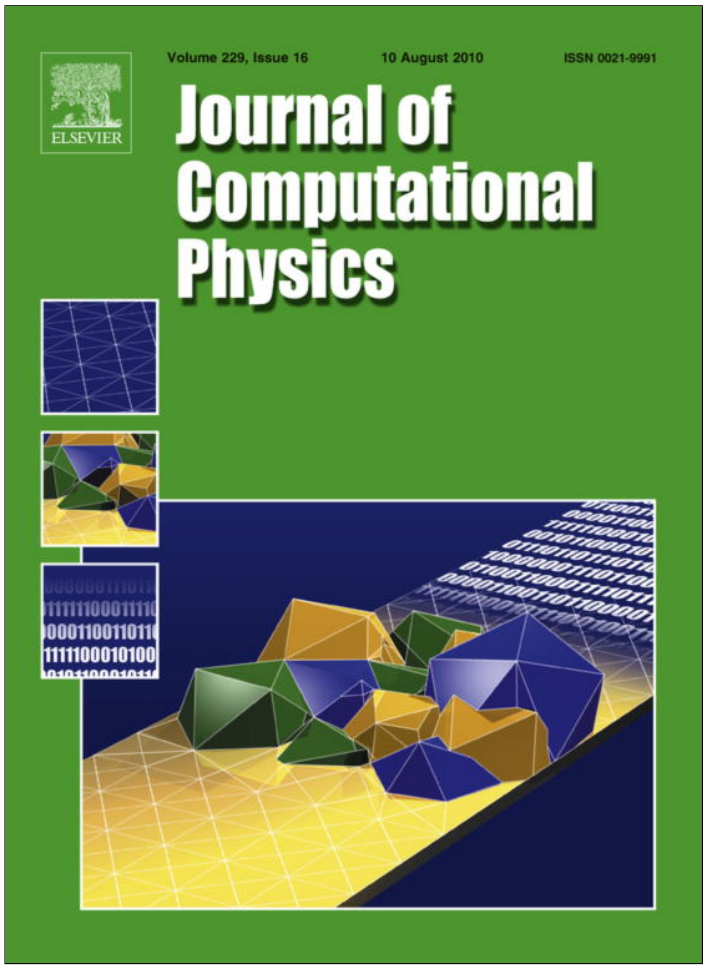

Journal of Computational Physics 229 (2010) 5692–5706

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Compact integration factor methods for complex domains and adaptive mesh refinement

Xinfeng Liu [a], Qing Nie [b,*]

[a] Department of Mathematics, University of South Carolina, Columbia, SC 29208, United States
[b] Department of Mathematics, University of California at Irvine, Irvine, CA 92697, United States

ARTICLE INFO

Article history:
Received 16 December 2009
Received in revised form 1 April 2010
Accepted 1 April 2010
Available online 13 April 2010

Keywords:
Semi-implicit methods
Integration factor method
Reaction–diffusion equations
Adaptive mesh refinement

ABSTRACT

Implicit integration factor (IIF) method, a class of efficient semi-implicit temporal scheme, was introduced recently for stiff reaction–diffusion equations. To reduce cost of IIF, compact implicit integration factor (cIIF) method was later developed for efficient storage and calculation of exponential matrices associated with the diffusion operators in two and three spatial dimensions for Cartesian coordinates with regular meshes. Unlike IIF, cIIF cannot be directly extended to other curvilinear coordinates, such as polar and spherical coordinates, due to the compact representation for the diffusion terms in cIIF. In this paper, we present a method to generalize cIIF for other curvilinear coordinates through examples of polar and spherical coordinates. The new cIIF method in polar and spherical coordinates has similar computational efficiency and stability properties as the cIIF in Cartesian coordinate. In addition, we present a method for integrating cIIF with adaptive mesh refinement (AMR) to take advantage of the excellent stability condition for cIIF. Because the second order cIIF is unconditionally stable, it allows large time steps for AMR, unlike a typical explicit temporal scheme whose time step is severely restricted by the smallest mesh size in the entire spatial domain. Finally, we apply those methods to simulating a cell signaling system described by a system of stiff reaction–diffusion equations in both two and three spatial dimensions using AMR, curvilinear and Cartesian coordinates. Excellent performance of the new methods is observed.

1. Introduction

Integration factor (IF) or exponentially differencing time (ETD) methods are widely used for solving temporal nonlinear partial differential equations (PDEs). In this approach, a linear part, which is often taken as the terms of high order derivatives of a nonlinear PDE, is exactly evaluated, leading to excellent temporal stability property (see [1] for review). One example is an application to advection–diffusion equations using exact treatment of their linear part, and the resulted explicit temporal scheme has stability properties similar to those of typical implicit schemes [2]. In general, IF methods and ETD methods are particularly efficient for systems with high order derivatives that requires small time step for a typical explicit temporal scheme [3–7].

For reaction–diffusion equations with stiff reactions, a class of semi-implicit integration factor (IIF) method was recently developed to deal with stiffness of the systems [8]. In IIF, the diffusion term is treated exactly such that the stability constraint due to spatial discretization for diffusion is removed while the reaction term is treated implicitly to handle the

* Corresponding author. Tel.: +1 949 824 5530; fax: +1 949 824 7993.
E-mail address: qnie@math.uci.edu (Q. Nie).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.04.003

stiffness of reaction terms. This leads to great stability property for IIF while the extra cost associated with the implicit treatment of reactions is minimal because of decoupling between calculations of diffusion and reactions in IIF. In particular, its second order scheme is linearly unconditionally stable and the high order scheme has large stability regions.

While the IF and ETD (explicit or implicit) schemes improve the stability conditions, the schemes require storage of exponentials of matrices resulted from discretization of the linear differential operators in the PDEs. Although the discretization matrices are sparse, their exponentials are not. For two and three spatial dimensions, the storage sometimes may be prohibitive computationally. To overcome this difficulty, recently compact representation of the discretization matrices was introduced in the context of implicit integration factor method [9]. This compact implicit integration factor (cIIF) has the same stability properties as the original IIF [8] but with significant improvement on storages and CPU savings.

In cIIF, the discretized solutions is represented in a form of matrix rather than a vector in IIF while the discretized diffusion operator are represented in matrices whose size is much smaller than the standard sparse matrices for diffusion operators [9]. Such compact representation of diffusion operators, as demonstrated in terms of a Cartesian coordinate for Laplacian operators in [9], is essential for derivation of compact IIF that has tremendous advantage in computational efficiency and storage over standard IIF in high spatial dimensions. However, it is not clear how cIIF can be directly applied to systems in other curvilinear coordinates, such as polar, spherical or cylinder coordinates that are often convenient for representing systems in non-rectangular geometries.

In this study, we generalize cIIF methods for Cartesian coordinate to other curvilinear coordinate systems using examples of polar and spherical coordinates. In this approach, the matrices derived from a compact representation of diffusion operator in non-Cartesian coordinate need to be diagonalized once and to be pre-calculated before the solution is updated at each time step. The new cIIF for polar and spherical coordinates is found to have similar stability properties as the cIIF for the Cartesian coordinate along with a similar computational cost.

One natural application of cIIF methods in high spatial dimensions and in different coordinates is for temporal PDEs incorporated with spatially adaptive techniques. As we know, when solving nonlinear PDEs with sharp gradients in localized spatial domain, a local mesh refinement technique is superior to the uniform grid approach because one can cluster the spatial grid points in regions where needed. As a result, a smaller number of total spatial grid points are used for adaptive methods compared to the uniform grid, leading to computational and storage savings in spatial discretization.

However, the temporal updating for typical explicit schemes on spatial adaptive approach still requires the time step be restricted by the smallest spatial mesh size. For example, for reaction–diffusion equations, the time step must be proportional to square of the smallest spatial mesh size. As a result, the time step for the entire spatial domain is dictated by the finest mesh [10] and the saving gained from spatial adaptive mesh refinement is compensated by the extra costs associated with the temporal schemes. One possible solution for this is to use multirate time integration schemes in which the time step can vary across the spatial domain [11–14]. Another alternative is to use IF or ETD methods that have better stability conditions. In particular, unconditionally stable schemes which allow large time steps independent of the spatial grid size, such as the second order cIIF (cIIF2), may be ideal for solving temporal solutions of nonlinear PDEs that requires adaptivity in spatial discretization.

In this paper, we apply cIIF2 to reaction–diffusion equations using a block-structured adaptive mesh refinement (AMR) algorithm [15] for both two and three spatial dimensions. The AMR, which is based on Cartesian meshes and overlapping grids, was previously used for simulations in fluids [16–23], materials [24,25], and heart tissues [26–28]. In this approach, a hierarchy of refinement grids is constructed dynamically based on a suitable error estimate of the solution, and then the composite overlapping and structured grids are used to approximate the space domain [16,29–31]. Similar to other typical temporal schemes for AMR, in our approach the solution at all grids, coarse and fine, are updated from the current step independently. Then the solutions at the coarse grids that are in common with the fine grids are replaced by the solution at the corresponding fine grids to form the overall updated solutions. Numerical examples show that cIIF2 with AMR allowing for uniform large time steps is superior to the explicit temporal schemes which require much smaller time steps due to stability constraints. To ease the programming complexity in implementation, we also use many capabilities of the *Overture* object-oriented class library [32,33].

Finally, we integrate the cIIF2 in polar coordinates for two spatial dimensions with AMR and integrate cIIF2 in Cartesian coordinates with AMR in three spatial dimension for a biological application in cell signaling. The integrated scheme is found to be robust and efficient.

This paper is organized as follows. The generalization of cIIF for systems in polar coordinate and the corresponding numerical examples are presented in Section 2; the cIIF in spherical coordinate is given in Section 3; the cIIF in combination with AMR is introduced in Section 4; and a biological application using cIIF and AMR in polar and Cartesian coordinates is presented in Section 5.

