Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **200X**

doi:10.3934/xx.xx.xx.xx

pp. **X–XX**

NUMERICAL INVESTIGATION OF ENSEMBLE METHODS WITH BLOCK ITERATIVE SOLVERS FOR EVOLUTION PROBLEMS

Lili Ju

Department of Mathematics, University of South Carolina, Columbia, SC 29208

Wei Leng

State Key Laboratory of Scientific and Engineering Computing, Chinese Academy of Sciences
Beijing 100190, China

Zhu Wang* and Shuai Yuan

Department of Mathematics, University of South Carolina, Columbia, SC 29208

(Communicated by the associate editor name)

Abstract. The ensemble method has been developed for accelerating a sequence of numerical simulations of evolution problems. Its main idea is, by manipulating the time stepping and grouping discrete problems, to make all members in the same group share a common coefficient matrix. Thus, at each time step, instead of solving a sequence of linear systems each of which contains only one right-hand-side vector, the ensemble method simultaneously solves a single linear system with multiple right-hand-side vectors for each group. Such a system could be solved efficiently by using direct linear solvers when the problems are of small scale, as the same LU factorization would work for the entire group members. However, for large-scale problems, iterative linear solvers often have to be used and then this appealing advantage becomes not obvious. In this paper we systematically investigate numerical performance of the ensemble method with block iterative solvers for two typical evolution problems: the heat equation and the incompressible Navier-Stokes equations. In particular, the block conjugate gradient (CG) solver is considered for the former and the block generalized minimal residual (GMRES) solver for the latter. Our numerical results demonstrate the effectiveness and efficiency of the ensemble method when working together with these block iterative solvers.

2010 *Mathematics Subject Classification.* Primary: 65M60.

Key words and phrases. Ensemble method, ensemble-based time stepping, block CG, block GMRES, iterative linear solver.

The first author's research was partially supported by U.S. National Science Foundation under grant number DMS-1818438 and U.S. Department of Energy under grant numbers DE-SC0016540 and DE-SC0020270. The second author's research was partially supported the National Key Research and Development Program of China under grant number 2016YFB0201304, National Natural Science Foundation of China under grant numbers 91430215, 91530323, 11501553 and 11771440, State Key Laboratory of Scientific and Engineering Computing (LSEC), and National Center for Mathematics and Interdisciplinary Sciences of Chinese Academy of Sciences (NCMIS). The third author's research was partially supported by U.S. Department of Energy under grant numbers DE-SC0016540 and DE-SC0020270, U.S. National Science Foundation under grant number DMS-1913073 and Office of the Vice President for Research at the University of South Carolina through an ASPIRE grant.

* Corresponding author: Zhu Wang.

1. **Introduction.** It is common to run a sequence of numerical simulations in scientific and engineering application problems, such as numerical weather prediction using the ensemble-based data assimilation, proper orthogonal decomposition reduced order modeling that requires offline data generation, and uncertainty quantifications by random sampling approaches. For these numerical simulations, one needs to first discretize the problems in both space and time, and then solves a sequence of linear systems:

$$\mathbf{A}_j \mathbf{x}_j = \mathbf{b}_j, \tag{1}$$

for $j = 1, 2, \ldots, J$, and J is the total amount of simulations, named the ensemble size; $\mathbf{A}_j$, $\mathbf{x}_j$ and $\mathbf{b}_j$ are respectively the coefficient matrix, state variable vector and right-hand-side (RHS) vector in the j-th discrete problem. The choice of linear solvers is usually determined by the size and structure of $\mathbf{A}_j$. It is well known that dense direct methods are limited by the size of the problem, and work well for sizes up to a few thousands; and sparse direct methods, depending on both size and sparsity pattern, require good ordering. When the fill-in generated by sparse direct solvers becomes too costly, iterative methods have to be used. Their computational complexity depends on the size, sparsity of the coefficient matrix, and preconditioning is usually applied in order to accelerate the convergence [37].

When all simulations in the sequence possess a common coefficient matrix $\mathbf{A}$ (that is, $\mathbf{A}_j = \mathbf{A}$ for all j), direct methods can easily share information among all RHS vectors: one factorization of $\mathbf{A}$ could be used in solving all the problems. If the problem size is too big for direct solvers, block iterative methods have to be used, which are also able to share system information among all RHS vectors by using the same Krylov subspace, and solve all the linear systems simultaneously. For sequences sharing a common coefficient matrix, block iterative algorithms, such as block CG and its variants for symmetric positive definite (SPD) system [28, 29, 25, 19], block GMRES and its variants for nonsymmetric systems [35, 13, 27, 18, 7, 4, 3, 1] and hybrid block algorithms [36], have been developed to solve such system with many RHS vectors. The block algorithms have been used to accelerate the solution even for linear systems with only one RHS vector in [29, 6]. The advantage of a block solver over individual ones lies in the following facts: (i) the product of a matrix and J vectors is more efficient than J times matrix-vector products [8, 12]; (ii) the search space generated from J RHS vectors is usually larger than that from one single vector, thus the Krylov subspace method could potentially converge in fewer iterations [31]; (iii) when some of the RHS vectors are dependent, the search space could be compressed and problems can be solved more efficiently; and (iv) it only accesses the coefficient matrix once a time for J problems; when accessing $\mathbf{A}$ represents a main computational bottleneck of a linear solver, or $\mathbf{A}$ needs to re-generate at each time step, this leads to a significant computational advantage [2]. Several block iterative algorithms were studied using parallel computing in [32], which shows that they have a high level of parallelization and, thus, can be applied for solving large-scale multisource problems.

The appearance of a common coefficient matrix for a group of problems is appealing, however, in general, the coefficient matrix would vary from one problem to another, then it becomes unapparent on how to share the information among RHS vectors. Hence, neither direct nor block iterative solvers can be straightforwardly applied. Approaches such as the seed/recycled Krylov subspace methods have been developed, which solve each RHS independently, while storing some information from the solve and using it in subsequent solves [5]. The accumulated information

enlarges the search space, thus would potentially reduce the number of iterations. For slowly-changing linear systems, subspace recycling techniques such as GCRO with deflated restarting (GCRO-DR) have been introduced for accelerating the solution process in [30]. The block version of GCRO-DR was recently introduced in [31], and its high-performance implementation is available in the Belos package of the Trilinos project developed at Sandia National Laboratories. Note that the underlying assumption of such approaches is that all the systems are closely related, which certainly holds in some applications, but is in lack of rigorous mathematical foundation.

As seen from the preceding discussion, the research for accelerating a sequence of numerical simulations mainly starts from the numerical linear algebra's point of view and the goal is to solve (1) more efficiently when $\mathbf{A}_j$ varies from one case to another. Recently, the ensemble method has been introduced to tackle this issue for evolution problems at the numerical algorithm level [21, 22, 20, 16, 26, 14, 15, 11]. The idea is to ensure all the linear systems in a group to share a common coefficient matrix by manipulating numerical discretization schemes. Then, either direct or block iterative solvers could be naturally applied. Such an approach would lead to the following system:

$$\mathbf{AX} = \mathbf{B}, \tag{2}$$

where $\mathbf{A}$ is the common coefficient matrix, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_J]$ and $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_J]$ consist of state variable vectors and RHS vectors from J discrete problems, respectively. To the best of our knowledge, the research investigations on ensemble methods primarily focus on the numerical analysis, including stability analyses and error estimates so far. The resultant linear systems are solved using direct solvers, but not with block iterative ones. For instance, the ensemble-based Monte-Carlo and multi-level Monte-Carlo methods have been developed in [23, 24]. With the LU factorization, it has been shown that the use of ensemble methods leads to significant computational savings over the individual simulations. But large-scale applications certainly need to be considered in practice. Hence, in this paper, we take a couple of widely used PDE models in heat transfer and incompressible fluid flows, and *for the first time*, perform a comprehensive study on numerical behaviors of the ensemble methods together with block iterative solvers.

The rest of this paper is organized as follows. In Section 2, we briefly review the ensemble-based time-stepping algorithms for the heat equation and the Navier-Stokes equations. Block iterative solvers are discussed in Section 3. Several numerical experiments are then performed and presented in Section 4 to demonstrate the effectiveness and efficiency of the ensemble method working with the block iterative solvers. Finally some concluding remarks are drawn in Section 5.

2. Ensemble method for evolution problems.

We consider two popular mathematical models governing heat transfer and incompressible fluid flows respectively: the heat equation and the Navier-Stokes equations. Assume that, for either case, one needs to complete J numerical simulations under different computational settings including distinct body source functions, boundary and initial conditions, and physical parameters. We first present the ensemble time-stepping schemes for the aforementioned mathematical models and discuss the associated stability conditions and error estimates.

