J Sci Comput (2016) 67:860–882
DOI 10.1007/s10915-015-0109-1

Two-Level Space–Time Domain Decomposition Methods for Three-Dimensional Unsteady Inverse Source Problems

Xiaomao Deng[1] · **Xiao-Chuan Cai**[2] · **Jun Zou**[3]

Received: 27 November 2014 / Revised: 8 September 2015 / Accepted: 16 September 2015 /
Published online: 28 September 2015

Abstract As the number of processor cores on supercomputers becomes larger and larger, algorithms with high degree of parallelism attract more attention. In this work, we propose a two-level space–time domain decomposition method for solving an inverse source problem associated with the time-dependent convection–diffusion equation in three dimensions. We introduce a mixed finite element/finite difference method and a one-level and a two-level space–time parallel domain decomposition preconditioner for the Karush–Kuhn–Tucker system induced from reformulating the inverse problem as an output least-squares optimization problem in the entire space-time domain. The new full space–time approach eliminates the sequential steps in the optimization outer loop and the inner forward and backward time marching processes, thus achieves high degree of parallelism. Numerical experiments validate that this approach is effective and robust for recovering unsteady moving sources. We will present strong scalability results obtained on a supercomputer with more than 1000 processors.

Keywords Space–time method · Multilevel method · Domain decomposition preconditioner · Unsteady inverse source problem · Parallel computing

✉ Xiao-Chuan Cai
cai@cs.colorado.edu

Xiaomao Deng
xm.deng@siat.ac.cn

Jun Zou
zou@math.cuhk.edu.hk

1 Laboratory for Engineering and Scientific Computing, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, Guangdong, People's Republic of China

2 Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309, USA

3 Department of Mathematics, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, People's Republic of China

Mathematics Subject Classification 49K20 · 65F22 · 65F08 · 65F10 · 65M32 · 65M55 · 65Y05 · 90C06

1 Introduction

In this paper, we consider an inverse problem associated with the time-dependent convection–diffusion equation defined in $\Omega \in \mathbf{R}^3$:

$$\begin{cases} \dfrac{\partial C}{\partial t} = \nabla \cdot (a(\mathbf{x})\nabla C) - \nabla \cdot (\mathbf{v}(\mathbf{x})C) + f(\mathbf{x},t), \quad 0 < t < T, \ \mathbf{x} \in \Omega \\ C(\mathbf{x},t) = p(\mathbf{x},t), \quad \mathbf{x} \in \Gamma_1 \\ a(\mathbf{x})\dfrac{\partial C}{\partial \mathbf{n}} = q(\mathbf{x},t), \quad \mathbf{x} \in \Gamma_2 \\ C(\mathbf{x},0) = C_0(\mathbf{x}), \quad \mathbf{x} \in \Omega, \end{cases} \tag{1}$$

where $f(\mathbf{x},t)$ is the source profile to be recovered, $a(\mathbf{x})$ and $\mathbf{v}(\mathbf{x})$ are the given diffusivity and convective coefficients, and Γ_1 and Γ_2 are two disjoint parts of the boundary $\partial\Omega$. Dirichlet and Neumann boundary conditions are imposed respectively on Γ_1 and Γ_2. When the observation data $C(\mathbf{x},t)$ is available at certain locations, several classes of inverse problems associated with the convection–diffusion equation (1) have been investigated, such as the recovery of the diffusivity coefficient with applications in, for examples, laminar wavy film flows [21], and flows in porous media [27], the recovery of the source with applications in, for examples, convective heat transfer problems [39], indoor airborne pollutant tracking [25], and ground water contamination modeling [29,32,36], etc.

The main focus of this work is to study the following inverse problem: given the measurement data $C^\varepsilon(\mathbf{x},t)$ of $C(\mathbf{x},t)$ at some locations inside Ω for the period $0 < t < T$ (ε denotes the noise level), we try to recover the space–time-varying source locations and intensities, i.e., the source function $f(\mathbf{x},t)$ in Eq. (1). The inverse source problem has been studied in different cases, for example, the recovering of the location and time-dependent intensity of point sources in [4,13,20,34], the piecewise-constant sources in [3,37] and Gaussian concentrated sources in [2,3]. Among these different approaches, the Tikhonov optimization method is most popular [4,13,19,34], which reformulates the original inverse source problem into an output least-squares optimization problem with PDE-constraints, by introducing some appropriate regularizations to ensure the stability of the resulting optimization problem with respect to the change of noise in the observation data [14,35].

We define the following objective functional with Tikhonov regularization:

$$J(f) = \frac{1}{2}\int_0^T \int_\Omega A(\mathbf{x})(C(\mathbf{x},t) - C^\varepsilon(\mathbf{x},t))^2 \, d\mathbf{x}dt + N_\beta(f), \tag{2}$$

where $C = C(f)$ is the solution to the system (1) corresponding to a given source f, $A(\mathbf{x})$ is the data range indicator function given by $A(\mathbf{x}) = \sum_{i=1}^s \delta(\mathbf{x}-\mathbf{x}_i)$, with $\mathbf{x}_1, \ \mathbf{x}_2, \ \ldots, \ \mathbf{x}_s$ being a set of specified locations where the concentration C is measured, and $C^\varepsilon(\mathbf{x},t)$ represents the measurement of $C(\mathbf{x},t)$ at a specified location $\mathbf{x}$ and time t. The term $N_\beta(f)$ in (2) is called the regularization with respect to the source. Since $f(\mathbf{x},t)$ depends on both space and time, we propose the following space–time H^1–H^1 regularization:

$$N_\beta(f) = \frac{\beta_1}{2}\int_0^T \int_\Omega |\dot{f}|^2 d\mathbf{x}dt + \frac{\beta_2}{2}\int_0^T \int_\Omega |\nabla f|^2 d\mathbf{x}dt. \tag{3}$$

Here β_1 and β_2 are two regularization parameters. Other regularizations, such as H^1–L^2, may be used, but we will show later by numerical experiments that H^1–H^1 regularization may offer better numerical reconstructions.

Various approaches are available for the minimization of the nonlinear functional $J(f)$ in (2) associated with the system (1) [22,34,38]. One of the approaches is the Lagrange multiplier method, which converts the constrained minimization of functional $J(f)$ into a unconstrained minimization of the corresponding Lagrange functional of $J(f)$. This results in the solution of a so-called Karush–Kuhn–Tucker (KKT) system [23], which involves three coupled time-dependent PDEs here, namely the governing equation (1) for the concentration C, its adjoint equation for the Lagrange multiplier and the equation for the identifying source function f; see Sect. 2 for more detail. For solving such a KKT system, the traditional reduced space SQP method is a popular and natural choice [2,13,34]. The SQP method solves the three coupled PDEs in the KKT system alternatively by iteration. One may see the essential sequential feature of the SQP: the outer iteration is sequential among the solutions of three PDEs, the governing equation (1) is forward in time, and the adjoint equation for the Lagrange multiplier is backward in time. The parallelization of the SQP may happen for the solution of each of the three time-dependent PDEs, which can be solved by, e.g., the traditional fast algorithms such as domain decomposition and multigrid methods [2]. SQP requires low memory, but it usually takes a large number of iterations to reach convergence. Because of its essential sequential feature, the reduced space SQP method is less ideal for parallel computers with a large number of processor cores, compared with the full space SQP methods. Full space methods were studied for steady state problems in [9,11], but for unsteady problems it needs to eliminate the sequential steps in the outer iteration of the SQP and solve the full space–time system as a coupled system. Because of the much larger size of the system, the full space approach may not be suitable for parallel computer systems with a small number of processor cores, but it has fewer sequential steps and thus offers a much higher degree of parallelism required by large scale supercomputers [12].

It is a very active research direction to construct efficient parallelization methods for highly nonlinear optimizations constrained with PDEs. An unsteady PDE-constrained optimization problem was solved in [38] for the boundary control of unsteady incompressible flows by solving a subproblem at each time step. It has the sequential time-marching process and each subproblem is steady-state. The parareal algorithms were studied in [5,15,24], which involve a coarse-grid solver for prediction and a fine-grid solver for correction in time. Parallel implicit time integrator method (PITA), space–time multigrid, multiple shooting methods can be categorized as improved versions of the parareal algorithm [17,18]. The parareal algorithm combined with domain decomposition method [26] or multigrid method can be powerful. However, most existing parareal related studies have focused mainly on the stability and convergence [16].

In this work, we will study an effective reconstruction of the time history and intensity profile of a source function simultaneously. For this aim, we propose a fully implicit, mixed finite element and finite difference discretization scheme for the globally coupled KKT system, and a corresponding space–time overlapping Schwarz preconditioner for solving the large-scale discretised KKT system. The method removes all the sequential inner time steps and achieves full parallelization in both space and time. We shall not compare the proposed method with traditional reduced space SQP methods, since it is likely that, for supercomputers with a small number of processor cores, the traditional approach is still a better choice. The focus of the current study is to formulate the technical details of our new space–time parallel algorithm, which may play a promising role for future exascale computing systems. Furthermore, to resolve the dilemma that the number of linear iterations

of one-level methods increases with the number of processors [33], we will develop a two-level space–time hybrid Schwarz preconditioner, which offers better performance in terms of the number of iterations and the total compute time.

The rest of the paper is arranged as follows. In Sect. 2, we describe the mathematical formulation of the inverse problem and the derivation of the KKT system. We propose, in Sect. 3, the main algorithm of the paper, and discuss several technical issues involved in the fully implicit discretization scheme and the one- and two-level overlapping Schwarz methods for solving the KKT system. Numerical experiments for the recovery of 3D sources are given in Sect. 4, and some concluding remarks are provided in Sect. 5.

