Communications in Applied Mathematics and Computational Science

vol. 10 no. 1 2015

msp

Communications in Applied Mathematics and Computational Science

msp.org/camcos

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

The subscription price for 2015 is US $85/year for the electronic version, and $120/year (+$15, if shipping outside the US) for print and electronic. Subscriptions, requests for back issues from the last three years and changes of subscribers address should be sent to MSP.

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

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

PUBLISHED BY

mathematical sciences publishers

nonprofit scientific publishing

http://msp.org/

COMM. APP. MATH. AND COMP. SCI.
Vol. 10, No. 1, 2015
dx.doi.org/10.2140/camcos.2015.10.1

msp

REVISIONIST INTEGRAL DEFERRED CORRECTION WITH ADAPTIVE STEP-SIZE CONTROL

ANDREW J. CHRISTLIEB, COLIN B. MACDONALD,
BENJAMIN W. ONG AND RAYMOND J. SPITERI

Adaptive step-size control is a critical feature for the robust and efficient numerical solution of initial-value problems in ordinary differential equations. In this paper, we show that adaptive step-size control can be incorporated within a family of parallel time integrators known as revisionist integral deferred correction (RIDC) methods. The RIDC framework allows for various strategies to implement step-size control, and we report results from exploring a few of them.

1. Introduction

The purpose of this paper is to show that local error estimation and adaptive step-size control can be incorporated in an effective manner within a family of parallel time integrators based on revisionist integral deferred correction (RIDC). RIDC methods, introduced in [10], are "parallel-across-the-step" integrators that can be efficiently implemented with multicore [10; 6], multi-GPGPU [4], and multinode [9] architectures. The "revisionist" terminology was adopted to highlight that (1) RIDC is a revision of the standard integral defect correction (IDC) formulation [12], and (2) successive corrections, running in parallel but (slightly) lagging in time, revise and improve the approximation to the solution.

RIDC methods have been shown to be effective parallel time-integration methods. They can typically produce a high-order solution in essentially the same amount of wall-clock time as the constituent lower-order methods. In general, for a given amount of wall-clock time, RIDC methods are able to produce a more accurate solution than conventional methods. These results have thus far been demonstrated with constant time steps. It has long been accepted that local error estimation and adaptive step-size control form a critical part of a robust and efficient strategy for solving initial-value problems in ordinary differential equations (ODEs), in particular problems with multiple timescales; see [15], for example. Accordingly, in order to assess the practical viability of RIDC methods, it is important to establish

MSC2010: 65H10, 65L05, 65Y05.
Keywords: initial-value problems, revisionist integral deferred correction, parallel time integrators, local error estimation, adaptive step-size control.

whether they can operate effectively with variable step sizes. It turns out that there are subtleties associated with modifying the RIDC framework to incorporate functionality for local error estimation and adaptive step-size control: there are a number of different implementation options, and some of them are more effective than others.

The remainder of this paper is organized as follows. In Section 2, we review the ideas behind RIDC as well as strategies for local error estimation and step-size control. We then combine these ideas to propose various strategies for RIDC methods with error and step-size control. In Section 3, we describe the implementation of these strategies within the RIDC framework and suggest avenues that can be explored for a production-level code. In Section 4, we demonstrate that the use of local error estimation and adaptive step-size control inside RIDC is computationally advantageous. Finally, in Section 5, we summarize the conclusions reached from this investigation and comment on some potential directions for future research.

2. Review of relevant background

We are interested in numerical solutions to initial-value problems (IVPs) of the form

$$\begin{cases} y'(t) = f(t, y(t)), \quad t \in [a, b], \\ y(a) = y_a. \end{cases} \tag{1}$$

where $y(t) : \mathbb{R} \to \mathbb{R}^m$, $y_a \in \mathbb{R}^m$, and $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$. We first review RIDC methods, a family of parallel time integrators that can be applied to solve (1). Then, we review strategies for local error estimation and adaptive step-size control for IVP solvers.

2.1. ***RIDC.*** RIDC methods [10; 6; 4] are a class of time integrators based on integral deferred correction [12] that can be implemented in parallel via pipelining. RIDC methods first compute an initial (or *provisional*) solution, typically using a standard low-order scheme, followed by one or more corrections. Each correction revises the current solution and increases its formal order of accuracy. After initial startup costs, the predictor and all the correctors can be executed in parallel. It has been shown that parallel RIDC methods with uniform step-sizes give almost perfect parallel speedups [10]. In this section, we review RIDC algorithms, generalizing the overall framework slightly to allow for nonuniform step-sizes on the different correction levels.

We denote the nodes for correction level ℓ by

$$a = t_0^{[\ell]} < t_1^{[\ell]} < \cdots < t_{N^{[\ell]}}^{[\ell]} = b,$$

where $N^{[\ell]}$ denotes the number of time steps on level ℓ. In practice, the nodes on each level are obtained dynamically by the step-size controller.

2.1.1. *The predictor.* To generate a provisional solution, a low-order integrator is applied to solve the IVP (1). For example, a first-order forward Euler integrator applied to (1) gives

$$\eta_n^{[0]} = \eta_{n-1}^{[0]} + \big(t_n^{[0]} - t_{n-1}^{[0]}\big) f(t_{n-1}^{[0]}, \eta_{n-1}^{[0]}), \tag{2}$$

for $n = 1, 2, \ldots, N^{[0]}$, with $\eta_0^{[0]} = y_a$, and where we have indexed the prediction level as level 0. We denote $\eta^{[\ell]}(t)$ as a continuous extension [15] of the numerical solution at level ℓ, i.e., a piecewise polynomial $\eta^{[0]}(t)$ that satisfies

$$\eta^{[0]}(t_n^{[0]}) = \eta_n^{[0]}.$$

The continuous extension of a numerical solution is often of the same order of accuracy as the underlying discrete solution [15]. Indeed, for the purposes of this study, we assume $\eta^{[\ell]}(t)$ is of the same order as $\eta_n^{[\ell]}$.

2.1.2. *The correctors.* Suppose an approximate solution $\eta(t)$ to IVP (1) is computed. Denote the exact solution by $y(t)$. Then, the error of the approximate solution is $e(t) = y(t) - \eta(t)$. If we define the defect as $\delta(t) = f(t, \eta(t)) - \eta'(t)$, then

$$e'(t) = y'(t) - \eta'(t) = f(t, \eta(t) + e(t)) - f(t, \eta(t)) + \delta(t).$$

The error equation can be written in the form

$$\left[e(t) - \int_a^t \delta(\tau)\, d\tau \right]' = f(t, \eta(t) + e(t)) - f(t, \eta(t)), \tag{3}$$

subject to the initial condition $e(a) = 0$. In RIDC, the corrector at level ℓ solves for the error $e^{[\ell-1]}(t)$ of the solution $\eta^{[\ell-1]}(t)$ at the previous level to generate the corrected solution $\eta^{[\ell]}(t)$,

$$\eta^{[\ell]}(t) = \eta^{[\ell-1]}(t) + e^{[\ell-1]}(t).$$

For example, a corrector at level ℓ that corrects $\eta^{[\ell-1]}(t)$ by applying a first-order forward Euler integrator to the error equation (3) takes the form

$$e^{[\ell-1]}(t_n^{[\ell]}) - e^{[\ell-1]}(t_{n-1}^{[\ell]}) - \int_{t_{n-1}^{[\ell]}}^{t_n^{[\ell]}} \delta^{[\ell-1]}(\tau)\, d\tau = \Delta t_n^{[\ell]} \Big[f\big(t_{n-1}^{[\ell]}, \eta^{[\ell-1]}(t_{n-1}^{[\ell]}) + e^{[\ell-1]}(t_{n-1}^{[\ell]})\big) - f\big(t_{n-1}^{[\ell]}, \eta^{[\ell-1]}(t_{n-1}^{[\ell]})\big) \Big],$$

where $\Delta t_n^{[\ell]} = t_n^{[\ell]} - t_{n-1}^{[\ell]}$. After some algebraic manipulation, one obtains

$$\eta_n^{[\ell]} = \eta_{n-1}^{[\ell]} + \Delta t_n^{[\ell]} \Big[f\big(t_{n-1}^{[\ell]}, \eta^{[\ell]}(t_{n-1}^{[\ell]})\big) - f\big(t_{n-1}^{[\ell]}, \eta^{[\ell-1]}(t_{n-1}^{[\ell]})\big) \Big] + \int_{t_{n-1}^{[\ell]}}^{t_n^{[\ell]}} f\big(\tau, \eta^{[\ell-1]}(\tau)\big)\, d\tau. \tag{4}$$

The integral in (4) is approximated using quadrature,

$$\int_{t_{n-1}^{[\ell]}}^{t_n^{[\ell]}} f\big(\tau, \eta^{[\ell-1]}(\tau)\big)\, d\tau \approx \sum_{i=1}^{|\vec{\mathcal{T}}_n^{[\ell]}|} \alpha_{n,i}^{[\ell-1]} f(\tau_i, \eta^{[\ell-1]}(\tau_i)), \quad \tau_i \in \vec{\mathcal{T}}_n^{[\ell]}, \tag{5}$$

where the set of quadrature nodes, $\vec{\mathcal{T}}_n^{[\ell]}$, for a first-order corrector satisfies

1. $|\vec{\mathcal{T}}_n^{[\ell]}| = \ell + 1$,
2. $\vec{\mathcal{T}}_n^{[\ell]} \subseteq \{t_n^{[\ell-1]}\}_{n=0}^{N^{[\ell-1]}}$,
3. $\min(\vec{\mathcal{T}}_n^{[\ell]}) \le t_{n-1}^{[\ell]}$,
4. $\max(\vec{\mathcal{T}}_n^{[\ell]}) \ge t_n^{[\ell]}$.

The quadrature weights, $\alpha_{n,i}^{[\ell-1]}$, are found by integrating the interpolating Lagrange polynomials exactly,

$$\alpha_{n,i}^{[\ell-1]} = \prod_{j=1, j\neq i}^{|\vec{\mathcal{T}}_n^{[\ell]}|} \int_{t_{n-1}^{[\ell]}}^{t_n^{[\ell]}} \frac{(t-\tau_j)}{(\tau_i-\tau_j)}\, dt, \quad \tau_i \in \vec{\mathcal{T}}_n^{[\ell]}. \tag{6}$$

The term $f\big(t_{n-1}^{[\ell]}, \eta^{[\ell-1]}(t_{n-1}^{[\ell]})\big)$ in (4) is approximated using Lagrange interpolation,

$$f\big(t_{n-1}^{[\ell]}, \eta^{[\ell-1]}(t_{n-1}^{[\ell]})\big) \approx \sum_{i=1}^{|\vec{\mathcal{T}}_n^{[\ell]}|} \gamma_{n,i}^{[\ell-1]} f\big(\tau_i, \eta^{[\ell-1]}(\tau_i)\big), \quad \tau_i \in \vec{\mathcal{T}}_n^{[\ell]}, \tag{7}$$

where the same set of nodes, $\vec{\mathcal{T}}_n^{[\ell]}$, for the quadrature is used for the interpolation. The interpolation weights are given by

$$\gamma_{n,i}^{[\ell-1]} = \prod_{j=1, j\neq i}^{|\vec{\mathcal{T}}_n^{[\ell]}|} \frac{(t_{n-1}^{[\ell]}-\tau_j)}{(\tau_i-\tau_j)}, \quad \tau_i \in \vec{\mathcal{T}}_n^{[\ell]}. \tag{8}$$

2.2. ***Adaptive step-size control.*** Adaptive step-size control is typically used to achieve a user-specified error tolerance with minimal computational effort by varying the step-sizes used by an IVP integrator. This is commonly done based on a local error estimate. It is also generally desirable that the step-size vary smoothly over the course of the integration. We review common techniques for estimating the local error, followed by algorithms for optimal step-size selection.

2.2.1. *Error estimators.* Two common approaches for estimating the local truncation error of a single-step IVP solver are through the use of Richardson extrapolation (commonly used within a step-size selection framework known as step doubling) and embedded Runge–Kutta pairs [15]. Step doubling is perhaps the more intuitive technique. The solution after each step is estimated twice: once as a full step and

once as two half steps. The difference between the two numerical estimates gives an estimate of the truncation error. For example, denoting the exact solution to IVP (1) at time $t_n + \Delta t$ as $y(t_n + \Delta t)$, the forward Euler step starting from the exact solution at time t_n and using a step-size of size Δt is

$$\eta_{1,n+1} = y(t_n) + \Delta t\ f(t_n, y_n),$$

and the forward Euler step using two steps of size $\Delta t/2$ is

$$\eta_{2,n+1} = \left(y(t_n) + \frac{\Delta t}{2}\ f(t_n, y_n)\right) + \frac{\Delta t}{2} f\left(t_n + \frac{\Delta t}{2}, y(t_n) + \frac{\Delta t}{2}\ f(t_n, y_n)\right).$$

Because forward Euler is a first-order method (and thus has a local truncation error of $\mathbb{O}(\Delta t^2)$), the two numerical approximations satisfy

$$y(t_n + \Delta t) = \eta_{1,n+1} + (\Delta t)^2 \phi + \mathbb{O}(\Delta t^3) + \cdots,$$
$$y(t_n + \Delta t) = \eta_{2,n+1} + 2\left(\frac{\Delta t}{2}\right)^2 \phi + \mathbb{O}(\Delta t^3) + \cdots,$$

where a Taylor series expansion gives that ϕ is a constant proportional to $y''(t_n)$. The difference between the two numerical approximations gives an estimate for the local truncation error of $\eta_{2,n+1}$,

$$e_{n+1} = \eta_{2,n+1} - \eta_{1,n+1} = \frac{\Delta t^2}{2}\phi + \mathbb{O}(\Delta t^3).$$

An alternative approach to estimating the local truncation error is to use embedded RK pairs [11]. An s-stage Runge–Kutta method is a single-step method that takes the form

$$\eta_{n+1} = \eta_n + \Delta t \sum_{i=1}^{s} b_i k_i,$$

where

$$k_i = f\left(t_i + c_i h, \eta_n + \Delta t \sum_{j=1}^{s} a_{ij} k_j\right), \quad i = 1, 2, \ldots, s.$$

The idea is to find two single-step RK methods, typically one with order p and the other with order $p - 1$, that share most (if not all) of their stages but have different quadrature weights. This is represented compactly in the extended Butcher tableau

$$\begin{array}{c|c} c & A \\ \hline & b \\ & \hat{b} \end{array}$$

Denoting the solution from the order-p method as

$$\eta^*_{n+1} = \eta_n + \Delta t \sum_{i=1}^{s} \hat{b}_i k_i, \tag{9a}$$

and the solution from the order-$(p-1)$ method as

$$\eta_{n+1} = \eta_n + \Delta t \sum_{i=1}^{s} b_i k_i, \tag{9b}$$

the error estimate is

$$e_{n+1} = \eta_{n+1} - \eta^*_{n+1} = \Delta t \sum_{i=1}^{s} (b_i - \hat{b}_i) k_i, \tag{9c}$$

which is $\mathbb{O}(\Delta t^p)$.

A third approach for approximating the local truncation error is possible within the deferred correction framework. We observe that in solving the error equation (3), one is in fact obtaining an approximation to the error. As discussed in Section 3.3, it can be shown that the approximate error after ℓ first-order corrections satisfies $o(\Delta t^{p_0+\ell+1})$. We shall see in Section 3.3 that this error estimate proves to be a poor choice for optimal step-size selection because in our formulation the time step selection for level ℓ does not allow for the refinement of time steps at earlier levels.

2.2.2. *Optimal step-size selection.* Given an error estimate from Section 2.2.1 for a step Δt, one would like to either accept or reject the step based on the error estimate and then estimate an optimal step-size for the next time step or retry the current step. Following [16], Algorithm 1 outlines optimal step-size selection given an estimate of the local truncation error. In lines 1–4, one computes a scaled error estimate. In line 5, an optimal time step is computed by scaling the current time step. In lines 6–10, a new time step is suggested; a more conservative step-size is suggested if the previous step was rejected.

3. RIDC with adaptive step-size control

There are numerous adaptive step-size control strategies that can be implemented within the RIDC framework. We consider three of them in this paper as well as discuss other strategies that are possible.

3.1. ***Adaptive step-size control: prediction level only.*** One simple approach to step-size control with RIDC is to perform adaptive step-size control on the prediction level only, e.g., using step doubling or embedded RK pairs as error estimators for the step-size control strategy. The subsequent correctors then use this grid unchanged (i.e., without performing further step-size control). With this strategy, corrector ℓ is

Input:
y_n: approximate solution at time t_n;
y_{n+1}: approximate solution at time t_{n+1};
e_{n+1}: error estimate for y_{n+1};
p: order of integrator;
m: number of ODEs;
`atol`, `rtol`: user specified tolerances;
`prev_rej`: flag that indicates whether the previous step was rejected;
$\alpha < 1$: safety factor;
$\beta > 1$: allowable change in step-size.

Output:
`accept_flag`: flag to accept or reject this step;
Δt_{new}: optimal time step

Set $a(i) = \max\{|y_n(i)|, |y_{n+1}(i)|\},\ i = 1, 2, \ldots, m.$
Compute $\tau(i) = \texttt{atol} + \texttt{rtol} * a(i),\ i = 1, 2, \ldots, m.$
Compute $\epsilon = \sqrt{\frac{\sum_{i=1}^{m} (e(i)/\tau(i))^2}{m}}.$
Compute $\Delta t_{opt} = \Delta t\ (\frac{1}{\epsilon})^{1/(p+1)}.$
if `prev_rej` **then**
    $\Delta t_{new} = \alpha \min\{\Delta t, \max\{\Delta t_{opt}, \Delta t/\beta\}\}$
else
    $\Delta t_{new} = \alpha \min\{\beta \Delta t, \max\{\Delta t_{opt}, \Delta t/\beta\}\}$
end
if $\epsilon > 1$ **then**
    `accept_flag` $= 1$
else
    `accept_flag` $= 0$
end

Algorithm 1: Optimal step-size selection algorithm. The approximate solution, the error estimate, and its order are provided as inputs. For the numerical experiments in Section 4, we fix $\alpha = 0.9$, $\beta = 10$.

lagged behind corrector $\ell - 1$ so that each node simultaneously computes an update on its level (after an initial startup period). This is illustrated graphically in Figure 1. In principle, near optimal parallel speedup is maintained with this approach provided the computational overhead for the RIDC method (i.e., the interpolation, quadrature, and linear combination of solutions) is small compared to the advance of predictor

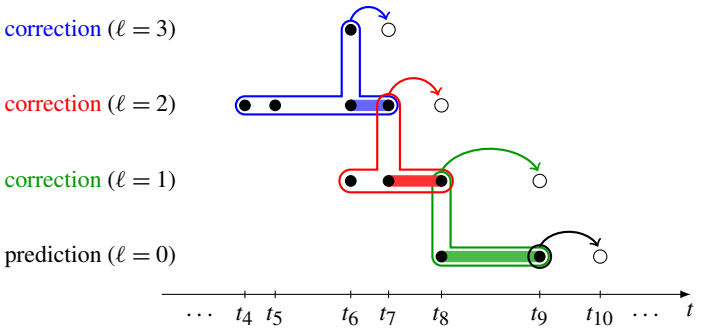


Figure 1. Schematic diagram of step-size control on the prediction level only. The filled circles denote previously computed and stored solution values at particular times. The corrections are run in parallel (but lagging in time) and the open circles indicate which values are being simultaneously computed. The stencil of points required by each level is shown by the "bubbles" surrounding certain grid points; the thick horizontal shading indicates the integrals needed in (4).

from t_n to t_{n+1}; in this implementation, a small memory footprint similar to [10] can be used. Additionally, an interpolation step is circumvented because the nodes are the same on each level. There are however a few potential drawbacks to this approach. First, it is not clear how to distribute the user-defined tolerance among the levels. Clearly, satisfying the user-specified tolerance on the prediction level defeats the purpose of the deferred correction approach. Estimating a reduced tolerance criterion may be possible a priori, but such an estimate would at present be ad hoc. Second, there is no reason to expect the corrector (4) should take the same steps to satisfy an error tolerance when computing a numerical approximation to the error equation (3).

3.2. ***Adaptive step-size control: all levels.*** A generalization of the above formulation is to utilize adaptive step-size control to solve the error equations (3) as well. The variant we consider is step doubling on all levels, where each predictor and corrector performs Algorithm 1; embedded RK pairs can also be used to estimate the error for step-size adaptivity on all levels. Intuitively, step-size control on every level gives more opportunity to detect and adapt to error than simply adapting using the (lowest-order) predictor. For example, this allows the corrector take a smaller step if necessary to satisfy an error tolerance when solving the error equation. Some drawbacks are: (i) an interpolation step is necessary because the nodes are generally no longer in the same locations on each level, (ii) more memory registers are required, and (iii) there is a potential loss of parallel efficiency because a corrector may be stalled waiting for an adequate stencil to become available to compute a quadrature approximation to the integral in (4). Another issue — both a potential benefit and a potential drawback — is the number of parameters that can be tuned

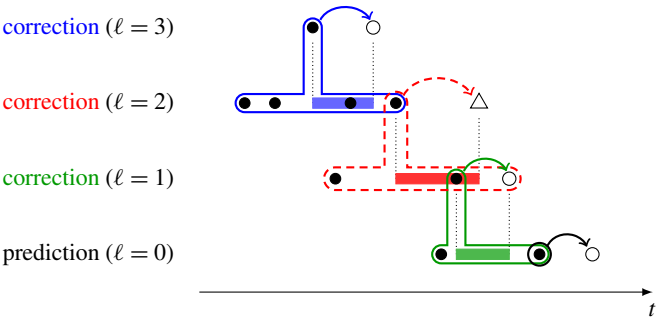


Figure 2. Schematic diagram of a scenario when step-size control is applied on all levels. Unlike in Figure 1, here each level has its own grid in time. Solid circles indicate particular times and levels where the solution is known. In this particular diagram, levels $\ell = 0, 1, 3$ are all able to advance simultaneously to the open circles. However, correction level $\ell = 2$ is unable to advance to the time indicated by the triangle symbol because correction level $\ell = 1$ has not yet computed far enough. The stencil of points required by each level is shown by the "bubbles" surrounding certain grid points; the thick horizontal shading indicates the integrals needed in (4). Note in particular that the dashed stencil includes a open circle at level $\ell = 1$ that is not yet computed.

for each problem. A discussion on the effect of tolerance choices for each level is provided in Section 4. One can in practice also tune step-size control parameters α, β, `atol`, and `rtol` for Algorithm 1 separately on each level. Figure 2 highlights that some nodes might not be able to compute an updated solution on their current level if an adequate stencil is not available to approximate the integral in (4) using quadrature. In this example, the level $\ell = 2$ correction is unable to proceed because it would require interpolated solution values not yet available from level $\ell = 1$, whereas the prediction level $\ell = 0$ and corrections $\ell = 1$ and $\ell = 3$ are all able to advance the solution by one step.

3.3. ***Adaptive step-size control: using the error equation.*** A third strategy one might consider is adaptive step-size control for the error equation (3) using the solution to the error equation itself as the error estimate. (One still uses step doubling or embedded RK pairs to obtain an error estimate for step-size control on the predictor equation (1).) At first glance, this looks promising provided the order of the integrator can be established because it is used to determine an optimal step-size. One would expect computational savings from utilizing available error information, as opposed to estimating it via step doubling or an embedded RK pair.

If first-order predictor and first-order correctors are used to construct the RIDC method, the analysis in [17] can be easily extended to the proposed RIDC methods with adaptive step-size control. We note that the numerical quadrature approximation given in (5) and the numerical interpolation given in (7) are accurate to the order $\mathcal{O}(\Delta t_n^{\ell+2})$; this is sufficient for the inductive proof in [17] to hold. Hence, one

can show that the method has a formal order of accuracy $\mathbb{O}(\Delta t^{\ell+2})$, where $\Delta t = \max_{n,\ell}(t_n^{[\ell]} - t_{n-1}^{[\ell]})$.

Although the formal order of accuracy can be established, using the error estimate from successive levels is a poor choice for optimal step-size selection. Consider step-size selection for level ℓ, time step $t_n^{[\ell]}$, using $\eta_n^{[\ell]} - \eta^{[\ell-1]}(t_n^{[\ell]})$ as the error estimator in Algorithm 1. The optimal step-size is chosen to control the local error estimate via the step-size $\Delta t_n^{[\ell]} = t_n^{[\ell]} - t_{n-1}^{[\ell]}$. However, the local error for the correctors generally contains contributions from the solutions at all the previous levels. The validity of the asymptotic local error expansion of the RIDC method in terms of $\Delta t_n^{[\ell]}$ requires that $\Delta t = \max_{n,\ell}(t_n^{[\ell]} - t_{n-1}^{[\ell]})$ be sufficiently small, and it is not normally possible to guarantee this in the context of an IVP solver. In other words, the step-size controller for a corrector at a given level cannot control the entire local error, and hence standard step-control strategies, which are predicated on the validity of error expansions in terms of only the step-size to be taken, cannot be expected to perform well. We present some numerical tests in Section 4.2.4 to illustrate the difficulties with using successive errors as the basis for step-size control.

3.4. ***Further discussion.*** There are many other strategies/implementation choices that affect the overall performance of the adaptive RIDC algorithm. Some have already been discussed in the previous section. We summarize some of the implementation choices that must be made:

- The choice of how to estimate the error of the discretization must be made. Three possibilities have already been mentioned: step doubling, embedded RK pairs, and solutions to the error equation (3). A combination of all three is also possible.
- If an IVP method with adaptive step-size control is used to solve (3), choices must be made as to how the tolerances and step-size control parameters, α and β, are to be chosen for each correction level.

We also list a few implementation details that should be considered when designing adaptive RIDC schemes.

- If adaptive step-size control is implemented on all levels, some correction levels may sit idle because the information required to perform the quadrature and interpolation in (4) is not available. This idle time adversely affects the parallel efficiency of the algorithm. One possibility to decrease this idle time is instead of taking an "optimal step" (as suggested by the step-size control routine), one could take a smaller step for which the quadrature and interpolation stencil is available. There is some flexibility in choosing exactly which points are used in the quadrature stencil, and it might also be possible to choose a stencil to minimize the time that correction levels are sitting idle.

- Because values are needed from lower-order correction levels, the storage required by a RIDC scheme depends on when values can be overwritten (see, e.g., the stencils in Figures 1 and 2). Thus to avoid increasing the storage requirements, the prediction level and each correction level should not be allowed to get too far ahead of higher correction levels. Although this is also the case for the nonadaptive RIDC schemes [10; 6], if adaptive step-size control is implemented on all levels (Figure 2), the memory footprint is likely to increase. Some consideration should thus be given to a potential trade-off between parallel efficiency and the overall memory footprint of the scheme.
- It is important to reduce round-off error when computing the quadrature weights (6) and the interpolation weights (8). This can be done by through careful scaling and control of the order of the floating-point operations [3].
- If one wishes to use higher-order correctors and predictors to construct RIDC integrators, we note that the convergence analysis in [7; 8; 5] only holds for uniform steps. A nonuniform mesh introduces discrete "roughness" (see [8]); hence, an increase of only one order per correction level is guaranteed even though a high-order method is used to solve (3).
- RIDC methods necessarily incur computational overhead costs, for example, quadrature evaluation (5), interpolation evaluation (7), and the combination of these components in (4). Parallel speedup can only be expected if the computational overhead is small compared to the advance of predictor from t_n to t_{n+1}.

Additionally, the RIDC framework, by construction, solves a series of error equations to generate a successively more accurate solution. This framework can be potentially be exploited to generate *order-adaptive* RIDC methods. For example, one might control the number of corrector levels adaptively based on an error estimate.

4. Numerical examples

We focus on the solutions to three nonlinear IVPs. The first is presented in [1]; we refer to it as the Auzinger IVP:

$$\begin{cases} y_1' = -y_2 + y_1(1 - y_1^2 - y_2^2), \\ y_2' = y_1 + 3y_2(1 - y_1^2 - y_2^2), \\ y(0) = (1, 0)^T, \quad t \in [0, 10], \end{cases} \tag{AUZ}$$

that has the analytic solution $y(t) = (\cos t, \sin t)^T$.

The second is the IVP associated with the Lorenz attractor:

$$\begin{cases} y_1' = \sigma(y_2 - y_1), \\ y_2' = \rho y_1 - y_2 - y_1 y_3, \\ y_3' = y_1 y_2 - \beta y_3, \\ y(0) = (1, 1, 1)^T, \quad t \in [0, 1]. \end{cases} \tag{LORENZ}$$

For the parameter settings $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, this system is highly sensitive to perturbations, and an IVP integrator with adaptive step-size control may be advantageous.

The third is the restricted three-body problem from [15]; we refer to it as the Orbit IVP:

$$\begin{cases} y_1'' = y_1 + 2y_2' - \mu' \dfrac{y_1 + \mu}{D_1} - \mu \dfrac{y_1 - \mu'}{D_2}, \\ y_2'' = y_2 - 2y_1' - \mu' \dfrac{y_2}{D_1} - \mu \dfrac{y_2}{D_2}, \\ D_1 = ((y_1 + \mu)^2 + y_2^2)^{3/2}, \; D_2 = ((y_1 - \mu')^2 + y_2^2)^{3/2}, \\ \mu = 0.012277471, \; \mu' = 1 - \mu. \end{cases} \tag{ORBIT}$$

Choosing the initial conditions

$$y_1(0) = 0.994, \quad y_1'(0) = 0, \quad y_2(0) = 0,$$
$$y_2'(0) = -2.00158510637908252240537862224,$$

gives a periodic solution with period $t_{\text{end}} = 17.065216560159625588917206249$.

We now present numerical evidence to demonstrate that:

1. RIDC integrators with nonuniform step-sizes converge and achieve their designed orders of accuracy.
2. RIDC methods with adaptive step-size constructed using step doubling (on the prediction level only) and embedded RK error estimators (on the prediction level only) converge.
3. RIDC methods with adaptive step-size control based on step doubling to estimate the local error on the prediction and correction levels converge; however, the step-sizes selected are poor (many rejected steps), even for the smooth Auzinger problem.
4. RIDC methods with adaptive step-size control based on step doubling to estimate the local error on the prediction level but using the solution to the error equation for step-size control results is problematic.

The numerical examples chosen are canonical problems designed to illustrate the step-size adaptivity properties of the RIDC methods. Because the computational

overhead is significant compared to the advance of an Euler solution from time t_n to t_{n+1}, a runtime analysis does not reveal parallel speedup for any of these examples. Whereas the number of function evaluations is an effective parameter for comparing algorithms, we need a different metric to compare a parallel algorithm to a sequential algorithm. Where appropriate, we tabulate the number of *sets of concurrent function evaluations* as a proxy for measuring parallel speedup when the function evaluation costs dominate. A set of concurrent function evaluations consists of function evaluations that can be evaluated in parallel.

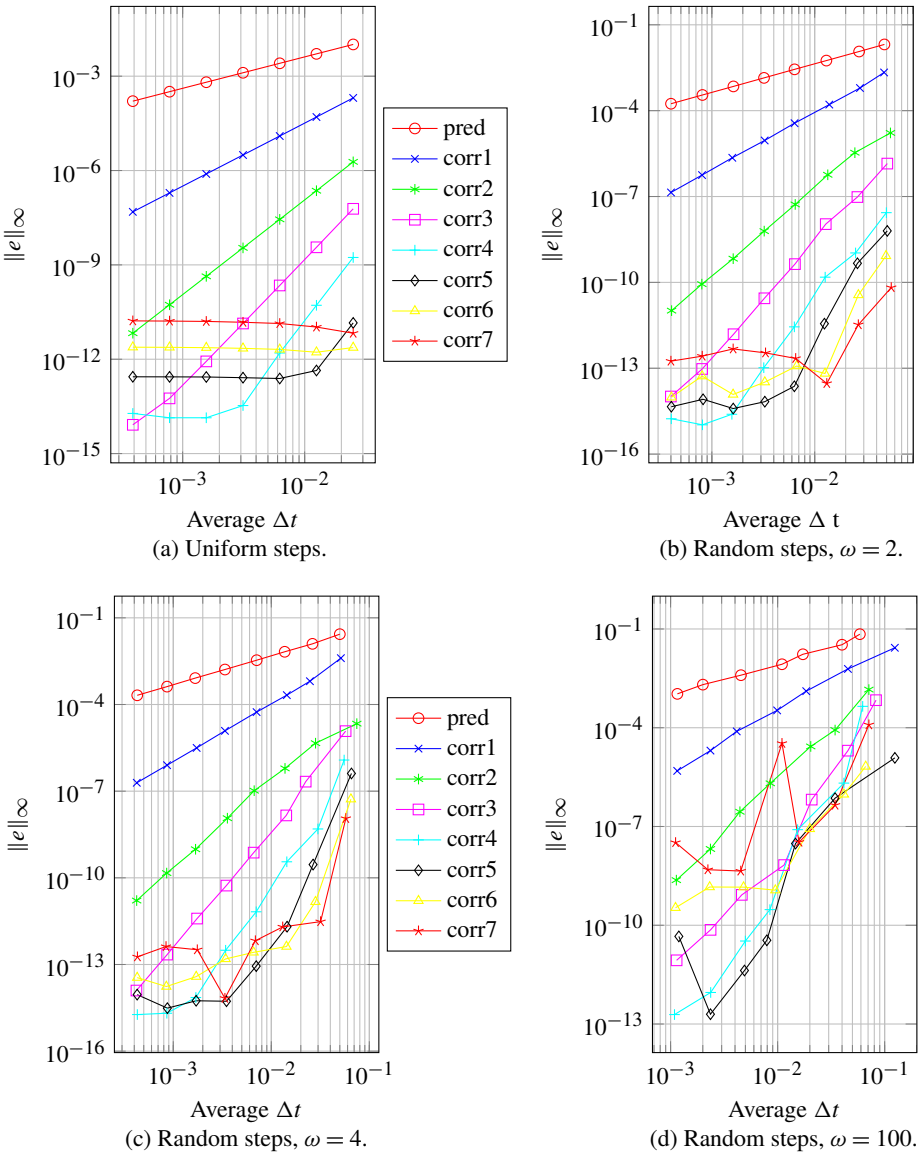
4.1. ***RIDC with nonuniform step-sizes.*** For our first numerical experiment, we demonstrate that RIDC integrators with nonuniform step-sizes converge and achieve their design orders of accuracy. Figure 3 shows the classical convergence study (error as a function of mean step-size) for the RIDC integrator applied to (AUZ). Figure 3(a) shows the convergence of RIDC integrators with uniform step-sizes; Figure 3(b)–(d) shows the convergence of RIDC integrators when random step-sizes are chosen. The random step-sizes are chosen so that

$$\Delta t_n^{[\ell]} \in \left[\frac{1}{\omega}\Delta t_{n-1}^{[\ell]}, \omega \Delta t_{n-1}^{[\ell]}\right], \quad \omega \in \mathbb{R},$$

where ω controls how rapidly a step-size is allowed to change. The figures show that RIDC integrators with nonuniform step-sizes achieve their designed order of accuracy (each additional correction improves the order of accuracy by one), at least up to order 6. In Figure 3 (corresponding to RIDC with uniform step-sizes), we observe that the error stagnates at a value significantly larger than machine precision. This is likely due to numerical issues associated with quadrature on equispaced nodes [14]. We note that $\omega = 1$ gives the uniformly distributed case. We also observe that as the ratio of the largest to the smallest cell increases, the performance of higher-order RIDC methods degrades, likely due to round-off error associated with calculating the quadrature and interpolation weights.

Figure 4 shows the convergence study (error as a function of mean step-size) for (LORENZ). The reference solution is computed using an RK-45 integrator with a fine time step. Similar observations can be made that RIDC methods with nonuniform step-sizes converge with their designed orders of accuracy (at least up to order 6).