For simplicity of presentation, we assume the diffusion parameter ν_j, in the j-th problem, to be a constant function. The finite element (FE) method is used for

the spatial discretization in this paper, but other types of numerical methods could be used as well. A uniform mesh $\mathcal{T}^h$ with size h, and a uniform time partition with the time step size Δt are taken throughout our discussion. Denoted by Ω the computational domain, by Γ_D the boundary of Ω on which the Dirichlet condition is imposed and by Γ_N where the Neumann boundary condition is imposed. We use $(\cdot,\cdot)$ for the L^2-inner product on Ω, $\langle\cdot,\cdot\rangle_{\Gamma_i}$ for the inner product on Γ_i, and $\|\cdot\|$ for the L^2 norm. Define the spaces

$$V_{g_D} := \{u \in H^1(\Omega) \mid u = g_D \text{ on } \Gamma_D\}$$

and $V^h_{g_D}$ the space of piecewise continuous functions on Ω that reduce to polynomials of degree $\leq m$ on each element of $\mathcal{T}^h$. Additionally, we assume the total number of time steps to be N and adopt the following notations: for $n = 0, \ldots, N$,

$$t_n = n\Delta t, \quad u_j^n = u_j(t_n), \quad f_j^n = f_j(t_n), \quad g_{N,j}^n = g_{N,j}(t_n),$$

with

$$\begin{aligned} \overline{u}^n &= \frac{1}{J}\sum_{j=1}^{J} \widetilde{u}_j^n, \quad u_j'^n = \widetilde{u}_j^n - \overline{u}^n, \\ \overline{\nu} &= \frac{1}{J}\sum_{j=1}^{J} \nu_j, \quad \nu' = \nu_j - \overline{\nu}, \quad \text{and} \quad |\nu_j - \overline{\nu}|_\infty = \max_j |\nu_j - \overline{\nu}|. \end{aligned} \tag{3}$$

and time discretizations

$$\text{[first order: backward Euler]} \qquad \widetilde{u}_j^n = u_j^n, \quad D_t u_j^{n+1} = \tfrac{u_j^{n+1} - u_j^n}{\Delta t}; \tag{4}$$

$$\text{[second order: BDF2]} \qquad \widetilde{u}_j^n = 2u_j^n - u_j^{n-1}, \quad D_t u_j^{n+1} = \tfrac{3u_j^{n+1} - 4u_j^n + u_j^{n-1}}{2\Delta t} \tag{5}$$

2.1. **Heat equation.** For a group of heat transfer problems, the governing heat equation describes the distribution of temperature in a given region over time: to find the scalar unknown function $u_j(\mathbf{x}, t)$, for $j = 1, \ldots, J$, satisfying

$$\begin{cases} \frac{\partial u_j}{\partial t} - \nabla\cdot(\nu_j \nabla u_j) &= f_j(\mathbf{x},t) & \text{in } \Omega \times [0,T], \\ u_j &= g_{D,j}(\mathbf{x},t) & \text{on } \Gamma_D \times [0,T], \\ \frac{\partial u_j}{\partial n} &= g_{N,j}(\mathbf{x},t) & \text{on } \Gamma_N \times [0,T], \\ u_j(\mathbf{x},0) &= u_j^0(\mathbf{x}) & \text{in } \Omega. \end{cases} \tag{6}$$

In the ensemble-based time stepping, we introduce the ensemble average of diffusion coefficients and discretize the equation in an implicit-explicit manner. The resulting semi-discrete system reads:

$$D_t u_j^{n+1} - \nabla\cdot(\overline{\nu}\nabla u_j^{n+1}) = f_j^{n+1} + \nabla\cdot(\nu'\nabla\widetilde{u}_j^n), \tag{7}$$

associated with the same boundary and initial conditions, provided $\frac{\partial \widetilde{u}_j^n}{\partial n} := g_{N,j}^{n+1}$.

Using the standard conforming FE method, we take $V^h_{g_D,j}$ and V_0^h as the trail and test spaces, respectively, and obtain the fully discrete system: to find $u_{j,h}^{n+1} \in V^h_{g_D,j}$ such that

$$(D_t u_{j,h}^{n+1}, v) + (\overline{\nu}\nabla u_{j,h}^{n+1}, \nabla v) = (f_j^{n+1}, v) - (\nu'\nabla\widetilde{u}_{j,h}^n, \nabla v) + \langle \nu_j g_{N,j}^{n+1}, v\rangle_{\Gamma_N}, \ \forall\, v \in V_0^h. \tag{8}$$

The choices of D_t and $\widetilde{u}_{j,h}^n$, as specified in (4) or (5), lead to the ensemble-based time-stepping algorithms of first- or second-order accuracy, respectively. The scheme needs an initial condition $u_{j,h}^0$ to start with, for which the projection of u_j^0 onto the FE space can be taken. The second-order scheme is a two-step method that requires one more initial condition $u_{j,h}^1$, which could be obtained from the first-order scheme.

As for the stability condition and error estimate, we have the following results ([23, 24]):

Lemma 2.1. Suppose that $f_j \in L^2(H^{-1}(D); 0, T)$, the ensemble scheme (8) is stable if the following parameter deviation condition is satisfied:

$$\frac{|\nu_j - \overline{\nu}|_\infty}{\overline{\nu}} < 1 \quad \text{for the first-order scheme;} \tag{9}$$

or

$$\frac{|\nu_j - \overline{\nu}|_\infty}{\overline{\nu}} < \frac{1}{3} \quad \text{for the second-order scheme.} \tag{10}$$

Lemma 2.2. Let u_j^n and $u_{j,h}^n$ be the solutions of equations (6) and (8) at time t_n, respectively. Assume $f_j \in L^2(H^{-1}(\Omega); 0, T)$ and the stability condition, (9) or (10), holds. Then there exists a generic constant $C > 0$ independent of J, h and Δt such that

$$\|u_j^N - u_{j,h}^N\|^2 + \big(\overline{\nu} - |\nu_j - \overline{\nu}|_\infty\big)\Delta t \sum_{n=1}^{N} \|\nabla(u_j^n - u_{j,h}^n)\|^2 \le C(\Delta t^2 + h^{2m}) \tag{11}$$

for the first order scheme and

$$\frac{1}{4}\|u_j^N - u_{j,h}^N\|^2 + \big(\frac{\overline{\nu}}{3} - |\nu_j - \overline{\nu}|_\infty\big)\Delta t \sum_{n=1}^{N} \|\nabla(u_j^n - u_{j,h}^n)\|^2 \le C(\Delta t^4 + h^{2m}) \tag{12}$$

for the second order scheme.

2.2. **Navier-Stokes equations.** We next consider a group of incompressible flow problems, the governing Navier-Stokes equations (NSE) describe the motion of incompressible Newtonian fluid flows: to find the vector-valued velocity function $\mathbf{u}_j(\mathbf{x}, t)$ and the scalar function $p_j(\mathbf{x}, t)$, for $j = 1, \ldots, J$, satisfying

$$\begin{cases} \frac{\partial \mathbf{u}_j}{\partial t} + \mathbf{u}_j \cdot \nabla \mathbf{u}_j - \nabla \cdot (\nu_j \nabla \mathbf{u}_j) + \nabla p_j &= \mathbf{f}_j(\mathbf{x}, t) & \text{in } \Omega \times [0, T], \\ \nabla \cdot \mathbf{u}_j &= 0 & \text{in } \Omega \times [0, T], \\ \mathbf{u}_j &= \mathbf{g}_j^D(\mathbf{x}, t) & \text{on } \Gamma_D \times [0, T], \\ \mathbf{u}_j(\mathbf{x}, 0) &= \mathbf{u}_j^0(\mathbf{x}) & \text{in } \Omega. \end{cases} \tag{13}$$

After introducing ensemble averages of diffusion coefficient and velocity field and using the ensemble-based time stepping, we achieve the following semi-discrete system associated with the same boundary and initial conditions of (13):

$$\begin{cases} D_t \mathbf{u}_j^{n+1} + \overline{\mathbf{u}}^n \cdot \nabla \mathbf{u}_j^{n+1} - \nabla \cdot (\overline{\nu} \nabla \mathbf{u}_j^{n+1}) + \nabla p_j^{n+1} \\ \qquad\qquad = \mathbf{f}_j^{n+1} - (\widetilde{\mathbf{u}}_j^n - \overline{\mathbf{u}}^n) \cdot \nabla \widetilde{\mathbf{u}}_j^n + \nabla \cdot \big((\nu_j - \overline{\nu}) \nabla \widetilde{\mathbf{u}}_j^n\big), \\ \nabla \cdot \mathbf{u}_j^{n+1} = 0. \end{cases} \tag{14}$$

Define the skew-symmetric trilinear form

$$b^*(\mathbf{u}, \mathbf{v}, \mathbf{w}) := \frac{1}{2}(\mathbf{u} \cdot \nabla \mathbf{v}, \mathbf{w}) - \frac{1}{2}(\mathbf{u} \cdot \nabla \mathbf{w}, \mathbf{v}).$$

Let

$$Q := L_0^2(\Omega) = \left\{ q \in L^2(\Omega) \Big| \int_\Omega q \, dx = 0 \right\},$$

and Q^h is the space of piecewise continuous functions on Ω that reduce to polynomials of degree $\le s$ on each element of $\mathcal{T}^h$. Assume that, in order to guarantee the stability of FE approximations, the pair of spaces (V_0^h, Q^h) satisfies the discrete inf-sup (or LBB^h) condition. One example for which the LBB^h stability condition

holds is the family of Taylor-Hood P^{s+1}-P^s element pairs (i.e., $m = s+1$ in the definition of V_0^h), for $s \geq 1$ [17]. The fully discrete system reads: to find $\mathbf{u}_{j,h}^{n+1} \in V_{g_D,j}^h$ and $p_{j,h}^{n+1} \in Q^h$ such that

$$\begin{cases} \left(D_t \mathbf{u}_{j,h}^{n+1}, \mathbf{v}_h\right) + b^*(\overline{\mathbf{u}}_h^n, \mathbf{u}_{j,h}^{n+1}, \mathbf{v}_h) + (\overline{\nu}\nabla \mathbf{u}_{j,h}^{n+1}, \nabla \mathbf{v}_h) - (p_{j,h}^{n+1}, \nabla \cdot \mathbf{v}_h) \\ \qquad = (\mathbf{f}_j^{n+1}, \mathbf{v}_h) - b^*(\widetilde{\mathbf{u}}_{j,h}^n - \overline{\mathbf{u}}_h^n, \widetilde{\mathbf{u}}_{j,h}^n, \mathbf{v}_h) - \left((\nu_j - \overline{\nu})\nabla \widetilde{\mathbf{u}}_{j,h}^n, \nabla \mathbf{v}_h\right), \forall\, \mathbf{v}_h \in V_0^h, \\ \left(\nabla \cdot \mathbf{u}_{j,h}^{n+1}, q_h\right) = 0, \quad \forall\, q_h \in Q^h. \end{cases} \tag{15}$$

The choices of D_t and $\widetilde{\mathbf{u}}_{j,h}^n$, as specified in (4) or (5), respectively lead to the ensemble-based time stepping of first- or second-order accuracy in time. The initial condition $\mathbf{u}_{j,h}^0$ can be obtained by projecting $\mathbf{u}_j^0$ onto the FE space, while the second initial condition required for starting the second-order scheme can be sought by the first-order scheme.