2 Strong Formulation of KKT System

We now derive the KKT system of the minimisation of $J(f)$ in (2) combined with the Eq. (1). To do so, we formally write the first equation in (1) as an operator equation $L(C, f) = 0$. By introducing a Lagrange multiplier or adjoint variable $G \in H^1(0, T; H^1(\Omega))$, the Lagrange functional [3,22]

$$\mathscr{J}(C, f, G) = \frac{1}{2}\int_0^T \int_\Omega A(\mathbf{x})(C - C^\varepsilon)^2 d\mathbf{x}dt + N_\beta(f) + (G, L(C, f)) \tag{4}$$

transforms the PDE-constrained minimization of $J(f)$ in (2) into an unconstrained saddle-point optimization problem, where $(G, L(C, f))$ denotes the inner product of G and $L(C, f)$. Two approaches are available for the resulting saddle-point optimization of $\mathscr{J}$, the optimize–then–discretize approach and the discretize–then–optimize approach. The first approach derives a continuous optimality system and then applies certain discretization scheme, such as a finite element method to obtain a discrete system ready for computation. The second approach discretizes the Lagrange functional $\mathscr{J}$, and then the objective functional becomes a finite dimensional quadratic polynomial. The solution algorithm is then based on the polynomial system. The two approaches perform the approximation and discretization at different stages, both have been applied successfully [28]. We shall use the optimize–then–discretize approach in this work.

The first-order optimality conditions of $\mathscr{J}$ in (4), i.e., the KKT system, is obtained by taking its variations with respect to G, C and f as

$$\begin{cases} \mathscr{J}_G(C, f, G)v = 0 \\ \mathscr{J}_C(C, f, G)w = 0 \\ \mathscr{J}_f(C, f, G)g = 0 \end{cases} \tag{5}$$

for all $v, w \in L^2(0, T; H^1_{\Gamma_1}(\Omega))$ with zero traces on Γ_1 and $g \in H^1(0, T; H^1(\Omega))$. Then by using integration by parts, we may derive the following strong form of the KKT system:

$$\begin{cases} \dfrac{\partial C}{\partial t} - \nabla \cdot (a\nabla C) + \nabla \cdot (\mathbf{v}(\mathbf{x})C) - f = 0 \\ -\dfrac{\partial G}{\partial t} - \nabla \cdot (a\nabla G) - \mathbf{v}(\mathbf{x}) \cdot \nabla G + A(\mathbf{x})C = A(\mathbf{x})C^\varepsilon \\ G + \beta_1 \dfrac{\partial^2 f}{\partial t^2} + \beta_2 \Delta f = 0. \end{cases} \tag{6}$$

To derive the boundary, initial and terminal conditions for each variable of the equations, we make use of the property that (5) holds for arbitrary directional functions v, w and g. For

the state equation, i.e., the first one in (6), it maintains the same conditions as in (1). For the adjoint equation, i.e., the second one in (5) or (6), we can deduce by integration by parts for any test function $w \in L^2(0,T; H^1_{\Gamma_1}(\Omega))$ with $w(\cdot,0)=0$ and $a(\mathbf{x})\partial w/\partial \mathbf{n} = 0$ on Γ_2:

$$\begin{aligned}\mathscr{J}_C(C,f,G)w = &\int_0^T\int_\Omega A(\mathbf{x})(C-C^\varepsilon)w d\mathbf{x}dt + \int_\Omega G(\mathbf{x},T)w(\mathbf{x},T)d\mathbf{x}\\ &-\int_0^T\int_\Omega\left(\frac{\partial G}{\partial t}+\nabla\cdot(a(\mathbf{x})\nabla G)+\mathbf{v}(\mathbf{x})\cdot\nabla G\right)w d\mathbf{x}dt\\ &-\int_0^T\int_{\Gamma_1}\left(a(\mathbf{x})\frac{\partial w}{\partial \mathbf{n}}\right)G d\Gamma dt\\ &+\int_0^T\int_{\Gamma_2}\left(a(\mathbf{x})\frac{\partial G}{\partial \mathbf{n}}+\mathbf{v}(\mathbf{x})\cdot\mathbf{n}\right)w d\Gamma dt.\end{aligned}$$

By the arbitrariness of w, the boundary and terminal conditions for G are derived:

$$\begin{aligned}&G(\mathbf{x},t)=0, \quad \mathbf{x}\in\Gamma_1,\ t\in[0,T]\\ &a(\mathbf{x})\frac{\partial G}{\partial \mathbf{n}}+\mathbf{v}(\mathbf{x})\cdot\mathbf{n}=0, \quad \mathbf{x}\in\Gamma_2,\ t\in[0,T]\\ &G(\mathbf{x},T)=0, \quad \mathbf{x}\in\Omega.\end{aligned}$$

Similarly for the third equation of (5) or (6), we can deduce

$$\begin{aligned}\mathscr{J}_f(C,f,G)g &= -\int_0^T\int_\Omega G g d\mathbf{x}dt + \int_0^T\int_\Omega(\dot f\dot g+\nabla f\cdot\nabla g)d\mathbf{x}dt\\ &= -\int_0^T\int_\Omega G g d\mathbf{x}dt + (\dot f g)|_{t=0,T} - \int_0^T\int_\Omega \ddot f g\\ &\quad +\int_0^T\int_{\partial\Omega}\frac{\partial f}{\partial\mathbf{n}}g d\Gamma dt - \int_0^T\int_\Omega \Delta f g d\mathbf{x}dt\\ &= -\int_0^T\int_\Omega(G+\ddot f+\Delta f)g d\mathbf{x}dt + (\dot f g)|_{t=0,T} + \int_0^T\int_{\partial\Omega}\frac{\partial f}{\partial\mathbf{n}}g d\Gamma dt.\end{aligned}$$

Using the arbitrariness of g, we derive the boundary, initial and terminal conditions for f:

$$\frac{\partial f}{\partial t}=0 \quad \text{for } t=0,T,\ \mathbf{x}\in\Omega; \qquad \frac{\partial f}{\partial \mathbf{n}}=0 \quad \text{for } \mathbf{x}\in\partial\Omega,\ t\in[0,T]. \tag{7}$$

3 A Fully Implicit and Fully Coupled Method

In this section, we first introduce a mixed finite element and finite difference method for the discretization of the continuous KKT system derived in the previous section, then we briefly mention the algebraic structure of the discrete system of equations. In the second part of the section, we introduce the one- and two-level space–time Schwarz preconditioners that are the most important components for the success of the overall algorithm.

3.1 Fully Implicit Space–Time Discretization

In this subsection, we introduce a fully implicit finite element/finite difference scheme to discretize (6). To discretize the state and adjoint equations, i.e., the first two equations in

(6), we use a second-order Crank–Nicolson finite difference scheme in time and a piecewise linear continuous finite element method in space. Consider a regular triangulation $\mathscr{T}^h$ of domain Ω, and a time partition P^τ of the interval $[0, T]$: $0 = t^0 < t^1 < \cdots < t^M = T$, with $t^n = n\tau$, $\tau = T/M$. Let V^h be the piecewise linear continuous finite element space on $\mathscr{T}^h$, and $\mathring{V}^h$ be the subspace of V^h with zero trace on Γ_1. We introduce the difference quotient and the averaging of a function $\psi(\mathbf{x}, t)$ as

$$\partial_\tau \psi^n(\mathbf{x}) = \frac{\psi^n(\mathbf{x}) - \psi^{n-1}(\mathbf{x})}{\tau}, \quad \bar{\psi}^n(\mathbf{x}) = \frac{1}{\tau}\int_{t^{n-1}}^{t^n} \psi(\mathbf{x}, t)dt,$$

with $\psi^n(\mathbf{x}) := \psi(\mathbf{x}, t^n)$. Let π_h be the finite element interpolation associated with the space V^h, then we obtain the discretizations for the state and adjoint equations by finding the sequence of approximations $C_h^n, G_h^n \in V^h$ for $n = 0, 1, \ldots, M$ such that $C_h^0 = \pi_h C_0$, $G_h^M = \mathbf{0}$, and $C_h^n(\mathbf{x}) = \pi_h p(\mathbf{x}, t^n)$, $G_h^n(\mathbf{x}) = 0$ for $\mathbf{x} \in \Gamma_1$, and satisfying

$$\begin{cases} \left(\partial_\tau C_h^n, v_h\right) + \left(a\nabla \bar{C}_h^n, \nabla v_h\right) + \left(\nabla \cdot \left(\mathbf{v}\bar{C}_h^n\right), v_h\right) = \left(\bar{f}_h^n, v_h\right) + \langle \bar{q}^n, v_h\rangle_{\Gamma_2}, \quad \forall\, v_h \in \mathring{V}^h \\ -\left(\partial_\tau G_h^n, w_h\right) + \left(a\nabla \bar{G}_h^n, \nabla w_h\right) + \left(\nabla \cdot (\mathbf{v} w_h), \bar{G}_h^n\right) \\ = -\left(A(\mathbf{x})\left(\bar{C}_h^n(\mathbf{x}) - \bar{C}^{\varepsilon,n}(\mathbf{x})\right), w_h\right), \quad \forall w_h \in \mathring{V}^h. \end{cases} \tag{8}$$

Unlike the approximations of the forward and adjoint equations in (8), we shall approximate the source function f differently. We know that the source function satisfies an elliptic equation [see the third equation in (6)] in the space–time domain $\Omega \times (0, T)$. So we shall apply $\mathscr{T}^h \times P^\tau$ to generate a partition of the space–time domain $\Omega \times (0, T)$, and then apply the piecewise linear finite element method in both space (three dimensions) and time (one dimension), denoted by W_h^τ, to approximate the source function f. Then the equation for $f \in W_h^\tau$ can be discretized as follows: Find the sequence of f_h^n for $n = 0, 1, \ldots, M$ such that

$$-(G_h^n, g_h^\tau) + \beta_1(\partial_\tau f_h^n, \partial_\tau g_h^\tau) + \beta_2(\nabla f_h^n, \nabla g_h^\tau) = 0, \quad \forall\, g_h^\tau \in W_h^\tau. \tag{9}$$

The coupled system (8)–(9) is the so-called fully discretized KKT system. In the Appendix, we provide some details of the discrete structure of this KKT system.