4.2. ***Adaptive RIDC.*** We study four different variants of RIDC methods with adaptive step-size control: (i) step doubling is used for adaptive step-size control on the prediction level only (Section 4.2.1); (ii) an embedded RK pair is used for adaptive step-size control on the prediction level only (Section 4.2.2); (iii) step doubling is used for adaptive step-size control on the prediction and correction levels (Section 4.2.3); and (iv) step doubling is used for adaptive step-size control

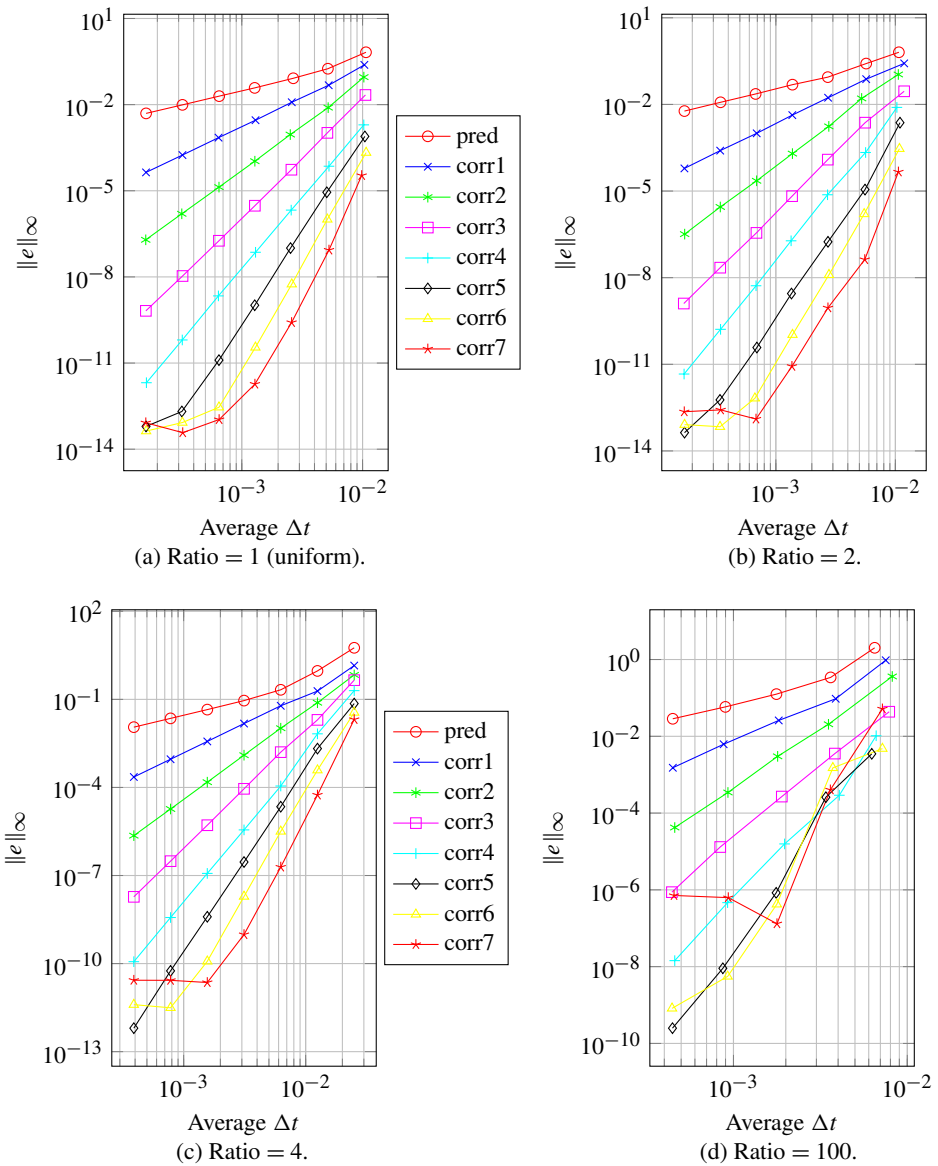


(a) Uniform steps.
(b) Random steps, $\omega = 2$.
(c) Random steps, $\omega = 4$.
(d) Random steps, $\omega = 100$.

Figure 3. Auzinger IVP: The design order is illustrated for the RIDC methods.

on the prediction level, and the computed errors from the error equation (3) are used for adaptive step-size control on the correction levels.

4.2.1. *Step doubling on the prediction level only.* In this numerical experiment, we solve the orbit problem (ORBIT) using a fourth-order RIDC method (constructed using forward Euler integrators), and adaptive step-size control on the prediction level only, where step doubling is used to provide the error estimate. As shown in

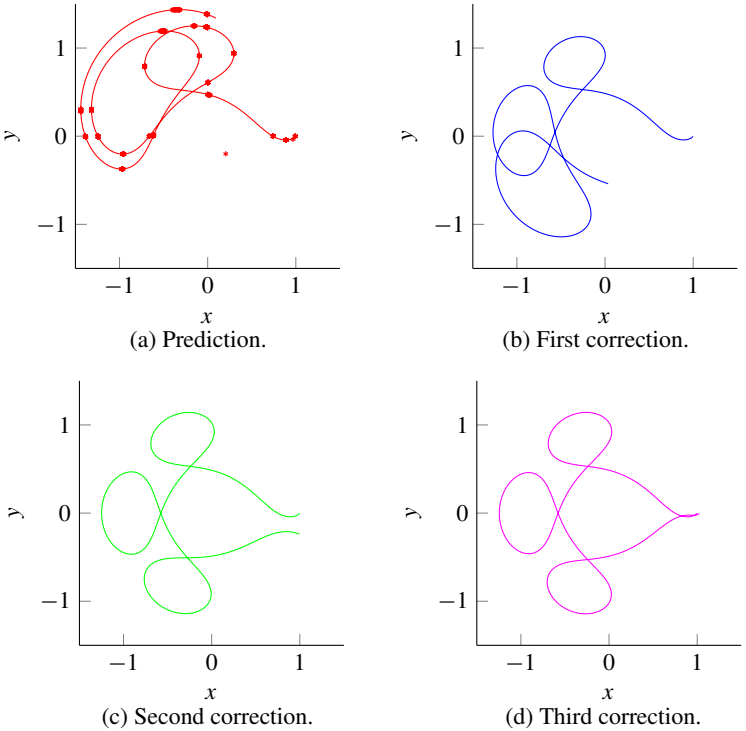


(a) Ratio = 1 (uniform). (b) Ratio = 2.

(c) Ratio = 4. (d) Ratio = 100.

Figure 4. Lorenz IVP: the design order is illustrated for the RIDC methods.

Figure 5, successive correction loops are able to reduce the error in the solution and recover the desired orbit. The red circles in Figure 5(a) indicate rejected steps. Figure 6(a) shows that RIDC with step doubling only on the prediction level converges as the tolerance is reduced. In this experiment, the RIDC integrator is *reset* after every 100 accepted steps. By "reset" [10], we mean that the highest-order solution after every 100 steps is used as an initial condition to reinitialize the



(a) Prediction. (b) First correction.

(c) Second correction. (d) Third correction.

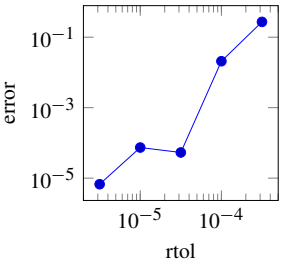
Figure 5. Orbit problem: although the prediction level gives a highly inaccurate solution, successive correction loops are able to reduce the error and produce the desired orbit. The red circles on the prediction level (a) indicate rejected steps.

provisional solution; e.g., instead of solving (1), one solves a sequence of problems

$$\begin{cases} y'(t) = f(t, y), \quad t \in [t_{100(i-1)}, \min(b, t_{100i})], \\ y(t_{100(i-1)}) = \eta_{100(i-1)}^{[P-1]}, \end{cases}$$

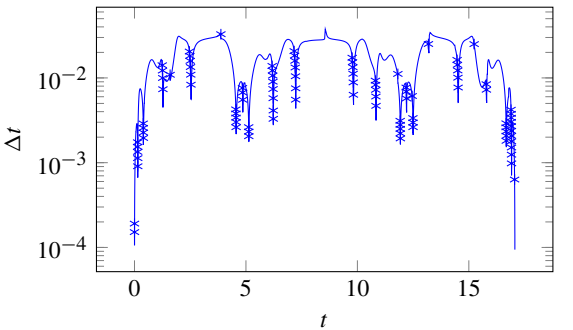
if $(L-1)$ correctors are applied and $\eta_0^{[L-1]} = y_a$. The time steps chosen by the RIDC integrator with resets performed every 100 and 400 steps are shown in Figure 6(b) and (c).

In Figure 6(b), $\Delta t_{\min} = 1.06 \times 10^{-4}$. If a nonadaptive fourth-order RIDC method was used with $\Delta t_{\min}$, 160814 uniform time steps would have been required. By adaptively selecting the time steps for this example and tolerance, the adaptive RIDC method required approximately one one-hundredth of the functional evaluations, corresponding to a one hundred-fold speedup. The effective parallel speedup can be computed by taking the ratio of the total number of function evaluations required and the number of sets of concurrent function evaluations required. For the computation in Figure 6(b) where a reset is performed after every 100 steps, the parallel speedup

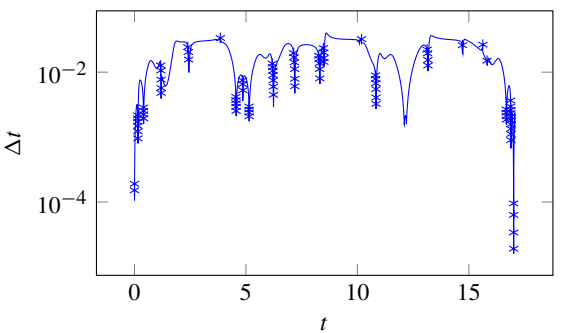


rtol	atol	error	naccept	nreject
$10^{-3.5}$	$10^{-6.5}$	$2.72\cdot10^{-1}$	1456	99
$10^{-4.0}$	$10^{-7.0}$	$2.08\cdot10^{-2}$	2650	81
$10^{-4.5}$	$10^{-7.5}$	$5.35\cdot10^{-5}$	4730	68
$10^{-5.0}$	$10^{-8.0}$	$7.39\cdot10^{-5}$	8436	42
$10^{-5.5}$	$10^{-8.5}$	$6.72\cdot10^{-6}$	15031	10

(a) Convergence study.



(b) Adaptive step-sizes selected (reset every 100 steps).



(c) Adaptive step-sizes selected (reset every 400 steps).

Figure 6. Orbit problem: (a) convergence of a fourth-order RIDC method constructed with forward Euler integrators and adaptive step-size control on the prediction level (using step doubling). Convergence is measured relative to the exact solution as the tolerance is decreased. A reset is performed after every 100 accepted steps for this convergence study. In (b), the step-sizes selected for $\texttt{rtol} = 10^{-3.5}$ and $\texttt{atol} = 10^{-6.5}$ are displayed as the solid curve and rejected steps as $\times$s; a reset is performed after every 100 steps. In (c), the reset is performed after every 400 steps. Observe that although the number of rejected steps increases, the overall Δt chosen remains qualitatively similar.

(if four processors are available) can be computed using

$$\frac{(1456 \times 5) + 99}{(1456 \times 2) + (14 \times 6) + 99} = 2.38.$$

The numerator consists of the total number of function evaluations arising from the number of steps taken and the computation of the error estimate using step doubling and the number of function evaluations arising from the rejected steps. The denominator consists of the number of concurrent function evaluations (including startup costs for the RIDC method). Note that three of the processors sit idle while that step doubling computation is being processed. The parallel speedup can be improved if more levels are chosen, or if the number of resets are reduced. If a reset is performed after every 400 steps (Figure 6(c)), the parallel speedup is

$$\frac{(1591 \times 5) + 88}{(1591 \times 2) + (4 \times 6) + 88} = 2.44.$$

4.2.2. *Embedded RK on the prediction level only.* In this numerical experiment, we repeat the orbit problem (ORBIT) using a fourth-order RIDC method constructed again using forward Euler integrators, but the step-size adaptivity on the prediction level uses a Heun–Euler embedded RK pair. This simple scheme combines Heun's method, which is second order, with the forward Euler method, which is first order. Figure 7(a) shows the convergence of this adaptive RIDC method as the tolerance is reduced. As the previous example, the RIDC integrator is reset after every 100 accepted steps for the convergence study. In Figure 7(b) and (c), we show the time steps chosen by the RIDC integrator with resets performed after 100 or 400 steps, respectively.

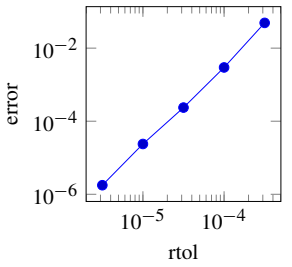
For the computation in Figure 7(b) where a reset is performed after every 100 steps, the parallel speedup (if four processors are available) is

$$\frac{(2441 \times 5) + 60}{(2441 \times 2) + (24 \times 6) + 60} = 2.41.$$

If a reset is performed after every 400 steps (Figure 7(c)), the parallel speedup is

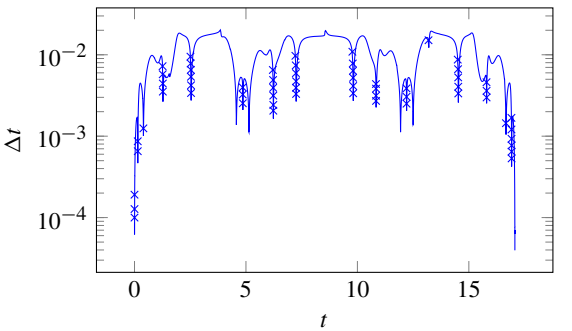
$$\frac{(2276 \times 5) + 80}{(2276 \times 2) + (5 \times 6) + 80} = 2.46.$$

Not surprisingly, the time steps chosen by the RIDC method are dependent on the specified tolerances and the error estimator (and consequently the integrators used to obtain a provisional solution to (1)) used for the control strategy. One can easily construct a RIDC integrator using higher-order embedded RK pairs to solve for a provisional solution to (1), and then use the forward Euler method to solve the error equation (3) on subsequent levels. For example, Figure 8 shows the step-sizes chosen when the Bogacki–Shampine method [2] (a 3(2) embedded RK pair) and the popular Runge–Kutta–Fehlberg 4(5) pair [13] are used to compute the provisional solution (and error estimate) for the RIDC integrator. The same

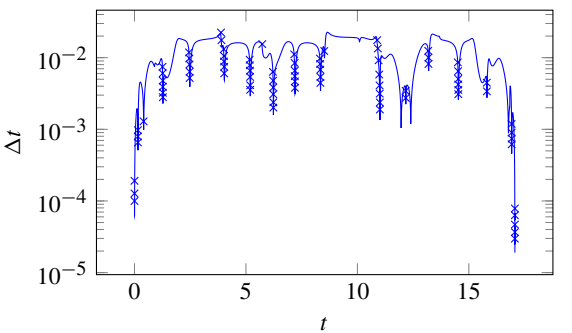


rtol	atol	error	naccept	nreject
$10^{-3.5}$	$10^{-6.5}$	$4.91\cdot10^{-2}$	2082	93
$10^{-4.0}$	$10^{-7.0}$	$2.96\cdot10^{-3}$	3754	71
$10^{-4.5}$	$10^{-7.5}$	$2.36\cdot10^{-4}$	6703	50
$10^{-5.0}$	$10^{-8.0}$	$2.28\cdot10^{-5}$	11945	20
$10^{-5.5}$	$10^{-8.5}$	$1.77\cdot10^{-6}$	21277	10

(a) Convergence study.



(b) Adaptive step-sizes selected (reset every 100 steps).



(c) Adaptive step-sizes selected (reset every 400 steps).

Figure 7. Orbit problem: (a) convergence of a fourth-order RIDC method constructed with forward Euler integrators and adaptive step-size control on the prediction level (using an embedded RK pair to estimate the error). Convergence is measured relative to the exact solution as the tolerance is decreased. A reset is performed after every 100 accepted steps for this convergence study. In (b), the step-sizes selected for $\texttt{rtol} = 10^{-3.5}$ and $\texttt{atol} = 10^{-6.5}$ are displayed as the solid curve and rejected steps as $\times$s; a reset is performed after every 100 steps. In (c), the reset is performed after every 400 steps.

tolerance of $\texttt{rtol} = 10^{-3.5}$ is used to generate both graphs. As the order and accuracy of the predictor increases, one can take larger time steps. For this example, using higher-order embedded RK pairs as step-size control mechanisms for RIDC methods result in less variations in time steps.

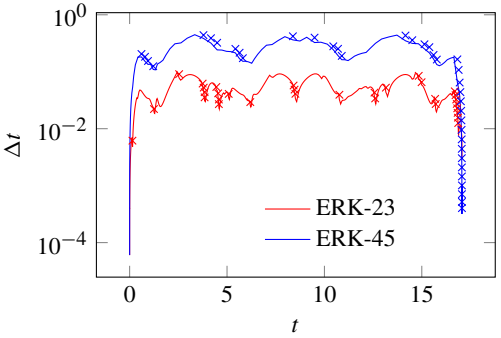
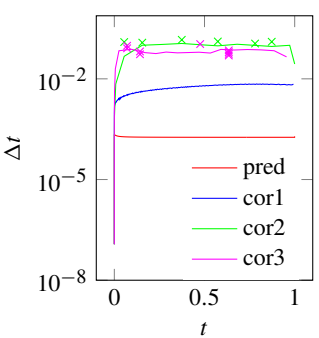


Figure 8. Step-sizes selected by RIDC methods constructed using a Bogacki–Shampine method, a 3(2) embedded pair (red) and the Runge–Kutta–Fehlberg 4(5) pair. Rejected steps are indicated with ×s.

4.2.3. *Step doubling on all levels.* As mentioned in Section 3.2, it might be advantageous to use adaptive step-size control when solving the error equations. This affords a myriad of parameters that can be used to tune the step-size control mechanism. In this set of numerical experiments, we explore how the choice of tolerances for the prediction/correction levels affect the step-size selection.

We first solve the Auzinger IVP using step doubling on all the levels, i.e., both predictor and corrector levels. In Figure 9, we show the computed step-sizes when we naively choose the same tolerances on each level. As expected, the predictor has to take many steps (to satisfy the stringent user-supplied tolerance), whereas life is easy for the correctors. The effective parallel speedup is

$$\frac{(5479 + 196 + 19 + 24) \times 2 + 15}{(5481 \times 2) + 15} = 1.04.$$



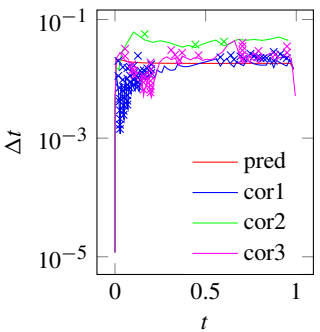
ℓ	rtol	atol	error	naccept	nreject
0	10^{-8}	10^{-10}	$2.028 \cdot 10^{-5}$	5479	0
1	10^{-8}	10^{-10}	$8.793 \cdot 10^{-7}$	196	0
2	10^{-8}	10^{-10}	$2.618 \cdot 10^{-8}$	19	6
3	10^{-8}	10^{-10}	$1.486 \cdot 10^{-6}$	24	9

Figure 9. Auzinger IVP: step-size control is implemented on all prediction and correction levels. The same tolerances are used for each level. As expected, the predictor has a hard time (forward Euler must satisfy a stringent tolerance); on the other hand, life is easy for the correctors. Rejected steps are indicated with ×s. For this set of tolerances, 5481 sets of concurrent function evaluations are needed.

In principle, the correctors are not even needed. Equally important to note is that the error *increases* after the last correction loop. This might seem surprising at first glance but ultimately may not unreasonable because the steps selected to solve the third correction are not based on the solution to the error equation but rather the original IVP.

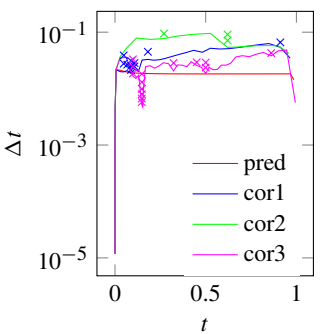
Instead of naively choosing the same tolerances on each level, we now change the tolerance at each level, as described in Figure 10. By making this simple change, the number of accepted steps on each level are now on the same order of magnitude. Not surprisingly, the predictor still selects good steps. Interestingly in Figure 10(a), the first correction is "noisy", especially initially. For this set of tolerances, the effective parallel speedup is

$$\frac{(58+7+30+61)\times 2+(52+7+24)}{(135\times 2)+(52+7+24)} = 1.52.$$



ℓ	rtol	atol	error	naccept	nreject
0	$1{\cdot}10^{-4}$	$1{\cdot}10^{-6}$	$2.026{\cdot}10^{-3}$	58	0
1	$1{\cdot}10^{-6}$	$1{\cdot}10^{-8}$	$6.945{\cdot}10^{-5}$	78	52
2	$1{\cdot}10^{-8}$	$1{\cdot}10^{-10}$	$1.265{\cdot}10^{-7}$	30	7
3	$1{\cdot}10^{-10}$	$1{\cdot}10^{-12}$	$9.579{\cdot}10^{-8}$	61	24

(a) Set 1 of tolerances.



ℓ	rtol	atol	error	naccept	nreject
0	$1{\cdot}10^{-4}$	$1{\cdot}10^{-6}$	$2.026{\cdot}10^{-3}$	58	0
1	$1{\cdot}10^{-5}$	$1{\cdot}10^{-7}$	$1.805{\cdot}10^{-4}$	29	12
2	$1{\cdot}10^{-7}$	$1{\cdot}10^{-9}$	$1.172{\cdot}10^{-6}$	20	6
3	$1{\cdot}10^{-9}$	$1{\cdot}10^{-11}$	$7.216{\cdot}10^{-7}$	39	11

(b) Set 2 of tolerances.

Figure 10. Auzinger IVP: different tolerances at each level. With the first set of tolerances, the step-size controller for the predictor is well behaved, as it is for the second and third correctors. The step-size controller for the first corrector however is noisy. 135 sets of concurrent function evaluations are needed to generate (b). With the second set of tolerances, the step-size controller for all correctors is reasonably well behaved. Here, 64 sets of concurrent function evaluations are needed.

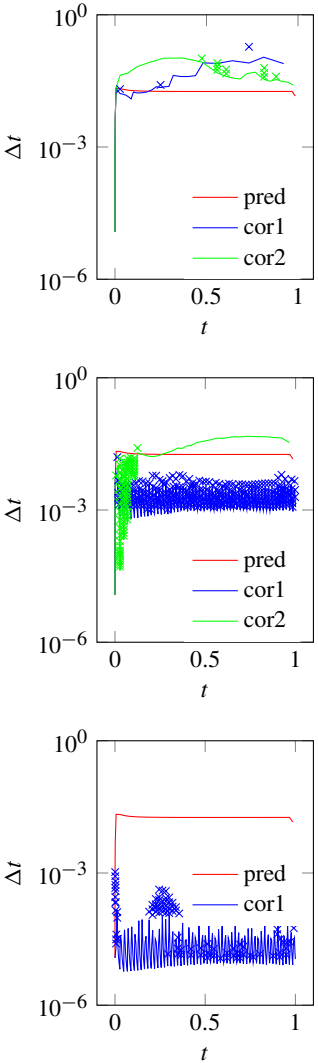
By picking a different set of tolerances, we can eliminate the noise, as shown in Figure 10(b). For this set of tolerances, the parallel speedup is

$$\frac{(58+24+20+39)\times 2+(12+6+11)}{(64\times 2)+(12+6+11)} = 1.98.$$

4.2.4. *Using solutions from the error equation.* As mentioned in Section 3.3, using the solution from the error equation (3) as the local error estimate for step-size control on a given level is potentially problematic because the step-size controller can only control the local error introduced on that level whereas the true local error generally contains contributions from all previous levels. For completeness, we present the results of this adaptive RIDC formulation applied to the Orbit problem (Figure 12) and the Auzinger problem (Figure 11). Step doubling is used for step-size adaptivity on the predictor level, solutions from the error equation are used to control step-sizes for the corrector levels. For the Auzinger problem, we observe in the top figure that if the tolerances are held fixed on each level, each correction level improves the solution. If the tolerance is reduced slightly on each level, the step-size controller gives a poor step-size selection (many rejected steps), even for this smoothly varying problem. For the Orbit IVP, Figure 12 shows that the corrector improves the solution if the tolerances are held fixed at all levels; however the corrector requires *many* steps. A second correction loop was not attempted. Reducing the tolerance for the first corrector resulted in inordinately many rejected steps.

5. Conclusions

In this paper, we formulated RIDC methods that incorporate local error estimation and adaptive step-size control. Several formulations were discussed in detail: (i) step doubling on the prediction level, (ii) embedded RK pairs on the prediction level, (iii) step doubling on the prediction and error levels, and (iv) step doubling for the prediction level but using the solution from the error equation for step-size control; other formulations are also alluded to. A convergence theorem from [17] can be extended to RIDC methods that use adaptive step-size control on the prediction level. Numerical experiments demonstrate that RIDC methods with nonuniform steps converge as designed and illustrate the type of behavior that might be observed when adaptive step-size control is used on the prediction and correction levels. Based on our numerical study, we conclude that adaptive step-size control on the prediction level is viable for RIDC methods. In a practical application where a user gives a specified tolerance, this prescribed tolerance must be transformed to a specific tolerance that is fed to the predictor.



ℓ	rtol	atol	error	naccept	nreject
0	$1{\cdot}10^{-4}$	$1{\cdot}10^{-6}$	$2.031{\cdot}10^{-3}$	59	0
1	$1{\cdot}10^{-4}$	$1{\cdot}10^{-6}$	$7.249{\cdot}10^{-4}$	33	3
2	$1{\cdot}10^{-4}$	$1{\cdot}10^{-6}$	$6.513{\cdot}10^{-6}$	26	10

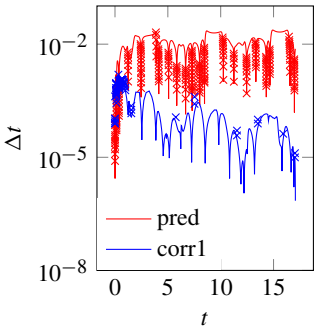
ℓ	rtol	atol	error	naccept	nreject
0	$1{\cdot}10^{-4}$	$1{\cdot}10^{-6}$	$2.031{\cdot}10^{-3}$	59	0
1	$1{\cdot}10^{-5}$	$1{\cdot}10^{-7}$	$1.063{\cdot}10^{-5}$	657	305
2	$1{\cdot}10^{-6}$	$1{\cdot}10^{-8}$	$9.446{\cdot}10^{-8}$	75	76

ℓ	rtol	atol	error	naccept	nreject
0	$1{\cdot}10^{-4}$	$1{\cdot}10^{-6}$	$2.031{\cdot}10^{-3}$	59	0
1	$1{\cdot}10^{-7}$	$1{\cdot}10^{-9}$	$1.178{\cdot}10^{-7}$	60571	94

Figure 11. Auzinger problem: step doubling on prediction level, using successive levels for error estimation for step control on the error equation. Step-size controller for the corrector is noisy.

Acknowledgments

This publication was based on work supported in part by award no. KUK-C1-013-04, made by King Abdullah University of Science and Technology (KAUST), AFRL and AFOSR under contract and grants FA9550-12-1-0455, NSF grant number DMS-0934568, NSERC grant number RGPIN-228090-2013, and the Oxford Center for Collaborative and Applied Mathematics (OCCAM).



ℓ	rtol	atol	error	naccept	nreject
0	$1{\cdot}10^{-4}$	$1{\cdot}10^{-4}$	2.405	2261	230
1	$1{\cdot}10^{-4}$	$1{\cdot}10^{-4}$	$7.234{\cdot}10^{-1}$	475181	84

Figure 12. Orbit problem: step doubling on prediction level, using successive levels for error estimation for step control on the error equation.

References

[1] W. Auzinger, H. Hofstätter, W. Kreuzer, and E. Weinmüller, *Modified defect correction algorithms for ODEs, I: general theory*, Numer. Algorithms **36** (2004), no. 2, 135–155. MR 2005h:65096

[2] P. Bogacki and L. F. Shampine, *A 3(2) pair of Runge–Kutta formulas*, Appl. Math. Lett. **2** (1989), no. 4, 321–325. MR 1025845 Zbl 0705.65055

[3] B. Bradie, *A friendly introduction to numerical analysis: with C and MATLAB materials on website*, Pearson Education, Upper Saddle River, NJ, 2006.

[4] A. Christlieb, A. Melfi, and B. Ong, *Distributed parallel semi-implicit time integrators*, preprint, 2012. arXiv 1209.4297v1

[5] A. Christlieb, M. Morton, B. Ong, and J.-M. Qiu, *Semi-implicit integral deferred correction constructed with additive Runge–Kutta methods*, Commun. Math. Sci. **9** (2011), no. 3, 879–902. MR 2865808 Zbl 1271.65109

[6] A. Christlieb and B. Ong, *Implicit parallel time integrators*, J. Sci. Comput. **49** (2011), no. 2, 167–179. MR 2012k:65067 Zbl 1243.65076

[7] A. Christlieb, B. Ong, and J.-M. Qiu, *Comments on high-order integrators embedded within integral deferred correction methods*, Commun. Appl. Math. Comput. Sci. **4** (2009), 27–56. MR 2010e:65094 Zbl 1167.65389

[8] ———, *Integral deferred correction methods constructed with high order Runge–Kutta integrators*, Math. Comp. **79** (2010), no. 270, 761–783. MR 2011c:65122 Zbl 1209.65073

[9] A. J. Christlieb, R. D. Haynes, and B. W. Ong, *A parallel space-time algorithm*, SIAM J. Sci. Comput. **34** (2012), no. 5, C233–C248. MR 3023735 Zbl 1259.65143

[10] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput. **32** (2010), no. 2, 818–835. MR 2011g:65105 Zbl 1211.65089

[11] J. R. Dormand and P. J. Prince, *A family of embedded Runge–Kutta formulae*, J. Comput. Appl. Math. **6** (1980), no. 1, 19–26. MR 81g:65098 Zbl 0448.65045

[12] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT **40** (2000), no. 2, 241–266. MR 2001e:65104 Zbl 0959.65084

[13] E. Fehlberg, *Low-order classical Runge–Kutta formulas with step size control and their application to some heat transfer problems*, technical report, R-315, NASA, 1969.

[14] S. Güttel and G. Klein, *Efficient high-order rational integration and deferred correction with equispaced data*, *Electron. Trans. Numer. Anal.* **41** (2014), 443–464.

[15] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations, I: Nonstiff problems*, 2nd ed., Springer Series in Computational Mathematics, no. 8, Springer, Berlin, 1993. MR 94c:65005

[16] E. Hairer and G. Wanner, *Solving ordinary differential equations, II: Stiff and differential-algebraic problems*, 2nd ed., Springer Series in Computational Mathematics, no. 14, Springer, Berlin, 1996. MR 97m:65007 Zbl 0859.65067

[17] Y. Xia, Y. Xu, and C.-W. Shu, *Efficient time discretization for local discontinuous Galerkin methods*, Discrete Contin. Dyn. Syst. Ser. B **8** (2007), no. 3, 677–693. MR 2008e:65307 Zbl 1141.65076

Received October 9, 2013. Revised December 10, 2014.

ANDREW J. CHRISTLIEB: *Department of Mathematics, Michigan State University, East Lansing, 48823, United States*

COLIN B. MACDONALD: macdonald@maths.ox.ac.uk
Mathematical Institute, Oxford University, Oxford, OX2 6GG, United Kingdom

BENJAMIN W. ONG: ongbw@mtu.edu
Department of Mathematics, Michigan Technological University, Houghton, MI 49931, United States

RAYMOND J. SPITERI: spiteri@cs.usask.ca
Department of Computer Science, University of Saskatchewan, Saskatoon S7N 5C9, Canada

COMM. APP. MATH. AND COMP. SCI.
Vol. 10, No. 1, 2015
dx.doi.org/10.2140/camcos.2015.10.27

AN ADAPTIVELY WEIGHTED GALERKIN FINITE ELEMENT METHOD FOR BOUNDARY VALUE PROBLEMS

YIFEI SUN AND CHAD R. WESTPHAL

We introduce an adaptively weighted Galerkin approach for elliptic problems where diffusion is dominated by strong convection or reaction terms. In such problems, standard Galerkin approximations can have unacceptable oscillatory behavior near boundaries unless the computational mesh is sufficiently fine. Here we show how adaptively weighting the equations within the variational problem can increase accuracy and stability of solutions on under-resolved meshes. Rather than relying on specialized finite elements or meshes, the idea here sets a flexible and robust framework where the metric of the variational formulation is adapted by an approximate solution. We give a general overview of the formulation and an algorithmic structure for choosing weight functions. Numerical examples are presented to illustrate the method.

1. Introduction

In this paper, we consider numerically approximating solutions to the diffusion, convection, reaction problem

$$\begin{cases} -\varepsilon \Delta u + \boldsymbol{b} \cdot \nabla u + cu = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{1-1}$$

Here, u is the solution, Δu and ∇u are the Laplacian and gradient of u, $\partial\Omega$ is the boundary of domain Ω, and f is a known data function. We assume that coefficients c and ε are positive constants, that $\boldsymbol{b}$ is a constant vector, and that boundary conditions are homogenous Dirichlet although nonzero boundary data or Neumann/mixed boundary conditions may easily be considered under appropriate smoothness assumptions. When $|\boldsymbol{b}| \gg \varepsilon$, we may consider this as a convection-dominated diffusion problem, which may have solutions with boundary layers downstream from $\boldsymbol{b}$. When $c \gg \varepsilon$ and the reaction term dominates, layer phenomena are also possible. It is well known that standard finite element and finite difference

Westphal's work was supported by the National Science Foundation under grant DMS-1216297. The authors wish to thank the anonymous reviewers for their helpful suggestions.
MSC2010: 65N30, 65N12, 35J20.
Keywords: finite element methods, convection-dominated diffusion, boundary layers, adaptive, weighted.

approaches to such problems can yield solutions with undesirable overshoots and/or oscillatory behavior near these boundary layers when the computational mesh is not sufficiently resolved.

The difficulty associated with boundary layers is, of course, not limited to (1-1) but is evident in many applications where diffusive terms are dominated by convective or reactive terms. Throughout this paper, we assume sufficient regularity of the data and domain to ensure solutions are sufficiently smooth, which is a separate issue from boundary layer behavior.

There are many well studied numerical approaches to ameliorate layer effects. Through the use of specialized graded meshes [24; 25; 11; 8] or adaptive mesh refinement [9], it is possible to develop a mesh that has sufficient resolution near the layers to resolve the solution and eliminate the effects of the high gradients on the solution in areas where the solution is smooth, which are commonly referred to as the "pollution effects". It is also possible to augment the weak form of the problem by adding mesh-dependent stabilization terms to the formulation [16; 17; 2]. These terms may or may not be consistent with the original problem, but they generally improve the solution on coarse meshes, and their influence diminishes as the mesh is resolved. The variational problem can also be modified through a Petrov–Galerkin formulation, where the test and trial spaces are different. This includes streamline upwind Petrov–Galerkin (SUPG) formulations [5; 6; 18; 1] as well as methods with spaces enhanced by bubble functions [4]. Such problems have also been studied in the context of discontinuous Galerkin (DG) [11; 15] and discontinuous Petrov–Galerkin (DPG) [10; 13] methods. Here continuity requirements in the trial and test spaces are relaxed, and additional degrees of freedom on the element boundaries lead to additional jump conditions in the variational problem. Further comparisons on earlier work for such problems can be found in [22; 12; 14]. Broadly speaking, there are many ingredients in designing a finite element formulation (i.e., reformulating the equations, choosing/adapting the mesh, choosing test/trial spaces, etc.), and improvements on the standard Galerkin approach have been realized by many modifications and combinations of choices in the basic ingredients.

In this work, we introduce an adaptively weighted Galerkin finite element approach to (1-1) for cases exhibiting boundary layers. By generalizing the standard Galerkin weak form with weighted inner products, we may essentially redistribute the strength by which the variational problem is enforced across the domain. The use of weighted norms and weighted inner products is, of course, not a new idea. In [21], a weighted Galerkin formulation is used for a parabolic problem where the diffusion coefficient changes sign within the interior of the domain. A weighted Galerkin approach is coupled with a mapping technique in [23] to solve elliptic problems on unbounded domains. And in the least-squares finite element paradigm, using weighted norms to generalize L^2 residual minimization problems allows

for robust treatment of problems with boundary singularities in weighted $H^1(\Omega)$ or $H(\mathrm{div})$ norms [19; 20; 7].