2. Compact implicit integration factor (cIIF) method in polar coordinate

2.1. Derivation

In this section, we derive cIIF method for a two-dimensional reaction–diffusion equation in an annular domain using a polar coordinate. The system with no-flux boundary conditions for both directions takes a form

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = D\left(\frac{\partial^2 \mathbf{u}}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2 \mathbf{u}}{\partial \theta^2} + \frac{1}{r}\frac{\partial \mathbf{u}}{\partial r}\right) + \mathbf{F}(\mathbf{u}), \quad (r,\theta) \in \Omega = \{a < r < b,\ c < \theta < d\}; \\ \frac{\partial \mathbf{u}}{\partial r}(a,\theta) = \frac{\partial \mathbf{u}}{\partial r}(b,\theta) = \frac{\partial \mathbf{u}}{\partial \theta}(r,c) = \frac{\partial \mathbf{u}}{\partial \theta}(r,d) = 0. \end{cases} \tag{1}$$

After discretizing the spatial domain by a rectangular mesh: $(r_i, \theta_j) = (a + (i-1)h_r, c + (j-1)h_\theta)$ where $h_r = (b-a)/(N-1)$, $h_\theta = (d-c)/(M-1)$, $r_i = (i-1)h_r$ and $1 \leqslant i \leqslant N$ and $1 \leqslant j \leqslant M$, we use the second order central difference discretization for the diffusion terms:

$$\frac{du_{i,j}}{dt} = D\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_r^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{r_i^2 h_\theta^2} + \frac{u_{i+1,j} - u_{i-1,j}}{2r_i h_r}\right) + \mathbf{F}(u_{i,j}). \tag{2}$$

To express (2) in a compact form, we define the matrix $\mathbf{U}$ for the discretized solutions:

$$\mathbf{U} = \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,M} & u_{1,M} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,M} & u_{2,M} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{N,1} & u_{N,2} & \cdots & u_{N,M} & u_{N,M} \end{pmatrix}_{N\times M},$$

and

$$\mathbf{G}_{P\times P} = \begin{pmatrix} -2 & 2 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ & & \ddots & \ddots & \ddots & \\ 0 & 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 2 & -2 \end{pmatrix}_{P\times P},$$

$$\mathbf{E}_{P\times P} = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ -1/r_2 & 0 & 1/r_2 & 0 & \cdots & 0 \\ 0 & -1/r_3 & 0 & 1/r_3 & & 0 \\ & & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & -1/r_{P-1} & 0 & 1/r_{P-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}_{P\times P},$$

$$\mathbf{F}_{P\times P} = \begin{pmatrix} 1/r_1^2 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1/r_2^2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1/r_3^2 & 0 & \cdots & 0 \\ & & \ddots & \ddots & \ddots & \\ 0 & 0 & \cdots & 0 & 1/r_{P-1}^2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1/r_P^2 \end{pmatrix}_{P\times P}.$$

After defining $\mathbf{A_1} = \frac{D}{h_r^2}\mathbf{G}_{N\times N}, \mathbf{B} = \frac{D}{h_\theta^2}\mathbf{G}_{M\times M}, \mathbf{A_2} = \frac{D}{2h_r}\mathbf{E}_{N\times N}, \mathbf{C} = \mathbf{F}_{N\times N}$, and $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$, we re-write the semi-discretized form (2) as

$$\frac{d\mathbf{U}}{dt} = \mathbf{AU} + \mathbf{CUB} + \mathcal{F}(\mathbf{U}). \tag{3}$$

Assume $\mathbf{B}$ is diagonalizable: $\mathbf{B} = \mathbf{P}\widetilde{\mathbf{D}}\mathbf{P}^{-1}$, where $\widetilde{\mathbf{D}}$ is a diagonal matrix with the eigenvalues of $\mathbf{B}$ as the diagonal elements. Let $\mathbf{V} = \mathbf{UP}$, then (3) can be written as

$$\frac{d\mathbf{V}}{dt} = \mathbf{AV} + \mathbf{CV}\widetilde{\mathbf{D}} + \mathcal{F}(\mathbf{VP}^{-1})\mathbf{P}. \tag{4}$$

Notice that both matrices $\mathbf{C}$ and $\widetilde{\mathbf{D}}$ are diagonal, with the ith and jth diagonal elements in $\mathbf{C}$ and $\widetilde{\mathbf{D}}$ being c_i and d_j, respectively. Define $\widetilde{\mathbf{C}} = (\tilde{c}_{ij}) = (c_i d_j)$, where $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, M$, and an operation '(e^*)' by taking exponentials of a matrix element by element,

$$(e^*)^{\widetilde{\mathbf{C}}} = (e^{\tilde{c}_{ij}}).$$

Define another operator '$\odot$' for element by element multiplication between two matrices:

$$(\mathbf{H} \odot \mathbf{L})_{i,j} = (h_{ij} l_{ij}),$$

where $\mathbf{H} = (h_{ij})$, $\mathbf{L} = (l_{ij})$.

Then, direct calculations on (4) leads to

$$\frac{d(e^{-\mathbf{A}t}\mathbf{V}\odot(e^*)^{-\widetilde{\mathbf{C}}t})}{dt}=e^{-\mathbf{A}t}\mathcal{F}(\mathbf{V}\mathbf{P}^{-1})\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}t}. \tag{5}$$

Integration of (5) over one time step from t_n to $t_{n+1}\equiv t_n+\Delta t$, where Δt is the time step, results in

$$\mathbf{V}_{n+1}=e^{\mathbf{A}\Delta t}\mathbf{V}_{\mathbf{n}}\odot(e^*)^{\widetilde{\mathbf{C}}\Delta t}+e^{\mathbf{A}t_{n+1}}\left(\int_0^{\Delta t}e^{-\mathbf{A}\tau}\mathcal{F}(\mathbf{V}(\mathbf{t_n}+\tau)\mathbf{P}^{-1})\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}\tau}d\tau\right)\odot(e^*)^{\widetilde{\mathbf{C}}t_{n+1}}. \tag{6}$$

Since $\mathbf{V}=\mathbf{UP}$, then the equation for $\mathbf{U}$ can be recovered from (6):

$$\mathbf{U}_{n+1}=e^{-\mathbf{A}\Delta t}\mathbf{U_n}\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}\Delta t}\mathbf{P}^{-1}+e^{\mathbf{A}\Delta t}\left(\int_0^{\Delta t}e^{-\mathbf{A}\tau}\mathcal{F}(\mathbf{U}(\mathbf{t_n}+\tau))\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}\tau}d\tau\right)\odot(e^*)^{\widetilde{\mathbf{C}}\Delta t}\mathbf{P}^{-1}. \tag{7}$$

To construct a scheme of rth order truncation error, we define

$$\mathcal{G}(\tau)\equiv e^{-\mathbf{A}\tau}\mathcal{F}(\mathbf{U}(t_n+\tau))\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}\tau}, \tag{8}$$

and approximate $\mathcal{G}(\tau)$ using a $(r-1)$th order Lagrange polynomial, $\mathcal{P}(\tau)$, at a set of interpolation points $t_{n+1}, t_n,\ldots,t_{n+2-r}$:

$$\mathcal{P}(\tau)=\sum_{i=-1}^{r-2}e^{i\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n-i})\mathbf{P}\odot(e^*)^{i\widetilde{\mathbf{C}}\Delta t}\prod_{\substack{j=-1\\ j\neq i}}^{r-2}\frac{\tau+j\Delta t}{(j-i)\Delta t}. \tag{9}$$

The second, third and fourth order approximations to $\mathcal{G}(\tau)$ are listed as the following.

1. Given $\mathcal{G}(0)=\mathcal{F}(\mathbf{U}_n)\mathbf{P}, \mathcal{G}(\Delta t)=e^{-\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n+1})\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}\Delta t}$, the second order approximation to $\mathcal{G}(\tau)$ is
$$\mathcal{P}(\tau)=\frac{1}{\Delta t}\left[\mathcal{F}(\mathbf{U}_n)\mathbf{P}(\Delta t-\tau)+e^{-\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n+1})\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}\Delta t}\tau\right],\quad 0\leqslant\tau\leqslant\Delta t.$$
2. Given $\mathcal{G}(-\Delta t)=e^{\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n-1})\mathbf{P}\odot(e^*)^{\widetilde{\mathbf{C}}\Delta t}, \mathcal{G}(0)=\mathcal{F}(\mathbf{U}_n)\mathbf{P}, \mathcal{G}(\Delta t)=e^{-\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n+1})\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}\Delta t}$, the third order approximation to $\mathcal{G}(\tau)$ is
$$\mathcal{P}(\tau)=\frac{1}{2\Delta t^2}\left[e^{\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n-1})\mathbf{P}\odot(e^*)^{\widetilde{\mathbf{C}}\Delta t}\tau(\tau-\Delta t)-2\mathcal{F}(\mathbf{U}_n)\mathbf{P}(\tau+\Delta t)(\tau-\Delta t)+e^{-\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n+1})\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}\Delta t}\tau(\tau+\Delta t)\right],$$
$$0\leqslant\tau\leqslant\Delta t.$$
3. Given $\mathcal{G}(-2\Delta t)=e^{2\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n-2})\mathbf{P}\odot(e^*)^{2\widetilde{\mathbf{C}}\Delta t}, \mathcal{G}(-\Delta t)=e^{\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n-1})\mathbf{P}\odot(e^*)^{\widetilde{\mathbf{C}}\Delta t}, \mathcal{G}(0)=\mathcal{F}(\mathbf{U}_n), \mathcal{G}(\Delta t)=e^{-\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n+1})\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}\Delta t}$, the fourth order approximation to $\mathcal{G}(\tau)$ is
$$\mathcal{P}(\tau)=\frac{1}{6\Delta t^3}\Big[-e^{2\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n-2})\mathbf{P}\odot(e^*)^{2\widetilde{\mathbf{C}}\Delta t}(\tau-\Delta t)\tau(\tau+\Delta t)+3e^{\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n-1})P\odot(e^*)^{\widetilde{\mathbf{C}}\Delta t}(\tau-\Delta t)\tau(\tau+2\Delta t)$$
$$-3\mathcal{F}(\mathbf{U}_n)\mathbf{P}(\tau-\Delta t)(\tau+\Delta t)(\tau+2\Delta t)+e^{-\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n+1})\mathbf{P}\odot(e^*)^{-\widetilde{\mathbf{C}}\Delta t}\tau(\tau+\Delta t)(\tau+2\Delta t)\Big],\quad 0\leqslant\tau\leqslant\Delta t.$$