It is seen that the ensemble method is semi-implicit. It first approximates the advection using the a priori known quantity $\overline{\mathbf{u}}_h^n$ and approximates the diffusion using the viscosity average $\overline{\nu}$, and then treats the remainders explicitly, therefore, leads to a single coefficient matrix for all the members in the group. It could be shown that the following stability conditions and error estimates hold [16, 15], under the regularity assumptions on the NSE given by

$$\begin{aligned} &\mathbf{u}_j \in H^2\left(0, T; H^{k+1}(\Omega)\right) \cap H^3\left(0, T; H^1(\Omega)\right), \\ &p_j \in L^2\left(0, T; H^{s+1}(\Omega)\right), \text{and } \mathbf{f}_j \in L^2\left(0, T; L^2(\Omega)\right). \end{aligned}$$

Lemma 2.3. The ensemble scheme (15) is stable for all $j = 1, \ldots, J$, if there exists some μ, $0 \leq \mu < 1$, and the following time-step condition and parameter deviation condition are satisfied:

$$C\frac{\Delta t}{h\overline{\nu}} \left\|\nabla \mathbf{u}_{j,h}'^n\right\|^2 < 1 - \sqrt{\mu} \tag{16}$$

and

$$\frac{|\nu_j - \overline{\nu}|_\infty}{\overline{\nu}} \leq \sqrt{\mu} \quad \text{for the first-order scheme,} \tag{17}$$

or

$$\frac{|\nu_j - \overline{\nu}|_\infty}{\overline{\nu}} \leq \frac{\sqrt{\mu}}{3} \quad \text{for the second-order scheme.} \tag{18}$$

Lemma 2.4. Suppose that the P^2-P^1 Taylor-Hood FE pair is used for the spatial discretization. For the first-order ensemble time-stepping, assume that $\|\mathbf{u}_j^0 - \mathbf{u}_{j,h}^0\|$, $\|\nabla(\mathbf{u}_j^0 - \mathbf{u}_{j,h}^0)\|$ are both $\mathcal{O}(h)$ accurate or better, then we have

$$\frac{1}{2}\|\mathbf{u}_j^N - \mathbf{u}_{j,h}^N\|^2 + C\overline{\nu}\Delta t \|\nabla\left(\mathbf{u}_j^N - \mathbf{u}_{j,h}^N\right)\|^2 \leq C(h^2 + \Delta t^2 + h\Delta t).$$

For the second-order ensemble time-stepping, assume that the initial errors $\|\mathbf{u}_j^0 - \mathbf{u}_{j,h}^0\|$, $\|\nabla(\mathbf{u}_j^0 - \mathbf{u}_{j,h}^0)\|$, $\|\mathbf{u}_j^1 - \mathbf{u}_{j,h}^1\|$ and $\|\nabla(\mathbf{u}_j^1 - \mathbf{u}_{j,h}^1)\|$ are all at least $\mathcal{O}(h^2)$ accurate, then the approximation error of the ensemble scheme at time t_N satisfies

$$\frac{1}{4}\|\mathbf{u}_j^N - \mathbf{u}_{j,h}^N\|^2 + C\overline{\nu}\Delta t \|\nabla\left(\mathbf{u}_j^N - \mathbf{u}_{j,h}^N\right)\|^2 \leq C(h^4 + \Delta t^4 + h\Delta t^3), \tag{19}$$

where C is a generic positive constant which does not depend on J, Δt, and h.

3. **Block-based iterative solvers.** Since the ensemble method leads to a single system (2) for the entire group, direct methods can be used to efficiently solve it, as only one single LU factorization is needed for advancing all the simulations at each time step. However, for large-scale problems, iterative linear solvers have to be used due to the memory restriction and computational cost. Because of the different algebraic structures in discrete heat equation and discrete Navier-Stokes equations, we will investigate and discuss block-based iterative algorithms for them separately in this section.

3.1. **Solution to discrete heat equation.** Assume that there are locally n_u degrees of freedom for each of the J problems and define the basis functions to be $\{\psi_1(x), \psi_2(x), \ldots, \psi_{n_u}(x)\}$. Denote the approximate solution of j-th problem by $u_{j,h}(t,x) = \sum_{i=1}^{n_u} u_i^{(j)}(t)\psi_i(x)$. The mass and stiffness matrices are defined by $\mathbb{M} = [m_{ik}]$ and $\mathbb{S} = [s_{ik}]$ with elements $m_{ik} = \int_\Omega \psi_k\psi_i\, dx$, and $s_{ik} = \int_\Omega \nabla\psi_k\cdot\nabla\psi_i\, dx$. At time t_n, the vector related to the source term and boundary term is given by $\mathbf{h}_j^n = [h_i^{(j)}]$ with entries $h_i^{(j)} = \int_\Omega f_j^n\psi_i dx + \langle \nu g_{N,j}^n, \psi_i\rangle$, and the approximate solution vector is denoted by $\mathbf{u}_j^n$.

The full discretization of the group of heat equations based on the ensemble method, (8), yields the following linear system:

$$\mathbb{A}\mathbb{U}^{n+1} = \mathbb{B}^{n+1}, \tag{20}$$

where $\mathbb{A}$ is a sparse SPD, $n_u \times n_u$ matrix and $\mathbb{B}^{n+1}$ is a $n_u \times J$ matrix (or so-called the block vectors in the literature). Furthermore,

$$\begin{aligned} \mathbb{A} &= \frac{1}{\Delta t}\mathbb{M} + \overline{\nu}\mathbb{S}, \\ \mathbb{B}^{n+1} &= [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_J] \text{ with } \mathbf{b}_j = \mathbf{h}_j^{n+1} - \nu_j'\mathbb{S}\mathbf{u}_j^n + \frac{1}{\Delta t}\mathbb{M}\mathbf{u}_j^n \end{aligned}$$

for the first-order scheme; and

$$\begin{aligned} \mathbb{A} &= \frac{3}{2\Delta t}\mathbb{M} + \overline{\nu}\mathbb{S}, \\ \mathbb{B}^{n+1} &= [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_J] \text{ with } \mathbf{b}_j = \mathbf{h}_j^{n+1} - \nu_j'\mathbb{S}\widetilde{\mathbf{u}}_j^n + \frac{4}{2\Delta t}\mathbb{M}\mathbf{u}_j^n - \frac{1}{2\Delta t}\mathbb{M}\mathbf{u}_j^{n-1} \end{aligned}$$

for the second-order scheme.

If one has access to a semi-implicit numerical solver (the coefficient matrix $\mathbb{A}^{(j)}$ and vector $\mathbf{b}^{(j)}$) for any individual simulation, then it is straightforward to obtain the linear system for the aforementioned ensemble method. Take the first-order semi-implicit scheme for example, the coefficient matrix $\mathbb{A}^{(j)} = \frac{1}{\Delta t}\mathbb{M} + \nu_j\mathbb{S}$ and vector $\mathbf{b}^{(j)} = \mathbf{h}_j^{n+1} + \frac{1}{\Delta t}\mathbb{M}\mathbf{u}_j^n$ can be extracted from individual simulation codes for $j = 1, \ldots, J$, then the matrices in ensemble simulations are: $\mathbb{A} = \frac{1}{J}\sum_{j=1}^J \mathbb{A}^{(j)}$ and $\mathbb{B}^{n+1} = [\mathbf{b}_1, \ldots, \mathbf{b}_J]$ with $\mathbf{b}_j = \mathbf{b}^{(j)} + (\mathbb{A} - \mathbb{A}^{(j)})\mathbf{u}_j^n$. Meanwhile, $\mathbb{A}$ inherits the same sparse, SPD structure of $\mathbb{A}^{(j)}$.

Among the iterative solvers in the general family of Krylov subspace methods, the conjugate gradient method (CG) developed by Hestenes and Stiefel is the most well known for solving a real SPD system [37]. For a SPD system with multiple RHS vectors, a block conjugate gradient method (BCG) has been developed as a generalization of CG in this context in [28] and further in [29, 25]. Comparing with CG, BCG has the advantage of potentially faster convergence because the search space is augmented when multiple problems are considered together instead

of a single problem. It becomes more attractive when accessing $\mathbb{A}^{(j)}$ represents the main computational bottleneck of a linear solver (e.g. when $\mathbb{A}^{(j)}$ is stored outside of the system memory, or the elements of $\mathbb{A}^{(j)}$ have to be regenerated at each use) as it can explore multiple search directions in a single pass over $\mathbb{A}$. However, in practice, BCG could fail due to the rank deficiency issues for which the block search direction vectors become linearly dependent. A simple solution was developed in [19], which extracts a set of basis vectors from the search space at each iteration and uses them as new search directions, thus could avoid inverting a potentially non-full rank matrix. The algorithm is named breakdown-free BCG (BFBCG) and is presented in Algorithm 1.

Algorithm 1: The breakdown-free block CG [19]

Input: matrix $\mathbb{A} \in \mathcal{R}^{n\times n}$, matrix $\mathbb{B} \in \mathcal{R}^{n\times J}$, initial guess $\mathbb{X}_0 \in \mathcal{R}^{n\times J}$, preconditioner $\mathbb{K} \in \mathcal{R}^{n\times n}$, tolerance $\mathsf{tol} \in \mathcal{R}$ and maximum number of iteration $\mathsf{maxit} \in \mathcal{R}$.