For problem (1-1), when the computational mesh is relatively coarse, weakening the problem near boundary layers in the right way can reduce or eliminate the pollution effects, stabilizing the numerical approximation. When the mesh is sufficiently fine to resolve the solution with no ill effects, the approach defaults to the standard Galerkin approach. Here, we explore this idea via an adaptive approach, whereby an approximate solution is used to generate a weight function to define a subsequent problem. While there are many successful methods in the literature, we are particularly motivated by practicality. In many cases, solutions tend to be smooth except for small regions representing a layer, and adopting an exotic approximation space to represent the global solution seems excessive. Generally, if it were computable, the interpolant of even simple finite element spaces would provide a sufficient approximation. Our approach is designed to generalize the standard Galerkin approach, where the mesh and trial space can be chosen based on resolving the features of the solution, and an initial approximation helps redefine the metric of an improved variational formulation. For simplicity, we describe our approach separately from adaptive mesh refinement though mesh adaptation can be used alongside our weighting approach. The weighted Galerkin approach here can also be viewed as a Petrov–Galerkin formulation, where the basis functions of the test space are generated adaptively, based on an approximate solution.

The organization of this paper is as follows. In the following section, we introduce the idea of an adaptively weighted variational problem and describe the construction of an appropriate weight function. Numerical results are given in Section 3, and a brief look at the how the coercivity of the weighted bilinear form is enhanced from the standard approach is given in Section 4.

2. Weighted Galerkin formulation

Throughout this paper, we use standard notation for the $L^2(\Omega)^d$ norm, $\|\cdot\|$, and inner product, $(\cdot,\cdot)$, and use $\|\cdot\|_{H^k}$ to denote the norm corresponding to the Sobolev space $H^k(\Omega)^d$. The space of continuous functions on Ω is denoted by $C^0(\Omega)^d$, and we recall that for $\phi \in H^1(\Omega)^d$ and $\psi \in C^0(\Omega)^d$ we have $\phi\psi \in H^1(\Omega)^d$ for $d = 1, 2, 3$. When the dimension of the problem is understood in context, we drop the d superscript. Since the relative balance between diffusion, convection, and reaction terms in (1-1) determines the behavior of the solution, for the remainder of this paper, we take $\varepsilon = 1$ without loss of generality.

Defining the space $V = \{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\}$ and the bilinear form

$$a(u, v) := (\nabla u, \nabla v) + (\boldsymbol{b} \cdot \nabla u, v) + (cu, v), \tag{2-1}$$

the standard variational formulation of (1-1) for a given $f \in L^2(\Omega)$ is to find $u \in V$ such that $a(u, v) = (f, v)$ for all $v \in V$. The discrete Galerkin formulation is analogous: define a finite element space, $V^h \subset V$, and find $u^h \in V^h$ such that

$$a(u^h, v^h) = (f, v^h) \quad \text{for all } v^h \in V^h. \tag{2-2}$$

Since the main idea in this paper is in modifying the variational framework to achieve better global approximations on relatively coarse resolutions, the choice of the finite element space can be made according to its approximation properties in the interior of the domain or according to simplicity or availability of code. In Section 3, we give numerical results using simple conforming piecewise polynomial spaces. In many cases, solutions tend to be relatively smooth up to boundary layers, and finding accurate approximations up to boundary layers is desirable.

Now let $w \in C^0(\Omega)$ be a weight function such that $0 < w_{\min} \leq w(\boldsymbol{x}) \leq 1$. We define a weighted bilinear form by multiplying each side of the PDE in (1-1) by wv and integrating by parts:

$$\begin{aligned}(f, wv) &= (-\nabla \cdot \nabla u, wv) + (\boldsymbol{b} \cdot \nabla u, wv) + (cu, wv)\\ &= (\nabla u, \nabla(wv)) + (\boldsymbol{b} \cdot \nabla u, wv) + (cu, wv)\\ &= (\nabla u, w\nabla v + v\nabla w) + (\nabla u, wv\boldsymbol{b}) + (cu, wv)\\ &= (\nabla u, w\nabla v) + (\nabla u, (\nabla w + w\boldsymbol{b})v) + (cu, wv) =: W(u, v).\end{aligned}$$

The discrete weighted variational formulation of (1-1) is thus to find $u^h \in V^h$ such that

$$W(u^h, v^h) = (wf, v^h) \quad \text{for all } v^h \in V^h. \tag{2-3}$$

At this point, the weight function need only be sufficiently smooth and positive. Within these requirements, the construction of the weight function is motivated by producing a more robust numerical approximation. Notice that

$$W(v, v) = \|w^{1/2}\nabla v\|^2 + (\nabla v, (\nabla w + w\boldsymbol{b})v) + c\|w^{1/2}v\|^2, \tag{2-4}$$

which indicates in general that choosing $(\nabla w + w\boldsymbol{b})$ small will make the cross term small and $W(v, v)$ will more resemble a weighted $H^1(\Omega)$ measure. In Section 4, we explore the connection between w and the coercivity of $W(\cdot, \cdot)$. In a practical sense, we may view the role of w in (2-3) as being able to dampen the effect of large values of $|\nabla u^h|$. Formally, we know that asymptotic solutions of (1-1) near boundaries may have terms of the form

$$e^{-kx} \quad \text{and} \quad e^{\pm\sqrt{c}x},$$

where $k = |\boldsymbol{b}|$, and thus, boundary layer solutions may change on the order of $\max\{e^{kh}, e^{\sqrt{c}h}\}$ across individual elements of mesh size h. This dramatic growth

of the solution is often problematic, and we are motivated to choose w small in these regions to decouple the effects from the rest of the problem. In other words, in regions where $|\nabla u^h|$ is relatively large, w should be chosen to make $(\nabla w + w\boldsymbol{b})$ small. When the mesh size is small enough to reasonably represent the solution, the standard Galerkin approach yields acceptable results. We now describe an adaptive approach to generate an appropriate weight function, based on the coefficients from the original PDE, an initial approximation to the solution, and the mesh size.

If u^h_{old} represents an initial approximation to the solution to (1-1), we simply want to choose w large/small where the magnitude of ∇u^h_{old} is small/large. While there are many empirical approaches to constructing w from this guiding principle, we describe one here that is simple to implement and tends to give robust results. Let Ω^h be a triangulation of the domain with elements denoted by τ_i, for i ranging from 1 to the number of elements, and choose the approximation space V^h. Our approach is given by the following algorithm:

(1) **Start**: Initially set $w = 1$ uniformly.

(2) **Solve**: Obtain an initial solution u^h_{old} by solving (2-2).

(3) **Construct weight**:

- For each element, τ_i, compute $d_i := \|\nabla u^h_{\text{old}}\|_{\tau_i}$.
- Denote the minimum/maximum values of this set by $d_{\min} = \min_{\tau_i \in \Omega^h} d_i$ and $d_{\max} = \max_{\tau_i \in \Omega^h} d_i$.
- Set $\widehat{w}_i = 1 - (1 - w_{\min})(d_i - d_{\min})/(d_{\max} - d_{\min})$.
- Construct $w(\boldsymbol{x}) \in C^0(\Omega)$ as a piecewise linear function from elementwise values $\widehat{w}_i$. See Section 3 for details.

(4) **Re-solve**: Using w, find u^h by solving problem (2-3).

Algorithm 1. Adaptively weighted Galerkin approximation.

This basic approach can be modified to accommodate other features. As described, this requires two PDE solves on the same mesh. However, it is straightforward to refine the mesh, either locally or globally, between steps (2) and (4) and use the weight from the coarse solve to enhance the fine solve. Similarly, nonlinear or time-dependent problems that rely on an iterative approach based on an approximate solution can incorporate steps (3) and (4) to the iterative method.

In the next section, we show numerical test problems and give details on constructing $w \in C^0(\Omega)$ from the elementwise values $\widehat{w}_i$.

3. Numerical results

In this section, we consider two test problems in 1D where the features of the weighted Galerkin approach can clearly be seen. We then provide an example in 2D that shows how the approach can be incorporated into a more realistic setting.

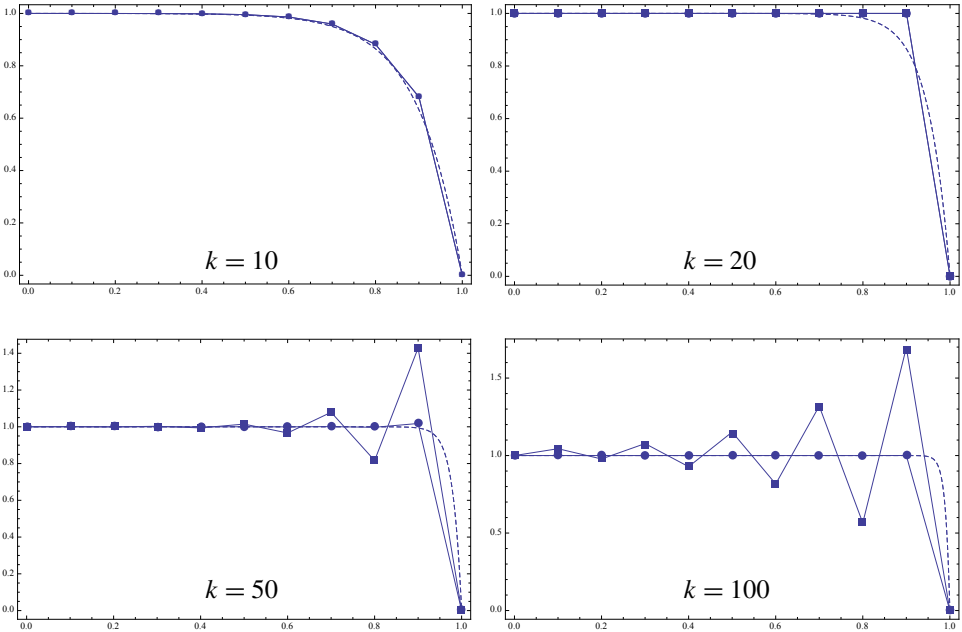


Figure 1. Standard Galerkin (squares) and weighted Galerkin (circles) approximations to (3-1) compared with exact solutions (dashed), for $n = 10$ and $k = 10, 20, 50, 100$.

3.1. ***1D convection-dominated.*** For this first example, we consider the interplay between diffusion and convection in the ODE model problem

$$\begin{cases} -u'' + ku' = 0 \quad \text{in } (0,1), \\ \qquad u(0) = 1, \\ \qquad u(1) = 0, \end{cases} \tag{3-1}$$

which has exact solution $u(x) = (e^{kx} - e^k)/(1 - e^k)$.

We discretize $\Omega = (0, 1)$ into n evenly spaced subintervals of mesh size $h = 1/n$, with nodes $0 = x_0, x_1, \ldots, x_n = 1$, and define V^h as the set of continuous piecewise linear functions satisfying the boundary conditions in (3-1). We follow the basic approach in Algorithm 1, using $w_{\min} = e^{-hk}$, and we construct $w(x)$ as a piecewise linear function on the existing mesh by setting nodal values according to

$$w(x_i) = \begin{cases} \widehat{w}_1 & \text{for } i = 0, \\ \min\{\widehat{w}_i, \widehat{w}_{i+1}\} & \text{for } i = 1, 2, \ldots, n-1, \\ \widehat{w}_n & \text{for } i = n. \end{cases}$$

For $k \le 2n$, the standard Galerkin approach does not give overshoots or oscillatory behavior, but as k increases beyond this, such undesirable behavior occurs. This range also corresponds to the spectrum of the system matrix, which is all real

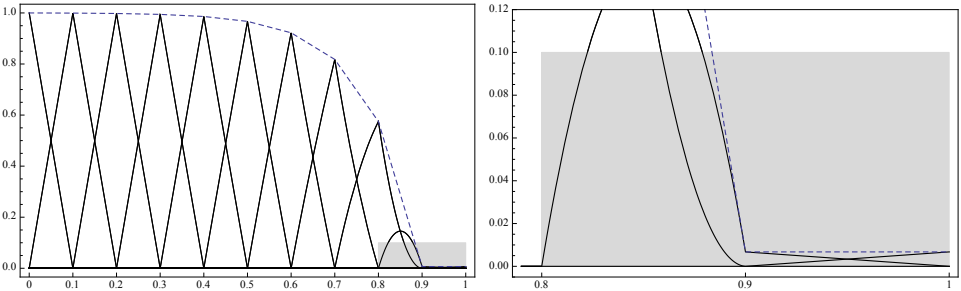


Figure 2. Basis functions $\psi_i = w\phi_i$: on the entire domain (left) and detail near the boundary layer (right).

for $k \leq 2n$ and complex for $k > 2n$. We are thus free to fix the value of n and vary k to explore the possible numerical behavior of the boundary layer near $x = 1$. In Figure 1, results are shown for $n = 10$ and $k = 10, 20, 50, 100$. Improved results can clearly be seen in all cases, where for small k the weighted approach is essentially the same as the standard approach, and as k increases, the weighted approach appropriately isolates the behavior of the solution near the boundary layer.

One way to view the role of the weight functions is as a Petrov–Galerkin formulation, where the basis functions on the test space are created by weighting the basis of the trial space. For example, here $\{\phi_i\}_{i=0}^n$ represents the standard piecewise linear basis and $u^h \in V^h = \text{span}\{\phi_i\}$. Given a weight function, w, the weighted Galerkin method is to find $u^h \in V^h$ such that

$$a(u^h, z^h) = (f, z^h) \quad \text{for all } z^h \in W^h,$$

where $v^h \in W^h = \text{span}\{\psi_i\}$, where $\psi_i = w\phi_i$ for each $i = 0, 1, \ldots, n$. Figure 2 shows this basis for the 1D convection-dominated problem with $n = 10$ and $k = 50$. Near the boundary layer, there is a clear upwinding effect, and in places where the solution is smooth, the basis functions for W^h and V^h are essentially the same. Our adaptive approach tends to resemble the SUPG approach in areas where $|\nabla u^h|$ is large and the standard Galerkin approach otherwise.

3.2. ***1D reaction-dominated.*** Consider the reaction-dominated diffusion ODE

$$\begin{cases} -u'' + cu = c & \text{in } (0, 1), \\ u(0) = 0, \\ u(1) = 0, \end{cases} \tag{3-2}$$

which yields a solution, $u(x) = 1 - (e^{\sqrt{c}(1-x)} + e^{\sqrt{c}x})/(1 + e^{\sqrt{c}})$, that develops boundary layers at $x = 0$ and $x = 1$ for $c \gg 1$. It's easy to see that as $c \to \infty$ the solution approaches $u = 1$ for $x \in (0, 1)$, yet the boundary conditions require $u(0) = u(1) = 0$. We use the weighted Galerkin approach as described above but

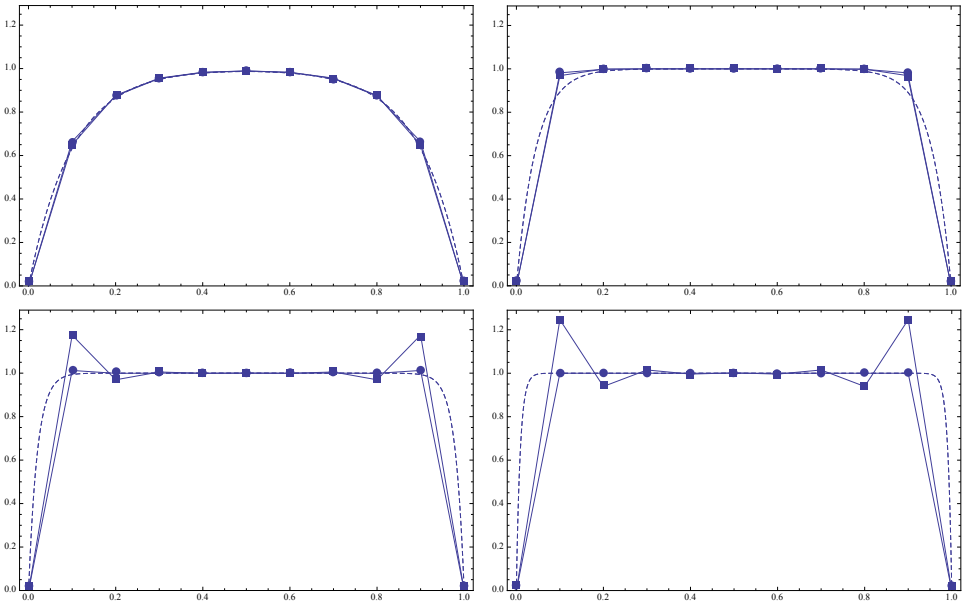


Figure 3. Standard Galerkin (squares) and weighted Galerkin (circles) approximations to (3-2) for $c = 100, 500, 2500, 12500$ with exact solutions (dashed).

with $w_{\min} = e^{-\sqrt{c}/n}$, a choice motivated by the change of the asymptotic solution in the elements nearest each boundary.

Figure 3 shows results qualitatively similar to those in Figure 1. As c increases, boundary layers form near $x = 0$ and $x = 1$. For n small relative to $\sqrt{c}$, both the standard and weighted Galerkin approaches give accurate approximations, but for large values of $\sqrt{c}$, the weighted approach yields better solutions.

3.3. ***2D convection-dominated.*** As an example in 2D, we consider

$$-\Delta u + \boldsymbol{b} \cdot \nabla u = 0 \quad \text{in } \Omega \quad \text{with } \boldsymbol{b} = \frac{200}{\sqrt{x^2 + y^2}} \begin{bmatrix} -y \\ x \end{bmatrix}.$$

We use $\Omega = (0, 1)^2$ with zero Dirichlet boundary conditions on the north, east, and west boundaries and $u(x, 0) = 16x^2(1 - x)^2$ on the south boundary. Here, $\boldsymbol{b}$ represents a convection term of magnitude $k = |\boldsymbol{b}| = 200$, which is in a counterclockwise circular rotation. The solution forms a boundary layer on the west boundary (see Figure 4).

We discretize Ω using a uniform mesh of triangles of size h and use standard P_1 elements for the approximation u^h. As in the previous examples, we follow the structure of Algorithm 1. To construct $w(\boldsymbol{x})$ as a P_1 finite element function on the existing mesh, we choose $w_{\min} = e^{-kh}$ and the nodal values of w as the minimum of d_i on all adjacent triangles. We then find u^h as the solution of (2-3). Figure 4

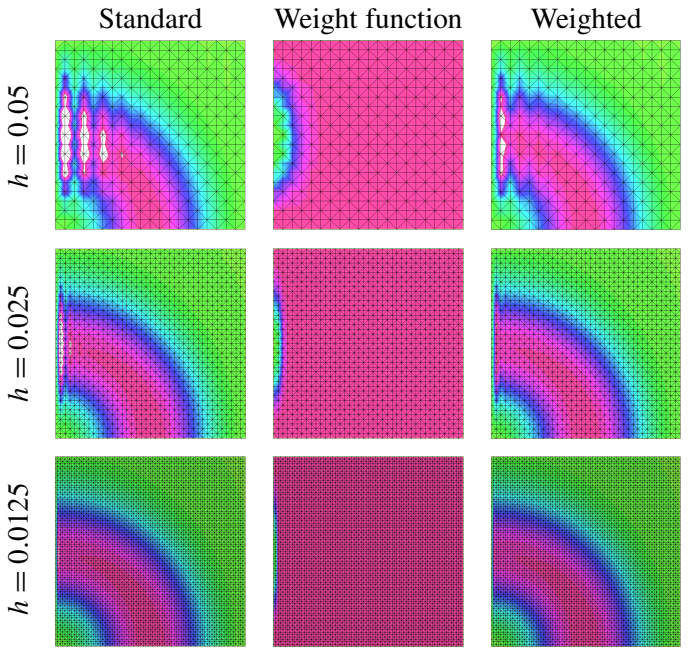


Figure 4. Numerical approximations and adaptively generated weight functions for 2D convection-dominated diffusion problem. Values in [0, 1] are shown in color (green corresponds to 0 and pink corresponds to 1) while values greater than 1 are in white. Overshoot values are given in Table 1.

shows plots of the approximations, comparing the standard Galerkin solution and the weighted Galerkin solution for three mesh resolutions. Note that the boundary conditions dictate that $u \leq 1$, and we take values exceeding $u = 1$ to be considered an overshoot. Weighted Galerkin solutions show significantly less oscillatory behavior than the standard approach. Table 1 shows overshoot values for both approaches.

In Figure 5, we compare the weighted and standard Galerkin approaches by plotting the L^2 error, $\|u^h - u^*\|$, and the H^1 seminorm error, $\|\nabla(u^h - u^*)\|$, for increasing mesh resolution ($n = 10, 20, 40, 80$). We use the numerical solution on a very fine mesh, u^* ($n = 600$), as a proxy for the exact solution. Both methods approach the asymptotic optimal rates of $O(h^2)$ and $O(h)$, but the weighted approach gives better approximations on under-resolved meshes.

h	Standard scheme	Weighted scheme
0.05	1.70	1.15
0.025	1.44	1.04
0.0125	1.10	1.00

Table 1. Overshoot values for numerical approximations, $\max_{\boldsymbol{x} \in \Omega} |u^h|$.

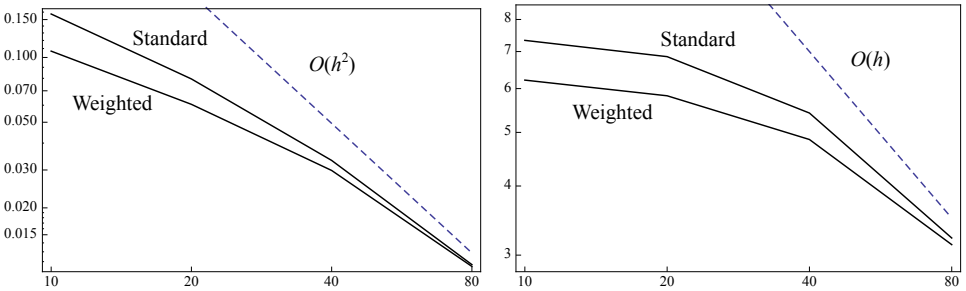


Figure 5. L^2 norm and H^1 seminorm errors for 2D convection-dominated diffusion problem.

4. On coercivity

We briefly explore the connection between the behavior of convection-dominated problems and the coercivity of the variational problem. Understanding when coercivity is lost and the effects on the linear systems leads to a better understanding of how to construct an improved variational problem using weighted inner products.

Definition 4.1. A bilinear form $a(\cdot\,,\cdot)$ on a normed linear space H is said to be **coercive** on $V \subseteq H$ if there exists $\alpha > 0$ such that

$$a(v, v) \geq \alpha \|v\|_H^2 \quad \text{for all } v \in V.$$

Coercivity is of great interest since, if $(H, (\cdot\,,\cdot))$ is a Hilbert space, V is a subspace of H, and $a(\cdot\,,\cdot)$ is an inner product on V, then $(V, a(\cdot\,,\cdot))$ need not be complete if $a(\cdot\,,\cdot)$ is not coercive [3]. When many standard approaches are employed for (1-1), solutions exhibit the well known numerical instability of oscillatory behavior near boundary layers for h not sufficiently fine (e.g., see [17; 12; 22; 14]).

The Galerkin variational formulation of (1-1) for a given $f \in L^2(\Omega)$ is to find a $u \in V$ such that

$$a(u, v) = (f, v) \quad \text{for all } v \in V,$$

where $a(u, v) := (\nabla u, \nabla v) + (\boldsymbol{b} \cdot \nabla u, v) + (cu, v)$.

In the following, we assume constant $\boldsymbol{b}$ and examine coercivity of $a(\cdot\,,\cdot)$ in the absence of boundary conditions:

Proposition 4.2. If we choose $H = V = H^1(\Omega)$, then coercivity holds for $a(\cdot\,,\cdot)$ if and only if $k < 2\sqrt{c}$, where $k = |\boldsymbol{b}|$.

Proof. We first prove the "if" part. Since $k < 2\sqrt{c}$, we can find $0 < c_0 < \min\{1, c\}$ such that

$$k \leq 2\sqrt{(1 - c_0)(c - c_0)}.$$

By the Cauchy–Schwarz inequality, we have that

$$\begin{aligned}|(\boldsymbol{b}\cdot\nabla v, v)| &\le \|\boldsymbol{b}\cdot\nabla v\|\|v\| \le k\cdot\|\nabla v\|\|v\| \\ &\le 2\sqrt{(1-c_0)(c-c_0)}\cdot\|\nabla v\|\|v\| \\ &\le (1-c_0)\|\nabla v\|^2 + (c-c_0)\|v\|^2. \end{aligned} \tag{4-1}$$

Thus,

$$(1-c_0)\|\nabla v\|^2 + (\boldsymbol{b}\cdot\nabla v, v) + (c-c_0)\|v\|^2 \ge 0,$$

which gives us

$$\|\nabla v\|^2 + (\boldsymbol{b}\cdot\nabla v, v) + c\|v\|^2 \ge c_0(\|\nabla v\|^2 + \|v\|^2) = c_0\|v\|_{H^1}^2.$$

To prove the "only if" part, we need to show that, for every $c_0 > 0$, we can always find $v \in H^2(\Omega)\cap V$ such that

$$a(v, v) < c_0\|v\|_{H^1}^2.$$

Since $k \ge 2\sqrt{c}$, for all $c_0 > 0$ satisfying $c_0 < \min\{1, c\}$, we have

$$2\sqrt{(c-c_0)(1-c_0)} < k.$$

We let $\boldsymbol{x} \in \mathbb{R}^d$, where d is the dimension of $\boldsymbol{b}$, and

$$\boldsymbol{a} = -\frac{1}{k}\sqrt{\frac{c-c_0}{1-c_0}}\boldsymbol{b}.$$

Then if we choose $v = e^{\boldsymbol{a}\cdot\boldsymbol{x}}$, we will get

$$\nabla v = v\boldsymbol{a} \tag{4-2}$$

and

$$\boldsymbol{b}\cdot\nabla v = (\boldsymbol{a}\cdot\boldsymbol{b})v = k^2\cdot\left(-\frac{1}{k}\sqrt{\frac{c-c_0}{1-c_0}}v\right) = -k\sqrt{\frac{c-c_0}{1-c_0}}v. \tag{4-3}$$

We notice (4-2) and (4-3) give us the following relation:

$$|\nabla v| = \sqrt{\frac{c-c_0}{1-c_0}}|v|.$$

Thus,

$$\begin{aligned}\|\nabla v\| = \left(\int_\Omega(\nabla v\cdot\nabla v)\right)^{1/2} &= \left(\int_\Omega|\nabla v|^2\right)^{1/2} \\ &= \sqrt{\frac{c-c_0}{1-c_0}}\left(\int_\Omega v^2\right)^{1/2} = \sqrt{\frac{c-c_0}{1-c_0}}\|v\|, \end{aligned} \tag{4-4}$$

and

$$\begin{aligned}(\boldsymbol{b}\cdot\nabla v, v) &= -k\sqrt{\frac{c-c_0}{1-c_0}}\|v\|^2 \\ &< -2\sqrt{(c-c_0)(1-c_0)}\sqrt{\frac{c-c_0}{1-c_0}}\|v\|^2 \\ &= -2(c-c_0)\|v\|^2. \end{aligned} \tag{4-5}$$

By plugging (4-4) into (4-5), we have

$$(\boldsymbol{b}\cdot\nabla v, v) < -2(1-c_0)\|\nabla v\|^2. \tag{4-6}$$

After adding up (4-5) and (4-6), we get

$$(\boldsymbol{b}\cdot\nabla v, v) < -(c-c_0)\|v\|^2 - (1-c_0)\|\nabla v\|^2,$$

which leads to

$$a(v,v) = \|\nabla v\|^2 + (\boldsymbol{b}\cdot\nabla v, v) + c\|v\|^2 < c_0(\|\nabla v\|^2 + \|v\|^2) = c_0\|v\|_{H^1}^2. \quad \square$$

This shows, in part, why convection-dominated problems using the standard Galerkin approach may perform poorly in practice. To address how the weighted variational approach improves the outlook, we recall (2-4),

$$W(v,v) = \|w^{1/2}\nabla v\|^2 + (\nabla v, (\nabla w + w\boldsymbol{b})v) + c\|w^{1/2}v\|^2.$$

When $|\boldsymbol{b}|$ is large and $w = 1$, it is clear that the cross term can dominate the expression. To illustrate the impact of this term, let $k = |\boldsymbol{b}|$ and assume that w is such that

$$|\nabla w + w\boldsymbol{b}| \le \theta|w\boldsymbol{b}| = \theta k|w|$$

for the smallest $\theta \ge 0$ possible. The unweighted case ($w = 1$) corresponds to $\theta = 1$, and we can expect $\theta \to 0$ as $\nabla w + w\boldsymbol{b} \to \boldsymbol{0}$ whenever w is uniformly bounded away from zero (i.e., $w \ge w_{\min} > 0$). The construction of w described in this paper leads to $\nabla w \cdot \boldsymbol{b} < 0$ (i.e., boundary layers form downstream of $\boldsymbol{b}$, where w increases in the opposite direction to $\boldsymbol{b}$), and thus, typically $\theta \in [0, 1]$. Experimental evidence with the construction of w in the test problem in Section 3.1 shows that in regions of Ω near boundary layers elementwise values of θ are in $[0, 1)$ while throughout the interior of Ω values of θ are near 1.

Recalling the general inequality $xy \le (\epsilon/2)x^2 + (1/2\epsilon)y^2$ for any $\epsilon > 0$ and the Cauchy–Schwarz inequality, the coercivity of $W(\cdot,\cdot)$ then follows from

$$\begin{aligned} W(v,v) &= \|w^{1/2}\nabla v\|^2 + (\nabla v, (\nabla w + w\boldsymbol{b})v) + c\|w^{1/2}v\|^2 \\ &\ge \|w^{1/2}\nabla v\|^2 - \theta k\|w^{1/2}\nabla v\|\|w^{1/2}v\| + c\|w^{1/2}v\|^2 \\ &\ge \|w^{1/2}\nabla v\|^2 - \theta k\Big(\frac{\epsilon}{2}\|w^{1/2}\nabla v\|^2 + \frac{1}{2\epsilon}\|w^{1/2}v\|^2\Big) + c\|w^{1/2}v\|^2 \\ &= \Big(1 - \frac{\theta k\epsilon}{2}\Big)\|w^{1/2}\nabla v\|^2 + \Big(c - \frac{\theta k}{2\epsilon}\Big)\|w^{1/2}v\|^2 \\ &\ge c_0(\|w^{1/2}\nabla v\|^2 + \|w^{1/2}v\|^2), \end{aligned}$$

where $0 < c_0 = \min\{1 - \theta k\epsilon/2,\, c - \theta k/2\epsilon\}$ when $\epsilon \in [\theta k/2c,\, 2/\theta k]$ and $k \le 2\sqrt{c}/\theta$. This somewhat formal view shows that the weighted variational problem induces a weighted H^1 norm that may be more desirable than the standard H^1 measure since when $\theta < 1$ the coercivity of $W(\cdot,\cdot)$ will hold for a larger range of k than the standard Galerkin approach, which requires $k \le 2\sqrt{c}$.

When coercivity holds, both the standard Galerkin and weighted Galerkin approaches can easily be shown to have optimal-order error bounds. That is, when (1-1) has full regularity, we have

$$\|u - u^h\| = Ch^2\|u\|_{H^2} \quad \text{and} \quad \|\nabla u - \nabla u^h\| = Ch\|u\|_{H^2},$$

where u is the exact solution, u^h is the numerical approximation with mesh size h (see, e.g., Figure 5). When the coercivity bound holds for a wider range of parameters, it is reasonable to expect more robust numerical results.

5. Conclusion

The weighted scheme we present in this paper seeks to provide a natural reformulation of the variational approach that has an underlying metric adapted to the specific problem. The approach tends to induce an upwinding effect and is flexible in that it does not require specialized meshing or the use of exotic elements. It is possible also to extend the idea to problems with boundary singularities, where overall convergence rates are affected by the loss of smoothness. This study is the subject of a forthcoming investigation.

References

[1] P. B. Bochev, M. D. Gunzburger, and J. N. Shadid, *Stability of the SUPG finite element method for transient advection-diffusion problems*, Comput. Methods Appl. Mech. Engrg. **193** (2004), no. 23-26, 2301–2323. MR 2005a:65100 Zbl 1067.76563

[2] P. B. Bochev and K. Peterson, *A parameter-free stabilized finite element method for scalar advection-diffusion problems*, Cent. Eur. J. Math. **11** (2013), no. 8, 1458–1477. MR 3056328 Zbl 1273.65173

[3] S. C. Brenner and L. R. Scott, *The mathematical theory of finite element methods*, 3rd ed., Texts in Applied Mathematics, no. 15, Springer, New York, 2008. MR 2008m:65001 Zbl 1135.65042

[4] F. Brezzi and A. Russo, *Choosing bubbles for advection-diffusion problems*, Math. Models Methods Appl. Sci. **4** (1994), no. 4, 571–587. MR 95h:76079 Zbl 0819.65128

[5] A. N. Brooks and T. J. R. Hughes, *Streamline upwind/Petrov–Galerkin methods for advection dominated flows*, Proceedings of the Third International Conference on Finite Elements in Flow Problems (Banff, AB, 1980), vol. 2, University of Calgary, Calgary, AB, 1980, edited by D. H. Norrie, pp. 283–292. Zbl 0449.76077

[6] A. N. Brooks and T. J. R. Hughes, *Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations*, Comput. Methods Appl. Mech. Engrg. **32** (1982), no. 1-3, 199–259. MR 83k:76005 Zbl 0497.76041

[7] Z. Cai and C. R. Westphal, *A weighted $H(\mathrm{div})$ least-squares method for second-order elliptic problems*, SIAM J. Numer. Anal. **46** (2008), no. 3, 1640–1651. MR 2008m:65314 Zbl 1168.65069

[8] G. F. Carey and H. T. Dinh, *Grading functions and mesh redistribution*, SIAM J. Numer. Anal. **22** (1985), no. 5, 1028–1040. MR 86h:65123 Zbl 0577.65076

[9] W. Castaings and I. M. Navon, *Mesh refinement strategies for solving singularly perturbed reaction-diffusion problems*, Comput. Math. Appl. **41** (2001), no. 1-2, 157–176. MR 1808513 Zbl 0984.65125

[10] P. Causin, R. Sacco, and C. L. Bottasso, *Flux-upwind stabilization of the discontinuous Petrov–Galerkin formulation with Lagrange multipliers for advection-diffusion problems*, Math. Model. Numer. Anal. **39** (2005), no. 6, 1087–1114. MR 2006j:76085 Zbl 1084.65105

[11] B. Cockburn, B. Dong, and J. Guzmán, *Optimal convergence of the original DG method for the transport-reaction equation on special meshes*, SIAM J. Numer. Anal. **46** (2008), no. 3, 1250–1265. MR 2009c:65299 Zbl 1168.65058

[12] R. Codina, *Comparison of some finite element methods for solving the diffusion-convection-reaction equation*, Comput. Methods Appl. Mech. Engrg. **156** (1998), no. 1-4, 185–210. MR 99d:65280 Zbl 0959.76040

[13] L. Demkowicz and J. Gopalakrishnan, *A class of discontinuous Petrov–Galerkin methods, II: Optimal test functions*, Numer. Methods Partial Differential Equations **27** (2011), no. 1, 70–105. MR 2011k:65155 Zbl 1208.65164

[14] J. Douglas, Jr. and T. F. Russell, *Numerical methods for convection-dominated diffusion problems based on combining the method of characteristics with finite element or finite difference procedures*, SIAM J. Numer. Anal. **19** (1982), no. 5, 871–885. MR 84b:65093 Zbl 0492.65051

[15] H. Egger and J. Schöberl, *A hybrid mixed discontinuous Galerkin finite-element method for convection-diffusion problems*, IMA J. Numer. Anal. **30** (2010), no. 4, 1206–1234. MR 2011j:65271 Zbl 1204.65133

[16] L. P. Franca, S. L. Frey, and T. J. R. Hughes, *Stabilized finite element methods, I: Application to the advective-diffusive model*, Comput. Methods Appl. Mech. Engrg. **95** (1992), no. 2, 253–276. MR 92m:76089 Zbl 0759.76040

[17] L. P. Franca, T. E. Tezduyar, and A. Masud (eds.), *Finite element methods: 1970's and beyond*, Centro Internacional de Métodos Numéricos en Ingeniería, Barcelona, 2004.