In terms of $\mathcal{P}(\tau)$, (5) takes the form,

$$\mathbf{U}_{n+1}=e^{\mathbf{A}\Delta t}\mathbf{U}_n\mathbf{P}\odot(e^*)^{\widetilde{\mathbf{C}}\Delta t}\mathbf{P}^{-1}+e^{\mathbf{A}\Delta t}\left(\int_0^{\Delta t}\mathcal{P}(\tau)d\tau\right)\odot(e^*)^{\widetilde{\mathbf{C}}\Delta t}\mathbf{P}^{-1}. \tag{10}$$

So the new rth order implicit schemes are

$$\mathbf{U}_{n+1}=e^{\mathbf{A}\Delta t}\mathbf{U}_n\mathbf{P}\odot(e^*)^{\widetilde{\mathbf{C}}\Delta t}\mathbf{P}^{-1}+\Delta t\left(\alpha_{n+1}\mathcal{F}(U_{n+1})+\sum_{i=0}^{r-2}\alpha_{n-i}e^{(i+1)\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n-i})\mathbf{P}\odot(e^*)^{(i+1)\widetilde{\mathbf{C}}\Delta t}\mathbf{P}^{-1}\right), \tag{11}$$

where $\alpha_{n+1}, \alpha_n, \alpha_{n-1},\ldots,\alpha_{n-r+2}$ are coefficients calculated from integrals of the polynomial in $\mathcal{P}(\tau)$,

$$\alpha_{n-i}=\frac{1}{\Delta t}\int_0^{\Delta t}\prod_{\substack{j=-1\\ j\neq i}}^{r-2}\frac{\tau+j\Delta t}{(j-i)\Delta t}d\tau,\quad -1\leqslant i\leqslant r-2. \tag{12}$$

In Table 1, the value of coefficients, α_j, for the schemes with order up to four are listed.

Table 1
Coefficients for clIF schemes of order one, two, three and four.

Order	α_{n+1}	α_n	α_{n-1}	α_{n-2}
1	1	0	0	0
2	$\frac{1}{2}$	$\frac{1}{2}$	0	0
3	$\frac{5}{12}$	$\frac{2}{3}$	$-\frac{1}{12}$	0
4	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$

In particular, the second order approximation of $\int_0^{\Delta t}\mathcal{G}(\tau)d\tau$ is

$$\int_0^{\Delta t}\mathcal{G}(\tau)d\tau \approx \frac{\mathcal{F}(\mathbf{U}_n)P + e^{-\mathbf{A}\Delta t}\mathcal{F}(\mathbf{U}_{n+1})P \odot (e^*)^{-\widetilde{\mathbf{C}}\Delta t}}{2}\Delta t, \tag{13}$$

leading to the second order IIF scheme

$$\mathbf{U}_{n+1} = e^{\mathbf{A}\Delta t}\left(\mathbf{U_n} + \frac{\Delta t}{2}\mathcal{F}(\mathbf{U_n})\right)\widetilde{\mathbf{B}} + \frac{\Delta t}{2}\mathcal{F}(\mathbf{U_{n+1}}), \tag{14}$$

where $\widetilde{\mathbf{B}} = \mathbf{P}\odot(e^*)^{\widetilde{\mathbf{C}}\Delta t}\mathbf{P}^{-1}$.

Like the one-dimensional form [8], the nonlinear reaction term at t_{n+1} in (14) is decoupled from the diffusion term. As a result, only a local nonlinear system needs to be solved at each spatial grid point. The two matrices $e^{\mathbf{A}\Delta t}$ and $(e^*)^{\mathbf{C}\Delta t}$ have a size of $N\times N$ and $N\times M$, respectively, similar to the compact integration factor method in a Cartesian coordinate system [9].

Remark 1. If system (1) has Dirichlet boundary condition (s) in the r or (and) θ direction (s), for instance, in θ direction:

$$\mathbf{u}(r,c) = f(r), \quad \mathbf{u}(r,d) = g(r),$$

the solution matrix $\mathbf{U}$ becomes

$$\mathbf{U} = \begin{pmatrix} u_{1,2} & u_{1,3} & \cdots & u_{1,M-2} & u_{1,M-1} \\ u_{2,2} & u_{2,3} & \cdots & u_{2,M-2} & u_{2,M-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{N,2} & u_{N,3} & \cdots & u_{N,M-2} & u_{N,M-1} \end{pmatrix}_{N\times(M-2)},$$

and the corresponding matrix $\mathbf{B}$ becomes

$$\mathbf{B} = \frac{D}{h_\theta^2}\begin{pmatrix} -2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ & & \ddots & \ddots & \ddots & \\ 0 & 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 1 & -2 \end{pmatrix}_{(M-2)\times(M-2)}.$$

All other three matrices $\mathbf{A_1}$, $\mathbf{A_2}$, $\mathbf{C}$ remain the same. The semi-discretized form (3) then becomes

$$\frac{d\mathbf{U}}{dt} = \mathbf{AU} + \mathbf{CUB} + \mathcal{F}(\mathbf{U}) + \mathbf{CS}, \tag{15}$$

where

$$\mathbf{S} = \frac{D}{h_\theta^2}\begin{pmatrix} f(a) & 0 & \cdots & 0 & g(a) \\ f(a+\frac{b-a}{N-1}) & 0 & \cdots & 0 & g(a+\frac{b-a}{N-1}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f(b) & 0 & \cdots & 0 & g(b) \end{pmatrix}_{N\times(M-2)}.$$

By following a similar analysis, the second order scheme similar to (14) becomes

$$\mathbf{U}_{n+1} = e^{\mathbf{A}\Delta t}\left(\mathbf{U_n} + \frac{\Delta t}{2}\mathcal{F}(\mathbf{U_n})\right)\widetilde{\mathbf{B}} + \frac{\Delta t}{2}\mathcal{F}(\mathbf{U_{n+1}}) + \frac{\Delta t}{2}(e^{\mathbf{A}\Delta t}\mathbf{CS}\widetilde{\mathbf{B}} + \mathbf{CS}). \tag{16}$$

Remark 2. Since the corresponding matrix $\mathbf{B}$ in (15) for the Dirichlet boundary in the θ direction is real symmetric positive definite (SPD), the matrices $\mathbf{P}$ and $\widetilde{\mathbf{D}}$ can be calculated analytically. Let $\eta = \pi/(M-1)$, the eigenvalues of $\mathbf{B}$ are

$$\mu_j = -\frac{2D}{h_\theta^2}(1-\cos(j\eta)) = -\frac{4D}{h_\theta^2}\sin^2(j\eta/2), \quad j = 1,2,\ldots,M-2,$$

and the corresponding eigenvectors are

$$w_j = [\sin(j\eta), \sin(2j\eta), \ldots, \sin((M-2)j\eta)]^T = [\sin(jk\eta)]_{k=1}^{M-2}.$$

Then $\widetilde{\mathbf{P}} = [w_1, w_2, \ldots, w_{M-2}]$. One can then apply Gram–Schmidt process to obtain an orthogonal matrix $\mathbf{P}$, i.e. $\mathbf{P}^{-1} = \mathbf{P}^T$.

For the case when matrix $\mathbf{B}$ is in a form as depicted in (3) for the no-flux boundary conditions in θ direction, the analytical form of $\mathbf{P}$ and $\widetilde{\mathbf{D}}$ can be calculated similarly. Denote $\eta = \pi/(M-1)$, the eigenvalues of $\mathbf{B}$ are

$$\mu_j = -\frac{D}{h_\theta^2}(1-\cos(j\eta)) = -\frac{4D}{h_\theta^2}\sin^2(j\eta/2), \quad j = 1,2,\ldots,M-2,\ \mu_{M-1} = -\frac{4D}{h_\theta^2},\ \mu_M = 0,$$

and the first $M-2$ corresponding eigenvectors are

$$w_j = [1, \cos(j\eta), \cos(2j\eta), \ldots, \cos((M-2)j\eta), (-1)^j]^T, \quad j = 1,2,\ldots,M-2.$$

The other two eigenvectors corresponding to $\mu_{M-1} = -\frac{4D}{h_\theta^2}$ and $\mu_M = 0$ are

$$w_{M-1} = [1,-1,\ldots,(-1)^{M+1}]^T = [(-1)^{k+1}]_{k=1}^{k=M}, \quad w_M = [1,1,\ldots,1]^T.$$

So $\widetilde{\mathbf{P}} = [w_1, w_2, \ldots, w_M]$ and a Gram–Schmidt process leads to an orthogonal matrix $\mathbf{P}$, i.e. $\mathbf{P}^{-1} = \mathbf{P}^T$.

For other cases in which the eigenvectors and eigenvalues of the corresponding matrix B cannot be calculated analytically, one can pre-calculate those matrices, for example, using the function "eig" in Matlab. Because those matrices never change during the temporal updating if a fixed time-step is used, one needs to compute them once and store them. Many numerical methods are available for diagonalization of matrices [34,35]. The "eig" function in Matlab is based on LAPACK [36]. In principle, as long as the errors for calculating the eigenvectors and eigenvalues are significantly smaller than the spatial and temporal discretization errors, the calculations of such eigenvector and eigenvalue should not affect the order of accuracy of the methods. This is observed in our simulations shown below. The CPU time for diagonalization of a matrix with dimensions 320×320, which is a typical size we use, is 0.25 s when using "eig". Since this portion of CPU time, as shown in the following sections, is significantly smaller than the overall CPU time for the entire temporal evolution of the simulation, we do not include this portion of the CPU time in the tables shown below.

