied to evolutionary problems in partial differential equations", *Numer. Math.* **50**:4 (1987), 405–418. MR 88f:65146

[Shen and Zhong 1996] J. W. Shen and X. Zhong, "Semi-implicit Runge–Kutta schemes for the non-autonomous differential equations in reactive flow computations", in *Proceedings of the 27th AIAA Fluid Dynamics Conference* (New Orleans, 1996), Amer. Inst. Aeronautics and Astronautics, June 1996.

[Widlund 1967] O. B. Widlund, "A note on unconditionally stable linear multistep methods", *Nordisk Tidskr. Informations-Behandling* **7** (1967), 65–70. MR 35 #6373

[Zhong 1996] X. Zhong, "Additive semi-implicit Runge–Kutta methods for computing high-speed nonequilibrium reactive flows", *J. Comput. Phys.* **128**:1 (1996), 19–31. MR 97e:80019

Received December 5, 2005.

ANITA T. LAYTON: alayton@math.duke.edu
Department of Mathematics, Duke University, Box 90320, Durham, NC 27708, United States
http://www.math.duke.edu/faculty/alayton

MICHAEL L. MINION: minion@amath.unc.edu
Department of Mathematics, CB 3250 Phillips Hall, University of North Carolina, Chapel Hill, NC 27599, United States
http://amath.unc.edu/Minion

COMM. APP. MATH. AND COMP. SCI.
Vol. 2, No. 1, 2007

THE EXTENDED FINITE ELEMENT METHOD FOR BOUNDARY LAYER PROBLEMS IN BIOFILM GROWTH

BRYAN G. SMITH, BENJAMIN L. VAUGHAN JR. AND DAVID L. CHOPP

In this paper, we use the eXtended Finite Element Method, with customized enrichment functions determined by asymptotic analysis, to study boundary layer behavior in elliptic equations with discontinuous coefficients. In particular, we look at equations where the coefficients are discontinuous across a boundary internal to the domain. We also show how to implement this method for Dirichlet conditions at an interface. The method requires neither the mesh to conform to the internal boundary, nor the mesh to have additional refinement near the interface, making this an ideal method for moving interface type problems. We then apply this method to equations for linearized biofilm growth to study the effects of biofilm geometry on the availability of substrate and the effect of tip-splitting in biofilm growth.

1. Introduction

Consider the equation

$$\nabla \cdot (\beta \nabla u) + \kappa u = f, \tag{1}$$

on a two-dimensional domain, Ω, containing an interface Γ. The coefficients β, κ, and f may be discontinuous across Γ, leading to discontinuities in the normal derivative of the solution, u. In addition, large variations between these coefficients can lead to the existence of boundary layers near the interface, particularly when β is small relative to κ or f.

This type of equation governs a wide range of physical processes, and thus a variety of numerical methods have been devised to solve it. The three most commonly used approaches are finite difference methods — such as those described in [9; 11; 14] — finite element methods, and boundary element methods. One method that has been shown to perform well in comparisons with other methods is

MSC2000: 65N30, 92B05.

Keywords: X-FEM, extended finite element method, level set method, elliptic equations, Helmholtz equation, biofilms.

Smith was supported in part by the DoD NDSEG Fellowship Program and the Chicago Chapter of the ARCS Foundation. Vaughan and Chopp were supported in part by a grant from the NIH under contract #R01- GM067248.

.

the eXtended Finite Element Method (X-FEM) [8; 16]. In fact, it was shown in [12] that this method produced significantly more accurate solutions near the interface Γ. The X-FEM is a partition of unity method [15] in which extra functions containing information about the location of an interface are added to the standard finite element approximation. The method is particularly suited to problems involving a moving interface, since it does not require a conforming mesh, and can be easily coupled to interface tracking methods such as the level set method [17].

The X-FEM has been used to model crack growth [7; 16; 18; 20], solidification [3; 10], arbitrary fixed material interfaces and voids [19], rigid particles in Stokes flow [21], and multiphase fluids [2]. The majority of this work has relied on generic enrichment functions, such as Heaviside or absolute value functions, to enforce interfacial boundary conditions. However, in [7], it was shown that a special enrichment function given by the exact near-tip asymptotic functions, could be added to the approximation at the crack tip to improve accuracy. In this paper, we show how the X-FEM can be used to efficiently model systems that exhibit boundary layer behavior at the interface by expanding the function space with customized enrichment functions along the entire interface, presenting as an example a new exponential enrichment function that captures solutions which grow or decay rapidly near the interface.

The application to which we apply this method concerns the growth of biofilms. Biofilms are microbial communities that grow attached to solid surfaces, and are one of the most ubiquitous forms of life on the planet. More than 90% of all bacteria live in biofilms [1]. Biofilms are used in wastewater treatment and bioremediation of contaminated soils, lakes, and rivers. Biofilms also cause disease in plants and animals, and damage pipes, heat exchangers, and ship hulls. Consequently, understanding biofilms is important for a wide range of health and engineering disciplines. The work we present here is toward a goal of building continuum level model tools for simulating biofilm growth and development.

In this paper, we solve the linearized biofilm growth equations in order to determine how the geometry of the colony affects the profile of its growth, addressing two specific phenomena: fingering growth due to a tip-splitting instability and the shadowing effect generated by large colonies.

2. Numerical method

The eXtended Finite Element method (X-FEM) is a modified finite element method used for approximating the solution to (1). However, unlike conventional finite element methods, the X-FEM does not require the mesh to conform to the interface geometry, allowing us to utilize a standard Cartesian mesh. This is particularly

useful in problems involving moving interfaces, due to the fact that the mesh does not have to be regenerated at each timestep.

The X-FEM avoids the need for a conforming mesh by supplementing the standard finite element functions with enrichment functions that contain information about the interface. The X-FEM approximation for the solution u is written as

$$u^h(x, y) = \sum_{n_i \in N} \phi_i(x, y)\, u_i + \sum_k \sum_{n_j \in N_E} \phi_j(x, y)\, \psi_k(\varphi)\, a_j, \tag{2}$$

where N is the set of all nodes in the domain, N_E is the set of enriched nodes, ϕ is a standard finite element test function, ψ_k is the enrichment function, and φ is the signed distance function representing the interface.

Here we describe only the construction of the enrichment functions as well as the application of boundary conditions. A detailed treatment of the full X-FEM implementation is found in [12].

2.1. ***Determining enrichment functions.*** The enrichment functions added to the standard finite element approximation in the X-FEM serve two purposes. First, they encode the location of the interface into the function space itself, which allows for the application of both Dirichlet and Neumann interface conditions without the need for a conforming mesh. Second, they may contain information about the asymptotic behavior of the solution near the interface, allowing the method to accurately capture boundary layer behavior without the need for a very fine mesh. Here we test three types of enrichment functions. The first two, the step enrichment and the absolute value enrichment, are generic enrichment functions that can be applied to any problem that is continuous across the interface. The third, the exponential enrichment, is a custom enrichment function that is tailored to problems that display exponential behavior within the interfacial boundary layer. Each enrichment is a function only of the signed distance from the interface:

$$\varphi = \pm \min_{X \in \Gamma} \|x - X\|,$$

where the sign depends on whether x is inside or outside the region enclosed by the interface Γ. This signed distance function is typically determined using the Level Set Method [17].

The step and absolute value enrichment functions are defined as

$$\psi_{\text{step}}(\varphi) = \begin{cases} 1 & \varphi > 0, \\ -1 & \varphi \le 0, \end{cases}$$

and

$$\psi_{\text{abs}}(\varphi) = \begin{cases} \varphi & \text{if } \varphi > 0, \\ -\varphi & \text{if } \varphi \le 0. \end{cases}$$

These are simple enrichment functions that can be implemented without any *a priori* knowledge of solution behavior near the interface.

The exponential enrichment function is defined as

$$\psi_{\exp}(\varphi) = \begin{cases} 1 - e^{\mu\varphi} & \text{if } \varphi > 0, \\ -\varphi & \text{if } \varphi \leq 0. \end{cases}$$

This enrichment is most effective when asymptotic analysis indicates the presence of a boundary layer which exhibits exponential behavior. The value of the parameter μ depends on the specific parameters of the problem. It is important to note that both the absolute value and exponential enrichments decay to zero at the interface. This condition is necessary for stability when a nonzero Dirichlet condition is applied at the interface.

While the X-FEM has traditionally only added enrichment functions to elements containing the interface, it can be advantageous to enrich beyond this layer of elements when using enrichment functions other than the step enrichment. This is particularly important when using a custom enrichment because it allows the enrichment function to be applied throughout the boundary layer, thus capturing the behavior more completely. In the case of the absolute value enrichment, extending the enrichment distance effectively increases the order of the finite element approximation in the critical region near the interface. It is also useful to enrich to a different distance on either side of the interface when the enrichment function is asymmetric. As it is difficult to determine the optimal enrichment distance analytically, trial and error is generally the best method. For the problems presented here, an enrichment distance of $\mu \log 10^{-3}$ is used for exponential functions and $\mu \log 10^{-2}$ is used for linear functions. These correspond to the distance it takes the function $e^{\mu\varphi}$ to decay to 10^{-3} and 10^{-2} respectively.

2.2. ***Interface conditions.*** Once the enrichment function has been determined, the next consideration is the proper application of the boundary conditions. Here we only consider problems which are continuous across the interface, so there are two types of conditions that can be applied: Dirichlet conditions, $u = h(x, y)$, and Neumann jump conditions, $\left[\!\left[\beta \hat{n} \cdot \nabla u\right]\!\right] = v(x, y)$, where $[\![\cdot]\!]$ indicates the jump across Γ.

Neumann jump conditions. The Neumann jump condition is enforced primarily by introducing a line source term to the right hand side of (1). This line source has strength $v(x, y)$ and is written as

$$\int_{\Gamma} v(x, y)\, \delta(x - X(s))\, d\Gamma, \tag{3}$$

where $X(s)$ is the parameterized interface.

This technique can be used with both continuous and discontinuous enrichment functions and typically yields the best results with the Step enrichment. However, approximations using continuous enrichment functions can also utilize Lagrange multipliers to enforce jump conditions. In this case, the governing equation is rewritten as

$$\nabla \cdot (\beta \nabla u) + \kappa u + \left[\!\left[\beta \frac{\partial u}{\partial n} \right]\!\right] \lambda = f + \int_{\Gamma} v\,(x,\, y)\, \delta\,(x - X\,(s))\; d\Gamma,$$

along with the condition

$$\left[\!\left[\beta \hat{n} \cdot \nabla u \right]\!\right] = v\,(x,\, y)\,.$$

A one dimensional finite element mesh is laid down on the interface using piecewise linear elements, and the Lagrange multipliers are approximated by:

$$\lambda^h = \sum_{m_i \in M} \theta_i \lambda_i,$$

where M is the set of all Lagrange multiplier nodes.

Dirichlet conditions. Dirichlet conditions on an internal interface have not been considered in previous implementations of the X-FEM. This type of condition is unusual because it divides the domain into two independent pieces, which are typically solved separately. However, in order to avoid the generation of meshes which conform to each piece, it can be useful to solve on the whole domain at once.

In contrast to the derivative jump conditions, Dirichlet interface conditions can only be applied using Lagrange multipliers. This is done in a similar manner as above, with the governing equation rewritten to include the extra terms:

$$\nabla \cdot (\beta \nabla u) + \kappa u + \lambda u \;|_{\Gamma} = f,$$

with the condition

$$u|_{\Gamma} = h\,(x,\, y)\,.$$

For systems with continuous enrichment functions, only one set of Lagrange multipliers is necessary to apply the boundary condition. Discontinuous enrichments, however, do not have a single value at the interface, so the condition must be applied to the positive and negative sides separately, requiring two sets of Lagrange multipliers.

2.3. ***Evaluation of interface derivatives.*** In order to accurately determine the derivative jump across the interface, a domain integral method, similar to the one described in [8], is employed. We first consider a section of interface, Γ_i, within the support, Ω_i, of a test function ϕ_i. Equation (1) is multiplied by ϕ_i and integrated

over Ω_i. Integrating by parts yields

$$\int_{\Omega_i} \left(-\beta \nabla u \cdot \nabla \phi_i + \kappa u \phi_i - f \phi_i\right) d\Omega_i = \int_{\Gamma_i} \left[\!\left[\beta \hat{n} \cdot \nabla u\right]\!\right]_{\Gamma_i} \phi_i \, d\Gamma_i. \tag{4}$$

Using a first order approximation, the derivative jump is assumed to be constant along Γ_i. Equation (4) can then be rewritten as

$$\left[\!\left[\beta \hat{n} \cdot \nabla u\right]\!\right]_{\Gamma_i} = \frac{1}{\int_{\Gamma_i} \phi_i \, d\Gamma_i} \int_{\Omega_i} \left(-\beta \nabla u \cdot \nabla \phi_i + \kappa u \phi_i - f \phi_i\right) d\Omega_i. \tag{5}$$

To then determine the jump at a single point on the interface, x_d, the jump across the interface segment is determined using each test function, ϕ_i, which has support containing x_d. In order to avoid oscillations due to roundoff error, interface segments are discarded if the integral of ϕ_i over the segment is less than 10^{-13}. These values are then weighted appropriately and summed:

$$\left[\!\left[\beta \hat{n} \cdot \nabla u\right]\!\right]_{x_d} = \sum_j \left[\!\left[\beta \hat{n} \cdot \nabla u\right]\!\right]_{\Gamma_j} \phi_j \left(x_d\right).$$

3. Results

In this section, different types of enrichment functions are applied to the solution of two example problems. In each problem, exponential enrichment functions are compared to both absolute value enrichments and step enrichments when finding a solution that contains a boundary layer. For each problem the interface Γ is the circle $\left(x - \frac{1}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2 = \frac{1}{16}$ within the domain $0 \le x, y \le 1$.

We solve each example problem using both types of interface conditions discussed in Section 2.2. These two types of boundary conditions demonstrate how the custom enrichment function can improve the accuracy of both the solution and its derivative at the interface.

3.1. ***Example 1: enriching with the exact solution.*** The first example shows the improvement in the finite element approximation when the exact solution is included in the enriched function space. The differential equation is

$$\nabla^2 u + \frac{\kappa}{\epsilon}\left(\frac{1}{\epsilon} - \frac{1}{r}\right) u = 0, \tag{6}$$

where

$$\kappa\left(x, y\right) = \begin{cases} 0 & \text{if } r \le \frac{1}{4}, \\ 1 & \text{if } r > \frac{1}{4}. \end{cases}$$

The exact solution to (6) is

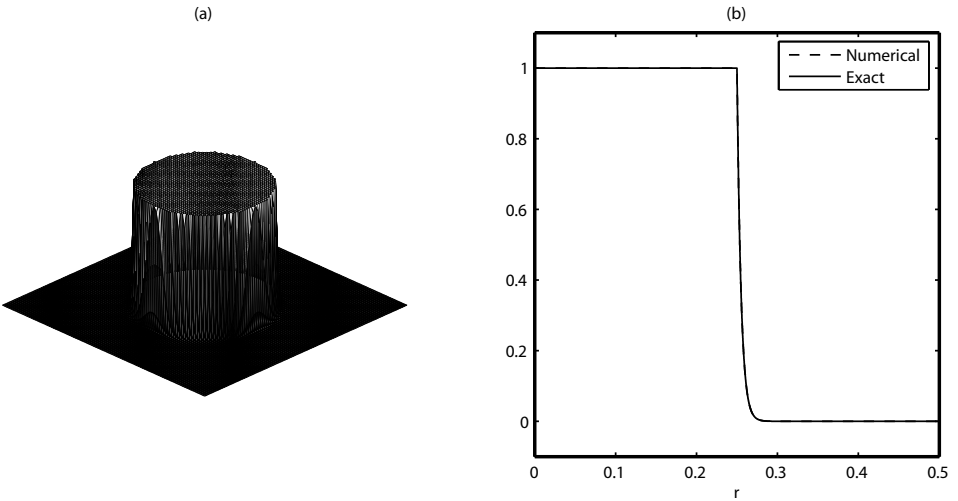


Figure 1. Solution of Example 1. (a) A surface plot of the solution on the entire domain. (b) A cross section of the solution, taken at an angle of one radian. The numerical approximation overlaps the exact solution and is computed using the exponential enrichment function on a 799×799 grid.

$$u(x, y) = \begin{cases} 1 & \text{if } r \leq \frac{1}{4}, \\ \exp\left[\frac{1}{\epsilon}\left(\frac{1}{4} - r\right)\right] & \text{if } r > \frac{1}{4}, \end{cases}$$

where ϵ is $\frac{1}{200}$. The expression $\frac{1}{4} - r$ is equivalent to the level set variable φ, so the exact solution on the exterior of the circle is contained within the exponential enrichment function.

Equation (6) is solved with two sets of boundary conditions. The first prescribes continuity and sets a derivative jump condition on Γ:

$$[\![u]\!] = 0, \qquad \left[\!\left[\frac{\partial u}{\partial n}\right]\!\right] = \frac{1}{\epsilon}.$$

Because the derivative jump along the interface is already known, the important simulation results to consider are the solution values along the interface and throughout the domain.

Table 1 shows the convergence results for the X-FEM using three types of enrichment functions: the step enrichment, absolute value enrichment, and exponential enrichment. The error given is the maximum error at the nodes, defined as

$$\|T_n\|_\infty = \max_{\substack{i \\ n_i \in N}} \{\left|u(x_i, y_i) - u_i^h\right|\},$$

n	Step		Absolute Value		Exponential	
	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio
49	1.3367×10^{-1}		2.8126×10^{-1}		3.6922×10^{-2}	
99	7.8584×10^{-2}	1.701	4.9647×10^{-2}	5.665	8.1788×10^{-3}	4.514
199	2.6213×10^{-2}	2.998	9.8545×10^{-3}	5.038	1.8534×10^{-3}	4.413
399	8.5907×10^{-3}	3.051	3.5610×10^{-3}	2.760	4.6404×10^{-4}	3.994
799	2.9094×10^{-3}	2.953	9.7458×10^{-4}	3.663	1.1565×10^{-4}	4.013

Table 1. Domain results for Example 1 with Neumann jump interface conditions.

where N is the set of all nodes, (x_i, y_i) is the location of the i-th node, and u_i^h is the computed value at the node. The ratios of successive errors are included as well to show convergence rates. As Table 1 shows, the exponential enrichment is more accurate than both the absolute value and step enrichments, and it converges more uniformly as well.

Of greater importance for moving interface problems is the accuracy of the solution at the interface. This can be problematic for some methods even though they exhibit good accuracy away from the interface [12]. Table 2 shows the error interpolated along the interface. This error is computed at 1,000 evenly spaced points along the parameterized interface. The results are similar to those seen in Table 1, but the disparity between the exponential enrichment and the other enrichments is more significant.

The second boundary condition we consider is a Dirichlet condition on both sides of the interface:

$$u = 1.$$

n	Step		Absolute Value		Exponential	
	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio
49	3.4331×10^{-1}		4.5436×10^{-1}		3.6860×10^{-2}	
99	1.4444×10^{-1}	2.377	1.3363×10^{-1}	3.400	9.1137×10^{-3}	4.045
199	3.9447×10^{-2}	3.662	4.5511×10^{-2}	2.936	1.9652×10^{-3}	4.638
399	1.1170×10^{-2}	3.531	1.4101×10^{-2}	3.228	5.3682×10^{-4}	3.661
799	3.1200×10^{-3}	3.580	3.6471×10^{-3}	3.866	1.3277×10^{-4}	4.043

Table 2. Interface results for Example 1 with Neumann Jump interface conditions.

n	Step		Absolute Value		Exponential	
	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio
49	2.5623×10^{-2}		1.9174×10^{-1}		1.1620×10^{-2}	
99	2.2492×10^{-2}	1.135	6.9352×10^{-2}	2.765	4.4235×10^{-3}	2.627
199	1.4385×10^{-2}	1.564	2.1398×10^{-2}	3.241	8.7237×10^{-4}	5.071
399	6.0928×10^{-3}	2.361	5.9632×10^{-3}	3.588	3.1170×10^{-4}	2.799
799	1.9458×10^{-3}	3.131	1.5402×10^{-3}	3.872	6.0066×10^{-5}	5.189

Table 3. Domain results for Example 1 with Dirichlet interface conditions.

The exact solution remains the same as above. Here, the value at the interface is known, so the important computational results are the solution values within the domain and the jump in the derivative across the interface.

Table 3 contains the convergence results for the X-FEM using this boundary condition. Once again, the asymptotic enrichments perform much better than the other two, by more than an order of magnitude. Table 4 shows the gradient results on the interface. Due to the magnitude of the gradient jump $(\sim 10^2)$, this table presents relative errors rather than absolute errors:

$$\left\|T_n'\right\|_\infty = \frac{1}{1/\epsilon} \max_{\substack{i \\ n_i \in N}} \{\left|u\left(x_i, y_i\right) - u_i^h\right|\}.$$

3.2. ***Example 2: enriching with an asymptotic solution.*** In the second example, the custom enrichment function does not contain the exact solution, but instead includes the asymptotic approximation of the solution near the interface. The

n	Step		Absolute Value		Exponential	
	$\left\|T_n'\right\|_\infty$	ratio	$\left\|T_n'\right\|_\infty$	ratio	$\left\|T_n'\right\|_\infty$	ratio
49	4.2738×10^{-1}		2.2663×10^{-1}		3.1433×10^{-2}	
99	1.9041×10^{-1}	2.245	9.1789×10^{-2}	2.469	1.5427×10^{-2}	2.038
199	6.8870×10^{-2}	2.765	3.4742×10^{-2}	2.642	3.8923×10^{-3}	3.964
399	3.1636×10^{-2}	2.177	1.7591×10^{-2}	1.975	1.4946×10^{-3}	2.604
799	1.3817×10^{-2}	2.290	8.5566×10^{-3}	2.056	5.5810×10^{-4}	2.678

Table 4. Interfacial gradient results for Example 1 with Dirichlet interface conditions.

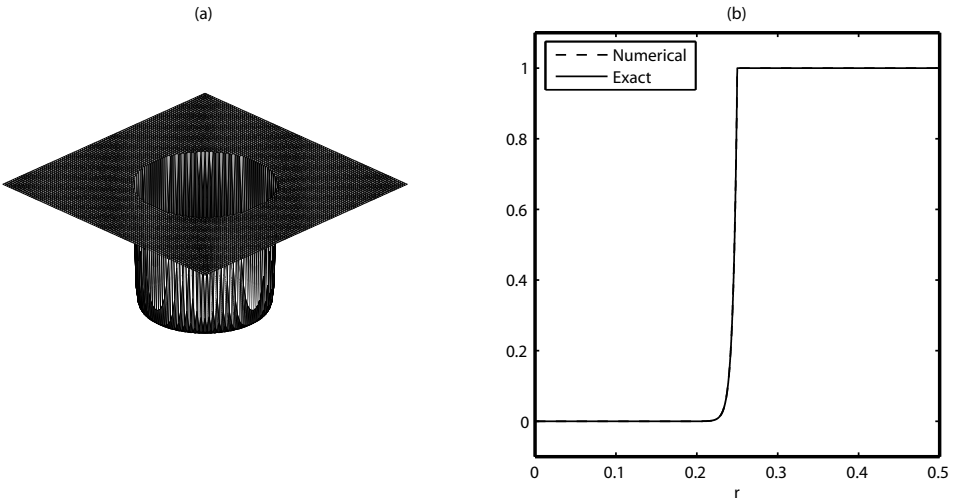


Figure 2. Solution of Example 2. (a) A surface plot of the solution on the entire domain. (b) A cross section of the solution, taken at an angle of one radian. The numerical approximation overlaps the exact solution and is computed using the exponential enrichment function on a 799×799 grid.

equation in this example is the Helmholtz Equation on the interior of a circle with radius $\frac{1}{4}$:

$$\nabla^2 u + \kappa^2 u = 0, \tag{7}$$

where κ is given by

$$\kappa\,(x, y) = \begin{cases} 200 & \text{if } r \leq \frac{1}{4}, \\ 0 & \text{if } r > \frac{1}{4}. \end{cases}$$

The exact solution to (7) is

$$u\,(x, y) = \begin{cases} \frac{I_0(\kappa r)}{I_0\left(\frac{\kappa}{4}\right)} & \text{if } r \leq \frac{1}{4}, \\ 1 & \text{if } r > \frac{1}{4}, \end{cases}$$

where $I_n\,(x)$ is the modified Bessel function of the first kind.

An asymptotic analysis of (7) shows that the behavior of the solution in the boundary layer near the interface is of the form $u \approx e^{\kappa r}$. Therefore, the exponential enrichment function is appropriate here as well, using κ as the parameter. It is important to note that while an exact solution to (7) is available, we only use information from the asymptotic analysis to construct our approximation.

As in Example 1, we once again consider two types of interface conditions. First, the solution is again prescribed to be continuous, with a jump in the normal

n	Step		Absolute Value		Exponential	
	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio
49	1.6496×10^{-1}		2.8971×10^{-1}		1.8642×10^{-2}	
99	7.7465×10^{-2}	2.130	5.8281×10^{-2}	4.971	5.4660×10^{-3}	3.429
199	2.8396×10^{-2}	2.728	1.1361×10^{-2}	5.130	1.5259×10^{-3}	3.582
399	8.8436×10^{-3}	3.211	2.4660×10^{-3}	4.607	3.8795×10^{-4}	3.933
799	2.8451×10^{-3}	3.108	8.1249×10^{-4}	3.035	9.7831×10^{-5}	3.966

Table 5. Domain results for Example 2 with Neumann jump interface conditions.

derivative at the interface:

$$[\![u]\!] = 0, \qquad \left[\!\!\left[\frac{\partial u}{\partial n}\right]\!\!\right] = \kappa \frac{I_1(\kappa r)}{I_0\left(\frac{\kappa}{4}\right)}.$$

Table 5 shows the error on the domain, and Table 6 shows the error on the interface. The results are similar to those in the Example 1.

Like the first example, the second set of boundary conditions applies a Dirichlet condition at the interface, and the solution gradient at the interface is the desired result:

$$u = 1.$$

Table 7 shows that the asymptotic enrichment again outperforms the other generic enrichment functions with an error of less that 0.1% for the largest system. For the sake of completeness, Table 8 charts the maximum error on the domain.

Finally, the exponential enrichment function not only increases the accuracy at the nodes and the interface, it improves the approximation within the elements as well. Because the asymptotic solution is contained in the interpolant, fewer

n	Step		Absolute Value		Exponential	
	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio
49	3.2720×10^{-1}		3.6044×10^{-1}		3.4735×10^{-2}	
99	1.2621×10^{-1}	2.593	1.1859×10^{-1}	3.040	7.1808×10^{-3}	4.837
199	3.9873×10^{-2}	3.165	4.0367×10^{-2}	2.938	1.9359×10^{-3}	3.709
399	1.0574×10^{-2}	3.771	1.2436×10^{-2}	3.246	4.9710×10^{-4}	3.887
799	2.9643×10^{-3}	3.567	3.3563×10^{-3}	3.705	1.2127×10^{-4}	4.106

Table 6. Interface results for Example 2 with Neumann jump interface conditions.

n	Step		Absolute Value		Exponential	
	$\|T_n'\|_\infty$	ratio	$\|T_n'\|_\infty$	ratio	$\|T_n'\|_\infty$	ratio
49	3.5875×10^{-1}		2.3265×10^{-1}		3.6010×10^{-2}	
99	1.5830×10^{-1}	2.266	9.0700×10^{-2}	2.565	1.6945×10^{-2}	2.125
199	5.7291×10^{-2}	2.763	4.0960×10^{-2}	2.214	4.0448×10^{-3}	4.189
399	2.4665×10^{-2}	2.323	1.8754×10^{-2}	2.184	1.6265×10^{-3}	2.487
799	1.2950×10^{-2}	1.905	9.9692×10^{-3}	2.067	5.9443×10^{-4}	2.736

Table 7. Interface gradient results for Example 2 with Dirichlet interface conditions.

n	Step		Absolute Value		Exponential	
	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio	$\|T_n\|_\infty$	ratio
49	3.0534×10^{-2}		1.3923×10^{-1}		6.6740×10^{-3}	
99	2.2035×10^{-2}	1.386	6.1419×10^{-2}	2.267	3.1346×10^{-3}	2.129
199	1.4526×10^{-2}	1.517	1.6752×10^{-2}	3.666	7.4375×10^{-4}	4.215
399	6.6449×10^{-3}	2.186	4.8119×10^{-3}	3.481	2.8600×10^{-4}	2.601
799	2.0239×10^{-3}	3.283	1.2758×10^{-3}	3.772	5.6394×10^{-5}	5.071

Table 8. Domain results for Example 2 with Dirichlet interface conditions.

elements are needed to capture the boundary layer. Figure 3 shows the exact solution plotted next to the numerical approximation for varying mesh sizes. Each plot is generated by evaluating the exact solution and the numerical approximation at 10,000 points along a line from $r = 0.23$ to $r = 0.26$ at an angle of one radian. This clearly demonstrates the advantage of using a customized enrichment function, as the exponential enrichment accurately approximates the solution on a 25×25 grid, where the entire boundary layer is contained within one element.

4. Biofilm growth

We now consider the problem of bacterial biofilms. The addition of substrate nutrients to the top of the film drives the growth of the biofilm. Here we explore how the geometry of the biofilm colonies affects the profile of their growth. We first consider the phenomenon of tip-splitting [4], where the colony does not grow directly toward the substrate, but develops fingers that grow at an angle into the film. We then analyze how the shadowing effect of a large colony decays with distance.

The system is considered on a domain Ω, periodic in x and consisting of two regions, the interior of the bacterial colonies, Ω_b, and the exterior fluid film, Ω_f.

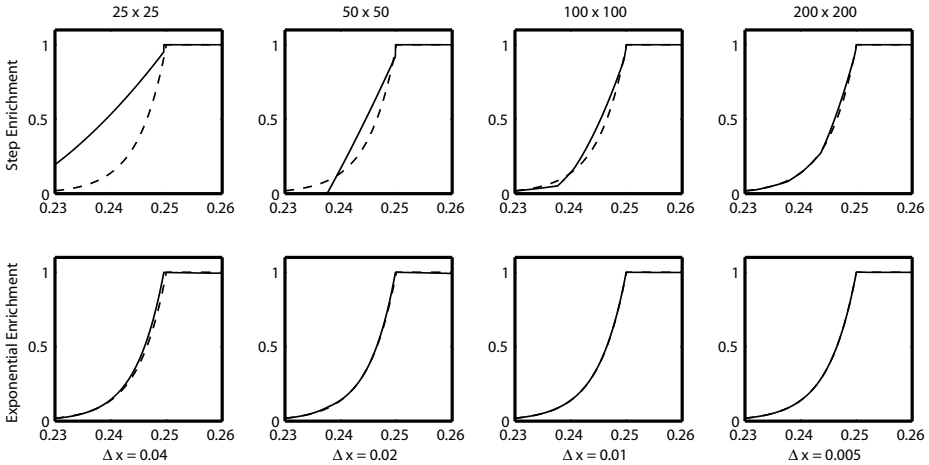


Figure 3. Cross sections generated using various mesh sizes. Each cross section is taken at an angle of one radian, to avoid alignment with the mesh. Note that in the coarsest plot, the entire boundary layer is contained in one element.

The substrate concentration s and velocity potential φ are governed by

$$D\nabla^2 s = \alpha s \tag{8}$$

and

$$\nabla^2 \varphi = \beta s, \tag{9}$$

where

$$D(x, y) = \begin{cases} 120 & \text{if } (x, y) \in \Omega_b, \\ 150 & \text{if } (x, y) \in \Omega_f, \end{cases}$$

$$\alpha(x, y) = \begin{cases} 3.6 \times 10^6 & \text{if } (x, y) \in \Omega_b, \\ 0 & \text{if } (x, y) \in \Omega_f, \end{cases}$$

and

$$\beta(x, y) = \begin{cases} 10^6 & \text{if } (x, y) \in \Omega_b, \\ 0 & \text{if } (x, y) \in \Omega_f. \end{cases}$$

Figure 4 shows the boundary conditions applied to each equation. This follows the description given in [6] without erosion, with the parameters chosen to approximate the behavior of the nonlinear system described in [5]. Both the substrate concentration and the velocity potential are continuous throughout the domain. In the substrate equation, a derivative jump condition is applied at the interface between the colony and the liquid, and in the velocity equation, a Dirichlet condition

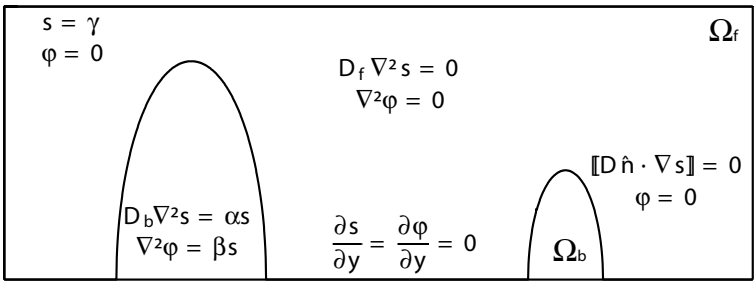


Figure 4. The domain and governing equations for the biofilm problem.

is prescribed:

$$\left[\!\left[D\hat{n}\cdot\nabla s\right]\!\right]=0, \qquad \varphi=0.$$

In both cases, a Dirichlet condition is applied at the top boundary:

$$s=10^{-5}, \qquad \varphi=0,$$

and a Neumann condition is applied at the bottom:

$$\frac{\partial s}{\partial y}=\frac{\partial \varphi}{\partial y}=0.$$

We solve for the growth rate profile for a given colony via a three step process. First, the substrate concentration is found by solving (8) with the given boundary conditions. This solution is then fed back into (9) to find the velocity potential within the colony, and finally the rate of growth at the biofilm surface is given by the normal derivative of the velocity potential at the interface, $\hat{n}\cdot\nabla\varphi$. We calculate this quantity at a large number of points along the parameterized interface to obtain an overall profile of the colony's growth. Both the substrate and velocity equations are solved using the X-FEM with the exponential enrichment function. The relevant boundary layer exists on the interior of the colony, and we use $\mu=\sqrt{\alpha/D_b}$ as the parameter in the exponential enrichment definition.