[18] T. J. R. Hughes and A. N. Brooks, *A theoretical framework for Petrov–Galerkin methods with discontinuous weighting functions: application to the streamline-upwind procedure*, Finite elements in fluids (Banff, AB, 1980), vol. 4, Wiley, New York, 1982, edited by R. H. Gallager et al., pp. 47–65. MR 84d:76001 Zbl 0571.76002

[19] E. Lee, T. A. Manteuffel, and C. R. Westphal, *Weighted-norm first-order system least squares (FOSLS) for problems with corner singularities*, SIAM J. Numer. Anal. **44** (2006), no. 5, 1974–1996. MR 2008a:65221 Zbl 1129.65087

[20] ———, *Weighted-norm first-order system least-squares (FOSLS) for div/curl systems with three dimensional edge singularities*, SIAM J. Numer. Anal. **46** (2008), no. 3, 1619–1639. MR 2009c:65316 Zbl 1170.65095

[21] H. Lu, *Galerkin and weighted Galerkin methods for a forward-backward heat equation*, Numer. Math. **75** (1997), no. 3, 339–356. MR 97m:65180 Zbl 0876.65071

[22] K. W. Morton, *Numerical solution of convection-diffusion problems*, Applied Mathematics and Mathematical Computation, no. 12, Chapman & Hall, London, 1996. MR 98b:65004 Zbl 0861.65070

[23] H.-S. Oh, B. Jang, and Y. Jou, *The weighted Ritz–Galerkin method for elliptic boundary value problems on unbounded domains*, Numer. Methods Partial Differential Equations **19** (2003), no. 3, 301–326. MR 2004c:65148 Zbl 1021.65058

[24] G. I. Shishkin, *Grid approximation of singularly perturbed boundary value problems with a regular boundary layer*, Soviet J. Numer. Anal. Math. Modelling **4** (1989), no. 5, 397–418. MR 91b:65138 Zbl 0825.65057

[25] ———, *Grid approximation of singularly perturbed boundary value problems with convective terms*, Soviet J. Numer. Anal. Math. Modelling **5** (1990), no. 2, 173–187. MR 92h:65142 Zbl 0816.65051

Received October 27, 2013. Revised June 12, 2014.

YIFEI SUN: yifei@cims.nyu.edu
Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, United States

CHAD R. WESTPHAL: westphac@wabash.edu
Department of Mathematics and Computer Science, Wabash College, Crawfordsville, IN 47933, United States

COMM. APP. MATH. AND COMP. SCI.
Vol. 10, No. 1, 2015
dx.doi.org/10.2140/camcos.2015.10.43

AN ADAPTIVE FINITE VOLUME METHOD FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS IN COMPLEX GEOMETRIES

DAVID TREBOTICH AND DANIEL T. GRAVES

We present an adaptive, finite volume algorithm to solve the incompressible Navier–Stokes equations in complex geometries. The algorithm is based on the embedded boundary method, in which finite volume approximations are used to discretize the solution in cut cells that result from intersecting the irregular boundary with a structured Cartesian grid. This approach is conservative and reduces to a standard finite difference method in grid cells away from the boundary. We solve the incompressible flow equations using a predictor-corrector formulation. Hyperbolic advection terms are obtained by higher-order upwinding without the use of extrapolated data in covered cells. The small-cell stability problem associated with explicit embedded boundary methods for hyperbolic systems is avoided by the use of a volume-weighted scheme in the advection step and is consistent with construction of the right-hand side of the elliptic solvers. The Helmholtz equations resulting from viscous source terms are advanced in time by the Crank–Nicolson method, which reduces solver runtime compared to other second-order time integrators by a half. Incompressibility is enforced by a second-order approximate projection method that makes use of a new conservative cell-centered gradient in cut cells that is consistent with the volume-weighted scheme. The algorithm is also capable of block structured adaptive mesh refinement to increase spatial resolution dynamically in regions of interest. The resulting overall method is second-order accurate for sufficiently smooth problems. In addition, the algorithm is implemented in a high-performance computing framework and can perform structured-grid fluid dynamics calculations at unprecedented scale and resolution, up to 262,144 processor cores. We demonstrate robustness and performance of the algorithm by simulating incompressible flow for a wide range of Reynolds numbers in two and three dimensions: Stokes and low Reynolds number flows in both constructed and image data geometries ($\mathrm{Re} \ll 1$ to $\mathrm{Re} = 1$), flow past a cylinder ($\mathrm{Re} = 300$), flow past a sphere ($\mathrm{Re} = 600$) and turbulent flow in a contraction ($\mathrm{Re} = 6300$).

MSC2010: 35K57, 35Q35, 76D05, 76D07.

Keywords: incompressible Navier–Stokes, embedded boundary method, finite volume method, cut cell method, projection method, adaptive mesh refinement.

1. Introduction

In this paper, we describe a conservative, high-resolution algorithm for the incompressible Navier–Stokes equations in complex geometries. The primary outcome of this work is a simulation capability that can be applied to a wide range of flows where high resolution is sought — from low Reynolds number flow in geologic or engineered porous media, for example, to direct numerical simulation of turbulence. Our approach is based on an adaptive, finite volume embedded boundary method. In the context of a complete description of the overall algorithm, we present several novel numerical techniques including: a volume-weighted scheme for finite volume discretizations that avoids the small-cell problem associated with hyperbolic solvers based on cut cell methods and a stable second-order time integration method that is faster than other second-order schemes used in the context of embedded boundary methods. We demonstrate second-order convergence of the algorithm. We apply the algorithm to benchmark flow past a cylinder in 2D and 3D, flow past a sphere in 3D and high Reynolds number flow in a 2D contraction as well as Stokes flow and low Reynolds number flow in packed bed geometries and realistic subsurface materials. We also demonstrate the adaptive mesh refinement capability of the algorithm as well as scalable performance to 262,144 processor cores.

1.1. ***Equations of motion.*** We consider the incompressible Navier–Stokes equations with constant density:

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = -\nabla p + \nu\Delta\boldsymbol{u}, \tag{1-1}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{1-2}$$

where $\boldsymbol{u}$ is the fluid velocity, ∇p is the pressure gradient and ν is the kinematic viscosity. To close the system, we specify boundary conditions for a bounded inflow-outflow problem. For example, for flow in the x direction in a two-dimensional channel, the boundary conditions are, at inflow,

$$\boldsymbol{u} = (\tfrac{3}{2}\bar{u}(1-y^2/a^2), 0), \quad \frac{\partial p}{\partial x} = 0, \tag{1-3}$$

where $\bar{u}$ is average inflow velocity and a is half the width of the channel in the y direction; at solid walls,

$$\boldsymbol{u} = 0, \quad \frac{\partial p}{\partial y} = 0; \tag{1-4}$$

and at outflow,

$$\frac{\partial \boldsymbol{u}}{\partial x} = 0, \quad p = 0. \tag{1-5}$$

Given initial conditions $\boldsymbol{u}^0 = \boldsymbol{u}(\boldsymbol{x}, 0)$ and $p^0 = p(\boldsymbol{x}, 0)$, the system of equations defined by (1-1)–(1-5) constitutes an initial boundary value problem (IBVP) that can

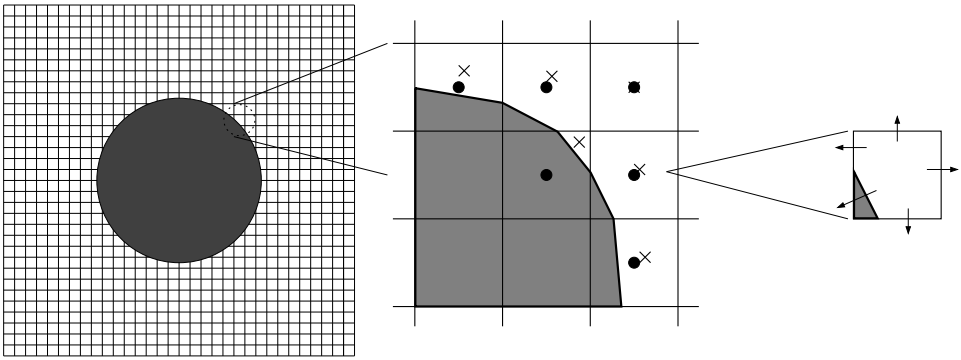


Figure 1. Left: example of an irregular geometry on a Cartesian grid. Middle: close-up view of embedded boundaries "cutting" regular cells. Right: single cut cell showing boundary fluxes. The shaded area represents the volume of cells excluded from the domain. Dots represent cell centers. Each "×" represents a centroid.

be solved by a variety of methods (e.g., immersed boundary [27], ghost fluid [18] or discontinuous Galerkin [46]). We are ultimately interested in efficient, scalable computations to obtain high resolution for a wide range of Reynolds number flows in complex geometries. We address this problem with a predictor-corrector projection formulation based on a finite volume, embedded boundary method with adaptive mesh refinement.

1.2. *Numerical approach.*

1.2.1. *Embedded boundary method.* Cartesian grid methods have become an increasingly popular modeling approach to solving partial differential equations (PDEs) in complex geometries. There are several Cartesian grid approaches (e.g., immersed boundary [42], immersed interface [28] and ghost fluid [18]); however, we focus on the cut cell approach, which is based on finite volume approximations. A cut cell, or embedded boundary, method refers to a finite volume discretization in irregular cells on a Cartesian grid that result from the intersection of the boundary and the rectangular cells of the grid (see Figure 1). Conservative numerical approximations to the solution can be found from discrete integration over the nonrectangular control volumes, or cut cells, with fluxes located at centroids of the edges or faces of a control volume. This approach has been used as the basis for second-order accurate methods for elliptic, parabolic and hyperbolic PDEs in two and three dimensions [21; 32; 15].

One of the advantages of the method is that the problem of generating the description of complex geometry on the grid (starting from, for example, surface tessellations produced by a CAD system or implicit function representation of x-ray microtomography images) has been made more tractable [1; 29]. Another advantage of the embedded boundary method is that it is amenable to adaptive mesh refinement (AMR) [10]. Block structured AMR is a technique to add grid

resolution efficiently and dynamically in areas of interest while leaving the rest of the domain at a coarser resolution. AMR was originally applied to finite difference methods for inviscid shock hydrodynamics [7] and has been extended to inviscid, incompressible flow [30] and viscous flow [2; 31] in rectangular domains. AMR has been combined with embedded boundary methods to model inviscid and viscous compressible flows in complex geometries [40; 15; 19].

For incompressible flows, embedded boundary methods have been mostly applied to inviscid flows (e.g., [3]). As attractive as cut cell methods are for efficient gridding of complex geometries, these methods are still gaining ground in the engineering community for modeling of incompressible viscous flows perhaps due to the high resolution that is required in and around cut cells to resolve viscous boundary layers. Therefore, AMR and high-performance computing have become necessary partners for cut cell methods to be effective 3D modeling tools. Furthermore, discussion of such methods usually centers around the "small-cell problem" due to the arbitrary nature of the cut cell approach; also of importance are accuracy of gradients and higher-order strategies.

Several methods have been proposed for incompressible viscous flow using the cut cell approach. In [60], a single-grid (nonadaptive) finite volume method is used for 2D incompressible viscous flows and is demonstrated on an array of cylinders in a channel. In [43], adaptivity is combined with a volume-of-fluid method and applied to practical engineering problems in 3D. Cell-merging is used to treat the small-cell problem. Second-order accuracy is demonstrated with particular attention given to the pressure gradient. In [25], a cut cell method on a staggered grid is applied to a moderate Reynolds number flow in 3D. A "cell-linking" method is proposed that links small cells with a master cell, placing the cell a small distance from the master and inducing a high-diffusion flux that forces the two velocities to take the same value. A cell-merging technique was used in [13] as part of a cut cell projection method. In a precursor [52] to the work presented here, an embedded boundary method was used to model fluid-particle flow through a packed bed geometry. This work was later generalized to AMR in a computationally efficient framework using novel stenciling techniques in cut cells [53]. The small-cell problem was addressed by a linear hybridization of conservative and nonconservative estimates of the convective derivative akin to [11; 6] with redistribution of the unconserved mass.

1.2.2. *Projection method.* Projection methods address the time-discretization issue of the constrained evolution equations of incompressible flow. These methods are based on the Hodge–Helmholtz decomposition of a vector field into a divergence-free part and a gradient of a scalar field, effectively separating the vortical dynamics induced by a viscous, divergence-free velocity field from the potential flow problem. Projection methods have taken several different paths since Chorin's original method was introduced [12], primarily depending on the choice of scheme for higher-order

discretization of the nonlinear advection term (e.g., [24; 55]). Our approach is based on the work of Bell, Colella and Glaz (BCG) [4] and the family of methods that followed (e.g., [26; 2; 49; 31]). The BCG method makes use of high-resolution finite difference methods for hyperbolic PDEs, such as Godunov or upwinding schemes, combined in a fractional step approach with fast iterative methods for elliptic and parabolic PDEs to achieve second-order spatial and temporal accuracy. BCG was made more robust for larger Courant–Friedrichs–Lewy (CFL) numbers with the introduction of an intermediate marker-and-cell (MAC) projection in the advection step [5]. In [49], the BCG method was extended to time-dependent domains using quadrilateral, mapped grids and a consistent decomposition of the velocity field that standardized the implementation of boundary conditions for projection methods. The projection method was generalized to a nonadaptive embedded boundary approach for time-dependent domains in [34].

In this paper, we combine these methods — adaptive, finite volume and projection — using the predictor-corrector projection formulation in [49], the adaptive approach in [31] and the computational fluid dynamics tools for adaptive embedded boundary methods in [53] to solve the incompressible Navier–Stokes equations in complex geometries. The algorithm is implemented in the Chombo software framework, which supports adaptive, embedded boundary methods and also enables large scale computations (http://chombo.lbl.gov). The resulting algorithm is conservative, second-order accurate and scalable to 262,144 processor cores. The central new idea of the algorithm is a volume-weighted scheme that avoids the small-cell problem associated with explicit embedded boundary methods and leads to better stability properties than previous approaches in [53; 15; 52].

We organize the discussion of the algorithm as follows. The finite difference algorithm is described in its entirety in Section 2. The embedded boundary, finite volume method is described in Section 3 for the case of cut cells where the discretization requires special stencils that differ from the finite difference method. For ease of exposition, the algorithm is described in 2D; the 3D discretization is included if it is not an obvious extension from 2D. We include brief descriptions of algorithm modifications needed for AMR throughout the discussion and particularly for hyperbolic and elliptic discretizations near coarse-fine interfaces. Accuracy of the method, performance measurements and simulation results are presented in Section 4. Conclusions are summarized and discussed in Section 5.

2. Algorithm discretization

A second-order accurate in time discretization of the evolution equation (1-1) is

$$U^{n+1} = U^n + \Delta t(\nu \Delta U^{n+1/2} - (U \cdot \nabla)U^{n+1/2} - \nabla p^{n+1/2}),$$

where U^n is an approximation of the velocity field at the discrete time $t^n = t^{n-1} + \Delta t$. We choose a second-order upwind method for hyperbolic terms, a second-order implicit discretization of parabolic terms and an approximate projection method to enforce incompressibility with special centering of the pressure gradient. We combine these methods in a semi-implicit predictor-corrector formulation based on [49] to advance the solution.

2.1. ***Temporal discretization.*** The momentum equation (1-1) can be formulated as a parabolic equation of the form $U_t = \mathcal{L}(U) + f(U)$, where $\mathcal{L}$ is a second-order elliptic operator such as the Laplacian. Second-order accuracy in time can be achieved by the Crank–Nicolson method for parabolic equations as in [4]. It has been previously reported that, in the presence of embedded boundaries, the Crank–Nicolson scheme is unstable for parabolic equations, and in particular, when the embedded boundary is moving, coefficients are strongly varying or discontinuities exist in the solution [32]. Instead, the Runge–Kutta method of [54] is recommended to achieve second-order accuracy. In practice, we have not experienced such instabilities with Crank–Nicolson for stationary boundaries nor in current work with moving boundaries (e.g., [34]). Furthermore, significant computational savings are gained from the use of Crank–Nicolson, which requires only D (number of space dimensions) solutions to the Helmholtz problem while the second-order Runge–Kutta method as described in [54] requires $2D$ solutions to the Helmholtz problem.

The Crank–Nicolson discretization is

$$\left(I - \frac{\nu \Delta t}{2}\Delta\right) U^{n+1,*} = \left(I + \frac{\nu \Delta t}{2}\Delta\right) U^n + \Delta t f^{n+1/2}, \tag{2-1}$$

$$f^{n+1/2} = -(U \cdot \nabla) U^{n+1/2} - \nabla p^{n-1/2}. \tag{2-2}$$

The intermediate velocity, $U^{n+1,*}$, in (2-1) is a second-order approximation to the solution that satisfies the boundary conditions but does not necessarily satisfy the incompressibility constraint due to the lagged pressure gradient in (2-2).

2.2. ***Projection formulation.*** The projection method [12] is used to enforce incompressibility in discretization (2-1). In general, a smooth vector field, $\boldsymbol{w}$, on a simply connected domain, Ω, can be orthogonally decomposed into a divergence-free component, $\boldsymbol{w}_d$, and a gradient of a scalar potential, ψ,

$$\begin{aligned} \boldsymbol{w} &= \boldsymbol{w}_d + \nabla \psi, \\ \nabla \cdot \boldsymbol{w}_d &= 0, \\ \Delta \psi &= \nabla \cdot \boldsymbol{w} \end{aligned}$$

with boundary conditions $\boldsymbol{w}_d \cdot \boldsymbol{n} = 0$ and $\partial\psi/\partial n = \boldsymbol{w} \cdot \boldsymbol{n}$ on $\partial\Omega$. We can apply a discrete version of the projection to the discretization (2-1) to obtain a divergence-free velocity and pressure gradient

$$\begin{aligned} U^{n+1} &= \boldsymbol{P}(W), \\ \nabla p^{n+1/2} &= \frac{1}{\Delta t}\boldsymbol{Q}(W), \\ W &= U^{n+1,*} + \Delta t \nabla p^{n-1/2}, \end{aligned}$$

where $\boldsymbol{Q} = \boldsymbol{G}\boldsymbol{L}^{-1}\boldsymbol{D}$, $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{Q}$ and $\boldsymbol{L}$, $\boldsymbol{D}$ and $\boldsymbol{G}$ are discrete representations of the Laplacian, divergence and gradient, respectively. These projection operations procedurally reduce to the solution of the Poisson problem and an update of the velocity and pressure gradient by the gradient of the solution to the Poisson problem

$$\boldsymbol{L}\phi = \boldsymbol{D}(W), \tag{2-3}$$

$$U^{n+1} = W - \boldsymbol{G}(\phi), \tag{2-4}$$

$$\nabla p^{n+1/2} = \frac{1}{\Delta t}\boldsymbol{G}(\phi). \tag{2-5}$$

We note that the projection target, W, contains the intermediate velocity augmented by the lagged pressure gradient resulting in a pressure formulation with improved stability in comparison to the pressure correction formulation in [49].

The form of $\boldsymbol{G}$, and thus $\boldsymbol{L}$, depends on the centering of the projection target, W, which is cell-centered in this case. However, the discretization of divergence, based on the discrete form of the divergence theorem, is applied as a sum of differences of *face-centered* data in each direction. In compact notation, we have

$$\boldsymbol{D}(W)_{\boldsymbol{i}} = \frac{1}{h}\sum_{d=1}^{D}(W_{\boldsymbol{i}+\hat{e}^d/2} - W_{\boldsymbol{i}-\hat{e}^d/2}). \tag{2-6}$$

If we also consider that the gradient is applied to the cell-centered solution to Poisson's equation, $\phi_{\boldsymbol{i}}$, then $\boldsymbol{D}$ and $\boldsymbol{G}$ are not discrete adjoints and $\boldsymbol{L} \neq \boldsymbol{D}\boldsymbol{G}$. This projection is, therefore, approximate and $\boldsymbol{D}(\boldsymbol{P}(W)) = O(h^2)$, the same magnitude as the truncation error. Also, the operator is not idempotent; i.e., $\boldsymbol{P}^2 \neq \boldsymbol{P}$.

If the divergence and gradient are discrete adjoints and $\boldsymbol{L} \equiv \boldsymbol{D}\boldsymbol{G}$, then the projection is discretely exact, i.e., $\boldsymbol{D}(\boldsymbol{P}(W)) = 0$; see [12]. This is the case of the so-called MAC projection, defined to be $\boldsymbol{P}^{\text{mac}} \equiv (\boldsymbol{I} - \boldsymbol{Q}^{\text{mac}})$ and $\boldsymbol{Q}^{\text{mac}} \equiv \boldsymbol{G}^{\text{mac}}(\boldsymbol{L}^{\text{mac}})^{-1}\boldsymbol{D}$. The discrete Laplacian operator can then be defined as the conservative divergence of the face-centered gradient:

$$\boldsymbol{L}_{\boldsymbol{i}} \equiv \boldsymbol{D}(\boldsymbol{G}^{\text{mac}}(\phi))_{\boldsymbol{i}} = \frac{1}{h}\sum_{d=1}^{D}(\boldsymbol{G}^{\text{mac},d}(\phi)_{\boldsymbol{i}+\hat{e}^d/2} - \boldsymbol{G}^{\text{mac},d}(\phi)_{\boldsymbol{i}-\hat{e}^d/2}). \tag{2-7}$$

This is the finite difference discretization of the Laplacian used in the various elliptic operators throughout the algorithm such as in (2-3).

The cell-centered projection can be constructed by wrapping two averaging operators around the MAC projection:

$$\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{A}^{F\to C}(\boldsymbol{Q}^{\mathrm{mac}}(\boldsymbol{A}^{C\to F})).$$

First, an operator to average cell-centered velocities to face centers is needed for the divergence in (2-3). For a face with a normal direction d, the averaging operation is

$$\boldsymbol{A}^{C\to F}(W^d)_{\boldsymbol{i}+\hat{e}^d/2} = \tfrac{1}{2}(W^d_{\boldsymbol{i}+\hat{e}^d} + W^d_{\boldsymbol{i}}). \tag{2-8}$$

Then, an averaging operator that is used to average gradients from face centers to cell centers is defined by

$$\boldsymbol{A}^{F\to C}(\boldsymbol{G}^{\mathrm{mac}}(\phi)^d)_{\boldsymbol{i}} = \tfrac{1}{2}(\boldsymbol{G}^{\mathrm{mac},d}_{\boldsymbol{i}+\hat{e}^d/2}(\phi) + \boldsymbol{G}^{\mathrm{mac},d}_{\boldsymbol{i}-\hat{e}^d/2}(\phi)). \tag{2-9}$$

The averaging operator, $\boldsymbol{A}^{F\to C}$, effectively results in a centered-difference for the gradient away from boundaries.

The face-centered gradient, $\boldsymbol{G}^{\mathrm{mac}}$, of a cell-centered scalar, $\phi_{\boldsymbol{i}}$, is defined to be the finite difference approximation in the normal direction of the face:

$$\boldsymbol{G}^{\mathrm{mac}}(\phi)^d_{\boldsymbol{i}+\hat{e}^d/2} = \frac{1}{h}(\phi_{\boldsymbol{i}+\hat{e}^d} - \phi_{\boldsymbol{i}}).$$

For homogeneous Neumann domain boundary conditions, this gradient is 0; for Dirichlet domain boundary conditions, we use an odd extension of the solution at the boundary to obtain the gradient.

The transverse gradient at a face is the average of neighboring normal gradients in transverse directions, d', to a d-face:

$$\boldsymbol{G}^{\mathrm{mac}}(\phi)^{d'}_{\boldsymbol{i}+\hat{e}^d/2} = \frac{1}{N_{\mathcal{G}}} \sum_{\boldsymbol{i}+\hat{e}^{d'}/2\in\mathcal{G}^{d',d}} \left(\boldsymbol{G}^{\mathrm{mac}}(\phi)^{d'}_{\boldsymbol{i}+\hat{e}^{d'}/2}\right),$$

where $\mathcal{G}^{d',d}$ is the set of faces in the transverse d' direction and $N_{\mathcal{G}}$ is the number of faces in this set. On a regular grid, $\mathcal{G}$ is the set of four neighboring adjacent faces in a d' direction. At solid wall domain boundaries, linear extrapolation of transverse gradients is used to preserve a constant pressure gradient as in Poiseuille flow. For transverse gradients whose face stencil crosses an orthogonal domain boundary, the appropriate one-sided difference is taken.

MAC gradients that are preprocessed by the averaging operator, $\boldsymbol{A}^{F\to C}$, in (2-9) make use of linear extrapolation at boundary faces from interior faces to avoid over-specification of the problem. For the cell-centered projection target that is preprocessed by the averaging operator, $\boldsymbol{A}^{C\to F}$, in (2-8), boundary conditions are applied to the normal component. Referring to the channel boundary conditions

(1-3)–(1-5), these are $W \cdot \boldsymbol{n} = u_{\text{in}}$ at the inlet, $W \cdot \boldsymbol{n} = 0$ at no-flow solid walls and a quadratic extrapolation at the outlet that satisfies the Neumann boundary condition. Projection operator gradients are matched at coarse-fine interfaces by simple averaging as in [30].

2.3. ***Hyperbolic discretization.*** The advection term, $(U \cdot \nabla)U_{i,j}^{n+1/2}$, in (1-1) is discretized in conservation form since the flow is incompressible:

$$\nabla \cdot (UU)_{i,j}^{n+1/2} = \frac{1}{h}\left(u_{i+1/2,j}^{n+1/2} U_{i+1/2,j}^{n+1/2} - u_{i-1/2,j}^{n+1/2} U_{i-1/2,j}^{n+1/2} + v_{i,j+1/2}^{n+1/2} U_{i,j+1/2}^{n+1/2} - v_{i,j-1/2}^{n+1/2} U_{i,j-1/2}^{n+1/2}\right),$$

where i and j are the cell-centered grid indices in two dimensions, n is the number of the timestep and $U = (u, v)$. Here, we consider two dimensions for ease of exposition — one direction that is normal to the flow (x) and one transverse (y) — with obvious extension of the transverse discretization to a third dimension (z).

We use an upstream-centered Taylor expansion to extrapolate the cell-centered velocity to the half step in time and cell edges:

$$\widehat{U}_{i+1/2,j}^{n+1/2} = U_{i,j}^{n} + \frac{\Delta x}{2}\frac{\partial U^n}{\partial x} + \frac{\Delta t}{2}\frac{\partial U^n}{\partial t}.$$

Substitution of the PDE for the temporal derivative into the Taylor expansion yields extrapolated velocities in all directions from the cell center to both sides of a cell edge (or face in 3D):

$$\widehat{U}_{i,j}^{x,+} = U_{i,j}^{n} + \tfrac{1}{2}\min\left[\left(1 - u_{i,j}^{n}\frac{\Delta t}{\Delta x}\right), 1\right](\delta_x^N U)_{i,j}^{n} - \frac{\Delta t}{2\Delta y} v_{i,j}^{n} (\delta_y^T U)_{i,j}^{n} + \frac{\nu \Delta t}{2}\Delta U_{i,j}^{n},$$

$$\widehat{U}_{i,j}^{x,-} = U_{i,j}^{n} - \tfrac{1}{2}\min\left[\left(1 + u_{i,j}^{n}\frac{\Delta t}{\Delta x}\right), 1\right](\delta_x^N U)_{i,j}^{n} - \frac{\Delta t}{2\Delta y} v_{i,j}^{n} (\delta_y^T U)_{i,j}^{n} + \frac{\nu \Delta t}{2}\Delta U_{i,j}^{n},$$

$$\widehat{U}_{i,j}^{y,+} = U_{i,j}^{n} + \tfrac{1}{2}\min\left[\left(1 - v_{i,j}^{n}\frac{\Delta t}{\Delta y}\right), 1\right](\delta_y^N U)_{i,j}^{n} - \frac{\Delta t}{2\Delta x} u_{i,j}^{n} (\delta_x^T U)_{i,j}^{n} + \frac{\nu \Delta t}{2}\Delta U_{i,j}^{n},$$

$$\widehat{U}_{i,j}^{y,-} = U_{i,j}^{n} - \tfrac{1}{2}\min\left[\left(1 + v_{i,j}^{n}\frac{\Delta t}{\Delta y}\right), 1\right](\delta_y^N U)_{i,j}^{n} - \frac{\Delta t}{2\Delta x} u_{i,j}^{n} (\delta_x^T U)_{i,j}^{n} + \frac{\nu \Delta t}{2}\Delta U_{i,j}^{n},$$

where superscripts x and y refer to the coordinate direction of the extrapolation and "+" and "−" indicate the direction of the extrapolation from the cell center to the inside of an edge (inside left/bottom of an edge is the "+" state; inside right/bottom is the "−" state).

The monotonized second-order normal slopes with van Leer limiting [56; 15] are

$$(\delta_x^N U)_{i,j}^{n} = \begin{cases} (\delta_x U)^{vL} & \text{if } (U_{i+1,j}^{n} - U_{i,j}^{n})(U_{i,j}^{n} - U_{i-1,j}^{n}) > 0, \\ 0 & \text{if } (U_{i+1,j}^{n} - U_{i,j}^{n})(U_{i,j}^{n} - U_{i-1,j}^{n}) \le 0, \end{cases}$$

where

$$(\delta_x U)^{vL} = \text{sign}(U^n_{i+1,j} - U^n_{i-1,j}) \times \min\big(2|U^n_{i,j} - U^n_{i-1,j}|, 2|U^n_{i+1,j} - U^n_{i,j}|, \tfrac{1}{2}|U^n_{i+1,j} - U^n_{i-1,j}|\big).$$

The upwinded transverse slopes are

$$(\delta^T_y U)^n_{i,j} = \begin{cases} U^n_{i,j+1} - U^n_{i,j} + \frac{1}{2}\nu\Delta t(\Delta U^n_{i,j+1} - \Delta U^n_{i,j}) & \text{if } v^n_{i,j} < 0, \\ U^n_{i,j} - U^n_{i,j-1} + \frac{1}{2}\nu\Delta t(\Delta U^n_{i,j} - \Delta U^n_{i,j-1}) & \text{if } v^n_{i,j} \geq 0, \end{cases}$$

$$(\delta^T_x U)^n_{i,j} = \begin{cases} U^n_{i+1,j} - U^n_{i,j} + \frac{1}{2}\nu\Delta t(\Delta U^n_{i+1,j} - \Delta U^n_{i,j}) & \text{if } u^n_{i,j} < 0, \\ U^n_{i,j} - U^n_{i-1,j} + \frac{1}{2}\nu\Delta t(\Delta U^n_{i,j} - \Delta U^n_{i-1,j}) & \text{if } u^n_{i,j} \geq 0 \end{cases}$$

with a stability correction due to [35]. All slopes make use of one-sided differences at domain boundaries; at coarse-fine boundaries, we use linear interpolation and flux matching [31].

A Riemann problem is then solved to obtain the edge states, $\bar{U}$. For example, x-face states are

$$\bar{U}_{i+1/2,j} = \begin{cases} \widehat{U}^{x,+}_{i,j} & \text{if } \frac{1}{2}(u^n_{i,j} + u^n_{i+1,j}) > 0, \\ \widehat{U}^{x,-}_{i+1,j} & \text{if } \frac{1}{2}(u^n_{i,j} + u^n_{i+1,j}) < 0, \\ \frac{1}{2}\big(\widehat{U}^{x,+}_{i,j} + \widehat{U}^{x,-}_{i+1,j}\big) & \text{if } \frac{1}{2}(u^n_{i,j} + u^n_{i+1,j}) = 0. \end{cases}$$

To make up for the omitted pressure gradient in the velocity extrapolation, the solution to the Riemann problem is projected onto the space of divergence-free vectors using a MAC projection

$$U^{n+1/2} = \boldsymbol{P}^{\text{mac}}(\bar{U}) = \bar{U} - \boldsymbol{G}^{\text{mac}}((\boldsymbol{D}\boldsymbol{G}^{\text{mac}})^{-1}\boldsymbol{D}(\bar{U})). \tag{2-10}$$

The divergence is calculated as

$$\boldsymbol{D}(\bar{U}^{n+1/2})_{i,j} = \big[(\bar{u}_{i+1/2,j} - \bar{u}_{i-1/2,j}) + (\bar{v}_{i,j+1/2} - \bar{v}_{i,j-1/2})\big]/h.$$

Both components of velocity have been accounted for up to this point, including the boundary conditions for the normal component. The transverse component of velocity at domain boundaries is taken to be the "+" or "−" state on the inside of the boundary edge in keeping with the idea of an inviscid predictor step.

Our method has a stability constraint on the timestep due to the CFL condition for the advection terms

$$\Delta t < \frac{\sigma h}{u^{\max}}, \tag{2-11}$$

where $\sigma \leq 1$ and $u^{\max}$ is the magnitude of the maximum local wave speed. For adaptive calculations, all levels of refinement use the same timestep. We note that subcycling in time is possible; however, it requires the solution to an additional Poisson equation to enforce the divergence-free constraint with free-stream preservation [31].

3. Finite volume embedded boundary method

In grid cells where the irregular domain intersects the Cartesian grid, finite volume discretizations must be constructed to obtain conservative discretizations of flux-based operations defined by finite differences in the previous section. First, the underlying description of space is given by rectangular control volumes on a Cartesian grid $\Upsilon_{\boldsymbol{i}} = [(\boldsymbol{i} - \frac{1}{2}\boldsymbol{V})h, (\boldsymbol{i} + \frac{1}{2}\boldsymbol{V})h]$, $\boldsymbol{i} \in \mathbb{Z}^D$, where D is the dimensionality of the problem, h is the mesh spacing and $\boldsymbol{V}$ is the vector whose entries are all 1. Given an irregular domain Ω, we obtain control volumes $V_{\boldsymbol{i}} = \Upsilon_{\boldsymbol{i}} \cap \Omega$ and faces $A_{\boldsymbol{i} \pm \hat{e}^d/2}$, which are the intersections of the boundary of $\partial V_{\boldsymbol{i}}$ with the coordinate planes $\{\vec{x} : x_d = (i_d \pm \frac{1}{2})h\}$. The intersections of the boundary of the irregular domain with the Cartesian control volume are defined as $A_{\boldsymbol{i}}^B = \partial\Omega \cap \Upsilon_{\boldsymbol{i}}$. For ease of exposition, it is assumed that there is only one control volume per Cartesian cell. However, the algorithm described here has been generalized to allow for boundaries whose width is less than the mesh spacing, i.e., multivalued cells. In regular cells, the finite volume approximation reduces to the finite difference method described in Section 2.

To construct finite volume methods using this description, several quantities need to be derived from the geometric objects:

- volume fractions, $\kappa_{\boldsymbol{i}}$, and area fractions, $\alpha_{\boldsymbol{i}}$,

$$\kappa_{\boldsymbol{i}} = \frac{|V_{\boldsymbol{i}}|}{h^D}, \quad \alpha_{\boldsymbol{i}+\hat{e}_s/2} = \frac{|A_{\boldsymbol{i}+\hat{e}_s^d/2}|}{h^{(D-1)}}, \quad \alpha_{\boldsymbol{i}}^B = \frac{|A_{\boldsymbol{i}}^B|}{h^{D-1}},$$

- centroids of the faces and of $A_{\boldsymbol{i}}^B$,

$$\vec{x}_{\boldsymbol{i}+\hat{e}^d/2} = \left[\frac{1}{|A_{\boldsymbol{i}+\hat{e}^d/2}|}\int_{A_{i+\hat{e}^d/2}} \vec{x}\, dA\right], \quad \vec{x}_{\boldsymbol{i}}^B = \left[\frac{1}{|A_{\boldsymbol{i}}^B|}\int_{A_{\boldsymbol{i}}^B} \vec{x}\, dA\right],$$

- the average of outward normals of $\partial\Omega$ over $A_{\boldsymbol{i}}^B$,

$$\hat{n}_{\boldsymbol{i}} = \frac{1}{|A_{\boldsymbol{i}}^B|}\int_{A_{\boldsymbol{i}}^B} \hat{n}\, dA,$$

where D is the dimension of space and $1 \leq d \leq D$. We assume we can compute all derived quantities to $O(h^2)$.

Geometric objects are determined by a hierarchical application of the divergence theorem to discrete values of implicit function representations of the geometry on the grid (see [34; 29; 45] for details on embedded boundary grid generation).

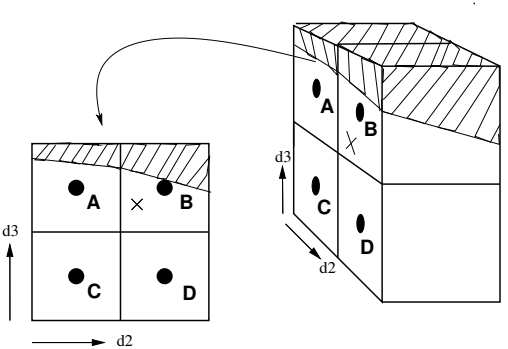


Figure 2. 3D bilinear flux interpolation stencil showing the interpolation point marked by an "×" for a 3D face in the $\hat{e}^1$ direction using face-centered points A, B, C and D.