3.2 One- and Two-Level Space–Time Schwarz Preconditioner

The unknowns of the KKT system (8)–(9) are often ordered physical variable by physical variable, namely in the form

$$\tilde{U} = (C^0, C^1, \ldots, C^M, G^0, G^1, \ldots, G^M, f^0, f^1, \ldots, f^M)^T.$$

Such ordering is used extensively in reduced space SQP methods [13]. In our all-at-once method, the unknowns C, G and f are ordered mesh point by mesh point and time step by time step, and all unknowns associated with a point stay together as a block. At each mesh point $\mathbf{x}_j$, $j = 1, \ldots, N$, and time step t^n, $n = 0, \ldots, M$, the unknowns are arranged in the order of C_j^n, G_j^n, f_j^n. Such ordering avoids zero values on the main diagonal of the matrix and has better cache performance for point-block LU (or ILU) factorization based subdomain solvers. More precisely, we define the solution vector as

$$U = (C_1^0, G_1^0, f_1^0, \ldots, C_N^0, G_N^0, f_N^0, C_1^1, G_1^1, f_1^1, \ldots, C_N^1, G_N^1, f_N^1, \ldots, C_1^M, G_1^M, f_1^M, \ldots, C_N^M, G_N^M, f_N^M)^T.$$

then the linear system (8)–(9) is rewritten as

$$F_h U = b, \tag{10}$$

where F_h is a sparse block matrix of size $(M+1)(3N)$ by $(M+1)(3N)$ with the following block structure:

$$F_h = \begin{pmatrix} S_{00} & S_{01} & \mathbf{0} & \cdots & \mathbf{0} \\ S_{10} & S_{11} & S_{12} & \cdots & \mathbf{0} \\ \mathbf{0} & \ddots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & S_{M-1,M-2} & S_{M-1,M-1} & S_{M-1,M} \\ \mathbf{0} & \cdots & \mathbf{0} & S_{M,M-1} & S_{M,M} \end{pmatrix},$$

where the block matrices S_{ij} for $0 \le i, j \le M$ are of size $3N \times 3N$ and most of its elements are zero matrices except the ones in the tridiagonal stripes $\{S_{i,i-1}\}$, $\{S_{i,i}\}$, $\{S_{i,i+1}\}$. It is noted that if we denote the submatrices for C, G and f of size $N \times N$ respectively by S_{ij}^C, S_{ij}^G, S_{ij}^f in each block S_{ij}, the sparsity of the matrices are inconsistent, namely, S_{ij}^f is the densest and S_{ij}^G is the sparest. This is due to the discretization scheme we have used. The system (10) is large-scale and ill-conditioned, therefore is difficult to solve because the space–time coupled system is denser than the decoupled system, especially in three dimensions.

We shall design the preconditioner by extending the classical spatial Schwarz preconditioner to include both spatial and temporal variables. Such an approach eliminates all sequential steps and the unknowns at all time steps are solved simultaneously. We use a right-preconditioned Krylov subspace method to solve (10),

$$F_h M^{-1} U' = b,$$

where M^{-1} is a space–time Schwarz preconditioner and $U = M^{-1}U'$.

Denoting the space–time domain by $\Theta = \Omega \times (0, T)$, an overlapping decomposition of Θ is defined as follows: we divide Ω into N_s subdomains, $\Omega_1, \Omega_2, \ldots, \Omega_{N_s}$, then partition the time interval $[0, T]$ into N_t subintervals using the partition: $0 < T_1 < T_2 < \cdots < T_{N_t}$. By coupling all the space subdomains and time subintervals, a decomposition of Θ is $\Theta = \cup_{i=1}^{N_s}(\cup_{j=1}^{N_t}\Theta_{ij})$, where $\Theta_{ij} = \Omega_i \times (T_{j-1}, T_j)$. For convenience, the number of subdomains, i.e. $N_s N_t$, is equal to the number of processors. These subdomains Θ_{ij} are then extended to Θ'_{ij} to overlap each other. The boundary of each subdomain is extended by an integral number of mesh cells in each dimension, and we trim the cells outside of Θ. The corresponding overlapping decomposition of Θ is $\Theta = \cup_{i=1}^{N_s}(\cup_{j=1}^{N_t}\Theta'_{ij})$. See the left figure of Fig. 1 for the overlapping extension. The matrix on each subdomain $\Theta'_{ij} = \Omega'_i \times (T'_{j-1}, T'_j)$, $i = 1, 2, \ldots, N_s$, $j = 1, 2, \ldots, N_t$ is the discretized version of the following system of PDEs

$$\begin{cases} \dfrac{\partial C}{\partial t} = \nabla \cdot (a(\mathbf{x})\nabla C) - \nabla \cdot (\mathbf{v}(\mathbf{x})C) + f(\mathbf{x}, t), \quad (\mathbf{x}, t) \in \Theta'_{ij} \\ \dfrac{\partial G}{\partial t} = -\nabla \cdot (a(\mathbf{x})\nabla G) - \mathbf{v}(\mathbf{x}) \cdot \nabla G \\ \qquad\qquad + A(\mathbf{x})(C(\mathbf{x}, t) - C^{\varepsilon}(\mathbf{x}, t)), \quad (\mathbf{x}, t) \in \Theta'_{ij} \\ \beta_1 \dfrac{\partial^2 f}{\partial t^2} + \beta_2 \Delta f + G = 0, \quad (\mathbf{x}, t) \in \Theta'_{ij} \end{cases} \tag{11}$$

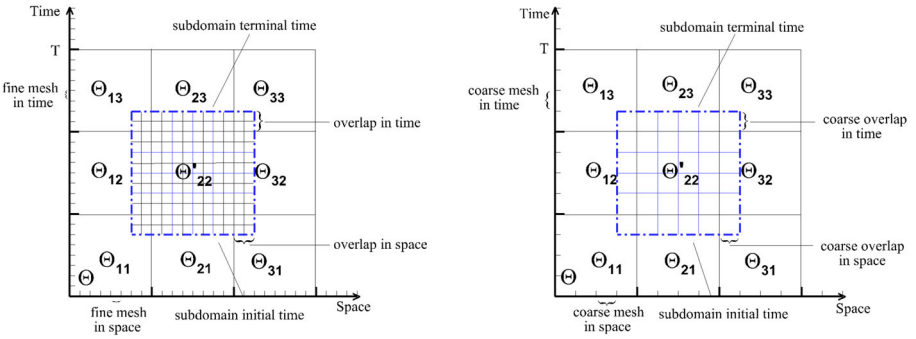

Fig. 1 *Left* a sample overlapping decomposition in space–time domain Θ on a fine mesh. *Right* the same decomposition on the coarse mesh

with the following boundary conditions

$$C(\mathbf{x},t)=0,\quad G(\mathbf{x},t)=0,\quad f(\mathbf{x},t)=0,\ \ \mathbf{x}\in\partial\Omega_i',\ t\in[T'_{j-1},T'_j] \tag{12}$$

along with the initial and terminal time boundary conditions

$$\begin{cases} C(\mathbf{x},T'_{j-1})=0,\quad G(\mathbf{x},T'_{j-1})=0,\quad f(\mathbf{x},T'_{j-1})=0,\ \ \mathbf{x}\in\Omega_i' \\ C(\mathbf{x},T'_j)=0,\quad G(\mathbf{x},T'_j)=0,\quad f(\mathbf{x},T'_j)=0,\ \ \mathbf{x}\in\Omega_i'. \end{cases} \tag{13}$$

One may notice from (13) that the homogenous Dirichlet boundary conditions are applied in each time interval (T'_{j-1},T'_j), as the solution of the subdomain problem is not really physical. This is one of the major differences between the space–time Schwarz method and the parareal algorithm [24]. The time boundary condition for each subproblem of the parareal algorithm is obtained by an interpolation of the coarse solution, and if the coarse mesh is fine enough, the solution of the subdomain problem is physical. Therefore, the parareal algorithms can be used as a solver, but our space–time Schwarz method can only be used as a preconditioner. Surprisingly, as we shall see from our numerical experiments in Sect. 4, the Schwarz preconditioner is an excellent one even though the time boundary conditions violate the physics.

We solve the subdomain problems using the same method as for the global problem (10), no time-marching is performed in our new algorithm, and all unknowns affiliated with the subdomain are solved simultaneously. Let M_{ij} be the matrix generated in the same way as the global matrix F_h in (10) but for the subproblem (11)–(13), and $\tilde{M}_{ij}^{-1}$ be an exact or approximate inverse of M_{ij}. By denoting the restriction matrix from Θ to the subdomain Θ'_{ij} by R_{ij}^{δ}, with overlapping size δ, we propose the following one-level space–time restricted Schwarz preconditioner for the global matrix F_h:

$$M_{one\text{-}level}^{-1}=\sum_{j=1}^{N_t}\sum_{i=1}^{N_s}(R_{ij}^{\delta})^T\tilde{M}_{ij}^{-1}R_{ij}^{0}.$$

As it is well known, any one-level domain decomposition methods are not scalable with the increasing number of subdomains or processors [33]. Instead one should have multilevel methods in order to observe possible scalable effects [1,33]. We now propose a two-level space–time additive Schwarz preconditioner. To do so, we partition Ω with a fine mesh Ω^h and a coarse mesh Ω^c. For the time interval, we have a fine partition P^{τ} and a coarse

partition P^{τ_c} with $\tau < \tau_c$. We will adopt a nested mesh, i.e., the nodal points of the coarse mesh $\Omega^c \times P^{\tau_c}$ are a subset of the nodal points of the fine mesh $\Omega^h \times P^{\tau}$. In practice, the size of the coarse mesh should be adjusted properly to obtain the best performance. On the fine level, we simply apply the previously defined one-level space–time additive Schwarz preconditioner; and to efficiently solve the coarse problem, a parallel coarse preconditioner is also necessary. Here we use the overlapping space–time additive Schwarz preconditioner and for simplicity divide $\Omega^c \times P^{\tau_c}$ into the same number of subdomains as on the fine level, using the non-overlapping decomposition $\Theta = \cup_{i=1}^{N_s}(\cup_{j=1}^{N_t}\Theta_{ij})$. When the subdomains are extended to overlapping ones, the overlapping size is not necessarily the same as that on the fine mesh. See the right figure of Fig. 1 for a coarse version of the space–time decomposition. We denote and define the preconditioner on the coarse level by

$$M_c^{-1} = \sum_{j=1}^{N_t}\sum_{i=1}^{N_s}(R_{ij,c}^{\delta_c})^T \tilde{M}_{ij,c}^{-1} R_{ij,c}^0,$$

where δ_c is the overlapping size on the coarse mesh. Here the matrix $\tilde{M}_{ij,c}^{-1}$ is an approximate inverse of $M_{ij,c}$ which is obtained by a discretization of (11)–(13) on the coarse mesh on Θ'_{ij}.