Output: approximate solution $\mathbb{X}_s \in \mathcal{R}^{n\times J}$

$\mathbb{R}_0 = \mathbb{B} - \mathbb{A}\mathbb{X}_0$;
solve $\mathbb{K}\mathbb{Z}_0 = \mathbb{R}_0$;
$\mathbb{P}_0 = \mathsf{orth}(\mathbb{Z}_0)$;
for *i=0, ...,* maxit **do**
  $\mathbb{Q}_i = \mathbb{A}\mathbb{P}_i$;
  $\mathbb{T}_i = \mathbb{P}_i^\top \mathbb{Q}_i$;
  $\Theta_i = \mathbb{T}_i^{-1}(\mathbb{P}_i^\top \mathbb{R}_i)$;
  $\mathbb{X}_{i+1} = \mathbb{X}_i + \mathbb{P}_i\Theta_i$;
  $\mathbb{R}_{i+1} = \mathbb{R}_i - \mathbb{Q}_i\Theta_i$;
  if *converged* **then** exit;
  $\mathbb{Z}_{i+1} = \mathbb{K}^{-1}\mathbb{R}_{i+1}$;
  $\Lambda_i = -\mathbb{T}_i^{-1}(\mathbb{Q}_i^\top \mathbb{Z}_{i+1})$;
  $\mathbb{P}_{i+1} = \mathsf{orth}(\mathbb{Z}_{i+1} + \mathbb{P}_i\Lambda_i)$;
end
$\mathbb{X}_s = \mathbb{X}_{i+1}$.

Different from the original BCG, the breakdown-free BCG introduces an orthogonalization process, orth, which extracts an orthogonal basis $\mathbb{P}_i$ from the search space (denoted by $\mathcal{P}_i$). This helps to overcome situations in which two or more vector components in the residual matrix $\mathbb{R}_i$ are dependent because the lack of full rank in $\mathbb{R}_i$ would result in rank deficiency in $\mathbb{Z}_i$ and $\mathbb{P}_i$, and further fails the BCG method. Assume the rank of $\mathcal{P}_i$ is r_i, the resulting orthogonal matrix $\mathbb{P}_i \in \mathcal{R}^{n_u\times r_i}$. In Algorithm 1, the choice of Θ_i and Λ_i guarantees the column spaces of $\mathbb{R}_{i+1}$ is orthogonal to the search space $\mathcal{P}_i$, and $\mathcal{P}_{i+1}$ is conjugate to all previous search spaces.

At each iteration, the algorithm involves three matrix-matrix products

$$\mathbb{A}_{n_u\times n_u}(\mathbb{P}_i)_{n_u\times r_i}, \quad (\mathbb{P}_i)^\top_{r_i\times n_u}(\mathbb{Q}_i)_{n_u\times J}, \quad (\mathbb{Q}_i^\top)_{r_i\times n_u}(\mathbb{Z}_{i+1})_{n_u\times J},$$

three matrix updates that include three $n_u \times r_i$ matrix and $r_i \times J$ matrix products $\mathbb{P}_i\Theta_i$, $\mathbb{Q}_i\Theta_i$ and $\mathbb{P}_i\Lambda_i$, two solutions of linear systems with the coefficient matrix $(\mathbb{T}_i)_{r_i\times r_i}$, one solution of a linear system with coefficient matrix $\mathbb{M}_{n_u\times n_u}$, and an orthogonalization procedure. Suppose n_u is much bigger than r_i, and $\mathbb{M}$ is easy to invert as a preconditioner, the above mentioned matrix-matrix products and system

solvers require $\mathcal{O}(n_u J \max(\ell, r_i))$ flops, where ℓ is the number of nonzero entries in each row of $\mathbb{A}$. Special attention needs to be paid to the orthogonalization process orth, which could be implemented by the reduced (economy) SVD. However, in order to reduce the computational complexity, we choose the method of snapshots and drop the singular values smaller than 10^{-12}. To determine left singular vectors associated with the retained singular values, it only requires $\mathcal{O}(n_u J^2)$ flops, thus is comparable to the other operations in the algorithm.

The performance of the algorithm in the ensemble simulations will be investigated in Section 4, in which we choose the incomplete Cholesky factorization for preconditioning, but other types of preconditioners such as multigrid and domain decomposition can be certainly used as well.

3.2. **Solution to discrete Navier-Stokes equations.** Assume that there are locally n_u degrees of freedom for velocity and denote the basis functions by $\{\boldsymbol{\phi}_1(x), \boldsymbol{\phi}_2(x), \ldots, \boldsymbol{\phi}_{n_u}(x)\}$, and n_p local degrees of freedom for pressure with the basis functions defined by $\{\psi_1(x), \psi_2(x), \ldots, \psi_{n_p}(x)\}$. Let the approximate solution be $\mathbf{u}_{j,h}(t,x) = \sum_{i=1}^{n_u} \mathbf{u}_i^{(j)}(t)\boldsymbol{\phi}_i(x)$ and $p_{j,h} = \sum_{i=1}^{n_p} p_i^{(j)}(t)\psi_i(x)$. The mass and stiffness matrices for velocity are defined to be $\mathbb{M} = [m_{ik}]$ and $\mathbb{S} = [s_{ik}]$ with elements $m_{ik} = \int_\Omega \boldsymbol{\phi}_k \cdot \boldsymbol{\phi}_i \, dx$, and $s_{ik} = \int_\Omega \nabla\boldsymbol{\phi}_k : \nabla\boldsymbol{\phi}_i \, dx$. The source term is $\mathbf{h}_j^n = [h_i^{(j)}]$ with entries $h_i^{(j)} = \int_\Omega \mathbf{f}_j^n \cdot \boldsymbol{\phi}_i \, dx$. We also define the matrix $\mathbb{D} = [d_{ik}]$ with entries $d_{ik} = -\int_\Omega \psi_k \nabla\cdot\boldsymbol{\phi}_i \, dx$, the matrices from the discretization of the convection term $\overline{\mathbb{N}}^n = [n_{ik}]$ and $\mathbb{R}_j^n = [r_{ik}]$ with entries $n_{ik} = \int_\Omega (\overline{\mathbf{u}}_h^n \cdot \nabla\boldsymbol{\phi}_k)\cdot\boldsymbol{\phi}_i \, dx$ and $r_{ik} = \int_\Omega (\mathbf{u}_{j,h}'^n \cdot \nabla\boldsymbol{\phi}_k)\cdot\boldsymbol{\phi}_i \, dx$, respectively, and denote the approximate solution vector by $\mathbf{u}_j^n$.

The fully discrete linear system for the group of NSE problems based on the ensemble method, (15), reads

$$\begin{bmatrix} \mathbb{C} & \mathbb{D}^\intercal \\ \mathbb{D} & \mathbb{0} \end{bmatrix} \begin{bmatrix} \mathbb{U}^{n+1} \\ \mathbb{P}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbb{B}^{n+1} \\ \mathbb{0} \end{bmatrix}, \tag{21}$$

where

$$\begin{aligned} \mathbb{C} &= \frac{1}{\Delta t}\mathbb{M} + \overline{\mathbb{N}}^n + \overline{\nu}\mathbb{S}, \\ \mathbb{B}^{n+1} &= [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_J] \text{ with } \mathbf{b}_j = \mathbf{h}_j^{n+1} - \mathbb{R}_j^n \widetilde{\mathbf{u}}_j^n - \nu_j' \mathbb{S}\widetilde{\mathbf{u}}_j^n + \frac{1}{\Delta t}\mathbb{M}\mathbf{u}_j^n \end{aligned}$$

for the first-order scheme; and

$$\begin{aligned} \mathbb{C} &= \frac{3}{2\Delta t}\mathbb{M} + \overline{\mathbb{N}}^n + \overline{\nu}\mathbb{S}, \\ \mathbb{B}^{n+1} &= [\mathbf{b}_1, \ldots, \mathbf{b}_J] \text{ with } \mathbf{b}_j = \mathbf{h}_j^{n+1} - \mathbb{R}_j^n \widetilde{\mathbf{u}}_j^n - \nu_j' \mathbb{S}\widetilde{\mathbf{u}}_j^n + \frac{2}{\Delta t}\mathbb{M}\mathbf{u}_j^n - \frac{1}{2\Delta t}\mathbb{M}\mathbf{u}_j^{n-1} \end{aligned}$$

for the second-order scheme.

Similar to the heat equation case, if one has access to a semi-implicit numerical solver (the coefficient matrix $\mathbb{A}^{(j)}$ and vector $\mathbf{b}^{(j)}$ for the j-th problem), then the linear system for the ensemble simulations could be assembled in a straightforward manner. Taking the first-order scheme for example, we have

$$\mathbb{A}^{(j)} = \begin{bmatrix} \frac{1}{\Delta t}\mathbb{M} + \mathbb{N}_j^n + \nu_j\mathbb{S} & \mathbb{D}^\intercal \\ \mathbb{D} & \mathbb{0} \end{bmatrix}$$

and

$$\mathbf{b}^{(j)} = \begin{bmatrix} \mathbf{h}_j^{n+1} + \frac{1}{\Delta t}\mathbb{M}\mathbf{u}_j^n \\ \mathbb{0} \end{bmatrix}$$

for $j = 1, \ldots, J$, then the linear system in ensemble-based time stepping could be generated by using $\mathbb{A} = \frac{1}{J}\sum_{j=1}^{J} \mathbb{A}^{(j)}$ and $\mathbf{b}_j = \mathbf{b}^{(j)} + (\mathbb{A} - \mathbb{A}^{(j)})[\widetilde{\mathbf{u}}_j^n; \mathbf{0}]^\intercal$. Note that $\mathbb{A}$ keeps the same sparse structure as $\mathbb{A}^{(j)}$.