4.1. ***Tip splitting.*** As biofilm colonies grow, they do not always grow in a uniform fashion. Sometimes the tip of the colony splits into separate fingers instead [4]. This instability appears to be distinct from the fingering instability described in [6], where it is observed that peaks have a greater growth rate than valleys. Due to random fluctuations, one of the fingers generally overtakes the other resulting in an irregularly shaped colony. The conditions driving this instability are unknown, but a possible mechanism is the splitting of a mushroom-shaped tip due to the instability described in [6]. This is similar to the well known tip-splitting instability in the Stefan problem, where a flattened dendritic tip splits due to the Mullins-Sekerka

instability [13]. Here we look at the ways in which the shape of a biofilm colony can lead to mushrooming of the tip.

To study the onset of mushrooming, we examine the velocity profiles of individual colonies on a periodic domain. The period of the domain is large enough to prevent interaction between neighboring colonies, and all of the colonies are hemielliptical in shape. Due to this simple geometry, the interface is easily parameterized from 0 to π, and Figure 5 shows some examples of parameterized velocity profiles. In the first case, the biofilm has an aspect ratio (height/width) of 2.25, and the maximum velocity is obtained at the peak of the biofilm colony, precluding a change in the shape of the tip. In the second case, the colony is much taller and thinner, with an aspect ratio of 16, and the maximum velocity is obtained on either side of the peak, which may allow for mushrooming to take place. Thus, to determine whether or not mushrooming can occur, we consider the ratio of the maximum velocity to the peak velocity. If this ratio is one, the maximum occurs at the peak, and the colony will grow normally. The strength of the mushrooming instability increases as this ratio moves above unity.

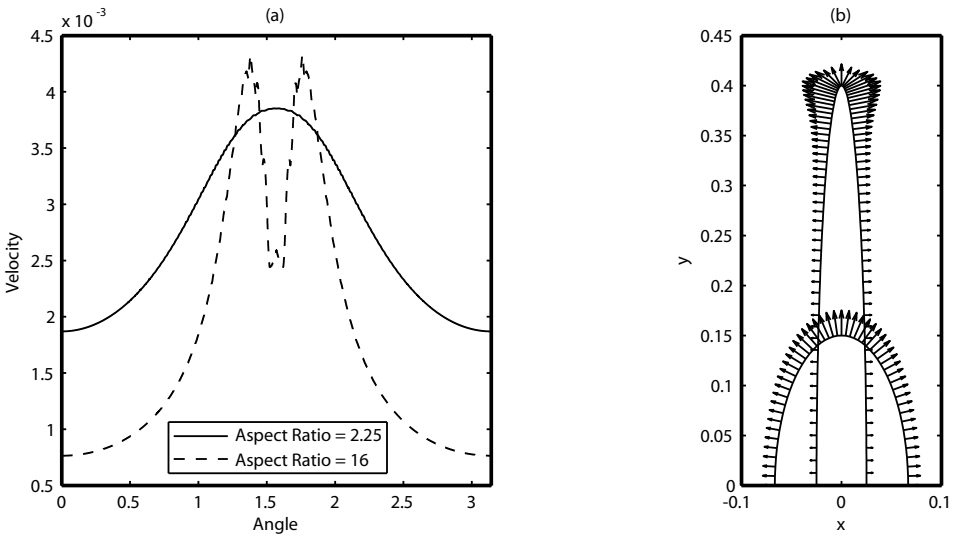


Figure 5. Representative velocity profiles. Shown here are two examples of velocity profiles, from colonies with different aspect ratios but the same mass. (a) Normal velocity vs. surface parameterization. In the second case, the maximum velocity is not attained at the tip, so mushrooming may occur. (b) Velocity vectors along the biofilm surface. The taller colony has a large aspect ratio, and the shape of the velocity field near the peak indicates that mushrooming may occur.

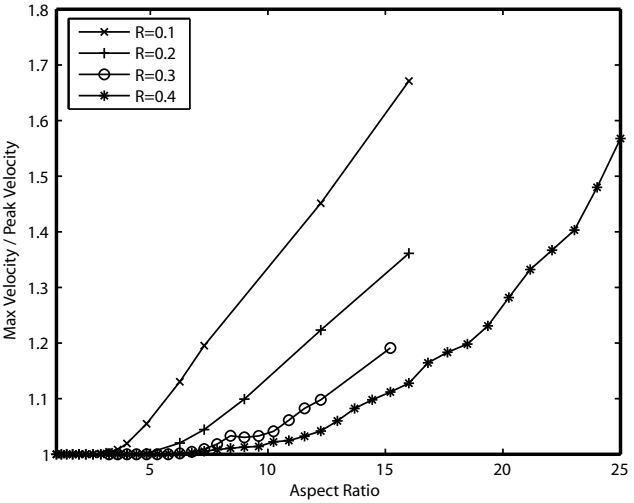


Figure 6. The effect of aspect ratio on mushrooming.

To demonstrate, we first explore how the aspect ratio of the hemielliptical colony affects the velocity profile. As we change the aspect ratio, we keep the area of the colony constant so that the total amount of biomass is unchanged. The colony shape is defined as

$$a^2 (x - x_0)^2 + \frac{y^2}{a^2} = R^2,$$

where a^2 is the aspect ratio of the ellipse, and $R^2/2$ is the area. We vary the aspect ratio between 1.0 to 25.0 (height to width) for colonies with three different masses. Figure 6 shows the relationship between the aspect ratio and the strength of the mushrooming instability. While the curves are qualitatively similar, the mass of the colony is clearly important. This mass effect can be removed by considering the radius of curvature at the colony peak, R/a^3, rather than the aspect ratio. Figure 7 contains the same velocity data as Figure 6 but plotted against the radius of curvature. In this case, all three curves lie on top of each other, indicating that tip splitting is a function of the radius of curvature at the tip of the colony. So that we may ensure that other factors are not also important, the height of the film and the period of the domain were varied. As Figure 8 shows, these variations have no effect.

This dependence on radius of curvature is probably due to elevated substrate concentrations within the biofilm colony. As the tip radius decreases, less biomass is present near the peak, and unconsumed substrate can diffuse farther down into the colony. To quantify the substrate penetration, we calculate the depth within the colony at which the concentration of substrate is 10% of the concentration at the colony tip. In order to compare biofilms containing different amounts of biomass, we normalize all of the values by the substrate penetration depth into a circular

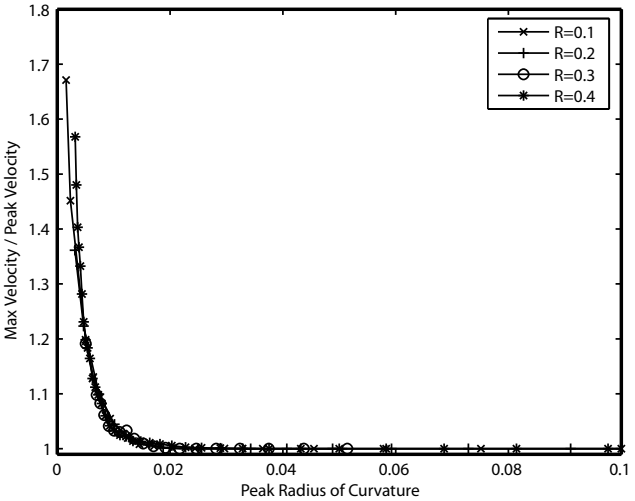


Figure 7. The effect of radius of curvature on mushrooming.

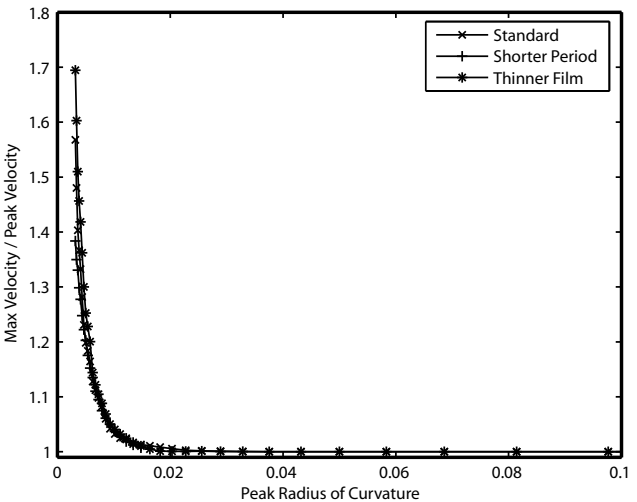


Figure 8. Domain period and film thickness have no effect on mushrooming.

colony with radius R. As shown in Figure 9, the substrate penetration depth is correlated with the radius of curvature.

4.2. ***Shadowing.*** Another important feature of biofilm growth is the interaction between neighboring colonies. Competition for substrate creates a shadowing effect near large colonies as nearby colonies are unable to attain their optimal growth rate. This shadowing effect depends on the relative sizes of the colonies as well as the separation distance. In order to determine the relative importance of these two

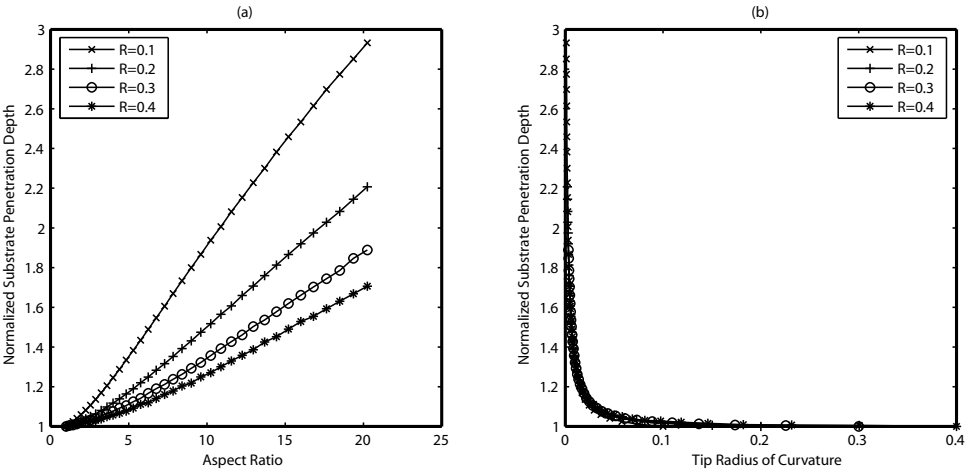


Figure 9. Normalized substrate penetration depths. (a) The normalized penetration depths for colonies of various sizes against the aspect ratio of the colony. (b) The normalized penetration depths against the radius of curvature at the colony tip.

parameters, we consider a system consisting of alternating large and small colonies along a periodic domain. In each case the large colony remains a standard size, with $a = 2.0$ and $R = 0.2$. The smaller colony maintains the same aspect ratio, but R is varied from 0.02 to 0.18. The different size cases are then simulated using a wide range of period lengths.

Proper determination of the effects of the interaction requires the maximum growth rate of an uninfluenced single colony, the natural growth rate for that colony. Due to the fact that the majority of a biofilm colony's growth occurs at or near the tip, this maximum growth rate is a reasonable measure of the overall growth rate of the colony. To find this rate, simulations are performed using a single colony on a periodic domain, using the same film height as in the combined system. The period of the domain is increased until maximum growth rate asymptotes to a constant value. Figure 10 shows maximum growth rate plotted against the period size for several representative colony sizes. Interestingly, the maximum growth rate of the smallest colonies is greater than that of the medium sized colonies. This effect is caused by mushrooming. When the colonies are relatively large, growth occurs primarily at the tip, and therefore the maximum growth rate scales with the size of the colony. However, in small colonies, growth occurs throughout the colony due to the greater substrate availability. This allows for large growth rates in areas far from the peak and creates the effect in Figure 10.

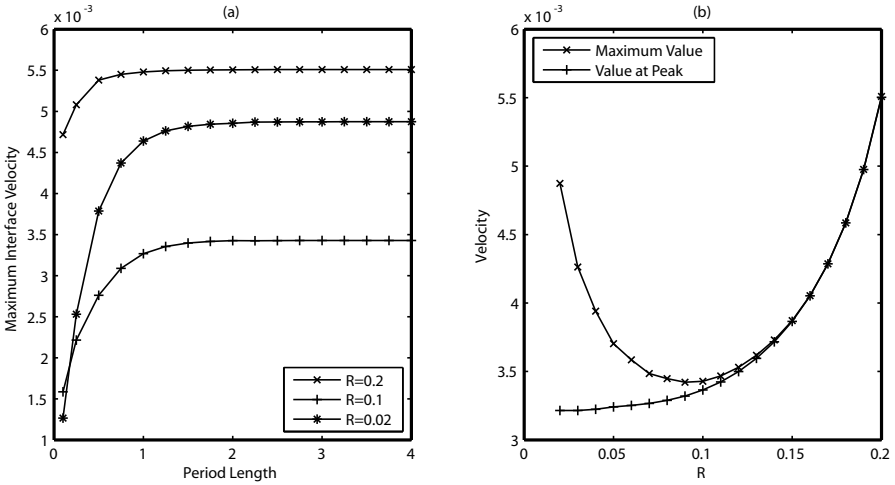


Figure 10. Selected natural growth rates. (a) The maximum growth rate for each colony size is reached on a domain with a relatively small period. (b) While the velocity at the peak decreases monotonically with colony size, the maximum velocity begins to increase again for the smallest colonies. This effect is caused by mushrooming.

Once the natural growth rate for each colony size has been found, it is possible to quantify the shadowing that takes place in the combined system. For each colony size and period length, the maximum growth rate can be expressed as a percentage of the natural rate for that colony. Figure 11 shows this ratio for each colony size, plotted against the length of the period. Using this data, it is then possible to create pictures of the "shadow" for a given influence percentage. For example, the portrait of a 50% influence shadow is shown in Figure 12. In the center of the figure is the large colony, surrounded by colonies placed so that their maximum growth rate is 50% of their natural rate. The tops of the colonies are connected to create a shadow region. Any colony which is contained within this shadow region will grow at a maximum of 50% of its natural rate. Of particular interest is the 97% shadow, shown in Figure 13, because rather than decaying asymptotically, the region of influence appears to have a sharp cutoff outside of which the presence of the large colony is virtually undetectable.

5. Conclusion

The eXtended Finite Element Method has already been shown to capture the location of an embedded interface without the need for a conforming mesh. This is

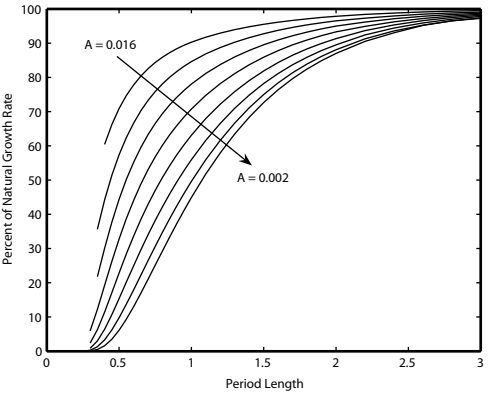


Figure 11. The effect of period length on shadowing. For all colony sizes, the shadowing effect of the large colony decreases for longer periods.

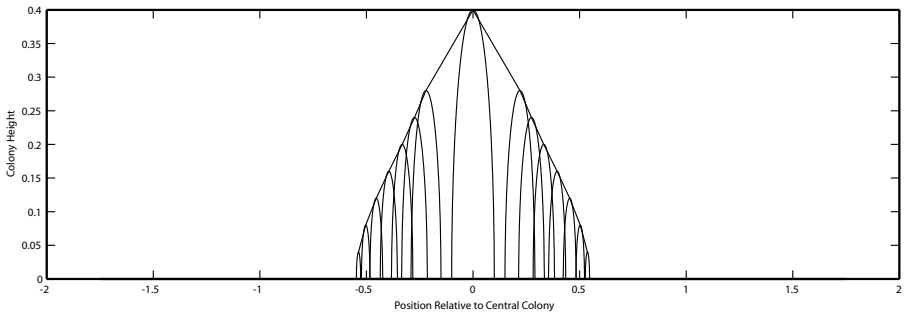


Figure 12. The 50% shadow. Any colony contained within this region around the large central colony will achieve less than 50% of its maximal growth rate.

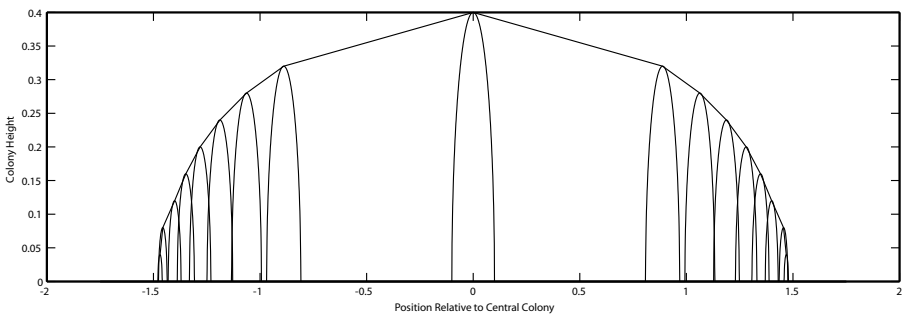


Figure 13. The 97% shadow. Any colony which falls outside of this region will achieve at least 97% of its maximal growth rate. While this shadow is significantly wider than the 50% shadow, it exhibits a steep drop off at the edges, so the effect of the central colony is largely confined to a finite area.

accomplished by adding "enriched" basis functions containing information about the interface position into the standard finite element approximation. Here we have used information about the asymptotic behavior of the solution near the interface to create customized enrichment functions which not only include the position of the interface but the boundary layer behavior of the solution as well.

The examples presented here have shown that when solutions exhibit extreme behavior in the boundary layer around the interface, the use of customized enrichment functions increases both the accuracy and the convergence rate of the numerical approximation. This is true even if the enrichment contains only the asymptotic approximation of the boundary layer behavior rather than the exact solution. While all of the problems presented here are in two dimensions, the method, including the customized enrichment functions, can be extended to three dimensions.

We then used this technique to simulate the growth of bacterial colonies in a thin film. In solving the linearized biofilm equations we have explored the relationship between the shape of the colony and the profile of its growth. The simulations have shown that the radius of curvature at the peak of the colony is the dominant factor in determining whether tip-splitting will occur as well as demonstrating that large colonies within a biofilm create a large but finite area of influence in which the growth of smaller colonies is inhibited.

References

[1] W. G. Characklis and K. C. Marshall, *Biofilms*, Wiley, 1990.

[2] J. Chessa and T. Belytschko, *An extended finite element method for two-phase fluids*, Transactions of the ASME **70** (2003), 10–17.

[3] J. Chessa et al., *The extended finite element method (xfem) for solidification problems.*, International Journal of Numerical Methods in Engineering **53** (2002), 1959–1977.

[4] D. L. Chopp, *Simulation of biofilms using the level set method*, Parametric and geometric deformable models: an application in biomaterials and medical imagery, Springer, 2006.

[5] D. L. Chopp, M. J. Kirisits, B. Moran, and M. R. Parsek, *The dependence of quorum sensing on the depth of a growing biofilm*, Bulletin of Mathematical Biology **65** (2003), 1053–1079.

[6] J. Dockery and I. Klapper, *Finger formation in biofilm layers*, SIAM J. Appl. Math. **62** (2001), no. 3, 853–869.

[7] J. E. Dolbow, N. Moës, and T. Belytschko, *Discontinuous enrichment in finite elements with a partition of unity method*, Finite Elements in Analysis and Design **36** (2000), 235–260.

[8] ———, *An extended finite element method for modeling crack growth with frictional contact*, Computational Methods in Applied Mechanics and Engineering **190** (2001), 6825–6846.

[9] F. Gibou, R. Fedkiw, L. Cheng, and M. Kang, *A second-order-accurate symmetric discretization of the poisson equation on irregular domains*, Journal of Computational Physics **176** (2002), 205–227.

[10] H. Ji, D. Chopp, and J. E. Dolbow, *A hybrid extended finite element/level set method for modeling phase transformation*, International Journal of Numerical Methods in Engineering **54** (2002), 1209–1233.

[11] H. Johansen and P. Colella, *A cartesian grid embedded boundary method for poisson's equation on irregular domains*, Journal of Computational Physics **147** (1998), 60–85.

[12] B. L. V. Jr., B. G. Smith, and D. L. Chopp, *A comparison of the extended finite element method for elliptic equations with discontinuous coefficients and singular sources*, Communications in Applied Mathematics and Computational Science **1** (2006), no. 1, 207–228.

[13] J. S. Langer, *Instabilities and pattern formation in crystal growth*, Review of Modern Physics **52** (1980), no. 1, 1–28.

[14] R. J. LeVeque and Z. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM Journal Numerical Analysis **31** (1994), 1019–1044.

[15] J. M. Melenk and I. Babuška, *The partition of unity finite element method: basic theory and applications*, Computational Methods in Applied Mechanics and Engineering **139** (1996), 289–314.

[16] N. Moës, J. Dolbow, and T. Belytschko, *A finite element method for crack growth without remeshing*, International Journal of Numerical Methods in Engineering **46** (1999), 131–150.

[17] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms base on hamilton-jacobi formulations*, Journal of Computational Physics **79** (1988), 12–49.

[18] M. Stolarska, D. L. Chopp, N. Moës, and T. Belytschko, *Modelling crack growth by level sets in the extended finite element method*, International Journal of Numerical Methods in Engineering **51** (2001), 943–960.

[19] N. Sukumar, D. Chopp, N. Moës, and T. Belytschko, *Modeling holes and inclusions by level sets in the extended finite element method*, International Journal of Numerical Methods in Engineering **48** (2000), 1549–1570.

[20] N. Sukumar, D. L. Chopp, and B. Moran, *Extended finite element method and fast marching method for three-dimensional fatigue crack propagation*, Engineering Fracture Mechanics **70** (2003), 29–48.

[21] G. J. Wagner, N. Moës, W. K. Liu, and T. Belytschko, *The extended finite element method for rigid particles in stokes flow*, International Journal of Numerical Methods in Engineering **51** (2001), 293–313.

Received July 8, 2006.

BRYAN G. SMITH: b-smith7@northwestern.edu
ESAM Department, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, United States

BENJAMIN L. VAUGHAN JR.: b-vaughan@northwestern.edu
ESAM Department, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, United States

DAVID L. CHOPP: d-chopp@northwestern.edu
ESAM Department, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, United States

COMM. APP. MATH. AND COMP. SCI.
Vol. 2, No. 1, 2007

A LOCAL CORRECTIONS ALGORITHM FOR SOLVING POISSON'S EQUATION IN THREE DIMENSIONS

PETER MCCORQUODALE, PHILLIP COLELLA,
GREGORY T. BALLS AND SCOTT B. BADEN

We present a second-order accurate algorithm for solving the free-space Poisson's equation on a locally-refined nested grid hierarchy in three dimensions. Our approach is based on linear superposition of local convolutions of localized charge distributions, with the nonlocal coupling represented on coarser grids. The representation of the nonlocal coupling on the local solutions is based on Anderson's Method of Local Corrections and does not require iteration between different resolutions. A distributed-memory parallel implementation of this method is observed to have a computational cost per grid point less than three times that of a standard FFT-based method on a uniform grid of the same resolution, and scales well up to 1024 processors.

1. Introduction

We want to compute the solution to Poisson's equation on $\mathbb{R}^3$ with a charge distribution ρ with support on a compact set Ω. Specifically, we seek the solution ϕ to

$$\Delta\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = \rho(x, y, z) \tag{1}$$

that has the far-field behavior

$$\phi(\boldsymbol{x}) = -\frac{Q}{4\pi|\boldsymbol{x}|} + o\left(\frac{1}{|\boldsymbol{x}|}\right), \qquad |\boldsymbol{x}| \to \infty; \tag{2}$$

$$Q = \int_\Omega \rho(\boldsymbol{x}) d\boldsymbol{x}. \tag{3}$$

Using the maximum principle for harmonic functions, it is not difficult to show that equations (1)–(2) have a unique solution. This solution can be written as a convolution with the Green's function G [16]:

$$\phi(\boldsymbol{x}) = (G * \rho)(\boldsymbol{x}) \equiv \int G(\boldsymbol{x} - \boldsymbol{y})\rho(\boldsymbol{y})\, d\boldsymbol{y}\ , \ G(z) = -\frac{1}{4\pi|z|}. \tag{4}$$

Keywords: Poisson's equation, local corrections, domain decomposition, adaptive mesh refinement.

Solutions to (1) have a strong form of elliptic local regularity. If $D \subset \Omega$ is contained in a ball of radius r, then the function

$$\phi_D(\boldsymbol{x}) = \int_D G(\boldsymbol{x} - \boldsymbol{y})\rho(\boldsymbol{y})d\boldsymbol{y} \tag{5}$$

is real analytic at all points not contained in D, and its derivatives are rapidly decaying functions of $dist(\boldsymbol{x}, D)/r$. This suggests that an efficient method for computing the potential ϕ would be to compute local convolutions of the form (5) on a disjoint union of patches, and then compute the smooth global coupling among the patches using a calculation with a much coarser (and less computationally expensive) discretization. In fact, this is the underlying approach to all $O(N) - O(N \log N)$ methods for potential theory, include the Fast Multipole Method (FMM) [12] and the Method of Local Corrections (MLC) [2] for particles, and FFT-based methods [13], multigrid [7] and domain decomposition [20] for gridded data.

In principle, the same strategy should also lead to efficient parallel methods. The local convolutions are independent, and therefore can be performed in parallel on separate processors, while the nonlocal coupling between patches is representable by such a small number of degrees of freedom so as to have a negligible impact on the computational cost. For the particle methods such as FMM and MLC, this is indeed the case [3]. For algorithms for gridded data, particularly on structured and locally-structured grids, the results are mixed. FFT-based methods are probably optimal in terms of the number of floating point operations required, but are limited to uniform grids and require some form of global communication of all the data (such as transpose) or complex mappings of data onto processors. Multigrid iteration is applicable to locally-structured multiresolution grids [4; 1] and effectively exploits local regularity to reduce the number of floating point operations to a few hundred per grid point. However, it has an unacceptably high communication cost, with communication / synchronization steps required after each local relaxation step — that is, every few tens of floating point operations per grid point. Furthermore, there is so little computation being done between communication steps that the opportunity to overlap computation with communication is limited. The domain decomposition methods have typically led to iterative methods by constructing a dense linear system for the degrees of freedom on the boundaries between subdomains using a Schur complement. Such approaches reduce that communication load somewhat, but are still iterative, and for Poisson's equation are substantially more compute-intensive than multigrid or FFT-based methods.

A natural strategy is to apply the ideas developed for particle methods to gridded data. For FMM, this has been done in two dimensions [9; 8] by applying the fast multipole method directly to volume potentials on the grid, with methods that have a computational cost per grid point of less than three times that of an FFT on a

uniform grid, and furthermore have the locality of the FMM approach with respect to communication. However, the direct extension of that approach to three dimensions, while feasible, will not have the same absolute floating-point performance of a modest integer multiple of that of an FFT-based method, due to the substantially larger cost per grid point of the FMM method for computing volume potentials in 3D relative to that of 2D. To deal with that problem, one can take the approach of Greengard and Lee [11], in which local volume potentials on patches are computed using fast transform methods, with the FMM at the boundaries of patches to resolve the mismatch in the solutions at patch boundaries as well as the nonlocal coupling between patches. Using FMM only on two-dimensional surfaces might reduce the cost of that part of the calculation so as to make the overall floating-point cost, relative to FFT, more like that of the 2D FMM-based algorithms. However, such an approach has been carried out to date only in 2D.

The starting point for our approach is an extension of Anderson's MLC algorithm in two dimensions to gridded data in two dimensions [5; 6]. In this approach, local convolutions are computed using the James–Lackner method [14; 17] of representing infinite-domain boundary condition in terms of solutions to two Dirichlet problems on nested domains, plus a boundary-to-boundary convolution. The nonlocal coupling between patches is represented by solving a coarse grid problem and interpolating a correction back to the fine grid patches in a manner similar to full-approximation-storage multigrid. Unlike multigrid, though, the method is noniterative. In the present work, we generalize the method to locally-refined grids in three dimensions. A principal technical issue is the generalization to three dimensions of the James–Lackner method for computing local convolutions. We do this using a simplified FMM to compute the boundary-boundary convolutions, combined with FFT methods to compute the volume potentials. Thus, the method is similar in spirit to the approach of Greengard and Lee [11], but with different technical details.

2. Preliminaries

We represent both the potential field, ϕ, and the charge, ρ, on a discrete, three-dimensional Cartesian grid, with grid points spaced equally in all three directions by the same mesh spacing h. A triple of integers $\boldsymbol{i} = (i_x, i_y, i_z)$ indexes a point in real space $\boldsymbol{x_i} = (i_x h, i_y h, i_z h)$. Typically, our computational domain will be described in terms of unions of rectangular patches of the form $\Omega^h = [\boldsymbol{l}, \boldsymbol{u}]$, where $\boldsymbol{l}$ and $\boldsymbol{u}$ are the integer triples corresponding to the lower and upper corners of the region. Our grids are node-centered, with Ω^h representing the region in physical space $[\boldsymbol{l}h, \boldsymbol{u}h]$. Thus a union of rectangular patches representing a disjoint union of regions in physical space may have nonempty intersections in index space. We

define a coarsening operator $\mathscr{C}$ as

$$\mathscr{C}(\Omega^h, C) = [\lfloor \boldsymbol{l}/C \rfloor, \lceil \boldsymbol{u}/C \rceil] \tag{6}$$

where the operators $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ round down and up to the nearest integer, respectively. We also need the *grow* operation $\mathscr{G}$, which extends or shrinks an index domain by a uniform amount in each direction:

$$\mathscr{G}(\Omega^h, p) = [\boldsymbol{l} - (p, p, p), \boldsymbol{u} + (p, p, p)]. \tag{7}$$

When $p < 0$, $\mathscr{G}$ returns a shrunken domain. We denote by $\partial\Omega^h$ the set of boundary points of Ω^h:

$$\partial\Omega^h = \Omega^h - \mathscr{G}(\Omega^h, -1). \tag{8}$$

A field ψ is represented on this discrete grid by ψ^h such that

$$\psi^h_{\boldsymbol{i}} \approx \psi(\boldsymbol{x}_{\boldsymbol{i}}). \tag{9}$$

We can also define a sampling operator $\mathscr{S}$ that projects a discrete field on Ω^h onto $\mathscr{C}(\Omega^h, C)$:

$$\mathscr{S}(\psi^h, C)_{\boldsymbol{i}} = \psi^h_{C\boldsymbol{i}}\ , \boldsymbol{i} \in \mathscr{C}(\Omega^h, C).$$

We denote by χ^h a discrete characteristic function defined as follows. For an interval $[l, u]$ with l and u integers, the function $\chi^h_{[l,u]}$ on the real line has the value 1 in the interval (l, u), 0 outside $[l, u]$, and $\frac{1}{2}$ at l and u. Then for a box $B = [\boldsymbol{l}, \boldsymbol{u}]$, the function χ^h_B is defined on index space as the product of interval functions over all dimensions: $\chi^h_B = \chi^h_{[\mathtt{l}_x, \mathtt{u}_x]}\chi^h_{[\mathtt{l}_y, \mathtt{u}_y]}\chi^h_{[\mathtt{l}_z, \mathtt{u}_z]}$. We also define a characteristic function χ for the corresponding region in $\mathbb{R}^3$,

$$\chi_B(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in [\boldsymbol{l}h, \boldsymbol{u}h]\ ; \\ 0, & \text{otherwise.} \end{cases}$$

We use a discretization of the Laplacian operators with the stencil points contained in a three-by-three block surrounding the evaluation point:

$$(\Delta^h \phi^h)_{\boldsymbol{i}} = \frac{1}{h^2} \sum_{\boldsymbol{j} \in \{-1,0,1\}^3} a_{\|\boldsymbol{j}\|} \phi^h_{\boldsymbol{i}+\boldsymbol{j}} \tag{10}$$

where $\|\boldsymbol{j}\|$ is the number of nonzero components in $\boldsymbol{j} \in \{-1, 0, 1\}^3$ and falls in the range $\{0, 1, 2, 3\}$. We shall use the 19-point Mehrstellen operator, specified by $a_0 = -4$, $a_1 = \frac{1}{3}$, $a_2 = \frac{1}{6}$, $a_3 = 0$. If $\phi^{\text{exact},h}$ is the exact solution evaluated at grid points, and the truncation error, $\tau^h_{\boldsymbol{j}}$, is defined as

$$\tau^h_{\boldsymbol{j}} = \rho^h_{\boldsymbol{j}} - (\Delta^h \phi^{\text{exact},h})_{\boldsymbol{j}}, \tag{11}$$

then we can use Taylor expansion, along with the fact that $\Delta^2\phi = \Delta\rho$, to determine that

$$\tau_{\boldsymbol{j}}^h = \rho_{\boldsymbol{j}}^h - (\Delta^h \phi^{\text{exact},h})_{\boldsymbol{j}} = \frac{-h^2}{12}\Delta\rho + O(h^4). \tag{12}$$

Thus a solution to the system

$$\Delta^h \phi^h = \rho^h \ , \ (\rho^h)_{\boldsymbol{j}} = \rho(\boldsymbol{j}h) \tag{13}$$

is second-order accurate: $\phi_{\boldsymbol{j}}^h = \phi_{\boldsymbol{j}}^{\text{exact},h} + O(h^2)$. The particular form of the truncation error in (12) leads to a strong localization of the $O(h^2)$ error: if $\boldsymbol{j}h$ is contained in the complement of the closure of the support of ρ, then it is not difficult to show that $\phi_{\boldsymbol{j}}^h = \phi_{\boldsymbol{j}}^{\text{exact},h} + O(h^4)$ [6]. More classically, one can also precondition the charge and solve

$$\Delta^h \phi^{*,h} = \rho^{*,h} = \rho^h + \frac{h^2}{12}\widetilde{\Delta}^h \rho^h, \tag{14}$$

where $\widetilde{\Delta}^h$ is any second-order accurate discretization of the Laplacian, to obtain a solution that is $O(h^4)$ everywhere. With infinite-domain boundary conditions, it is also possible to make a Mehrstellen correction to the solution *after* solving (13):

$$\phi^h := \phi^h + \frac{h^2}{12}\rho^h. \tag{15}$$

3. Convolutions on bounded domains

A basic component of our method of local corrections is a single-grid solver for Poisson's equation with infinite-domain boundary conditions. We follow the approach used for the 2D problem by James [14] and Lackner [17].