Remark 3. In comparison with the compact implicit integration factor (cIIF) methods in Cartesian coordinates in two dimensions, here we list the corresponding discretization form in space [9]:

$$\frac{d\mathbf{U}}{dt} = \mathbf{AU} + \mathbf{UB} + \mathcal{F}(\mathbf{U}); \tag{17}$$

and the corresponding second order cIIF method:

$$\mathbf{U}_{n+1} = e^{\mathbf{A}\Delta t}\left(\mathbf{U}_n + \frac{\Delta t}{2}\mathcal{F}(\mathbf{U}_n)\right)e^{\mathbf{B}\Delta t} + \frac{\Delta t}{2}\mathcal{F}(\mathbf{U}_{n+1}). \tag{18}$$

From the preceding derivation of cIIF methods in polar coordinates, we see diagonalization of matrix $\mathbf{B}$ as in Eq. (3) in general is required and the extension from Cartesian to polar coordinates needs extra care and calculations.

2.2. Numerical test

To test the accuracy and efficiency of the cIIF scheme (16) in polar coordinate, we study the following system of the polar coordinate:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = 0.1\left(\frac{\partial^2 \mathbf{u}}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2 \mathbf{u}}{\partial \theta^2} + \frac{1}{r}\frac{\partial \mathbf{u}}{\partial r}\right) + 0.2\mathbf{u}, \quad (x,y) \in \Omega = \{5 < r < 10,\ 0 < \theta < \pi\}; \\ \mathbf{u}(r,\theta)|_{r=5} = e^{0.1t}\cos(5\cos\theta)\cos(5\sin\theta); \\ \mathbf{u}(r,\theta)|_{r=10} = e^{0.1t}\cos(10\cos\theta)\cos(10\sin\theta); \\ \mathbf{u}(r,\theta)|_{\theta=0} = e^{0.1t}\cos(r); \quad \mathbf{u}(r,\theta)|_{\theta=\pi} = e^{0.1t}\cos(r). \end{cases} \tag{19}$$

This system (19) has an exact solution $\mathbf{u}(r,\theta) = e^{0.1t}\cos(r\cos\theta)\cos(r\sin\theta)$.

The maximal error of solution is measured by the maximal difference between the numerical solution and the exact solution, denoted by L^∞ error $= \max_{i,j}|\mathbf{U}_{i,j}^{\text{num}} - \mathbf{U}_{i,j}^{\text{exact}}|$. The simulation is carried up to time $t = 2$, and the number of grid points for r and θ is set equal to each other for convenience of comparison. The time-step size is chosen to be $\Delta t = 0.5h_r$ for checking order of accuracy. As seen in Table 2, the simulation using the scheme (16) is of order two and the time step is not constrained like a typical explicit temporal scheme. The stability property and computational cost of the scheme (16) is similar to the standard cIIF2 [9].

Table 2
Error, order of accuracy, and CPU time for cIIF2 in polar coordinate of (19). The unit for CPU time is second.

$N \times N$	L^∞ error	Order	CPU
40×40	4.58×10^{-3}	–	0.23
80×80	1.23×10^{-3}	1.90	0.52
160×160	3.1×10^{-4}	1.99	2.41
320×320	8.23×10^{-5}	1.91	64.24

3. cIIF in spherical coordinate

3.1. Derivation

We now derive the cIIF in spherical coordinate. Without loss of generality, we consider a system with no flux Neumann boundary conditions in each direction:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = D\left(\frac{\partial^2 \mathbf{u}}{\partial r^2} + \frac{2}{r}\frac{\partial \mathbf{u}}{\partial r} + \frac{1}{r^2 \sin^2\phi}\frac{\partial^2 \mathbf{u}}{\partial \theta^2} + \frac{\cos\phi}{r^2 \sin\phi}\frac{\partial \mathbf{u}}{\partial \phi} + \frac{1}{r^2}\frac{\partial^2 \mathbf{u}}{\partial \phi^2}\right) + \mathbf{F}(\mathbf{u}), \\ \frac{\partial}{\partial r}(a,\theta,\phi) = \frac{\partial}{\partial r}(b,\theta,\phi) = 0, \\ \frac{\partial}{\partial \theta}(r,c,\phi) = \frac{\partial}{\partial \theta}(r,d,\phi) = 0, \\ \frac{\partial}{\partial \phi}(r,\theta,e) = \frac{\partial}{\partial \phi}(r,\theta,f) = 0, \end{cases} \tag{20}$$

where $(x, y) \in \Omega = \{a < r < b,\ c < \theta < d,\ e < \phi < f\}$. Similar to the two-dimensional system (1) in polar coordinate, we denote h_r, h_θ, h_ϕ as the spatial step size, and N_r, N_θ, N_ϕ as the number of grid points in r, θ, ϕ direction, respectively. After applying the second order central difference discretization on the diffusion, we obtain a system of nonlinear ODEs as the following,

$$\begin{aligned} \frac{du_{i,j,k}}{dt} = D\Bigg(&\frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{h_r^2} + \frac{u_{i+1,j,k} - u_{i-1,j,k}}{r_i^2 h_r} + \frac{u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}}{r_i^2 \sin^2\phi_k}\frac{\cos\phi_k(u_{i,j,k+1} - u_{i,j,k-1})}{2r_i^2 \sin\phi_k h_\phi} \\ &+ \frac{u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}}{r_i^2 h_\phi^2}\Bigg) + \mathbf{F}(u_{i,j,k}). \end{aligned} \tag{21}$$

Next we define $\mathbf{A_1} = \frac{D}{h_r^2}\mathbf{G}_{N_r \times N_r}, \mathbf{A_2} = \frac{D}{h_r}\mathbf{E}_{N_r \times N_r}, \mathbf{A_r} = \mathbf{F}_{N_r \times N_r}, \mathbf{B} = \frac{D}{h_\theta^2}\mathbf{G}_{N_\theta \times N_\theta}, \mathbf{C_1} = \frac{D}{h_\phi^2}\mathbf{G}_{N_\phi \times N_\phi}$, where **G**, **E**, **F** are defined in the previous section and

$$\mathbf{C}_\phi = \begin{pmatrix} 1/\sin^2\phi_1 & 0 & & \cdots & 0 & 0 \\ 0 & 1/\sin^2\phi_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1/\sin^2\phi_3 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ & & \cdots & 0 & 1/\sin^2\phi_{N_\phi - 1} & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1/\sin^2\phi_{N_\phi} \end{pmatrix}_{N_\phi \times N_\phi}$$

and

$$\mathbf{C_2} = \frac{D}{2h_\phi} \cdot \begin{pmatrix} 0 & 0 & & \cdots & & 0 \\ \frac{-\cos\phi_2}{\sin\phi_2} & 0 & \frac{\cos\phi_2}{\cos\phi_2} & & & 0 \\ 0 & \frac{-\cos\phi_3}{\sin\phi_3} & 0 & \frac{\cos\phi_3}{\sin\phi_3} & & 0 \\ 0 & & \ddots & \ddots & \ddots & \\ & & & \frac{-\cos\phi_{N_\phi - 1}}{\sin\phi_{N_\phi - 1}} & 0 & \frac{\cos\phi_{N_\phi - 1}}{\cos\phi_{N_\phi - 1}} \\ 0 & \cdots & & & 0 & 0 \end{pmatrix}_{N_\phi \times N_\phi}.$$

Denoting $\mathbf{U} = (u_{i,j,k})$ and defining $\mathbf{A} = \mathbf{A_1} + \mathbf{A_2}$ and $\mathbf{C} = \mathbf{C_1} + \mathbf{C_2}$. We write (20) in the following compact representation,

$$\mathbf{U}_t = \mathbf{A} \circledcirc_r \mathbf{U} + \mathbf{A_r} \circledcirc_r \mathbf{B} \circledcirc_\theta \mathbf{C}_\phi \circledcirc_\phi \mathbf{U} + \mathbf{A_r} \circledcirc_r \mathbf{C} \circledcirc_\phi \mathbf{U} + \mathcal{F}(\mathbf{U}), \tag{22}$$

Table 3
Error, order of accuracy, and CPU time for cIIF2 in spherical coordinate of (26). The unit for CPU time is second.