Since $\mathbb{A}$ is non-symmetric, when its dimension is large, the generalized minimum residual method (GMRES) method could be used to solve the linear system having a single RHS vector. To speed up the iteration, we follow [10] and use the least-square commutator preconditioning. The preconditioner $\mathbb{K}$ has the following form:

$$\mathbb{K} = \begin{bmatrix} \mathbb{C} & \mathbb{D}^\top \\ \mathbb{0} & -\mathbb{K}_S \end{bmatrix} \text{ and } \mathbb{K}_S = (\mathbb{D}\widehat{\mathbb{M}}^{-1}\mathbb{D}^\top)(\mathbb{D}\widehat{\mathbb{M}}^{-1}\mathbb{C}\widehat{\mathbb{M}}^{-1}\mathbb{D}^\top)^{-1}(\mathbb{D}\widehat{\mathbb{Q}}^{-1}\mathbb{D}^\top),$$

where $\widehat{\mathbb{M}} = \mathsf{diag}(\mathbb{M})$ consists of the diagonal entries of the velocity mass matrix. This preconditioning is applicable when the mixed approximation is uniformly stable with respect to the inf-sup condition, which is also fully automated, without requiring the construction of any auxiliary operators.

The GMRES, developed by Saad and Schultz [33], is to find an approximate solution from the Krylov subspace that minimizes the residual. The algorithm has been extended to block versions (see [34, 18] for an introduction and [35] for analysis), which use block Arnoldi process in generating Krylov subspace vectors. The block GMRES algorithm, named by BGMRES, is presented in Algorithm 2 that makes use of Algorithm 3 for the block Arnoldi process.

Algorithm 2: The block GMRES

Input: matrix $\mathbb{A} \in \mathcal{R}^{n\times n}$, matrix $\mathbb{B} \in \mathcal{R}^{n\times J}$, initial guess $\mathbb{X}_0 \in \mathcal{R}^{n\times J}$, preconditioner $\mathbb{K} \in \mathcal{R}^{n\times n}$, tolerance $\mathsf{tol} \in \mathcal{R}$ and maximum number of iteration $\mathsf{maxit} \in \mathcal{R}$.

Output: approximate solution $\mathbb{X}_s \in \mathcal{R}^{n\times J}$.

$\mathbb{R}_0 = \mathbb{B} - \mathbb{A}\mathbb{X}_0$;

find $\mathbb{V}_1$ via a reduced QR factorization of $\mathbb{R}_0 = \mathbb{V}_1\overline{\mathbb{Z}}$;

for *m=1, 2, . . . ,* maxit **do**

  compute $\mathbb{V}_{m+1}$ by performing the $\mathsf{block\ Arnoldi\ algorithm}$ on $\{\mathbb{V}_i\}_{i=1}^m$ and form the block upper Hessenberg $\overline{\mathbb{H}}_m$;

  solve $\mathbb{Y}_m = \mathrm{argmin}\|\overline{\mathbb{H}}_m\mathbb{Y} - \mathbb{E}_1\overline{\mathbb{Z}}\|_F$ by Householder reflections, where $\mathbb{E}_1 \in \mathcal{R}^{(m+1)J\times J}$ is the matrix containing the first J columns of the identity;

  if *converged* **then** exit;

end

form the solutions: $\mathbb{X}_m = \mathbb{X}_0 + [\mathbb{V}_1, \mathbb{V}_2, \ldots, \mathbb{V}_m]\mathbb{Y}_m$.

Since the search space keeps increasing during the BGMRES process, the iterations can be restarted after certain steps. Meanwhile, within a restart cycle, the basis matrices could become linearly dependent, thus deflation can be executed to remove redundant information for improving the efficiency of computation. Such algorithms have been designed in [4, 3] that perform deflations either at the beginning of iterations or during the entire iteration process. As for the time dependent problems we considered here, the solution at previous time step serves as a good initial guess for the current iteration. Together with a well-designed preconditioner, the number of iterations in BGMRES should be small. Therefore, in our numerical

Algorithm 3: One step of block Arnoldi algorithm

Input: matrix $\mathbb{A} \in \mathcal{R}^{n\times n}$, matrices $\mathbb{V}_1, \ldots, \mathbb{V}_m \in \mathcal{R}^{n\times J}$ and $\overline{\mathbb{H}}_{m-1} \in \mathcal{R}^{mJ\times(m-1)J}$.
Output: $\mathbb{V}_{m+1} \in \mathcal{R}^{n\times J}$, $\overline{\mathbb{H}}_m \in \mathcal{R}^{(m+1)J\times mJ}$.
$\widehat{\mathbb{V}}_{m+1} = \mathbb{A}\mathbb{V}_m$;
for *i=1, 2, . . . , m* **do**
  $\mathbb{H}_{i,m} = \mathbb{V}_i^\top \widehat{\mathbb{V}}_{m+1}$;
  $\widehat{\mathbb{V}}_{m+1} = \widehat{\mathbb{V}}_{m+1} - \mathbb{V}_i\mathbb{H}_{i,m}$;
end
find $\mathbb{V}_{m+1}$, $\mathbb{H}_{m+1,m}$ such that $\widehat{\mathbb{V}}_{m+1} = \mathbb{V}_{m+1}\mathbb{H}_{m+1,m}$ via reduced QR factorization;
set $\overline{\mathbb{H}}_m = \begin{bmatrix} \overline{\mathbb{H}}_{m-1} & [\mathbb{H}_{1,m}, \ldots, \mathbb{H}_{m,m}]^\intercal \\ \mathbb{0} & \mathbb{H}_{m+1,m} \end{bmatrix}$.

experiments in Section 4, we choose to perform the deflation once every time step, at the beginning of the BGMRES iterative solve. The detailed algorithm, named BGMRES-D, is presented as Algorithm 4.

Algorithm 4: The block GMRES with deflations [4]

Input: matrix $\mathbb{A} \in \mathcal{R}^{n\times n}$, matrix $\mathbb{B} \in \mathcal{R}^{n\times J}$, initial guess $\mathbb{X}_0 \in \mathcal{R}^{n\times J}$, preconditioner $\mathbb{K} \in \mathcal{R}^{n\times n}$, tolerance ϵ_d, $tol \in \mathcal{R}$ and maximum number of iteration $\mathsf{maxit} \in \mathcal{R}$.
Output: Solution of linear system $\mathbb{A}\mathbb{X} = \mathbb{B}$.
$\mathbb{R}_0 = \mathbb{B} - \mathbb{A}\mathbb{X}_0$;
define a scaling matrix $\mathbb{D} = diag(\|B(\cdot,1)\|, \ldots, \|B(\cdot,J)\|) \in \mathcal{R}^{J\times J}$;
compute reduced QR factorization of $\mathbb{R}_0\mathbb{D}^{-1}$ as $\mathbb{R}_0\mathbb{D}^{-1} = \mathbb{Q}\mathbb{T}$;
compute SVD of $\mathbb{T} = \mathbb{U}\Sigma\mathbb{W}^\intercal$ and determine the first diagonal entry of Σ, σ_{P_d}, such that $\sigma_{P_d} < \epsilon_d$;
compute $\mathbb{S} = \Sigma(1:P_d, 1:P_d)\mathbb{W}(:, 1:P_d)^\intercal\mathbb{D}$;
find $\mathbb{V}_1 \in \mathbb{R}^{n\times P_d}$ with $\mathbb{V}_1 = \mathbb{Q}\mathbb{U}(:, 1:P_d)$;
for *m=1, 2, . . . ,* maxit **do**
  $\mathbb{V}_m = \mathbb{K}^{-1}\mathbb{V}_m$;
  compute $\mathbb{V}_{m+1}$ and form the block upper Hessenberg $\overline{\mathbb{H}}_m$ by performing the block Arnoldi iteration;
  solve $\mathbb{Y}_m = \text{argmin}\|\overline{\mathbb{H}}_m\mathbb{Y} - \mathbb{E}_1\|_F$ by Householder reflections, where $\mathbb{E}_1 \in \mathcal{R}^{(m+1)P_d\times P_d}$ is the matrix containing the first J columns of the identity;
  compute $\mathbb{R}_m = (\mathbb{B}_m - \overline{\mathbb{H}}_m\mathbb{Y}_m)\mathbb{S}$;
  if $\|\mathbb{R}_m\|_F \le tol \cdot \|\mathbb{B}\|_F$ **then**
    compute $\widetilde{\mathbb{X}} = \mathbb{Z}_m\mathbb{Y}_m\mathbb{S}$;
    compute $\mathbb{X}_j = \mathbb{X}_0 + \mathbb{K}^{-1}\widetilde{\mathbb{X}}$, and stop
  end
end
form the solution: compute $\widetilde{\mathbb{X}} = \mathbb{Z}_m\mathbb{Y}_m\mathbb{S}$;
compute $\mathbb{X}_j = \mathbb{X}_0 + \mathbb{K}^{-1}\widetilde{\mathbb{X}}$.

Comparing with the BGMRES in Algorithm 2, the BGMRES-D algorithm has an extra execution of SVD of the matrix $\mathbb{T} \in \mathcal{R}^{J\times J}$ at the beginning of the iterative solution, but it only takes $\mathcal{O}(J^3)$ flops. Due to the truncation of singular values, the dimension of $\mathbb{V}_1$ usually is smaller than, and never greater than, J, which reduces the computational efforts in performing the block Arnoldi iterations.

