A resilient and efficient CFD framework: Statistical learning tools for multi-fidelity and heterogeneous information fusion

Seungjoon Lee\textsuperscript{a}, Ioannis G. Kevrekidis\textsuperscript{b}, George Em Karniadakis\textsuperscript{a,*}

\textsuperscript{a} Division of Applied Mathematics, Brown University, Providence, RI 02912, United States
\textsuperscript{b} Department of Chemical and Biological Engineering; also PACM, Princeton University, Princeton, NJ 08544, United States

\section*{Article Info}

\textbf{Article history:} 
Received 25 November 2016
Received in revised form 7 May 2017
Accepted 10 May 2017
Available online 15 May 2017

\textbf{Keywords:} 
Gappy data
Patch dynamics
Machine learning
Domain decomposition
Exascale simulation
Gaussian process regression
Diffusion maps

\section*{Abstract}

Exascale-level simulations require fault-resilient algorithms that are robust against repeated and expected software and/or hardware failures during computations, which may render the simulation results unsatisfactory. If each processor can share some global information about the simulation from a coarse, limited accuracy but relatively costless auxiliary simulator we can effectively fill-in the missing spatial data at the required times by a statistical learning technique – multi-level Gaussian process regression, on the fly; this has been demonstrated in previous work \cite{1}. Based on the previous work, we also employ another (nonlinear) statistical learning technique, Diffusion Maps, that detects computational redundancy in time and hence accelerate the simulation by projective time integration, giving the overall computation a “patch dynamics” flavor. Furthermore, we are now able to perform information fusion with multi-fidelity and heterogeneous data (including stochastic data). Finally, we set the foundations of a new framework in CFD, called patch simulation, that combines information fusion techniques from, in principle, multiple fidelity and resolution simulations (and even experiments) with a new adaptive timestep refinement technique. We present two benchmark problems (the heat equation and the Navier–Stokes equations) to demonstrate the new capability that statistical learning tools can bring to traditional scientific computing algorithms. For each problem, we rely on heterogeneous and multi-fidelity data, either from a coarse simulation of the same equation or from a stochastic, particle-based, more “microscopic” simulation. We consider, as such “auxiliary” models, a Monte Carlo random walk for the heat equation and a dissipative particle dynamics (DPD) model for the Navier–Stokes equations. More broadly, in this paper we demonstrate the symbiotic and synergistic combination of statistical learning, domain decomposition, and scientific computing in exascale simulations.

© 2017 Elsevier Inc. All rights reserved.

\section{Introduction}

As demands for solving complex physics problems are growing in computational fluid dynamics, exascale simulations will be performed in the near future, e.g. \cite{2,3}. However, exascale simulations, which use massive numbers of processors, require not only an efficient algorithm but also a fault-resilient algorithm to address traditional open issues \cite{4}. The key issue is “fault resilience” against repeated and expected (but random) software or hardware failures during computations, which may
render the simulation results unsatisfactory or simply unusable. Many research approaches have been developed to recover the missing data on both sides, computer systems [5–8] and mathematical algorithms [9–12]. The other important issue is “computational efficiency” against computational redundancy from inadequate spatio-temporal discretization or dynamics of a problem itself. Over the last few decades, several numerical methods are introduced to overcome the computational redundancy [13,14].

This paper is motivated by the approach introduced by Lee et al. [1], which demonstrably achieved fault-resilience when solving PDEs and processor failures results in spatial gaps in the simulation output. This was achieved by an information fusion method with multi-resolution auxiliary data from a repeatedly reinitialized short-duration coarse resolution auxiliary simulation of the problem. Based on this algorithm, we now introduce a new framework, namely “patch simulation”, for accelerating simulations via a statistical learning technique, Diffusion Maps (DMaps), that helps detect computational redundancy in time and hence accelerating the simulation by projective time integration. The projective time integration was introduced by Gear and Kevrekidis [15] and used in Sirisup et al. [16] in a CFD context, demonstrating computational acceleration by an equation-free POD-assisted approach. However, choosing appropriate projection steps is still an open issue. In this paper, learning from the auxiliary data via diffusion maps provides a dynamics-informed temporal discretization, which leads to an acceleration of the computation in time.

The original concept of the patch simulation comes from “patch dynamics” [15,17], which predicts large scale spatio-temporal dynamics (macro level) from a series of short spatio-temporal dynamics (micro level). In order to extend this concept to a large scale simulation of a CFD problem, we employ two statistical learning techniques to provide additional and important spatio-temporal information from the auxiliary data, produced by a coarse, auxiliary, repeatedly restarted, relatively costless simulations (even experiments). The first method is a multi-level Gaussian process regression, which supports “fill-in” the missing data in space [18–21]. The other method is diffusion maps, which provides lower-dimensional information to estimate the appropriate projection time [22–25]. In this paper, we show that these two statistical learning techniques can use the auxiliary data to improve the accuracy and the efficiency simultaneously.