Coarsened geometries are obtained by coarsening of the graph. The volume of a coarse cell is exactly the volume of the fine cells that it comprises. This grid generation machinery is part of the Chombo software libraries.

The conservative approximation of the divergence of a flux $\vec{F}$ can now be defined by applying a discrete form of the divergence theorem

$$\boldsymbol{D}(\vec{F})_{\boldsymbol{v}} = \frac{1}{h\kappa_{\boldsymbol{v}}}\left(\sum_{d=1}^{D}\left(\alpha_{\boldsymbol{i}+\hat{e}^d/2}\widetilde{F}^d_{\boldsymbol{i}+\hat{e}^d/2} - \alpha_{\boldsymbol{i}-\hat{e}^d/2}\widetilde{F}^d_{\boldsymbol{i}-\hat{e}^d/2}\right) + \alpha^B_{\boldsymbol{v}} F^B_{\boldsymbol{v}}\right), \tag{3-1}$$

where $\widetilde{F}_{\boldsymbol{i}+\hat{e}^d/2}$ indicates that the flux has been interpolated to the face centroid using linear (2D) or bilinear (3D) interpolation of face-centered fluxes. For example, given the cell edge with outward normal $\hat{e}^1$, with centroid $\vec{x}$, the 2D linearly interpolated flux in the d direction ($d \neq 1$) is defined by

$$\begin{aligned}\widetilde{F}^d_{\boldsymbol{i}+\hat{e}^1/2} &= \eta F_{\boldsymbol{i}+\hat{e}^1/2} + (1-\eta)F_{\boldsymbol{i}+\hat{e}^1/2\pm\hat{e}^d}, \\ \eta &= 1 - \frac{|\vec{x}\cdot\hat{e}^d|}{h_d}, \\ \pm &= \begin{cases} + & \text{if } \vec{x}\cdot\hat{e}^d > 0, \\ - & \text{if } \vec{x}\cdot\hat{e}^d \leq 0. \end{cases}\end{aligned} \tag{3-2}$$

The 3D bilinear interpolation of the flux for a face with normal $\hat{e}^1$ can be written as

$$\begin{aligned}\widetilde{F}_{\boldsymbol{i}+\hat{e}^1/2} &= \omega F^d_{\boldsymbol{i}+\hat{e}^1/2} + (1-\omega)F^d_{\boldsymbol{i}\pm\hat{e}^{d'}+\hat{e}^1/2}, \\ \omega &= 1 - \frac{|\vec{x}\cdot\hat{e}^{d'}|}{h_{d'}}, \\ \pm &= \begin{cases} + & \text{if } \vec{x}\cdot\hat{e}^{d'} > 0, \\ - & \text{if } \vec{x}\cdot\hat{e}^{d'} \leq 0, \end{cases}\end{aligned} \tag{3-3}$$

where $d' \neq d$ and $d' \neq 1$ (see Figure 2).

3.1. ***Elliptic operators.*** The conservative discretization of the divergence theorem in (3-1) provides a flux-based formula for the discretization of the elliptic operations in the algorithm. We use the geometric multigrid approach described in [53] to solve these elliptic systems. In the context of Poisson's equation, as in the projections (2-3) and (2-10), the operator is the Laplacian and the flux is simply the gradient of a scalar, $\vec{F} = \nabla\varphi$, with Neumann boundary conditions, $F^B = \hat{n} \cdot \nabla\varphi = 0$, at the embedded boundary. However, in the context of Helmholtz, as in (2-1), the embedded boundary is a no-slip boundary for the velocity, requiring an elliptic operator with Dirichlet boundary conditions at the embedded boundary. In this case, the flux at the embedded boundary, $F^B = \hat{n} \cdot \nabla\varphi$, must be constructed while maintaining global second-order accuracy.

In [21], the flux at the embedded boundary due to a Dirichlet boundary condition is constructed by effectively casting a ray from the centroid of the boundary along the normal into the domain, interpolating φ to points along the ray (quadratic in 2D and biquadratic in 3D), computing the normal gradient of φ by differencing the interpolated points in 1D along the ray and obtaining a $\tau \approx O(h^2)$ local truncation error approximation of $\hat{n} \cdot \nabla\varphi$. In general, local truncation error on the interior of a domain is $\tau \approx O(h^2)$ and at the boundary $\tau \approx O(h/\kappa)$. It can be shown that global solution error is $\varepsilon = O(h^2)$. For the case of Dirichlet boundary conditions, the same conclusion holds for $\tau = O(1)$ at the boundary owing to the two orders of magnitude of freedom in the local truncation error resulting from the method of images (odd extensions) and the homogeneous condition at the boundary. (For Neumann boundary conditions, the minimum requirement to maintain second-order global error is $\tau = O(h)$.) The conclusion for Dirichlet boundary conditions, shown in [21] using potential theory, is that it is sufficient to have $O(1)$ boundary conditions to achieve second-order convergence of solution error for elliptic equations.

In practice, however, we have found that the second-order stencil for Dirichlet boundary conditions first described in [21] is not stable for lower Reynolds number flows (much less the Stokes limit) and flows where there exist steep gradients near the boundary. To fix this instability, we make use of the two orders of magnitude of freedom in the local truncation error and instead apply a lower-order truncation error stencil ($\tau \approx O(h)$) to interpolate the flux at the irregular boundary centroid, $\boldsymbol{B}$ [50; 52; 51; 47]. The flux, $\hat{n} \cdot \nabla\varphi$, is obtained by solving a least squares linear system for $\nabla\varphi$:

$$\begin{aligned}
\mathcal{A} \cdot \nabla\varphi &= \delta\varphi, \\
\mathcal{A} &= (\delta\vec{x}_1, \delta\vec{x}_2, \ldots, \delta\vec{x}_p)^T, \\
\delta\varphi &= (\delta\varphi_1, \delta\varphi_2, \ldots, \delta\varphi_p)^T, \\
\delta\vec{x}_m &= \vec{x}_m - \vec{x}_B, \\
\delta\varphi_m &= \varphi_m - \varphi_B.
\end{aligned}$$

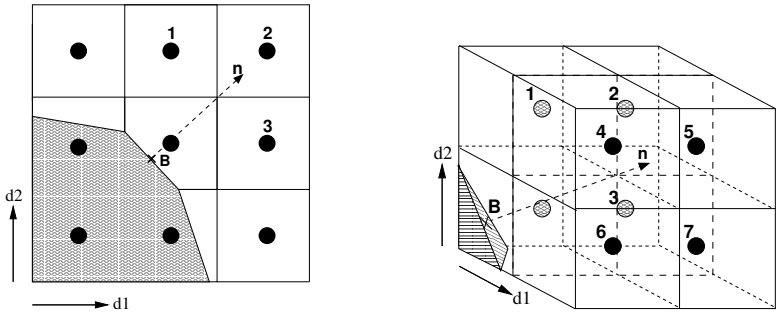


Figure 3. Least squares stencil to obtain flux for the Dirichlet boundary condition on the embedded boundary in 2D (left) and 3D (right) with radius 1.

The stencil of points ($m = 1, 2, \ldots, p$), which excludes the cut cell that contains the embedded boundary, is determined by the direction of the normal at the boundary. In 2D, the normal points to a quadrant that includes up, side and corner cells with $p = 3$, resulting in two equations and three unknowns in the least squares system. In 3D, the normal points to an octant with $p = 7$, resulting in three equations and seven unknowns. The stencils are shown in Figure 3.

In the case of very complex geometries such as those experienced in porous media flows where boundaries are very close together and can exhibit cusps and semidisconnected cavities, this least squares stencil approach based on direction of the normal can be relaxed to use any points available in a monotone path from the root cell with a radius greater than 1 but with the same restrictions on p. For adaptive calculations, we use higher-order (quadratic) interpolation to fill ghost cells for second-order elliptic operators (Laplacian) at coarse-fine boundaries in order to avoid $O(1)$ truncation error [30].

We also note that geometric multigrid coarsening can be challenging in very complex geometries. As an example, for the pressure-Poisson equations (2-3) and (2-10), the presence of a semidisconnected cavity in the domain can result in a Neumann problem with nonzero null space. We therefore rely on a combined embedded boundary-algebraic multigrid (EB-AMG) approach to solve elliptic equations in very complex geometry cases [48].

3.2. ***Advective derivative.*** Since the flow is incompressible, we make use of the conservative form of the advection term, $\nabla \cdot (\vec{u}\vec{u})$, in (3-1) with $\vec{F} = \vec{u}\vec{u}$. The problem with this discretization for advection is that the CFL stability constraint on the timestep is at best $\Delta t = O((h/v_i^{\max})(\kappa_i)^{1/D})$, where $v_i^{\max}$ is the magnitude of the maximum wave speed for the i-th control volume. This is the well-known small-cell problem for embedded boundary, cut cell methods. There have been a number of proposals to deal with this problem, including merging the small control volumes with nearby larger ones [44; 14; 43; 25], the development of specialized stencils

that guarantee the required cancellations [9; 8; 20; 23] or simply using a threshold volume below which the cell is considered completely covered, i.e., $\kappa = 0$, as in [17].

Our previous approach in [53] was to expand the range of influence of the small control volumes algebraically to obtain a stable method, akin to [11; 6; 41]. We used a linear hybridization of conservative and nonconservative estimates of $\nabla \cdot (\vec{u}\vec{u})$:

$$\nabla \cdot (\vec{u}\vec{u})_{\boldsymbol{i}}^{n+1/2} = \kappa_{\boldsymbol{i}} (\nabla \cdot (\vec{u}\vec{u}))_{\boldsymbol{i}}^{\mathrm{C}} + (1 - \kappa_{\boldsymbol{i}})(\nabla \cdot (\vec{u}\vec{u}))_{\boldsymbol{i}}^{\mathrm{NC}}.$$

The small denominator in $\nabla \cdot (\vec{u}\vec{u})$ is canceled, and a stable method is obtained. However, the method fails to conserve mass by an amount measured by the difference between the hybrid discretization and the conservative one:

$$\delta M_{\boldsymbol{i}} = \kappa_{\boldsymbol{i}} \left((\nabla \cdot (\vec{u}\vec{u}))_{\boldsymbol{i}}^{\mathrm{C}} - (\nabla \cdot (\vec{u}\vec{u}))_{\boldsymbol{i}}^{\mathrm{NC}} \right) = \kappa_{\boldsymbol{i}} (1 - \kappa_{\boldsymbol{i}}) (\nabla \cdot (\vec{u}\vec{u}))_{\boldsymbol{i}}^{\mathrm{C}} - (\nabla \cdot (\vec{u}\vec{u}))_{\boldsymbol{i}}^{\mathrm{NC}}.$$

To maintain overall conservation, $\delta M_{\boldsymbol{i}}$ can be redistributed into nearby cells $\boldsymbol{i}'$:

$$\begin{aligned} \nabla \cdot (\vec{u}\vec{u})_{\boldsymbol{i}'}^{n+1/2} &:= \nabla \cdot (\vec{u}\vec{u})_{\boldsymbol{i}'}^{n+1/2} + w_{\boldsymbol{i},\boldsymbol{i}'} \delta M_{\boldsymbol{i}}, \quad \boldsymbol{i}' \in N(\boldsymbol{i}), \\ w_{\boldsymbol{i},\boldsymbol{i}'} &\geq 0, \qquad \sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} w_{\boldsymbol{i},\boldsymbol{i}'} \kappa_{\boldsymbol{i}'} = 1, \end{aligned} \tag{3-4}$$

where $N(\boldsymbol{i})$ is some set of indices in the neighborhood of $\boldsymbol{i}$ and including $\boldsymbol{i}$. The sum condition (3-4) makes the redistribution step conservative. The weights $w_{\boldsymbol{i},\boldsymbol{i}'}$ must be bounded, independent of $(\kappa_{\boldsymbol{i}'})^{-1}$. We use volume-weighted redistribution,

$$w_{\boldsymbol{i},\boldsymbol{i}'} = \left(\sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} \kappa_{\boldsymbol{i}'} \right)^{-1},$$

where $N(\boldsymbol{i})$ is a set of indices, including $\boldsymbol{i}$, within a radius of influence of 1 and connected by a monotone path.

The success of this approach depends on the calculation of $\nabla \cdot (\vec{u}\vec{u})^{\mathrm{NC}}$ because it is almost entirely responsible for the update of $\nabla \cdot (\vec{u}\vec{u})_{\boldsymbol{i}}$ in control volumes with $\kappa_{\boldsymbol{i}} \ll 1$. Specifically, $\nabla \cdot (\vec{u}\vec{u})^{\mathrm{NC}}$ must be designed so that the solution in small control volumes comes into equilibrium with the larger control volumes around it. We now enforce this point by summing the conservative approximation itself in a domain of influence around the cut cell and normalizing it by the sum of volume fractions in those cells to obtain the nonconservative approximation:

$$\nabla \cdot (\vec{u}\vec{u})_{\boldsymbol{i}}^{\mathrm{NC}} = \frac{\sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} \left(\kappa_{\boldsymbol{i}'} \nabla \cdot (\vec{u}\vec{u})_{\boldsymbol{i}'}^{\mathrm{C}} \right)}{\sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} \kappa_{\boldsymbol{i}'}}. \tag{3-5}$$

To compute $\nabla \cdot (\vec{u}\vec{u})^{\mathrm{C}}$, fluxes $\vec{u}\vec{u}$ are interpolated to face centroids as in (3-2) or (3-3) and substituted into (3-1).

3.3. ***Volume-weighted scheme.*** In a new approach, we avoid the small-cell problem altogether by taking advantage of the structure of our finite volume elliptic solvers, which take the form $\kappa\mathcal{L} = \kappa\rho$, where $\mathcal{L}$ is the elliptic operator, ρ is the right-hand side and κ is the volume fraction of a cut cell. This volume-weighted form allows us to compute source terms in (2-1) that are also volume-weighted. We also introduce a conservative form of the cell-centered pressure gradient in the pressure-correction form of the projection. The overall algorithm is as follows:

(1) Initially a cell-centered velocity is obtained from the projection of the prescribed conditions, $U^0 = \boldsymbol{P}(U^{\text{init}})$, similar to a potential flow solution. The pressure gradient is constructed to balance the viscous stress from this initial velocity, $\nabla p^{-1/2} = \nu\Delta U^0$, to ensure a stable calculation (see Section 4.3 for details) and then made to be volume-weighted, $\kappa\nabla p^{-1/2}$.

(2) If the flow is inertial (say, Re > 0.1), then velocities are extrapolated from cell centers to cell edges as in Section 2.3 and only the conservative volume-weighted advection term is computed, $\kappa(\nabla\cdot(UU)^{n+1/2})$. If the Reynolds number is low (say, Re < 0.1) or approaches the Stokes limit such that $\vec{u}\cdot\nabla\vec{u} \ll \nu\Delta\vec{u}$, then this step is unnecessary.

(3) The implicit Helmholtz equation (2-1) is solved in the form of our finite volume elliptic equation $\kappa\mathcal{L} = \kappa\rho$, where the right-hand side has volume-weighted terms including the source term:

$$\kappa\Big(I - \frac{\nu\Delta t}{2}\Delta\Big)U^{n+1,*} = \kappa\Big(U^n + \Delta t\Big(\frac{\nu\Delta t}{2}\Delta U^n - (U\cdot\nabla)U^{n+1/2} - \nabla p^{n-1/2}\Big)\Big). \quad (3\text{-}6)$$

(4) For the approximate projection, a volume-weighted Poisson equation

$$\kappa\Delta\delta = \kappa\boldsymbol{D}(U^{n+1,*}) \quad (3\text{-}7)$$

is solved, where $\delta = p^{n+1/2} - p^{n-1/2}$ indicates pressure correction form.

(5) The volume-weighted cell-centered gradient of the pressure correction, δ, is computed using a corollary to the divergence theorem for gradients:

$$\kappa\boldsymbol{G}(\delta) = V\iiint\frac{\partial\delta}{\partial x_i}\,dV = V\iint\delta(\hat{e}_i\cdot\hat{n}_i)\,dA. \quad (3\text{-}8)$$

The value of the pressure correction at the cell center can be used at the boundary centroid, or a more elaborate least squares system can be solved for the boundary value.

(6) The volume-weighted intermediate velocity is corrected with the volume-weighted gradient of the pressure correction

$$\kappa U^{n+1} = \kappa U^{n+1,*} - \kappa\boldsymbol{G}(\delta). \quad (3\text{-}9)$$

(7) If the flow is inertial, then the volume-weighting is removed from velocity in a normalization procedure similar to (3-5) so that the velocity can be used in the advection step of the next timestep

$$U^{n+1} = \frac{\sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} (\kappa_{\boldsymbol{i}'} U_{\boldsymbol{i}'}^{n+1})}{\sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} \kappa_{\boldsymbol{i}'}}. \tag{3-10}$$

Here, we note that normalization of the volume-weighted velocity is allowed because the velocity has already been sufficiently smoothed in the solution to the viscous Helmholtz equation (3-6). For consistency with step (2), this final step (7) is not necessary for low Reynolds number or Stokes flow.

4. Results

4.1. ***Accuracy.*** To demonstrate the accuracy of the algorithm, we consider incompressible flow inside a sphere. The fluid is initialized as a Gaussian vortex

$$\omega(r) = e^{-20(4r-0.5)^2},$$

where r is measured from the center of the sphere at $\boldsymbol{x}_0$ to a point $\boldsymbol{x}$ as $r^2 = (x-x_0)^2+(y-y_0)^2+(z-z_0)^2$ in 3D. The initial velocity can then be prescribed by

$$\begin{aligned} u(\boldsymbol{x}, t_0) &= \omega(r)((z-z_0)-(y-y_0))/r, \\ v(\boldsymbol{x}, t_0) &= \omega(r)((x-x_0)-(z-z_0))/r, \\ w(\boldsymbol{x}, t_0) &= \omega(r)((y-y_0)-(x-x_0))/r. \end{aligned}$$

In 2D, $r^2 = (x-x_0)^2 + (y-y_0)^2$ and

$$\begin{aligned} u(\boldsymbol{x}, t_0) &= -\omega(r)(y-y_0)/r, \\ v(\boldsymbol{x}, t_0) &= \quad \omega(r)(x-x_0)/r. \end{aligned}$$

The Reynolds number for this study is $\mathrm{Re} = 5$ based on vortex strength and diameter.

We estimate the error in the solution using the standard Richardson procedure, where computations of differing resolutions are evolved to the same time and compared in a certain norm (see [34] for details). Convergence rates for the 2D state variables are displayed in Tables 1, 2 and 3 for the L^1, L^2 and L^∞ norms,

Variable	$e_{8h\to 4h}$	Order	$e_{4h\to 2h}$	Order	$e_{2h\to h}$
u	1.781614×10^{-5}	1.981	4.513495×10^{-6}	1.990	1.135840×10^{-6}
v	1.788296×10^{-5}	1.981	4.529397×10^{-6}	1.991	1.139716×10^{-6}
$\nabla^x p$	1.245135×10^{-4}	1.998	3.117580×10^{-5}	1.999	7.798800×10^{-6}
$\nabla^y p$	1.245125×10^{-4}	1.998	3.117575×10^{-5}	1.999	7.798797×10^{-6}

Table 1. 2D solution error convergence rates using the L_1-norm for $h = \frac{1}{2048}$.

Variable	$e_{8h\to 4h}$	Order	$e_{4h\to 2h}$	Order	$e_{2h\to h}$
u	2.997859×10^{-5}	1.977	7.613610×10^{-6}	1.989	1.918581×10^{-6}
v	3.011462×10^{-5}	1.978	7.645604×10^{-6}	1.989	1.926329×10^{-6}
$\nabla^x p$	2.672975×10^{-4}	1.997	6.696069×10^{-5}	1.999	1.675127×10^{-5}
$\nabla^y p$	2.672958×10^{-4}	1.997	6.696059×10^{-5}	1.999	1.675127×10^{-5}

Table 2. 2D solution error convergence rates using the L_2-norm for $h=\frac{1}{2048}$.

Variable	$e_{8h\to 4h}$	Order	$e_{4h\to 2h}$	Order	$e_{2h\to h}$
u	1.234211×10^{-4}	1.980	3.129091×10^{-5}	1.990	7.878456×10^{-6}
v	1.237856×10^{-4}	1.980	3.138271×10^{-5}	1.990	7.899819×10^{-6}
$\nabla^x p$	1.396595×10^{-3}	1.995	3.504449×10^{-4}	1.998	8.771770×10^{-5}
$\nabla^y p$	1.396573×10^{-3}	1.995	3.504435×10^{-4}	1.998	8.771761×10^{-5}

Table 3. 2D solution error convergence rates using the L_∞-norm for $h=\frac{1}{2048}$.

respectively. We demonstrate second-order accuracy for all variables in all norms. A fixed step size of $\Delta t = 0.00025$, which is equivalent to a CFL number of $\sigma = 0.5$, is run for 512 steps at the finest resolution of $h=\frac{1}{2048}$. We note that the pressure gradient resulting from a single application of the projection is first-order (the scalar pressure is second-order) [16; 49]. To obtain a second-order pressure gradient, an additional approximate projection is required. The computational cost of this additional projection is minimized by initialization of the pressure to the value obtained from the first application of the projection.

We show convergence results in 3D in Tables 4, 5 and 6. At the finest resolution of $h=\frac{1}{256}$, a fixed step size of $\Delta t = 0.0005$, which is equivalent to a CFL number of $\sigma = 0.5$, is run for 64 steps. The convergence rates are second-order for all variables in all norms except that the velocity components are slightly less than second-order in the L^∞-norm. We attribute this slight degradation in accuracy to the solution not being fully resolved in the asymptotic regime for convergence.

4.2. ***Stability of the approximate projection.*** We demonstrate that the approximate projection operator is stable, i.e., $\|\boldsymbol{P}\| < 1$ (see [26]), by showing that the divergence of a velocity field diminishes with repeated application of the projection. The

Variable	$e_{8h\to 4h}$	Order	$e_{4h\to 2h}$	Order	$e_{2h\to h}$
u	1.071815×10^{-3}	1.865	2.941499×10^{-4}	1.948	7.623080×10^{-5}
v	1.069533×10^{-3}	1.864	2.937195×10^{-4}	1.948	7.613267×10^{-5}
w	1.068955×10^{-3}	1.865	2.933635×10^{-4}	1.948	7.603024×10^{-5}
$\nabla^x p$	1.593183×10^{-2}	1.806	4.556606×10^{-3}	1.933	1.192892×10^{-3}
$\nabla^y p$	1.592733×10^{-2}	1.805	4.556155×10^{-3}	1.933	1.192860×10^{-3}
$\nabla^z p$	1.593025×10^{-2}	1.806	4.556234×10^{-3}	1.933	1.192863×10^{-3}

Table 4. 3D solution error convergence rates using the L_1-norm for $h=\frac{1}{256}$.

Variable	$e_{8h\to4h}$	Order	$e_{4h\to2h}$	Order	$e_{2h\to h}$
u	2.294986×10^{-3}	1.846	6.382495×10^{-4}	1.900	1.709917×10^{-4}
v	2.288364×10^{-3}	1.847	6.361839×10^{-4}	1.900	1.704523×10^{-4}
w	2.287161×10^{-3}	1.847	6.357047×10^{-4}	1.900	1.703239×10^{-4}
$\nabla^x p$	4.846120×10^{-2}	1.748	1.442698×10^{-2}	1.922	3.808233×10^{-3}
$\nabla^y p$	4.845033×10^{-2}	1.748	1.442543×10^{-2}	1.921	3.808121×10^{-3}
$\nabla^z p$	4.843998×10^{-2}	1.748	1.442312×10^{-2}	1.921	3.807923×10^{-3}

Table 5. 3D solution error convergence rates using the L_2-norm for $h=\frac{1}{256}$.

Variable	$e_{8h\to4h}$	Order	$e_{4h\to2h}$	Order	$e_{2h\to h}$
u	1.891858×10^{-2}	1.841	5.280958×10^{-3}	1.781	1.536950×10^{-3}
v	1.903841×10^{-2}	1.849	5.285737×10^{-3}	1.803	1.514318×10^{-3}
w	1.911926×10^{-2}	1.846	5.319278×10^{-3}	1.793	1.534994×10^{-3}
$\nabla^x p$	4.893581×10^{-1}	1.565	1.653736×10^{-1}	1.868	4.529462×10^{-2}
$\nabla^y p$	4.880882×10^{-1}	1.562	1.652861×10^{-1}	1.868	4.528107×10^{-2}
$\nabla^z p$	4.886462×10^{-1}	1.564	1.653033×10^{-1}	1.868	4.528735×10^{-2}

Table 6. 3D solution error convergence rates using the L_∞-norm for $h=\frac{1}{256}$.

velocity field is initialized to a potential flow past an infinitely long cylinder with radius of 0.1. The cylinder is in the center of a unit square domain. We iteratively project the velocity field, U, and evaluate the norm of the divergence, $\kappa\boldsymbol{D}(U)$, and the norm of the pressure gradient, $\nabla\phi$, after each projection. Figures 4 and 5 show that all norms of both fields monotonically decrease with the number of projection iterations. Flattening of the curves near the end is due to the residual of the solution to the Poisson equation by multigrid iterations approaching machine accuracy.

4.3. ***Stability in the Stokes limit.*** We use the algorithm to compute a range of unsteady flows, including low Reynolds number flows where a steady state may

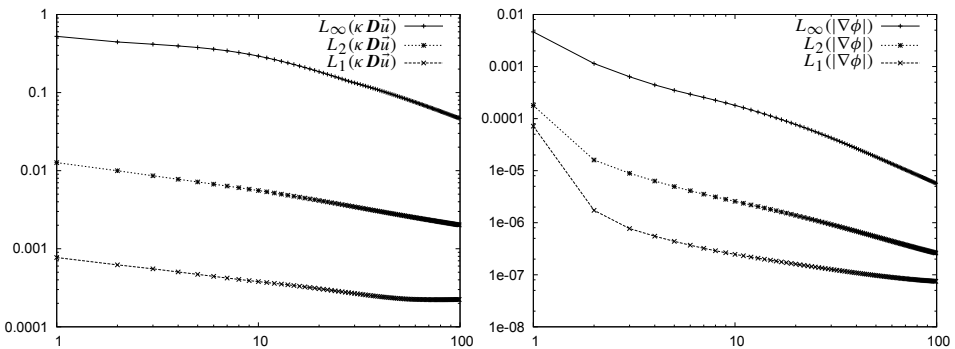


Figure 4. Norms (L_∞, L_2 and L_1) of $\kappa\boldsymbol{D}\vec{u}$ (left) and $\nabla\phi$ (right) versus the number of projection iterations in the 2D test with $h=\frac{1}{256}$.

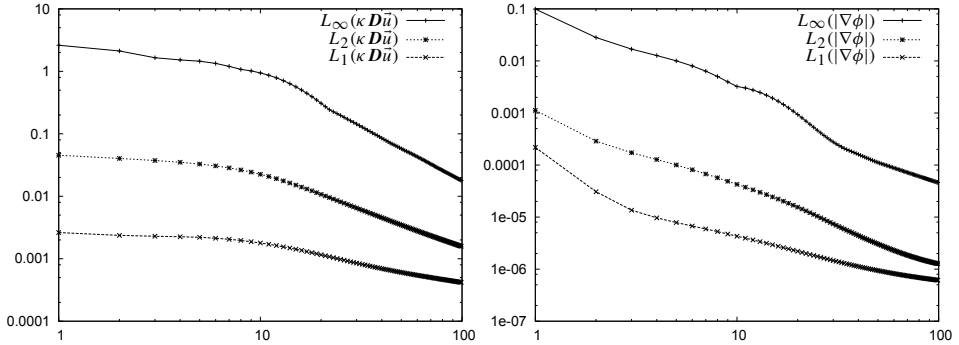


Figure 5. Norms (L_∞, L_2 and L_1) of $\kappa \boldsymbol{D}\vec{u}$ (left) and $\nabla\phi$ (right) versus the number of projection iterations in the 3D test with $h = \frac{1}{64}$.

exist. In this flow regime, parameters can approach the Stokes limit, $\mathrm{Re} \to 0$, where $\mathrm{Re} = U_c l_c/\nu$ and U_c and l_c are characteristic quantities, leading to the Stokes equations (Navier–Stokes less the advective derivative). However, the Stokes equations do not capture all the physics of flows even at $\mathrm{Re} = 0.1$, a value that has traditionally been considered well inside the Stokes limit. We demonstrate this point by considering flow near a sharp corner as in [36]. Figure 6 shows a comparison between solving the unsteady Stokes and Navier–Stokes equations to a steady state in an abrupt expansion channel, both for the same initial data at $\mathrm{Re} = 0.1$. The magnitude of the velocity and extent of the recirculation zone are noticeably greater when $\vec{u} \cdot \nabla\vec{u}$ is included in the calculation as seen in the difference in the location of the innermost contour. Comparison of plots of the velocity along a line through the recirculation zone shows a 5% difference. The difference between the solutions is more dramatic as the Reynolds number increases. The criterion for a steady-state solution is $U^{n+1} - U^n < \epsilon \Delta t$, where $\epsilon = 10^{-8}$.

To perform stable computations for low Reynolds number flows ($\mathrm{Re} < 0.1$), we must construct a well-posed IBVP such that both the momentum and continuity equations are satisfied by the initial conditions. We obtain a divergence-free potential flow field that satisfies the no-flow normal boundary conditions by projecting the velocity: $U^0 = \boldsymbol{P}(U^{\mathrm{init}})$, where $U^{\mathrm{init}} = 0$ inside the domain. The pressure gradient is calculated to balance the viscous stress due to the flow field: $\nabla p^{-1/2} = \nu \Delta U^0$. In this regime, since viscous effects can dominate inertial forces ($\nu > U_c l_c$), we define the viscous timestep to be $\Delta t_\nu = h^2/\nu$. The stability constraint for the algorithm in the low Reynolds number regime is $\Delta t = \min(\Delta t_{\mathrm{CFL}}, \Delta t_\nu)$.

4.4. ***Scalability and performance.*** The algorithm described here has been implemented in the Chombo software framework. Chombo provides a set of tools for implementing finite difference and finite volume methods for the solution of PDEs on block-structured adaptively refined rectangular grids. Chombo also supports

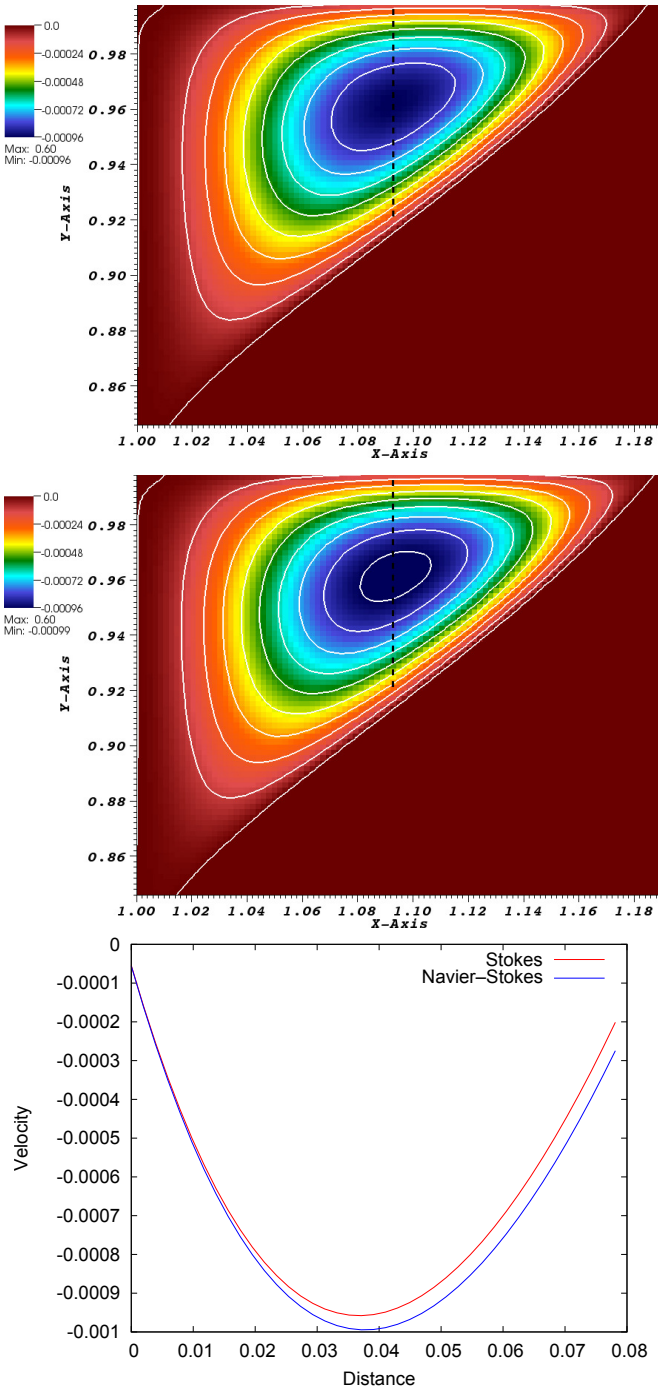


Figure 6. Zoomed-in view of the recirculation zone in the corner behind the backward-facing step for Re = 0.1. Top: solution to the Stokes equation. Middle: solution to the Navier–Stokes equation. Bottom: plots of velocity along the dotted lines in the top figures.

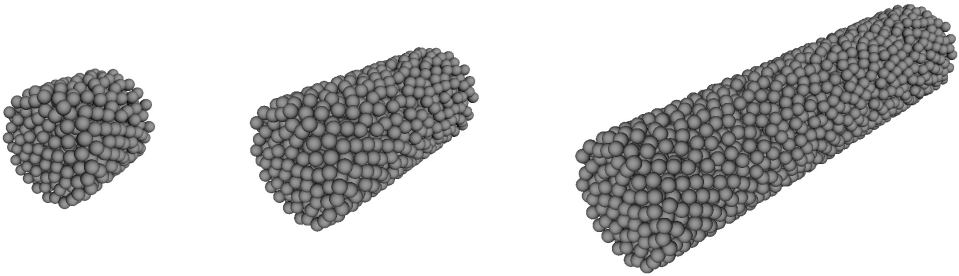


Figure 7. Packed cylinder geometries for weak scaling (replication) test in Figure 8. Left: 1 cm-long cylinder packed with 750 spheres ($2048 \times 2048 \times 2048$) with resolution $h \approx 2.44\,\mu$m. This geometry is used for $N = 512, 4096, 32768, 261144$. Middle: 2 cm-long cylinder packed with 1500 spheres ($2048 \times 1024 \times 1024$) with resolution $h \approx 4.88\,\mu$m. This geometry is used for $N = 1024, 8192, 65536$. Right: 4 cm-long cylinder packed with 3000 spheres ($4096 \times 1024 \times 1024$) with resolution $h \approx 4.88\,\mu$m. This geometry is used for $N = 2048, 16384, 131072$. The spheres have radii $250\,\mu$m.

computations in complex geometries with embedded boundaries. Chombo software libraries enable high-performance computing, data management and I/O for large-scale simulations.

We demonstrate the scalability and performance of our Chombo-based algorithm using a weak scaling test for flow through a cylinder packed with spheres (see Figure 7) as in [48]. In weak scaling, the problem domain is refined by the same factor as the increase in the number of processor cores (e.g., factor of 2 refinement in each spatial dimension, D, requires 2^D times the number of cores). These tests are conducted on the NERSC Cray XC30 system, Edison, for up to 131,072 cores and

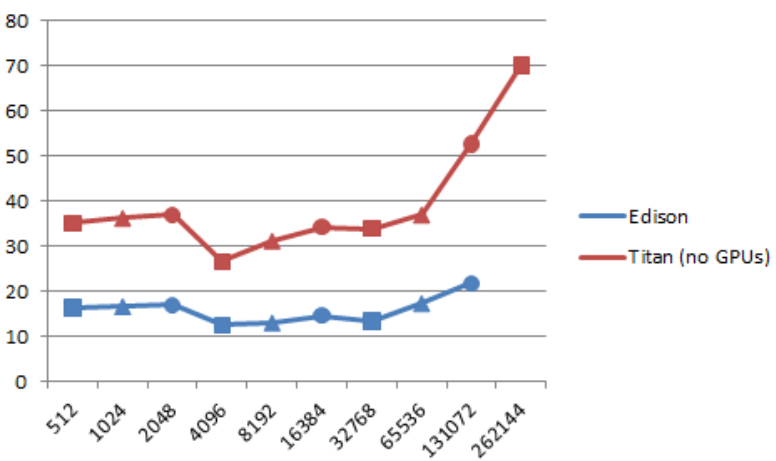


Figure 8. Weak scaling on the NERSC Cray XC30 (Edison) and OLCF Cray XK7 (Titan, no GPUs). The horizontal axis is concurrency, N, and the vertical axis is average time in seconds per timestep.