Let Ω be the support of the right-hand side ρ in (1). Clearly, we can represent the solution to (1)–(2) on Ω in terms of solutions of Poisson's equation with Dirichlet boundary conditions on a slightly larger domain, where the boundary conditions are computed using the convolution operator (4). We can reduce the convolution to a boundary-boundary convolution by solving an additional Dirichlet problem. Let Ω_1 and Ω_2 contain Ω with $\Omega_2 \supset \Omega_1 \supset \Omega$. Let ϕ^1 be the solution to

$$\Delta\phi^1 = \rho \text{ on } \Omega_1 \ ; \ \phi^1 = 0 \text{ on } \partial\Omega_1$$

and define a boundary charge distribution q

$$q \equiv \frac{\partial\phi^1}{\partial \boldsymbol{n}} \text{ on } \partial\Omega_1$$

where $\boldsymbol{n}$ is the unit outward normal. Then the boundary potential ϕ_B induced by q

$$\phi_B(\boldsymbol{x}) = \int_{\partial\Omega_1} G(\boldsymbol{x}-\boldsymbol{y})q(\boldsymbol{y})\,dA_{\boldsymbol{y}} \tag{16}$$

is a solution to Laplace's equation on $\mathbb{R}^3 - \partial\Omega_1$, and satisfies the jump relations $[\phi_B] = 0$, $[\frac{\partial \phi_B}{\partial \boldsymbol{n}}] = -q$ on $\partial\Omega_1$. Thus the function ϕ given by

$$\phi = \begin{cases} \phi_1 + \phi_B, & \text{on } \Omega_1; \\ \phi_B & \text{elsewhere} \end{cases}$$

is a solution to (1)–(2). In particular, ϕ is a solution to the Dirichlet problem

$$\Delta\phi = \rho \text{ on } \Omega_2 \; ; \; \phi = \phi_B \text{ on } \partial\Omega_2$$

for any $\Omega_2 \supset \Omega_1$. Note that the calculation of the Dirichlet boundary conditions requires only the convolution of the Green's function with the boundary charge q.

We use the representation described above to compute an approximation of the convolution (4). We assume that Ω is a cube, which we discretize to obtain the discrete domain Ω^h with mesh spacing h and containing $(N+1)^3$ points. We also define discrete domains $\Omega_1^h = \mathcal{G}(\Omega^h, s_1)$ and $\Omega_2^h = \mathcal{G}(\Omega^h, s_1 + s_2)$, for some $s_1, s_2 \geq 0$.

The 3D James–Lackner algorithm.

Step 1. Solve the homogeneous Dirichlet problem on Ω_1:

$$\Delta^h \phi_1^h = \rho \text{ on } \mathcal{G}(\Omega_1^h, -1) \; ; \; \phi_1^h = 0 \text{ on } \partial\Omega_1^h$$

and compute the discrete boundary charge $q_{\boldsymbol{i}} = D_B(\phi_1^h)_{\boldsymbol{i}}$, $\boldsymbol{i} \in \partial\Omega_1^h$. We use a fourth-order one-sided difference approximation of the normal derivative for D_B, e.g.,

$$D_B(f)_{0,j,k} = \frac{-25 f_{0,j,k} + 48 f_{1,j,k} - 36 f_{2,j,k} + 16 f_{3,j,k} - 3 f_{4,j,k}}{12h}.$$

Step 2. Given the discrete charge distribution q on $\partial\Omega_1$, compute an approximation to the convolution integral (16) to obtain $g_{\boldsymbol{i}} \approx \phi_B(\boldsymbol{i}h)$ for $\boldsymbol{i} \in \partial\Omega_2^h$.

Step 3. Solve the inhomogeneous Dirichlet problem on Ω_2:

$$\Delta^h \phi^h = \rho \text{ on } \mathcal{G}(\Omega_2^h, -1) \; ; \; \phi^h = g \text{ on } \partial\Omega_2^h.$$

The solution of the Dirichlet problems in Steps 1 and 3 can be done in $O(N^3 \log N)$ operations using a fast discrete sine transform to diagonalize Δ^h. Step 2 is performed using a fast multipole method that takes advantage of the fact that the charge is defined on a union of planar surfaces:

$$\partial\Omega_1 = \Omega_1(+,0) \cup \Omega_1(-,0) \cup \Omega_1(+,1) \cup \Omega_1(-,1) \cup \Omega_1(+,2) \cup \Omega_1(-,2)$$

where $\Omega_1(+, d)$ and $\Omega_1(-, d)$ are respectively the high and low faces of Ω_1 in which coordinate $d \in \{0, 1, 2\}$ is fixed. Then the integral in (16) can be split up as

$$\phi_B(\boldsymbol{x}) = \Phi^{+,0}(\boldsymbol{x}) + \Phi^{-,0}(\boldsymbol{x}) + \Phi^{+,1}(\boldsymbol{x}) + \Phi^{-,1}(\boldsymbol{x}) + \Phi^{+,2}(\boldsymbol{x}) + \Phi^{-,2}(\boldsymbol{x}) \quad (17)$$

where $\Phi^{\pm,d}$ is the contribution from face $\Omega_1(\pm, d)$:

$$\Phi^{\pm,d}(\boldsymbol{x}) = \int_{\Omega_1(\pm,d)} G(\boldsymbol{x} - \boldsymbol{y}) q(\boldsymbol{y})\, dA_{\boldsymbol{y}}, \quad (18)$$

which is a planar integral. Step 2, then, can be broken down as follows.

2a. Split each face $\Omega_1^h(\pm, d)$ into patches of dimensions $r \times r$ centered at points on the face coarsened by r, where r is divisible by 4. Then calculate the multipole moments up to order M of q^h on each patch. For the patch on the face $\Omega_1^h(-, 2)$ that is centered at the point $(i_0, i_1, -s_1/r)$ in r-coarsened coordinates, the (p_0, p_1) moment is

$$A_{i_0,i_1}^{p_0,p_1,-,2} = \sum_{-r/2 \le j_0 \le r/2} \sum_{-r/2 \le j_1 \le r/2} w_{j_0} w_{j_1} q_{(ri_0+j_0, ri_1+j_1, -s_1)} (j_0 h)^{p_0} (j_1 h)^{p_1}$$
$$(0 \le p_0 + p_1 \le M; \quad p_0, p_1 \ge 0) \quad (19)$$

where the w_j are the weights from Boole's rule of integration, which is $O(h^6)$ accurate:

$$w_j = \begin{cases} \frac{14}{45} & \text{if } |j| = \frac{r}{2}; \\ \frac{28}{45} & \text{if } \frac{r}{2} + j \equiv 0 \mod 4 \text{ and } |j| < \frac{r}{2}; \\ \frac{64}{45} & \text{if } j \text{ is odd;} \\ \frac{8}{15} & \text{if } \frac{r}{2} + j \equiv 2 \mod 4. \end{cases} \quad (20)$$

The moments for the other faces are computed analogously.

2b. On each face of $\partial\Omega_2^h$ coarsened by r in each dimension, plus a layer of coarse points of width P, add up the evaluations $\Phi^{\pm,d}$ of multipole expansions due to all patches of all faces of $\partial\Omega_1^h$. As an example,

$$\Phi^{-,2}(\vec{x}) = \sum_{i_0,i_1} \sum_{p_0,p_1} A_{i_0,i_1}^{p_0,p_1,-,2} \times \frac{(-1)^{p_0+p_1}}{p_0! p_1!} \frac{\partial^{p_0+p_1} G}{\partial z_0^{p_0} \partial z_1^{p_1}} (x_0 - i_0 rh, x_1 - i_1 rh, x_2 + s_1 h) \quad (21)$$

using two-dimensional Taylor expansions of the Green's function G around the points $(x_0 - i_0 rh, x_1 - i_1 rh, x_2 + s_1 h)$. The indices i_0, i_1 in the sum are over indices of coarse points on the face $\Omega_1^h(-, 2)$, and $p_0, p_1 \ge 0$; $p_0 + p_1 \le M$.

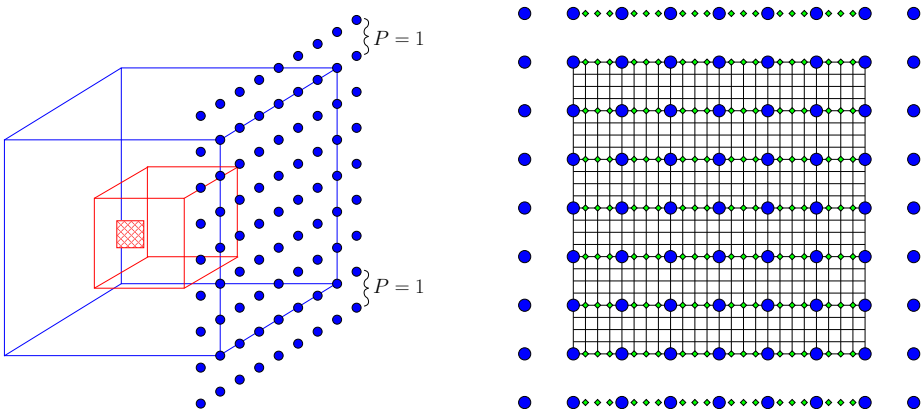


Figure 1. In Step 2 of our implementation of the 3D James–Lackner algorithm, multipole moments are calculated for each patch on each face of $\partial\Omega_1^h$, such as the patch shown cross-hatched in red. The multipole expansions are then evaluated at the coarse points on the faces of Ω_2^h augmented by an additional layer of width P, indicated with blue circles for one face. These evaluations are interpolated to all the fine points on the faces of $\partial\Omega_2^h$, located at intersections of the black lines, using two passes. The evaluation points of the first pass are shown as small green diamonds.

2c. On each face of $\partial\Omega_2^h$, interpolate from the coarse points to the remaining fine points on the face, using a tensor product of Lagrange interpolating polynomials as illustrated in Figure 1.

Choosing $r \approx \sqrt{N}$ provides sufficient accuracy for the solution and allows the integration step to be completed in $O((M^2+P)N^2)$ work. For $O(h^4)$ error, we set $M = 7$ and $P = 3$, and these are independent of N. Hence Step 2 requires $O(N^2)$ work.

We also note constraints required on s_2, the spacing between Ω_1^h and Ω_2^h. Convergence requirements of the multipole method force us to choose s_2 with care. In order for the multipole expansions from a patch to converge, the distance from the center of a patch on a face of Ω_1^h to the points on the faces of Ω_2^h, on which the expansion is evaluated, should be at least twice the radius of the patch. Here we define the radius of a patch as the maximum distance from the patch center to any point on the patch. Recall that we chose our patches to be $r \times r$ fine grid points. Thus our patches have a radius of $rh/\sqrt{2}$, and the distance requirement becomes $s_2 h \geq 2rh/\sqrt{2}$. We also need the number of cells along the length of Ω_2^h to be divisible by r. Combining these two requirements, we arrive at the following

formula for s_2:

$$s_2 = \frac{r}{2}\left\lceil 2\sqrt{2} + \frac{N+2s_1}{r} \right\rceil - \frac{N+2s_1}{2}. \tag{22}$$

For efficiency, both s_1 and s_2 should be as small as possible. If the distance of the support of ρ to $\partial\Omega_1$ is nonzero, then we can set $s_1 = 0$.

We now examine the computational costs in the single-grid solver, listing the operation counts for each step:

1. FFT-based Poisson solver on Ω_1^h: $O(N^3 \log N)$.
   Normal derivatives on faces of Ω_1^h: $O(N^2)$.
2. Integration to boundary conditions on faces of Ω_2^h using FMM: $O(N^2)$.
3. FFT-based Poisson solver on Ω_2^h: $O(N^3 \log N)$.

Thus the single-grid infinite-domain solver operation count is bounded by the fast Poisson solves that use Dirichlet boundary conditions, and the overall computational cost of an infinite-domain solution is $O(N^3 \log N)$.

4. Method of local corrections

The domain decomposition algorithm described here is the finite-difference analogue [6] of Anderson's Method of Local Corrections (MLC) [2], extended to locally-refined nested grids in three dimensions. To simplify the presentation, we describe the MLC algorithm on two levels. We use a fine-grid discretization Ω^h corresponding to a rectangular domain Ω that contains the support of the charge ρ. Within Ω^h we have a set of cubic patches Ω_k^h of equal size that overlap only at patch boundaries. These subdomains make up a region on which the charge is finely resolved. For each patch, the charge ρ_k^h is defined on Ω_k^h. Our method entails solving local problems on each of the Ω_k^h in parallel, as well as on a single coarse global mesh Ω^H. The spacing of the coarse mesh is $H = Ch$, where C is a specified *coarsening factor*.

Because our meshes are node-centered, the points of Ω^H map directly onto corresponding points in Ω^h, and no averaging is required to coarsen the mesh data. Thus, we can coarsen the mesh by sampling the mesh without having to interpolate. In particular, we coarsen a fine grid representation using the *sample* operator $\mathcal{S}^H$: for each point $\boldsymbol{x}_C$, we can find the coarse grid value $\psi^H(\boldsymbol{x}_C)$ (where ψ^H has grid spacing H) by finding the fine grid point $\boldsymbol{x}$ at the corresponding position in ψ^h (with grid spacing $h = H/C$):

$$\psi^H(\boldsymbol{x}_C) = (\mathcal{S}^H(\psi^h))(\boldsymbol{x}/C) = \psi^h(\boldsymbol{x}). \tag{23}$$

If $\rho = \rho(\boldsymbol{x})$ is the continuous charge, we set the discrete coarse-level charge ρ^H on Ω^H and fine-level charge ρ_k^h on each Ω_k^h to be

$$(\rho^H)_{\boldsymbol{i}} = \rho(\boldsymbol{i}H)\ ,\ \boldsymbol{i} \in \Omega^H;$$
$$(\rho_k^h)_{\boldsymbol{i}} = (\chi^h_{\Omega_k^h})_{\boldsymbol{i}}\ \rho(\boldsymbol{i}h)\ ,\ \boldsymbol{i} \in \Omega_k^h.$$

The algorithm has three computational steps interspersed by two communication steps.

Method of Local Corrections.

1. INITIAL LOCAL SOLUTION. Using the 3D James–Lackner algorithm, calculate a local infinite-domain solution on each local subdomain, Ω_k^h, augmented with an overlap region:

$$\Delta^h \phi_k^{h,\text{init}} = \rho_k^h \text{ on } \mathcal{G}(\Omega_k^h, s + Cb) \tag{24}$$

and construct a coarsened version of the solution, $\phi_k^{H,\text{init}}$, by sampling:

$$\phi_k^{H,\text{init}} = \mathcal{S}^H(\phi_k^{h,\text{init}}) \text{ on } \mathcal{G}(\Omega_k^H, s/C + b). \tag{25}$$

Here s is a correction radius, C is the coarsening factor, and b is the width of a layer for polynomial interpolation to be used in step 3.

2. GLOBAL COARSE SOLUTION. Couple the individual local solutions by solving another Poisson equation on a coarsened mesh covering the entire domain. First construct coarsened local charge fields:

$$R_k^H = \begin{cases} \Delta^H \phi_k^{H,\text{init}} & \text{on } \mathcal{G}(\Omega_k^H, s/C - 1); \\ 0 & \text{elsewhere} \end{cases} \tag{26}$$

and sum these charge fields to form a global coarse representation of the charge:

$$R^H = \sum_k R_k^H. \tag{27}$$

Then solve

$$\Delta^H \phi^H = R^H + (1 - \sum_k \chi_{\Omega_k^H})\rho^H \text{ on } \mathcal{G}(\Omega^H, s/C + b) \tag{28}$$

with infinite-domain boundary conditions, using the 3D James–Lackner algorithm. For this solve, we take the base domain to be $\mathcal{G}(\Omega^H, 2\lceil (s/C - 1)/2 \rceil)$ because the length must be divisible by 4.

3. FINAL LOCAL SOLUTION. Solve

$$\Delta^h \phi_k^h = \rho_k^h \text{ on } \Omega_k^h \tag{29}$$

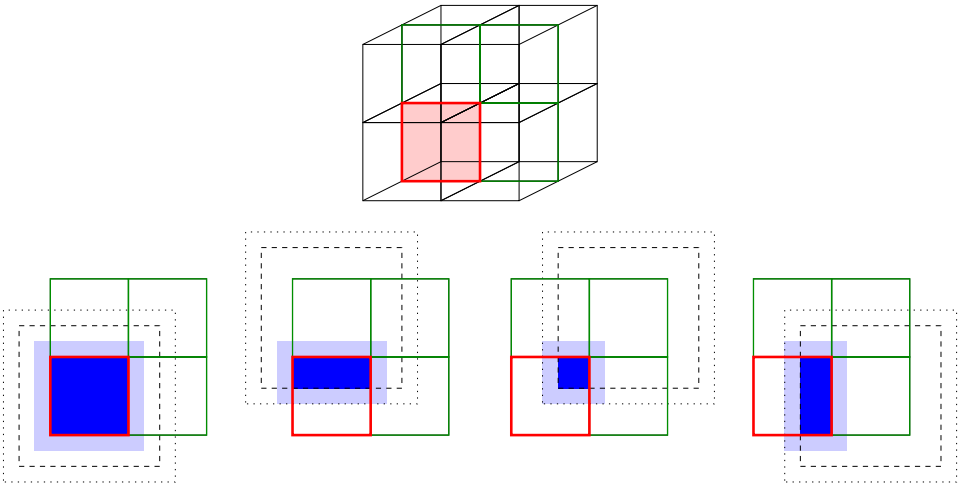


Figure 2. Setting boundary values for a final local solution in step 3 of MLC. For the face shaded red in the top of the figure, in a layout of eight cubes, the lower diagrams depict the regions from which data are copied from faces of different neighboring boxes. Solid lines indicate the boundaries of the boxes $\Omega_{k'}^h$, dashed lines the boundaries of the boxes $\mathcal{G}(\Omega_{k'}^h, s)$, and dotted lines the boundaries of the boxes $\mathcal{G}(\Omega_{k'}^h, s + Cb)$. Fine-grid data are copied to the red face from the nodes inside and on the edges of the regions shaded dark blue. Coarse grid data are copied from nodes inside and on the edges of the regions shaded both dark and light blue, and then interpolated to nodes on the red face that are inside and on the edges of the regions shaded dark blue.

with Dirichlet boundary conditions on $\partial\Omega_k^h$:

$$\phi_k^h(\boldsymbol{x}) = \sum_{k':\boldsymbol{x}\in\mathcal{G}(\Omega_{k'}^{h,\text{init}},s)} \phi_{k'}^{h,\text{init}}(\boldsymbol{x}) + \mathcal{I}(\phi^{H,\text{corr}}) \tag{30}$$

where $\mathcal{I}$ is the same interpolation operator used in step 2c. of the single-grid infinite-domain Poisson solver (setting the layer width P to b), and

$$\phi^{H,\text{corr}} = \phi^H(\boldsymbol{x}) - \sum_{k':\boldsymbol{x}\in\mathcal{G}(\Omega_{k'}^h,s)} \phi_{k'}^{H,\text{init}}(\boldsymbol{x}), \tag{31}$$

which is the global coarse solution with the local contribution subtracted. Figure 2 depicts the regions from which data are taken to set boundary conditions on a face.

Finally, we apply the Mehrstellen correction (15) to the solution.

For $O(h^2)$ accuracy of the method, we set $b = 2$ and $s = 2C$.

4.1. ***Separating the monopole component.*** In order to minimize the cost of the infinite-domain solution, we would like to set s_1, the amount by which we grow the domain in the initial Dirichlet solution for the James–Lackner algorithm, to be zero. In the present application, the charge on each patch is nonzero all the way out to the boundary, so that the conditions under which this would be valid do not hold. In particular, for the fixed-size patches (relative to the mesh spacing) we are using here, this leads to an $O(1)$ relative error in the monopole component of the field used to compute the boundary conditions for the second Dirichlet solution. We eliminate this error by separating out the monopole component on each patch, and treating it exactly.

Specifically, for a given patch B, we compute $\bar{\rho}$, the mean of ρ over B, and subtract $\bar{\rho}\chi_B^h$ from the right-hand side of (1) before solving, then add $\bar{\rho}\xi_B$ to the solution, where $\xi_B \equiv G * \chi_B$ is computed analytically and stored.

In the initial local solve, we replace (24) by

$$\Delta^h \widetilde{\phi}_k^{h,\text{init}} = \rho_k^h - \bar{\rho}_k^h \chi_{\Omega_k^h}^h \text{ on } \mathcal{G}(\Omega_k^h, s + Cb) \tag{32}$$

and then sample the solution:

$$\widetilde{\phi}_k^{H,\text{init}} = \mathcal{S}^H(\widetilde{\phi}_k^{h,\text{init}}) \text{ on } \mathcal{G}(\Omega_k^H, s/C + b). \tag{33}$$

The updated solutions are

$$\phi_k^{h,\text{init}} = \widetilde{\phi}_k^{h,\text{init}} + \bar{\rho}_k^h \xi_{\Omega_k^h}^h; \tag{34}$$

$$\phi_k^{H,\text{init}} = \widetilde{\phi}_k^{H,\text{init}} + \bar{\rho}_k^h \xi_{\Omega_k^H}^H. \tag{35}$$

In forming the right-hand side for the global coarse solve, we replace (26) by

$$R_k^H = \begin{cases} \Delta^H \widetilde{\phi}_k^{H,\text{init}} + \bar{\rho}_k^h \chi_{\Omega_k^H}^H & \text{on } \mathcal{G}(\Omega_k^H, s/C - 1); \\ 0 & \text{elsewhere.} \end{cases}$$

In the final local Dirichlet solves, we replace (29) by

$$\Delta^h \widetilde{\phi}_k^h = \rho_k^h - \bar{\rho}_k^h \chi_{\Omega_k^h}^h \text{ on } \Omega_k^h \tag{36}$$

and the Dirichlet boundary conditions (30) by

$$\widetilde{\phi}_k^h(\boldsymbol{x}) = \sum_{k' : \boldsymbol{x} \in \mathcal{G}(\Omega_{k'}^{h,\text{init}}, s)} \phi_{k'}^{h,\text{init}}(\boldsymbol{x}) + \mathcal{I}(\phi^{H,\text{corr}}) + \bar{\rho}_k^h \xi_{\Omega_k^h}^h(\boldsymbol{x}). \tag{37}$$

Finally, we have the solution

$$\phi_k^h = \widetilde{\phi}_k^h + \bar{\rho}_k^h \xi_{\Omega_k^h}^h. \tag{38}$$

4.2. ***Extending to more than two levels.*** In extending the MLC solver from two levels to three, we assume a hierarchical nesting of patches, such that each fine-level patch is contained in one and only one middle-level patch. We run the two-level MLC solver separately within each middle-level patch, except that we perform the global coarse solve (28) on the middle-level patches with a two-level MLC solver using all the middle-level patches and the domain of the coarsest level.

The MLC solvers between the middle and fine levels require an expansion of the middle-level patches, specifically by $2\lceil (s/C-1)/2\rceil$, taking s and C to be respectively the correction radius and coarsening ratio between these two levels. Then in the MLC solver between the middle and coarse levels, the finer-level patches Ω_k^h will overlap by this amount.

We may similarly extend to an arbitrary number of levels. In our implementation of the MLC solver on three levels, in order to retain accuracy we set $b=2$ and a larger buffer size $s=4C$ (instead of $s=2C$) relating the coarse and the middle levels where C is the coarsening factor between middle and coarse levels. Between the fine and middle levels, we retain buffer size $s=2C$ where C is the coarsening factor between these levels.

5. Results

As an example, we use right-hand sides built from ρ_m^{osc}, a spherically symmetric function with high-wavenumber component:

$$\rho_m^{\text{osc}}(r)=\begin{cases}((r-r^2)\sin(2m\pi r))^2, & \text{if } r<1\ ;\\ 0, & \text{if } r\geq 1\ .\end{cases}$$

The wavelength of ρ_m^{osc} is $1/(2m)$. If we set $\alpha=4m\pi$, then the integral of ρ_m^{osc} over space is

$$\int \rho_m^{\text{osc}}\,dV=\pi\Big(\frac{2}{105}+\frac{48}{\alpha^4}-\frac{1440}{\alpha^6}\Big),$$

and the exact solution of

$$\Delta\phi_m^{\text{osc}}=\rho_m^{\text{osc}}$$

with infinite-domain boundary conditions is

$$\phi_m^{\text{osc}}(r)=\begin{cases}r^6/84-r^5/30+r^4/40+\\ \quad 60/\alpha^6-9/\alpha^4-1/120+120/(\alpha^6 r)+\\ \quad (-120/(\alpha^6 r)-9/\alpha^4+300/\alpha^6+36r/\alpha^4+r^2/(2\alpha^2)\\ \quad -30r^2/\alpha^4-r^3/\alpha^2+r^4/(2\alpha^2))\cos(\alpha r)+\\ \quad (12/(\alpha^5 r)-360/(\alpha^7 r)-96/\alpha^5+120r/\alpha^5\\ \quad -3r/\alpha^3+8r^2/\alpha^3-5r^3/\alpha^3)\sin(\alpha r) & \text{if } r<1;\\ (-1/210-12/\alpha^4+360/\alpha^6)/r & \text{if } r\geq 1.\end{cases}$$

This solution is negative and has its minimum value at the origin:

$$\phi_m^{\text{osc}}(0) = -\frac{1}{120} - \frac{6}{\alpha^4}.$$

We test with three different charge densities on the unit cube $[0, 1]^3$, with m set to either 7, 15, or 30, and $R = 0.05$ in

$$\rho(\boldsymbol{x}) = \frac{1}{R^3}\left(\rho_m^{\text{osc}}(|\boldsymbol{x} - \boldsymbol{c}_1|/R) + \rho_m^{\text{osc}}(|\boldsymbol{x} - \boldsymbol{c}_2|/R) + \rho_m^{\text{osc}}(|\boldsymbol{x} - \boldsymbol{c}_3|/R)\right), \quad (39)$$

where $\boldsymbol{c}_1 = (\frac{3}{16}, \frac{7}{16}, \frac{13}{16})$, $\boldsymbol{c}_2 = (\frac{7}{16}, \frac{13}{16}, \frac{3}{16})$, and $\boldsymbol{c}_3 = (\frac{13}{16}, \frac{3}{16}, \frac{7}{16})$. This is a superposition of three disjoint spherical charge distributions. The wavelength is $\lambda = R/(2m) = 1/(40m)$. The solution, which is negative, attains its minimum value at the sphere centers,

$$\phi^{\text{exact}}(\boldsymbol{c}_1) = \phi^{\text{exact}}(\boldsymbol{c}_2) = \phi^{\text{exact}}(\boldsymbol{c}_3) = \left(-\frac{1}{120} - \frac{6}{\alpha^4}\right)/R + \left(-\frac{1}{105} - \frac{24}{\alpha^4} + \frac{720}{\alpha^6}\right)/D,$$

where $D = |\boldsymbol{c}_1 - \boldsymbol{c}_2| = |\boldsymbol{c}_1 - \boldsymbol{c}_3| = |\boldsymbol{c}_2 - \boldsymbol{c}_3|$ is the distance between any two sphere centers.

Our example uses three levels of boxes shown in Figure 3, with a coarsening factor of 4 between adjacent levels. The boxes are as follows.

- Fine level: all boxes are cubes of length 32. If the whole domain is split into $512 = 8^3$ subdomains of length $\frac{1}{8}$, then three of these subdomains contain the support of the charge; these subdomains are then fully refined with fine-level boxes.
- Middle level: all boxes are cubes of length 32 (becoming 36 after expansion, as described in Section 4.2). Boxes at this level cover the three subdomains with the support of the charge, plus an additional layer of boxes.
- Coarse level: these boxes cover the entire domain and are parallel slabs in one direction. The number of slabs is the domain length in coarse cells divided by 4, or the number of processors, whichever is less.

5.1. ***Convergence results.*** In reporting our convergence results, we show the max norms and L^2 norms of solution error (difference between calculated solution and exact solution) normalized by the max norm of the exact solution. We also show the L^2 norm of the error on the finest grids alone. (For all the cases discussed here, the max norm on the finest grids is equal to the max norm on the whole domain.) We also calculate a convergence rate, p, defined such that if ϵ_f and ϵ_c are the norms of the solution error with mesh spacings h_f and h_c, respectively, then

$$p = \log_2 \frac{\epsilon_f}{\epsilon_c} \Big/ \log_2 \frac{h_f}{h_c}. \quad (40)$$

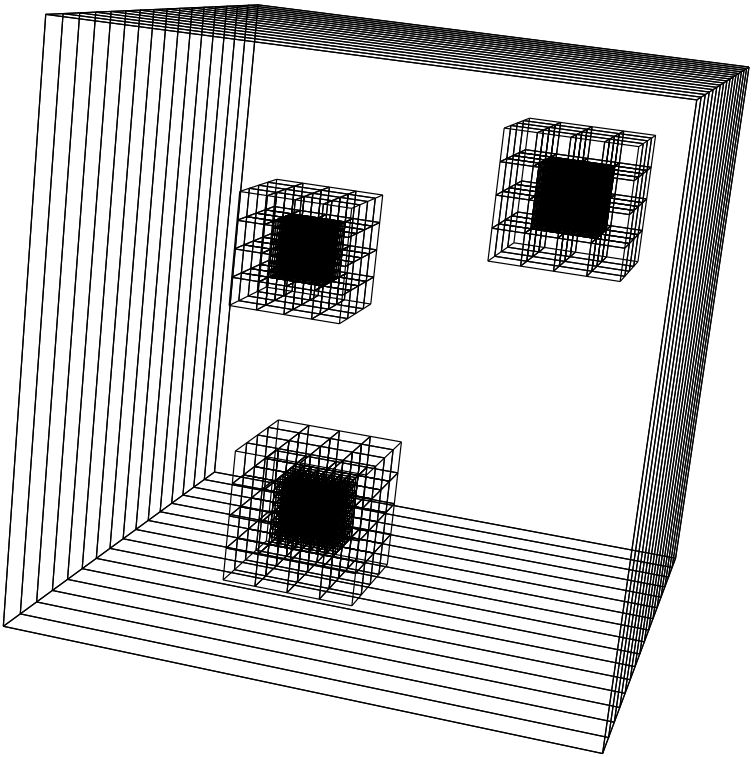


Figure 3. Boxes in three-level solve used in example. All boxes at the fine and middle levels have dimensions $32 \times 32 \times 32$. The coarse level is split into slabs across processors.

See Table 1 for convergence results with m set to 7, 15, and 30, in the example with three-level MLC separating monopole solutions. The tables show the fine-level mesh spacing h and norms of the normalized solution error ϵ^h, which is the difference between calculated solution and exact solution, divided by $\|\phi^{\text{exact}}\|_\infty$, when the finest-level mesh spacing is h. While overall the solution error is $O(h^2)$, there is considerable variation in the rates, depending on the norm used and the grid resolution. This variation is not surprising, given the fact that there are multiple parameter choices for the method that correspond to different asymptotic contributions to the error. The local James–Lackner computations have a contribution to the error that is $O(h^2)$ coming from the choice of multipole parameters, while the local truncation error for the Mehrstellen operator is $O(h^4)$, since we are applying the Mehrstellen correction in the form of (15). Finally, the choice of $b = 2$ in the boundary interpolation (30) for the final local solution step (29) corresponds to an error that is formally $O(h^6)$, although in this case, the contribution to the solution that is being interpolated is not sufficiently smooth to justify such an error estimate.