4. **Numerical Experiments.** The goal of this section is to test the performance of the ensemble method that uses the aforementioned preconditioned block iterative solvers. The performance of ensemble simulations will be compared with the corresponding individual simulations. To this end, we first validate our simulation codes by checking the convergence rates of ensemble approximations, then use the CPU time as a criterion for measuring the computational efficiency. In particular, for approximation errors, we define $\mathcal{E}_j^N := \|u_j(t_N) - u_{j,h}^N\|$, the L_2 error of the j-th approximation result at the final time t_N in the ensemble simulations; and let $\mathtt{E}_j^N$ be the L_2 error of the j-th individual simulation result at the final time. All simulations are implemented on Matlab and performed on a PC, equipped with an Intel Core i7 processor running at 2.9GHz. Our codes are developed within the framework of IFISS (incompressible flow and iterative solver) software [9], which are executed sequentially in all the numerical tests. We also note that parallel computing is often needed in many large-scale engineering problems and would like to leave it to our future work.

Problem 1. Consider a group of 100 ($J = 100$) heat transfer problems on a rectangular domain $[0, 1] \times [0, 2]$ over time interval $[0, 1]$. Dirichlet boundary conditions are imposed on the left and right edges. Neumann boundary conditions are imposed on the top and bottom. In the j-th problem, initial and boundary conditions as well as body source in (6) are selected to match the prescribed analytic solution

$$u_j(x, y, t) = (1 + w_j)[\sin(2\pi x)\cos(2\pi y) + \sin(4\pi t)],$$

where w_j represents a random perturbation in $[-0.2, 0.2]$. The diffusion coefficient $\nu_j = 0.01(1 + \epsilon_j)$ with ϵ_j a random number in $[-0.2, 0.2]$.

We first check the rate of convergence in $\mathcal{E}_j^N$ by considering two test cases: (i) the first one uses the first-order ensemble method together with bilinear elements; and (ii) the second one uses the second-order ensemble method together with biquadratic elements. Uniform square meshes with size h and uniform time discretization with step size Δt are selected for partitioning the spatial domain and temporal interval, respectively. Denoted by N_x, N_y and K the number of partitions in horizontal, vertical and temporal directions. Based on Lemma 2.2, when a uniform mesh refinement is taken, the ensemble simulation is expected to converge linearly in the first case and converge quadratically in the second case when $h \sim \mathcal{O}(\Delta t)$.

When the discrete systems are not of very large-scale, one still could use a direct solver such as LU or sparse LU in solving the systems. However, we would like to check the performance of ensemble methods working with a block iterative algorithm. Thus, we use the breakdown-free preconditioned BCG as the linear solver for all the ensemble simulations as discussed in subsection 3.1. An incomplete Cholesky factorization is applied for preconditioning. In the iterative algorithm, the maximum number of iterations is set to be $\mathsf{maxit} = 20$ in a restart cycle and convergence criteria is relative residual less than $\mathsf{tol} = 1 \times 10^{-8}$. Although the ensemble contains $J = 100$ members, due to the limit of space, we only list the results of three problems with $j = 1, 50$ and 100 in Tables 1 - 2 for these two test cases, respectively. Wherein,

$$\begin{array}{lll} \nu_1 = 1.1901 \times 10^{-2}, & \nu_{50} = 8.4951 \times 10^{-3}, & \nu_{100} = 1.0154 \times 10^{-2}\ ; \\ w_1 = 9.6995 \times 10^{-2}, & w_{50} = -9.4653 \times 10^{-2}, & w_{100} = -3.3367 \times 10^{-2}. \end{array}$$

It is observed that, in either case, the expected rate of convergence has been achieved.

TABLE 1. The L_2 errors at final time: first-order ensemble, Q_1 elements in Problem 1.

$(N_x,\, N_y,\, K)$	$\mathcal{E}_1^N$	Rate	$\mathcal{E}_{50}^N$	Rate	$\mathcal{E}_{100}^N$	Rate
(16, 32, 50)	5.8005×10^{-2}	–	4.3544×10^{-2}	–	4.8908×10^{-2}	–
(32, 64, 100)	2.9140×10^{-2}	0.99	2.1972×10^{-2}	0.99	2.4615×10^{-2}	0.99
(64, 128, 200)	1.4629×10^{-2}	0.99	1.1061×10^{-2}	0.99	1.2371×10^{-2}	0.99
(128, 256, 400)	7.3326×10^{-3}	1.00	5.5529×10^{-3}	1.00	6.2053×10^{-3}	1.00

TABLE 2. The L_2 errors at final time: second-order ensemble, Q_2 elements in Problem 1.

$(N_x,\, N_y,\, K)$	$\mathcal{E}_1^N$	Rate	$\mathcal{E}_{50}^N$	Rate	$\mathcal{E}_{100}^N$	Rate
(8, 16, 50)	3.1827×10^{-3}	–	2.4799×10^{-3}	–	2.7259×10^{-3}	–
(16, 32, 100)	7.6003×10^{-4}	2.07	5.8014×10^{-4}	2.10	6.4617×10^{-4}	2.08
(32, 64, 200)	1.9288×10^{-4}	1.98	1.4695×10^{-4}	1.98	1.6366×10^{-4}	1.98
(64, 128, 400)	4.9629×10^{-5}	1.96	3.7682×10^{-5}	1.96	4.2046×10^{-5}	1.96

Next, we compare the performance of ensemble simulations with individual simulations on the same group of problems. For this purpose, we take the first-order ensemble method in time and bilinear finite elements in space with the following parameters: $N_x = 128$, $N_y = 256$ and $K = 400$. The total number of degrees of freedom is $n_u = 33,153$. The individual simulations are based on a semi-implicit time stepping together with the standard preconditioned CG algorithm, but each problem in the group is solved separately. We choose the convergence criteria of CG to be same as the BFBCG: $\mathsf{maxit} = 20$ in each restart cycle and $\mathsf{tol} = 1 \times 10^{-8}$. Three individual simulations of problems with $j = 1, 50$ and 100 have the following approximation errors:

$$\mathtt{E}_1^N = 7.3268 \times 10^{-3}, \quad \mathtt{E}_{50}^N = 5.5584 \times 10^{-3}, \quad \mathtt{E}_{100}^N = 6.2051 \times 10^{-3}.$$

Comparing them with the corresponding ensemble simulation errors (listed in the last row of Table 1), we observe the accuracy of both approaches are very close. However, as listed in Table 3, the CPU time for time integrations in the ensemble with BFBCG solver is 5.658×10^2 seconds, while that of individual simulations with the preconditioned CG solver is 1.4640×10^3 seconds, which leads to a speedup factor of nearly 2.60.

TABLE 3. CPU time comparison in Problem 1.

Iteration & CPU time	BFBCG	100 CG
Average iteration number per time step	4	4× 100
Average execution time per step (seconds)	5.6362×10^{-1}	$(1.1482\times10^{-2})\times 100$
Total CPU time for integration (seconds)	5.658×10^2	1.464×10^3

In this case, neither BFBCG nor preconditioned CG needs to restart the iterations during the simulations. The number of iterations per time step in BFBCG is the same as the individual preconditioned CG solve. The average execution time per time step in BFBCG (for 100 problems) is 5.6362×10^{-1} seconds, and that in each individual PCG solve is 1.1482×10^{-2} seconds. For a fair comparison, we multiply

the execution time of a single preconditioned CG solve by 100, which costs about 2 times larger than one BFBCG solve. We observe that the main computational saving in BFBCG comes from the reduction of search directions. Thus, we plot the change in the rank of $\mathcal{P}_i$ with respect to iterations every 10 time steps in Figure 1. It is seen that the search dimension increases with iterations, but the largest rank of $\mathcal{P}_i$ during the simulation is 9, which is much less than J.

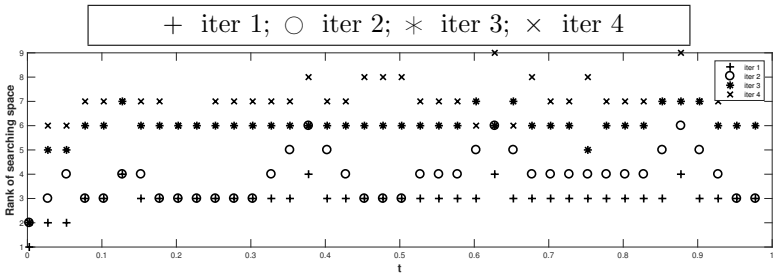

FIGURE 1. Time evolution of the dimension of the search space at different iterations in BFBCG.

Problem 2. Consider a group of 40 ($J = 40$) Navier-Stokes equations with Taylor-Green vortex solutions on a square domain $[-1, 1] \times [-1, 1]$ over the time interval $[0, 1]$. Dirichlet boundary conditions are imposed on the edges. Initial and boundary conditions and body force are selected to match the prescribed analytic solution. In the j-th problem,

$$\begin{aligned} u_j(x, y, t) &= \sin(\pi x)\cos(\pi y)e^{-2\nu_j \pi^2 t}, \\ v_j(x, y, t) &= -\cos(\pi x)\sin(\pi y)e^{-2\nu_j \pi^2 t}, \\ p_j(x, y, t) &= \frac{1}{4}(\cos 2\pi x + \cos 2\pi y)e^{-4\nu_j \pi^2 t}, \end{aligned}$$

and the diffusion coefficient $\nu_j = 0.01(1 + \epsilon_j)$ with ϵ_j a random number uniformly distributed in $[-0.2, 0.2]$.

We first check the rate of convergence in velocity approximations for the first-order and second-order ensemble methods, respectively. The $Q_2/Q1$ Taylor-Hood elements are used in both cases for spatial discretization. Uniform rectangular meshes with size h and uniform time discretization with step size Δt are selected respectively for partitioning the spatial domain and temporal interval. The same notation h, Δt, N_x, N_y and K as Problem 1 are used. In both tests, we fix big enough N_x and N_y so that the temporal error would dominate the approximation errors, and vary the time step size to check the rate of convergence. Based on Lemma 2.4, the ensemble simulation is expected to converge linearly in the first case and converge quadratically in the second case when a uniform time refinement is taken.