To combine the coarse preconditioner with the fine mesh preconditioner, we need a restriction operator I_h^c from the fine to coarse mesh and an interpolation operator I_c^h from the coarse to fine mesh. For our currently used nested structured mesh and linear finite elements, I_c^h is easily obtained using a linear interpolation on the coarse mesh and $I_h^c = (I_c^h)^T$. We note that when the coarse and fine meshes are nested, instead of using $I_h^c = (I_c^h)^T$, we may take I_h^c to be a simple restriction, e.g., the identity one which assigns the values on the coarse mesh using the same values on the fine mesh. In general, the coarse preconditioner and the fine preconditioner can be combined additively or multiplicatively. According to our experiments, the following multiplicative version works well:

$$\begin{cases} y = I_c^h F_c^{-1} I_h^c x \\ M_{two\text{-}level}^{-1} x = y + M_{one\text{-}level}^{-1}(x - F_h y), \end{cases} \tag{14}$$

where F_c^{-1} corresponds to the GMRES solver right-preconditioned by M_c^{-1} on the coarse level, and F_h is the discrete KKT system (10) on the fine level.

4 Numerical Experiments

In this section we present some numerical experiments to study the parallel performance and robustness of the newly developed algorithms. When using the one-level preconditioner, we use a restarted GMRES method (restarts at 50) to solve the preconditioned system; when using the two-level preconditioner, we use the restarted flexible GMRES (fGMRES) method [30] (restarts at 30), considering the fact that the overall preconditioner changes from iteration to iteration because of the iterative coarse solver. Although fGMRES needs more memory than GMRES, we have observed its number of iterations can be significantly reduced. The relative convergence tolerance of both GMRES and fGMRES is set to be 10^{-6}. The initial guesses for both GMRES and fGMRES method are zero. The size of the overlap between two neighboring subdomains, denoted by $iovlp$, is set to be 1 unless otherwise specified. The subsystem on each subdomain is solved by an incomplete LU factorization ILU(k), with k being its fill-in level, and $k = 0$ if not specified. The algorithms are implemented based on the

Portable, Extensible Toolkit for Scientific computation (PETSc) [6] and run on a Dawning TC3600 blade server system at the National Supercomputing Center in Shenzhen, China with a 1.271 PFlops/s peak performance.

In our computations, the settings for the model system (1) are taken as follows. The computational domain, the terminal time and the initial condition are taken to be $\Omega = (-2, 2)^3$, $T = 1$ and $C(\cdot, 0) = 0$ respectively. Let $L = S = H = 2$, then the homogeneous Dirichlet and Neumann conditions in (1) are respectively imposed on $\Gamma_1 = \{\mathbf{x} = (x_1, x_2, x_3);\ |x_1| = L \text{ or } |x_2| = S\}$ and $\Gamma_2 = \{\mathbf{x} = (x_1, x_2, x_3);\ |x_3| = H\}$. Furthermore, the diffusivity and convective coefficients are set to be $a(\mathbf{x}) = 1.0$ and $\mathbf{v}(\mathbf{x}) = (1.0, 1.0, 1.0)^T$.

In order to generate the observation data, we solve the forward convection–diffusion equation (1) on a very fine mesh $265 \times 265 \times 265$ with a small time step size $1/96$, and the resulting approximate solution $C(\mathbf{x}, t)$ is used as the noise-free observation data. The measurement data are chosen on a set of nested meshes (the concrete choice is given for each numerical example later), which may not necessarily be part of the fine mesh we used to compute the noise-free data and hence linear interpolations are needed to obtain the concentration at each selected measurement point. Then a random noise is added in the following form at the locations where the measurements are taken:

$$C^{\varepsilon}(\mathbf{x}_i, t) = C(\mathbf{x}_i, t) + \varepsilon\, r\, C(\mathbf{x}_i, t), \quad i = 1, \ldots, s.$$

here r is a random function with the standard Gaussian distribution, and ε is the noise level. We take $\varepsilon = 1\,\%$ in our numerical experiments if it is not specified otherwise. As for most inverse problems, the regularization parameters [see β_1 and β_2 in (3)] are important to effective numerical reconstructions. In this work we shall not discuss about the technical selection of these regularization parameters but choose them heuristically.

The numerical tests are designed to investigate the reconstruction effects with different types of three-dimensional sources by the proposed one- or two-level space–time Schwarz method, as well as the robustness of the algorithm with respect to different noise levels, different regularizations and amount of measurement data. In addition, parallel efficiency of the proposed algorithms is also studied.

4.1 Reconstruction of 3D Sources

We devote this subsection to test the numerical reconstruction of three representative 3D sources by the proposed one-level space–time method, with $np = 256$ processors. Each of the three examples are constructed with its own special difficulty.

Example 1 (two Gaussian sources). This example tests two moving Gaussian sources in Ω, namely the source f takes the form:

$$f(\mathbf{x}, t) = \sum_{i=1}^{2} \exp\Big(- \frac{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}{a^2} \Big),$$

with $a = 2.0$ and two moving centers of the sources are given by

$$\begin{cases} (x_1, y_1, z_1) = (L \sin(2\pi t), S\cos(2\pi t), H\cos(4\pi t)) \\ (x_2, y_2, z_2) = (L - 2L|\cos(4t)|, -S + 2S|\cos(4t)|, -H + 2Ht^2). \end{cases} \tag{15}$$

The moving traces of the sources are shown in Fig. 2.

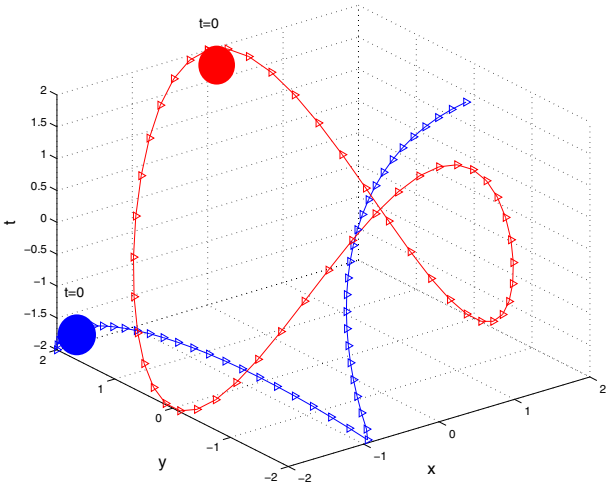

Fig. 2 The traces of two moving sources

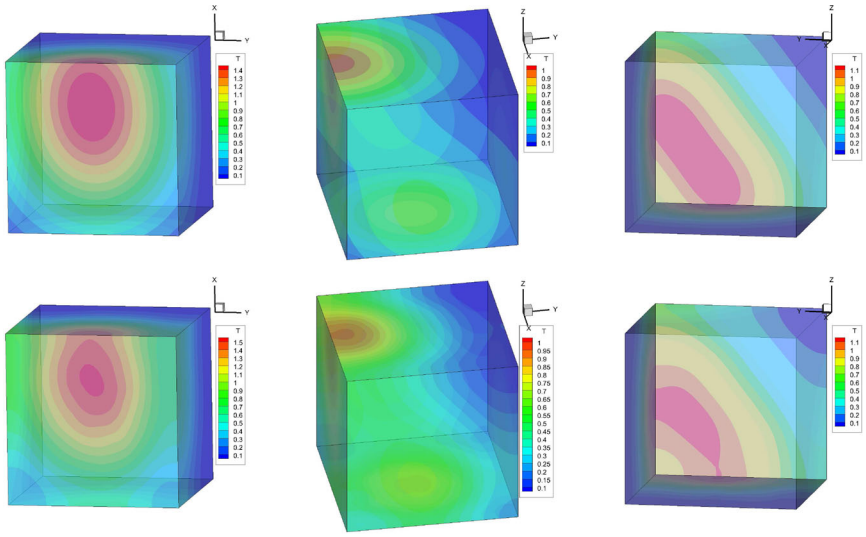

Fig. 3 Example 1: the source reconstructions at three moments $t = 10/39, 20/39, 30/39$ with measurements collected at the mesh $14 \times 14 \times 14$ (*bottom*), comparable with the exact source distribution (*top*)

In the first experiment, we use a $40 \times 40 \times 40$ mesh and time step size $1/39$ for the inversion process. The measurements are taken from the mesh $14 \times 14 \times 14$, which is uniformly located in Ω, with the mesh size being $1/13$. The regularization parameters are set to be $\beta_1 = 3.6 \times 10^{-6}$ and $\beta_2 = 3.6 \times 10^{-3}$. In Fig. 3, the numerically reconstructed sources are compared with the exact ones at three moments $t_1 = 10/39$, $t_2 = 20/39$, $t_3 = 30/39$. We can see that the source locations and intensities are quite close to the true values at the three chosen moments. Then we increase the noise level to $\varepsilon = 5\,\%$ and $\varepsilon = 10\,\%$, still with the same set of parameters. The reconstruction results are shown in Fig. 4. We observe that the reconstructed profiles deteriorate and become oscillatory as the noise level increases.