In fact, the choice $b = 2$ was made empirically, with that choice leading to the most uniform convergence behavior. Such empirical choices are a weakness in the algorithm, and one that we intend to correct in future work.

Table 2 shows solution error on the three-level example with $m = 7$ when it is run *without* separating the monopole solution. This does not converge in L^2 norm, and has very poor convergence in max norm, thus illustrating the need for separating the monopole contribution to the solution.

Table 3 shows convergence results for the same examples (m set to 7, 15, and 30) with boxes at only two levels of refinement instead of three. The boxes at the fine level are the same as in the three-level arrangement, but the middle level is removed, and there is a coarsening factor of 4 from the fine level to a coarse level covering the full domain and split into parallel slabs. The mesh spacing at the coarser level in the two-level arrangement is the same as that of the middle level in the three-level arrangement. Comparing the two-level results in Table 3 and the three-level results in Table 1, with finest-level mesh spacings of $h = 1/2048$ and $h = 1/4096$, we see that the increased accuracy in the two-level calculation is worth approximately a factor of two in mesh spacing, with the difference decreasing as the wavenumber m increases. For the $m = 30$ cases, the three-level computation has essentially the same error as the two-level computation at the same fine-grid resolution. As m decreases to 15 and 7, the error of the two-level calculation becomes much smaller than that of the three-level calculation. This is consistent with the observation that there are two competing sources of error: that induced by the local truncation error, which for a fixed h scales like m^2; and that coming from the error in the representation of far-field effects, which is only weakly dependent on m. Thus as m decreases from $m = 30$, the contribution of the local truncation error rapidly decreases, leaving only the contribution from the error in the representation of the far-field effects. These are more accurately represented by a single-level calculation than by a two-level calculation at the same resolution. Nonetheless, we shall see below that, in these cases, the two-level and three-level calculations provide roughly the same accuracy for a given computational cost.

We also ran the problem on different sizes of a *single* grid with the James–Lackner solver of Section 2 and Mehrstellen preconditioning (14). The results on the left side of Table 4 show solution error converging in max norm at a rate that is fourth order in the mesh spacing, as long as the oscillating right-hand side is resolved sufficiently. Nonetheless, the accuracy of the Mehrstellen method, by itself, is insufficient to make up for the lack of resolution in the coarsest-level calculation, so that the MLC method on the locally-refined grids substantially increases the accuracy of the overall solution. This is demonstrated in Table 4 by listing, beside the Mehrstellen result, the max norm error of the three-level MLC result whose coarsest level has the same mesh spacing as that of the Mehrstellen result.

m	h	$\|\epsilon^h_{\text{all}}\|_\infty$	p	$\|\epsilon^h_{\text{fine}}\|_2$	p	$\|\epsilon^h_{\text{all}}\|_2$	p	λ/h
7	1/2048	2.132 E−5		1.632 E−7		1.738 E−7		7.31
7	1/4096	4.735 E−6	2.17	2.379 E−8	2.78	4.712 E−8	1.88	14.63
7	1/8192	1.130 E−6	2.07	5.720 E−9	2.06	8.419 E−9	2.48	29.26
15	1/2048	2.437 E−5		2.009 E−7		2.357 E−7		3.41
15	1/4096	4.906 E−6	2.31	2.642 E−8	2.93	3.061 E−8	2.95	6.83
15	1/8192	1.157 E−6	2.08	6.648 E−9	1.99	9.737 E−9	1.65	13.65
30	1/2048	5.022 E−5		3.798 E−7		3.848 E−7		1.71
30	1/4096	5.274 E−6	3.25	3.795 E−8	3.32	6.296 E−8	2.61	3.41
30	1/8192	1.542 E−6	1.77	7.593 E−9	2.32	1.270 E−8	2.31	6.83

Table 1. Norms and convergence rates of solution error with adaptive three-level MLC separating monopole solutions, for example with $m = 7$, 15, and 30. The norms $\|\epsilon^h_{\text{fine}}\|$ are over the finest level, and $\|\epsilon^h_{\text{all}}\|$ are over all three levels.

m	h	$\|\epsilon^h_{\text{all}}\|_\infty$	p	$\|\epsilon^h_{\text{fine}}\|_2$	p	$\|\epsilon^h_{\text{all}}\|_2$	p	λ/h
7	1/2048	4.280 E−5		8.449 E−7		2.608 E−6		7.31
7	1/4096	2.794 E−5	0.62	7.009 E−7	0.27	2.500 E−6	0.06	14.63
7	1/8192	1.971 E−5	0.50	6.713 E−7	0.06	2.521 E−6	−0.01	29.26

Table 2. Norms and convergence rates of solution error with adaptive three-level MLC *without* separating monopole solutions, for the example with $m = 7$. Compare with Table 1, which shows results of MLC separating monopole solutions. The norms $\|\epsilon^h_{\text{fine}}\|$ are over the finest level, and $\|\epsilon^h_{\text{all}}\|$ are over all three levels.

5.2. ***Timing results.*** In this section we present computational results demonstrating the low communication overhead of our implementation of the MLC algorithm on up to 1024 processors.

We ran on NERSC's Seaborg IBM SP system, located at the National Energy Research Scientific Computing Center[1]. Seaborg contains POWER3 SMP High Nodes interconnected with a "Colony" switch. Each node is an 16-way Symmetric Multiprocessor (SMP) based on 375 MHz Power-3 processors[2], sharing between 16 and 64 Gigabytes of memory, and running AIX version 5.1.

[1] http://www.nersc.gov/nusers/resources/SP

[2] http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/nighthawk.html

m	h	$\|\epsilon^h_{\text{all}}\|_\infty$	p	$\|\epsilon^h_{\text{fine}}\|_2$	p	$\|\epsilon^h_{\text{all}}\|_2$	p	λ/h
7	1/2048	4.498 E−6		2.431 E−8		2.471 E−8		7.31
7	1/4096	9.698 E−7	2.21	7.664 E−9	1.67	3.373 E−8	−0.45	14.63
15	1/2048	7.845 E−6		7.889 E−8		1.232 E−7		3.41
15	1/4096	1.121 E−6	2.81	6.919 E−9	3.51	9.526 E−9	3.69	6.83
30	1/2048	3.681 E−5		3.380 E−7		3.381 E−7		1.71
30	1/4096	2.530 E−6	3.86	2.365 E−8	3.84	5.587 E−8	2.60	3.41

Table 3. Norms and convergence rates of solution error with adaptive *two*-level MLC, for examples with $m = 7$, 15, and 30. Compare with Table 1, which shows results with three-level MLC. The norms $\|\epsilon^h_{\text{fine}}\|$ are over the finer level, and $\|\epsilon^h_{\text{all}}\|$ are over both levels.

	one-grid Mehrstellen				three-level MLC			
m	H	$\|\epsilon^H\|_\infty$	p	λ/H	h	$\|\epsilon^h\|_\infty$	p	λ/h
7	1/256	3.529 E−2		0.91	1/4096	4.735 E−6		14.63
7	1/512	4.193 E−4	6.40	1.83	1/8192	1.130 E−6	2.07	29.26
7	1/1024	1.726 E−5	4.60	3.66				
15	1/256	1.019 E−2		0.43	1/4096	4.906 E−6		6.83
15	1/512	3.288 E−3	1.63	0.85	1/8192	1.157 E−6	2.08	13.65
15	1/1024	1.446 E−4	4.51	1.71				
30	1/256	4.556 E−2		0.21	1/4096	5.274 E−6		3.41
30	1/512	4.167 E−3	3.45	0.43	1/8192	1.542 E−6	1.77	6.83
30	1/1024	9.687 E−4	2.10	0.85				

Table 4. Max norms and convergence rates of solution error with Mehrstellen on a single grid (on left), for examples with $m = 7$, 15, and 30. Also shown (on right) are the max norms of the solution error for the three-level MLC, copied from Table 1, with the same mesh spacing H at the coarse level.

The solver is written in a mixture of C++ and Fortran 77, and calls the FFTW library [10] for the fast discrete sine transforms in the Dirichlet Poisson solves. We used the IBM C++ and Fortran 77 compilers, `mpCC` and `mpxlf`. C++ code was compiled with the IBM `mpCC` compiler, using options `-O2 -qarch=pwr3 -qtune=pwr3`. Fortran 77 was compiled with `mpxlf` with `-O2` optimization. We used the standard environment variable settings, and we collected timings in batch

Size	three-level example			two-level example	
N	fine	middle	coarse	fine	coarse
2048	50 923 779	6 440 067	2 146 689	50 923 779	135 005 697
4096	405 017 091	21 567 171	16 974 593	405 017 091	1 076 890 625
8192	3 230 671 875	99 228 483	135 005 697		

Table 5. Numbers of solution points at each level in the three-level MLC example (with timing results in Table 6) and the two-level MLC example (with timing results in Table 7).

mode using *loadleveler*. The timings reported are based on wall-clock times, obtained with `MPI_Wtime()`.

The times reported are for the runs with the shortest total times of m set to 7, 15, or 30. Timers were placed around large function calls rather than inner loops to reduce the effects of noise in the timing results. The bulk-synchronous nature of the algorithm allows us to fully separate computation times from communication times. Reported running times do not include one-time startup costs such as a preprocessing phase for the serial James–Lackner solver that computes a matrix for obtaining outer-grid boundary conditions from multipole coefficients due to charges on the inner-grid boundary. This matrix depends only on the problem size and accuracy parameters, and its computation is considered a fixed overhead to be amortized over many calls to the solver.

In measuring the performance, we scaled the work with the number of processors. The run parameters and timing results for the performance tests of the three-level MLC are shown in Table 6. Processors are allocated to SMP nodes in such a way that each node runs 16 processors.

Results for performance tests of the two-level MLC are shown in Table 7. Since execution slows down when the memory capacity of a node is close to being reached, in the two-level MLC runs, processors are allocated to SMP nodes in such a way that each SMP node runs only eight processors — that is, half of the processors on the node.

Results for performance tests of the parallelized single-grid solver are shown in Table 8. In these examples, as with the two-level MLC runs, processors are allocated to SMP nodes with eight processors per node.

We define grind time as the processor-time taken per fine-level solution point. Ideally the grind time would remain constant over problem sizes and numbers of processors. We see from Table 6 that for the three-level MLC, grind times are fairly stable, at around 22 to 23 μs/point.

	Size	Times for each stage (seconds)							Total	Grind
P	N	InitF	InitM	Crse	BndM	FinM	BndF	FinF	(s)	(μs/pt)
16	2048	44.99	12.52	3.51	0.33	0.66	2.64	4.89	69.57	21.86
128	4096	45.51	6.76	10.19	0.15	0.30	4.12	4.75	71.83	22.70
1024	8192	46.01	3.95	13.04	0.15	0.17	4.03	4.78	72.28	22.91

Table 6. Timing breakdowns for runs of an adaptive three-level MLC solver, with P processors and domain length N. Boxes at the fine and middle levels are all cubes of length 32. InitF: Initial fine-level local solve. InitM: Initial middle-level local solve. Crse: Coarse-level solve. BndM: Boundary communication to middle-level final solve. FinM: Final middle-level solve. BndF: Boundary communication to fine-level final solve. FinF: Final fine-level solve.

	Size	Times for each stage (seconds)				Total	Grind
P	N	Init	Crse	Bnd	Fin	(s)	(μs/pt)
64	2048	11.73	22.61	0.26	1.22	35.84	45.04
512	4096	13.25	47.46	0.50	1.21	62.44	78.93

Table 7. Timing breakdowns for runs of an adaptive two-level MLC solver, with P processors and domain length N. Boxes at the fine level are all cubes of length 32. Init: Initial fine-level local solve. Crse: Coarse-level solve. Bnd: Boundary communication to fine-level final solve. Fin: Final fine-level solve.

	Size		Times for each stage (seconds)				Total	Grind
P	N	points	Homo	Normal	FMM	Inhomo	(s)	(μs/pt)
4	256	16 974 593	10.53	0.08	2.23	57.34	70.20	16.54
32	512	135 005 697	13.39	0.87	4.51	22.93	41.72	9.89
256	1024	1 076 890 625	13.65	3.06	10.53	19.26	46.52	11.06

Table 8. Timing breakdowns for runs of infinite-domain solver on one level, with P processors and domain length N, and given number of points, which is $(N+1)^3$. Homo: Initial homogeneous Dirichlet Poisson solve. Normal: Copying of Poisson solution and evaluation of normal derivatives. FMM: Fast multipole method. Inhomo: Final inhomogeneous Dirichlet Poisson solve.

P	Size N	Communication in stages (seconds) Boundary	Coarse	Residuals	Total (s)	% of runtime
16	2048	0.37	0.22	0.08	0.68	0.97 %
128	4096	1.56	0.58	0.14	2.28	3.17 %
1024	8192	1.40	1.77	0.68	3.85	5.32 %

Table 9. Communication time in the adaptive three-level MLC solve, for the same runs as reported in Table 6. Boundary: Copying of solutions within and between fine and middle levels (as illustrated in Figure 2). Coarse: Communication in the solve at the coarsest level. Residuals: Copying of residuals at the fine and middle levels.

Grind times vary by almost a factor of two in the two-level MLC examples in Table 7. These results are not as consistent as with the three-level example, because the coarse-level solution takes a majority of the run time: it is computed using a conventional parallel FFT algorithm that does not scale as well as the local solves, and has 2.65 times as many points as the local fine grids (Table 5). This lack of scaling also had an impact on the memory requirements. The two-level calculations required substantially more memory than the three-level calculations, so that we were only able to use eight processors per node for these runs, rather than the full 16 processors per node used in the three-level runs. In reporting the number of processors and computing the grind times in Table 7, we report the number of processors actually used, whereas the system resources required corresponded to double that number.

The lower parallel performance of the two-level calculations also affects the tradeoffs between using the two-level and three-level algorithms from an accuracy standpoint. For the $m = 30$ case, the accuracy of the two-level and three-level calculations are almost the same, and the cost of the two-level calculation is far greater: for example, the system resources required for the 4096-resolution two-level calculation are the same as those used for the 8192-resolution three-level calculation. As m decreases, the tradeoffs favor the two-level calculation more, but the computational costs of obtaining a given level of accuracy using the two different strategies remains within a factor of two.

In the runs of the three-level MLC, as shown in Table 6, over half the time is spent in the initial fine-level solves. With the particular problem sizes, each processor holds data for 96 fine-level boxes. The grind time for the initial fine-level solves ranges from 14.1 to 14.6 μs/point, and for the final fine-level solves ranges from 1.50 to 1.54 μs/point. Overall, we are able to scale a problem up from 16 to

P	Size N	AMR iterations	Time (s)	Grind (μs/pt)	Communication time (s)	% of runtime
16	2048	9	352.04	110.61	25.81	7.49%
128	4096	9	468.41	148.03	78.87	17.65%

Table 10. Times for multigrid Poisson solver with Dirichlet boundary conditions. Compare with Table 6 for MLC on the same fine-level and middle-level grids.

1024 processors with, at worst, a 4% increase in the grind time. The increase is due primarily to the increased cost of the global FFT solution at the coarsest level. We believe that the performance of our implementation of the global FFT solver can be improved from that seen here.

We compare these results with timings for an adaptive node-centered multigrid algorithm for solving Poisson's equation with Dirichlet boundary conditions [18] on the same platform. This algorithm is run on three levels of boxes, with the fine and middle levels being the same as were used with MLC, but the coarse level being fully refined into cubes of length 32, instead of parallel slabs. Although we are solving a different problem here, we believe that these results are typical of the cost of using the same algorithm to solving the infinite-domain problem along the lines of the algorithm in [1]. Comparison of results of the multigrid timings in Table 10 with the MLC timings in Table 6 shows that the multigrid algorithm takes 5 to 7 times longer than MLC, although a count of the number of floating-point operations shows that it uses only 1.38 to 1.45 times as many such operations as MLC. Considerably more time is spent in communication in this algorithm than in MLC (comparing Table 10 with Table 9). On the example on 128 processors, the time for communication, at 78.87 seconds, exceeds the total time for the MLC solve on the same grids.

Finally, we can infer from these results a lower bound on the grind time required for a Hockney algorithm to solve Poisson's equation at the same resolution as that on our finest grid. Judging from the time in the FinF column of Table 6, the time per grid point of an FFT solver for a 32^3 grid is about 1.52 μs per grid point. Thus the cost per mesh point per processor of performing an infinite-domain solution on a uniform grid with linear dimension N using the James–Lackner algorithm is at least $1.52 \times 2 \times 0.2 \log_2 N$ μs, where the factor of 2 comes from the minimum cost of solving the two Dirichlet problems for the James–Lackner algorithm, and $0.2 = 1/\log_2 32$. This leads to grind times of 6.7 to 7.9 μs for the range of mesh resolutions given here. Thus, the grind times for the three-level MLC calculations are approximately three times the lower bound we've estimated here.

6. Conclusions and future work

We have described here an extension to Anderson's Method of Local Corrections for solving Poisson's equation in free space on nested multiresolution grids in three dimensions. This is a noniterative domain-decomposition method based on computing local convolutions with the free-space Green's function on overlapping rectangular subdomains with a fixed number of grid points, combined with a representation of the nonlocal coupling between subdomains by a coarse-grid calculation in a manner that is structurally similar to a single iteration of an FAS multigrid method. The extension to locally-refined grids and to more than two levels is straightforward. A key technical step is an extension to three dimensions of the James–Lackner method for computing local convolutions, based on using FFTs for computing the volume potentials combined with a simplified version of the fast multipole method for surface-surface convolutions. This is combined with an exact treatment of the contribution to the local potential from the piecewise-constant component of the charge in each rectangle.

We demonstrated second-order accuracy of the method for a nontrivial example. We also found that the computational cost of the method is approximately three times per grid point that of FFT calculation at the same resolution, and scales to 1024 processors at approximately 95% parallel efficiency, with less than 7% of the run time in MPI communication calls. We have also compared the performance of this method to that of a conventional AMR multigrid solver on the same grid hierarchy, and found that, on 128 processors, the latter takes seven times as much time overall to compute the result, and spends 16 times as much time in MPI communication than the present method. We know of no other method for Poisson's equation in 3D that exhibits the same combination of performance and scalability on multiresolution grid. We believe that the results presented here indicate the possibility of scaling effectively to a PetaFlop computer (10^5 processors).

The results given here, while extremely promising, must be viewed as a first step in developing a robust and automatic piece of software. There are free parameters in the method, such as the dependence of the degree of overlap on the level of refinement, that are ad-hoc, and need to be defined systematically. One of the principal difficulties in this area is estimating and controlling the different sources of error in the algorithm separately and with complete generality. One aspect of solving that problem is for all components of the algorithm to have tunable accuracy, as opposed the present situation, in which the fourth-order Mehrstellen algorithm is a fixed target. Also, the current formulation of the algorithm does not preserve the geometric locality of the charge distributions. For example, the field induced on a patch on the middle level in a three-level calculation includes contributions from the charge distribution on finer patches not covered by the middle

patch. This feature makes it difficult to estimate the error as it propagates down through refinement levels. We are currently working on a version of the algorithm that will preserve locality under coarsening. As is the case with other adaptive methods, the general question of criteria for determining the needed grid refinement, as a function of space, time, and data, is not completely resolved. However, our treatment of the coupling between refinement levels may simplify the problem, relative to conventional finite difference methods [18]. Finally, there is still room for further performance improvement. For example, the parallel FFT solver used for the coarsest level is implemented using the Chombo communication primitives, that were not designed for the global communications required in the transform step. Since this calculation is a parallel bottleneck for the overall algorithm, any improvements would significantly improve the overall scaling of the method.

There are a number of directions in which the method described here could be extended. These include cell-centered solvers, solvers for the 3D Helmholtz equations, and higher-order methods. The extension to other boundary conditions on the domain boundary (Dirichlet, Neumann) is straightforward using method of images ideas [9]; a more challenging question is the extension of this approach to the case of Cartesian-grid representations of irregular boundaries [19; 15; 18].

Acknowledgments

Peter McCorquodale and Phillip Colella are supported by the Mathematical, Information, and Computational Sciences Division of the Office of Science, U.S. Department of Energy under contract number DE-AC03-76SF00098. Gregory Balls and Scott Baden were supported by the National Partnership for Advanced Computational Infrastructure (NPACI) under NSF contract ACI9619020. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

References

[1] A. S. Almgren, T. Buttke, and P. Colella, *A fast adaptive vortex method in 3 dimensions*, J. Comput. Phys. **113** (1994), no. 2, 177–200.

[2] C. R. Anderson, *A method of local corrections for computing the velocity field due to a distribution of vortex blobs*, J. Comput. Phys. **62** (1986), 111–123.

[3] S. B. Baden, *Run-time partitioning of scientific continuum calculations running on multiprocessors*, Ph.D. thesis, UC Berkeley Computer Science Division, April 1987.

[4] D. Bai and A. Brandt, *Local mesh refinement multi-level techniques*, SIAM Journal Sci. Stat. Comput. **8** (1987), 109–134.

[5] G. T. Balls, *A finite difference domain decomposition method using local corrections for the solution of Poisson's equation*, Ph.D. thesis, University of California, Berkeley, 1999.

[6] G. T. Balls and P. Colella, *A finite difference domain decomposition method using local corrections for the solution of Poisson's equation*, J. Comput. Phys. **180** (2002), no. 1, 25–53.

[7] A. Brandt, *Multilevel adaptive methods for boundary-value problems*, Math. Comp. **31** (1977), no. 138, 333–390.

[8] H. Cheng, J. Huang, and T. J. Leiterman, *A fast adaptive solver for the modified Helmholtz equation in two dimensions*, J. Comput. Phys. **211** (2006), no. 2, 616–637.

[9] F. Ethridge and L. Greengard, *A new fast-multipole accelerated Poisson solver in two dimensions*, SIAM Journal Sci. Comput. **23** (2001), no. 3, 741–760.

[10] M. Frigo and S. G. Johnson, *FFTW: An adaptive software architecture for the FFT*, ICASSP Conference Proceedings, vol. 3, ICASSP, 1998, pp. 1381–1384.

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

[12] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys. **73** (1987), 325–348.

[13] R. W. Hockney and J. W. Eastwood, *Computer simulation using particles*, McGraw-Hill, 1981.

[14] R. A. James, *The solution of Poisson's equation for isolated source distributions*, J. Comput. Phys. **25** (1977), no. 2, 71–93.

[15] H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys. **147** (1998), no. 2, 60–85.

[16] O. D. Kellogg, *Foundations of potential theory*, Dover Publications, 1953.

[17] K. Lackner, *Computation of ideal MHD equilibria*, Computer Physics Communications **12** (1976), no. 1, 33–44.

[18] P. McCorquodale, P. Colella, D. Grote, and J.-L. Vay, *A node-centered local refinement algorithm for Poisson's equation in complex geometries*, J. Comput. Phys. **201** (2004), 34–60.

[19] G. H. Shortley and R. Weller, *The numerical solution of Laplace's equation*, J. Appl. Phys. **9** (1938), 334–348.

[20] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*, Cambridge University Press, 2004.

Received October 30, 2006.

PETER MCCORQUODALE: PWMcCorquodale@lbl.gov
Lawrence Berkeley National Laboratory, 1 Cyclotron Road, MS 50A-1148, Berkeley, CA 94720, United States

PHILLIP COLELLA: PColella@lbl.gov
Lawrence Berkeley National Laboratory, 1 Cyclotron Road, MS 50A-1148, Berkeley, CA 94720, United States

GREGORY T. BALLS: gballs@ucsd.edu
Center for Scientific Computation in Imaging, University of California, San Diego, 9500 Gilman Drive # 0854, La Jolla, CA 92093-0854, United States

SCOTT B. BADEN: baden@cs.ucsd.edu
Department of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0114, United States

COMM. APP. MATH. AND COMP. SCI.
Vol. 2, No. 1, 2007

DUAL-BASED *A POSTERIORI* ERROR ESTIMATE FOR STOCHASTIC FINITE ELEMENT METHODS

LIONEL MATHELIN AND OLIVIER LE MAÎTRE

We present an *a posteriori* error estimation for the numerical solution of a stochastic variational problem arising in the context of parametric uncertainties. The discretization of the stochastic variational problem uses standard finite elements in space and piecewise continuous orthogonal polynomials in the stochastic domain. The *a posteriori* methodology is derived by measuring the error as the functional difference between the continuous and discrete solutions. This functional difference is approximated using the discrete solution of the primal stochastic problem and two discrete adjoint solutions (on two imbricated spaces) of the associated dual stochastic problem. The dual problem being linear, the error estimation results in a limited computational overhead. With this error estimate, different adaptive refinement strategies of the approximation space can be thought of: applied to the spatial and/or stochastic approximations, by increasing the approximation order or using a finer mesh. In order to investigate the efficiency of different refinement strategies, various tests are performed on the uncertain Burgers' equation. The lack of appropriate anisotropic error estimator is particularly underlined.

1. Introduction

Simulation of physical systems is often challenged by incomplete knowledge of model parameters, including initial conditions, boundary conditions, external forcing, physical properties and modeling constants. In these situations, it is relevant to rely on a probabilistic framework and to consider the unknown model data as random quantities. Consequently, it becomes essential to assess the variability of the model solution induced by the variability of the model data, i.e., to propagate and quantify the impact of the uncertainty on the model solution. In a probabilistic framework, the uncertainty quantification consists in the determination of the probability law of the model solution induced by the probability law of the data, in order to establish confidence intervals, to estimate limits of predictability and/or to support model-based decision analysis.

Keywords: error analysis, stochastic finite element method, uncertainty quantification, refinement scheme.

Uncertainty propagation and quantification has recently received considerable attention, particularly through the development of efficient spectral techniques based on Polynomial Chaos (PC) expansions. PC based methods were originally developed for engineering problems in solid mechanics [10; 25] and subsequently applied to a large variety of problems, including flow through porous media [8; 9], thermal problems [12; 13], incompressible [19; 20; 30] and compressible flows [18; 21] (see also [14] for a review of recent developments in PC methods for fluid flows) and reacting systems [7; 24]. PC expansions consist in the representation of the uncertain data as functionals of a finite set of independent random variables with prescribed densities, the uncertainty germ, and in expanding the dependence the model solution using a suitable basis of uncorrelated functionals of the germs. A classic choice for the basis is a set of polynomials in the germ. If the germ has zero-mean normalized Gaussian components, one obtains the Wiener–Hermite PC basis [28; 5], which is formed of generalized Hermite polynomials. Other density types of the germ components result in various families of orthogonal polynomials or mixtures of orthogonal polynomials [29]. Two distinct types of solution methods can be used to compute the expansion coefficients of the stochastic solution: the sampling based approaches and the Galerkin projection. In the former type of methods, one solves a series of deterministic problems for different values of the uncertain model data and makes use of the resulting sample set of solutions to estimate the expansion coefficients (see for instance [24; 20]). The second type of methods, which is considered in the following, consists on the contrary in a projection of the model equations (weak formulation) on the expansion basis. This Galerkin projection results in a set of generally coupled deterministic problems for the stochastic modes of the solution.

Piecewise polynomials [27] and multiwavelets [15; 16] were recently proposed as elements of the stochastic basis. These representations were developed to address the limitations of global spectral representations for complex, steep or even discontinuous dependencies of the model solution with regard to the data, for instance when a bifurcation appears for values of the data in the uncertainty domain. A key aspect of these discontinuous stochastic approximations is that they naturally offer flexibility for a local adaptation of the representation to the solution. This adaptation allows for improvements of the computed solution, through local refinements of the approximation space, while maintaining the dimension of the representation basis and of the set of coupled problems to be solved at a reasonable level. The refinement of the stochastic approximation space can in fact consist in an increase of the local expansion order (p-refinement) or in using polynomials being continuous over smaller supports (h-refinement). For instance, in [15; 16] the domain of the random parameters is partitioned in subdomains over which independent discontinuous low order expansions are employed. Heuristic criteria,

based on the spectrum of the local expansion, is used to decide whether the local expansion is sufficient or whether it should be improved by means of h-refinement, i.e., by splitting the subdomain into smaller ones, and along which dimension of the germ. A similar strategy is pursued in [27] but in the context of hp-spectral approximations. The refinement is also based on heuristic arguments involving the relative contribution of the higher order terms to the local solution expansion.

Although these schemes have been shown to provide significant improvements over global PC expansions, in terms of robustness (see for instance [17]) and computational efficiency, they still lack rigorous criteria for triggering the refinement. The objective of the present paper is therefore the derivation of a rigorous error estimator, to be used in place of the heuristic error indicators. To this end, we have decided to extend the dual-based *a posteriori* error technique commonly used in the (deterministic) finite element community. This choice was motivated by the firm and rigorous theoretical foundations of this error estimate technique, and because of its variational framework which makes it suitable for extension to the Galerkin projection of stochastic problems.

The paper is organized as follows. In Section 2, the variational formulation of a generic stochastic problem, based on a mathematical model involving parametric (data) uncertainties, is considered. The stochastic variational problem and construction of the approximation space are detailed. The latter involves a finite element discretization in space and a piecewise continuous approximation along the stochastic dimensions. In Section 3, the dual-based *a posteriori* error estimation is introduced. The methodology makes use of a differentiable functional to measure the difference between the exact (continuous) and approximated (discrete) stochastic solutions. Provided the discrete solution is sufficiently close to the continuous one, their functional difference is shown to be well approximated by a simple estimate. This estimate involves the discrete solutions of the primal and associated dual problems, and the continuous adjoint solution of the dual problem. A classic surrogate of the continuous adjoint solution is proposed, resulting in an error estimate methodology requiring the resolution of the discrete primal problem and two dual problems on different approximation spaces. The dual problems to be solved being linear, the computational overhead of the error estimator is expected to be limited. In Section 4, we discuss the various strategies that can be subsequently used to improve the approximation in order to reduce the error. The reduction of the error can be performed by using smaller elements or by increasing the orders of the spatial and stochastic approximation spaces. As in the deterministic context, the determination of the optimal refinement strategy is an open question, which is made even more difficult and critical in the present stochastic context where the stochastic space (domain of the germ) may have many dimensions. Consequently, Section 5 presents some numerical tests aiming at showing the validity of the

proposed dual-based error estimator in deciding which spatial/stochastic elements need priority refinement. The test problem is based on the 1-D Burgers' equation, with uncertainty on the viscosity and a boundary condition. Different algorithms of increasing complexity are proposed for the local refinement of the stochastic and spatial approximations, based on the dual-based error estimation. Finally, major findings of this work and a few recommendations for future developments are summarized in Section 6.

2. Variational formulation of uncertain flow

2.1. ***Deterministic variational problem.*** We will consider the standard variational problem for u on a M-dimensional domain $\Omega_x \subset \mathbb{R}^M$ with homogeneous Dirichlet boundary condition ($u = 0$) on the boundary $\partial\Omega_x$ of Ω_x:

$$a(u; \varphi) = b(\varphi) \quad \forall \varphi \in \mathcal{V}_x, \tag{1}$$

to be solved for $u \in \mathcal{V}_x$, a suitable Hilbert space of Ω_x. In Eq. (1), a is a differentiable semilinear form and b a linear functional.

2.2. ***Stochastic variational problem.*** It is assumed that the mathematical model given by Eq. (1) involves some parameters, or data, denoted by a real-valued vector d. The data may for instance consist of some physical constants involved in the model. Clearly, the solution u of the variational problem depends on the data value, a fact stressed by making explicit the dependence of the variational problem with d:

$$a(u; \varphi|d) = b(\varphi|d) \quad \forall \varphi \in \mathcal{V}_x. \tag{2}$$

If the actual value of the data d is not exactly known, (is uncertain), it is suitable to consider d as a random quantity defined on an abstract probability space $(\Theta, \mathcal{B}, dP)$, Θ being the set of elementary outcomes θ, $\mathcal{B}$ the σ-algebra of the events and dP a probability measure. In this context, the solution of the model is also random. In the following, we adopt the convention consisting in using uppercase letters to denote random quantities. Therefore, the random solution U and data D are dependent stochastic quantities defined on the same probability space $(\Theta, \mathcal{B}, dP)$; the dependency between U and D is prescribed by the model. Uncertainty propagation and quantification thus consists in the inference of the probability law of U, given the probability law of D and the mathematical model relating the two. It is assumed that the problem is well-posed in the sense that problem (2) has almost surely a unique solution.