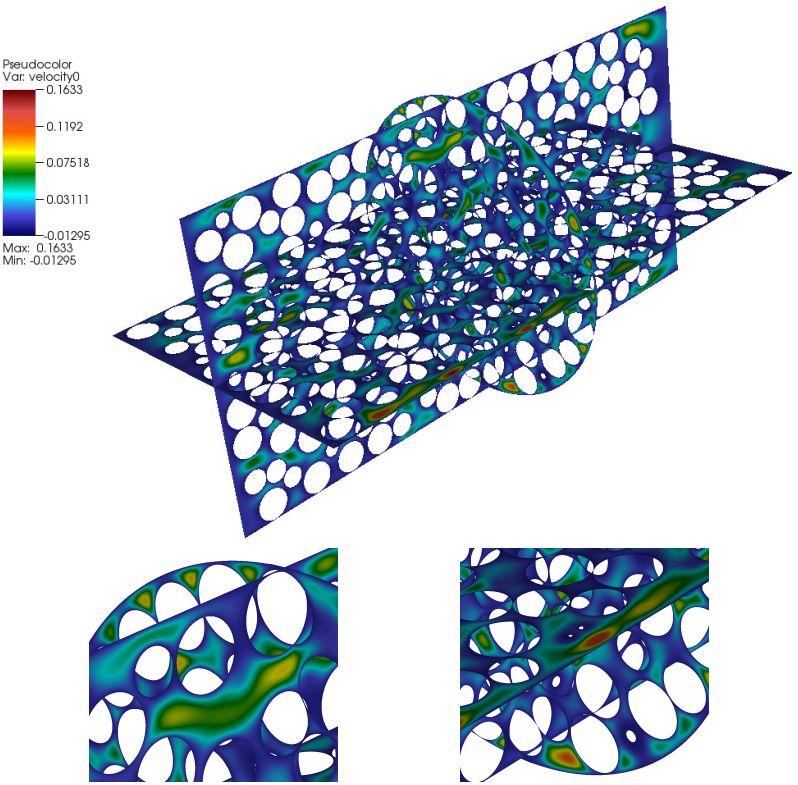


Figure 9. Steady-state flow through the packed cylinder geometry in Figure 7 (middle). Axial velocity is shown with magnified views of the top and bottom of the cross-sectional slice. Inlet axial velocity is 0.01 cm/sec with $\mathrm{Re} = 0.5$. Computation was performed on NERSC Cray XC30 Edison using 65,536 processor cores.

on the OLCF Cray XK7 system, Titan, for up to 262,144 CPU cores. We perform 10 timesteps of the algorithm and take the average time per timestep in seconds. We use a sweet spot for domain decomposition and load balancing of one box per processor core, where one box is 32^3 cells. Since a large number of spheres have to be randomly placed in a cylinder, it is difficult to guarantee a fixed number of spheres per box. However, the scaling is theoretically very close to replicated data as in [53]. Therefore, we take three different aspect ratios of the cylinder — where each aspect ratio is a weak scaling test in itself — and combine into one continuous scaling curve in Figure 8. The three sets of weak scaling data are depicted by shape: a 1-to-1 cylinder packed with 750 spheres run on 512, 4096, 32,768 and 262,144 cores (squares), a 2-to-1 cylinder with 1500 spheres run on 1024, 8192 and 65,536 cores (triangles) and a 4-to-1 cylinder with 3000 spheres run on 2048, 16,384 and 131,072 cores (circles).

On Edison, we observe excellent performance with about 83% efficiency from 512 to 131,072 cores — a relatively flat weak scaling curve — and an average time

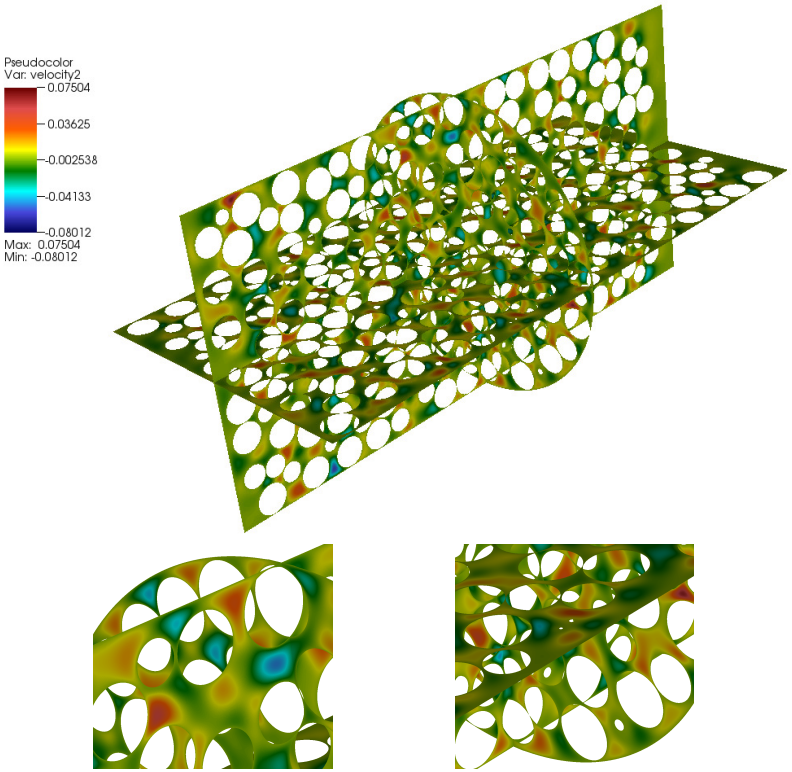


Figure 10. Steady-state flow through the packed cylinder geometry in Figure 7 (middle). Transverse (z) velocity shown with magnified views of the top and bottom of the cross-sectional slice. Inlet axial velocity is 0.01 cm/sec with Re = 0.5. Computation was performed on NERSC Cray XC30 Edison using 65,536 processor cores.

per timestep of 20 seconds at the highest concurrency. On Titan, we observe about 67% efficiency from 512 to 131,072 cores and only 50% up to 262,144 cores. We note a slight dip at $N = 4096$, and even slighter at $N = 32768$, in both curves that is likely due to a lower percentage of cut cells from refinement of the geometry. We do observe an upward trend in the weak scaling curve on Titan, particularly at the two highest concurrencies (it should be flat throughout), but overall the time only slightly doubles from the lowest concurrency to the highest. We consider the result on Titan to be good performance for the vast range of concurrencies and given the flow physics and geometry. Furthermore, performance is in the neighborhood of 1 timestep per minute at the highest concurrency on Titan, which is an acceptable metric for a large-scale fluid dynamics calculation. We do not make use of the GPUs on Titan for this scaling test, which may contribute to degraded performance.

4.5. ***Simulation results.*** We present simulation results for the 2D and 3D incompressible Navier–Stokes equations for a range of Reynolds numbers in various

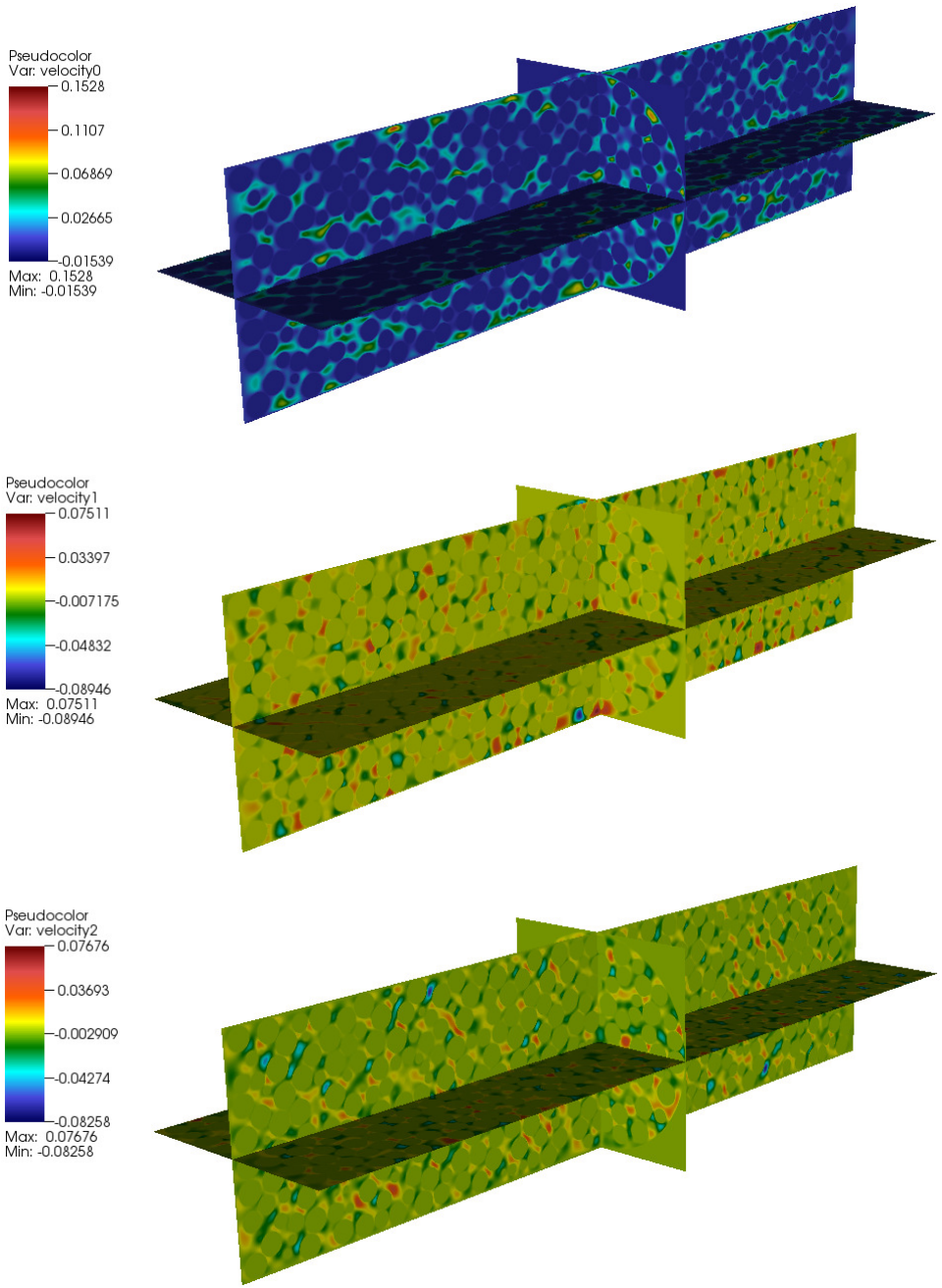


Figure 11. Steady-state flow through the packed cylinder geometry in Figure 7 (right). From top to bottom, we show the axial (x) and transverse (y) and (z) velocities. Inlet axial velocity is 0.01 cm/sec with $\mathrm{Re} = 0.5$. Spheres have not been voided in the visualization unlike in Figures 9 and 10. Computation was performed on OLCF Titan using 131,072 processor cores.

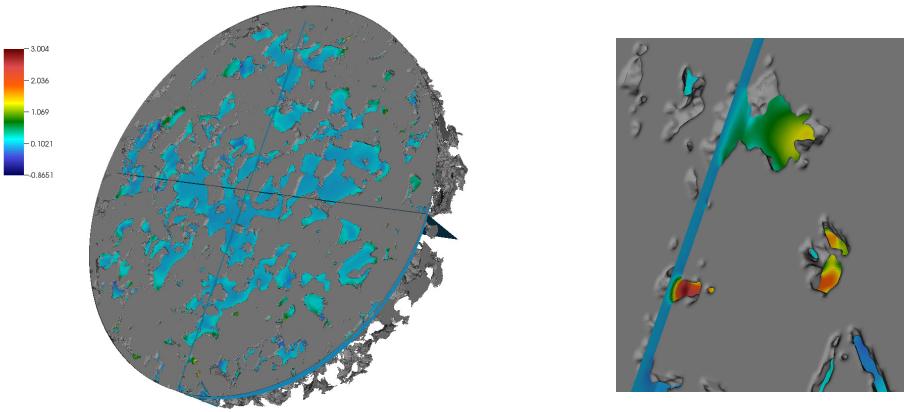


Figure 12. Steady-state flow through Bedford limestone: embedded boundary grid in gray with z-velocity data (left) and magnification of the bottom of the disk (right). Resolution is $h = 43$ nm with $2048 \times 2048 \times 320$ total grid cells and 29% porosity. Inlet velocity is 0.0376 cm/sec. Computation was performed on OLCF Titan using 40,960 processor cores.

geometries. Flow problems are set up such that the flow is typically from left to right in the x direction, the kinematic viscosity is that of water, $\nu = 0.01\,\text{cm}^2/\text{sec}$, and the average velocity at inflow is 1 cm/sec (Poiseuille in 2D and constant in 3D) unless otherwise stated. For flows where inertial forces have an effect (typically $\text{Re} > 0.1$), the CFL number is $\sigma = 0.9$. All units are specified in the CGS system. The maximum grid size resulting from domain decomposition and the AMR hierarchy is 256^2 cells per grid block (box) in 2D and 32^3 cells per grid block in 3D. The criterion for steady-state flow is $U^{n+1} - U^n < \epsilon \Delta t$, where $\epsilon = 10^{-8}$.

4.5.1. *Low Reynolds number flow.* We demonstrate the algorithm at the low end of the Reynolds number flow regime by showing steady-state results for the packed cylinder used in the scaling study. Using 65,536 processor cores on the NERSC Cray XC30 Edison, we simulate steady-state flow in the 2-to-1 cylinder packed with 1500 spheres as in Figure 7 (middle). In Figures 9 and 10, we show the axial (x) and transverse (z) velocities with magnified views to convey the tortuosity of the flow and the resolved viscous boundary layer in the pore space. The grid resolution for this simulation is $2048 \times 1024 \times 1024$ cells.

To demonstrate hero run capability, we also simulated steady-state flow in the 4-to-1 cylinder geometry packed with 3000 spheres in Figure 7 (right). The steady-state velocity is shown in Figure 11. This simulation made use of 131,072 processor cores on OLCF Titan. The Reynolds number is 0.5 for both simulations.

4.5.2. *Direct numerical simulation from image data.* In addition to synthetic geometries as in the packed cylinder, we demonstrate direct simulation from microtomography image data. Figure 12 shows flow in Bedford limestone with porosity

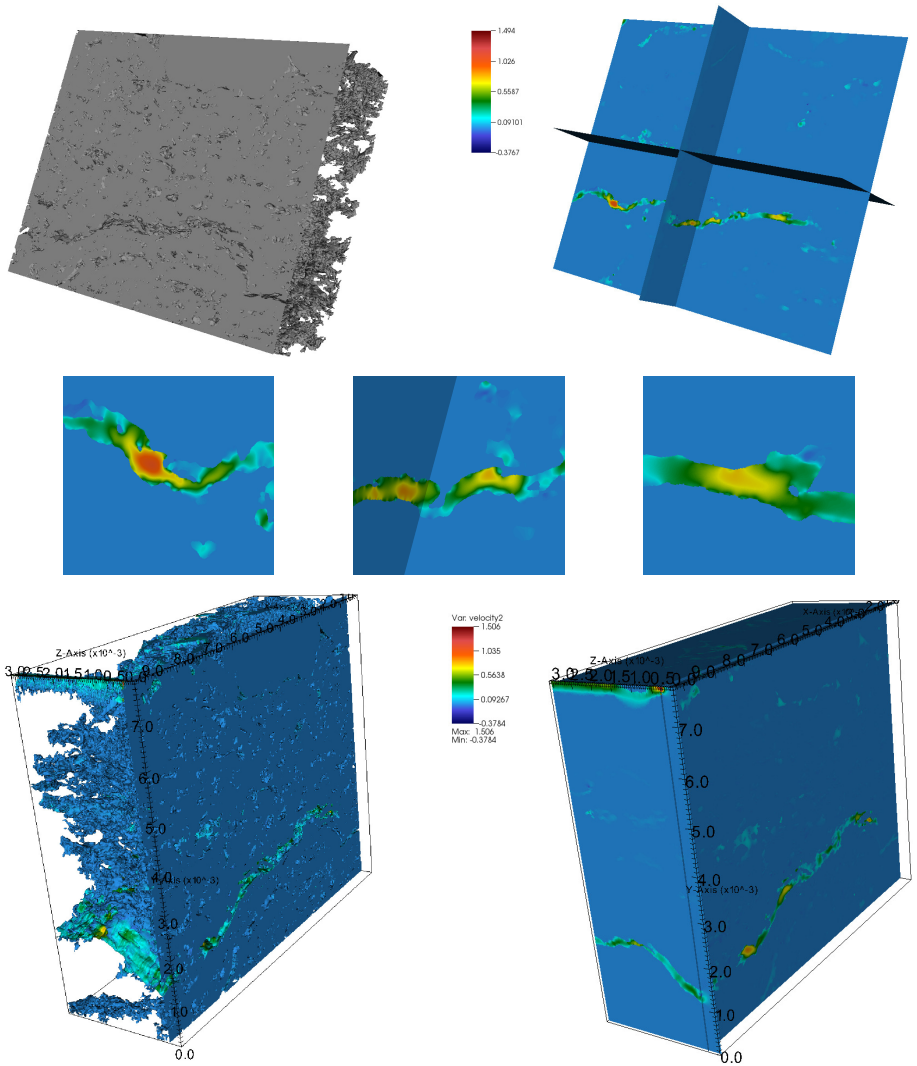


Figure 13. Steady-state flow through fractured shale: embedded boundary grid in gray (top left) with slices at mid-planes of z-velocity (top right), magnifications (middle) and rotated view of boundary data (bottom left) with slice planes near maximum velocity location (bottom right). Resolution is $h = 48.4$ nm with $1920 \times 1600 \times 640$ total grid cells and 18% porosity. Inlet velocity is 0.008 cm/sec. Computation was performed on NERSC Hopper using 60,000 processor cores.

of 29%. In this simulation, resolution of the cross section is critical in order to capture viscous effects in very tight pore space. The grid resolution of this simulation is $2048 \times 2048 \times 320$ cells or $h = 43$ nm. The image voxel size is 4.4 μm.

We also demonstrate the ability to model fully resolved steady-state flow in a fractured shale in Figure 13. The geometry in this case is obtained from focused

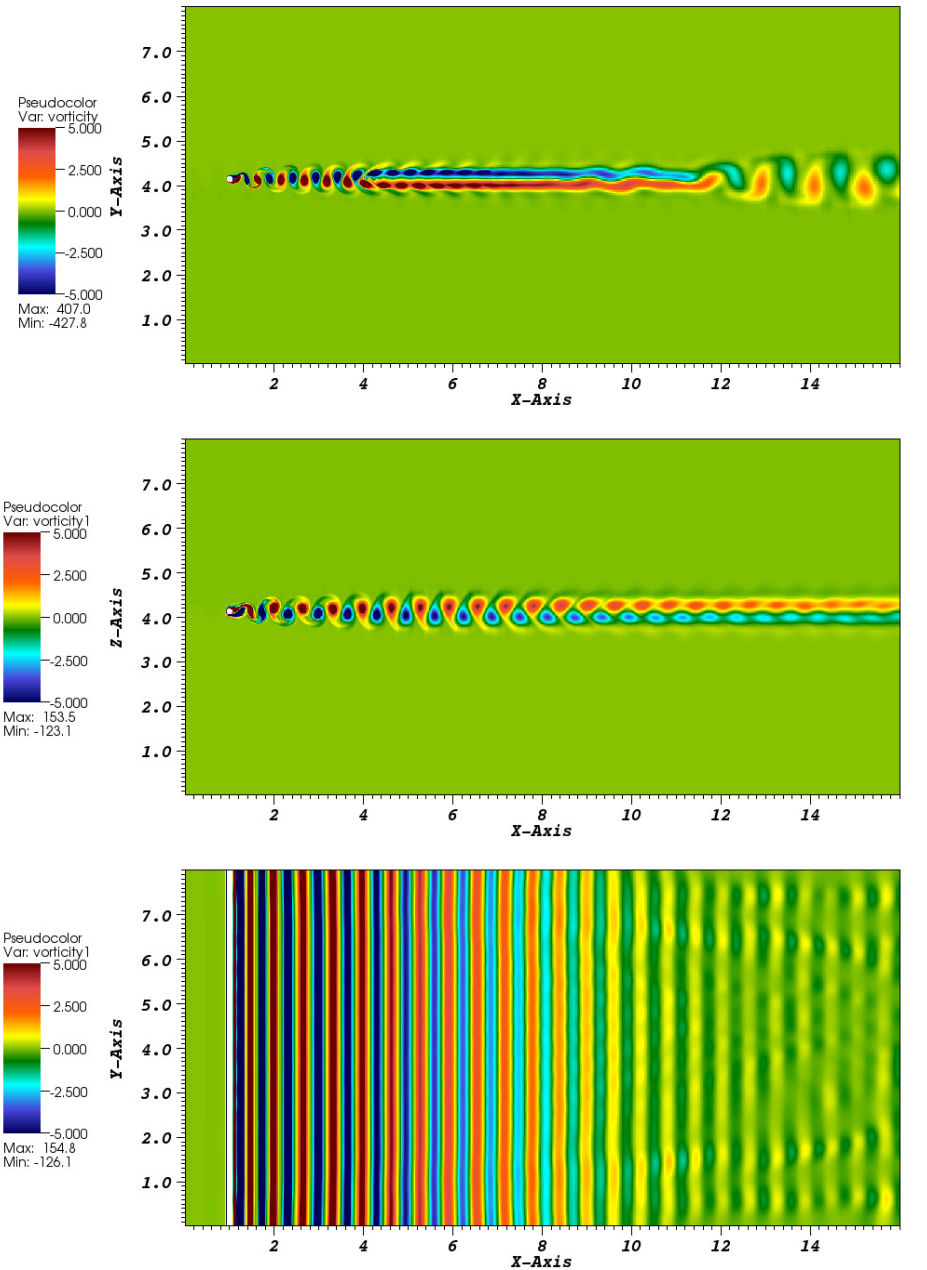


Figure 14. Flow past a cylinder at Re = 300. Top: 2D simulation of vorticity. Middle: 3D simulation of y vorticity in the x-z plane at $z = 4.125$. Bottom: 3D simulation of y vorticity in the x-y plane at $y = 4$. The cylinder has radius $r = 0.0625$ and is shown in white centered at $x = 1$, $y = 4.125$ in 2D and $x = 1$, $y = 4$, $z = 4.125$ in 3D. Time simulated is 98 seconds.

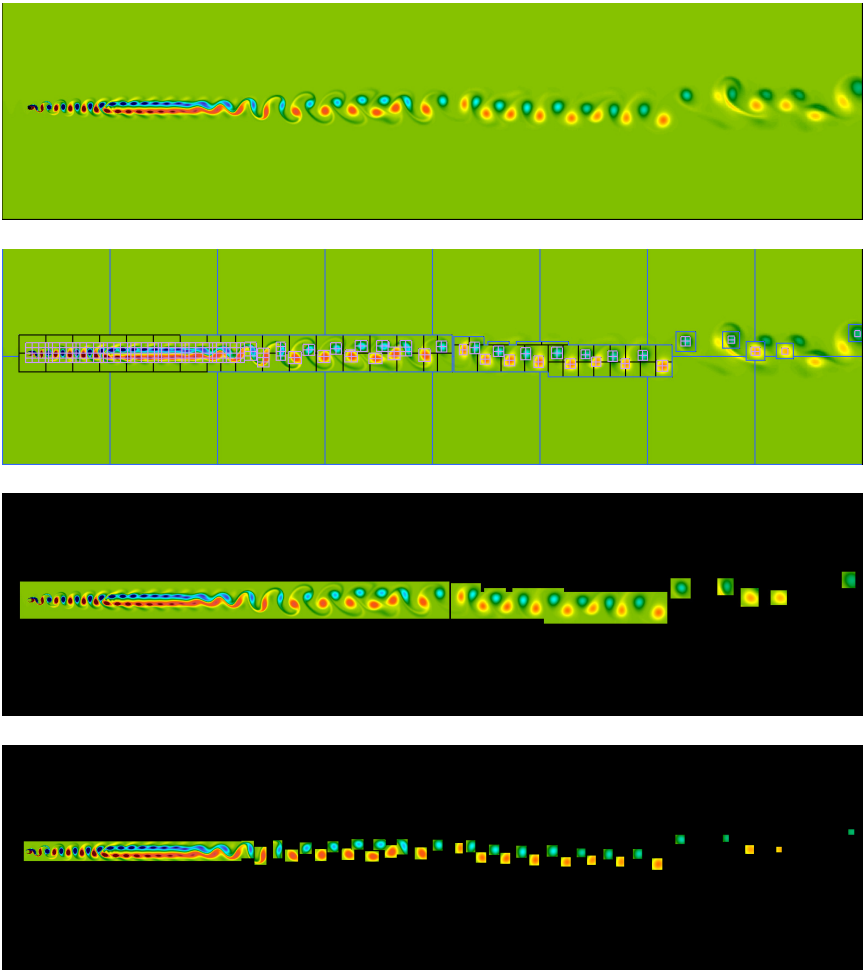


Figure 15. 2D flow past a cylinder at Re = 300 with extended wake ($l = 32$). From top to bottom: vorticity, AMR hierarchy of boxes enclosing the wake and two finest levels. The cylinder has radius $r = 0.0625$ and is shown as a black spot near the inlet. Wake extends approximately 250 cylinder diameters. Time simulated is 112 seconds.

ion beam scanning electron microscopy (FIB-SEM) image data. The pore space is tighter than the limestone, even though a fracture aperture is present, with a porosity of 18%. The grid resolution is $1920 \times 1600 \times 640$ or $h = 48$ nm. The image voxel size is 50 nm. In both the synthetic packed cylinder and the image data simulations, we make use of the EB-AMG method in [48] to solve elliptic problems in these very complex geometries.

4.5.3. *Flow past a cylinder* (Re = 300). We perform direct numerical simulation (DNS) of flow past a cylinder in both 2D and 3D. The diameter of the cylinder centered at $x = 1$ is $d = 0.125$ in a domain that has dimensions $l = 16$ and $w = 8$.

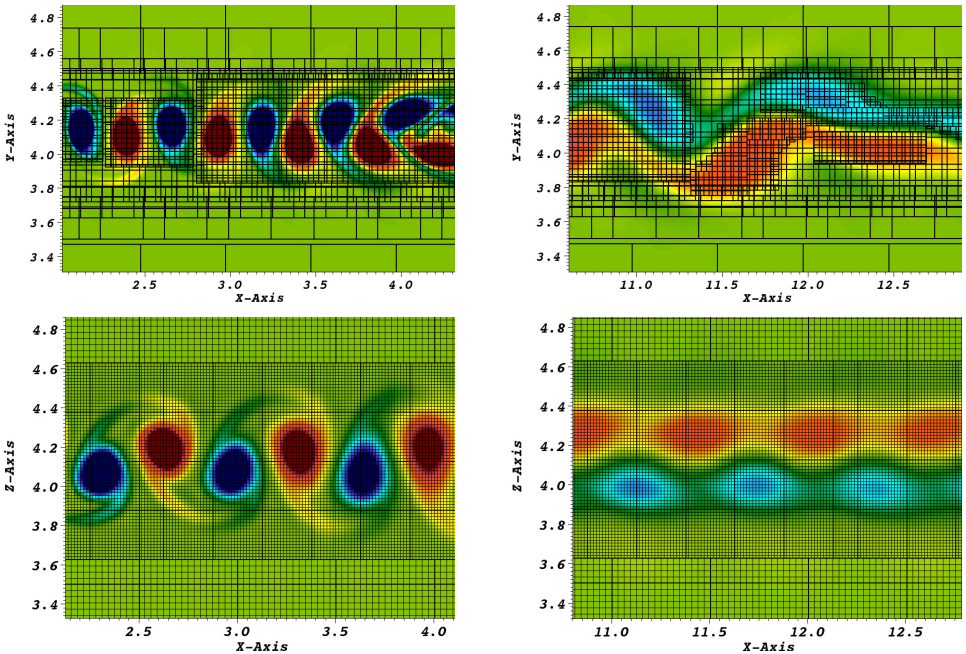


Figure 16. Zoomed-in view of Figure 14 showing AMR hierarchy. Top: 2D vorticity at left and right sections in the wake with grid block outlines for 4 levels of refinement, factor of 2. Each block has a maximum of 16^2 cells (cells not shown). The finest level ($h = \frac{1}{512}$) is gridded on 5% of the total domain. Bottom: 3D y vorticity in the x-z plane at $y = 4$ at left and right sections in the wake with grid block outlines and cells for 2 levels of refinement, factor of 2. Each block contains 16^3 cells in 3D (cells shown for the x-z plane). The finest level ($h = \frac{1}{64}$) is gridded on 12% of the total domain.

With $U_c = 1$, $l_c = d = 0.125$ and $\nu = 0.0004167$, the Reynolds number for this simulation is $\mathrm{Re} = 300$. In Figure 14 (top), we performed a highly resolved four-level AMR calculation in 2D at $\mathrm{Re} = 300$ where the finest level covers only 5% of the total domain at $t = 98$. The length of wake in this calculation is a very long $120d$, which is shown to be necessary to capture the halfway downstream secondary structures and far downstream tertiary structures. (We have also simulated an extended domain with twice the length ($l = 32$) shown in Figure 15 that depicts additional wake structures but with no comparison to 3D.) A recirculation zone persists within the secondary wake structure along the centerline between the vortices above and below. We simulate a domain width of $64d$ and use slip wall boundary conditions at boundaries transverse to the x direction of the flow to minimize the interaction of domain boundary effects with the wake. The additional mesh refinement tracks dynamically with the magnitude of vorticity in time using a refinement threshold of 2.0. Grid blocks are outlined (cells not shown) for this five-level calculation in Figure 16 (top). Each grid block contains a maximum of 16^2 cells for a given level of refinement.

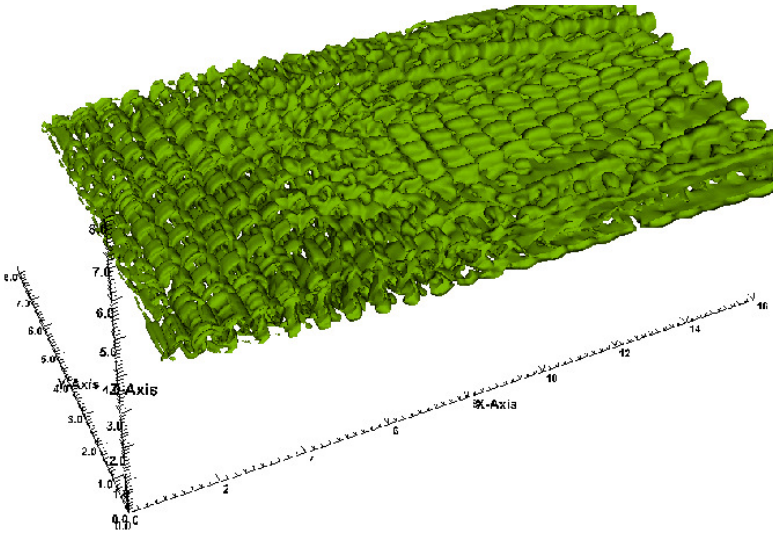


Figure 17. 3D isocontour ($\omega_z = 0.001$) of z vorticity for flow past a cylinder at Re = 300 and $t = 98$ seconds. The isocontour is contained by the finest level of grid blocks.

We performed a simulation in 3D for comparison to 2D using the same parameters but with only two additional AMR levels. Williamson notes a transition Reynolds number regime up to 300, beyond which velocity fluctuations become irregular and vortex formation is three-dimensional [57; 58; 59]. In the 3D simulation, a number of transient structures develop in the fluid that organize at very long time into a persistent train of vortices (see Figure 14, middle and bottom), which is only similar to the near wake in 2D (see Figure 14, top). The third dimension has a self-organizing effect on the wake structures as previously noted [22; 61]. The flow in the wake is clearly not two-dimensional as seen in the bowing of the peaks and valleys and narrowing of the length of the rows. This point is further emphasized in the isocontour plot of z vorticity in Figure 17. We also show the grid blocks with cells for this two-level calculation in Figure 16 (bottom). The finest level is gridded on 12% of the domain at $t = 98$. Each grid block contains a maximum of 16^3 cells for a given level of refinement.

4.5.4. *Flow past a sphere* (Re = 600). We perform DNS of 3D flow past a sphere. The diameter of the sphere, centered at $x = 1$, is $d = 0.125$ in a domain that has dimensions $l = 16$ and $w = 8$. With $U_c = 2$, $l_c = d = 0.125$ and $\nu = 0.0004167$, the Reynolds number for this simulation is Re = 600. In Figure 18, we show a resolved three-level AMR calculation where the finest level covers less than 1% of the total domain. (The domain is very large in order to minimize boundary interactions with the wake.) The base grid is $512 \times 256 \times 256$ with two additional levels of refinement, factor of 4. The maximum grid box size in the AMR hierarchy is 32^3.

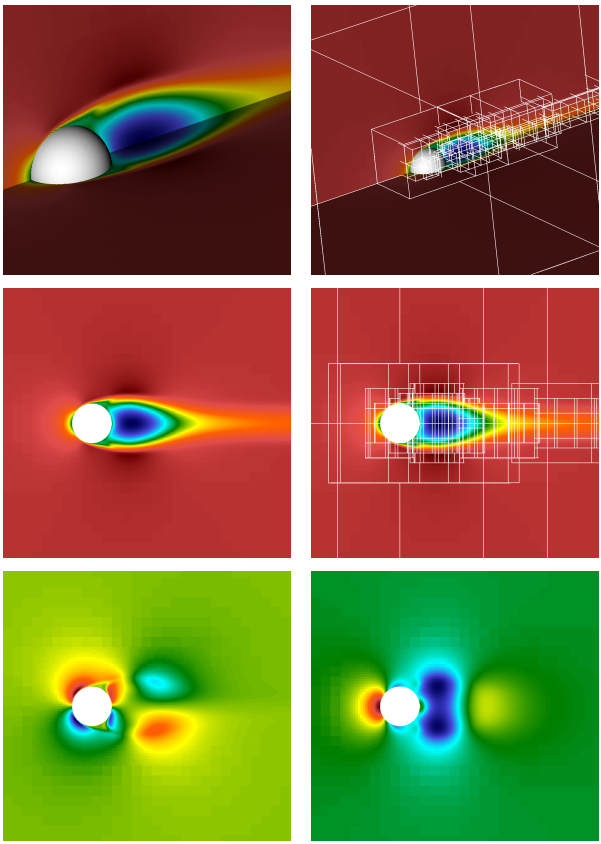


Figure 18. Flow past a sphere, Re = 600. Top: velocity in the x direction shown in 3D with AMR grids. Middle: velocity in the x direction in the x-y plane with grids. Bottom: velocity in the y direction and pressure in the x-y plane. The base grid is $512 \times 256 \times 256$ with 2 additional levels of refinement, factor of 4. The domain is 16 cm long and 8 cm wide by 8 cm wide. Sphere diameter is 0.125 cm. Inlet velocity is 2 cm/sec. Simulated time is 2 seconds and early in wake formation with no vortex shedding. The simulation was performed on 8192 nodes, 1 rank per node on ALCF BGQ Mira.

The 3D plots at the top of Figure 18 depict a notched wake in the velocity field but no oscillations at $t = 2$ seconds. We show outlines of the grid boxes to indicate refinement only around the sphere and the wake and not away from the interesting part of the flow. The second row of plots shows the same in a 2D slice. At the bottom of the figure, we show a transverse component of velocity as well as the pressure. Figure 19 depicts a wake that has begun to oscillate at $t = 2.5$ seconds.