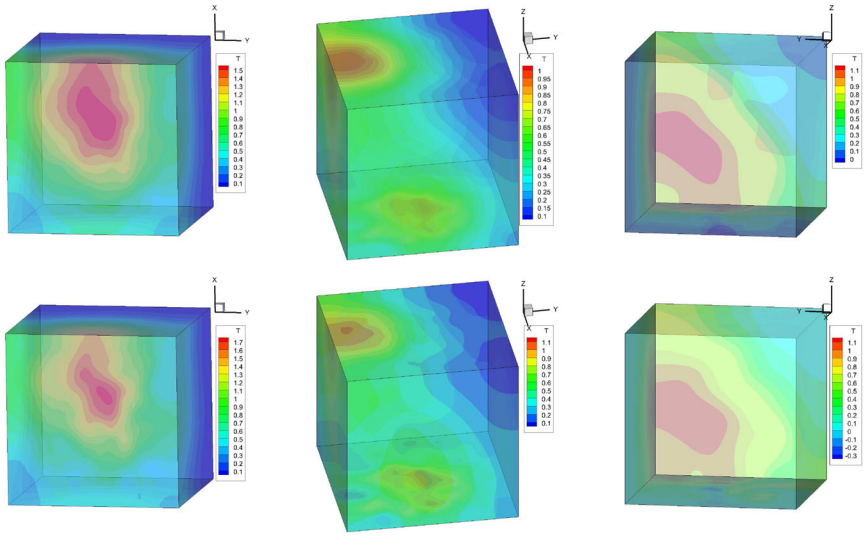

Fig. 4 Example 1: source reconstructions with noise level $\varepsilon = 5\,\%$ (*top*) and $\varepsilon = 10\,\%$ (*bottom*)

Table 1 L^2-norm errors at $t = 10/39, 20/39, 30/39$ with different noise levels and regularization parameters

β	L^2-norm errors	$\delta = 1\,\%$	$\delta = 5\,\%$	$\delta = 10\,\%$
$\beta_1 = 3.6e{-}6, \beta_2 = 3.6e{-}3$	e_1	0.043	0.045	0.053
	e_2	0.0491	0.057	0.076
	e_3	0.022	0.044	0.081
$\beta_1 = 3.6e{-}5, \beta_2 = 3.6e{-}3$	e_1	0.059	0.061	0.063
	e_2	0.064	0.067	0.076
	e_3	0.035	0.045	0.066
$\beta_1 = 3.6e{-}6, \beta_2 = 3.6e{-}4$	e_1	0.031	0.058	0.104
	e_2	0.036	0.094	0.108
	e_3	0.027	0.053	0.056
$\beta_1 = 3.6e{-}5, \beta_2 = 3.6e{-}4$	e_1	0.052	0.061	0.083
	e_2	0.058	0.081	0.127
	e_3	0.042	0.084	0.152

This is expected since the regularized solutions provided by the minimization of functional $J(f)$ in (2) become less accurate, so are their numerical approximate solutions obtained from the discretised KKT system (8)–(9). We have tested four different sets of regularization parameters for the H^1–H^1 regularization in (3), and present the L^2-norm errors between the reconstructed source f and the exact source function f^* at the aforementioned three moments t_1, t_2 and t_3. The L^2-norm error at time t_j is defined here by $e_j = \|(f - f^*)(t_j)\|_{L^2(\Omega)}$ for $j = 1, 2, 3$, and is shown in Table 1 for each set of regularization parameters.

Example 2 (Four constant sources). Appropriate choices of regularizations are important for the inversion process. In the previous example we have used a H^1–H^1 Tikhonov regularization in both space and time. In this example, we intend to compare the H^1–H^1 regularization with the following H^1–L^2 regularization

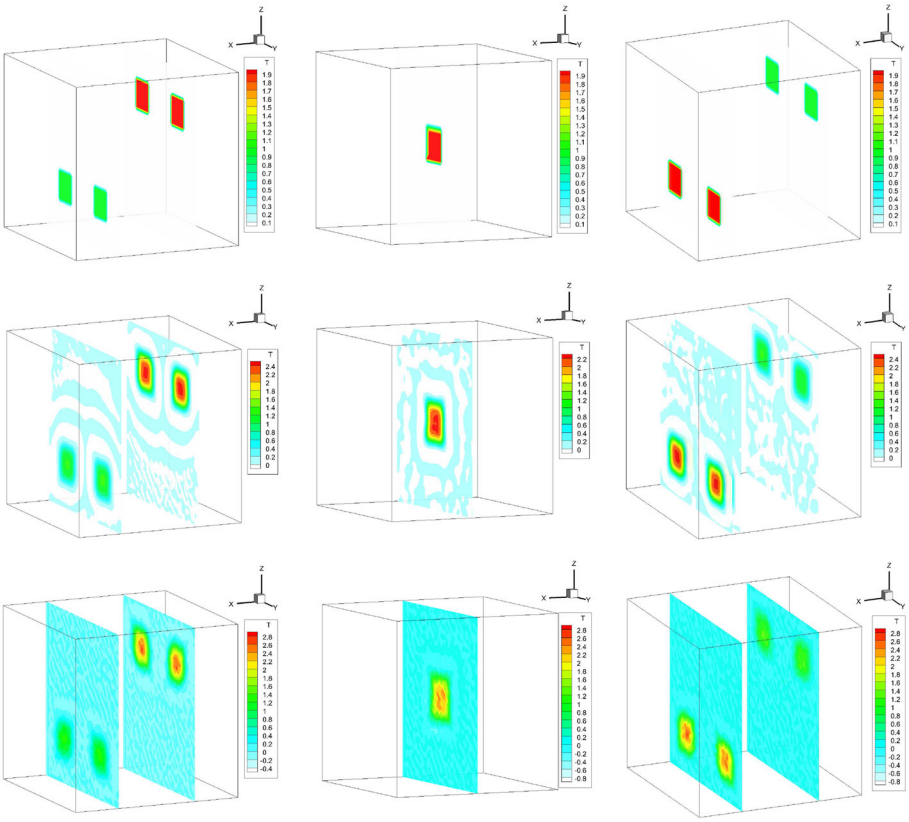

Fig. 5 Example 2: the source reconstructions with H^1–H^1 regularization (*mid*) and H^1–L^2 regularization (*bottom*), compared with the exact solution (*top*)

$$\tilde{N}_\beta(f) = \frac{\beta_1}{2}\int_0^T \int_\Omega |\dot{f}(\mathbf{x},t)|^2 d\mathbf{x}dt + \frac{\beta_2}{2}\int_0^T \int_\Omega f^2 d\mathbf{x}dt.$$

For the comparisons, we consider the case in which four constant sources move along the diagonals of the cube to their far corner. The four source distributions are specified by

$$f_i(\mathbf{x},t) = a_i \quad \text{for } |x - x_i| < 0.4, \quad |y - y_i| < 0.4, \quad |z - z_i| < 0.4$$

for $i = 1, 2, 3, 4$, where the constants a_i are given by $a_1 = a_4 = 2.0$, $a_2 = a_3 = 1.0$, and their traces are described respectively by

$$\begin{cases} (x_1, y_1, z_1) = (-L + 2Lt, -S + 2St, H - 2Ht) \\ (x_2, y_2, z_2) = (L - 2Lt, S - 2St, -H + 2Ht) \\ (x_3, y_3, z_3) = (L - 2Lt, -S + 2St, -H + 2Ht) \\ (x_4, y_4, z_4) = (-L + 2Lt, S - 2St, H - 2Ht). \end{cases}$$

Same mesh and measurements are used as in Example 1, and the regularization parameters are set to be $\beta_1 = 10^{-5}, \beta_2 = 10^{-3}$ in $N_\beta(f)$, and $\beta_1 = 10^{-5}, \beta_2 = 10^{-8}$ in $\tilde{N}_\beta(f)$, respectively. The reconstruction results are compared with the true solution at three moments $t_1 = 10/39$, $t_2 = 20/39$, $t_3 = 30/39$, and two slices at $x = 0.95$ and $x = -0.95$. It is

Table 2 L^2-norm errors of the reconstructed source at 3 moments for Example 2 with two regularizations

Time	Error with H^1–H^1 regularization	Error with H^1–L^2 regulzarization
10/39	0.019	0.023
20/39	0.012	0.019
30/39	0.019	0.026

observed from Fig. 5 that the resolution of the source profile is much better with the H^1–H^1 regularization $N_\beta(f)$ than with the H^1–L^2 regularization $\tilde{N}_\beta(f)$, and the latter presents a reconstruction process that is much less stable and much more oscillatory. Furthermore, we demonstrate in Table 2 the L^2-norm errors between the reconstructed source functions f_i and the exact sources f_i^* at the three specified moments t_1, t_2, t_3 for both the H^1–H^1 and H^1–L^2 regularizations. The L^2-norm error at time t_j is defined here by $e_j = \|\sum_{i=1}^4 (f_i - f_i^*)(t_j)\|_{L^2(\Omega)}$ for $j = 1, 2, 3$. We can see that the errors with the H^1–H^1 regularization are slightly smaller than that of the H^1–L^2 regularization.