We denote $\mathcal{V}_\Theta = L^2(\Theta, dP)$ the space of second order random variables. We thus have to solve, for $U \in \mathcal{V}_x \otimes \mathcal{V}_\Theta$,

$$A(U; \Phi|D) = B(\Phi|D) \quad \forall \Phi \in \mathcal{V}_x \otimes \mathcal{V}_\Theta, \tag{3}$$

where

$$A(U;\Phi|D) \equiv \int_\Theta a(U(\theta);\Phi(\theta)|D(\theta))\,dP(\theta),$$
$$B(\Phi|D) \equiv \int_\Theta b(\Phi(\theta)|D(\theta))\,dP(\theta). \tag{4}$$

2.3. ***Stochastic discretization.*** We assume that D is parameterized as a functional of a finite number N of independent identically distributed real valued random variables ξ_i, defined on $(\Theta, \mathcal{B}, dP)$ with value in $S_\xi \subset \mathbb{R}$:

$$D = D(\xi), \quad \xi = (\xi_1, \ldots, \xi_{\mathrm{N}}) \in (S_\xi)^{\mathrm{N}} \equiv \Omega_\xi \subset \mathbb{R}^{\mathrm{N}}. \tag{5}$$

The vector ξ of random parameters is often referred to as the uncertainty germ. We denote p the known probability density function of ξ_i such that, by virtue of the independence, the joint distribution of ξ is given by

$$p_\xi(\xi) = p_\xi(\xi_1, \ldots, \xi_{\mathrm{N}}) = \prod_{i=1}^{\mathrm{N}} p(\xi_i). \tag{6}$$

Without loss of generality, we shall restrict ourself in the following to germs having uniformly distributed components on $S_\xi = [-1, 1]$ and consequently we have

$$p(\xi_i) = \begin{cases} 1/2 & \text{if } \xi_i \in [-1, 1], \\ 0 & \text{otherwise,} \end{cases} \qquad \Omega_\xi = [-1, 1]^{\mathrm{N}}. \tag{7}$$

Note however that the developments given below can be easily extended to the situation where the ξ_i have different ranges and/or different distributions. The variational problem can be formulated in the image probability space $(\Omega_\xi, \mathcal{B}_\xi, p_\xi)$, using

$$\begin{aligned} A(U;\Phi|D) &= \int_\Theta a(U(\theta);\Phi(\theta)|D(\theta))\,dP(\theta) \\ &= \int_{\Omega_\xi} a(U(\xi);\Phi(\xi)|D(\xi)\ p_\xi(\xi))d\xi \equiv \langle a(U;\Phi|D)\rangle_{\Omega_\xi}, \end{aligned} \tag{8}$$

$$\begin{aligned} B(\Phi|D) &= \int_\Theta b(\Phi(\theta)|D(\theta))\,dP(\theta) \\ &= \int_{\Omega_\xi} b(\Phi(\xi)|D(\xi))\ p_\xi(\xi)d\xi \equiv \langle b(\Phi|D)\rangle_{\Omega_\xi}. \end{aligned} \tag{9}$$

Moreover, the stochastic functional space is now $\mathcal{V}_\xi = L^2(\Omega_\xi, p_\xi)$ and the variational problem becomes

$$A(U;\Phi|D) = B(\Phi|D) \quad \forall \Phi \in \mathcal{V}_x \otimes \mathcal{V}_\xi, \tag{10}$$

to be solved for $U \in \mathcal{V} \equiv \mathcal{V}_x \otimes \mathcal{V}_\xi$.

Following [27], we rely on piecewise orthogonal polynomials to construct the stochastic approximation space. The stochastic range Ω_ξ is divided into a collection of $\mathrm{N_b}$ nonoverlapping subdomains $\Omega_\xi^{(m)}$ referred to as stochastic elements (SEs) in the following. In this work, the SEs are hyper-rectangles:

$$\Omega_\xi = \bigcup_{m=1}^{\mathrm{N_b}} \Omega_\xi^{(m)}, \quad \Omega_\xi^{(m)} = [\xi_1^{(m),-}, \xi_1^{(m),+}] \times \cdots \times [\xi_\mathrm{N}^{(m),-}, \xi_\mathrm{N}^{(m),+}]. \tag{11}$$

On $\Omega_\xi^{(m)}$, the dependence of the data and solution with the random germ ξ is expressed as a truncated Fourier-like series,

$$U(\xi \in \Omega_\xi^{(m)}) = \sum_{k=0}^{\mathrm{P}(m)} u_k^{(m)} \Psi_k^{(m)}(\xi), \quad D(\xi \in \Omega_\xi^{(m)}) = \sum_{k=0}^{\mathrm{P}(m)} d_k^{(m)} \Psi_k^{(m)}(\xi), \tag{12}$$

where $\Psi_k^{(m)}(\xi)$ are orthogonal random polynomials in ξ and $u_k^{(m)}$, $d_k^{(m)}$ are the deterministic expansion coefficients over $\Omega_\xi^{(m)}$ of the solution and data respectively. The orthogonality of the random polynomials is defined with regard to the expectation over the respective SE. Denoting by $\langle\cdot\rangle_{\Omega_\xi^{(m)}}$ the expectation over the m-th SE, we can write the orthogonality of the polynomials as

$$\begin{aligned} \left\langle \Psi_k^{(m)} \Psi_{k'}^{(m)} \right\rangle_{\Omega_\xi^{(m)}} &= \frac{1}{\left|\Omega_\xi^{(m)}\right|} \int_{\Omega_\xi^{(m)}} \Psi_k^{(m)}(\xi)\, \Psi_{k'}^{(m)}(\xi)\, p_\xi(\xi)\, d\xi \\ &= \delta_{kk'} \left\langle \Psi_k^{(m)2} \right\rangle_{\Omega_\xi^{(m)}}, \end{aligned} \tag{13}$$

where

$$\left|\Omega_\xi^{(m)}\right| = \int_{\Omega_\xi^{(m)}} p_\xi(\xi)\, d\xi, \tag{14}$$

and $\delta_{kk'}$ is the usual Kronecker delta symbol. These polynomials vanish outside their respective support:

$$\Psi_k^{(m)}(\xi \notin \Omega_\xi^{(m)}) = 0 \quad \forall k = 0, \ldots, \mathrm{P}(m). \tag{15}$$

The number of terms $\mathrm{P}(m)$ in the expansions Eqs. (12) is a function of the selected stochastic expansion order $\mathrm{q}(m)$ of the SE:

$$\mathrm{P}(m) + 1 = \frac{(\mathrm{q}(m) + \mathrm{N})!}{\mathrm{q}(m)!\,\mathrm{N}!}. \tag{16}$$

The ξ_i being uniformly distributed, the polynomials $\Psi_k^{(m)}$ are simply rescaled and shifted multidimensional Legendre polynomials [1]. The stochastic approximation space is

$$\mathcal{V}_\xi^h = \mathrm{span}\Big(\{\Psi_k^{(m)}\},\ 1 \le m \le \mathrm{N_b},\ 0 \le k \le \mathrm{P}(m)\Big), \tag{17}$$

and the stochastic approximation can be improved by increasing the number $\mathrm{N_b}$ of SEs, i.e., through refinement of the partition of Ω_ξ, and/or by increasing the stochastic expansion order $\mathrm{q}(m)$ over some stochastic elements.

2.4. ***Finite element discretization.*** Consider a partition of Ω_x into a set of $\mathrm{N_x}$ nonoverlapping finite elements (FE) with respective support $\Omega_x^{(l)}$ for $l = 1, \ldots, \mathrm{N_x}$:

$$\Omega_x = \bigcup_{l=1}^{\mathrm{N_x}} \Omega_x^{(l)}. \tag{18}$$

The FE approximation of the continuous solution U, denoted by U^h, over the element $\Omega_x^{(l)}$, is given by

$$U^h(x \in \Omega_x^{(l)}) = \sum_{i=1}^{\mathrm{N_d}(l)} U_i^{(l)} \mathcal{N}_i^{(l)}(x), \tag{19}$$

where $\mathrm{N_d}(l)$ is the number of degrees of freedom of the l-th element and $\mathcal{N}_i^{(l)}$ the associated spatial shape functions. We denote $\mathrm{p}(l)$ the polynomial order of the shape functions over $\Omega_x^{(l)}$. The spatial approximation space is thus

$$\mathcal{V}_x^h = \mathrm{span}\left(\{\mathcal{N}_i^{(l)}\},\ 1 \le l \le \mathrm{N_x},\ 1 \le i \le \mathrm{N_d}(l)\right), \tag{20}$$

and the spatial approximation can be improved by a refinement of the partition of the spatial domain Ω_x or by increasing the spatial order $\mathrm{p}(l)$ of some finite elements.

2.5. ***The approximation space $\mathcal{V}^h$.*** From the stochastic and spatial approximation spaces defined above, the approximation space $\mathcal{V}^h$ of the stochastic variational problem is seen to be:

$$\mathcal{V}^h = \mathcal{V}_x^h \otimes \mathcal{V}_\xi^h. \tag{21}$$

The solution at a point (x, ξ) of $\Omega \equiv \Omega_x \times \Omega_\xi$ has for expression:

$$U(x \in \Omega_x^{(l)}, \xi \in \Omega_\xi^{(m)}) = \sum_{i=1}^{\mathrm{N_d}(l)} \sum_{k=0}^{\mathrm{P}(m)} u_{i,k}^{(l,m)}\, \mathcal{N}_i^{(l)}(x)\, \Psi_k^{(m)}(\xi), \tag{22}$$

where the deterministic coefficient $u_{i,k}^{(l,m)}$ is the k-th uncertainty mode of the m-th SE for the i-th degree of freedom of the l-th FE.

An immediate consequence of the tensored construction of the approximation space $\mathcal{V}^h$ is that the spatial FE discretization is the same for all the stochastic elements $\Omega_\xi^{(m)}$, and conversely the stochastic discretization is the same for all spatial finite elements $\Omega_x^{(l)}$. This is clearly not optimal as some portions of the stochastic domain Ω_ξ may require finer spatial discretization than others to achieve a similar accuracy. Conversely, the solution in some parts of spatial domain Ω_x

may exhibit more complex dependences with regard to $D(\xi)$, therefore requiring a finer stochastic discretization than at other locations. However, for the tensored construction $\mathcal{V}^h$, the discrete solution can be improved through a) refinement of the FE approximation space $\mathcal{V}_x^h$ uniformly over Ω_ξ, and b) refinement of the stochastic approximation space $\mathcal{V}_\xi^h$ uniformly over Ω_x.

In fact, this symmetric situation can be easily relaxed: an adaptation to each SE of the spatial discretization, i.e., the number of elements $\mathrm{N_x}$ and/or the number of degrees of freedom of the elements $\mathrm{N_d}$, causes no difficulty. This is due to the complete independence of the solution over different stochastic elements, a feature emerging from the absence of any differential operator along the uncertainty dimensions. Consequently, the adaptation of $\mathcal{V}_x^h$ with the SEs was actually implemented and used for the generation of the results presented hereafter. However, to simplify the presentation of the method and the notation, this feature is not detailed here. On the other hand, using a variable stochastic approximation for different spatial FE is much more cumbersome and remains to be investigated. This adaptation would require the development of nonobvious matching conditions of the stochastic approximation across FE boundaries.

3. Dual-based *a posteriori* error estimate

3.1. ***A posteriori error.*** For a finite dimensional subspace $\mathcal{V}^h \subset \mathcal{V}$, the discretized solution $U^h \in \mathcal{V}^h$ is the Galerkin approximation defined as the solution of the discrete problem

$$A(U^h; \Phi^h | D^h) = B(\Phi^h | D^h) \quad \forall \Phi^h \in \mathcal{V}^h. \tag{23}$$

Let $\mathcal{J} : \Omega \to \mathbb{R}$ be a differentiable functional of the solution. In the spirit of [3] and [2] among others, one is interested in approximating $\mathcal{J}(U)$ as closely as possible by $\mathcal{J}(U^h)$, i.e., to minimize the difference $\mathcal{J}(U) - \mathcal{J}(U^h)$ in some sense. We seek for an expression of $\mathcal{J}(U) - \mathcal{J}(U^h)$. To this end, let us define the Lagrangian $\mathcal{L}$ of the continuous solution by:

$$\mathcal{L}(U; Z) \equiv \mathcal{J}(U) + B(Z|D) - A(U; Z|D), \tag{24}$$

where $Z \in \mathcal{V}$ is the adjoint variable of the continuous problem. The adjoint variable Z is a Lagrange multiplier of the optimization problem for the minimization of $\mathcal{J}(U)$ under the constraints of Eq. (10). Formally, this minimum corresponds to the stationary points of $\mathcal{L}$:

$$\frac{\partial \mathcal{L}}{\partial U} = \mathcal{J}'(U; \Phi') - A'(U; \Phi', Z|D) = 0 \quad \forall \Phi' \in \mathcal{V}, \tag{25}$$

$$\frac{\partial \mathcal{L}}{\partial Z} = B(\Phi|D) - A(U; \Phi|D) = 0 \quad \forall \Phi \in \mathcal{V}. \tag{26}$$

Eq. (25) is the adjoint (or dual) problem, while Eq. (26) is the state (or primal) problem. The derivatives are here in the Gâteaux sense:

$$\mathcal{J}'(U;\Phi') = \lim_{\varepsilon\to 0} \frac{\mathcal{J}(U+\varepsilon\,\Phi') - \mathcal{J}(U)}{\varepsilon},$$
$$A'(U;\Phi', Z|D) = \lim_{\varepsilon\to 0} \frac{A(U+\varepsilon\,\Phi'; Z|D) - A(U; Z|D)}{\varepsilon}.$$

Assuming that these limits exist, the derivatives are unique. Note that following these definitions, the derivatives are generally nonadditive and nonlinear with regard to U and Φ'. However, the derivatives are homogeneous,

$$\mathcal{J}'(U;\alpha\Phi') = \alpha\,\mathcal{J}'(U;\Phi'), \quad A'(U;\alpha\Phi', Z|D) = \alpha\,A'(U;\Phi', Z|D),$$

so we will adopt the convention that functionals are at least homogeneous with regard to the first argument after the right-side of a semicolon, and linear with regard to the second argument, if any.

The discrete counterpart of the dual and primal problems are in turn

$$\mathcal{J}'(U^h;\Phi^{h'}) - A'(U^h;\Phi^{h'}, Z^h|D^h) = 0 \quad \forall\Phi^{h'}\in\mathcal{V}^h, \tag{27}$$
$$B(\Phi^h|D^h) - A(U^h;\Phi^h|D^h) = 0 \quad \forall\Phi^h\in\mathcal{V}^h. \tag{28}$$

Combining these results, one obtains at the solutions $\{U, Z\}\in\mathcal{V}$, $\{U^h, Z^h\}\in\mathcal{V}^h$

$$\begin{aligned}\mathcal{L}(U,Z) - \mathcal{L}(U^h,Z^h) &= \mathcal{J}(U) + B(Z) - A(U;Z) - \mathcal{J}(U^h) - B(Z^h) + A(U^h;Z^h)\\ &= \mathcal{J}(U) - \mathcal{J}(U^h),\end{aligned} \tag{29}$$

where the dependences of A and B on D have been dropped to simplify the notations. Here, the operators applied to discrete solutions are understood to correspond to their respective discrete counterpart. Then, the error estimates derived below account for the discretization error. It is seen from Eq. (29) that the difference in $\mathcal{J}$ for the continuous and discrete solutions is equal to the difference in their respective Lagrangian.

3.2. ***A posteriori error estimation.*** Following [4], among others, we now derive a more practical expression for the difference $\mathcal{J}(U) - \mathcal{J}(U^h)$. Let $K(\cdot)$ be a differentiable functional on a given functional space $\mathcal{W}$. The difference $K(v) - K(v^h)$, for v and $v'\in\mathcal{W}$, can be expressed as an integral between v and v^h of the derivative of K:

$$K(v) - K(v^h) = \int_{v^h}^{v} K'(v')\,dv'. \tag{30}$$

The integration path can be parameterized to obtain

$$K(v) - K(v^h) = \int_0^1 K'(v^h + s(v-v^h))(v-v^h)ds = \int_0^1 K'(v^h + s\,e_v; e_v)ds, \tag{31}$$

where $e_v \equiv v - v^h$. Using $K'(v) = 0$, we can rewrite the right-hand side of Eq. (31) as

$$K(v) - K(v^h) = \int_0^1 K'(v^h + s e_v; e_v) ds + \tfrac{1}{2} \left(K'(v^h; e_v) - K'(v^h; e_v) + K'(v; e_v) \right). \tag{32}$$

Making use of the Galerkin orthogonality and the trapezoidal rule we obtain

$$K(v) - K(v^h) = \frac{1}{2} K'(v^h; e_v) + \frac{1}{2} \int_0^1 K^{(3)}(v^h + s\, e_v; e_v^3)\, s\, (s-1)\, ds. \tag{33}$$

Applying this relation to the difference of the Lagrangian of the continuous and discrete solutions leads, after some algebra, to:

$$\mathcal{J}(U) - \mathcal{J}(U^h) = \frac{1}{2} \left[\rho(U^h, Z - \Phi'^h) + \rho^*(Z^h, U - \Phi^h) \right] + \widetilde{R}, \tag{34}$$

with the residuals

$$\rho(U^h, \cdot) \equiv B(\cdot) - A(U^h; \cdot), \tag{35}$$

$$\rho^*(Z^h, \cdot) \equiv \mathcal{J}'(U^h, \cdot) - A'(U^h; \cdot, Z^h). \tag{36}$$

The remainder term $\widetilde{R}$ in Eq. (34) has for expression

$$\widetilde{R} = \frac{1}{2} \int_0^1 \big(\mathcal{J}^{(3)}(U^h + s\, E_U; E_U^3) - A^{(3)}(U^h + s\, E_U; E_U^3, Z^h + s\, E_Z) - 3\, A''(U^h + s\, E_U; E_U^2\, E_Z) \big)\, s\, (s-1)\, ds, \tag{37}$$

with the error terms defined as $E_U = U - U^h$ and $E_Z = Z - Z^h$. Thus $\widetilde{R}$ is cubic in the error, suggesting that it can be neglected provided that the continuous and discrete solutions are sufficiently close. It is also seen that the residuals are functional of both the primal and dual continuous solutions U and Z, such that using Eq. (34) to estimate $\mathcal{J}(U) - \mathcal{J}(U^h)$ would require two surrogates of U and Z even if $\widetilde{R}$ is neglected. In fact, the expression can be further simplified to remove the contribution of U. Using an integration by part of $\widetilde{R}$, one obtains [4]

$$\rho^*(Z^h, U - \Phi^h) = \rho(U^h, Z - \Phi'^h) + \Delta\rho, \tag{38}$$

where

$$\Delta\rho = \int_0^1 \left[A''(U^h + s E_U; E_U^2, Z^h + s E_Z) - \mathcal{J}''(U^h + s E_U; E_U^2) \right] ds. \tag{39}$$

Introducing this result into Eq. (34) leads to the final expression for the approximation error:

$$\mathcal{J}(U) - \mathcal{J}(U^h) = \rho(U^h, Z - \Phi'^h) + r, \tag{40}$$

where

$$r = \int_0^1 \left[A''(U^h + sE_U; E_U^2, Z) - \mathcal{J}''(U^h + sE_U; E_U^2) \right] s\, ds. \tag{41}$$

The remainder term r is now quadratic in E_U and will be neglected, assuming again that the discrete solution U^h is indeed a close enough approximation of U.

3.3. ***Methodology.*** At this point we have an estimate of the approximation error given by

$$\mathcal{J}(U) - \mathcal{J}(U^h) \approx B(Z - Z^h | D^h) - A(U^h; Z - Z^h | D^h), \tag{42}$$

where we have substituted Φ'^h by the adjoint solution of the discrete problem in Eq. (40), as usual in *a posteriori* error methodology. To evaluate this estimate, one needs to know the solutions U^h and Z^h of the primal and dual discrete problems and the solution Z of the continuous dual problem given by Eq. (25). However, the continuous dual problem can not be solved as it requires the knowledge of the exact solution U. Instead, a surrogate of Z denoted $\widetilde{Z}$ is used. This surrogate is classically constructed by solving a discrete dual problem on a refined finite dimensional space $\mathcal{V}^{\widetilde{h}}$ containing $\mathcal{V}^h$. The methodology is thus the following. Given an approximation space $\mathcal{V}^h$ we solve the primal and dual problems Eqs. ((28),(27)) for U^h and $Z^h \in \mathcal{V}^h$. The refined space $\mathcal{V}^{\widetilde{h}} \supset \mathcal{V}^h$ is constructed by increasing the polynomial orders of both the approximation space $\mathcal{V}_x^h$ and $\mathcal{V}_\xi^h$, and we solve the following dual problem for $\widetilde{Z} \in \mathcal{V}^{\widetilde{h}}$

$$\mathcal{J}'(U^h; \Phi) - A'(U^h; \Phi, \widetilde{Z} | D^{\widetilde{h}}) = 0 \quad \forall \Phi \in \mathcal{V}^{\widetilde{h}}. \tag{43}$$

It yields the *a posteriori* error estimate given by

$$\mathcal{J}(U) - \mathcal{J}(U^h) \approx B(\widetilde{Z} - Z^h | D^h) - A(U^h; \widetilde{Z} - Z^h | D^h). \tag{44}$$

Two important remarks are necessary at this point. First, it is underlined that the dual problems are linear and significantly less expansive to solve than the primal problems, even in an enriched approximation space. Second, as shown by Eq. (43), the adjoint solution $\widetilde{Z}$ is based on a functional form A' constructed with the approximation of D on the enriched space $\mathcal{V}^{\widetilde{h}}$. As a consequence, the resulting error estimate based on $\widetilde{Z}$ accounts for possible error in the approximation of the uncertain data $D(\xi)$ on $\mathcal{V}_\xi^h$.

4. Refinement procedures

4.1. ***Global and local error estimates.*** The *a posteriori* error methodology described in Section 3 gives access to an estimate of $\mathcal{J}(U) - \mathcal{J}(U^h)$ according to Eq. (44). The global approximation error η is therefore

$$\begin{aligned}\eta &= \left|A(U^h;\widetilde{Z}-Z^h|D^h)-B(\widetilde{Z}-Z^h|D^h)\right| \\ &= \left|\left\langle a(U^h;\widetilde{Z}-Z^h|D^h)-b(\widetilde{Z}-Z^h|D^h)\right\rangle_{\Omega_\xi}\right| \\ &\leq \sum_{m=1}^{\mathrm{N_b}}\left|\Omega_\xi^{(m)}\right|\left|\left\langle a(U^h;\widetilde{Z}-Z^h|D^h)-b(\widetilde{Z}-Z^h|D^h)\right\rangle_{\Omega_\xi^{(m)}}\right|. \end{aligned} \tag{45}$$

Defining the local error on the element $\Omega_x^{(l)}\times\Omega_\xi^{(m)}$ by

$$\eta_{l,m}\equiv\left|\int_{\Omega_\xi^{(m)}}\int_{\Omega_x^{(l)}}\left[\widetilde{a}(U^h;\widetilde{Z}-Z^h|D^h)-\widetilde{b}(\widetilde{Z}-Z^h|D^h)\right]\,p_\xi(\xi)\,dx\,d\xi\right|, \tag{46}$$

where

$$\int_{\Omega_x}\widetilde{a}(u;v|d)dx=a(u;v|d),\qquad \int_{\Omega_x}\widetilde{b}(v|d)dx=b(v|d),$$

we obtain the following inequality:

$$\eta\leq\sum_{l=1}^{\mathrm{N_x}}\sum_{m=1}^{\mathrm{N_b}}\eta_{l,m}. \tag{47}$$

Then, the objective is to refine the approximation space $\mathcal{V}^h$ in order to reduce the global error η as estimated from the *a posteriori* error analysis. A popular strategy to ensure that the global error gets below a given threshold value ϵ_η is to refine the approximation such that

$$\eta_{l,m}<\frac{\epsilon_\eta}{\mathrm{N_x}\mathrm{N_b}}=\epsilon,\qquad\forall l,m\in[1,\mathrm{N_x}]\times[1,\mathrm{N_b}]. \tag{48}$$

4.2. ***Refinement strategies.*** If the criterion given in Eq.(48) is not satisfied for at least one SFE, the approximation space needs refinement. Different types of refinements are possible. First, from the tensored construction of the approximation space, $\mathcal{V}^h=\mathcal{V}_x^h\otimes\mathcal{V}_\xi^h$, it is seen that the refinement may concern the spatial or stochastic approximation spaces, or both. To distinguish these two types of refinement we shall refer in the following to x and ξ-refinement for the spatial and stochastic refinements respectively. Second, the refinement can be based on construction of finer partitions of the domains or on increased approximation orders, hereafter referred to as h- and p-refinements respectively. Therefore, we can choose between four fundamental types of refinements to reduce the approximation error to satisfy Eq. (48), h_ξ-, h_x-, p_ξ- or p_x-refinements, or any combination of the four.

The problem is thus to find the refinement strategy that yields the largest decay of the discretization error for the lowest computational cost. The difficulty here is that the local error estimate only provides some information about the elements (SEs and FEs) over which the approximation is insufficient. In other words, if for some l

and m the local error is such that $\eta_{l,m} > \epsilon$ then we can only safely consider that the approximation error over $\Omega_\xi^{(m)} \times \Omega_x^{(l)}$ is too large but nothing more. Specifically, it is not possible to decide (a) between h- or p-refinement and (b) whether one should enrich the approximation space $\mathcal{V}_x^h$ or $\mathcal{V}_\xi^h$.

Difficulty (a) is a classic problem in (deterministic) hp-finite-element methods. In the deterministic context, different strategies have been proposed to support the decision regarding h- or p-refinement, and most of these strategies are based on trial approaches. For instance, in [11], a systematic trial of h-refinement is performed. The efficiency of the h-refinement is subsequently measured by comparing the resulting error reduction with its theoretical value estimated using the convergence rate of the FE scheme. If the efficiency of the h-refinement is not satisfactory, a p-refinement is enforced at the following refinement step. This type of trial/verification approach has not been retained here because of its numerical cost. Difficulty (b) is on the contrary specific to stochastic finite-element methods and thus remains entirely to be investigated. A possible way to deal with difficulty (b) can be envisioned again by a trial approach where one would apply successively x and ξ-refinements to measure the respective effectiveness in error reduction. Again, trial approaches are expected to be overly expensive in the stochastic context where the size of the discrete problems to be solved can be many times larger than for the deterministic case: better approaches, yet to be thought, are needed here.

Another issue arising in the stochastic context is the potentially large dimensionality N of the stochastic domain Ω_ξ: an isotropic h_ξ-refinement, where SEs are broken into smaller ones along each dimension ξ_i, can quickly result in a prohibitively large number of SEs. This issue was already observed in [15; 16; 17] where adaptive multiwavelet approximations are used. Rather, it is desirable to gain further information on the structure of the local error $\eta_{l,m}$ in order to refine along the error's principal directions solely. Several approaches may be thought of to deal with this constraint. In the context of deterministic finite element method, several anisotropic error estimators have been rigorously derived based on higher order information. Among others, [23] and [22] use the Hessian matrix based on Clément interpolants [6] to derive an estimate of the directional errors. Thought attractive, this method has only been derived for first-order finite elements (P1) and its extension to higher order remains largely an open problem. This limitation precludes its use in the present context where approximation order q is routinely larger than one. As a result, we feel that the issue of anisotropic refinement remains largely to be addressed while being the most critical aspect of the refinement strategies; it is also the possible source of significant improvements for the Adaptive Stochastic Finite Element method. In fact, it is anticipated that the derivation of anisotropic refinement techniques will allow to deal with problems involving a larger set of random variables than currently tractable.

Considering all these difficulties, it was decided to first verify the effectiveness of the dual-based *a posteriori* error estimation in indicating which elements need refinement, and to delay the question of the refinement strategy decision to a future work. Consequently, we present in a next section some numerical tests which essential purposes are to prove that the proposed error estimator indeed detect areas of Ω where the error is the most significant. Still, we perform refinements, of increasing complexity, without pretending in any way that the decision algorithms used yield optimal approximation spaces, but merely that they allow for a reduction of the global error to an arbitrary small level.

5. Numerical examples

5.1. ***Uncertain Burgers' equation.*** To test the *a posteriori* error estimator, we consider the 1-D Burgers' equation on the spatial domain $\Omega_x \in [x^-, x^+]$:

$$\begin{cases} \dfrac{1}{2}\,(u\,(1-u))_x - \mu\, u_{xx} = 0 \quad \forall x \in [x^-, x^+], \\ u(x^-) = u^-, \quad u(x^+) = u^+. \end{cases} \tag{49}$$

This equation is widely used in particular in the fluid dynamics community as it features essential ingredients: diffusion as well as a quadratic convective term. Depending on the boundary conditions, the solution of the Burgers' equation exhibits areas where $u(x)$ is nearly constant and equal to u^- (for $x \simeq x^-$) and u^+ (for $x \simeq x^+$) with a central area, the transition layer, where u quickly evolves from u^- to u^+ according to an hyperbolic tangent profile having an increasing steepness with decreasing the fluid viscosity.

5.1.1. *Uncertainty settings.* We consider the random solution $U(x, \xi)$ of the Burgers' equation which arises when the viscosity μ is uncertain and parameterized by the random vector ξ: $\mu = \mu(\xi)$. As discussed above, ξ is uniformly distributed in $[-1, 1]^{\mathrm{N}}$. The number N of random variables depends on the parameterization. To ensure the existence of a solution to the stochastic problem, the parameterization is selected such that the viscosity is almost surely positive. The stochastic Burgers' equation is thus:

$$\begin{cases} \dfrac{1}{2}\,[U(x,\xi)\,(1-U(x,\xi))]_x - \mu(\xi)\,U_{xx}(x,\xi) = 0 \quad \forall x \in [x^-, x^+], \\ U(x^-,\xi) = u^-, \quad U(x^+,\xi) = u^+. \end{cases} \tag{50}$$

The viscosity is parameterized using $\mathrm{N} = 2$ random variables as follows

$$\mu(\xi) = \mu_0 + \mu_1\xi_1 + \mu_2\xi_2, \quad \mu_0 > 0. \tag{51}$$

The expectation of the viscosity is $\langle\mu\rangle_{\Omega_\xi} = \mu_0$, and provided that $|\mu_1| + |\mu_2| < \mu_0$, $\mu(\xi)$ is almost surely positive. We shall set in the following $\mu_0 = 1$, $\mu_1 = 0.62$ and

$\mu_2 = 0.36$. The resulting probability density function (pdf) of the random viscosity is plotted in Figure 1.

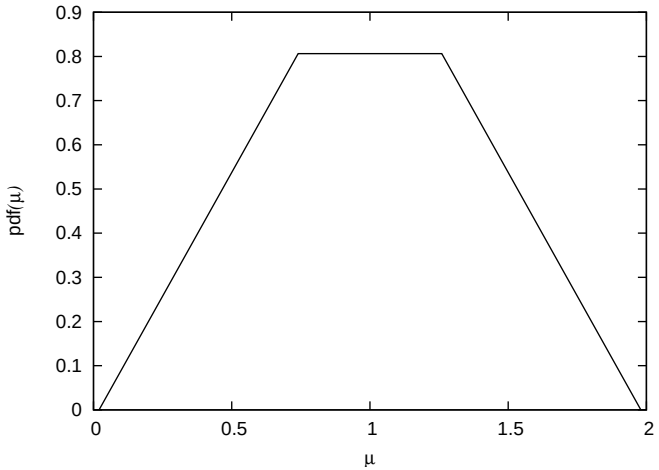


Figure 1. Probability density function of the viscosity.

Finally, we set $x^- = -10$ and $x^+ = 10$ and we use for the boundary conditions,

$$u^- = \frac{1}{2}\left[1 + \tanh\left(\frac{x^-}{4\,\mu_0}\right)\right] \approx 0, \quad u^+ = \frac{1}{2}\left[1 + \tanh\left(\frac{x^+}{4\,\mu_0}\right)\right] \approx 1. \tag{52}$$

For these boundary conditions,

$$u(x) = \frac{1}{2}\left[1 + \tanh\left(\frac{x}{4\,\mu_0}\right)\right],$$

is in fact solution of the deterministic Burgers' equation for $\mu = \mu_0$ [26].

5.1.2. *Variational problems.* The variational formulation of the Burgers' equation is derived. By means of integration by parts, one obtains for the primal problem to be solved for $U \in \mathcal{V}$:

$$A(U;\,\Phi|D) - B(\Phi|D) = \left\langle \int_{\Omega_x} \left[U(1-U) - 2\,\mu\,U_x\right]\,\Phi_x\,dx \right\rangle_{\Omega_\xi} = 0 \quad \forall \Phi \in \mathcal{V}^*, \tag{53}$$

where $\mathcal{V}^* = \mathcal{V}_x^* \otimes \mathcal{V}_\xi$ is constructed using the restriction of $\mathcal{V}_x$ to functions vanishing on $\partial\Omega_x$. For the derivation of the adjoint problem, an obvious choice is here to base the *a posteriori* error estimate on the solution itself, i.e., using

$$\mathcal{J}(U) = \left\langle \int_{\Omega_x} U\,dx \right\rangle_{\Omega_\xi}. \tag{54}$$

For this choice, we have

$$\mathcal{J}'(U;\,\Phi') = \lim_{\varepsilon\to 0} \frac{\mathcal{J}(U+\varepsilon\,\Phi') - \mathcal{J}(U)}{\varepsilon} = \left\langle \int_{\Omega_x} \Phi'\,dx \right\rangle_{\Omega_\xi} \qquad \forall \Phi' \in \mathcal{V}. \tag{55}$$

and

$$A'(U;\,\Phi',\,Z|D) = \lim_{\varepsilon\to 0} \frac{A(U+\varepsilon\,\Phi';\,Z|D) - A(U;\,Z|D)}{\varepsilon} \tag{56}$$

$$= \left\langle \int_{\Omega_x} \left[(1-2\,U)\,Z_x\,\Phi' - 2\,\mu\,Z_x\,\Phi'_x\right] dx \right\rangle_{\Omega_x}. \tag{57}$$

Thus the dual problem can be written as

$$\left\langle \int_{\Omega_x} \left[(1-2\,U)\,Z_x\,\Phi' - 2\,\mu\,Z_x\,\Phi'_x + \Phi'\right] dx \right\rangle_{\Omega_x} = 0 \quad \forall \Phi' \in \mathcal{V}, \tag{58}$$

for $Z \in \mathcal{V}$ and deterministic boundary conditions $Z(x^-) = Z(x^+) = 0$.