$N \times N \times N$	L^∞ error	Order	CPU
$20 \times 20 \times 20$	9.26×10^{-3}	–	0.56
$40 \times 40 \times 40$	2.43×10^{-3}	1.93	12.3
$80 \times 80 \times 80$	6.27×10^{-4}	1.95	80.5

where

$$\begin{aligned}(A \circledcirc U) &= \left(\sum_{l=1}^{N_r}(A)_{i,l}u_{l,j,k}\right),\\ (B \circledcirc U) &= \left(\sum_{l=1}^{N_\theta}(B)_{j,l}u_{i,l,k}\right),\\ (C \circledcirc U) &= \left(\sum_{l=1}^{N_\phi}(C)_{k,l}u_{i,j,l}\right).\end{aligned} \tag{23}$$

Assuming both matrices $\mathbf{B}$, $\mathbf{C}$ can be diagonalized, we have $\mathbf{B} = \mathbf{P_1 D_1 P_1^{-1}}, \mathbf{C} = \mathbf{P_2 D_2 P_2^{-1}}$, where $\mathbf{D}_1$ and $\mathbf{D}_2$ are diagonal matrices. Let $\mathbf{V} = \mathbf{P_1} \circledcirc \mathbf{P_2} \circledcirc \mathbf{U}$, then (22) becomes

$$\mathbf{V}_t = \mathbf{A} \circledcirc \mathbf{V} + \mathbf{A_r} \circledcirc \mathbf{D_1} \circledcirc \mathbf{C}_\phi \circledcirc \mathbf{V} + \mathbf{A_r} \circledcirc \mathbf{D_2} \circledcirc \mathbf{V} + \mathbf{P_1} \circledcirc \mathbf{P_2} \circledcirc \mathcal{F}(\mathbf{P_1^{-1}} \circledcirc \mathbf{P_2^{-1}} \circledcirc V). \tag{24}$$

Denote a_i, d_{1i}, c_i, d_{2i} as the ith diagonal element of matrices $\mathbf{A_r}$, $\mathbf{D}_1$, $\mathbf{C}_\phi$, $\mathbf{D}_2$, respectively, and define $(\widetilde{\mathbf{D}}_\mathbf{1})_{i,j,k} = (a_i d_{1j} c_k), (\widetilde{\mathbf{D}}_\mathbf{2})_{ij} = (a_i d_{2j})$. After defining a new operation similar to the system in polar coordinate:

$$((e^*)^{\widetilde{\mathbf{D}}_1} \odot \mathbf{V})_{i,j,k} = (e^{a_i d_{1j} c_k} v_{i,j,k});$$

and

$$((e^*)^{\widetilde{\mathbf{D}}_2} \odot V)_{i,j,k} = (e^{a_i d_{2j}} v_{i,j,k});$$

we obtain the second order scheme for (24) using a similar approach to the scheme (16) for polar coordinate,

$$\mathbf{U}_{n+1} = \mathbf{P_1^{-1}} \circledcirc \mathbf{P_2^{-1}} \circledcirc e^{\mathbf{A}\Delta t} \circledcirc (e^*)^{\widetilde{\mathbf{D}}_1 \Delta t} \odot (e^*)^{\widetilde{\mathbf{D}}_2 \Delta t} \odot \mathbf{P_1} \circledcirc \mathbf{P_2} \circledcirc \left(\mathbf{U_n} + \frac{\Delta t}{2}\mathbf{F}(\mathbf{U_n})\right) + \frac{\Delta t}{2}\mathbf{F}(\mathbf{U_{n+1}}). \tag{25}$$

The higher order schemes can be derived similarly. The approach for deriving the cIIF scheme in polar or spherical coordinate can be easily extended to systems in cylindrical coordinate.

3.2. Numerical test

We test the numerical scheme (25) using the following simple system in spherical coordinates,

$$\frac{\partial \mathbf{u}}{\partial t} = 0.1\left(\frac{\partial^2 \mathbf{u}}{\partial r^2} + \frac{2}{r}\frac{\partial \mathbf{u}}{\partial r} + \frac{1}{r^2 \sin^2\phi}\frac{\partial^2 \mathbf{u}}{\partial \theta^2} + \frac{\cos\phi}{r^2 \sin\phi}\frac{\partial \mathbf{u}}{\partial \phi} + \frac{1}{r^2}\frac{\partial^2 \mathbf{u}}{\partial \phi^2}\right) + 0.2\mathbf{u}, \tag{26}$$

with the similar Dirichlet boundary conditions as that of system (19). The exact solution of (26) has a similar form as that of (19). The solution is calculated up to $t = 2$ and we choose $h_r = h_\theta = h_\phi$ and $\Delta t = 0.5 h_r$ for convenience of comparison. Similar to the two-dimensional case in polar coordinate, the scheme (25) exhibits the second order accuracy and it is very efficient for moderate size of N, as seen in Table 3.

4. Compact implicit integration factor methods with adaptive mesh refinement

4.1. Adaptive mesh refinement (AMR)

Here, we use a block-structured Cartesian mesh refinement technique developed for hyperbolic conservation laws [10,15]. In this approach, finite difference Cartesian meshes are refined by adding overlapped finer meshes in desired spatial locations where the solutions exhibit sharp gradients (see Fig. 1 for an illustration). The refinement grids are aligned with the underlying base grid and are arranged in a hierarchy with the base grids belonging to level $l = 0$, the next finer grids being added to level $l = 1$ and so on. Grids on level l are refined by a refinement ratio n_r from the grids on level $l - 1$. The grids are properly nested so that a grid on level l is completely contained in the set of grids on the coarser level $l - 1$. This requirement is relaxed at physical boundaries to allow refinement grids to align with the boundary.

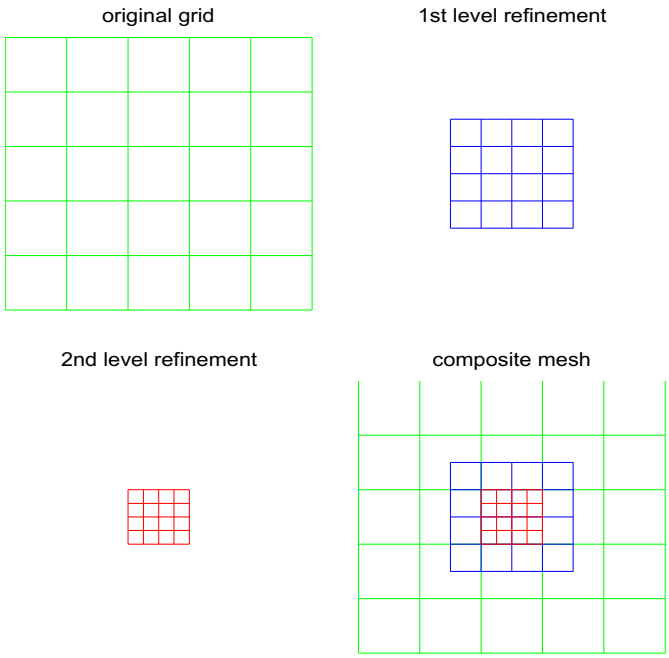

Fig. 1. An example of overlapping block-structured Cartesian mesh refinement. Two levels of refinement are applied to the local region.

In addition to re-gridding, the other two major steps in a AMR algorithm are the error estimation (or indicator) to determine where to refine the meshes and to interpolate solutions at the added grids. The error indicator is based on estimated magnitudes of the first and second derivatives in the numerical solution using finite difference approximation [10], and the region of refinement is determined by tagging cells where the error estimate exceeds a prescribed tolerance. In general, solution values on the new grid are interpolated from the solutions on the coarse grid. A basic AMR interpolation operation involve interpolation of solution values at new grid from the finest level grid available on the old grid hierarchy, interpolation at ghost points in the buffer zone of refinement grids, and interpolation of coarse grid points that are hidden by refinement grids [15]. In addition, each boundary face of each component grid block treated as either a physical boundary (where boundary condition is prescribed), a periodic boundary or an interpolation boundary. Typically, one or more lines of ghost points are created for each component grid block to aid in the application of boundary conditions. We use periodic boundary conditions in the buffer zones for the overlapping grid blocks.

The implementation of grid generation is handled by object-oriented programming using *Ogen* [15]. A programming flow chart for solving a time-dependent PDE using AMR at every re-gridding steps, defined as $n_{\text{re-grid}}$, along with a temporal scheme, named as **timeStep**(), can be written as the following [10]:

$$
\begin{aligned}
&\text{PDEsolve}(\mathcal{G}, t_{\text{final}})\\
&\{\\
&t = 0;\ \ n = 0;\\
&\mathbf{u}_i^n = \text{applyInitialCondition}(\mathcal{G})\\
&\text{while } t < t_{\text{final}}\\
&\quad \text{if}(n \bmod \ \ n_{\text{regrid}} == 0)\\
&\quad e_i = \text{estimateError}(\mathcal{G}, \mathbf{u}_i^n);\\
&\quad \mathcal{G}^* = \text{regrid}(\mathcal{G}, e_i);\\
&\quad \mathbf{u}_i^* = \text{interpolateToNewGrid}(\mathbf{u}_i^n, \mathcal{G}, \mathcal{G}^*);\\
&\quad \mathcal{G} = \mathcal{G}^*; \mathbf{u}_i^n = \mathbf{u}_i^*;\\
&\quad \text{end}\\
&\mathbf{u}_i^{n+1} = \text{timeStep}(\mathcal{G}, \mathbf{u}_i^n, \Delta t);\\
&t = t + \Delta t;\ \ n = n + 1;\\
&\text{interpolate}(\mathcal{G}, \mathbf{u}_i^n);\\
&\text{applyBoundaryConditions}(\mathcal{G}, \mathbf{u}_i^n, t);\\
&\text{end}\\
&\}
\end{aligned}
\tag{27}
$$

4.2. cIIF incorporated with AMR

We use reaction–diffusion equations in two spatial dimensions with a Cartesian coordinate and cIIF2 of second order accuracy as an example to illustrate integration of AMR and compact integration factor method [9]. As seen in (27), after the solution values at the newly added grids are interpolated from the solutions at the coarse grids at the current time step, the solutions at the next time step are updated from the solution at the current time step at each grid level independently. The solutions at the intersection grids between the coarse and fine grids are always using the solutions at the fine grids after updating. The operation "**timeStep**()" in (27) representing the step for temporal updating in AMR is where the cIIF2 is incorporated.

Because cIIF2 requires repeated use of exponentials of diffusion matrices, we calculate those matrices once for each grid block with every grid level that are needed for the entire simulation, and store them. Specifically, at the ith composite grid level and its kth grid block, we evaluate and store the exponential of the diffusion matrix $e^{G(\Delta x_{i_k}, N_{i_k})\Delta t}$ where N_{i_k} and Δx_{i_k} is the number of grid points and the grid size in x-direction of each grid block, respectively. We use a similar approach to handle $e^{G(\Delta y_{i_k}, M_{i_k})\Delta t}$ in the y direction, with M_{i_k} representing the number of points in y direction and Δy_{i_k} standing for the grid size in y direction.