Example 3 (Eight moving sources). This last example presents a very challenging case that eight Gaussian sources are initially located at the corners of the physical cubic domain, then move inside the cube following their own traces given below. The Gaussian sources are described by

$$f(\mathbf{x}, t) = \sum_{i=1}^{8} a_i \, e^{-(x-x_i)^2-(y-y_i)^2-(z-z_i)^2},$$

where the coefficients a_i and the source traces are represented by

$$a_1 = a_2 = a_3 = a_4 = 4.0; \quad a_5 = a_6 = a_7 = a_8 = 6.0,$$

and

$$\begin{aligned}
(x_1, y_1, z_1) &= (-L + 2L(1-t), -S + 2S(1-t), -H + 2H(1-t)) \\
(x_2, y_2, z_2) &= (-L + 2Lt, -S + 2St, -H + 2Ht) \\
(x_3, y_3, z_3) &= \big(-L + 2L\cos(\pi t)^2(1-t), -S + 2S\sin(\pi t)^2 t, -H + 2H\cos(\pi t)^2(1-t)\big) \\
(x_4, y_4, z_4) &= \big(-L + 2L\cos(\pi t)^2(1-t), -S + 2S\cos(\pi t)^2(1-t), -H + 2H\sin(\pi t)^2 t)\big) \\
(x_5, y_5, z_5) &= \big(-L + 2L\cos(2\pi t)^2\cos(\pi/2t), -S + 2S\sin(\pi\, t)^2\sin(\pi/2t), \\
&\quad -H + 2H\sin(\pi t)^2\sin(\pi/2t)\big) \\
(x_6, y_6, z_6) &= \big(-L + 2L\sin(\pi t)^2\sin(\pi/2t), -S + 2S\cos(2\pi\, t)^2\cos(\pi/2t), \\
&\quad -H + 2H\sin(\pi t)^2\sin(\pi/2t)\big) \\
(x_7, y_7, z_7) &= \big(-L + 2L\sin(\pi t)^2\sin(\pi/2t), -S + 2S\sin(\pi\, t)^2\sin(\pi/2t), \\
&\quad -H + 2H\cos(2\pi t)^2\cos(\pi/2t)\big) \\
(x_8, y_8, z_8) &= \big(-L + 2L\sin(\pi t)^2\sin(\pi/2t), -S + 2S\cos(2\pi\, t)^2\cos(\pi/2t), \\
&\quad -H + 2H\cos(2\pi t)^2\cos(\pi/2t)\big).
\end{aligned}$$

We shall use the mesh $64 \times 64 \times 64$ and the time step size $1/47$, with two regularization parameters $\beta_1 = 3.6 \times 10^{-5}$ and $\beta_2 = 3.6 \times 10^{-1}$. We compare the results recovered

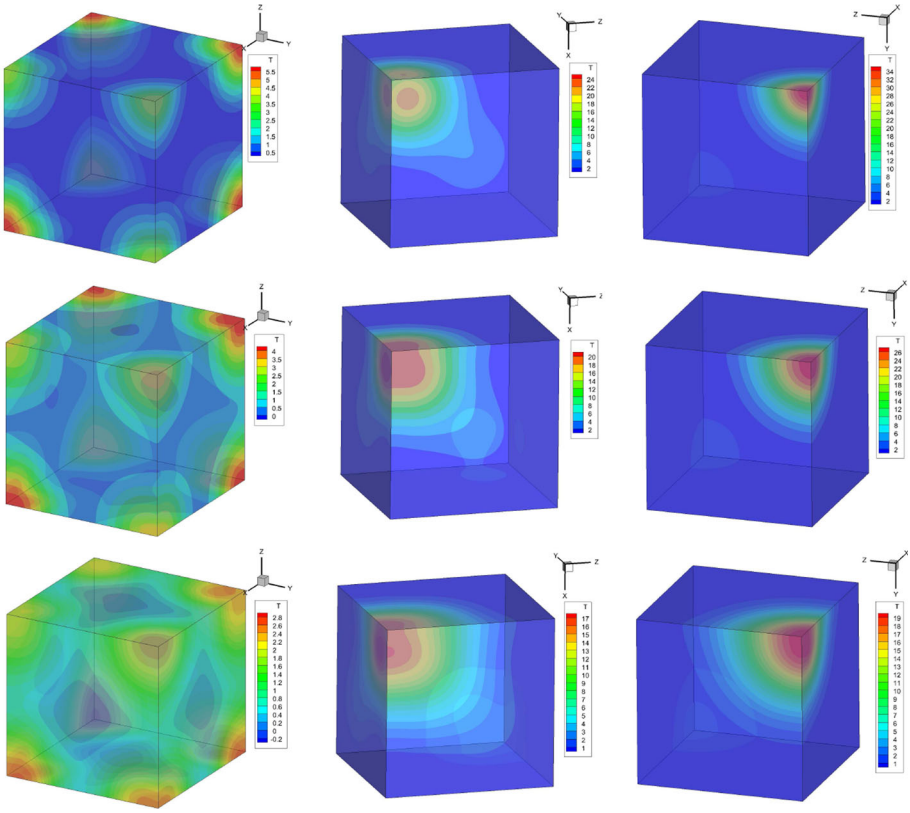

Fig. 6 Example 3: the source reconstructions with measurements collected at the mesh $22 \times 22 \times 22$ (*mid*) and $10 \times 10 \times 10$ (*bottom*), compared with the exact solution (*top*)

by two sets of measurements, collected on two meshes $22 \times 22 \times 22$ and $10 \times 10 \times 10$, with the mesh sizes $1/21$ and $1/9$ respectively. The solution is shown in Fig. 6, at three moments $t = 0.0, 10/47, 1.0$. Clearly better reconstructions are observed for the case with more measurements collected at the finer mesh $22 \times 22 \times 22$, though the coarser mesh $10 \times 10 \times 10$ is good enough for locating the sources, only with their recovered source intensities smaller than the true values.

4.2 Performance in Parallel Efficiency

In the previous subsection, we have shown with 3 representative examples that the proposed algorithm can successfully recover the intensities and distributions of unsteady sources and is robust with respect to the noise in the data, the choice of Tikhonov regularizations and the number of measurements. These numerical simulations are all computed using the proposed one-level space–time method with $np = 256$ processors. In this section, we focus on our proposed two-level space–time method and study its parallel efficiency with respect to the number of ILU fill-in levels, namely the number k in ILU(k), the overlap size $iovlp$ on the fine level, and the mesh size on the coarse level. We also compare the number of iterations

Table 3 Effects of ILU fill-in levels on the two-level method for Example 1 (columns 2–3), Example 2 (columns 4–5), and Example 3 (columns 6–7)

ILU(k)	Ex1		Ex2		Ex3	
	Its	Time (s)	Its	Time (s)	Its	Time (s)
0	47	10.498	55	12.448	81	17.238
1	28	33.633	36	47.766	60	49.622
2	18	230.552	23	232.914	48	257.798
3	15	1121.469	20	1132.841	45	1165.203

Table 4 Effects of the overlap size on the two-level method for Example 1 (columns 2–3), Example 2 (columns 4–5), and Example 3 (columns 6–7)

iovlp	Ex1		Ex2		Ex3	
	Its	Time (s)	Its	Time (s)	Its	Time (s)
1	47	10.498	55	12.448	81	17.238
2	39	13.071	51	23.663	69	27.952
4	37	27.423	49	45.225	68	47.032

and the total compute time of the one-level and two-level methods with increasing degrees of freedoms (DOFs) and the number of processors.

Firstly we test how the number of fGMRES iterations and the total compute time of the two-level method change with different ILU fill-in levels. We use the coarse mesh $21 \times 21 \times 21$ with the time step $1/20$, and the fine mesh $41 \times 41 \times 41$ with the time step $1/40$ for Examples 1, 2, and 3, and the overlap size $iovlp = 1$. We see that the total number of DOFs on the fine mesh is 16 times of the one on the coarse mesh. Table 3 shows the comparison with $np = 256$ processors. Column 2–3, 4–5 and 6–7 present the results for Examples 1, 2 and 3 respectively. It is observed that as the fill-in level increases the number of fGMRES iterations decreases, but the total compute time increases. When the fill-in level increases to 3, the compute time increases significantly and the number of iterations only reduces by 3 times. This suggests a suitable fill-in level to be $ilulevel = 0$ or 1.

Next we look at the impact of the overlap size. We still use the same fine and coarse meshes for all examples, and ILU(0) for the solver for each subdomain problem on both the coarse and fine meshes. The overlap size on the coarse mesh is set to be 1. We test different overlap sizes on the fine level, and the results are given in Table 4. It is observed that when the overlap size increases from 1 to 2 and then to 4, the number of fGMRES iterations decreases slowly and the total compute time increases. So we shall mostly use $iovlp = 1$ in our subsequent computations.

It is well known that the size of the coarse mesh is an important factor for a two-level method. Now we investigate the performance of our two-level method with different coarse meshes. We know that if the mesh is too coarse, both the number of outer iterations and the total compute time increase; on the other hand, if the mesh is not coarse enough, too much time is spent on the coarse solver, the number of outer iterations may decrease significantly, but the compute time may increase. For this experiment, we fix the fine mesh $43 \times 43 \times 43$ with the time step $1/42$, and the coarse mesh size in each direction is set to $1/2$ or $1/3$ of the fine mesh. We combine these options and obtain four sets of coarse meshes and their corresponding time steps. If we denote the ratios of the DOFs on the fine mesh compared to that on the coarse mesh by n, then $n = 8$, 16, 24 and 27 for these coarse meshes respectively. $np = 256$ processors are used. The fill-in level of the subdomain ILU solver is $ilulevel = 0$

Table 5 Effects of the coarse mesh size on the two-level method for Example 1 (columns 4–5), Example 2 (columns 6–7), and Example 3 (columns 8–9)

Coarse mesh	M	n	Ex1		Ex2		Ex3	
			Its	Time (s)	Its	Time (s)	Its	Time (s)
$22 \times 22 \times 22$	43	8	46	13.754	43	13.236	50	17.207
$22 \times 22 \times 22$	22	16	49	12.967	50	13.174	69	16.035
$22 \times 22 \times 22$	15	24	52	13.185	53	13.355	77	19.371
$15 \times 15 \times 15$	43	27	58	14.677	58	14.706	92	20.515

and the size of overlap on both the coarse and fine level is $iovlp = 1$. The computational results presented in Table 5 indicate that basically the number of fGMRES iterations increases when we decrease the coarse mesh size, however the compute time does not necessarily follow this trend, it decreases when the coarse mesh is fixed at $22 \times 22 \times 22$ and the time step is reduced from $1/42$ to $1/21$, then for the next two coarse mesh settings, the compute time grows slowly. As a result, the proper coarse mesh for this example is $22 \times 22 \times 22$ with time step $1/21$, i.e. 16 times coarser is the optimal choice for this test case.