For the discretization of the primal and dual problems, we use Chebyshev finite elements to construct $\mathcal{V}^h_x$, and Legendre polynomials (uniform distribution) for $\mathcal{V}^h_\xi$ [1].

To compute the surrogate of the exact adjoint solution, the approximation space $\mathcal{V}^h$ is extended to $\mathcal{V}^{\widetilde{h}}$ by increasing the orders of the Chebyshev (p) and Legendre polynomials (q), as seen in Section 3. This surrogate has to be close enough to the exact adjoint solution to yield correct error estimates through Eq. (46). This is controlled by the construction of $\mathcal{V}^{\widetilde{h}}$. For example, Table 1 shows the error estimate $\eta_{l,m}$, at some element (l, m), obtained using increasing polynomial orders when solving Eq. (58) for the surrogate of the adjoint solution. The convergence of the error estimate is observed. It is seen that increasing the orders to p + 1 and q + 1 provides an estimate within 15% of its "exact" value (taken as achieved for p and q increased by 4). Since the dimension of the stochastic problem quickly increases with the stochastic order, it has been decided to solve $\widetilde{Z}$ with orders p and q increased by one in the following numerical examples. However, if one is willing to pay the price of a better accuracy in the error estimate, we recommend the use of a a larger increase in the polynomial orders, noticing that thanks to the linearity of the dual problem, as seen from Eq. (58), its resolution only contributes to a reduced fraction of the global CPU time.

A fundamental point is that primal and dual problems do not involve any operator in the stochastic directions (derivatives in ξ_i) but in the spatial direction x solely. This has the essential implication that realizations of the Burgers' flow for different realizations of the viscosity are fully independent. As a result, the solution of the primal and dual problems over different SEs are uncoupled, allowing for straightforward parallelization with drastic speed-up of the computation. We took

degree $\Delta(\mathrm{p}, \mathrm{q})$	error estimate
1	$1.0667\ 10^{-3}$
2	$1.2078\ 10^{-3}$
3	$1.2140\ 10^{-3}$
4	$1.2151\ 10^{-3}$

Table 1. Convergence of the error estimate $\eta_{l,m}$ with the increase in the Legendre and Chebyshev polynomial orders to $\mathrm{p} + \Delta\mathrm{p}$ and to $\mathrm{q} + \Delta\mathrm{q}$ when computing the adjoint solution surrogate $\widetilde{Z}$.

advantage of this characteristic by solving SE-wise the primal and dual problems on a Linux-cluster having 4 nodes with dual processors. Another interesting property of the stochastic decoupling between SEs is that, during the refinement process, the approximation needs only to be updated for the stochastic subdomains $\Omega_\xi^{(m)}$ that have been x or ξ-refined.

5.2. ***Isotropic h_ξ-refinement.*** In a first series of tests, the spatial discretization if held fixed with $\mathrm{N_x} = 6$ Chebyshev finite elements having equal size and order $\mathrm{p} = 6$. For the refinement, only h_ξ-refinement is allowed here while the stochastic order is maintained to a constant value.

For the purpose of comparison, we show in Figure 2 the convergence of the error in the computed mean and variance of U at the point $x = 0.52$ when the partition of Ω_ξ is *uniformly* refined by increasing the number $\mathrm{N_b}$ of SEs from 2^2 to 100^2. The mean is given by

$$\left\langle U^h \right\rangle_{\Omega_\xi} = \sum_{m=1}^{\mathrm{N_b}} \left|\Omega_\xi^{(m)}\right| \left\langle U^h \right\rangle_{\Omega_\xi^{(m)}},$$

and the variance by

$$\sigma^2(U) \equiv \left\langle \left[U^h - \left\langle U^h \right\rangle_{\Omega_\xi} \right]^2 \right\rangle_{\Omega_\xi} = \sum_{m=1}^{\mathrm{N_b}} \left|\Omega_\xi^{(m)}\right| \left\langle \left[U^h - \left\langle U^h \right\rangle_{\Omega_\xi} \right]^2 \right\rangle_{\Omega_\xi^{(m)}}. \quad (59)$$

In this experiment, the SEs are squares with equal size. To estimate the errors, surrogates of the exact mean and variance of U were computed using $\mathrm{N_x} = 6$, $\mathrm{p} = 6$, $\mathrm{N_b} = 128^2$ and $\mathrm{q} = 6$. Note that these surrogates are in fact approximations of the exact mean and variance of the *semicontinuous* problem, the spatial discretization being held fixed. Consequently, it is not expected that the *a posteriori* error estimate η goes to zero since a small but finite spatial error persists even for $\mathcal{V}_\xi^h \to \mathcal{V}_\xi$. The plot in Figure 2 shows the convergence of the errors on the mean and variance at $x = 0.52$ of the semicontinuous solution for two stochastic orders $\mathrm{q} = 2$ and $\mathrm{q} = 4$.

The error is seen to quickly decrease as the number of SEs increases, illustrating the convergence of the solution process. The errors on the mean and variance converge with a similar rate which is function of the stochastic order q.

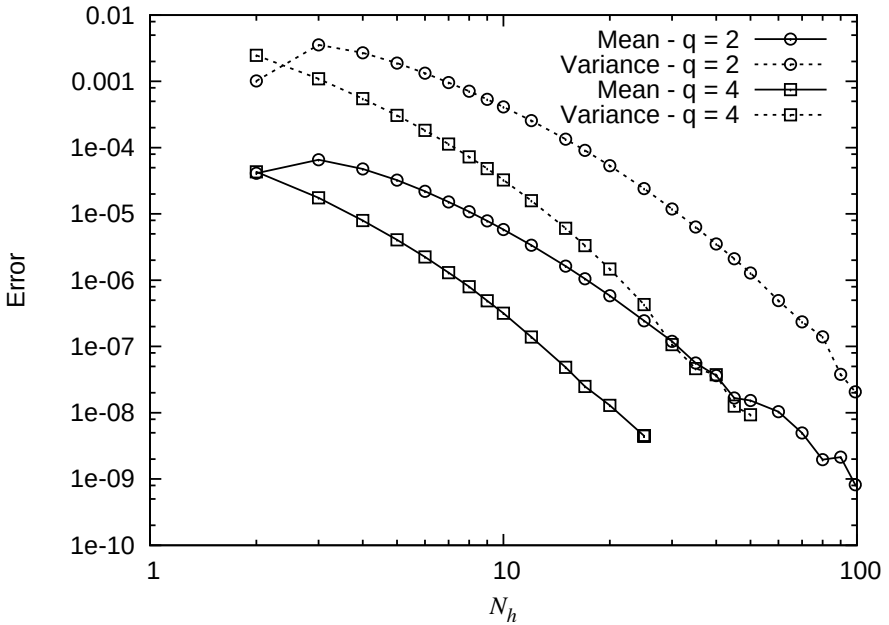


Figure 2. Evolution of the errors on the computed (semicontinuous) mean and variance of the solution at $x = 0.52$ as a function of $N_h = \sqrt{\mathrm{N_b}}$ when using uniform h_ξ-refinement. Two stochastic orders $\mathrm{q} = 2$ and $\mathrm{q} = 4$ are reported as indicated.

However, it is known that this uniform refinement is not optimal, since some areas of Ω_ξ may require a finer discretization than others. Thus, instead of employing a uniform refinement, we now use the *a posteriori* error estimate to identify the SEs requiring refinement. Following Eq. (48), an h_ξ-refinement is to be performed on a SE $\Omega_\xi^{(m)}$ whenever $\eta_{l,m} \geq \epsilon$ for some $l \in [1, \mathrm{N_x} = 6]$. If so, the refinement consists in splitting $\Omega_\xi^{(m)}$ into $2^{\mathrm{N}} = 4$ smaller SEs of equal size (i.e., isotropically). Applying this scheme for $\mathrm{q} = 2$ gives the evolution with the refinement iterations of the errors in the computed mean and variance of U^h at $x = 0.52$ reported in Figure 3. These results were generated using $\epsilon = 2\ 10^{-5}$. The errors are plotted as a function of the total number of dual and primal problems actually solved during the iterative refinement process. The evolution of the errors for the uniform refinement previously shown in Figure 2 is also reported for comparison. A dramatic improvement of the convergence of the errors on the two first moments is observed

when the *a posteriori* error based refinement scheme is used, compared to the uniform refinement. Specifically, an error of $\sim 10^{-7}$ in the (semicontinuous) mean and variance is achieved at a cost of roughly 128 resolutions of the primal and dual problems when using the adaptive h_ξ-refinement, while about 5000 primal problems have to be solved to reach a similar accuracy when using a uniform refinement. Clearly, the adaptive h_ξ-refinement out-performs the uniform refinement, not only in terms of CPU-cost, but also in terms of memory requirements.

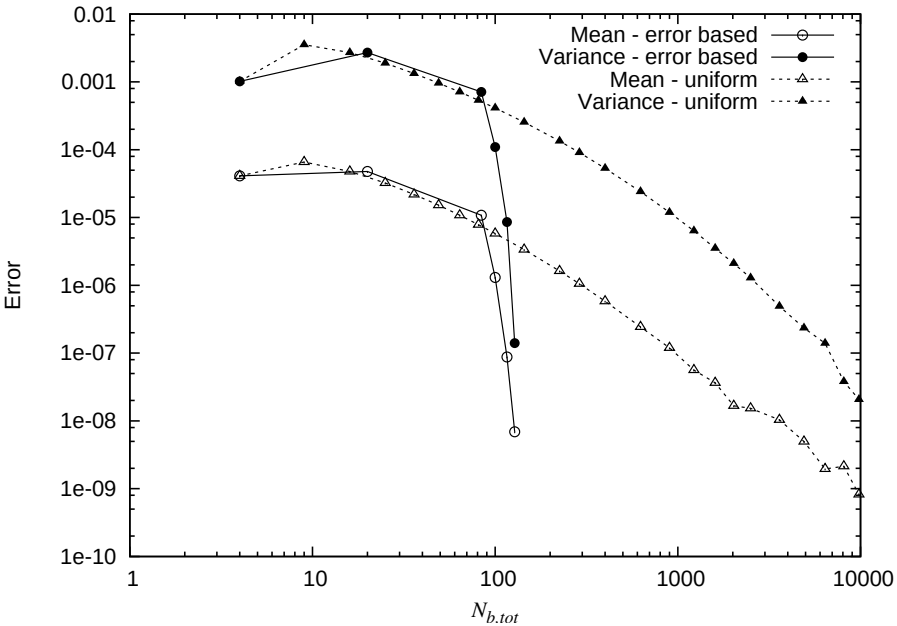


Figure 3. Evolution of the errors in computed (semicontinuous) mean and variance of the solution at $x = 0.52$ as a function of the number of primal and dual problems solves during the isotropic h_ξ-refinement and $\mathrm{q} = 2$. Also plotted are the evolutions of the errors for the uniform refinement.

A better appreciation of the performance of the adaptive h_ξ-refinement can be gained from the analysis of the data reported in Table 2, which presents the evolution of the number $\mathrm{N_b}$ of SEs, the number of resolutions of primal and dual problems and the errors in the first two moments as the refinement proceeds. Starting from a partition of Ω_ξ into 4 equal SEs, they are first all refined along the two-directions ξ_1 and ξ_2 leading to a partition involving 16 SEs. At the second iteration, all these SEs are still considered too coarse to match the prescribed accuracy and are refined again in the two stochastic directions, resulting in 64 SEs. After the third iteration,

only a fraction of the SEs needs further refinement and the process eventually stops after 6 iterations with a partition of the stochastic space into 97 SEs.

Iteration	N_b	# of resolutions	error on mean	error on variance
1	4	4	$4.1074\ 10^{-5}$	$1.0189\ 10^{-3}$
2	16	20	$4.7861\ 10^{-5}$	$2.7054\ 10^{-3}$
3	64	84	$1.0813\ 10^{-5}$	$7.1067\ 10^{-4}$
4	76	100	$1.3056\ 10^{-6}$	$1.0944\ 10^{-4}$
5	88	116	$8.7892\ 10^{-8}$	$8.5915\ 10^{-6}$
6	97	128	$6.9087\ 10^{-9}$	$1.4032\ 10^{-7}$

Table 2. Evolution of the SE discretization (N_b), number of primal and dual problems solves and errors on mean and variance of the solution (at $x = 0.52$), with h_ξ-refinement iteration and $q = 2$.

In a second series of test, the *a posteriori* error based isotropic h_ξ-refinement is applied with different stochastic orders q. The refinement criterion ϵ is increased to 5.10^{-5} while other numerical parameters are kept constant (say $p = 6$, $N_x = 6$). Figure 4 shows the resulting partition of Ω_ξ and surface response of the solution at $x = 0.1$ for $q = 1$, 3 and 5. It is seen that to satisfy the same error criterion a lower number of FEs is necessary when the stochastic order increases. Specifically, for $q = 1$, 174 SEs are needed compared to 10 for $q = 5$. It is also seen that the partition of Ω_ξ is essentially refined in the lower quadrant corresponding to lower values of the viscosity. An asymmetry of the resulting partition of Ω_ξ is also seen for $q = 1$, denoting the different contributions of ξ_1 and ξ_2 to the uncertainty of the solution as one may have expected from the parameterization in Eq. (51).

Furthermore, the surface responses in Figure 4 show that the refinement of Ω_ξ takes place in areas where the solution exhibits the steepest dependence with regard to ξ, but also in areas where it is essentially unaffected by the viscosity; this is due to the fact that the refinement is based on a criterion involving all spatial locations: the solution at different spatial locations requires refinement at different places in Ω_ξ.

5.3. ***Isotropic $h_{\xi,x}$-refinement.*** In the previous tests, an isotropic h_ξ-refinement only was applied. However, as discussed previously, the *a posteriori* error estimate incorporate both the stochastic and spatial errors. In fact, it is expected that when lowering μ a finer and finer spatial FE discretization in the neighborhood of $x = 0$ is needed as the solution becomes stiffer and stiffer. Consequently, one may find advantages in adapting the FE discretization to $\Omega_\xi^{(m)}$. This is achieved by introducing an additional test before applying the isotropic h_ξ-refinement. If the local error $\eta_{l,n}$

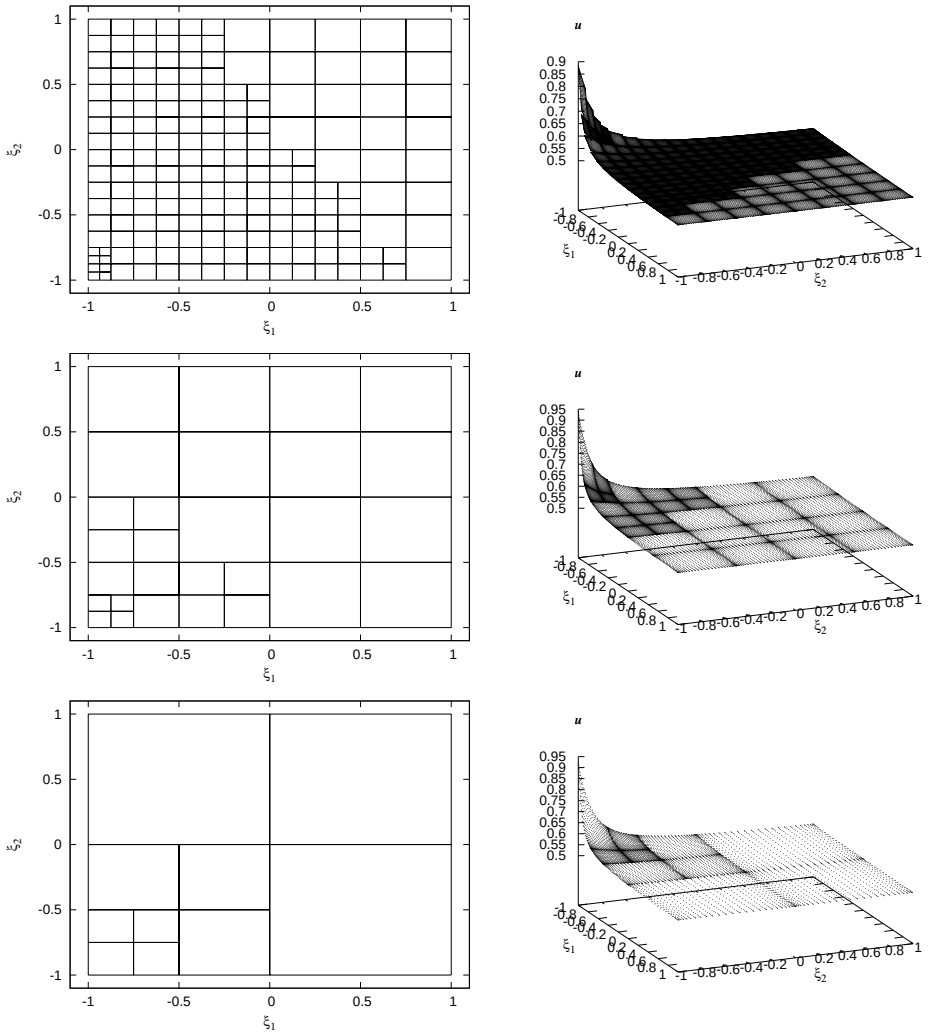


Figure 4. Partition of Ω_ξ (left) and surface response for $U(\xi)$ at $x = 0.1$ (right) at the end of the isotropic h_ξ-refinement process using $\epsilon = 5.10^{-5}$. Plots correspond to q = 1, 3 and 5 from top to bottom.

is greater than ϵ, the spatial discretization is first checked by computing an estimate of the spatial error $\eta^x_{l,m}$ from

$$\left(\eta^x_{l,m}\right)^2 = \int_{\Omega_x^{(l)}} \left\langle \left[U^h - \Pi^l(U^h)\right]^2 \right\rangle_{\Omega_\xi^{(m)}} dx, \tag{60}$$

where $\Pi^l(U^h)$ is the (spatial) Clément interpolant [6] of U^h over the spatial patch defined by the union of the FEs having a common point with the element $\Omega_x^{(l)}$. The order of the Clément interpolant is set to $\mathrm{p}(l,m)+1$. If this estimate of the spatial error is greater than a prescribed second threshold ϵ_x a h_x-refinement is applied to the FE $\Omega_x^{(l)}$ (for the SE $\Omega_\xi^{(m)}$ only), consisting in its partition into two Chebyshev elements of equal size. On the contrary, if $\eta^x_{l,m} < \epsilon_x$ for all $l \in [1, \mathrm{N_b}(m)]$, the h_ξ-refinement is applied as previously.

This strategy is applied to the test problem, with the initial discretization using $\mathrm{N_x} = 6$ identical FEs with $\mathrm{p} = 6$, over 4 equal SEs with $\mathrm{q} = 2$ and a refinement criteria $\epsilon = 10^{-4}$. The partition of Ω at the end of the refinement process is shown in Figure 5. The left plot shows the partition of Ω_ξ and highlights again the need for refinement for the lowest values of the viscosity. The right plot shows the dependence of the refinement of the FE discretization with ξ. Specifically, it is seen that h_x-refinement essentially occurs for the lowest values of the viscosity (i.e., when the solution exhibits the steepest spatial evolutions) and in the neighborhood of $x = 0$ as one may have expected.

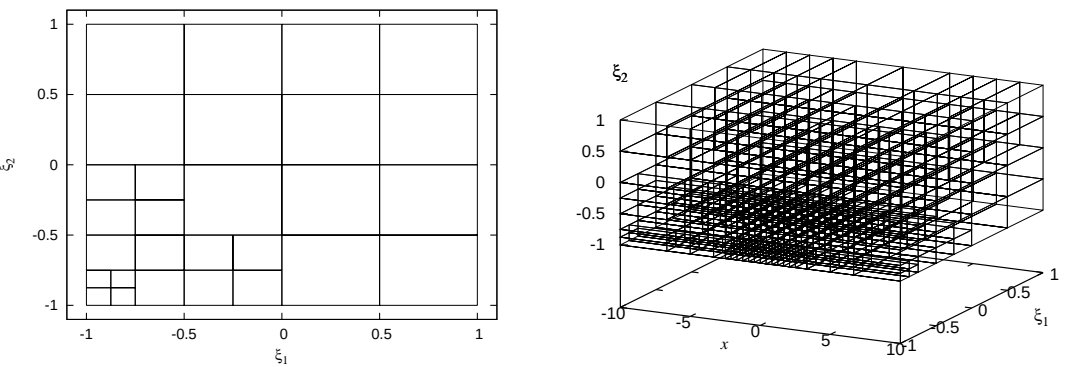


Figure 5. Partition of Ω_ξ (left) and Ω (right) after the $h_{\xi,x}$-refinement procedure. Numerical parameters are given in the text.

Additional insights about the distribution of the local *a posteriori* error estimate $\eta_{l,m}$ in Ω can be gained examining Figure 6, where plotted is the local error magnitude as spheres. A large sphere corresponds to a large error $\eta_{l,m}$, with a scaling of the spheres' diameter as $d \sim \eta_{l,m}^{0.25}$. As already stated, it is seen that the maximum error occurs around $x = 0$ and that it decreases very quickly as one gets away from that location. This plot clearly exemplifies the h-refinement strategy: divide elements where a large error occurs to make the error magnitude below the prescribed tolerance $\eta_{l,m}$.

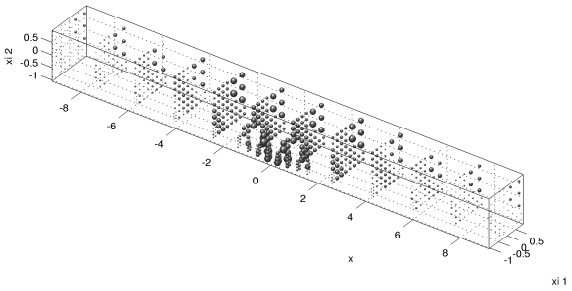


Figure 6. Distribution of the local *a posteriori* error estimate $\eta_{l,m}$ after $h_{\xi,x}$-refinement. The spheres' diameter d scales as $d \sim \eta_{l,m}^{0.25}$.

We present in Figure 7 the expectation (left) and variance (right) of the approximate solution U^h after refinement as a function of x. The plot of the expectation $\langle U^h \rangle_{\Omega_\xi}$ is also compared with the deterministic solution $u(x)$ for the mean viscosity $\mu_0 = 1$. This deterministic solution has for expression:

$$u(x;\, \mu = 1) = \frac{1}{2}\left[1 + \tanh\frac{x}{4}\right]. \tag{61}$$

It is seen that the expected solution also has an hyperbolic tangent-like profile but is not equal to the deterministic solution: the differences are due to the nonlinearities of the Burgers' equation. The right plot in Figure 7 depicts the solution variance $\sigma^2(U^h)$. The boundary conditions being deterministic the variance vanishes at x^- and x^+. The uncertainty in the viscosity produces a symmetric variance with regard to $x = 0$ as it only affects the steepness of the hyperbolic tangent-like profile since

$$U(x, \xi) \approx \frac{1}{2}\left[1 + \tanh\frac{x}{4\,\mu(\xi)}\right]. \tag{62}$$

Also, due to the selected boundary conditions, we have at the center of the spatial domain $U(\xi) = (u^- + u^+)/2 = 1/2$ almost surely, provided that $\mu(\xi) > 0$. Therefore, the variance of U^h vanishes at $x = 0$ as shown in Figure 7.

The probability density functions of U^h, together with the solution's quantiles, are reported in Figure 8 as functions of x. The quantiles are defined as the level $u(Q)$, for $Q \in]0, 1[$, such that the probability of $U^h(x) < u$ is equal to Q. The plot of the pdf shows dramatic changes with x. For $x = x^-$ the pdf is a Dirac of unit mass (no-uncertainty); then when x increases the pdf evolves from a sharp lower tail distribution to a long lower tail distribution. At $x = 0$ it is again a Dirac (no-uncertainty). For x increasing further to x^+ the opposite evolution is observed (due to the central symmetry of the settings). Note that the distribution of the solution is bounded since U almost surely $\in [u^-, 1/2]$ for $x \leq 0$ and U almost

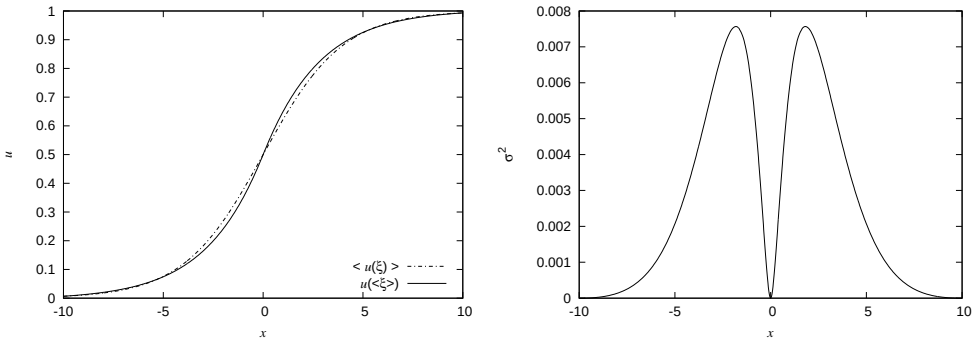


Figure 7. Expectation (left) and variance (right) of the approximate solution $U^h(x,\xi)$ at the end of the $h_{\xi,x}$-refinement process.

surely $\in [1/2, u^+]$ for $x \geq 0$. The quantiles reflect the complexity of the distribution with important changes with x of the spacing between quantiles.

To further illustrate the need of refinement to properly capture the solution distribution, we present in Figure 9 the convergence of the pdf of U^h at $x = 0.52$ along the $h_{\xi,x}$-refinement process. The left plot shows the pdf in linear-log scales to appreciate the improvement in the tails of the distribution, while the right plot in linear-linear scales shows the improvement in the high density region. It is seen that during the first iterations of the refinement process the pdf presents under-estimated right-tails and some spurious oscillations, which are due to discontinuities of the approximate solution across SEs boundaries.

5.4. ***Anisotropic h/q-refinement.*** In the previous tests, an isotropic h-refinement was used in the stochastic domain. As a result, each refined SE is split into 2^N SEs. For large N this simple procedure quickly results in a prohibitively large number of SEs. Instead, one finds advantage in splitting $\Omega_\xi^{(m)}$ only along the stochastic directions yielding the largest error reduction. Obviously, the *a posteriori* error estimate does not provide enough information to decide along which directions $\Omega_\xi^{(m)}$ should be split: an anisotropic error estimator is necessary to this end. In the absence of an such estimator, we rely on a criterion, inspired from [16; 27], which is based on the relative contributions of each stochastic directions to the local variance. The local variance is defined as

$$\sigma^2_{\Omega_\xi^{(m)}}(U) = \left\langle \left[U - \langle U \rangle_{\Omega_\xi^{(m)}} \right]^2 \right\rangle_{\Omega_\xi^{(m)}}. \tag{63}$$

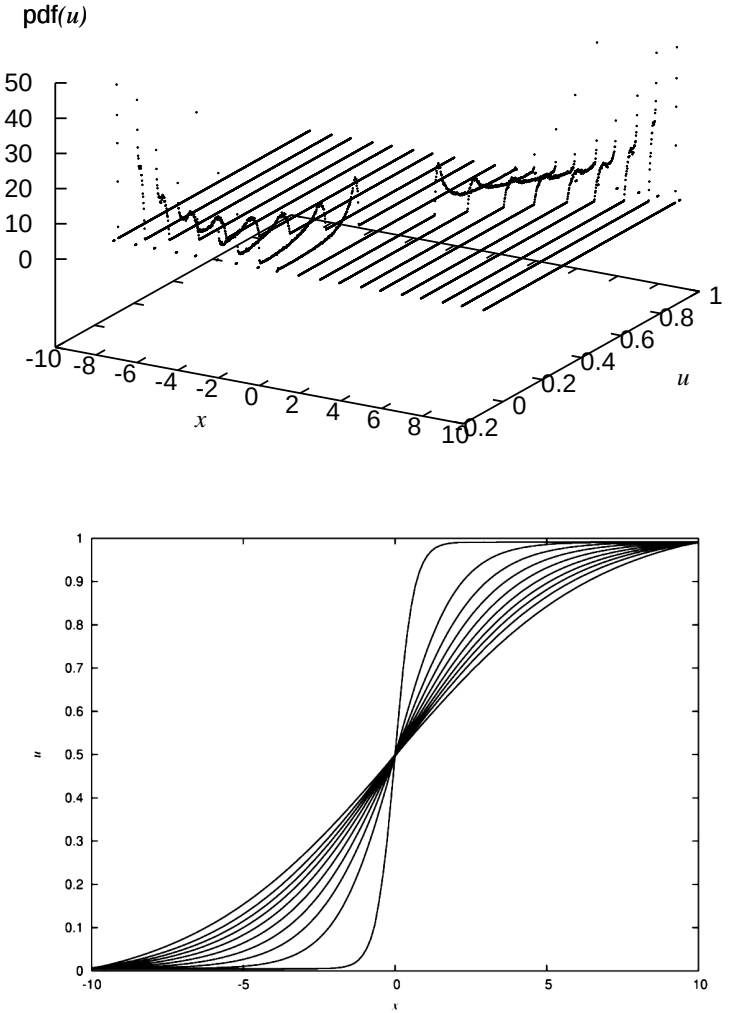


Figure 8. Top: pdf of the approximate solution U^h as a function of x at the end of the $h_{\xi,x}$-refinement. The pdf-axis is truncated for clarity. Bottom: quantiles $u(Q)$ of the solution, as a function of x, for $Q = 0.05$ to 0.95 with constant increment $\Delta Q = 0.1$.

Since the stochastic expansion of U over $\Omega_\xi^{(m)}$ is of the form

$$U(\xi \in \Omega_\xi^{(m)}) = \sum_{k=0}^{\mathrm{P}(m)} u_k^{(m)} \Psi_k^{(m)}(\xi),$$

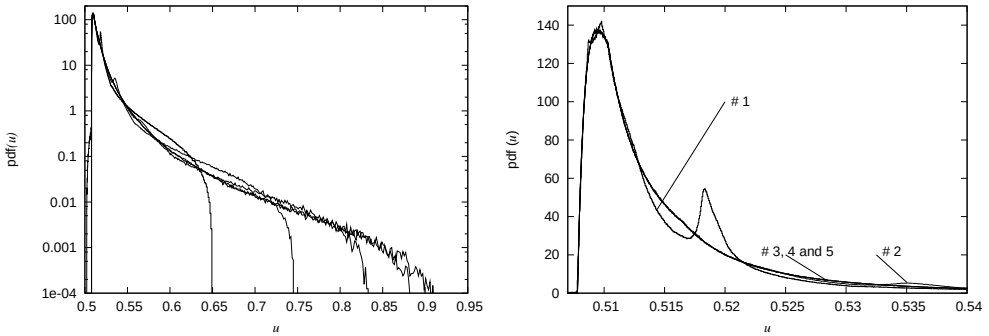


Figure 9. Probability density function at $x = 0.52$ for different steps of the $h_{\xi,x}$-refinement process. Linear-log plot (left) and linear-linear plot (right).

and because by convention $\Psi_k^{(m)} = 1$ for $k = 0$ (i.e., mode 0 is the mean mode), the local variance becomes

$$\sigma^2_{\Omega_\xi^{(m)}}(U) = \sum_{k=1}^{\mathrm{P}(m)} \left(u_k^{(m)}\right)^2 \left\langle \Psi_k^{(m)2} \right\rangle_{\Omega_\xi^{(m)}}, \tag{64}$$

and we define

$$\sigma^2_{\Omega_\xi^{(m)} \times \Omega_x^{(l)}}(U) = \sum_{k=1}^{\mathrm{P}(m)} \left\langle \Psi_k^{(m)2} \right\rangle_{\Omega_\xi^{(m)}} \int_{\Omega_x^{(l)}} \left(u_k^{(m)}(x)\right)^2 dx. \tag{65}$$

It is seen that the integral of the local variance on the FE $\Omega_x^{(l)}$ is a weighted sum of the integral of the squared stochastic expansion coefficients over the FE. The idea is thus to define, for each direction $i = 1, \ldots, \mathrm{N}$, the contribution of the polynomial of degree $\mathrm{q}(m)$ in ξ_i to this variance integrated on $\Omega_x^{(l)}$. This contribution is denoted $\sigma^2_{l,m}(U; i, \mathrm{q}(m))$. Using the respective contributions of each direction, it is decided that $\Omega_\xi^{(m)}$ has to be split along the i-th stochastic direction if the following test is satisfied for at least one FE:

$$\frac{\sigma^2_{l,m}(U; i, \mathrm{q}(m))}{\sum_{i=1}^{\mathrm{N}} \sigma^2_{l,m}(U; i, \mathrm{q}(m))} \geq \epsilon_2. \tag{66}$$

where $0 < \epsilon_2 < 1$ is an additional threshold parameter. If none of the stochastic directions satisfies the previous test, it is on the contrary decided to increment by one unit the stochastic expansion order $\mathrm{q}(m)$ over $\Omega_\xi^{(m)}$.