As discussed in subsection 3.2, the BGMRES-D algorithm together with the least-square commutator preconditioner is used for solving the discrete systems resulted from the ensemble-based time stepping. The maximum number of iterations in each restart cycle is $\mathsf{maxit} = 50$ and stopping criterion is fulfilled if the relative residual is no greater than $\mathsf{tol} = 1 \times 10^{-8}$. The ensemble size is $J = 40$. However due to the limit of space, we only list the results of three problems for $j = 1, 20$ and 40 in

Tables 4 - 5. Wherein,

$$\nu_1 = 8.0619 \times 10^{-3}, \quad \nu_{20} = 1.1681 \times 10^{-2}, \quad \nu_{40} = 1.0804 \times 10^{-2},$$

and $\mathcal{E}_j^N$ denotes the velocity approximation errors in L_2 norm at the final time. It is seen, in both cases, that the expected rates of convergence have been obtained.

TABLE 4. The L_2 errors at final time: first-order ensemble, Q_2/Q_1 elements in Problem 2.

$(N_x,\, N_y,\, K)$	$\mathcal{E}_1^N$	Rate	$\mathcal{E}_{20}^N$	Rate	$\mathcal{E}_{40}^N$	Rate
(128, 128, 5)	3.7899×10^{-3}	–	3.4660×10^{-3}	–	3.6324×10^{-3}	–
(128, 128, 10)	1.9331×10^{-3}	0.97	1.7626×10^{-3}	0.98	1.8478×10^{-3}	0.98
(128, 128, 20)	9.7905×10^{-4}	0.98	8.9096×10^{-4}	0.98	9.3418×10^{-4}	0.98
(128, 128, 40)	4.8947×10^{-4}	1.00	4.4499×10^{-4}	1.00	4.6666×10^{-4}	1.00

TABLE 5. The L_2 errors at final time: second-order ensemble, Q_2/Q_1 elements in Problem 2.

$(N_x,\, N_y,\, K)$	$\mathcal{E}_1^N$	Rate	$\mathcal{E}_{20}^N$	Rate	$\mathcal{E}_{40}^N$	Rate
(256, 256, 5)	1.0597×10^{-3}	–	9.5139×10^{-4}	–	9.9694×10^{-4}	–
(256, 256, 10)	2.5992×10^{-4}	2.03	2.3215×10^{-4}	2.03	2.4328×10^{-4}	2.03
(256, 256, 20)	6.3997×10^{-5}	2.02	5.7010×10^{-5}	2.03	5.9748×10^{-5}	2.02
(256, 256, 40)	1.5905×10^{-5}	2.00	1.4112×10^{-5}	2.01	1.4796×10^{-5}	2.01

To illustrate the efficiency of the ensemble method, we take the second test case when $N_x = N_y = 256$ and $K = 40$ for example, and compare the execution time of ensemble simulations with 40 individual simulations using the same mesh and time step sizes. In this case, the total number of degrees of freedom is $n_u = 132,098$ and $n_p = 16,641$. In individual simulations, preconditioned GMRES algorithm is applied to solve discrete linear systems with the same stopping criterion as the preconditioned BGMRES-D, but each problem in the group is solved separately. We choose the convergence criteria of GMRES to be same as the BGMRES-D: $\mathsf{maxit} = 50$ in each restart cycle and $\mathsf{tol} = 1 \times 10^{-8}$. Three individual simulations of problems with $j = 1, 20$ and 40 have the following approximation errors:

$$\mathtt{E}_1^N = 1.0377 \times 10^{-5}, \quad \mathtt{E}_{20}^N = 1.9140 \times 10^{-5}, \quad \mathtt{E}_{40}^N = 1.7075 \times 10^{-5}.$$

Comparing with Table 5, we find that the accuracy of ensemble simulations is close to individual simulations. The corresponding execution times of both approaches are listed in Table 6. It is shown that the ensemble simulation is over 10 times faster than the individual simulations.

TABLE 6. CPU time comparison in Problem 2.

Iteration & CPU time	BGMRES-D	40 GMRES
average iteration number per time step	5	5× 40
average execution time per step (seconds)	19.658	12.032 × 40
total CPU time for integration (seconds)	1.965×10^3	2.103×10^4

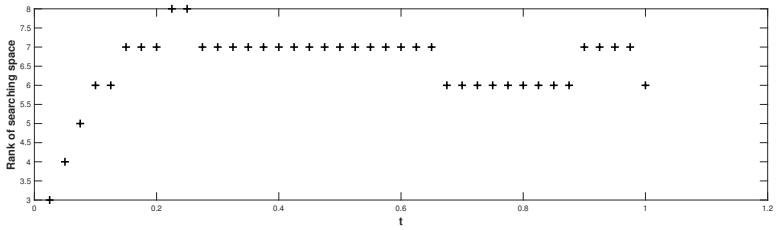

FIGURE 2. Time evolution of the rank of RHS vectors using the BGMRES-D solver in Problem 2.

It is also observed that neither the BGMRES-D nor the single GMRES solver restarts the iterations during the simulations. The iteration number per time step in BGMRES-D is the same as that of the GMRES in each individual simulation. But as the deflation is performed at each time step, the dimension of the RHS matrices in BGMRES-D is always no greater than J. The time evolution of its rank is plotted in Figure 2, which shows the maximum value of the rank is 8. This indicates fewer matrix-vector products are evaluated in ensemble simulations than individual ones. On the other hand, the least-square commutator preconditioning involves two discrete Poisson solves and matrix-vector products. Since for the group of problems, the ensemble simulations only apply one preconditioning to the coefficient matrix per iteration, while each individual simulation would require one preconditioner to the coefficient matrix at every iteration, which leads to the significant computational savings.

Problem 3. Next, we consider a group of two-dimensional flow past a square cylinder problems with the ensemble size $J = 40$. The flow problems are governed by Navier-Stokes equations in the domain $\Omega = \Omega_0/\Omega_s$ over the time interval $[0, 60]$ with $\Omega_0 = [0, 8] \times [-1, 1]$ and interior obstacle $\Omega_s = [1.75, 2.25] \times [-0.25, 0.25]$. Dirichlet condition $u = 1 - y^2, v = 0$ is imposed at the inflow boundary ($x = 0$ and $-1 \le y \le 1$), zero Dirichlet condition on the top and bottom of the channel ($0 \le x \le 8$ and $y = \pm 1$), and do-nothing boundary on the outflow boundary ($x = 8$ and $-1 < y < 1$). The flow is at rest at the initial time. Viscosity coefficients vary among the problems, in particular, $\nu_j = (1+\epsilon_j)/300$ with ϵ_j a random perturbation uniformly distributed in $[-0.2, 0.2]$ in the j-th problem.

In the simulations, we use second order schemes with a uniform time step size $\Delta t = 0.01$ for time integration, and keep the same parameters in block and individual iterative solvers as those in Problem 2. As there is no analytic solutions, we only show the ensemble simulation results together with individual ones, and compare their CPU times. Due to the limit of space, we only show the velocity magnitude field at the final time for three problems associated with Reynolds numbers: $\mathsf{Re}_1 = 126.7001$, $\mathsf{Re}_{20} = 176.9705$ and $\mathsf{Re}_{40} = 150.6167$ in Figure 3, and the evolution of speed at a point behind the cylinder, $(6, 2)$, in Figure 4. The simulation results are so close that no obvious difference can be observed.

The time evolution of the dimension of RHS vectors in the ensemble simulation is shown in Figure 5. It is seen that the size of RHS matrix does shrink after deflations. The corresponding simulation times of the ensemble simulations with preconditioned BGMRES-D and individual simulations with preconditioned GMRES are listed in Table 7. It is shown that, comparing with the individual simulations, the ensemble simulation saves about 85% CPU time.

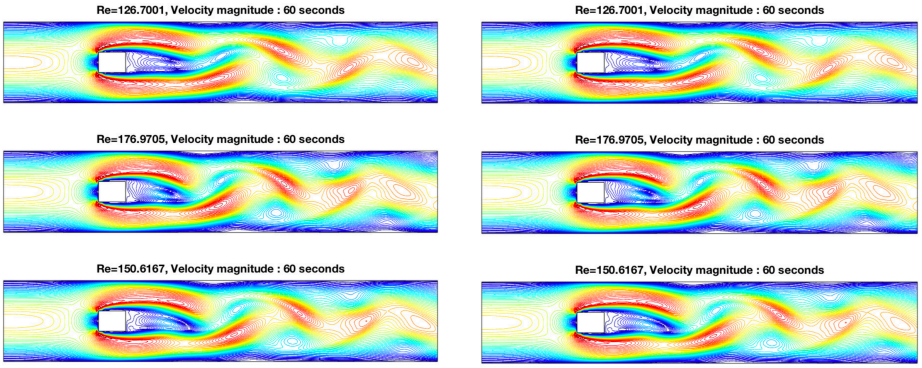

FIGURE 3. Speed fields at $t = 60$ for three problems in the ensemble simulation (left column) and individual simulations (right column) in Problem 3.