Lastly we compare the performance of the one-level and two-level space–time Schwarz preconditioners in Tables 6 and 7. On the coarse level, a restarted GMRES is used, with the one-level space–time Schwarz preconditioner. ILU(0) is used as the local preconditioner on each subdomain and the coarse overlap size is set to be 1. A tighter convergence tolerance on the coarse mesh can reduce the number of outer fGMRES iterations, but often increases the total compute time. In the following numerical examples, we set the tolerance to be 10^{-1} and the maximum number of GMRES iterations to four on the coarse mesh.

In the following experiments for Examples 1, 2 and 3, we use three sets of fine meshes, $33 \times 33 \times 33$, $49 \times 49 \times 49$ and $67 \times 67 \times 67$, and the corresponding time steps are $1/32$, $1/48$ and $1/66$ respectively, while the coarse meshes are chosen to be $17 \times 17 \times 17$, $17 \times 17 \times 17$ and $23 \times 23 \times 23$, with the corresponding time steps being $1/16$, $1/48$ and $1/66$. So the DOFs on the fine meshes are 16, 27 and 27 times of the ones on the coarse meshes for Examples 1, 2 and 3 respectively. We use $np = 64$, 128 and 512 processors for the three sets of meshes respectively and compare their performance with the one-level method in Table 6. Savings in terms of the number of iterations and the total compute time are obtained for the two-level method with all three sets of meshes. As we observe that the number of iterations of the two-level method is mostly reduced by at least 4 times compared to the one for the one-level method, but the compute time is usually reduced by 2–4 times.

Next we fix the space mesh to be $49 \times 49 \times 49$ and the time step to be $1/48$, resulting in a very large-scale discrete system with 17,294,403 DOFs. For the two-level method, we set the coarse mesh to be $17 \times 17 \times 17$ with the time step $1/48$, which implies that the DOFs on the fine mesh is about 27 times of the ones on the coarse mesh. Then the problem is solved with $np = 128$, 256, 512, and 1024 processors respectively. The performance results of the one-level and two-level methods are presented in Table 7. We observe that when the number of sources is small, both the one-level and two-level methods are scalable with up to 512 processors, but the two-level method takes much less compute time. The strong scalability deteriorates when the number of processors is too large for the size of the problems. As the number of sources increases, the scalability becomes slightly worse for both one-level and two-level methods, even though the two-level method is still faster in terms of the total compute time.

Table 6 Comparisons between the one-level and two-level space–time preconditioners for Examples 1–3 with different meshes

np	Mesh	*M*	*level*	Its	Time (s)
Ex1					
64	$33 \times 33 \times 33$	33	1	175	53.635
			2	57	20.653
128	$49 \times 49 \times 49$	49	1	346	200.664
			2	83	47.812
512	$67 \times 67 \times 67$	67	1	491	675.985
			2	105	212.72
Ex2					
64	$33 \times 33 \times 33$	33	1	228	72.338
			2	77	20.246
128	$49 \times 49 \times 49$	49	1	365	214.058
			2	85	47.078
512	$67 \times 67 \times 67$	67	1	599	841.652
			2	121	216.92
Ex3					
64	$33 \times 33 \times 33$	33	1	297	82.834
			2	76	21.738
128	$49 \times 49 \times 49$	49	1	405	238.712
			2	93	57.244
512	$67 \times 67 \times 67$	67	1	716	872.766
			2	137	263.222

Table 7 Comparisons between the one-level and two-level space–time preconditioners for Examples 1–3 with different number of processors

np	*level*	Ex1		Ex2		Ex3	
		Its	Time (s)	Its	Time (s)	Its	Time (s)
128	1	346	200.664	365	214.815	405	238.712
	2	83	47.812	85	47.072	93	57.244
256	1	343	127.035	363	152.334	408	145.213
	2	82	24.744	87	26.424	90	36.307
512	1	343	69.482	363	95.707	400	101.343
	2	82	16.461	101	19.453	100	18.611
1024	1	351	41.821	393	58.785	433	59.534
	2	85	10.132	100	11.352	104	15.815

5 Concluding Remarks

In this work we have proposed and studied a fully implicit, space–time coupled, mixed finite element and finite difference discretization method, and a parallel one- and two-level domain decomposition solver for the three-dimensional unsteady inverse convection–diffusion prob-

lem. With a suitable number of measurements, this all-at-once approach provides acceptable reconstruction of the physical sources in space and time simultaneously. The classical overlapping Schwarz preconditioner is extended successfully to the coupled space–time problem with a homogenous Dirichlet boundary condition applied on both the spatial and temporal part of the space–time subdomain boundaries. The one-level method is easier to implement, but the two-level hybrid space–time Schwarz method performs much better in terms of the number of iterations and the total compute time. Good scalability results were obtained for problems with more than 17 millions degrees of freedom on a supercomputer with more than 1000 processors. The approach is promising to more general unsteady inverse problems in large-scale applications.

Acknowledgments The authors would like to thank the anonymous referees for their insightful comments and suggestions that helped us improve the quality of the paper. The work was partly supported by NSFC 11501545, 91330111, Shenzhen Program JCYJ20140901003939012, KQCX20130628112914303, 201506303000093 and 863 Program 2015AA01A302. The second author was partly support by NSF CCF-1216314. The third author was substantially supported by Hong Kong RGC Grants 404611 and 405513.

6 The Discrete Structure of the KKT System

The KKT system (8)–(9) is formulated as follows:

$$\begin{cases} \left(\partial_\tau C_h^n, v_h\right) + \left(a\nabla \bar{C}_h^n, \nabla v_h\right) + \left(\nabla\cdot\left(\mathbf{v}\bar{C}_h^n\right), v_h\right) = \left(\bar{f}_h^n, v_h\right) + \langle \bar{q}^n, v_h\rangle_{\Gamma_2}, \quad \forall\, v_h \in \mathring{V}^h \\ -\left(\partial_\tau G_h^n, w_h\right) + \left(a\nabla \bar{G}_h^n, \nabla w_h\right) + \left(\nabla\cdot(\mathbf{v}w_h), \bar{G}_h^n\right) \\ = -\left(A(\mathbf{x})\left(\bar{C}_h^n(\mathbf{x}) - \bar{C}^{\varepsilon,n}(\mathbf{x})\right), w_h\right), \quad \forall w_h \in \mathring{V}^h \\ -\left(G_h^n, g_h^\tau\right) + \beta_1\left(\partial_\tau f_h^n, \partial_\tau g_h^\tau\right) + \beta_2\left(\nabla f_h^n, \nabla g_h^\tau\right) = 0, \quad \forall\, g_h^\tau \in W_h^\tau. \end{cases} \tag{16}$$

To better understand the discrete structure of (16), we denote the identity and zero matrices as I and $\mathbf{0}$ respectively, and the basis functions of the finite element spaces V^h and W_h^τ by $\phi = (\phi_i)^T$, $i = 1, \ldots, N$ and g_j^n, $j = 1, \ldots, N$, $n = 0, \ldots, M$, respectively, let

$$\begin{aligned} A &= (a_{ij})_{i,j=1,\ldots,N}, \quad a_{ij} = (a\nabla\phi_i, \nabla\phi_j) \\ B &= (b_{ij})_{i,j=1,\ldots,N}, \quad b_{ij} = (\phi_i, \phi_j) \\ E &= (e_{ij})_{i,j=1,\ldots,N}, \quad e_{ij} = (\nabla\cdot(\mathbf{v}\phi_i), \phi_j) \\ L^{mn} &= (l_{ij}^{mn})_{i,j=1,\ldots,N,0\le m,n\le M}, \quad l_{ij}^{mn} = \left(\frac{\partial g_i^m}{\partial t}, \frac{\partial g_j^n}{\partial t}\right) \\ K^{mn} &= (k_{ij}^{mn})_{i,j=1,\ldots,N,0\le m,n\le M}, \quad k_{ij}^{mn} = \left(\nabla g_i^m, \nabla g_j^n\right) \\ D^{mn} &= (d_{ij}^{mn})_{i,j=1,\ldots,N,0\le m,n\le M}, \quad d_{ij}^{mn} = \left(g_i^m, g_j^n\right), \end{aligned}$$

and based on these element matrices we define

$$\begin{aligned} A_1 &= B + \frac{\tau}{2}(A+E), \quad A_2 = -B + \frac{\tau}{2}(A+E) \\ B_1 &= B + \frac{\tau}{2}\left(A + E^T\right), \quad B_2 = -B + \frac{\tau}{2}\left(A + E^T\right) \\ B_3 &= \text{zeros except 1 at the measurement locations} \\ W^{mn} &= \beta_1 L^{mn} + \beta_2 K^{mn}, \end{aligned}$$