4.5.5. *High Reynolds number flow in a contraction* (Re = 6300). We show results for another example of the effectiveness of AMR when combined with the embedded boundary method by demonstrating DNS of flow in a sudden contraction. This example is intended to model flow of oil upward in a long (over 4 km) pipe buried

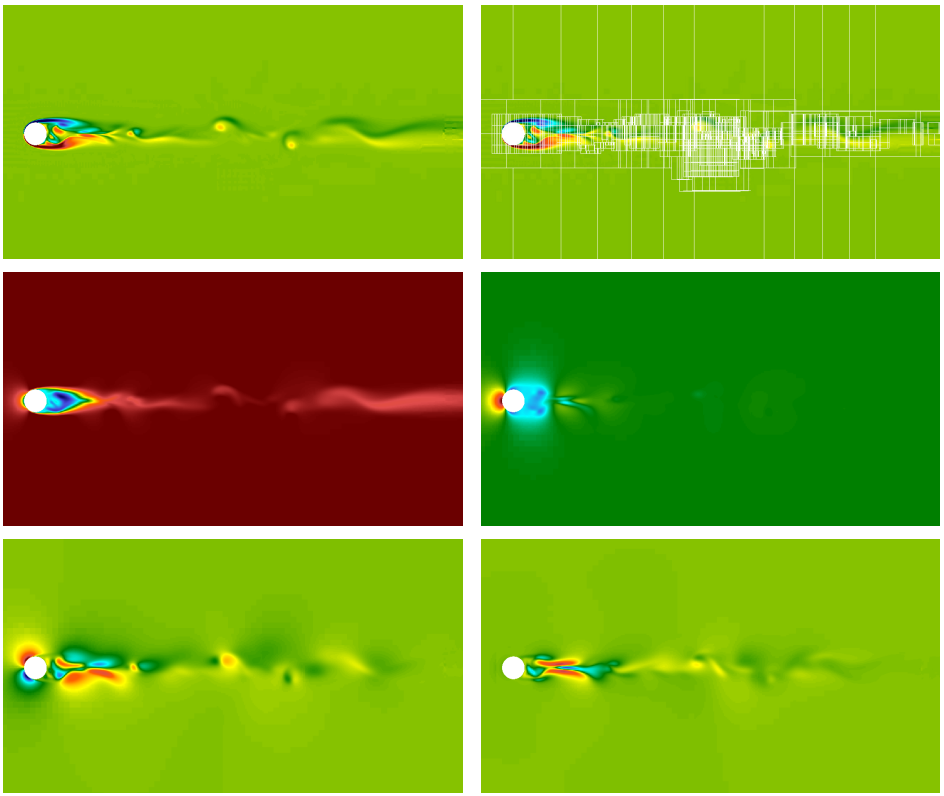


Figure 19. Flow past a sphere, Re = 600. Top: vorticity in the z direction and with AMR grids. Middle: velocity in the x direction and pressure. Bottom: velocity in the y direction and velocity in the z direction. The base grid is $512 \times 256 \times 256$ with 2 additional levels of refinement, factor of 4. The domain is 16 cm long and 8 cm by 8 cm in cross section. Sphere diameter is 0.125 cm. Inlet velocity is 2 cm/sec. Simulated time is 2.5 seconds and early in wake formation, but vortices have begun to shed. All plots are shown in x-y plane. The simulation was performed on 8192 nodes, 1 rank per node on ALCF BGQ Mira.

in the sea bed that undergoes essentially a contraction at the sea floor near a blowout preventer (as in the Deepwater Horizon Macondo well [33; 39]). In such a scenario, it is critical to solve for the bulk flow characteristics like pressure drop and flow rate over the entire length of the pipe in order to assess the likelihood of success for intervention strategies. However, it is equally critical to resolve the microscopic, by comparison, boundary layer effects near the blowout preventer in order to be able to determine failure points. Here, we focus on Newtonian flow in a 1 m-length section of the pipe near a 4-to-1 contraction in two dimensions of Cartesian coordinates for demonstration purposes only. The base grid contains 2048×1024 cells (0.488 mm resolution) with 32 boxes of 256^2 cells. Three additional levels of refinement (factor of 4) are added dynamically for an effective resolution of 7.6 μm

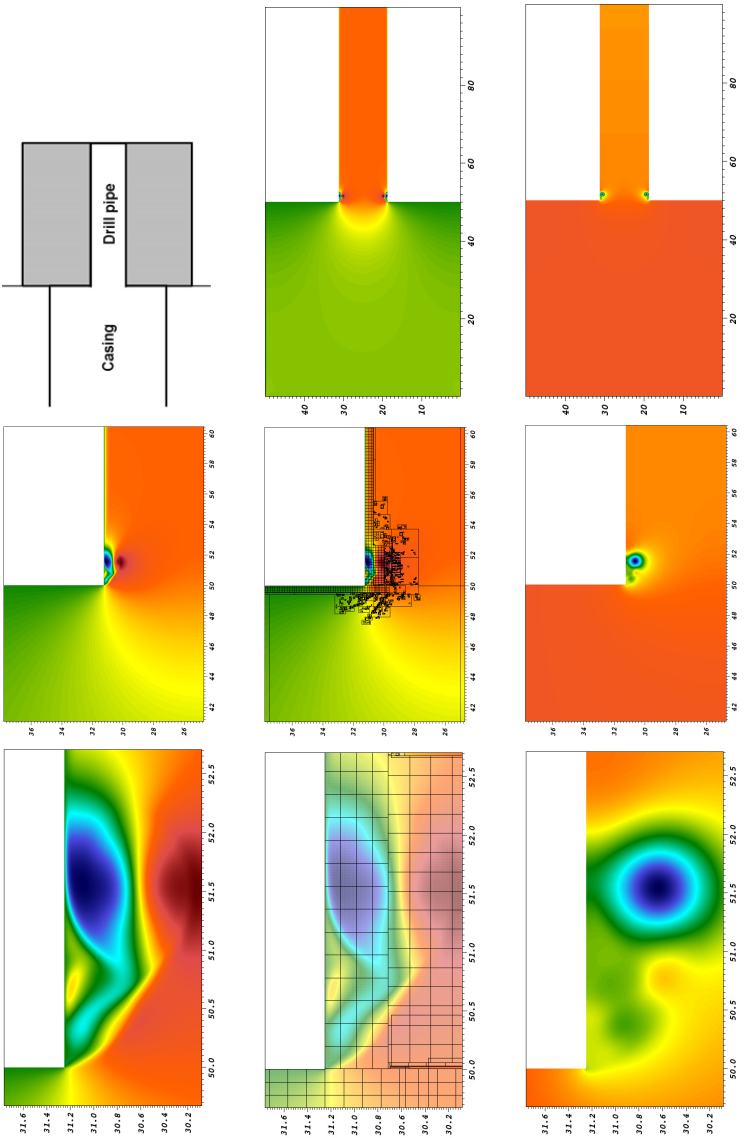


Figure 20. Turbulent flow in 2D contraction, Re $\approx$ 6300. Top left: conceptual model for oil flow entering a hypothetical failed blowout preventer depicted as a wellbore contracted into a (stuck) drill pipe. Top middle: axial fluid velocity in 100 cm-long by 50 cm-wide section of contraction. Top right: pressure. Middle left: increased magnification of velocity near the contraction corner with finer levels of AMR boxes (center). Middle right: increased magnification of pressure. Bottom left: increased magnification of velocity with finer levels of AMR boxes and mesh (bottom middle). Bottom right: increased magnification of pressure. Velocity range is -65.52 cm/sec (blue) to 84.66 cm/sec (red). Inlet average velocity is 10 cm/sec. Pressure range is -6662 bar (blue) to 3398 bar (red). The base grid is 2048×1024 with 3 additional levels of refinement, factor of 4. Time is $t = 0.125$ sec. Computations were performed on 8192 cores at NERSC.

near the contraction. The finest AMR level contains 8975 boxes and covers less than 3% of the domain. The Reynolds number is approximately 6300. Figure 20 depicts transient turbulent flow in the contraction at $t = 0.125$ seconds. We show both velocity and pressure with extreme gradients near and just downstream of the contraction. We show increased magnification in the lower figures with box boundaries and, in the bottom row, mesh resolution.

5. Conclusions

We present a conservative, second-order accurate method to solve the incompressible Navier–Stokes equations in complex geometries. The method is based on a finite volume, embedded boundary approach that makes use of the discrete form of the divergence theorem to discretize the solution in irregular control volumes resulting from the intersection of solid boundaries with a regular, Cartesian grid. The method reduces to a standard finite difference approach in regular cells away from the boundary. We introduce several novel ideas including a volume-weighted scheme that avoids the small-cell problem associated with cut cell methods and a conservative cell-centered gradient for approximate projections. We have coupled the embedded boundary method with AMR to provide a high-performance, high-resolution simulation tool for modeling multiscale, multiphysics problems in complex geometries. The algorithm scales to 262,144 processor cores and is amenable to direct simulation from image data. The cut cell algorithm described here is the basis for a high-performance production code that models 3D engineering scale problems involving incompressible viscous flow and transport in complex geometries. We demonstrate the robustness of the algorithm for a wide range of Reynolds numbers and flow geometries — from creeping flow in realistic pore space to transitional flows past bluff bodies to turbulent pipe flow.

We model moderate Reynolds number phenomena for flow past a cylinder at a fidelity that has not yet been achieved. Typically, only the near wake of the cylinder is modeled numerically as in [22; 61]. In 2D, we observe secondary, tertiary and even quaternary structures far downstream of the near wake at a scale that is much broader than previously modeled, up to 250 cylinder diameters downstream in the wake. By comparison, in 3D, we observe a very long and persistent single train of vortices with coherent structures in the cross-channel direction for the same length wake as in the 2D case. Similarly, we have also demonstrated high resolution of turbulent flow past a sphere in the early stages of wake formation. This capability could prove to be very effective in an investigation of both near and far wake dynamics in high Reynolds number flows past bluff bodies.

We have shown that the method is also suitable for direct numerical simulation of high Reynolds number internal flows. The adaptive capability captures small-scale,

microscopic features in the viscous boundary layer near a singular geometric feature such as a contraction while also resolving bulk flow properties in a domain that is 6 orders of magnitude larger than the finest spatial resolution of the boundary layer. This demonstration was motivated by a large-scale engineering model for worst-case discharge and failure point analysis of the Deepwater Horizon Macondo oil well blowout in 2010.

With demonstrated capability to perform direct simulation from image data, the algorithm has served as the basis for low Reynolds number reactive transport simulations in realistic pore space [37; 38] and is proving to be a useful tool for modeling flow in fractured subsurface materials. The algorithm is also amenable to methods for tracking fluid-fluid and fluid-solid interfaces [34], providing a consistent approach to modeling multiphase flow and time-dependent boundary problems.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and in part by the Office of Basic Energy Sciences Energy Frontier Research Centers and used resources of the National Energy Research Scientific Computing Center all under contract number DE-AC02-05CH11231. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported by the Office of Science of the DOE under contract number DE-AC05-00OR22725. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the DOE under contract DE-AC02-06CH11357. Simulation data in Figure 12 is based upon synchrotron microtomography imagery acquired by Jonathan Ajo-Franklin and Marco Voltolini at the Advanced Light Source, Beamline 8.3.2, which is supported by the Office of Science, Office of Basic Energy Sciences, of the DOE under contract DE-AC02-05CH11231. Simulation data in Figure 13 is based upon FIB-SEM imagery obtained by Lisa Chan at Tescan USA and processed by Terry Ligocki (LBNL), courtesy of Tim Kneafsey (LBNL).

References

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

[2] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, *A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations*, J. Comput. Phys. **142** (1998), no. 1, 1–46. MR 99k:76096 Zbl 0933.76055

[3] A. S. Almgren, J. B. Bell, P. Colella, and T. Marthaler, *A Cartesian grid projection method for the incompressible Euler equations in complex geometries*, SIAM J. Sci. Comput. **18** (1997), no. 5, 1289–1309. MR 98d:76112 Zbl 0910.76040

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

[5] J. B. Bell, P. Colella, and L. H. Howell, *An efficient second-order projection method for viscous incompressible flow*, 10th Computational Fluid Dynamics Conference (Honolulu, 1991), AIAA, 1991, pp. 360–367.

[6] J. B. Bell, P. Colella, and M. L. Welcome, *Conservative front-tracking for inviscid compressible flow*, 10th Computational Fluid Dynamics Conference (Honolulu, 1991), AIAA, 1991, pp. 814–822.

[7] M. J. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys. **82** (1989), no. 1, 64–84. Zbl 0665.76070

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

[9] M. J. Berger and R. J. LeVeque, *Stable boundary conditions for Cartesian grid calculations*, Comput. Syst. Eng. **1** (1990), no. 2–4, 305–311.

[10] M. J. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys. **53** (1984), no. 3, 484–512. MR 85h:65211 Zbl 0536.65071

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

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

[13] M.-H. Chung, *Cartesian cut cell approach for simulating incompressible flows with rigid bodies of arbitrary shape*, Computers and Fluids **35** (2006), no. 6, 607–623. Zbl 1160.76369

[14] W. J. Coirier and K. G. Powell, *An accuracy assessment of Cartesian-mesh approaches for the Euler equations*, J. Comput. Phys. **117** (1995), no. 1, 121–131. MR 95k:76088 Zbl 0817.76055

[15] P. Colella, D. T. Graves, B. J. Keen, and D. Modiano, *A Cartesian grid embedded boundary method for hyperbolic conservation laws*, J. Comput. Phys. **211** (2006), no. 1, 347–366. MR 2006i:65142 Zbl 1120.65324

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

[17] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, *A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method)*, J. Comput. Phys. **152** (1999), no. 2, 457–492. MR 2000c:76061 Zbl 0957.76052

[18] R. P. Fedkiw and X.-D. Liu, *The ghost fluid method for viscous flows*, Innovative methods for numerical solutions of partial differential equations (Arcachon, France, 1998) (M. M. Hafez and J.-J. Chattot, eds.), World Sci. Publ., 2002, pp. 111–143. MR 1928585 Zbl 1078.76586

[19] D. T. Graves, P. Colella, D. Modiano, J. Johnson, B. Sjogreen, and X. Gao, *A Cartesian grid embedded boundary method for the compressible Navier–Stokes equations*, Commun. Appl. Math. Comput. Sci. **8** (2013), no. 1, 99–122. MR 3143820 Zbl 1282.76006

[20] C. Helzel, M. J. Berger, and R. J. LeVeque, *A high-resolution rotated grid method for conservation laws with embedded geometries*, SIAM J. Sci. Comput. **26** (2005), no. 3, 785–809. MR 2005i:35180 Zbl 1074.35071

[21] H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys. **147** (1998), no. 1, 60–85. MR 99m:65231 Zbl 0923.65079

[22] G. E. Karniadakis and G. S. Triantafyllou, *Three-dimensional dynamics and transition to turbulence in the wake of bluff objects*, J. Fluid Mech. **238** (1992), 1–30. Zbl 0754.76043

[23] B. Keen and S. Karni, *A second order kinetic scheme for gas dynamics on arbitrary grids*, J. Comput. Phys. **205** (2005), no. 1, 108–130. MR 2005k:76105 Zbl 1087.76088

[24] J. Kim and P. Moin, *Application of a fractional-step method to incompressible Navier–Stokes equations*, J. Comput. Phys. **59** (1985), no. 2, 308–323. MR 87a:76046 Zbl 0582.76038

[25] M. P. Kirkpatrick, S. W. Armfield, and J. H. Kent, *A representation of curved boundaries for the solution of the Navier–Stokes equations on a staggered three-dimensional Cartesian grid*, J. Comput. Phys. **184** (2003), no. 1, 1–36. Zbl 1118.76350

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

[27] M.-C. Lai and C. S. Peskin, *An immersed boundary method with formal second-order accuracy and reduced numerical viscosity*, J. Comput. Phys. **160** (2000), no. 2, 705–719. MR 2000m:76085 Zbl 0954.76066

[28] R. J. LeVeque and Z. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal. **31** (1994), no. 4, 1019–1044. MR 95g:65139 Zbl 0811.65083

[29] T. J. Ligocki, P. O. Schwartz, J. Percelay, and P. Colella, *Embedded boundary grid generation using the divergence theorem, implicit functions, and constructive solid geometry*, SciDAC 2008 (Seattle), J. Phys. Conf. Ser., no. 125, 2008.

[30] D. F. Martin and P. Colella, *A cell-centered adaptive projection method for the incompressible Euler equations*, J. Comput. Phys. **163** (2000), no. 2, 271–312. MR 2001g:76040 Zbl 0991.76052

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

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

[33] M. K. McNutt, S. Chu, J. Lubchenco, T. Hunter, G. Dreyfus, S. A. Murawski, and D. M. Kennedy, *Applications of science and engineering to quantify and control the Deepwater Horizon oil spill*, Proc. Natl. Acad. Sci. USA **109** (2012), no. 50, 20222–20228.

[34] G. H. Miller and D. Trebotich, *An embedded boundary method for the Navier–Stokes equations on a time-dependent domain*, Commun. Appl. Math. Comput. Sci. **7** (2012), no. 1, 1–31. MR 2893419 Zbl 1273.35215

[35] M. L. Minion, *On the stability of Godunov-projection methods for incompressible flow*, J. Comput. Phys. **123** (1996), no. 2, 435–449. MR 96j:76111 Zbl 0848.76050

[36] H. K. Moffatt, *Viscous and resistive eddies near a sharp corner*, J. Fluid Mech. **18** (1964), no. 1, 1–18. Zbl 0118.20501

[37] S. Molins, D. Trebotich, C. I. Steefel, and C. Shen, *An investigation of the effect of pore scale flow on average geochemical reaction rates using direct numerical simulation*, Water Resour. Res. **48** (2012), no. 3.

[38] S. Molins, D. Trebotich, L. Yang, J. B. Ajo-Franklin, T. J. Ligocki, C. Shen, and C. I. Steefel, *Pore-scale controls on calcite dissolution rates from flow-through laboratory and numerical experiments*, Environ. Sci. Technol. **48** (2014), no. 13, 7453–7460.

[39] C. M. Oldenburg, B. M. Freifeld, K. Pruess, L. Pan, S. Finsterle, and G. J. Moridis, *Numerical simulations of the Macondo well blowout reveal strong control of oil flow by reservoir permeability and exsolution of gas*, Proc. Natl. Acad. Sci. USA **109** (2012), no. 50, 20254–20259.

[40] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome, *An adaptive Cartesian grid method for unsteady compressible flow in irregular regions*, J. Comput. Phys. **120** (1995), no. 2, 278–304. MR 96d:76081 Zbl 0842.76056

[41] ———, *An adaptive Cartesian grid method for unsteady compressible flow in irregular regions*, J. Comput. Phys. **120** (1995), no. 2, 278–304. MR 96d:76081 Zbl 0842.76056

[42] C. S. Peskin, *Flow patterns around heart valves: a numerical method*, J. Comput. Phys. **10** (1972), no. 2, 252–271. Zbl 0244.92002

[43] S. Popinet, *Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries*, J. Comput. Phys. **190** (2003), no. 2, 572–600. MR 2013029 Zbl 1076.76002

[44] J. J. Quirk, *An alternative to unstructured grids for computing gas dynamics flows around arbitrarily complex two-dimensional bodies*, Computers and Fluids **23** (1994), no. 1, 125–142. Zbl 0788.76067

[45] P. O. Schwartz, J. Percelay, T. Ligocki, H. Johansen, D. T. Graves, P. Devendran, P. Colella, and E. Ateljevich, *High-accuracy embedded boundary grid generation using the divergence theorem*, preprint, 2015, To appear in *Commun. Appl. Math. Comput. Sci.*

[46] K. Shahbazi, P. F. Fischer, and C. R. Ethier, *A high-order discontinuous Galerkin method for the unsteady incompressible Navier–Stokes equations*, J. Comput. Phys. **222** (2007), no. 1, 391–407. MR 2007k:76097 Zbl 1216.76034

[47] D. Trebotich, *Simulation of biological flow and transport in complex geometries using embedded boundary/volume-of-fluid methods*, SciDAC 2007 (Boston), J. Phys. Conf. Ser., no. 78, 2007.

[48] D. Trebotich, M. F. Adams, S. Molins, C. I. Steefel, and C. Shen, *High-resolution simulation of pore-scale reactive transport processes associated with carbon sequestration*, Comput. Sci. Eng. **16** (2014), no. 6, 22–31.

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

[50] D. Trebotich, P. Colella, G. Miller, A. Nonaka, T. Marshall, S. Gulati, and D. Liepmann, *A numerical algorithm for complex biological flow in irregular microdevice geometries*, Technical proceedings of the 2004 Nanotechnology Conference and Trade Show (Boston), vol. 2, Nano Science and Technology Institute, 2004, pp. 470–473.

[51] D. Trebotich, G. H. Miller, and M. D. Bybee, *A penalty method to model particle interactions in DNA-laden flows*, J. Nanosci. Nanotechnol. **8** (2008), no. 7, 3749–3756.

[52] D. Trebotich, G. H. Miller, P. Colella, D. T. Graves, D. F. Martin, and P. O. Schwartz, *A tightly coupled particle-fluid model for DNA-laden flows in complex microscale geometries*, Computational Fluid and Solid Mechanics 2005 (Cambridge, MA) (K. J. Bathe, ed.), Elsevier, 2005, pp. 1018–1022.

[53] D. Trebotich, B. Van Straalen, D. T. Graves, and P. Colella, *Performance of embedded boundary methods for CFD with complex geometry*, SciDAC 2008 (Seattle), J. Phys. Conf. Ser., no. 125, 2008.

[54] E. H. Twizell, A. B. Gumel, and M. A. Arigu, *Second-order, L_0-stable methods for the heat equation with time-dependent boundary conditions*, Adv. Comput. Math. **6** (1996), no. 1, 333–352. MR 97m:65164 Zbl 0872.65084

[55] J. van Kan, *A second-order accurate pressure-correction scheme for viscous incompressible flow*, SIAM J. Sci. Statist. Comput. **7** (1986), no. 3, 870–891. MR 87h:76008 Zbl 0594.76023

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

[57] C. H. K. Williamson, *Defining a universal and continuous Strouhal–Reynolds number relationship for the laminar vortex shedding of a circular cylinder*, Phys. Fluids **31** (1988), no. 10, 2742–2744.

[58] ———, *The existence of two stages in the transition to three-dimensionality of a cylinder wake*, Phys. Fluids **31** (1988), no. 11, 3165–3168.

[59] ———, *Vortex dynamics in the cylinder wake*, Annual review of fluid mechanics (Palo Alto, CA) (J. L. Lumley, M. Van Dyke, and H. L. Reed, eds.), vol. 28, Annual Reviews, 1996, pp. 477–539. MR 96j:76051

[60] T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy, *An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries*, J. Comput. Phys. **156** (1999), no. 2, 209–240. Zbl 0957.76043

[61] H.-Q. Zhang, U. Fey, B. R. Noack, M. König, and H. Eckelmann, *On the transition of the cylinder wake*, Phys. Fluids **7** (1995), no. 4, 779–794.

Received February 28, 2014. Revised October 29, 2014.

DAVID TREBOTICH: dptrebotich@lbl.gov
Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

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

COMM. APP. MATH. AND COMP. SCI.
Vol. 10, No. 1, 2015
dx.doi.org/10.2140/camcos.2015.10.83

HIGH-ACCURACY EMBEDDED BOUNDARY GRID GENERATION USING THE DIVERGENCE THEOREM

PETER SCHWARTZ, JULIE PERCELAY, TERRY J. LIGOCKI,
HANS JOHANSEN, DANIEL T. GRAVES, DHARSHI DEVENDRAN,
PHILLIP COLELLA AND ELI ATELJEVICH

We present an algorithm to produce the necessary geometric information for finite volume calculations in the context of Cartesian grids with embedded boundaries. Given an order of accuracy for the overall calculation, we show what accuracy is required for each of the geometric quantities and we demonstrate how to calculate the moments using the divergence theorem. We demonstrate that, for a known flux, these moments can be used to create a flux divergence of the expected order.

1. Introduction

This work is motivated by the desire to solve partial differential equations (PDEs) conservatively in the context of complex geometries. As an example, consider Poisson's equation. Given a charge density ρ, Poisson's equation can be written as

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

for the potential ϕ. If we integrate this equation over a control volume Ω and apply the divergence theorem, this becomes

$$\int_{\partial\Omega} \nabla\phi \cdot \hat{n}\, dA = \int_{\Omega} \rho\, dV,$$

where $\hat{n}$ is the outward-facing unit normal to the surface. Our volumes are Cartesian cells cut by an embedded boundary. Refer to Figure 1 for an illustration. We refer to the unshaded region as the "volume". The boundaries of the volume aligned to coordinate directions that connect to other volumes we refer to as "faces". We refer to the section of the embedded boundary that cuts the volume as the "EB face".

Formally, the underlying description of space is given by rectangular control volumes on a Cartesian mesh $\Upsilon_{\boldsymbol{i}} = [(\boldsymbol{i} - \frac{1}{2}\boldsymbol{u})h, (\boldsymbol{i} + \frac{1}{2}\boldsymbol{u})h]$, $\boldsymbol{i} \in \mathbb{Z}^D$, where D

Research at LBNL was supported financially by the Office of Advanced Scientific Computing Research of the US Department of Energy under contract number DE-AC02-05CH11231. All work was done using the Chombo software infrastructure developed by LBNL [4; 5].
MSC2010: 65N12, 65N50, 65N08.
Keywords: Cartesian grid embedded boundaries, grid generation, finite volume methods.

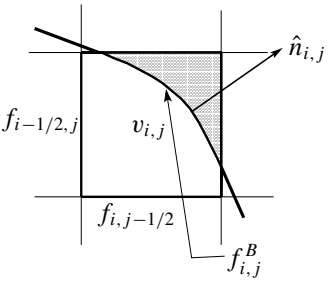


Figure 1. Illustration of cut cell notation. The shaded region is outside the solution domain. The volume v is connected to other volumes via the faces aligned with the coordinate planes. The EB face is formed by the intersection of the embedded boundary and the cell.

is the dimensionality of the problem, h is the mesh spacing, and $\boldsymbol{u}$ is the vector whose entries are all one (note we use bold font $\boldsymbol{u} = (u_1, \ldots, u_d, \ldots, u_D)$ to indicate a vector quantity). Given an irregular domain Ω, we obtain control volumes $V_{\boldsymbol{i}} = \Upsilon_{\boldsymbol{i}} \cap \Omega$ and faces $A_{\boldsymbol{i},d\pm} = A_{\boldsymbol{i}\pm\boldsymbol{e}_d/2}$, which are the intersection of the boundary of $\partial V_{\boldsymbol{i}}$ with the coordinate planes $\{\boldsymbol{x} : x_d = (i_d \pm \frac{1}{2})h\}$ ($\boldsymbol{e}_d$ is the unit vector in the d direction). We also define $A_{B,\boldsymbol{i}}$ to be the intersection of the boundary of the irregular domain with the Cartesian control volume: $A_{B,\boldsymbol{i}} = \partial\Omega \cap \Upsilon_{\boldsymbol{i}}$. From here on, the subscript $\boldsymbol{i}$ is implied as the analysis applies to any given volume $V_{\boldsymbol{i}}$.

Given a flux $\boldsymbol{F}$ (in the case of the Poisson equation (1), $\boldsymbol{F} = \nabla\phi$), we can rewrite the volume integral of the divergence of $\boldsymbol{F}$ as an integral over each face in the volume:

$$\int_V \nabla \cdot \boldsymbol{F}\, dV = \sum_{d=1}^{D} \left(\int_{A_{d+}} F_d\, dA - \int_{A_{d-}} F_d\, dA + \int_{A_B} F_d\, n_d(\boldsymbol{x})\, dA \right), \qquad (2)$$

where we define $n_d(\boldsymbol{x})$ to be the d-th component of the outward-facing unit normal to the EB face. The accuracy with which one computes the integrals above will depend on the accuracy of the geometric description of the volume and its associated faces.

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

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

$$\boldsymbol{p}! = \prod_{d=1}^{D} p_d!.$$

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

$$m_v^{\boldsymbol{p}}(\bar{\boldsymbol{x}}) = \int_V (\boldsymbol{x} - \bar{\boldsymbol{x}})^{\boldsymbol{p}}\, dV. \qquad (3)$$

Clearly, the volume of the cut cell $|V| = m_v^{\boldsymbol{z}}$, where $\boldsymbol{z}$ is the zero vector. We define the face moments $m_{d\pm}^{\boldsymbol{p}}(\bar{\boldsymbol{x}})$ to be the $\boldsymbol{p}$-th moments (relative to the point $\bar{\boldsymbol{x}}$) of the faces $A_{d\pm}$:

$$m_{d\pm}^{\boldsymbol{p}}(\bar{\boldsymbol{x}}) = \int_{A_{d\pm}} (\boldsymbol{x} - \bar{\boldsymbol{x}})^{\boldsymbol{p}}\, dA. \tag{4}$$

We define two moments corresponding to the embedded boundary face A_B:

$$m_B^{\boldsymbol{p}}(\bar{\boldsymbol{x}}) = \int_{A_B} (\boldsymbol{x} - \bar{\boldsymbol{x}})^{\boldsymbol{p}}\, dA, \tag{5}$$

$$m_{B,d}^{\boldsymbol{p}}(\bar{\boldsymbol{x}}) = \int_{A_B} (\boldsymbol{x} - \bar{\boldsymbol{x}})^{\boldsymbol{p}} n_d(\boldsymbol{x})\, dA. \tag{6}$$

Note that (6) includes the normal to the embedded boundary face.

Given a sufficiently smooth function ψ, we can approximate ψ in the neighborhood of $\bar{\boldsymbol{x}}$ using a multidimensional Taylor expansion:

$$\psi(\boldsymbol{x}) = \sum_{|\boldsymbol{q}|<Q} \frac{1}{\boldsymbol{q}!} \psi^{(\boldsymbol{q})}(\bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})^{\boldsymbol{q}} + O(h^Q) \tag{7}$$

with the multi-index partial derivative notation

$$\psi^{(\boldsymbol{q})} = \partial^{\boldsymbol{q}}\psi = \frac{\partial^{q_1}}{\partial x_1^{q_1}} \cdots \frac{\partial^{q_D}}{\partial x_D^{q_D}} \psi. \tag{8}$$

We express averages over volumes as

$$\langle \nabla \cdot \boldsymbol{F} \rangle_V \equiv \frac{1}{|V|} \int_V (\nabla \cdot \boldsymbol{F})\, dV.$$

We define the volume fraction κ to be fraction of the volume of the cell inside the solution domain so that

$$\kappa = h^{-D}|V| = h^{-D} m_v^{\boldsymbol{z}}. \tag{9}$$

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

$$\begin{aligned} \kappa \langle \nabla \cdot \boldsymbol{F} \rangle_V &= \frac{1}{h^D} \int_V \nabla \cdot \boldsymbol{F}\, dV \\ &= \frac{1}{h^D} \sum_{d=1}^{D} \left(\int_{A_{d+}} F_d(\boldsymbol{x})\, dA - \int_{A_{d-}} F_d(\boldsymbol{x})\, dA + \int_{A_B} F_d(\boldsymbol{x}) n_d(\boldsymbol{x})\, dA \right). \end{aligned} \tag{10}$$

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

$$\kappa \langle \nabla \cdot \nabla \phi \rangle_V = \kappa \langle \rho \rangle_V$$

for ϕ [9; 17], which avoids very large negative eigenvalues from terms with κ^{-1}. Since κ is only a diagonal scaling, the accuracy of the method is primarily dependent on the accuracy of the discretization of $(\nabla \cdot \nabla\phi)$.

Explicit algorithms for hyperbolic systems with flux $\boldsymbol{F}$ may use a hybrid operator that is a linear combination of a conservative discretization for $\kappa\langle\nabla \cdot \boldsymbol{F}\rangle$ and a nonconservative (not in flux-divergence form) but stable approximation of $\nabla \cdot \boldsymbol{F}$ to advance the solution. The loss of conservation for this hybrid operator can be calculated and redistributed to neighboring cells so that the overall scheme is globally conservative [2; 6; 3; 8; 14; 13]. Suppose we have $\kappa D^{\mathrm{C}}(\boldsymbol{F})$, an $O(h^P)$ conservative discretization of $\kappa\nabla \cdot \boldsymbol{F}$. Suppose we also have $D^{\mathrm{NC}}(\boldsymbol{F})$, an $O(h^P)$ nonconservative discretization of $\nabla \cdot \boldsymbol{F}$, i.e.,

$$D^{\mathrm{NC}}(\boldsymbol{F}) = \nabla \cdot \boldsymbol{F} + O(h^P),$$
$$\kappa D^{\mathrm{C}}(\boldsymbol{F}) = \kappa\nabla \cdot \boldsymbol{F} + O(h^P).$$

The hybrid operator $D^{\mathrm{H}}(\boldsymbol{F})$ is given by

$$\begin{aligned} D^{\mathrm{H}}(\boldsymbol{F}) &= \kappa D^{\mathrm{C}}(\boldsymbol{F}) - (1-\kappa)D^{\mathrm{NC}}(\boldsymbol{F}) \\ &= \nabla \cdot \boldsymbol{F} + O(h^P), \end{aligned}$$

and the resulting loss of conservation in each cell is given by

$$\begin{aligned} \delta &= \kappa D^{\mathrm{C}}(\boldsymbol{F}) - \kappa D^{\mathrm{H}}(\boldsymbol{F}) \\ &= \kappa(1-\kappa)(D^{\mathrm{C}}(\boldsymbol{F}) - D^{\mathrm{NC}}(\boldsymbol{F})) = O(h^P), \end{aligned}$$

where for stability arguments δ must be redistributed in a κ-weighted way [3; 8]. This is of the order of the truncation error so that the accuracy of the method only depends on the accuracy of the two discretizations of the flux divergence. We wish to investigate what accuracy is necessary for the moment calculations to achieve a given order of accuracy in $\kappa D^{\mathrm{C}}(\boldsymbol{F})$.

Embedded boundaries have been used in a wide variety of applications, and several different grid generation techniques have emerged. In an early paper in the field, Pember et al. [16] use a piecewise-planar approximation for grid generation for solutions of inviscid, polytropic gas dynamics. Aftosmis et al. [1] use triangulation to generate grids for a wide variety of extremely complex geometries; their software is still widely used. Singh et al. [18] use triangulation to reconstruct moving boundaries. The embedded boundary algorithms in [3; 8; 14; 13] use a second-order, implicit function–based approach. Miller et al. [12] use a time-dependent, second-order implicit function approach to cut four-dimensional cells to generate moving geometries. Sussman and Puckett [19] use piecewise-planar volume of fluid reconstruction in their incompressible flow algorithm. Nourgaliev et al. [15] and Marella et al. [11] use a piecewise-linear reconstruction in their sharp interface

methods. All these algorithms use piecewise-planar approximations to the cutting surface to generate geometric information. These algorithms are first order in $D(\boldsymbol{F})$ (second-order fluxes), so second-order geometric information is accurate enough for their purposes. In the graphics and computational geometry community, there has been a substantial amount of work done in geometric moment computation using the divergence theorem. Yang et al. [20] use the divergence theorem to compute geometric moments for image analysis. Gonzalez-Ochoa et al. [7] have a fast divergence theorem–based moment algorithm specialized for surfaces that are described as polynomials. These algorithms are highly focused on reducing computational complexity, and though their accuracy is measured for a particular refinement, no concern is given to the algorithms' convergence rate with grid refinement. The current work is intended to be a careful exploration of the required accuracy for geometric moments in the context of embedded boundary calculations using higher-order finite volume methods.