The anisotropic h/p-refinement strategy now follows the general algorithm:

1. solve the primal and dual problems for the current approximation space $\mathcal{V}^h$; get U^h and Z^h.
2. Solve the adjoint problem in the enriched space $\mathcal{V}^{\widetilde{h}}$; get $\widetilde{Z}$.
3. Compute the local error $\eta_{l,m}$ from Eq. (46) for $m = 1, \ldots, \mathrm{N_b}$ and $l = 1, \ldots, \mathrm{N_x}(m)$.
   If $\eta_{l,m} < \epsilon$ for $m = 1, \ldots, \mathrm{N_b}$, $l = 1, \ldots, \mathrm{N_x}(m)$, then end computation.
4. For $m = 1, \ldots, \mathrm{N_b}$ and $l = 1, \ldots, \mathrm{N_x}(m)$
   If $\eta_{l,m} > \epsilon$:
   a. Compute the estimate of the spatial error $\eta^x_{l,m}$ using Eq. (60).
   b. If $\eta^x_{l,m} > \epsilon_x$, mark element for h_x-refinement.
   c. If the element has not been marked for h_x-refinement,
      (a) Compute the directional variances.
      (b) For $i = 1, \ldots, \mathrm{N}$ if the directional variance is greater than ϵ_2 then mark element $\Omega^{(m)}_\xi$ for h_ξ-refinement in direction i.
5. For $m = 1, \ldots, \mathrm{N_b}$: if $\Omega^{(m)}_\xi$ has not been marked for some h_ξ-refinement, and none of the elements $\Omega^{(m)}_\xi \times \Omega^{(l)}_x$, $l = 1, \ldots, \mathrm{N_x}(m)$, are marked for h_x-refinement but there exists at least one $l \in [1, \mathrm{N_x}(m)]$ such that $\eta_{l,m} > \epsilon$, then increase $\mathrm{q}(m)$ by one.
6. Construct the refined approximation space and restart from 1.

This refinement scheme has been successfully applied to the test problem, with $\mu_1 = 0.82$ and $\mu_2 = 0.16$. The viscosity parameterization was changed to increase the contribution of the first direction compared to the second to the solution uncertainty. Note that the pdf of μ is affected by this change of the parameterization, but the uncertainty range is kept constant. For illustration purposes, we present in Figure 10 an example of the partition of the stochastic space into SEs with variable stochastic expansion orders. The initial discretization involves $\mathrm{N_b} = 4$ equal SEs with $\mathrm{q} = 2$. At the first iteration, all SEs were split isotropically, the expansion order being kept constant. At the second iteration, the SEs with boundary at $\xi_1 = -1$ were further refined but in the ξ_1 direction only. For the following iterations, no further h_ξ-refinement was required while some SEs still have a significant estimated error: it yielded increase in the stochastic expansion order $\mathrm{q}(m)$. Again, the final expansion order is the greatest for the SEs with $\xi_1 = -1$ and/or $\xi_2 = -1$ boundaries (where viscosity is small), and is the lowest for the SE having boundary $\xi_1 = 1$ and $\xi_2 = 1$ where q has been kept constant.

5.5. ***Tests for*** $\mathbf{N = 3}$**.** To conclude this series of tests, an additional uncertainty source is considered by taking the left boundary condition as random, U^-. The random boundary condition is assumed independent of the viscosity value and

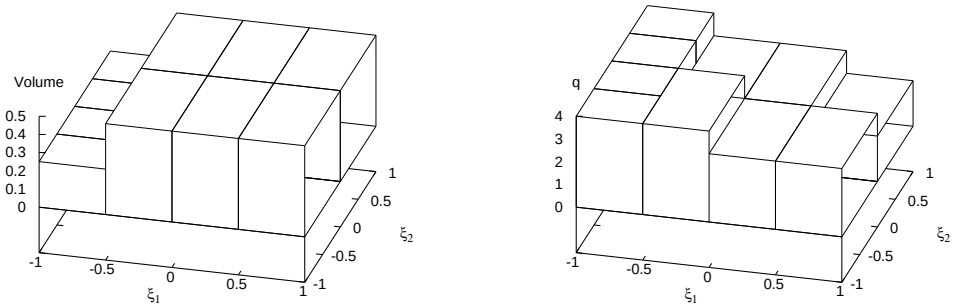


Figure 10. Volume $\left|\Omega_\xi^{(m)}\right|$ and corresponding stochastic expansion order q over the partition of Ω_ξ. Anisotropic h/q-refinement.

consequently parameterized using an additional random variable ξ_3. The complete uncertainty settings are:

$$\mu(\xi) = 1 + 0.5\,\xi_1 + 0.05\,\xi_2, \quad U^-(\xi) = u_0^- + u'\,\xi_3, \tag{67}$$

with u_0^- given by Eq. (52) and $u' = 5.\,10^{-4}$. This low value of u' is selected as it is known that small perturbations of the boundary condition leads to $O(1)$ changes in the solution of the Burgers' equation (see [31]). This is due to the "supersensitivity" of the transition layer location with the boundary condition: the low variability in U^- will result in large variability of the solution but essentially around the center of the spatial domain and not in the neighborhood of x^- where the solution variability is low. This problem is thus well suited to test the effectiveness of the *a posteriori* error methodology in providing correct local error estimators. Moreover, as the sensitivity of the solution with regard to U^- increases when the viscosity is lowered, a finer partition of Ω_ξ is expected for low values of ξ_1, while the contribution of ξ_2 will be less as seen from Eq.(67).

The spatial discretization ($\mathrm{N_x} = 20$, $\mathrm{p} = 6$) and stochastic orders q being held fixed, we proceed with the previously described *a posteriori* error based anisotropic h_ξ-refinement scheme. The target precision is set to $\epsilon_\eta = 0.001$. In Figure 11 we show the reduction of the *a posteriori* error η along the refinement process for orders $\mathrm{q} = 1$ and 2. The evolution of the error estimate for a uniform refinement of the stochastic space is also reported for comparison. Because the stochastic space now has 3 dimensions, the increase in number of SEs for the uniform refinement is seen to be dramatically large for a low resulting reduction of the *a posteriori* error. On the contrary, using the local error estimate to guide the refinement process is seen to significantly improve the error reduction with the number of SEs. It is

also remarked that the anisotropic refinement requires 3 iterations to achieve the prescribed precision for $q = 1$, while only 2 iterations are needed for $q = 2$.

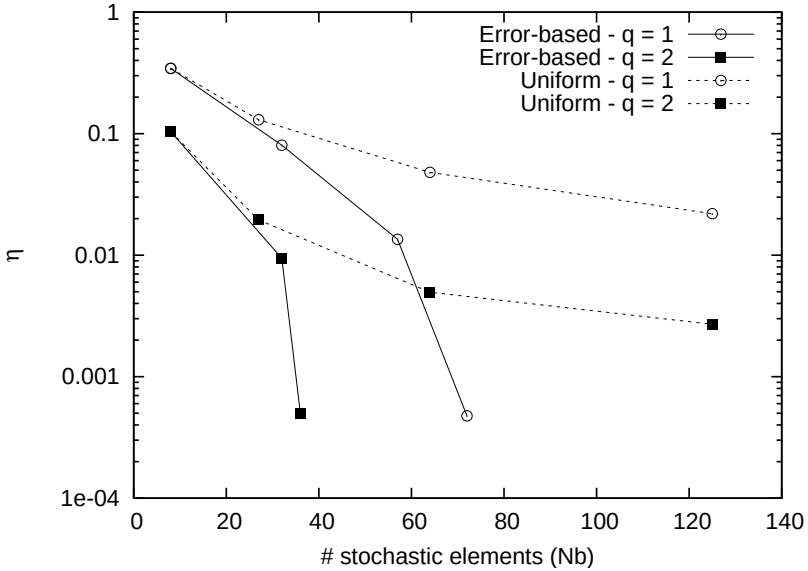


Figure 11. Reduction of the *a posteriori* error estimate η with the number N_b of stochastic elements involves in the partition of Ω_ξ. Plotted are the results for the anisotropic h_ξ-refinement procedure (labeled Error-based) and uniform refinement, using $q = 1$ and 2 as indicated.

Figure 12 depicts the partition of the stochastic space at the end of the h_ξ-refinement process. The initial partition uses $N_b = 2^N = 8$ identical SEs. In fact, the anisotropic h_ξ-refinement process never requires refinement along the second dimension ξ_2: the plots of Figure 12 thus show the partition of Ω_ξ in a plane where ξ_2 is constant. The independence of the partition with regard to ξ_2 denotes the capability of the proposed scheme to detect the weak influence of ξ_2 on the solution. On the contrary, it is seen that for fixed ξ_2 and ξ_3 a finer division of Ω_ξ along the first direction is necessary when ξ_1 decreases, because of the steeper behavior of the solution when the viscosity decreases. In contrast, for fixed ξ_1 and ξ_2 the partition is uniform along the third direction, but is finer for low viscosity and $q = 1$, as one may have anticipated from the behavior of the Burgers' solution.

To conclude these tests, we show in Figure 13 the variance of the stochastic solution along the spatial domain, for the two stochastic orders $q = 1$ and 2, at the end of the anisotropic refinement process. The effect of the uncertain boundary condition on the solution variance can be appreciated through comparison with

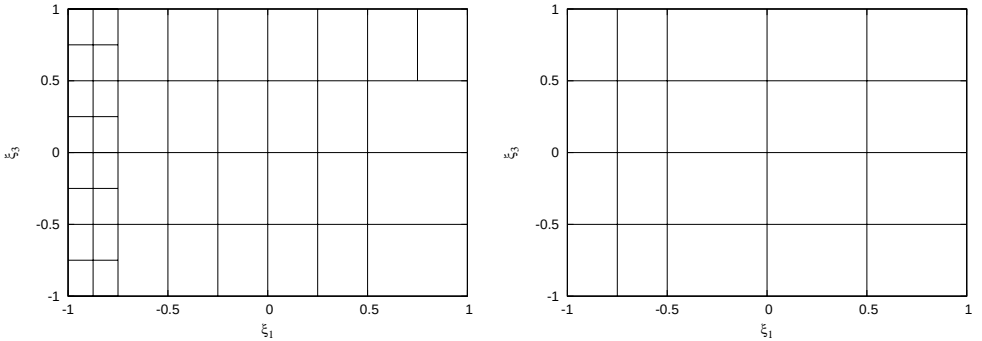


Figure 12. Partition of Ω_ξ at the end of the anisotropic h_ξ-refinement process in a plane corresponding to constant ξ_2. Left: $q = 1$ and right $q = 2$.

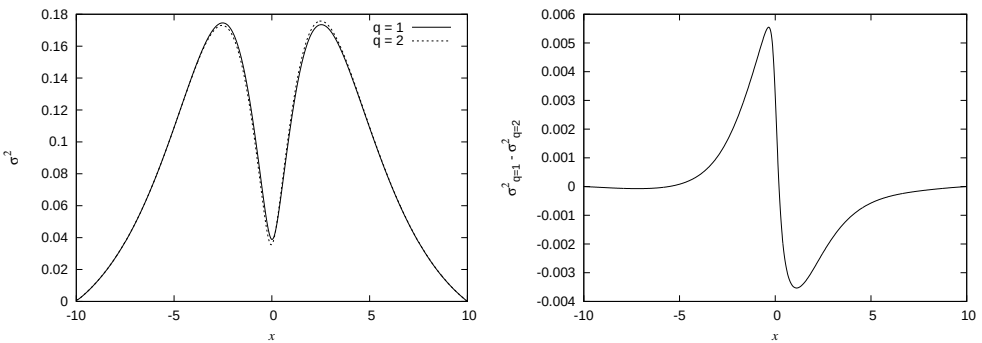


Figure 13. Left: comparison of the variances in $U(\xi)$ as a function of x for stochastic expansion orders $q = 1$ and $q = 2$ at the end of the anisotropic h_ξ-refinement process. Right : difference of the two variances predicted for $q = 1$ and $q = 2$.

the result reported in Figure 7. Specifically, the variance of the solution at the center of the spatial domain is now different from zero. It is seen that even so both orders leads to similar estimated error, small but noticeable differences are visible in the spatial distribution of the solution variance. These difference in terms of predicted variance can be better appreciated from the right plot in Figure 13 where the differences for $q = 1$ and $q = 2$ are plotted.

6. Concluding remarks

A dual-based *a posteriori* error analysis has been proposed in the context of stochastic finite element methods with stochastic discretization involving piecewise

continuous orthogonal polynomials approximations. The error estimation involves the resolution of a linear stochastic dual problem, which computational cost is deemed negligible compared to the primal problem (provided the latter is nonlinear). Numerical tests on the uncertain Burgers' equation have demonstrated the effectiveness of the methodology in providing relevant error estimates that can be localized in the spatial and stochastic domain.

The principal limitation of the proposed method is the lack of resulting information regarding the structure of the estimated error. Specifically, the respective contributions of the spatial and stochastic approximations to the estimated error are not accessible. At a finer level, the error estimator does not allow for the discrimination between the relative contributions of the stochastic directions to the overall error. We believe this is the most severe limitation in view of anisotropic refinement of the stochastic approximation space required to treat problems with high dimensional uncertainty germs. However, we consider that the proposed methodology constitutes a significant improvement compared to error indicators previously proposed in the stochastic context [16; 17; 27], which were based on the spectrum of the local stochastic expansion.

Several potential improvements of the refinement strategy have been identified throughout this work. It includes the derivation of rigorous and efficient anisotropic error estimators for high order approximation schemes. Another area of potential application of the *a posteriori* estimator is the coarsening of the approximation space in view of application to, say, unsteady flows. Both of these developments are the subject of on-going work.

References

[1] M. Abramowitz and I. Stegun, *Handbook of mathematical functions*, Dover, 1970.

[2] I. Babuška and A. Miller, *A feedback finite element method with a posteriori error estimation. I. The finite element method and some basic properties of the a posteriori error estimator*, Comput. Methods Appl. Mech. Engrg. **61** (1987), no. 1, 1–40. MR 88d:73036 Zbl 0593.65064

[3] I. Babuška and W. Rheinboldt, *A posteriori error estimates for the finite element method*, Int. J. Numer. Meth. Engrg. **12** (1978), 1597–1615.

[4] R. Becker and R. Rannacher, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numer. **10** (2001), 1–102. MR 2004g:65147 Zbl 1047.76016

[5] R. H. Cameron and W. T. Martin, *The orthogonal development of non-linear functionals in series of Fourier–Hermite functionals*, Ann. of Math. (2) **48** (1947), 385–392. MR 8,523a Zbl 0029.14302

[6] P. Clément, *Approximation by finite element functions using local regularization*, Rev. Française Automat. Informat. Recherche Opérationnelle Sér. Rouge Anal. Numér. **9** (1975), no. R-2, 77–84. MR 53 #4569 Zbl 0368.65008

[7] B. Debusschere, H. Najm, A. Matta, O. Knio, R. Ghanem, and O. Le Maître, *Protein labelling reactions in electrochemical flow: numerical simulation and uncertainty propagation*, Phys. Fluids **15** (2003), no. 8, 2238–2250.

[8] R. Ghanem and S. Dham, *Stochastic finite element analysis for multiphase flow in heterogeneous porous media*, Transp. Porous Media **32** (1998), no. 3, 239–262. MR 2001e:76085

[9] R. G. Ghanem, *Probabilistic characterization of transport in heterogeneous media*, Comp. Meth. App. Mech. Eng. **158** (1998), 199–220.

[10] R. G. Ghanem and P. D. Spanos, *Stochastic finite elements: a spectral approach*, Springer, New York, 1991. MR 91k:73102

[11] V. Heuveline and R. Rannacher, *Duality-based adaptivity in the hp-finite element method*, J. Numer. Math. **11** (2003), no. 2, 95–113. MR 2004m:65196 Zbl 1050.65111

[12] T. Hien and M. Kleiber, *Stochastic finite element modeling in linear transcient heat transfer*, Comp. Met. App. Mech. Eng. **144** (1997), 111–124.

[13] M. Kaminski and T. Hien, *Stochastic finite element modeling of transcient heat transfer in layered composites*, Int. Com. Heat and Mass Trans. **26** (1999), no. 6, 801–810.

[14] O. M. Knio and O. P. Le Maître, *Uncertainty propagation in CFD using polynomial chaos decomposition*, Fluid Dynam. Res. **38** (2006), no. 9, 616–640. MR 2007f:76158

[15] O. P. Le Maître, O. M. Knio, H. N. Najm, and R. G. Ghanem, *Uncertainty propagation using Wiener–Haar expansions*, J. Comput. Phys. **197** (2004), no. 1, 28–57. MR 2005a:76123

[16] O. P. Le Maître, H. N. Najm, R. G. Ghanem, and O. M. Knio, *Multi-resolution analysis of Wiener-type uncertainty propagation schemes*, J. Comput. Phys. **197** (2004), no. 2, 502–531. MR 2005b:65142

[17] O. P. Le Maître, H. N. Najm, P. P. Pébay, R. G. Ghanem, and O. Knio, *Multi-resolution analysis for uncertainty quantification in chemical systems*, SIAM J. Sci. Comp. (2007), in press.

[18] O. Le Maître, M. T. Reagan, B. Debusschere, H. N. Najm, R. G. Ghanem, and O. M. Knio, *Natural convection in a closed cavity under stochastic non-Boussinesq conditions*, SIAM J. Sci. Comput. **26** (2004), no. 2, 375–394. MR 2005i:76094

[19] O. P. Le Maître, O. M. Knio, H. N. Najm, and R. G. Ghanem, *A stochastic projection method for fluid flow, I: Basic formulation*, J. Comput. Phys. **173** (2001), no. 2, 481–511. MR 2002k:76111

[20] O. P. Le Maître, M. T. Reagan, H. N. Najm, R. G. Ghanem, and O. M. Knio, *A stochastic projection method for fluid flow. II. Random process*, J. Comput. Phys. **181** (2002), no. 1, 9–44. MR 2003g:76090

[21] L. Mathelin, M. Y. Hussaini, and T. A. Zang, *Stochastic approaches to uncertainty quantification in CFD simulations*, Numer. Algorithms **38** (2005), no. 1-3, 209–236. MR 2006a:76089 Zbl 02161195

[22] S. Micheletti, S. Perotto, and M. Picasso, *Stabilized finite elements on anisotropic meshes: a priori error estimates for the advection-diffusion and the Stokes problems*, SIAM J. Numer. Anal. **41** (2003), no. 3, 1131–1162. MR 2004g:65157 Zbl 1053.65089

[23] J. Peraire, J. Peiró, and K. Morgan, *Adaptive remeshing for three-dimensional compressible flow computations*, J. Comp. Phys. **103** (1992), no. 2, 269–285.

[24] M. T. Reagan, H. N. Najm, R. G. Ghanem, and O. M. Knio, *Uncertainty quantification in reacting flow simulations through non-intrusive spectral projection*, Combust. Flames **132** (2003), 545–555.

[25] G. Schuëller, *Computational stochastic mechanics: recent advances*, Comput. Struct. **79** (2001), 2225–2234.

[26] R. Walters and L. Huyse, *Uncertainty analysis for fluid mechanics with applications*, Tech. report, NASA CR-2002-211449, 2002.

[27] X. Wan and G. E. Karniadakis, *An adaptive multi-element generalized polynomial chaos method for stochastic differential equations*, J. Comput. Phys. **209** (2005), no. 2, 617–642. MR 2006e:65007 Zbl 1078.65008

[28] N. Wiener, *The homogeneous chaos*, Amer. J. Math. **60** (1938), no. 4, 897–936. Zbl 0019.35406 JFM 64.0887.02

[29] D. Xiu and G. E. Karniadakis, *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput. **24** (2002), no. 2, 619–644. MR 2003m:60174 Zbl 1014.65004

[30] ———, *Modeling uncertainty in flow simulations via generalized polynomial chaos*, J. Comput. Phys. **187** (2003), no. 1, 137–167. MR 2004c:76110 Zbl 1047.76111

[31] ———, *Supersensitivity due to uncertain boundary conditions*, Internat. J. Numer. Methods Engrg. **61** (2004), no. 12, 2114–2138. MR 2005f:76078 Zbl 1075.76623

Received November 28, 2006. Revised July 25, 2007.

LIONEL MATHELIN: mathelin@limsi.fr
LIMSI - CNRS, BP 133, 91403 Orsay cedex, France
http://www.limsi.fr/Individu/mathelin

OLIVIER LE MAÎTRE: olm@iup.univ-evry.fr
Laboratoire de Mécanique et d'Energétique d'Evry and LIMSI- CNRS, 40, rue du Pelvoux, CE 1455 Courcouronnes, 91020 Evry cedex, France
http://gmfe16.cemif.univ-evry.fr:8080/~olm/

COMM. APP. MATH. AND COMP. SCI.
Vol. 2, No. 1, 2007

HYBRID NUMERICAL TREATMENT OF TWO-FLUID PROBLEMS WITH PASSIVE INTERFACES

NICHOLAS G. COGAN

We consider the coupled motion of a passive interface separating two immiscible fluids of different viscosities. There are several applications where the velocity of the two fluids is needed everywhere within the domain. Examples include the transport of bacteria and diffusing substances within a biofilm matrix and the transport of cations throughout the mucociliary and periciliary layer in the lung lining. In this investigation, we use a hybrid approach which employs the boundary integral method to determine the interface velocity and the method of regularized stokeslets to determine the velocity elsewhere in the domain.

Our approach capitalizes on the strengths of the two methods, yielding an intuitive, efficient procedure for determining the velocity of a two-fluid system throughout the domain. A key feature of the method is the extension to two-fluid systems with varying viscosity. We describe the results of three numerical simulations designed to test the numerical method and motivate its use.

1. Introduction

We consider the dynamics of two immiscible fluids of different viscosities separated by a passive interface. The motion of both fluids is assumed to be dominated by viscosity and described mathematically by the incompressible Stokes equations. In the absence of a background flow the fluids move because of stress differences at the interface that arise due to different viscosities. Due to the linearity of Stokes equations, the dynamics of the system with a background flow is the superposition of the motion due to jump in the traction across the interface and the background motion of the fluids.

Though the fluid equations are linear, the coupling between two fluids renders the full system nonlinear. Moreover, the interface between the fluids is typically heterogeneous and not aligned with a regular grid, complicating numerical approximations as well as being time-dependent. In particular, standard grid-based methods are

MSC2000: 76B70.
Keywords: two-fluid, boundary integral method, regularized stokeslets, biofilm.
This work is supported in part by NSF grant DMS #0548511.

prone to large approximation errors while standard finite element methods require substantial computation in order to triangulate the dynamic domain.

Several methods have been introduced to address these issues [2; 10; 11; 12; 17]. One class of these methods transforms the equations into integral equations and then attempts to solve the integral equations as accurately as possible, with the least computational effort. In [2; 10], methods are developed to produce solutions that are second order accurate both on and near the interface. The main idea is to take a standard quadrature approximation of the solution to the integral equation and add a correction term. This correction term is computed using asymptotic analysis of the error due to smoothing of the nearly singular kernel and that due to discretization of the integral equation. It is shown that a second order accurate method can be developed in this manner. In [12], the computational effort was addressed. A fast multipole method was introduced to solve the integral equation associated with the bi-harmonic equation. This method requires $\mathcal{O}(N)$ operations, where N is the number of points in the discretization.

Rather than transform the equations into integral equations, a different class of methods attempts to solve the associated primitive variable formulation using finite difference methods. For typical applications there is a jump in the material properties across a curve in the domain. This jump introduces unacceptable errors in the solution via standard finite difference discretization. In [11; 17], the standard finite difference stencil is altered for points near the interface in order to maintain the accuracy. LeVeque and Li [11] introduced a method incorporating the interface jump condition into the discretized Laplacian operator. The resulting scheme is second order accurate. Wei et al. [17] introduced a method termed the matched interface and boundary method. Rather than use the jump condition to alter the discretization, this method uses the jump conditions iteratively. At irregular points on the grid (i.e., those whose finite difference stencil overlaps the two regions), values of the stencil are given at fictitious grid points and are iteratively determined to satisfy the lowest order approximate jump condition. This subtle difference allows for the generation of discretizations of very high order.

The overall method described in this report is a hybrid method that uses different methods to determine the velocities at points on the interface and points within the domain, but away from the interface. We use the well described boundary integral method (BIM) to evaluate the velocity at the interface. These velocities are used as data for the method of Regularized Stokeslets (RS) to obtain the velocity at points on a regular background grid. This hybrid approach is an efficient method that capitalizes on the strengths of each of these methods and is a conceptually straightforward method for tracking the dynamics of a two-fluid system. Although the method is applicable in higher dimensions, for simplicity we will restrict our domain to two dimensions throughout.

We organize the manuscript as follows: the first sections introduce the notation and governing equations and briefly describe the methods that are used. The purpose of this investigation is not to develop more powerful implementations of these methods, but to describe how to use each of them to increase the efficiency of the numerical simulations. Therefore we describe the methods briefly and describe the development of the numerical method; we then focus on several simulations aimed at validating and applying the numerical methods and finally summarize the results. The focus of this report is on the development of the method rather than the applications which will be described elsewhere; hence we give two examples where this method is applicable.

2. Derivation of the hybrid approach

2.1. ***Model equations.*** We consider the coupled motion of two fluids of different viscosities. We assume that viscosity dominates both fluids, so the inertial terms may be neglected. The fluids occupy a region Ω and are separated by a surface, Γ. We denote the two sub-regions as $\Omega^{(1)}$ and $\Omega^{(2)}$ for the external and internal fluids, respectively.

The dynamics of both fluids are governed by the incompressible Stokes equations

$$\nabla \cdot \sigma^{(*)} = \mathbf{F} \tag{1}$$

$$\nabla \cdot \mathbf{U}^{(*)} = 0, \tag{2}$$

where $* = 1, 2$ denotes variables in the external and internal regions, respectively, $\mathbf{U}$ is the velocity vector and $\mathbf{F}$ denotes an applied force . Stokes equations describe conservation of momentum and mass with stress tensors $\sigma^* = -P^*\mathbf{I} + \mu^*(\nabla\mathbf{U}^* + \nabla\mathbf{U}^{*\mathbf{T}})$ that contain both the hydrostatic pressures, P^*, and the viscous stresses proportional to the deformation gradient tensor with viscosities μ^* that are generally different.

The solution to Equations (1) and (2) when $\mathbf{F}$ is a force applied at a single point, $\mathbf{F} = \mathbf{f}\delta(\mathbf{x} - \mathbf{x_0})$, is referred to as a *stokeslet* or the Greens' function. The key property of Stokes equations that is exploited by both BIM and RS is the existence of a functional representation of the stokeslet for a variety of domains [13]. The free space Green's function, $\mathbf{G}$, in two dimensions is,

$$\mathbf{G}_{ij}(\mathbf{x}) = -\delta_{ij} \ln r + \frac{(\mathbf{x} - \mathbf{x_0})_i (\mathbf{x} - \mathbf{x_0})_j}{r^2}. \tag{3}$$

The related stress tensor,$\mathbf{T}$, is,

$$\mathbf{T}_{ijk} = -4\frac{(\mathbf{x} - \mathbf{x_0})_i (\mathbf{x} - \mathbf{x_0})_j (\mathbf{x} - \mathbf{x_0})_k}{r^4}, \tag{4}$$

where $r = |\mathbf{x} - \mathbf{x_0}|$.

Instead of using the free-space Green's function we could use the Green's function that enforces the zero-flow boundary condition by subtracting image singularities if the domain is bounded (e.g. channel flow) [13]. We also note that $\mathbf{G}$ and $\mathbf{T}$ are often referred to as the single and double layer potentials, respectively.

2.2. ***Boundary integral method.*** The boundary integral method (BIM) exploits the linearity of the basic flows and translates the differential equations (1) and (2) to integral equations determining the velocities within the domain. These are used along with continuity of the flow to determine an integral equation that determines the velocity of the interface. This method has several advantages including reduction in the dimensionality of the problem, ability to handle generic interfaces and incorporation of different material properties [15; 6].

To derive the BIM equations, we relate the unknown velocity $\mathbf{U}^*$ to the flow induced by a singular force with intensity $\mathbf{f}$ at a point $\mathbf{x_0}$, $\mathbf{U}'$. Thus $\mathbf{U}'$ is a fundamental solution to the incompressible Stokes equations

$$\nabla \cdot \sigma' = \mathbf{f}\delta(\mathbf{x} - \mathbf{x_0}) \tag{5}$$

$$\nabla \cdot \mathbf{U}' = 0, \tag{6}$$

where $\sigma' = -P\mathbf{I} + \mu(\nabla\mathbf{U} + \nabla\mathbf{U}^{\mathbf{T}})$. This is a convenient flow to use since the solution is given by Equation (3).

The reciprocal relation for the bulk flow is determined by relating solutions to Equations (1) and (2), with $\mathbf{F} = 0$, to (5), (6). By direct calculation, we find that

$$\nabla \cdot (\mathbf{U}\sigma') - \nabla \cdot (\mathbf{U}'\sigma) = \delta(\mathbf{x} - \mathbf{x_0})\mathbf{U}, \tag{7}$$

which is the classical reciprocal relation.

Integrating the reciprocal relation, with various placements of the singular force, we recast Equations (1) and (2), with $* = 1$, as an integral equation whose domain is the interface Γ. The integral equation relates the bulk fluid velocity to the traction jump across the interface, denoted $\Delta\sigma = (\sigma^{(1)} - \sigma^{(2)})$, and the velocity; see [13, Chapter 5]. The motion of the bulk fluid is

$$\mathbf{U}_j^{(1)}(\mathbf{x_0}) = -\frac{1}{4\pi\mu^{(1)}} \int_\Gamma \Delta\sigma_{ik}\eta_k(\mathbf{x})\mathbf{G}_{ij}(\mathbf{x}, \mathbf{x_0})\, dl(\mathbf{x}) + \frac{1-\lambda}{4\pi} \int_\Gamma \mathbf{U}_i(\mathbf{x})\mathbf{T}_{ijk}(\mathbf{x}, \mathbf{x_0})\eta_k(\mathbf{x})\, dl(\mathbf{x}), \tag{8}$$

where $\lambda = \dfrac{\mu^{(2)}}{\mu^{(1)}}$ and we denote the outward normal to the interface as η.

Equation (8) governs the j-th component of the external fluid velocity. In a similar manner, we obtain an integral equation for the motion in $\Omega^{(2)}$,

$$\mathbf{U}_j^{(2)}(\mathbf{x_0}) = -\frac{1}{4\pi\mu^{(1)}\lambda}\int_\Gamma \Delta\sigma_{ik}\eta_k(\mathbf{x})\mathbf{G}_{ij}(\mathbf{x},\mathbf{x_0})\,dl(\mathbf{x}) + \frac{1-\lambda}{4\pi\lambda}\int_\Gamma \mathbf{U}_i(\mathbf{x})\mathbf{T}_{ijk}(\mathbf{x},\mathbf{x_0})\eta_k(\mathbf{x})\,dl(\mathbf{x}). \quad (9)$$

These two integral equations govern the coupled motion of the external and internal materials. Because the flows must be continuous at the boundary, we can obtain the boundary velocity by taking the limit of (8) and (9) as $\mathbf{x}_0$ moves to the boundary. These limits both converge to

$$U_j(\mathbf{x}_0) = -\frac{1}{2\pi\mu^{(1)}(\lambda+1)}\int_\Gamma \Delta\sigma_{ik}\eta_k(\mathbf{x})\mathbf{G}_{ij}(\mathbf{x},\mathbf{x_0})\,dl(\mathbf{x}) + \frac{\kappa}{2\pi}\int_\Gamma \mathbf{U}_i(\mathbf{x})\mathbf{T}_{ijk}(\mathbf{x},\mathbf{x_0})\eta_k(\mathbf{x})\,dl(\mathbf{x}), \quad (10)$$

where $\kappa = \frac{1-\lambda}{1+\lambda}$. The latter integral must be handled with care. There are many methods for evaluating this integral that depend on the dimension of Γ as well as the kernel of the integral. In this situation, the singularity is integrable and straightforward quadrature rules work well [14]. The methods used are described in more detail below.

To close the system we impose a constitutive relation relating the jump in traction, $\Delta\sigma$, to the curvature, ω. We assume that the surface traction is proportional to the curvature, with constant of proportionality γ. This constitutive relation is interpreted as a traction arising from surface tension. We note that this is substantially different than what needs to be done in higher dimensions (see [13, Section 5.5] for a description in three-dimensions).

2.3. ***Regularized stokeslets.*** This method, described in [5], is an efficient method to approximate the solution to Stokes equations in the presence of immersed boundaries or obstacles. Conceptually, RS uses the fact that Stokes equations are linear so the velocity at a given point is linearly related to the force applied at that point. Given a collection of points, $\mathbf{x_i}$, at which singular forces are applied , $\mathbf{F} = \mathbf{f}\delta(\mathbf{x}-\mathbf{x_i})$, the total flow is given as the superposition of each of the stokeslets. However, this representation of the velocity is singular at each of the points. In RS, the forces are applied over a small ball, regularizing Stokes equations. The forces are given by $\mathbf{F} = \mathbf{f}\phi_\epsilon(\mathbf{x}-\mathbf{x_0})$, where ϕ_ϵ is a smooth function with support ϵ. For particular choices of ϕ_ϵ, analytic expressions for the solution to Stokes equations with regularized forces can be recovered. These 'regularized stokeslets' are analytic

and converge to the classical stokeslet as ϵ tends toward zero. The flow due to a collection of forces is a linear combination of the regularized stokeslets.