So at each time step, the storage cost for cIIF2 is of order $\sum_k \sum_i M_{i_k} N_{i_k}$, which is of the same order of the number of grid points in the entire domain. The operation count is of order $\sum_k \sum_i (M_{i_k}^2 N_{i_k} + M_{i_k} N_{i_k}^2)$ because the dominant cost of cIIF2 method is the vector–matrix multiplication associated with exponential of matrices. The overall computational complexity is in the same order of cIIF for uniform meshes [9]. In contrast to a typical explicit temporal scheme integrated with AMR for which the time step must be constrained by the finest spatial grid size, the time step for cIIF2 with AMR is not restricted by the size of spatial grid at all in terms of stability because cIIF2 is linearly unconditionally stable.

The same approach in combining AMR with cIIF2 can be directly applied to systems in polar coordinate as well as three-dimensional systems in spherical, cylindrical or Cartesian coordinate.

4.3. Numerical tests

A linear reaction–diffusion equations with an exact solution is tested:

$$\begin{cases} \frac{\partial u}{\partial t} = a\Delta u - 2a \times (\tanh(x))^2 u, & (x,y,z) \in \Omega, \\ \mathbf{n} \cdot \nabla u = 0, & (x,y,z) \in \partial\Omega, \end{cases} \tag{28}$$

where $\Omega = \{-10 < x < 10,\ 0 < y < \pi,\ 0 < z < \pi\}$. The analytically approximated solution of (28) takes a form

$$u(x,y,z,t) = e^{-at} \tanh(x) \cos(y) \cos(z). \tag{29}$$

This solution satisfies Eq. (28) and boundary conditions in y and z directions exactly, and it approximates the boundary condition in x direction with $\frac{\partial u}{\partial x} \approx 10^{-9}$ at $x = -10, 10$, which is much smaller than the error in simulations. We also study a corresponding two-dimensional case of (28) by omitting the z variable.

In all simulations, the diffusion coefficient is $a = 0.1$ and the solutions are studied at final time $t = 1$. The differences between the numerical solution and the analytical solution are measured by both maximum error at all grid points, denoted by L^∞ error, and an average error, denoted by L^2, for the case of a polar coordinate system, L^2 error $= \sqrt{\sum (\mathbf{U}_{i,j}^{\text{num}} - \mathbf{U}_{i,j}^{\text{exact}})^2 r_i h_r h_\theta}$. For other curvilinear coordinate systems, such as spherical coordinates, the L^2 error is defined similarly. The solution has a sharp gradient around $x = 0$ where the mesh is refined using AMR based on the error indicator given in [10]. The ratio of each refinement level is chosen to be two for all testing cases.

First, we compare cIIF2 with a uniform grid at four different spatial resolutions with the cIIF2 with AMR using four levels of local refinement (see Tables 4 and 5). For both two and three-dimensional cases, the coarsest grid size and the finest grid size are chosen to be the same for either the uniform grid or AMR. As expected, the errors using AMR are comparable with the errors using uniform grid when the uniform mesh size is the same as the local refinement mesh size, because the sharp gradient in the solution is localized around $x = 0$. However, the CPU cost for the AMR is much cheaper than the uniform grid with the finest grids. For the two dimensional case, AMR is cheaper than the uniform grid in the second finest mesh size

Table 4
cIIF2 with uniform grids and cIIF2 with AMR in two dimensions. In AMR, the coarsest grid is 20×20. $\Delta t = 0.005$ for all cases.

Method	$N \times N$	L^∞ error	L^2 error	CPU (s)
cIIF2 (uniform)	20×20	5.6×10^{-2}	4.8×10^{-2}	4.5
	40×40	1.4×10^{-2}	1.2×10^{-2}	12.3
	80×80	3.4×10^{-3}	2.9×10^{-3}	67.8
	160×160	8.4×10^{-4}	7.4×10^{-4}	357.9
cIIF2 (AMR)	4-levels	8.4×10^{-4}	7.4×10^{-4}	40.3

Table 5
cIIF2 with uniform grids and cIIF2 with AMR in three dimensions. In AMR, the coarsest grid is $10 \times 10 \times 10$. $\Delta t = 0.01$ for all cases.

Method	$N \times N \times N$	L^∞	L^2	CPU (s)
cIIF2 (uniform)	$10 \times 10 \times 10$	3.6×10^{-1}	2.9×10^{-1}	5.2
	$20 \times 20 \times 20$	9.2×10^{-2}	7×10^{-2}	45.2
	$40 \times 40 \times 40$	2.4×10^{-2}	1.4×10^{-2}	350.1
	$80 \times 80 \times 80$	5.6×10^{-3}	3.3×10^{-3}	2798.7
cIIF2 (AMR)	4-levels	5.6×10^{-3}	3.3×10^{-3}	378.2

while for the three-dimensional case, AMR is only slightly more expensive than the uniform grid in the second finest mesh size.

Next, we compare cIIF2 and Runge–Kutta both in second order using AMR by varying the level of refinement. We study three, four, and five levels of local mesh refinement in a two-dimensional system using both cIIF and Runge–Kutta method. As seen in Tables 6–8, Runge–Kutta method converges only when the time-step size is sufficiently small to satisfy stability constraint due to the smallest spatial grid size. In Runge–Kutta method, more levels of refinement, leading to smaller size in the finest grid, requires smaller time-step size for convergence. However, cIIF2 converges at all levels of refinement for large

Table 6
Comparison between cIIF2(AMR) and RK2(AMR) for three-level of local mesh refinement.

Δt	cIIF2(20(40(80))		RK2(20(40(80))			
	L^∞	L^2	CPU (s)	L^∞ error	L^2 error	CPU (s)
10^{-2}	1.2×10^{-2}	8×10^{-3}	13.2	–	–	–
5×10^{-3}	3.4×10^{-3}	2.1×10^{-3}	28.5	–	–	–
2.5×10^{-3}	8.9×10^{-4}	5×10^{-4}	67.6	8.8×10^{-4}	4.9×10^{-4}	62.9

Table 7
Comparison between cIIF2(AMR) and RK2(AMR) for four-level of local mesh refinement.

Δt	cIIF2(20(40(80(160)))		RK2(20(40(80(160)))			
	L^∞	L^2	CPU (s)	L^∞ error	L^2 error	CPU (s)
10^{-2}	3.1×10^{-3}	2.8×10^{-3}	19.2	–	–	–
5×10^{-3}	8.5×10^{-4}	7.2×10^{-4}	40.3	–	–	–
2.5×10^{-3}	2.2×10^{-4}	1.8×10^{-4}	89.5	–	–	–
1.25×10^{-3}	5.3×10^{-5}	4.4×10^{-5}	192.1	–	–	–
6.25×10^{-4}	1.5×10^{-5}	1.1×10^{-5}	405.2	1.5×10^{-5}	1.1×10^{-5}	369.2

Table 8
Comparison between cIIF2(AMR) and RK2(AMR) for five-level of local mesh refinement.

Δt	cIIF2(20(40(80(160(320))))		RK2(20(40(80(160(320))))			
	L^∞	L^2	CPU (s)	L^∞ error	L^2 error	CPU (s)
10^{-2}	1.1×10^{-3}	8×10^{-4}	22.3	–	–	–
5×10^{-3}	2.7×10^{-4}	2×10^{-4}	50.2	–	–	–
2.5×10^{-3}	5.7×10^{-5}	5×10^{-5}	110.5	–	–	–
1.25×10^{-3}	1.6×10^{-5}	1.2×10^{-5}	212.4	–	–	–
6.25×10^{-4}	3.1×10^{-6}	2.8×10^{-6}	445.8	–	–	–
3.125×10^{-4}	7.2×10^{-7}	6.7×10^{-7}	921.7	–	–	–
1.5625×10^{-4}	2.0×10^{-7}	1.5×10^{-7}	1932.8	1.9×10^{-7}	1.5×10^{-7}	1625.7

Table 9
Comparison between cIIF2(AMR) and RK2(AMR) for three-level of local mesh refinement in a three-dimensional case.

Δt	cIIF2(10(20(40))		RK2(10(20(40))			
	L^∞	L^2	CPU (s)	L^∞ error	L^2 error	CPU (s)
10^{-2}	2.6×10^{-2}	2.2×10^{-2}	241.5	–	–	–
5×10^{-3}	5.2×10^{-3}	5.1×10^{-3}	483.9	–	–	–
2.5×10^{-3}	1.4×10^{-3}	1.3×10^{-3}	970.5	1.4×10^{-3}	1.3×10^{-3}	921.3

time steps with good accuracy. For the time-step size in which both cIIF2 and Runge–Kutta converge, both methods achieve similar accuracy and use similar amount of CPU time. Similar results are also obtained when cIIF2 and Runge–Kutta method using AMR are tested on the three-dimensional system (see Table 9).

5. An example in cell biology

In this section, we integrate cIIF2 in polar (2D) and spherical (3D) coordinates with AMR for two-dimensional and three-dimensional models describing intracellular dynamics of chemical reactions of several diffusive species within a cell [37].

When a hormone or growth factor binds to a cell-surface receptor, a cascade of proteins inside the cell relays the signal to specific intracellular targets. A class of proteins referred as scaffolds are thought to play important roles during this process [38–40]. Scaffold usually dynamically binds to two or more consecutively-acting components of a signaling cascade. Experimental work suggests that scaffolds may promote signal transmission by tethering consecutively-acting kinases near each other [41,42]. However, it has also been experimentally observed that some scaffold inhibit signaling when overexpressed [43–45]. Supporting these observations, computations of non-spatial models have demonstrated that scaffold proteins may either enhance or suppress signaling, depending on the concentration of scaffold. In [37], a model of generic, spatially localized scaffold protein was developed for one spatial dimension and the simulations suggested that a scaffold protein could boost signaling locally (in and near the region where it was localized) while simultaneously suppressing signaling at a distance.