2. Accuracy of the discrete divergence

Given an order of accuracy Q, suppose we can approximate $\boldsymbol{F}$ to $O(h^Q)$. If we expand $\boldsymbol{F}$ in (2) using (7), we get

$$\int_V (\nabla \cdot \boldsymbol{F})\, dV = \sum_{|\boldsymbol{q}|<Q} \sum_{d=1}^{D} \frac{1}{\boldsymbol{q}!} F_d^{(q)}(\bar{\boldsymbol{x}}) \left(\int_{A_{d+}} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{q}}\, dA - \int_{A_{d-}} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{q}}\, dA + \int_{A_B} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{q}} n_d(\boldsymbol{x})\, dA \right) + O(h^{Q+D-1}).$$

The accuracy of the expression is $O(h^{Q+D-1})$ because it is an $O(h^Q)$ Taylor series integrated over an area of order h^{D-1}. Using our moment definitions ((4) and (6)),

$$\int_V (\nabla \cdot F)\, dV = \sum_{|\boldsymbol{q}|<Q} \sum_{d=1}^{D} \frac{1}{\boldsymbol{q}!} F_d^{(q)}(\bar{\boldsymbol{x}}) \left(m_{d+}^{\boldsymbol{q}} - m_{d-}^{\boldsymbol{q}} + m_{B,d}^{\boldsymbol{q}} \right) + O(h^{Q+D-1}). \quad (11)$$

Now suppose we can discretize each moment to $O(h^R)$. We define our discrete moments as

$$M_v^{\boldsymbol{q}} = \int_V (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{q}}\, dV + O(h^R) = m_v^{\boldsymbol{q}} + O(h^R), \quad (12)$$

$$M_{d\pm}^{\boldsymbol{q}} = \int_{A_{d\pm}} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{q}}\, dA + O(h^R) = m_{d\pm}^{\boldsymbol{q}} + O(h^R), \quad (13)$$

$$M_{B,d}^{\boldsymbol{q}} = \int_{A_B} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{q}} n_d\, dA + O(h^R) = m_{B,d}^{\boldsymbol{q}} + O(h^R), \quad (14)$$

$$M_B^{\boldsymbol{q}} = \int_{A_B} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{q}}\, dA + O(h^R) = m_B^{\boldsymbol{q}} + O(h^R). \quad (15)$$

We take these moment definitions and put them into (11) to get our discrete, κ-weighted average divergence of the flux:

$$\begin{aligned}\kappa D(\boldsymbol{F})_i &= \frac{1}{h^D}\sum_{|\boldsymbol{q}|<Q}\sum_{d=1}^{D}\frac{1}{\boldsymbol{q}!}F_d^{(q)}(\bar{\boldsymbol{x}})\left(M_{d+}^{\boldsymbol{q}}-M_{d-}^{\boldsymbol{q}}+M_{B,d}^{\boldsymbol{q}}\right)\\ &= \kappa\langle\nabla\cdot\boldsymbol{F}\rangle_V + O(h^{Q-1}) + O(h^{R-D}).\end{aligned} \tag{16}$$

The first error term is from (11), and the second is from (13) and (14); both have been divided by the full cell volume, h^D. So if one desires a weighted divergence of accuracy $O(h^P)$, one needs fluxes approximated to accuracy $O(h^Q)$ and moments approximated to accuracy $O(h^R)$, where

$$\begin{aligned}Q &= P+1,\\ R &= P+D.\end{aligned} \tag{17}$$

3. Algorithm

Consider a volume V at cell i (see Figure 1), and let $\bar{\boldsymbol{x}}$ be some point in cell i. We may choose $\bar{\boldsymbol{x}}$ as the origin of our coordinate system so that $\|\boldsymbol{x}-\bar{\boldsymbol{x}}\| = O(h)$ for any point $\boldsymbol{x}$ in V. In the following derivation, we will set $\bar{\boldsymbol{x}} = 0$ without loss of generality.

In (2), if we choose $\boldsymbol{F} = \boldsymbol{x}^{\boldsymbol{q}}\boldsymbol{e}_d$, we get D equations of the form

$$q_d\int_V \boldsymbol{x}^{\boldsymbol{q}-\boldsymbol{e}_d}\,dV = \int_{A_{d+}}\boldsymbol{x}^{\boldsymbol{q}}\,dA - \int_{A_{d-}}\boldsymbol{x}^{\boldsymbol{q}}\,dA + \int_{A_B}\boldsymbol{x}^{\boldsymbol{q}}n_d\,dA.$$

Recall that the normal to the embedded boundary is a function of space $\boldsymbol{n}(\boldsymbol{x})$, so we assume that we can expand this last integral using a Taylor series of $\boldsymbol{n}$ about $\bar{\boldsymbol{x}} = 0$ to an appropriate order of accuracy S:

$$\int_{A_B}\boldsymbol{x}^{\boldsymbol{q}}n_d(\boldsymbol{x})\,dA = \sum_{|\boldsymbol{s}|<S}\frac{1}{\boldsymbol{s}!}\partial^{\boldsymbol{s}}n_d(0)m_B^{\boldsymbol{q}+\boldsymbol{s}} + O(h^{|\boldsymbol{q}|+D+S-1}). \tag{18}$$

The advantage of this expansion is that m_B does not include $n_d(\boldsymbol{x})$ in the integrand. Combining and moving the first term of the Taylor series to the left-hand side, we have

$$\begin{aligned}&q_d m_v^{\boldsymbol{q}-\boldsymbol{e}_d} - n_d(0)m_B^{\boldsymbol{q}}\\ &\qquad = m_{d+}^{\boldsymbol{q}} - m_{d-}^{\boldsymbol{q}} + \sum_{0<|\boldsymbol{s}|<S}\frac{1}{\boldsymbol{s}!}\partial^{\boldsymbol{s}}n_d(0)m_B^{\boldsymbol{q}+\boldsymbol{s}} + O(h^{|\boldsymbol{q}|+D+S-1}).\end{aligned} \tag{19}$$

Note that, for higher-order moments, we need fewer terms in the Taylor series for the normal to obtain a truncation error of $O(h^R)$ because the truncation error depends on $|\boldsymbol{q}|+S$. In particular, if $|\boldsymbol{q}| = Q$, (17) shows that we only need $S = 0$,

0. Compute all derivatives of the normal, $\partial^s n_d$ (see Section 3.2).
1. Compute one-dimensional moments $M^{\boldsymbol{q}}_{V,1}$ using root-finding and integration.
2. Irregular one-dimensional moments $M^{\boldsymbol{q}}_{B,1}$ are zero.
3. Loop through other dimensions as follows:

for $(D = 2, 3)$
for $(|\boldsymbol{q}| = \{Q, Q-1, \ldots, 0\})$
Set ρ:

$$\rho = M^{\boldsymbol{q}}_{d+,D-1} - M^{\boldsymbol{q}}_{d-,D-1} + \sum_{1 \le |\boldsymbol{s}| < S} \frac{\partial^{\boldsymbol{s}} n_d}{\boldsymbol{s}!} M^{\boldsymbol{q}+\boldsymbol{s}}_{B,D}.$$

Solve for $M^{\boldsymbol{q}-\boldsymbol{e}_d}_{V,D}$ and $M^{\boldsymbol{q}}_{B,D}$:

$$q_d M^{\boldsymbol{q}-\boldsymbol{e}_d}_{V,D} - n_d M^{\boldsymbol{q}}_{B,D} = \rho.$$

end loop over moments
end loop over dimensions

Figure 2. Outline of the moment algorithm. The second subscript of the moments refers to dimensionality ($M^{\boldsymbol{p}}_{v,3}$ refers to the three-dimensional $M^{\boldsymbol{p}}_{v}$, for example).

and the third term on the right-hand side drops out. Also, notice that $m^{\boldsymbol{q}}_{d+}$ and $m^{\boldsymbol{q}}_{d-}$ are integrals of lower dimension.

We drop the truncation error term in (19) and define the approximate moments ($M^{\boldsymbol{q}}_{v}$, etc.) using the set of equations

$$q_d M^{\boldsymbol{q}-\boldsymbol{e}_d}_{v} - n_d(0) M^{\boldsymbol{q}}_{B} = M^{\boldsymbol{q}}_{d+} - M^{\boldsymbol{q}}_{d-} + \sum_{1 \le |\boldsymbol{s}| < S} \frac{1}{\boldsymbol{s}!} \partial^{\boldsymbol{s}} n_d(0) M^{\boldsymbol{q}+\boldsymbol{s}}_{B}. \tag{20}$$

If the moments on the left are treated as unknowns and the moments on the right are treated as known, then this set of equations forms a system of linear equations for a fixed $|\boldsymbol{q}|$. The key to the algorithm is to compute the moments in an order that assures the quantities on the right are known. In particular, we compute the moments in lower dimensions first and use them as known quantities in the equation for higher dimension. For a given dimension, we generate moments in order of decreasing $|\boldsymbol{q}|$ starting with $|\boldsymbol{q}| = Q$. Because the third term on the right-hand side of (20) depends on moments with orders $|\boldsymbol{q}|+1, \ldots, Q$, the procedure guarantees that term is known. The algorithm can also be described as a recursion as shown by Ligocki et al. [10] (the conference paper associated with this work). Figure 2 shows an outline of the algorithm. The number of unknowns is $N_{|\boldsymbol{q}|-1} + N_{|\boldsymbol{q}|}$, where $N_{|\boldsymbol{q}|}$ is the number of monomials of degree $|\boldsymbol{q}|$. (One can think of $N_{|\boldsymbol{q}|}$ as the length of the $|\boldsymbol{q}|$-th row of the Pascal's triangle formed by the moments.) The number of equations is $DN_{|\boldsymbol{q}|}$. Since $N_{|\boldsymbol{q}|-1} < N_{|\boldsymbol{q}|}$, this is an overdetermined system that can be solved using least squares. The normal and all its derivatives are computed as shown in Section 3.2.

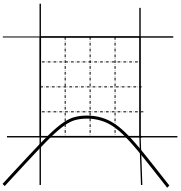


Figure 3. Illustration of an under-resolved geometry. When the curvature of the implicit function is too great to be resolved, we refine locally and sum the finer moments to compute the under-resolved cell's moments.

3.1. ***Under-resolved geometry.*** Note that the above algorithm can fail in the case of an under-resolved geometry. If the curvature of the implicit function is too high, the situation shown in Figure 3 can result. To resolve this, we refine the grid locally until the implicit function is resolved. We use the divergence of the normal as a measure for under-resolution. Given a tolerance ϵ, if $D(\boldsymbol{n}) > \epsilon$ at volume $\boldsymbol{v}$ and the set of finer volumes $S = \{\boldsymbol{v}_f\}$ that compose the volume $\boldsymbol{v}$, we get the equation

$$\boldsymbol{m}_{\boldsymbol{v}} = \sum_{\boldsymbol{v}_f \in S} \boldsymbol{m}_{\boldsymbol{v}_f}.$$

If the geometry is not smooth enough to be resolved, a maximum level of resolution is defined. This paper is primarily concerned with convergence rates. Convergence tests require smooth, resolved geometries, so none of the geometries presented in this paper need this refinement.

3.2. ***Calculation of derivatives of the normal using an implicit function.*** Our geometry is defined as the zero set of an implicit function $\psi(\boldsymbol{x})$, the zero level set of which is the embedded boundary. The normal $\hat{n}$ of the embedded boundary is defined as

$$\hat{n} = \frac{\nabla \psi}{L},$$

where $L = |\nabla \psi|$. For the d-th component of the normal, the multi-index product rule can be used for higher partial derivatives $\boldsymbol{p}$ of the quantity $\psi^{(\boldsymbol{e}_d)} = L n_d$:

$$\begin{aligned} \psi^{(\boldsymbol{p}+\boldsymbol{e}_d)} &= \partial^{\boldsymbol{p}} \psi^{(\boldsymbol{e}_d)} \\ &= \partial^{\boldsymbol{p}} (L n_d) \\ &= \sum_{\boldsymbol{q} \le \boldsymbol{p}} \binom{\boldsymbol{p}}{\boldsymbol{q}} \partial^{\boldsymbol{p}-\boldsymbol{q}} L \partial^{\boldsymbol{q}} n_d. \end{aligned}$$

Note that the left-hand side is a known value from the implicit function. On the right-hand side, we have unknown derivatives of L and n_d. We can rewrite to express the highest normal derivative as

$$L \partial^{\boldsymbol{p}} n_d = \psi^{(\boldsymbol{p}+\boldsymbol{e}_d)} - \sum_{\boldsymbol{q} \le \boldsymbol{p},\ \boldsymbol{q} \ne \boldsymbol{p}} \binom{\boldsymbol{p}}{\boldsymbol{q}} \partial^{\boldsymbol{p}-\boldsymbol{q}} L \partial^{\boldsymbol{q}} n_d.$$

Next, we will evaluate the derivatives of L. To find a recursion formula, we first apply the chain rule to ∂L^2:

$$\begin{aligned}\partial^{\boldsymbol{e}^i} L^2 &= \partial^{\boldsymbol{e}^i} \sum_d (\psi^{(\boldsymbol{e}_d)})^2,\\ 2L\partial^{\boldsymbol{e}^i} L &= \sum_d 2\psi^{(\boldsymbol{e}_d)} \psi^{(\boldsymbol{e}_d+\boldsymbol{e}^i)},\\ \partial^{\boldsymbol{e}^i} L &= \sum_d \frac{\psi^{(\boldsymbol{e}_d)}}{L} \psi^{(\boldsymbol{e}_d+\boldsymbol{e}^i)}\\ &= \sum_d n_d \psi^{(\boldsymbol{e}_d+\boldsymbol{e}^i)}.\end{aligned}$$

Looking at the right-hand side, we have calculated a derivative $\partial^{\boldsymbol{e}^i} L$ from n_d and known derivatives of ψ. Extending to the case of $\boldsymbol{p} \geq \boldsymbol{e}^i$, we have

$$\begin{aligned}\partial^{\boldsymbol{p}} L &= \partial^{\boldsymbol{p}-\boldsymbol{e}^i}(\partial^{\boldsymbol{e}^i}\psi) && (21)\\ &= \partial^{\boldsymbol{p}-\boldsymbol{e}^i} \sum_d n_d \psi^{(\boldsymbol{e}_d+\boldsymbol{e}^i)} && (22)\\ &= \sum_d \sum_{\boldsymbol{r}\leq\boldsymbol{q}} \binom{\boldsymbol{q}}{\boldsymbol{r}} \partial^{\boldsymbol{r}} n_d \partial^{\boldsymbol{q}-\boldsymbol{r}+\boldsymbol{e}_d+\boldsymbol{e}^i}\psi, \quad \text{where } \boldsymbol{q} = \boldsymbol{p}-\boldsymbol{e}^i. && (23)\end{aligned}$$

Again, this expresses higher-order derivatives of L in terms of lower-order derivatives of n_d and known derivatives of ψ. The choice of $\boldsymbol{e}^i$ is arbitrary but is such that all the $\partial^{\boldsymbol{r}} n_d$ have already been computed. To calculate all the powers needed in (23), we can choose $\boldsymbol{e}^i = \boldsymbol{e}_d$, calculate the required derivatives for $d = 0$ first, then add those with derivatives in $d = 1$, etc. This guarantees that the requested derivatives will have already been calculated in the recursion.

4. Results

4.1. ***Geometric description.*** Our convergence tests are calculated using a geometry of an ellipsoid (or an ellipse in two dimensions) centered in a unit domain. The implicit function, $\psi(\boldsymbol{x})$, that defines the ellipsoid is given by

$$\psi(\boldsymbol{x}) = \sum_{d=1}^{D} \left(\frac{x_d - x_{0,d}}{A_d}\right)^2 - R^2,$$

where $\boldsymbol{x}_0$ is put at the center of the domain. The stretching constant $\boldsymbol{A} = \{1, 2, 3\}$ in three dimensions ($\boldsymbol{A} = \{1, 2\}$ in two dimensions). The constant, $R = 0.15$, sets the unexpanded radius of the ellipsoid. The surface of the ellipsoid is described by the surface where $\psi(\boldsymbol{x}) = 0$.

Variable	ϵ^{2h}	ϖ	ϵ^{h}	Variable	ϵ^{2h}	ϖ	ϵ^{h}
$M_v^{(0,0)}$	3.405×10^{-12}	7.08	2.525×10^{-14}	$M_B^{(0,0)}$	2.719×10^{-10}	6.92	2.244×10^{-12}
$M_v^{(1,0)}$	1.330×10^{-13}	6.30	1.688×10^{-15}	$M_B^{(1,0)}$	1.802×10^{-11}	6.98	1.426×10^{-13}
$M_v^{(2,0)}$	2.028×10^{-14}	7.00	1.582×10^{-16}	$M_B^{(2,0)}$	2.684×10^{-12}	6.97	2.143×10^{-14}
$M_v^{(3,0)}$	2.783×10^{-15}	6.51	3.052×10^{-17}	$M_B^{(3,0)}$	1.495×10^{-13}	7.06	1.122×10^{-15}
$M_v^{(4,0)}$	1.490×10^{-15}	7.03	1.141×10^{-17}	$M_B^{(4,0)}$	4.426×10^{-14}	7.00	3.466×10^{-16}
$M_v^{(0,1)}$	9.842×10^{-13}	7.05	7.402×10^{-15}	$M_B^{(0,1)}$	1.634×10^{-11}	7.01	1.266×10^{-13}
$M_v^{(1,1)}$	3.844×10^{-14}	6.40	4.550×10^{-16}	$M_B^{(1,1)}$	1.931×10^{-13}	5.29	4.945×10^{-15}
$M_v^{(2,1)}$	6.041×10^{-15}	7.02	4.655×10^{-17}	$M_B^{(2,1)}$	1.952×10^{-13}	6.97	1.561×10^{-15}
$M_v^{(3,1)}$	8.044×10^{-16}	6.61	8.225×10^{-18}	$M_B^{(3,1)}$	9.528×10^{-15}	7.05	7.168×10^{-17}
$M_v^{(0,2)}$	1.826×10^{-13}	7.02	1.411×10^{-15}	$M_B^{(0,2)}$	6.589×10^{-13}	6.94	5.363×10^{-15}
$M_v^{(1,2)}$	7.132×10^{-15}	6.63	7.202×10^{-17}	$M_B^{(1,2)}$	3.190×10^{-15}	5.70	6.125×10^{-17}
$M_v^{(2,2)}$	4.375×10^{-16}	5.76	8.079×10^{-18}	$M_B^{(2,2)}$	1.459×10^{-14}	7.06	1.094×10^{-16}
$M_v^{(0,3)}$	6.396×10^{-15}	5.75	1.185×10^{-16}	$M_B^{(0,3)}$	5.994×10^{-13}	7.06	4.493×10^{-15}
$M_v^{(1,3)}$	5.497×10^{-16}	6.13	7.872×10^{-18}	$M_B^{(1,3)}$	2.194×10^{-14}	6.76	2.025×10^{-16}
$M_v^{(0,4)}$	8.242×10^{-16}	6.76	7.610×10^{-18}	$M_B^{(0,4)}$	2.491×10^{-14}	5.74	4.657×10^{-16}

Table 1. Volume (left) and embedded boundary (right) moment convergence rates for $h = 1/128$ using the L_∞ norm in two dimensions. The implicit function is an ellipse described in Section 4.1.

4.2. ***Moment convergence tests.*** To test the convergence rate of the moments, we use Richardson extrapolation, which means that the exact solution on a finer level of refinement is used as an exact solution. Since we are dealing with integrals, the coarsening operation is simple addition. Define $A^{h\to 2h}$ to be the operator to get the exact solution on the coarse level from the fine solution. Given S_f, the set of fine volumes that cover a coarse volume $\boldsymbol{i}$,

$$A^{h\to 2h}(\boldsymbol{M})_{\boldsymbol{i}} = \sum_{\boldsymbol{i}_f \in S_f} \boldsymbol{M}_{\boldsymbol{i}_f}.$$

$\boldsymbol{M}_h$ is defined to be our solution on a grid with resolution h. For an exact solution $\boldsymbol{m}^e$, we use $\boldsymbol{m}^e_{2h} = A^{h\to 2h}(\boldsymbol{M}_h)$ and the error is given by

$$\epsilon^h = \boldsymbol{M}^h(t) - \boldsymbol{m}^e(t). \tag{24}$$

The order of convergence ϖ is estimated by

$$\varpi = \frac{\log(\|\epsilon^{2h}\|/\|\epsilon^h\|)}{\log(2)}. \tag{25}$$

Variable	ϵ^{2h}	ϖ	ϵ^{h}	Variable	ϵ^{2h}	ϖ	ϵ^{h}
$M_v^{(0,0,0)}$	7.744×10^{-13}	5.73	1.461×10^{-14}	$M_v^{(3,0,1)}$	3.329×10^{-17}	7.54	1.789×10^{-19}
$M_v^{(1,0,0)}$	6.130×10^{-15}	6.75	5.706×10^{-17}	$M_v^{(0,1,1)}$	1.577×10^{-15}	7.87	6.745×10^{-18}
$M_v^{(2,0,0)}$	9.836×10^{-16}	7.89	4.153×10^{-18}	$M_v^{(1,1,1)}$	1.173×10^{-16}	7.97	4.682×10^{-19}
$M_v^{(3,0,0)}$	1.096×10^{-16}	7.84	4.773×10^{-19}	$M_v^{(2,1,1)}$	1.845×10^{-17}	7.75	8.553×10^{-20}
$M_v^{(4,0,0)}$	5.936×10^{-17}	7.98	2.356×10^{-19}	$M_v^{(0,2,1)}$	1.314×10^{-16}	7.59	6.834×10^{-19}
$M_v^{(0,1,0)}$	5.449×10^{-14}	6.44	6.281×10^{-16}	$M_v^{(1,2,1)}$	1.383×10^{-17}	7.72	6.545×10^{-20}
$M_v^{(1,1,0)}$	1.132×10^{-15}	7.77	5.178×10^{-18}	$M_v^{(0,3,1)}$	1.338×10^{-17}	7.26	8.748×10^{-20}
$M_v^{(2,1,0)}$	1.761×10^{-16}	8.11	6.382×10^{-19}	$M_v^{(0,0,2)}$	1.036×10^{-14}	7.61	5.304×10^{-17}
$M_v^{(3,1,0)}$	3.908×10^{-17}	7.87	1.672×10^{-19}	$M_v^{(1,0,2)}$	4.254×10^{-16}	7.53	2.306×10^{-18}
$M_v^{(0,2,0)}$	4.901×10^{-15}	7.95	1.985×10^{-17}	$M_v^{(2,0,2)}$	2.390×10^{-17}	7.40	1.414×10^{-19}
$M_v^{(1,2,0)}$	3.301×10^{-16}	7.87	1.412×10^{-18}	$M_v^{(0,1,2)}$	1.772×10^{-16}	7.98	7.008×10^{-19}
$M_v^{(2,2,0)}$	3.225×10^{-17}	7.74	1.505×10^{-19}	$M_v^{(1,1,2)}$	1.369×10^{-17}	8.02	5.263×10^{-20}
$M_v^{(0,3,0)}$	4.400×10^{-16}	7.70	2.114×10^{-18}	$M_v^{(0,2,2)}$	1.307×10^{-17}	7.41	7.664×10^{-20}
$M_v^{(1,3,0)}$	2.666×10^{-17}	7.61	1.367×10^{-19}	$M_v^{(0,0,3)}$	5.093×10^{-16}	7.45	2.913×10^{-18}
$M_v^{(0,4,0)}$	2.817×10^{-17}	7.66	1.393×10^{-19}	$M_v^{(1,0,3)}$	2.115×10^{-17}	7.52	1.156×10^{-19}
$M_v^{(0,0,1)}$	1.570×10^{-14}	7.04	1.196×10^{-16}	$M_v^{(0,1,3)}$	1.376×10^{-17}	7.67	6.749×10^{-20}
$M_v^{(1,0,1)}$	2.987×10^{-16}	7.50	1.654×10^{-18}	$M_v^{(0,0,4)}$	2.252×10^{-17}	7.43	1.310×10^{-19}
$M_v^{(2,0,1)}$	3.498×10^{-16}	7.96	1.408×10^{-18}				

Table 2. Volume moment convergence rates for $h = 1/128$ using the L_∞ norm in three dimensions. The implicit function is the ellipsoid described in Section 4.1.

We set $P = 5$, so in two dimensions, we expect all the moments to converge to order $h^{P+D=7}$. Table 1 shows these two-dimensional results, and we show the expected convergence, even using the L_∞ norm. In three dimensions, we expect (with $P = 5$) the moments to all converge to $O(h^8)$. Tables 2 and 3 show these three-dimensional results, and we show the expected convergence, also using the L_∞ norm.

4.3. ***Flux divergence convergence tests.*** We use an analytic function as a flux function $F = \nabla\psi$, where

$$\psi = \prod_{d=1}^{D} \cos(2\pi(x_d - x_{0,d})),$$

where $\boldsymbol{x}_0$ is the center of the domain. We set $P = 4$ and compute D^K using (16). In both two and three dimensions, we get the expected rates of convergence. Near the embedded boundary, we get third-order convergence, and away from the irregular boundary, we get fourth-order. Table 4 shows the $O(3, 3.5, 4)$ rates that one would expect in the L_∞, L_2, and L_1 norms.

Variable	ϵ^{2h}	ϖ	ϵ^{h}	Variable	ϵ^{2h}	ϖ	ϵ^{h}
$M_B^{(0,0,0)}$	1.687×10^{-10}	4.44	7.790×10^{-12}	$M_B^{(3,0,1)}$	8.564×10^{-16}	8.20	2.906×10^{-18}
$M_B^{(1,0,0)}$	3.121×10^{-12}	6.66	3.089×10^{-14}	$M_B^{(0,1,1)}$	1.200×10^{-13}	6.00	1.873×10^{-15}
$M_B^{(2,0,0)}$	1.705×10^{-13}	7.46	9.697×10^{-16}	$M_B^{(1,1,1)}$	5.201×10^{-15}	8.10	1.896×10^{-17}
$M_B^{(3,0,0)}$	1.612×10^{-14}	7.94	6.575×10^{-17}	$M_B^{(2,1,1)}$	4.043×10^{-16}	7.47	2.285×10^{-18}
$M_B^{(4,0,0)}$	2.437×10^{-15}	8.02	9.407×10^{-18}	$M_B^{(0,2,1)}$	9.659×10^{-15}	7.22	6.470×10^{-17}
$M_B^{(0,1,0)}$	1.041×10^{-11}	4.97	3.315×10^{-13}	$M_B^{(1,2,1)}$	5.686×10^{-16}	8.06	2.132×10^{-18}
$M_B^{(1,1,0)}$	1.111×10^{-13}	6.32	1.386×10^{-15}	$M_B^{(0,3,1)}$	4.861×10^{-16}	7.11	3.521×10^{-18}
$M_B^{(2,1,0)}$	7.297×10^{-15}	7.93	2.997×10^{-17}	$M_B^{(0,0,2)}$	3.829×10^{-13}	7.54	2.057×10^{-15}
$M_B^{(3,1,0)}$	4.923×10^{-16}	7.59	2.561×10^{-18}	$M_B^{(1,0,2)}$	1.190×10^{-14}	7.96	4.786×10^{-17}
$M_B^{(0,2,0)}$	5.984×10^{-13}	5.69	1.157×10^{-14}	$M_B^{(2,0,2)}$	1.127×10^{-15}	8.13	4.031×10^{-18}
$M_B^{(1,2,0)}$	8.115×10^{-15}	7.41	4.779×10^{-17}	$M_B^{(0,1,2)}$	5.905×10^{-15}	7.82	2.606×10^{-17}
$M_B^{(2,2,0)}$	9.585×10^{-16}	7.80	4.315×10^{-18}	$M_B^{(1,1,2)}$	4.104×10^{-16}	7.85	1.778×10^{-18}
$M_B^{(0,3,0)}$	2.451×10^{-14}	6.54	2.637×10^{-16}	$M_B^{(0,2,2)}$	5.964×10^{-16}	7.71	2.856×10^{-18}
$M_B^{(1,3,0)}$	1.044×10^{-15}	7.87	4.472×10^{-18}	$M_B^{(0,0,3)}$	4.159×10^{-14}	7.63	2.096×10^{-16}
$M_B^{(0,4,0)}$	1.820×10^{-15}	7.70	8.776×10^{-18}	$M_B^{(1,0,3)}$	1.530×10^{-15}	7.71	7.299×10^{-18}
$M_B^{(0,0,1)}$	9.229×10^{-12}	7.76	4.251×10^{-14}	$M_B^{(0,1,3)}$	7.425×10^{-16}	7.76	3.423×10^{-18}
$M_B^{(1,0,1)}$	7.297×10^{-14}	7.99	2.868×10^{-16}	$M_B^{(0,0,4)}$	2.263×10^{-15}	7.54	1.216×10^{-17}
$M_B^{(2,0,1)}$	1.618×10^{-14}	7.80	7.266×10^{-17}				

Table 3. Embedded boundary moment convergence rates for $h = 1/128$ using the L_∞ norm in three dimensions. The implicit function is the ellipsoid described in Section 4.1.

5. Conclusions

Given a desired order of accuracy, we present analysis which shows the accuracy required for the geometric moments in the context of Cartesian grids with embedded

D	Norm	ϵ^{2h}	ϖ	ϵ^{h}
2	L_∞	8.839×10^{-6}	2.98	1.122×10^{-6}
2	L_1	1.512×10^{-7}	3.96	9.728×10^{-9}
2	L_2	7.424×10^{-7}	3.45	6.772×10^{-8}
3	L_∞	2.287×10^{-5}	2.98	2.898×10^{-6}
3	L_1	2.663×10^{-7}	3.98	1.685×10^{-8}
3	L_2	1.225×10^{-6}	3.49	1.094×10^{-7}

Table 4. Convergence results for the error in the divergence of an analytical flux ($D^K(F)$) (described in the text) in two and three dimensions for $h = 1/128$ (ϵ^h is the error for $h = 1/128$; ϵ^{2h} is the error for $h = 1/64$).

boundaries. We demonstrate using convergence tests that our moments are calculated to the expected order and that, for a known flux, these moments can be used to create a flux divergence of the expected order. This work provides the foundation for higher-order finite volume, embedded boundary methods.

References

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

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

[3] P. Colella, D. T. Graves, B. J. Keen, and D. Modiano, *A Cartesian grid embedded boundary method for hyperbolic conservation laws*, J. Comput. Phys. **211** (2006), no. 1, 347–366. MR 2006i:65142 Zbl 1120.65324

[4] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Modiano, D. B. Serafini, and B. Van Straalen, *Chombo software package for AMR applications: design document*, Technical Report LBNL-6616E, Lawrence Berkeley National Laboratory, 2014.

[5] P. Colella, D. T. Graves, T. J. Ligocki, G. Miller, D. Modiano, P. Schwartz, B. Van Straalen, J. Pillod, D. Trebotich, and M. Barad, *EBChombo software package for Cartesian grid, embedded boundary applications*, Technical Report LBNL-6615E, Lawrence Berkeley National Laboratory, 2014.

[6] Z. Dragojlovic, F. Najmabadi, and M. Day, *An embedded boundary method for viscous, conducting compressible flow*, J. Comput. Phys. **216** (2006), no. 1, 37–51. MR 2223435 Zbl 1173.76372

[7] C. Gonzalez-Ochoa, S. McCammon, and J. Peters, *Computing moments of objects enclosed by piecewise polynomial surfaces*, ACM Trans. Graph. **17** (1998), no. 3, 143–157.

[8] D. T. Graves, P. Colella, D. Modiano, J. Johnson, B. Sjogreen, and X. Gao, *A Cartesian grid embedded boundary method for the compressible Navier–Stokes equations*, Commun. Appl. Math. Comput. Sci. **8** (2013), no. 1, 99–122. MR 3143820 Zbl 1282.76006

[9] H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys. **147** (1998), no. 1, 60–85. MR 99m:65231 Zbl 0923.65079

[10] T. J. Ligocki, P. O. Schwartz, J. Percelay, and P. Colella, *Embedded boundary grid generation using the divergnce theorem, implicit functions, and constructive solid geometry*, SciDAC 2008 (Philadelphia), J. Phys. Conf. Ser., no. 125, IOP, 2008.

[11] S. Marella, S. Krishnan, H. Liu, and H. S. Udaykumar, *Sharp interface Cartesian grid method, I: An easily implemented technique for 3D moving boundary computations*, J. Comput. Phys. **210** (2005), no. 1, 1–31. MR 2006e:65161 Zbl 1154.76359

[12] G. H. Miller and D. Trebotich, *An embedded boundary method for the Navier–Stokes equations on a time-dependent domain*, Commun. Appl. Math. Comput. Sci. **7** (2012), no. 1, 1–31. MR 2893419 Zbl 1273.35215

[13] S. Molins, D. Trebotich, C. I. Steefel, and C. Shen, *An investigation of the effect of pore scale flow on average geochemical reaction rates using direct numerical simulation*, Water Resour. Res. **48** (2012), no. 3.

[14] A. Nonaka, D. Trebotich, G. Miller, D. Graves, and P. Colella, *A higher-order upwind method for viscoelastic flow*, Commun. Appl. Math. Comput. Sci. **4** (2009), no. 1, 57–83. MR 2010j:65138 Zbl 1166.76039

[15] R. R. Nourgaliev, M.-S. Liou, and T. G. Theofanous, *Numerical prediction of interfacial instabilities: sharp interface method (SIM)*, J. Comput. Phys. **227** (2008), no. 8, 3940–3970. MR 2009d:76054 Zbl 1275.76164

[16] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome, *Adaptive Cartesian grid methods for representing geometry in inviscid compressible flow*, 11th Computational Fluid Dynamics Conference (Orlando, 1993), American Institute of Aeronautics and Astronautics, 1993, pp. 948–958.

[17] P. Schwartz, M. Barad, P. Colella, and T. Ligocki, *A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions*, J. Comput. Phys. **211** (2006), no. 2, 531–550. MR 2006e:65194 Zbl 1086.65532

[18] R. Singh and W. Shyy, *Three-dimensional adaptive Cartesian grid method with conservative interface restructuring and reconstruction*, J. Comput. Phys. **224** (2007), no. 1, 150–167. MR 2008b:76124 Zbl 1248.76111

[19] M. Sussman and E. G. Puckett, *A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows*, J. Comput. Phys. **162** (2000), no. 2, 301–337. MR 2001c:76099 Zbl 0977.76071

[20] L. Yang, F. Albregtsen, and T. Taxt, *Fast computation of three-dimensional geometric moments using a discrete divergence theorem and a generalization to higher dimensions*, Graph. Model. Im. Proc. **59** (1997), no. 2, 97–108.

Received March 12, 2014. Revised July 11, 2014.

PETER SCHWARTZ: poschwartz@lbl.gov
Applied Numerical Algorithms Group, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

JULIE PERCELAY: julie.percelay@gmail.com
Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

TERRY J. LIGOCKI: tjligocki@lbl.gov
Applied Numerical Algorithms Group, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

HANS JOHANSEN: hjohansen@lbl.gov
Applied Numerical Algorithms Group, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

DANIEL T. GRAVES: dtgraves@lbl.gov
Applied Numerical Algorithms Group, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

DHARSHI DEVENDRAN: pdevendran@lbl.gov
Applied Numerical Algorithms Group, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

PHILLIP COLELLA: pcolella@lbl.gov
Applied Numerical Algorithms Group, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

ELI ATELJEVICH: eli.ateljevich@water.ca.gov
California Department of Water Resources, 1416 9th Street, Sacramento, CA 95821, United States

Guidelines for Authors

Authors may submit manuscripts in PDF format on-line at the Submission page at msp.org/camcos.

Originality. Submission of a manuscript acknowledges that the manuscript is original and and is not, in whole or in part, published or under consideration for publication elsewhere. It is understood also that the manuscript will not be submitted elsewhere while under consideration for publication in this journal.

Language. Articles in CAMCoS are usually in English, but articles written in other languages are welcome.

Required items. A brief abstract of about 150 words or less must be included. It should be self-contained and not make any reference to the bibliography. If the article is not in English, two versions of the abstract must be included, one in the language of the article and one in English. Also required are keywords and subject classifications for the article, and, for each author, postal address, affiliation (if appropriate), and email address.

Format. Authors are encouraged to use LaTeX but submissions in other varieties of TeX, and exceptionally in other formats, are acceptable. Initial uploads should be in PDF format; after the refereeing process we will ask you to submit all source material.

References. Bibliographical references should be complete, including article titles and page ranges. All references in the bibliography should be cited in the text. The use of BibTeX is preferred but not required. Tags will be converted to the house format, however, for submission you may use the format of your choice. Links will be provided to all literature with known web locations and authors are encouraged to provide their own links in addition to those supplied in the editorial process.

Figures. Figures must be of publication quality. After acceptance, you will need to submit the original source files in vector graphics format for all diagrams in your manuscript: vector EPS or vector PDF files are the most useful.

Most drawing and graphing packages (Mathematica, Adobe Illustrator, Corel Draw, MATLAB, etc.) allow the user to save files in one of these formats. Make sure that what you are saving is vector graphics and not a bitmap. If you need help, please write to graphics@msp.org with details about how your graphics were generated.

White space. Forced line breaks or page breaks should not be inserted in the document. There is no point in your trying to optimize line and page breaks in the original manuscript. The manuscript will be reformatted to use the journal's preferred fonts and layout.

Proofs. Page proofs will be made available to authors (or to the designated corresponding author) at a Web site in PDF format. Failure to acknowledge the receipt of proofs or to return corrections within the requested deadline may cause publication to be postponed.

Communications in Applied Mathematics and Computational Science

vol. 10 no. 1 2015