Then the system (16) takes the following form

$$\begin{pmatrix} BC & BG & Bf \end{pmatrix} \begin{pmatrix} C^0 \\ C^1 \\ \vdots \\ C^{M-2} \\ C^{M-1} \\ C^M \\ G^0 \\ G^1 \\ G^2 \\ \vdots \\ G^{M-2} \\ G^{M-1} \\ G^M \\ f^0 \\ f^1 \\ f^2 \\ \vdots \\ f^{M-2} \\ f^{M-1} \\ f^M \end{pmatrix} = \begin{pmatrix} C^0 \\ \langle \bar{q}^1, \phi \rangle_{\Gamma_2} \\ \vdots \\ \langle \bar{q}^{M-1}, \phi \rangle_{\Gamma_2} \\ \langle \bar{q}^M, \phi \rangle_{\Gamma_2} \\ \tau/2 B_3 \left(C^{\varepsilon,0} + C^{\varepsilon,1} \right) \\ \vdots \\ \tau/2 B_3 \left(C^{\varepsilon,M-2} + C^{\varepsilon,M-1} \right) \\ \tau/2 B_3 \left(C^{\varepsilon,M-1} + C^{\varepsilon,M} \right) \\ G^M \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix},$$

where the block matrices BC, BG and Bf are given by

$$BC := \begin{pmatrix} I & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ A_2 & A_1 & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \ddots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & A_2 & A_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & A_2 & A_1 \\ \frac{\tau}{2} B_3 & \frac{\tau}{2} B_3 & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \ddots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \frac{\tau}{2} B_3 & \frac{\tau}{2} B_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \frac{\tau}{2} B_3 & \frac{\tau}{2} B_3 \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix},$$

$$BG := \begin{pmatrix}
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
B_1 & B_2 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \ddots & \ddots & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & B_1 & B_2 & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & B_1 & B_2 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & I \\
-D^{00} & -D^{01} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
-D^{10} & -D^{11} & -D^{12} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \ddots & \ddots & \ddots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & -D^{M-1,M-2} & -D^{M-1,M-1} & -D^{M-1,M} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & -D^{M,M-1} & -D^{MM}
\end{pmatrix}$$

$$Bf := \begin{pmatrix}
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
-\frac{\tau}{2}B & -\frac{\tau}{2}B & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \ddots & \ddots & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \ddots & \cdots & -\frac{\tau}{2}B & -\frac{\tau}{2}B & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & -\frac{\tau}{2}B & -\frac{\tau}{2}B \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
W^{00} & W^{01} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
W^{10} & W^{11} & W^{12} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \ddots & \ddots & \ddots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & W^{M-1,M-2} & W^{M-1,M-1} & W^{M-1,M} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & W^{M,M-1} & W^{MM}
\end{pmatrix}.$$

References

1. Aitbayev, R., Cai, X.-C., Paraschivoiu, M.: Parallel two-level methods for three-dimensional transonic compressible flow simulations on unstructured meshes. In: Proceedings of Parallel CFD'99 (1999)
2. Akcelik, V., Biros, G., Draganescu, A., Ghattas, O., Hill, J., Waanders, B.: Dynamic data-driven inversion for terascale simulations: real-time identification of airborne contaminants. In: Proceedings of Supercomputing, Seattle, WA (2005)
3. Akcelik, V., Biros, G., Ghattas, O., Long, K.R., Waanders, B.: A variational finite element method for source inversion for convective–diffusive transport. Finite Elem. Anal. Des. **39**, 683–705 (2003)
4. Atmadja, J., Bagtzoglou, A.C.: State of the art report on mathematical methods for groundwater pollution source identification. Environ. Forensics **2**, 205–214 (2001)
5. Baflico, L., Bernard, S., Maday, Y., Turinici, G., Zerah, G.: Parallel-in-time molecular-dynamics simulations. Phys. Rev. E. **66**, 2–5 (2002)
6. Balay, S., Buschelman, K., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Smith, B.F., Zhang, H.: PETSc users manual. In: Technical Report, Argonne National Laboratory (2014)

7. Battermann, A.: Preconditioners for Karush–Kuhn–Tucker Systems Arising in Optimal Control. In: Master Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia (1996)
8. Biros, G., Ghattas, O.: Parallel preconditioners for KKT systems arising in optimal control of viscous incompressible flows. In: Proceedings of Parallel CFD'99, Williamsburg, Virginia, USA (1999)
9. Cai, X.-C., Liu, S., Zou, J.: Parallel overlapping domain decomposition methods for coupled inverse elliptic problems. Commun. Appl. Math. Comput. Sci. **4**, 1–26 (2009)
10. Cai, X.-C., Sarkis, M.: A restricted additive Schwarz preconditioner for general sparse linear systems. SIAM J. Sci. Comput. **21**, 792–797 (1999)
11. Chen, R.L., Cai, X.-C.: Parallel one-shot Lagrange–Newton–Krylov–Schwarz algorithms for shape optimization of steady incompressible flows. SIAM J. Sci. Comput. **34**, 584–605 (2012)
12. Deng, X.M., Cai, X.-C., Zou, J.: A parallel space–time domain decomposition method for unsteady source inversion problems. Inverse Probl. Imag. (2015)
13. Deng, X.M., Zhao, Y.B., Zou, J.: On linear finite elements for simultaneously recovering source location and intensity. Int. J. Numer. Anal. Model. **10**, 588–602 (2013)
14. Engl, H.W., Hanke, M., Neubauer, A.: Regularization of Inverse Problems. Kluwer Academic Publishers, Dordrecht (1998)
15. Farhat, C., Chandesris, M.: Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications. Int. J. Numer. Methods Eng. **58**, 1397–1434 (2003)
16. Gander, M.J., Hairer, E.: Nonlinear convergence analysis for the parareal algorithm. In: Proceedings of the 17th International Conference on Domain Decomposition Methods, vol. 60, pp. 45–56 (2008)
17. Gander, M.J., Petcu, M.: Analysis of a Krylov subspace enhanced parareal algorithm for linear problems. In: Cances E. et al. (eds.) Paris-Sud Working Group on Modeling and Scientific Computing 2007–2008. ESAIM Proceedings of EDP Science, LesUlis, vol. 25, pp. 114–129 (2008)
18. Gander, M.J., Vandewalle, S.: Analysis of the parareal time-parallel time-integration method. SIAM J. Sci. Comput. **29**, 556–578 (2007)
19. Gorelick, S., Evans, B., Remson, I.: Identifying sources of groundwater pollution: an optimization approach. Water Resour. Res. **19**, 779–790 (1983)
20. Hamdi, A.: The recovery of a time-dependent point source in a linear transport equation: application to surface water pollution. Inverse Probl. **24**, 1–18 (2009)
21. Karalashvili, M., Groß, S., Marquardt, W., Mhamdi, A., Reusken, A.: Identification of transport coefficient models in convection–diffusion equations. SIAM J. Sci. Comput. **33**, 303–327 (2011)
22. Keung, Y.L., Zou, J.: Numerical identifications of parameters in parabolic systems. Inverse Probl. **14**, 83–100 (1998)
23. Kuhn, H.W., Tucker, A.W.: Nonlinear programming. In: Proceedings of 2nd Berkeley Symposium. University of California Press, Berkeley, pp. 481–492 (1951)
24. Lions, J.-L., Maday, Y., Turinici, G.: A parareal in time discretization of PDE's. C. R. Acad. Sci. Ser. I Math. **332**, 661–668 (2001)
25. Liu, X., Zhai, Z.: Inverse modeling methods for indoor airborne pollutant tracking literature review and fundamentals. Indoor Air **17**, 419–438 (2007)
26. Maday, Y., Turinici, G.: The parareal in time iterative solver: a further direction to parallel implementation. Domain Decompos. Methods Sci. Eng. **40**, 441–448 (2005)
27. Nilssen, T.K., Karlsen, K.H., Mannseth, T., Tai, X.-C.: Identification of diffusion parameters in a nonlinear convection–diffusion equation using the augmented Lagrangian method. Comput. Geosci. **13**, 317–329 (2009)
28. Prudencio, E., Byrd, R., Cai, X.-C.: Parallel full space SQP Lagrange–Newton–Krylov–Schwarz algorithms for PDE-constrained optimization problems. SIAM J. Sci. Comput. **27**, 1305–1328 (2006)
29. Revelli, R., Ridolfi, L.: Nonlinear convection–dispersion models with a localized pollutant source II—-a class of inverse problems. Math. Comput. Model. **42**, 601–612 (2005)
30. Saad, Y.: A flexible inner–outer preconditioned GMRES algorithm. SIAM J. Sci. Comput. **14**, 461–469 (1993)
31. Samarskii, A.A., Vabishchevich, P.N.: Numerical Methods for Solving Inverse Problems of Mathematical Physics. Walter de Gruyter, Berlin (2007)
32. Skaggs, T., Kabala, Z.: Recovering the release history of a groundwater contaminant. Water Resour. Res. **30**, 71–80 (1994)
33. Smith, B., Bjørstad, P., Gropp, W.: Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge University Press, Cambridge (2004)
34. Wong, J., Yuan, P.: A FE-based algorithm for the inverse natural convection problem. Int. J. Numer. Methods. Fluid **68**, 48–82 (2012)
35. Woodbury, K.A.: Inverse Engineering Handbook. CRC Press, Boca Raton (2003)

36. Yang, X.-H., She, D.-X., Li, J.-Q.: Numerical approach to the inverse convection-diffusion problem. In: 2007 International Symposium on Nonlinear Dynamics (2007 ISND), Journal of Physics: Conference Series, vol. 96,p. 012156 (2008)
37. Yang, L., Deng, Z.-C., Yu, J.-N., Luo, G.-W.: Optimization method for the inverse problem of reconstructing the source term in a parabolic equation. Math. Comput. Simul. **80**, 314–326 (2009)
38. Yang, H., Prudencio, E., Cai, X.-C.: Fully implicit Lagrange–Newton–Krylov–Schwarz algorithms for boundary control of unsteady incompressible flows. Int. J. Numer. Methods Eng. **91**, 644–665 (2012)
39. Zhang, J., Delichatsios, M.A.: Determination of the convective heat transfer coefficient in three-dimensional inverse heat conduction problems. Fire Saf. J. **44**, 681–690 (2009)