Here we present simulations for the corresponding two and three dimensional models [37]. The system contains a scaffold protein (S), which can bind to two other proteins (A and B). In the absence of scaffold protein, A and B can bind directly to each other. In the presence of the scaffold protein S, A first binds to S, forming AS; B binds to AS, forming ASB. In addition, A and B bind to each other on the scaffold and an AB complex can be released from the scaffold; and a symmetrical path, where B binds to the scaffold before A, is also available. Denote [ ] as the concentration of the proteins, the mass reaction equations with diffusion take the form,

$$\begin{aligned}
\frac{d[S]}{dt} &= -j_{on}([A][S]+[B][S])+j_{off}([AS]+[BS])+j_{con}[ABS],\\
\frac{d[AS]}{dt} &= j_{on}([A][S]-[AS][B])-j_{off}([AS]-[ABS]),\\
\frac{d[BS]}{dt} &= j_{on}([B][S]-[BS][A])-j_{off}([BS]-[ABS]),\\
\frac{d[ABS]}{dt} &= j_{on}([AS][B]+[BS][A])-(2j_{off}+j_{con})[ABS],\\
\frac{d[A]}{dt} &= D\Delta[A]-k_{on}[A][B]+k_{off}[AB]-j_{on}([A][S]+[BS][A])+j_{off}([AS]+[ABS]),\\
\frac{d[B]}{dt} &= D\Delta[B]-k_{on}[A][B]+k_{off}[AB]-j_{on}([B][S]+[AS][B])+j_{off}([BS]+[ABS]),\\
\frac{d[AB]}{dt} &= D\Delta[AB]+k_{on}[A][B]-k_{off}[AB]+j_{con}[ABS].
\end{aligned} \tag{30}$$

In the system (30), D is the diffusion constant for A, B and AB; k_{on}, k_{off} are the on and off rates for the off-scaffold reactions, j_{on}, j_{off}, j_{con} are the rate constants for the on-scaffold reactions.

First, we consider a two-dimensional model assuming that the cell is a two-dimensional disk: $\Omega = \{0 \leqslant x^2 + y^2 \leqslant 100\ \mu m\}$, with no-flux boundary conditions for A, B, AB. In this simulation, the initial concentrations for A and B are 1 μM and uniformly distributed throughout the cell. And the scaffolds initially are localized in part of the cell: $\{9 \leqslant r \leqslant 10;\ 2\pi/5 \leqslant \theta \leqslant 3\pi/5\}$ where r, θ are polar coordinates. The total S is set initially at 50 μM and uniformly distributed in this region. The values for all other parameters can be found in the caption of Fig. 2, which shows contour plots of the biological desirable

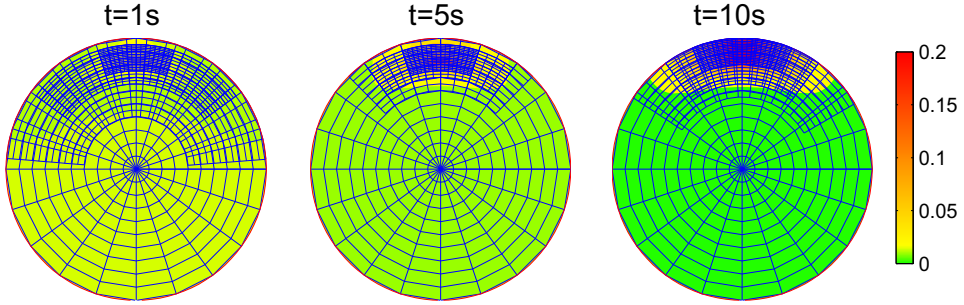

Fig. 2. Contour plots for concentration of AB in the system (30) calculated using cIIF2 with AMR. The parameters for the simulation are $D = 1\ \mu m^2\ s^{-1}$, $k_{on} = 0.1\ (\mu M\ s)^{-1}$, $k_{off} = 0.3\ s^{-1}$, $j_{on} = 1\ (\mu M\ s)^{-1}$, $j_{off} = 0.005\ s^{-1}$, $j_{con} = 0.1\ (\mu M\ s)^{-1}$.

Table 10
Spatial resolution study for the system (30) in polar coordinate using cIIF2 in uniform grids and cIIF2 in AMR. All solutions are evaluated at $T = 10$ s and $\Delta t = 10^{-2}$ is used for all cases.

Method	$N \times N$	L^∞ error	L^2 error	CPU (s)
cIIF2	20×20	8.7×10^{-1}	6.8×10^{-1}	5.4
	40×40	2.2×10^{-1}	1.8×10^{-1}	16.8
	80×80	5.5×10^{-2}	4.5×10^{-2}	72.1
	160×160	1.4×10^{-2}	1.1×10^{-2}	324.1
cIIF2 (AMR)	4-levels	1.4×10^{-2}	1.2×10^{-2}	42.2

Table 11
Spatial resolution study for the system (30) using cIIF2 in uniform grids and cIIF2 in AMR for a three-dimensional case. All solutions are evaluated at $T = 1$ s and $\Delta t = 10^{-2}$ is used for all cases.

Method	$N \times N \times N$	L^∞ error	L^2 error	CPU (s)
cIIF2	$10 \times 10 \times 10$	9.9×10^{-1}	7.7×10^{-1}	11.2
	$20 \times 20 \times 20$	2.5×10^{-1}	1.9×10^{-1}	89.9
	$40 \times 40 \times 40$	5.1×10^{-2}	4.9×10^{-2}	723.3
	$80 \times 80 \times 80$	1.3×10^{-2}	1.2×10^{-2}	5867.4
cIIF2 (AMR)	4-levels	1.4×10^{-2}	1.3×10^{-2}	482.3

product protein *AB* at three different times. The simulations suggest that a localized scaffold could boost signaling (product formation) near the region where the scaffold is localized while simultaneously suppressing signaling at a distance in a two-dimensional cell, which is consistent with the one dimensional findings [37].

Next, we check accuracy of our implementation by a numerical resolution study. The simulation with a global uniform fine grids: 640×640 along with a relatively small time step $\Delta t = 10^{-5}$, is considered as an "exact" solution. All errors are calculated based on difference between the numerical solutions and the "exact" solutions. For the simulation using AMR, an error indicator based on magnitudes of first and second derivatives [10] is applied for re-gridding to determine where to refine the local meshes. The region of refinement is determined by tagging cells in which the error indicator exceeds a specified tolerance [10,15], which is chosen to be 10^{-3} for this case. In Table 10, the errors for concentration of product protein *AB* are estimated at $T = 10$ s for different spatial resolutions with a fixed time-step size. A second order accuracy in space is observed without adjusting the time step, mainly due to the nice stability condition of cIIF2. In the meantime, cIIF2 with AMR based on four levels of refinement with the finest grid equivalent to $N = 160$ around the scaffold area (see Fig. 2) can achieve similar accuracy as the uniform case with the finest grid $N = 160$. However, the CPU time for cIIF2 with AMR is only about 1/8 of cIIF2 with uniform grids.

Finally, we present a three-dimensional study for the system (30) within a cubic cell: $\Omega = \{0 \leqslant x \leqslant 10\ \mu\text{m}, 0 \leqslant y \leqslant 10\ \mu\text{m}, 0 \leqslant z \leqslant 10\ \mu\text{m}\}$. The initial concentrations of *A* and *B* are chosen at 1 μM, uniformly distributed throughout the cubic cell, and the scaffolds *S* is localized in a ball within the cell: $4\ \mu\text{m}^2 \leqslant x^2 + y^2 + z^2 \leqslant 9\ \mu\text{m}^2$, with a total amount of 50 μM scaffold. The scaffold are initially uniformly distributed throughout the prescribed ball inside the cell. All other parameters are chosen to be the same as the two-dimensional case in Fig. 2. To perform the resolution study, we use a solution calculated based on a global uniform grid $160 \times 160 \times 160$ and a time step 10^{-4} as an "exact" solution. The dynamical local mesh refinement is also applied using the same procedure as for the two dimensional case. In Table 11 we see a similar second order accuracy in space for cIIF2 with uniform grids. For the three-dimensional case, to acquire similar accuracy, cIIF2 with AMR using four levels of refinement with the finest grid equivalent to $N = 80$ only needs 1/12 of CPU time of the cIIF2 in the corresponding finest uniform grids ($N = 80$). Clearly, cIIF2 with AMR is superior to cIIF2 in uniform grids.

6. Conclusions and discussion

Implicit integration factor (IIF), which treats diffusion term explicitly and reaction term implicitly, is particularly efficient for stiff reaction diffusion systems [8]. IIF can be easily implemented in different coordinate systems including polar and spherical coordinate systems. In two and three spatial dimensions, compact implicit integration factor (cIIF) was found to be more efficient than IIF due to its compact representation for the diffusion operators [9]. However, unlike IIF, it is difficult to apply cIIF *directly* to non-Cartesian coordinate systems, such as polar or spherical coordinates. In this paper, we have presented a method that allows cIIF to be incorporated into any curvilinear coordinates. In particular, we have developed a class of cIIF in polar (2D) and spherical (3D) coordinates, in which the stability condition and computational cost are similar to the original cIIF in a Cartesian system [9].

Because cIIF is semi-implicit and has large stability region (for example, the second order of cIIF is linearly unconditionally stable), this class of schemes is particularly suitable for solving reaction diffusion equations using adaptive mesh refinement (AMR), which cluster spatial grids around locations with sharp spatial gradients of solutions. When explicit temporal

schemes are applied to AMR, a small number of local fine grids usually restrict the overall time-step size due to stability constraint. In this paper, we have integrated cIIF with AMR for stiff reaction–diffusion systems by taking advantage of the excellent stability property of cIIF. Numerical examples have shown application of second order cIIF, a linearly unconditionally stable scheme, could allow large time steps for small local spatial grids in AMR. From the point of view of stability, the time-step size has been found to be independent of the spatial grid size, as expected. In general, cIIF exhibits great advantages over explicit temporal schemes when AMR is used for spatial discretization of reaction–diffusion equation.