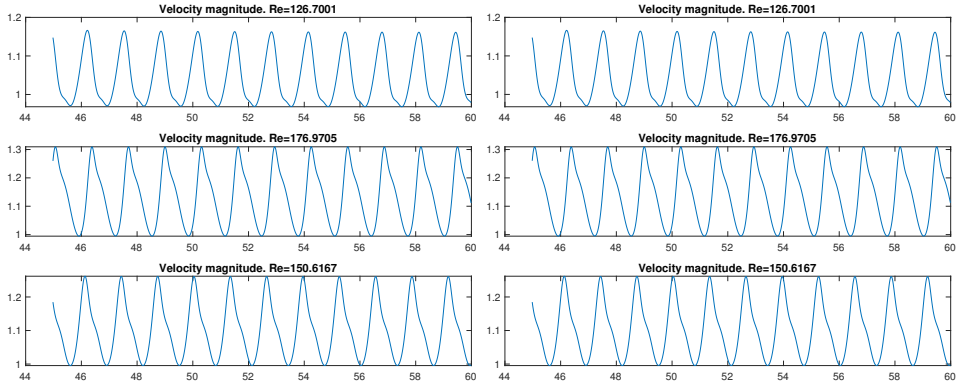

FIGURE 4. Time evolutions of velocity magnitude for three problems in the ensemble simulation (left column) and individual simulations (right column) in Problem 3.

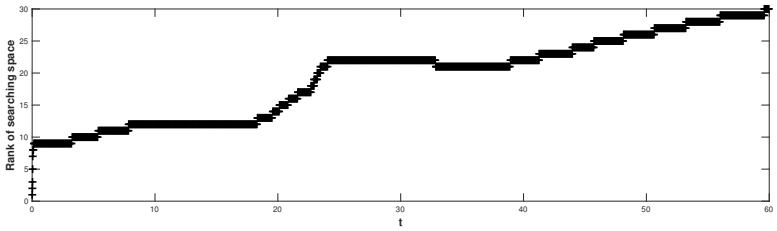

FIGURE 5. Time evolution of the rank of RHS vectors using BGMRES-D solver in Problem 3.

Remark 1. *We note that the speedup factor of the ensemble simulations over the individual ones in the Problem 3 is not as great as Problem 2, it is because the computational saving of the block iterative algorithms is problem dependent. The solutions in Problem 3 are rapidly changing with time, as small differences in the viscosity could result in relatively large changes in the state, thus the deflation of*

TABLE 7. CPU time comparison in Problem 3.

Iteration & CPU time	BGMRES-D	40 GMRES
Average iteration number per time step	7	$8\times$ 40
Average execution time per step (seconds)	4.7894	1.7313×40
Total CPU time for integration (seconds)	7.106×10^4	4.836×10^5

the RHS matrix is not as effective as that in Problem 2, which causes the speedup less eminent.

5. **Conclusions.** The ensemble method has recently been developed for efficiently solving a group of evolution problems. It, using an ensemble-based time stepping, leads to a single linear system of multiple right-hand-side vectors for the entire group. Thus, all the problems can be solved simultaneously at each time step, which naturally share information among right hand sides. In this paper, we demonstrate, for the first time, the efficacy of the ensemble method when it works together with block iterative solvers. Our future work would extend the ensemble method to multi-physics systems and investigate the efficiency and scalability of block iterative solvers in parallel and high performance computing.

REFERENCES

[1] E. Agullo, L. Giraud and Y.-F. Jing, Block GMRES method with inexact breakdowns and deflated restarting, *SIAM Journal on Matrix Analysis and Applications*, **35** (2014), 1625–1651.

[2] A. H. Baker, J. M. Dennis and E. R. Jessup, On improving linear solver performance: A block variant of GMRES, *SIAM Journal on Scientific Computing*, **27** (2006), 1608–1626.

[3] H. Calandra, S. Gratton, R. Lago, X. Vasseur and L. M. Carvalho, A modified block flexible GMRES method with deflation at each iteration for the solution of non-hermitian linear systems with multiple right-hand sides, *SIAM Journal on Scientific Computing*, **35** (2013), S345–S367.

[4] H. Calandra, S. Gratton, J. Langou, X. Pinel and X. Vasseur, Flexible variants of block restarted GMRES methods with application to geophysics, *SIAM Journal on Scientific Computing*, **34** (2012), A714–A736.

[5] T. F. Chan and M. K. Ng, Galerkin projection methods for solving multiple linear systems, *SIAM Journal on Scientific Computing*, **21** (1999), 836–850.

[6] A. T. Chronopoulos and A. B. Kucherov, Block s-step Krylov iterative methods, *Numerical Linear Algebra with Applications*, **17** (2010), 3–15.

[7] D. Darnell, R. B. Morgan and W. Wilcox, Deflated GMRES for systems with multiple shifts and multiple right-hand sides, *Linear Algebra and its Applications*, **429** (2008), 2415–2434.

[8] I. S. Duff, A. M. Erisman and J. K. Reid, *Direct methods for sparse matrices*, Oxford University Press, 2017.

[9] H. C. Elman, A. Ramage and D. J. Silvester, IFISS: A computational laboratory for investigating incompressible flow problems, *SIAM Review*, **56** (2014), 261–273.

[10] H. C. Elman, D. J. Silvester and A. J. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*, Oxford university press, 2005.

[11] J. A. Fiordilino, A second order ensemble timestepping algorithm for natural convection, *SIAM Journal on Numerical Analysis*, **56** (2018), 816–837.

[12] W. D. Gropp, D. K. Kaushik, D. E. Keyes and B. F. Smith, Toward realistic performance bounds for implicit CFD codes, in *Proceedings of parallel CFD*, vol. 99, Citeseer, 1999, 233–240.

[13] G.-D. Gu and Z.-H. Cao, A block GMRES method augmented with eigenvectors, *Applied Mathematics and Computation*, **121** (2001), 271–289.

[14] M. Gunzburger, N. Jiang and M. Schneier, An ensemble-proper orthogonal decomposition method for the nonstationary Navier–Stokes equations, *SIAM Journal on Numerical Analysis*, **55** (2017), 286–304.
[15] M. Gunzburger, N. Jiang and Z. Wang, A second-order time-stepping scheme for simulating ensembles of parameterized flow problems, *Computational Methods in Applied Mathematics*, **19** (2017), 681–701.
[16] M. Gunzburger, N. Jiang and Z. Wang, An efficient algorithm for simulating ensembles of parameterized flow problems, *IMA Journal of Numerical Analysis*, **39** (2019), 1180–1205.
[17] M. D. Gunzburger, *Finite element methods for viscous incompressible flows: a guide to theory, practice, and algorithms*, Elsevier, 2012.
[18] M. H. Gutknecht, Block Krylov space methods for linear systems with multiple right-hand sides: an introduction, *in: Modern Mathematical Models, Methods and Algorithms for Real World Systems*, 420–447.
[19] H. Ji and Y. Li, A breakdown-free block conjugate gradient method, *BIT Numerical Mathematics*, **57** (2017), 379–403.
[20] N. Jiang, S. Kaya and W. Layton, Analysis of model variance for ensemble based turbulence modeling, *Computational Methods in Applied Mathematics*, **15** (2015), 173–188.
[21] N. Jiang and W. Layton, An algorithm for fast calculation of flow ensembles, *International Journal for Uncertainty Quantification*, **4** (2014), 273–301.
[22] N. Jiang and W. Layton, Numerical analysis of two ensemble eddy viscosity numerical regularizations of fluid motion, *Numerical Methods for Partial Differential Equations*, **31** (2015), 630–651.
[23] Y. Luo and Z. Wang, An ensemble algorithm for numerical solutions to deterministic and random parabolic PDEs, *SIAM Journal on Numerical Analysis*, **56** (2018), 859–876.
[24] Y. Luo and Z. Wang, A multilevel Monte Carlo ensemble scheme for solving random parabolic PDEs, *SIAM Journal on Scientific Computing*, **41** (2019), A622–A642.
[25] J. McCarthy, Block-conjugate-gradient method, *Physical Review D*, **40** (1989), 2149.
[26] M. Mohebujjaman and L. G. Rebholz, An efficient algorithm for computation of MHD flow ensembles, *Computational Methods in Applied Mathematics*, **17** (2017), 121–137.
[27] R. B. Morgan, Restarted block-GMRES with deflation of eigenvalues, *Applied Numerical Mathematics*, **54** (2005), 222–236.
[28] D. P. O'Leary, The block conjugate gradient algorithm and related methods, *Linear Algebra and its Applications*, **29** (1980), 293–322.
[29] D. P. O'Leary, Parallel implementation of the block conjugate gradient algorithm, *Parallel Computing*, **5** (1987), 127–139.
[30] M. L. Parks, E. De Sturler, G. Mackey, D. D. Johnson and S. Maiti, Recycling Krylov subspaces for sequences of linear systems, *SIAM Journal on Scientific Computing*, **28** (2006), 1651–1674.
[31] M. L. Parks, K. M. Soodhalter and D. B. Szyld, A block recycled GMRES method with investigations into aspects of solver performance, *arXiv preprint arXiv:1604.01713*.
[32] V. Puzyrev and J. M. Cela, A review of block Krylov subspace methods for multisource electromagnetic modelling, *Geophysical Journal International*, **202** (2015), 1241–1252.
[33] Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM Journal on scientific and statistical computing*, **7** (1986), 856–869.
[34] Y. Saad, *Iterative methods for sparse linear systems*, vol. 82, Siam, 2003.
[35] V. Simoncini and E. Gallopoulos, Convergence properties of block GMRES and matrix polynomials, *Linear Algebra and its Applications*, **247** (1996), 97–119.
[36] V. Simoncini and E. Gallopoulos, A hybrid block GMRES method for nonsymmetric systems with multiple right-hand sides, *Journal of computational and applied mathematics*, **66** (1996), 457–469.
[37] L. N. Trefethen and D. Bau III, *Numerical linear algebra*, vol. 50, SIAM, 1997.

Received xxxx 20xx; revised xxxx 20xx.

E-mail address: ju@math.sc.edu
E-mail address: wleng@lsecc.cc.ac.cn
E-mail address: wangzhu@math.sc.edu
E-mail address: syuan@email.sc.edu