A specific blob in three dimensions is

$$\phi_\epsilon(\mathbf{x}) = \frac{3\epsilon^3}{2\pi(|\mathbf{x}|^2+\epsilon^2)^{5/2}}.$$

The corresponding Green's function is

$$G_\epsilon(r) = \frac{1}{2\pi}\left(\ln(\sqrt{r^2+\epsilon^2}+\epsilon) - \frac{\epsilon}{\sqrt{r^2+\epsilon^2}}\right),$$

where $r = |\mathbf{x}-\mathbf{x_i}|$. The velocity at $\mathbf{x}$ due to a collection of forces centered at $\mathbf{x_i}$ is

$$\mathbf{U}(\mathbf{x}) = -\mathbf{f_i}\frac{1}{4\pi\mu}\sum_{i=1}^{N}\left(\ln(\sqrt{r_i^2+\epsilon^2}+\epsilon) - \frac{\epsilon(\sqrt{r_i^2+\epsilon^2}+2\epsilon)}{(\sqrt{r^2+\epsilon^2}+\epsilon)\sqrt{r_i^2+\epsilon^2}}\right) + \frac{1}{4\pi\mu}\big(\mathbf{f_i}\cdot(\mathbf{x}-\mathbf{x_i})\big)(\mathbf{x}-\mathbf{x_i})\left(\frac{\sqrt{r_i^2+\epsilon^2}+2\epsilon}{(\sqrt{r_i^2+\epsilon^2}+\epsilon)^2\sqrt{r_i^2+\epsilon^2}}\right). \quad (11)$$

This relationship can also be used to determine forces that yield particular flows. For example, if the velocities at a collection of N points , $\mathbf{x}_i$, are known, one has $\mathcal{U} = \mathcal{M}\mathcal{F}$. The vectors $\mathcal{U}$ and $\mathcal{F}$ are $2N \times 1$ and $\mathcal{M}$ is a $2N \times 2N$ matrix. Inverting this relation gives the forces in terms of the (known) velocities. It should be noted that $\mathcal{M}$ is typically not invertible. This can be avoided by adding a constant to the normal component of the forces, affecting the pressure but not the velocity [5] and using an iterative solver. We can then reconstruct the velocity everywhere within the domain, since it is a superposition of the background flow and the flow due to the calculated forces. This method can be made second-order accurate everywhere using the techniques described in [2].

2.4. ***Hybrid method.*** Although much work has been done for determining the velocity and evolution of the interface, in many applications, the velocity is needed throughout the domain. In applications where the concentration of advected substances (such as oxygen) is needed throughout the domain such as biofilms [4], the velocity at all points of a regular grid is required.

The general method begins by initializing the interface between the two fluids, Γ. We parameterize the coordinates of the interface by s, $\Gamma(x, y, t) = (x(s,t), y(s,t))$. The interface is discretized into control points and Equation (10) is solved at each of the discrete points. We then use RS to determine the velocity at regular grid points. The velocities can then used to determine the transport of a chemical throughout the domain.

Because the surface traction is assumed to be proportional to the curvature, the right-hand-side of Equation (10) is

$$-\frac{1}{2\pi\mu^{(1)}(\lambda+1)}\int_\Gamma \Delta\sigma_{ik}\eta_k(\mathbf{x})\mathbf{G}_{ij}(\mathbf{x},\mathbf{x_0})\,dl(\mathbf{x}) = -\frac{\gamma}{2\pi\mu^{(1)}(\lambda+1)}\int_\Gamma \omega(\mathbf{G}_{1j}\eta_1+\mathbf{G}_{2j}\eta_2)\,dl(\mathbf{x}).$$

The curvature of the boundary is

$$\omega = \frac{x_s y_{ss} - y_s x_{ss}}{(x_s^2+y_s^2)^{(3/2)}}. \tag{12}$$

The outward normal is calculated using the parameterization of the interface.

To solve Equation (10), we are then confronted with a system of coupled integral equations which can be written as

$$\mathbf{W} = \mathbf{b} + \frac{\kappa}{2\pi}\int_\Gamma \mathbf{KW}\,dl(\mathbf{x}). \tag{13}$$

where $\mathbf{W} = (\mathbf{U}_1^{(1)}, \mathbf{U}_2^{(1)})$. The vector $\mathbf{b}$ contains the stokeslet and the tensor $\mathbf{K}$ contains the related stress tensor, both of which are known.

A straightforward method for solving the discretized integral equations is Nyström's method [16], which requires a quadrature rule:

$$\int_a^b y(s)ds = \sum_{j=1}^{n}\omega_j y(s_j),$$

where ω_j denotes the weights of the quadrature rule. For our simulations we use Gauss-Legendre quadrature. Although the kernel of Equation (13) has an integrable singularity, it has been shown that the convergence of Nyström's method is the same as the rate of convergence of the quadrature and, if the singularity is of a known type, the rate of convergence can be increased by correcting the quadrature weights [1]. We note that we do not use this acceleration technique in the present investigation.

Once we have solved this system, we have the velocity of the interface at the discrete control points. We could proceed in a similar manner and find the velocity at each point in a regular lattice using Equations (8) and (9). Instead we use the method described above. We first determine the forces that must be applied to the fluid in $\Omega^{(1)}$ so that the velocities at the control points on the interface match those obtained by the BIM. Using these forces and Equation (11), we find the velocities at regular grid points in $\Omega^{(1)}$. The forces required to force the internal fluid to match the velocity of the interface are proportional to the forces obtained for the external fluid and we can then determine the velocity within $\Omega^{(2)}$. The velocities are continuous at the interface and both agree with the velocity found using BIM.

This hybrid method is computationally less expensive than naive implementation of the boundary integral method. The operation count using the BIM equation to determine the velocity at each point on a regular $(M \times M)$ grid using a trapezoidal approximation of the integral, with N discrete points on the interface, is $\mathcal{O}(N^3M^2)$ for each time step. For the RS the operation count is $\mathcal{O}(N^2 + NM^2)$ since each time step requires a matrix inversion (using GMRES $\mathcal{O}(N^2)$ and the summation of the stokeslet solutions for each of the N control points at each of the M^2 points on the regular grid). Thus in problems which require the velocity at all points on a background grid, the interface can be accurately discretized without undo cost.

It should be noted that the above discussion does not consider less naive methods for solving the integral equations that arise from BIM. There are several techniques that are used to reduce the operation count. In particular fast summation multipole methods can be much less computationally expensive [2; 12]. Using these methods the operation count for using BIM alone can be reduced to $\mathcal{O}(NM^2)$ for each time step. Thus for problems for which the fluid flow throughout the domain is needed accurately throughout the domain (i.e., where M is much large than N), BIM with multipole methods can be slightly better than the hybrid method. In our simulations it is typical to have $N = 100$, while $M = 250$ so that the relative difference in the orders is very small, about 0.5%.

3. Simulations

3.1. ***Steady flow.*** We first describe the numerical results of applying the method for a single time step to validate the implementation of Nyström's method as well as demonstrate the behavior of the method for systems with differing viscosities. We begin with a square domain located at $(-2, 2) \times (-2, 2)$. A fluid of viscosity μ_{int} located at the interior of a circle of radius one. The fluid outside of the circle has viscosity μ_{ext}. We use methods described in [10] to derive an analytic solution to the problem with given forces. We parameterize the circle by $\mathbf{x} = (\cos(\theta), \sin(\theta))$ and take the analytic solutions given in [5] and scale the pressure with the internal and external viscosities.

The analytic representation of the pressure and velocities, namely

$$p(r,\theta) = \begin{cases} \mu_{\text{ext}} r^{-3} \sin(3\theta) & \text{for } r \geq 1, \\ \mu_{\text{int}} r^{3} \sin(3\theta) & \text{for } r < 1, \end{cases}$$

$$u_1(r,\theta) = \begin{cases} \frac{1}{8} r^{-2} \sin(2\theta) - \frac{3}{16} r^{-4} \sin(4\theta) + \frac{1}{4} r^{-2} \sin(4\theta) & \text{for } r \geq 1, \\ \frac{3}{8} r^{2} \sin(2\theta) + \frac{1}{16} r^{4} \sin(4\theta) - \frac{1}{4} r^{2} \sin(2\theta) & \text{for } r < 1, \end{cases}$$

$$u_2(r,\theta) = \begin{cases} \frac{1}{8} r^{-2} \cos(2\theta) + \frac{3}{16} r^{-4} \cos(4\theta) - \frac{1}{4} r^{-2} \cos(4\theta) & \text{for } r \geq 1, \\ \frac{3}{8} r^{2} \cos(2\theta) - \frac{1}{16} r^{4} \cos(4\theta) - \frac{1}{4} r^{2} \cos(2\theta) & \text{for } r < 1, \end{cases}$$

are used in the equations

$$[\sigma_{ij}]\eta_j = -f_i, \text{ for } i = 1, 2,$$

to calculate the boundary forces. Here the stress tensor σ is

$$\sigma_{ij} = -p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right).$$

For our calculations we prescribe the boundary forces at discrete points representing the boundary and solve Equation (13) to determine the boundary velocities. The velocities at the control points are used to determine the forces to apply to the fluid regions to match the calculated velocities which in turn is used to determine the external and internal velocities via the RS. Because we are only considering the steady problem for prescribed boundary forces we do not need to use the curvature constitutive assumption to specify the boundary traction. Instead, $\nabla\sigma_{ij}\eta_j = f_i$

To consider the convergence of the numerical approximation to the exact solution, we allow the the number of points discretizing the interface, N, to increase while the background grid used to determine the velocity away from the interface is fixed. Following [5], we measure the error along the line $(x, 3/10)$. We first consider the case where the viscosities are equal as in [5] (see Table 1). The velocity field and the x- and y- components of the velocity along the line $(x, 3/10)$ are shown in in Figure 1. By comparing the ratios of the errors as the number of interface points is doubled we find the order of the comparison. For a first order method the ratio is two while a second order method would have a ratio of four. These ratios are provided in the tables indicating that the method is between first and second order for each of the examples.

Next, we consider the behavior when the viscosities of the two fluids are not equal. In general the convergence rate is reduced as the jump in the viscosity increases . The results for two simulations with different viscosities are summarized in Tables 2 and 3. To determine the rate of convergence we compare the errors for subsequent refinements. We note that this convergence would be improved if the methods in [2] were implemented; however, our results are in agreement with those in [5] for the single viscosity case.

# boundary points	L_2 error in u_1	ratio	L_2 error in u_2	ratio
$N = 50$	2.99×10^{-2}		3.14×10^{-2}	
$N = 100$	7.89×10^{-3}	3.7896	6.46×10^{-3}	4.8606
$N = 200$	1.83×10^{-3}	4.3833	1.03×10^{-3}	6.2718
$N = 400$	4.47×10^{-4}	4.0268	3.53×10^{-3}	2.9178

Table 1. Velocity errors: $\mu_1 = \mu_2 = 1$.

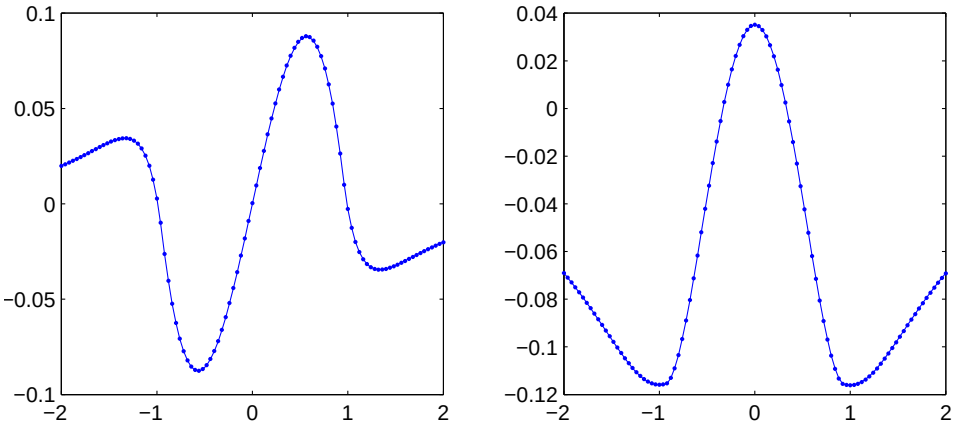


Figure 1. Comparison of the analytic velocities (solid) and the numerical approximation (dotted) along the line $(x, 3/10)$. Left caption shows u_1 while the right shows u_2. The background mesh has 100×100 points and the boundary is discretized with 200 points. The solution is being approximated well.

3.2. ***Viscous suctioning.*** We have just computed the approximate solution for a single time step. To determine the behavior of the numerical methods for a moving boundary, we choose to examine a problem that is similar to the well known Hele-Shaw problem with a singular sink term [3]. Here, we consider the flow of a two-fluid system where an initially circular blob of fluid with viscosity μ_2 is

# boundary points	L_2 error in u_1	ratio	L_2 error in u_2	ratio
$N = 50$	3.33×10^{-2}		2.42×10^{-2}	
$N = 100$	8.86×10^{-3}	3.7584	9.76×10^{-3}	2.4795
$N = 200$	2.54×10^{-3}	3.4881	2.50×10^{-3}	3.9040
$N = 400$	9.67×10^{-4}	2.6266	6.17×10^{-4}	4.0518

Table 2. Velocity errors: $\mu_1 = 1$, $\mu_2 = 2$.

# boundary points	L_2 error in u_1	ratio	L_2 error in u_2	ratio
$N = 50$	2.68×10^{-2}		5.06×10^{-2}	
$N = 100$	9.94×10^{-3}	2.6962	1.96×10^{-2}	2.5816
$N = 200$	4.78×10^{-3}	2.0795	8.90×10^{-3}	2.2022
$N = 400$	1.50×10^{-3}	3.1867	4.35×10^{-3}	2.0460

Table 3. Velocity errors: $\mu_1 = 1$, $\mu_2 = 5$.

immersed in a viscous fluid of viscosity μ_1. The circle is initially of radius one and centered at the origin. At time $t = 0$, we initiate a singular sink term at a point interior to the circle that draws the interior fluid out of the domain causing motion of the interface and the external fluid. This problem has been treated in many investigations (see [7] for analytic treatment and references). In general, in the absence of surface tension there is a singularity in finite time whenever $\mu_2 > \mu_1$. With small surface tension, the problem is regularized and various smooth solutions can be found depending on the location of the sink relative to the circle as well as the strength of the point-sink.

To include a singular sink term in our scheme, we consider the background flow that is the solution to

$$\Delta \mathbf{U} = \nabla p,$$
$$\nabla \cdot \mathbf{U} = q\delta(\mathbf{x} - \mathbf{x_0}),$$

namely

$$\mathbf{U}_{\text{suction}} = q\frac{\mathbf{x} - \mathbf{x_0}}{|\mathbf{x} - \mathbf{x_0}|}.$$

This is added to the flow obtained due to the interface motion to give the motion of the interface.

The computational domain $[-1.5, 1.5] \times [-1.5, 1.5]$ is discretized into a regular background grid with 200×200 grid points. This initial interface is a circle of radius one centered at the origin. There are 150 regularly spaced control points. The viscosities are $\mu_1 = 1$ and $\mu_2 = .1$, for the internal and external viscosities, respectively. The analytic studies rely on complex function theory and mapping techniques to determine the dynamics of the interface. In general, our method is able to capture, qualitatively, the behaviors that are found analytically. We are also able to determine the velocities throughout the domain. In particular, the solutions exhibit singularities at a time, $t_b(\gamma)$, that depends on the value of the surface tension. In our simulations, t_b is an increasing function of γ, indicating the role of surface tension regularization. We have not done a complete investigation of the behavior as a function of surface tension as that is not the focus of this investigation; however, we show the velocity of the fluids near the developing cusp to indicate that our method can capture the fluid flow near the interface as well as the motion of the interface. The results for various simulations are shown in Figures 2 and 3.

3.3. ***Advection.*** One strength of the method described above is efficient approximation of the velocity everywhere in the domain. This is important in transport problems such as biofilm disinfection, where the main goal is to determine the concentration of biocide and nutrient as it moves in the bulk fluid and into the

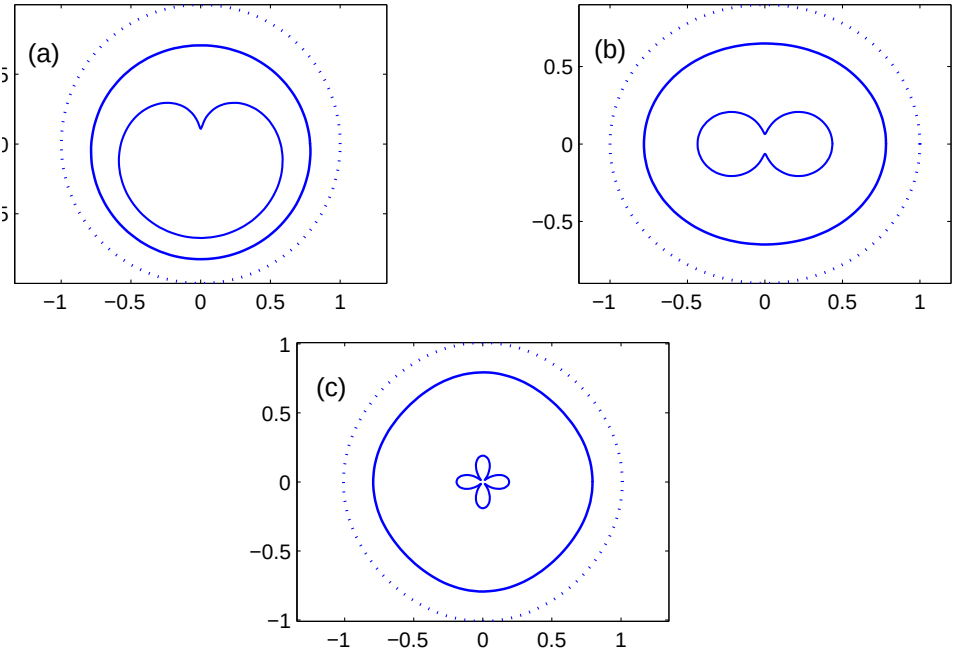


Figure 2. Results for viscous suctioning. (a) Initially circular interface with singularity at $(0, 0.1)$. (b) To break the symmetry, we perturb the circle into an ellipse with major axis 1 and minor axis 0.9 in the y-direction. The singularity is placed at the origin. (c) To break the symmetry, we perturb the circle by adding a periodic fluctuation in the radius with amplitude .01. The singularity is placed at the origin. In all simulations the $\mu_1 = 1$ and $\mu_2 = 0.1$ and the initial interface is indicated with the dotted line while the other curves are shown after 100 and 250 time steps.

biofilm domain. Several models treat the biofilm as a viscous fluid with a viscosity that is substantially different than that of the external fluid [4; 9; 8].

The domain is a channel with a parabolic background flow. Within the channel a generic biofilm interface separates the biofilm fluid, with viscosity $\mu_{biofilm}$ from the bulk fluid with viscosity μ_{bulk}. Although biofilms display viscoelastic properties, the relaxation time has been measured to be on the order of minutes [9]. Because transport of biocide within the biofilm typically takes place on a time scale of hours to days, we treat the biofilm as a viscous fluid immersed in a fluid of much less viscosity [8]. Measurements of biofilm viscosity indicate that the viscosities differ by several orders of magnitude [9]. We set $\mu_{biofilm} = \mu_{bulk} \times 10^6$ and impose a parabolic background flow which is altered by the presence of the biofilm.

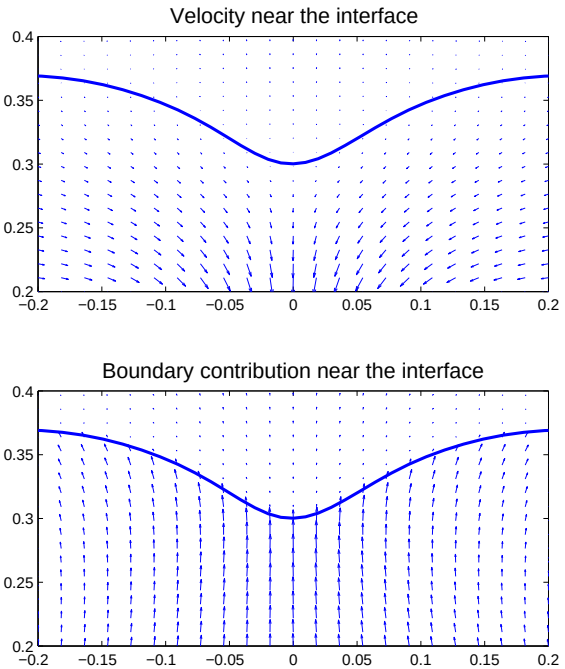


Figure 3. Zoom in of the internal and external velocity field near the forming cusp in Figure 2(a). The top figure shows the total velocity, while the bottom shows the contribution from the interface (without the background flow). Note that the scales have been altered for visualization and do not reflect the magnitudes.

Following the methods described above, we determine the velocity of the interface and fluid in both the bulk and biofilm regions. This is used to track the advection and diffusion of a chemical whose concentration, C, is determined mathematically by a conservation law

$$\frac{\partial C}{\partial t} + (\mathbf{U} \cdot \nabla)C = D\Delta C,$$

where the concentration is fixed at C_0 at the leading edge of the channel. No-flux and outflow conditions are applied at the channel walls and trailing edge of the channel. Given the velocity at time t we determine the concentration at time $t + \Delta t$ using upwinding and ADI to solve the discretized equation.

In Figure 4 we show the developing concentration contours as well as the biofilm/bulk fluid interface for various times. We plan to indicate the effects of including the motion of the biofilm in a future investigation.

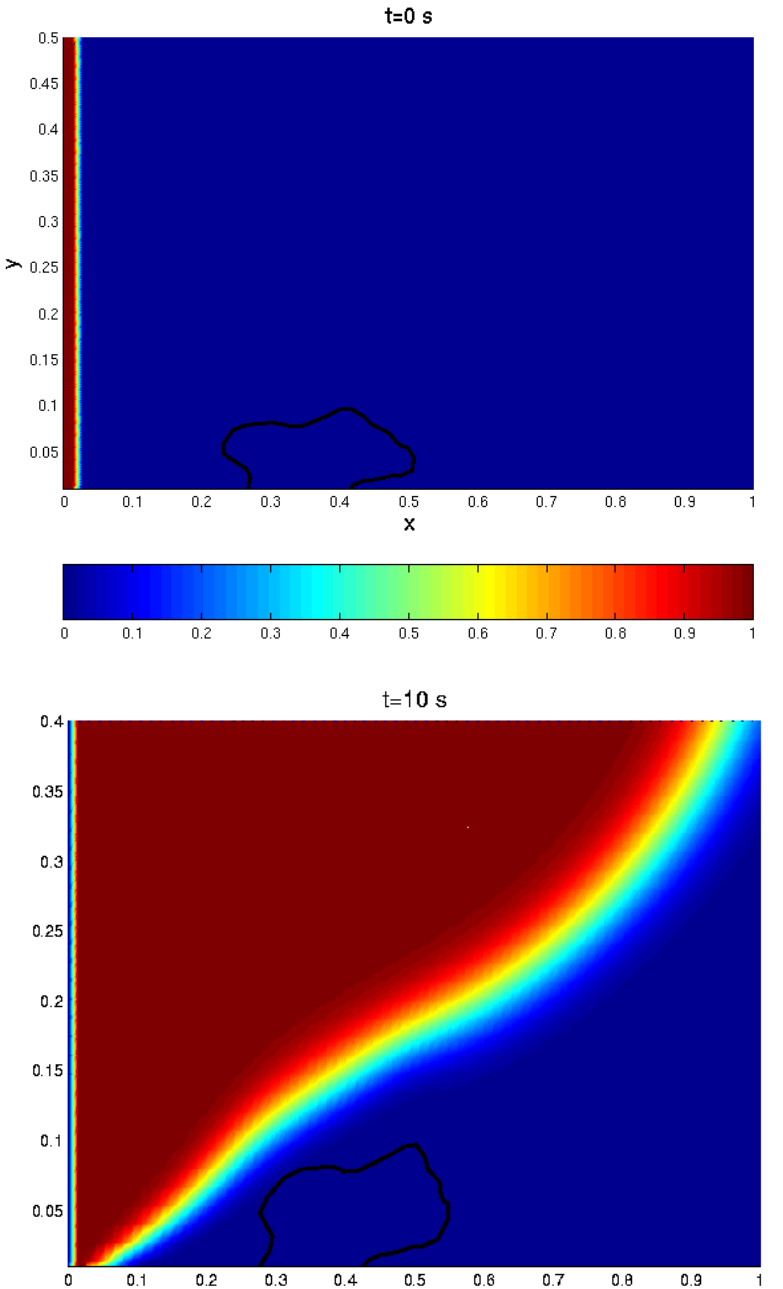
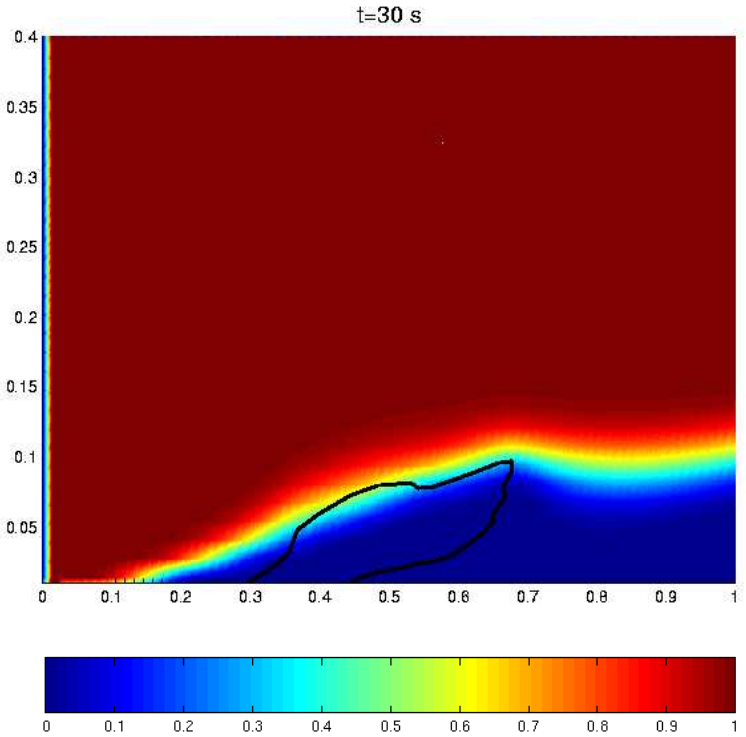
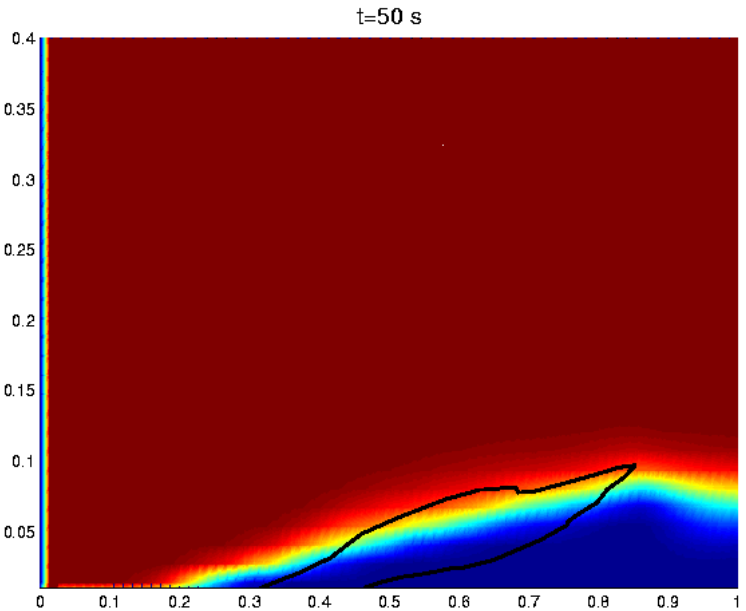


Figure 4. Time-dependent concentration profiles indicating the diffusion and advection of a chemical through the two-fluid domain. We show only part of the domain: $[0, 1] \times [0, 0.4]$. The dynamic fluid/biofilm interface is in black and the concentration for all figures is indicated by the colorbar.


t=30 s
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1


t=50 s
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

4. Discussion

This investigation describes the development of a hybrid method for numerically approximating the motion to two viscous fluids separated by an interface. The interface velocity at control points is determined by solving an integral equation. The velocity at the control points is then used as data to determine the flow outside and inside the interface using the RS. Our method capitalizes on the strengths of both of the methods, since RS is an efficient method but leads to errors at the interface which is precisely where BIM is being applied. We have also indicated that the computational complexity of the hybrid approach is comparable to fast multipole methods applied to BIM, motivating the use of the method for problems where the total flow is needed. We also feel that this investigation indicates an area where the strengths of various methods including RS, multipole methods and matched boundary methods can be exploited.

To apply the numerical method, we studied three different problems. The first was a static problem for which there is an analytic solution. We found that when the viscosities of the fluids are equal, the method behaves as in [5]. However, we have extended the treatment to case with differing viscosities. We then treated a viscous suctioning problem where we were able to capture the qualitative nature of the development of cusps. More importantly for this method, we were able to examine the velocity near the interface. Both the total flow and the flow without the background flow are readily calculated. Finally, we applied our method to determine the concentration of a chemical as it diffuses and advects throughout a channel filled with two immiscible fluids of extremely different viscosities. The last example is the motivation for the numerical method. In particular, the efficiency of the method is desirable since in this situation, the simulations must be carried out for relatively long times (hours, whereas a typical time scale is on the order of seconds to minutes). Other applications where this method could be of use include the diffusion of various cations through the mucociliary layer and into the periciliary layer lining the lungs. This transport problem is a necessary component toward understanding the production and motion of the mucous lining, in particular, for Cystic Fibrosis patients. Reports of the application of this method for more practical problems will be described elsewhere.

References

[1] Bradley K. Alpert, *High-order quadratures for integral operators with singular kernels*, Journal of Computational and Applied Mathematics **60** (1995), 367–378.

[2] J. T. Beale and M.-C. Lai, *A method for computing nearly singular integrals*, SIAM J. Numer. Anal. **38** (2001), 1902–1925.

[3] Hector D. Ceniceros, Thomas Y. Hou, and Helen Si, *Numerical study of Hele-Shaw flow with suction*, Physics of Fluids **11** (1999), no. 9, 2471–2486.

[4] N. G. Cogan and James P. Keener, *The role of the biofilm matrix in structural development*, Mathematical Medicine and Biology **21** (2004), no. 2, 147–166.

[5] Ricardo Cortez, *The method of regularized stokeslets*, SIAM J. Sci. Comput. **23** (2001), no. 4, 1204–1225.

[6] Vittorio Cristini, Jerzy Blawzdziewicz, and Michael Loewenberg, *Drop breakup in three-dimensional viscous flows*, Phys. Fluids Letters **10** (1998), 1781–1783.

[7] L. J. Cummings and S. D. Howison, *Two-dimensional Stokes flow with suction and small surface tension*, European Journal of Applied Mathematics **10** (1999), 681–705.

[8] Isaac Klapper, *Effect of heterogeneous structure in mechanically unstressed biofilms on overall growth*, Bulleting of Mathematical Biology **66** (2004), 809–824.

[9] Isaac Klapper, C.J. Rupp, R. Cargo, B. Purvedorj, and P. Stoodley, *Viscoelastic fluid description of bacterial biofilm material properties*, Biotechnology and Bioengineering **80** (2002), no. 3, 289–296.

[10] Anita T. Layton, *An explicit jump method for the two-fluid stokes equations with an immersed elastic boundary*, Elsevier Science (2006), Submitted.

[11] R. J. Leveque and Z. Li, *Immersed interface methods for Stokes flow with elastic boundaries or surface tension*, SIAM J. Sci. Comput. **18** (1997), 709–735.

[12] A. Mayo, *Fast high order accurate solution of Laplace's equation on irregular regions*, SIAM J. Sci. Comput. **6** (1985), 144–157.

[13] C. Pozrikidis, *Boundary integral and singularity methods for linearized viscous flow*, Cambridge University Press, 1992.

[14] _____, *Interfacial dynamics for stokes flow*, Journal of Computational Physics **169** (2001), 250–301.

[15] _____, *Modeling and simulation of capsules and biological cells*, Chapman & Hall/CRC Press, 2003.

[16] William T. Vetterling, Saul A. Teukolsky, WIlliam H. Press, and Brian Flannerly, *Numerical recipes in c: The art of scientific computing*, Cambridge University Press, Caqmbridge, 2002.

[17] Y.C. Zhoua, Shan Zhaoa, Michael Feig, and G.W. Wei, *High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources*, Journal of Computational Physics **213** (2006), no. 1, 1–30.

Received November 13, 2006.

NICHOLAS G. COGAN: cogan@math.fsu.edu
Department Mathematics, Florida State University, 208 Love Building, Tallahasse, FL 32306-4510
www.math.fsu.edu/∼cogan

Communications in Applied Mathematics and Computational Science

Volume 2 No. 1 2007