Many physical and biological applications often involve complex geometries, which are more suitable using non-Cartesian coordinates, and sharp gradients in solutions, which may require local mesh refinement. To deal with those challenges, we have presented a method that combines cIIF in polar and spherical coordinates with AMR for spatial discretization of reaction–diffusion equations. This approach has been then used for studying how localized scaffold protein might boost or inhibit signaling in a cell. The overall numerical method has been observed to be robust and efficient in solving systems that consist of multiple diffusive species with stiff reactions in both two and three dimensions. Although the methods presented in this paper are mainly for cIIF2, the same approach can be directly applied to high order cIIF and other ETD methods [2,6,7,3]. In addition, the presented approach based on the finite difference framework for spatial discretization may also be extended for method using fast Fourier transformation [46,47].

Acknowledgments

This work was partially supported by the NSF Grants DMS0511169 and DMS0917492 and NIH Grants R01GM75309, R01GM67247 and NIH P50GM76516.

References

[1] A.-K. Kassam, L.N. Trefethen, Fourth-order time stepping for stiff PDEs, SIAM Journal on Scientific Computing 26 (2005) 1214–1233.
[2] G. Beylkin, J.M. Keiser, L. Vozovoi, A new class of time discretization schemes for the solution of nonlinear PDEs, Journal of Computational Physics 147 (1998) 362–387.
[3] T.Y. Hou, J. Lowengrub, M.J. Shelley, Removing the stiffness from interfacial flows with surface tension, Journal of Computational Physics 114 (1994) 312.
[4] P.H. Leo, J.S. Lowengrub, Q. Nie, Microstructural evolution in orthotropic elastic media, Journal of Computational Physics 157 (2000) 44–88.
[5] H.J. Jou, P.H. Leo, J.S. Lowengrub, Microstructual evolution in inhomogeneous elastic media, Journal of Computational Physics 131 (1997) 109.
[6] Q. Du, W. Zhu, Stability analysis and applications of the exponential time differencing schemes, Journal of Computational Mathematics 22 (2004) 200.
[7] Q. Du, W. Zhu, Modified exponential time differencing schemes: analysis and applications, BIT, Numerical Mathematics 45 (2005) 307–328.
[8] Q. Nie, Y.-T. Zhang, R. Zhao, Efficient semi-implicit schemes for stiff systems, Journal of Computational Physics 214 (2006) 521–537.
[9] Q. Nie, F.Y.M. Wan, Y.-T. Zhang, X.F. Liu, Integration factor methods for high spatial dimensions, Journal of Computational Physics 227 (2008) 5238–5255.
[10] W.D. Henshaw, D.W. Schwendeman, An adaptive numerical method for high-speed reactive flows on overlapping grids, Journal of Computational Physics 191 (2003) 420–447.
[11] C. Dawson, R. Kirby, High resolution schemes for conservation laws with locally varying time steps, SIAM Journal on Scientific Computing 22 (6) (2001) 2256–2284.
[12] R. Kirby, On the convergence of high resolution methods with multiple time scales for hyperbolic conservation laws, Mathematics of Computation 72 (243) (2003) 1239–1250.
[13] H. Tang, G. Warnecke, A Class of High Resolution Schemes for Hyperbolic Conservation Laws and Convection Diffusion Equations with Varying Time and Space Grids, Univ., Fak. für Mathematik, 2003.
[14] E.M. Constantinescu, A. Sandu, Multirate timestepping methods for hyperbolic conservation laws, Journal of Scientific Computing 33 (3) (2007) 239–278.
[15] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, Journal of Computational Physics 53 (1984) 484–512.
[16] G. Chesshire, W.D. Henshaw, Composite overlapping meshes for the solution of partial differential equations, Journal of Computational Physics 90 (1) (1990) 1–64.
[17] J.L. Steger, J.A. Benek, On the use of composite grid schemes in computational aerodynamics, Computer Methods in Applied Mechanics and Engineering 64 (1-3) (1987) 301–320.
[18] W.D. Henshaw, G. Chesshire, Multigrid on composite meshes, SIAM Journal on Scientific and Statistical Computing 8 (1987) 914.
[19] M. Hinatsu, J.H. Ferziger, Numerical computation of unsteady incompressible flow in complex geometry using a composite multigrid technique, International Journal for Numerical Methods in Fluids 13 (8) (1991).
[20] R.L. Meakin, Moving Body Overset Grid Methods for Complete Aircraft Tiltrotor Simulations, AIAA Paper, 1993, pp. 93–3350.
[21] J.Y. Tu, L. Fuchs, Calculation of flows using three-dimensional overlapping grids and multigrid methods, International Journal for Numerical Methods in Engineering 38 (2) (1995). ISSN 0029-5981.
[22] N.A. Petersson, Hole-cutting for three-dimensional overlapping grids, SIAM Journal on Scientific Computing 21 (2) (1999) 646–665.
[23] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, Journal of Computational Physics 82 (1) (1989) 64–84.
[24] S. Torabi, S. Wise, J. Lowengrub, A. Ratz, A. Voigt, A new method for simulating strongly anisotropic Cahn-Hilliard equations, Materials Science and Technology-Association for Iron and Steel Technology 3 (2007) 1432.
[25] H.G. Lee, J. Lowengrub, J. Goodman, Modeling pinchoff and reconnection in a Hele-Shaw cell. II. Analysis and simulation in the nonlinear regime, Physics of Fluids 14 (2002) 514.
[26] E.M. Cherry, H.S. Greenside, C.S. Henriquez, A space-time adaptive method for simulating complex cardiac dynamics, Physical Review Letters 84 (6) (2000) 1343–1346.
[27] E.M. Cherry, H.S. Greenside, C.S. Henriquez, Efficient simulation of three-dimensional anisotropic cardiac tissue using an adaptive mesh refinement method, Chaos: An Interdisciplinary Journal of Nonlinear Science 13 (2003) 853.
[28] J.A. Trangenstein, C. Kim, Operator splitting and adaptive mesh refinement for the Luo–Rudy I model, Journal of Computational Physics 196 (2) (2004) 645–679.
[29] K. Brislawn, D.L. Brown, G. Chesshire, J. Saltzman, Adpative-refined Overlapping Grids for the Numerical Solution of Hyperbolic Systems of Conservation Laws, Report LA-UR-95-257, Los Alamos National Laboratory, 1995.
[30] E.P. Boden, E.F. Toro, A combined Chimera-AMR technique for computing hyperbolic pdes, in: Djilali (Ed.), Proceedings of the Fifth Annual Conference of the CFD Society of Canada, 1997, pp. 5.13–5.18.

[31] R.L. Meankin, Composite overset structured grids, in: J.F. Thompson, B.K. Soni, N.P. Weatherill (Eds.), Handbook of Grid Generation, CRC Press, 1999, pp. 1–20 (Chapter 11).
[32] D.L. Brown, W.D. Henshaw, D.J. Quinlan, Overture: an object-oriented framework for solving partial differential equations, Lecture Notes in Computer Science (1997) 177–184.
[33] D.L. Brown, G.S. Chesshire, W.D. Henshaw, D.J. Quinlan, Overture: an object oriented software system for solving partial differential equations in serial and parallel environments, in: Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, 1997.
[34] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, 1990.
[35] G.L.G. Sleijpen, H.A. Van der Vorst, A Jacobi–Davidson iteration method for linear eigenvalue problems, SIAM Review 42 (2) (2000) 267–293.
[36] E. Anderson, Z. Bai, C. Bischof, LAPACK Users' Guide, Society for Industrial Mathematics, 1999.
[37] L. Bardwell, X.F. Liu, R.D. Moore, Q. Nie, Spatially-localized Scaffold Proteins May Simultaneously Boost and Suppress Signaling, Preprint, 2009.
[38] A.J. Whitmarsh, R.J. Davis, Structural organization of MAP-kinase signaling modules by scaffold proteins in yeast and mammals, Trends in Biochemical Sciences 23 (1998) 481–485.
[39] D.K. Morrison, R.J. Davis, Regulation of MAP kinase signaling modules by scaffold proteins in mammals, Annual Review of Cell and Developmental Biology 19 (2003) 91–118.
[40] W. Wong, J.D. Scott, AKAP signalling complexes: focal points in space and time, Nature Reviews Molecular Cell Biology 5 (2004) 959–970.
[41] S.H. Park, A. Zarrinpar, W.A. Lim, Rewriting MAP kinase pathways using alternative scaffold assembly mechanisms, Science 299 (2003) 1061–1064.
[42] K. Harris, R.E. Lamson, B. Nelson, T.R. Hughes, M.J. Marton, C.J. Roberts, C. Boone, P.M. Pryciak, Role of scaffolds in MAP kinase pathway specificity revealed by custom design of pathway-dedicated signaling proteins, Current Biology 11 (2001) 1815–1824.
[43] M. Dickens, J.S. Rogers, J. Cavanagh, A. Raitano, Z. Xia, J.R. Halpern, M.E. Greenberg, C.L. Sawyers, R.J. Davis, A cytoplasmic inhibitor of the JNK signal transduction pathway, Science 277 (1997) 693–696.
[44] L. Cohen, W.J. Henzel, P.A. Baeuerle, IKAP is a scaffold protein of the IkB kinase complex, Nature 395 (1998) 292–296.
[45] R.L. Kortum, R.E. Lewis, The molecular scaffold KSR1 regulates the proliferative and oncogenic potential of cells, Molecular and Cellular Biology 24 (2004) 4407–4416.
[46] E.O. Brigham, The Fast Fourier Transform and its Applications, Prentice Hall, Englewood Cliffs, NJ, 1988.
[47] C. Van Loan, Computational Frameworks for the Fast Fourier Transform, Society for Industrial Mathematics, 1992.