Furthermore, in order to investigate multiscale phenomena in complex problems, many research efforts have sought to couple numerical simulations and even experiments, with multi-fidelity and heterogeneous models via information fusion techniques [26–28]. In this paper, the auxiliary data is generalized to stochastic descriptions such as a random walk model for the heat equation and a dissipative particle dynamics (DPD) model for the Navier–Stokes equations. This results in a guarantee of the generality of the auxiliary data, which can come, in principle, from any scale, any fidelity and any heterogeneous model. The heat equation (linear) and the lid-driven cavity flow (nonlinear) are simple problems but they have been used as general benchmark problems for demonstrating effectiveness of new algorithms. These problems serve also as useful benchmarks in stochastic modeling and uncertainty quantification [29].

The paper is organized as follows: In section 2, we introduce a general patch simulation framework with a flow chart. In section 3, we introduce the two statistical learning techniques used here, the multi-level Gaussian process regression and Diffusion Maps (DMaps). In section 4, we introduce heterogeneous and multi-fidelity models as the source of auxiliary data for each benchmark problem. In section 5, we present the computational domains and simulation set up for the benchmark problems. In section 6, we present results of parametric studies and analyze them in terms of the overall accuracy. In section 7, we summarize our results and discuss open issues for further development of a robust and efficient CFD framework.

2. The patch simulation framework

In the new CFD framework we propose, we split up the simulation into a main simulation and an auxiliary simulation. The main simulation computes a solution to a PDE on some (non-overlapping) fine-resolution sub-domains with spatial gaps. The auxiliary simulation, on the other hand, computes a solution on the entire domain but with less accuracy. This auxiliary data can come from any fidelity, scale, or model. In previous work (see reference [1]), data from the auxiliary simulation provide information about the global continuity of the solution fields to the gappy, main simulation via multi-level Gaussian process regression, namely coKrifting. Then We estimate field variables at each local boundary using both global (with low accuracy) and local (with high accuracy) information. This leads to a fault-resilience with respect to spatial gaps.

In [16], a data-driven “equation-free/Galerkin-free POD-assisted computation” was presented that exploited brief bursts of the full Navier–Stokes simulation over the entire computational domain and the low-dimensionality of the behavior to accelerate the overall simulation. In this paper we show how to extend this approach so as to avoid doing the detailed simulation over the entire spatial domain. The main new enabling ingredient is the availability of a “cheap and coarse” auxiliary simulation, which is reliable only over short time integration due to error accumulation in time, and which is therefore repeatedly (and carefully) reinitialized. Keeping the same assumption (low-dimensionality of the long-term dynamics) this new ingredient will, as we will show in section 6, enable significant computational savings over the spatial extent of the full detailed simulation. While this ingredient allows the full simulation to be performed systematically in only parts of the domain, the same idea can be also used in enabling the “filling in” of computational information loss through hardware/software failure locally in space–time. We repeat that the main assumption is that the long-term dynamics lie on a slow manifold, which is parameterized by a few POD basis functions [30]. Then, we perform simulations on this manifold through short time computations via “projective time integration”, and this results in accelerating the main simulation. However, it is nontrivial to systematically and accurately estimate the length of the appropriate projection steps due to rapid changes of dynamics
in a transient period. The auxiliary data help estimate this projection time, or “a jump size”, for the main simulation via a statistical learning technique, diffusion maps. Finally, this framework bears some similarities with “patch dynamics” for a micro–macro simulator [15,17]. The projection time represents a macro time scale for dynamics on the slow manifold while detailed computation on the fine-resolution sub-domains represents the micro (fine)-level of the description of the dynamics.

A flow chart of the patch simulation is shown in Fig. 1. A cycle of the main simulation with fine-resolution sub-domains is described in Fig. 1(a) (see reference [1] for details). We first check spatial gaps due to computational faults and choose a buffer size for each fine-resolution sub-domain. Next, we estimate the local boundary condition of each sub-domain with auxiliary data and obtain the solution (in embarrassingly parallel) during a macro timestep, called the non-interaction timestep. After each macro timestep, we save a snapshot of the field variables on fine-resolution sub-domains. We repeat this cycle until we have “N” snapshots. After that, we employ the projective time integration scheme on each fine-resolution sub-domain independently with an appropriate projection time. This constitutes one complete cycle of the patch simulation algorithm. The patch simulation repeats this cycle until the main simulation ends or all faults are fixed. The number of saved snapshots, N, is a hyper-parameter of the patch simulation and will be discussed in section 6.3.

As shown in Fig. 1(a), the auxiliary data provides two different types of information (colored by blue) through two different statistical learning techniques (colored by red) to the main simulation, which has missing data at local boundaries.
of each fine-resolution sub-domain. As shown in Fig. 1(b), first is the global and spatial information with low accuracy (blue arrows), via the multi-level Gaussian process regression, employed to estimate local boundary conditions of each fine-resolution sub-domains. Second is the temporal information of a projection size (a red arrow), via diffusion maps, for the projective time integration. Details of these statistical learning techniques are described below.

3. Statistical learning algorithms

In order to connect the auxiliary simulation with the main simulation, we take advantage of two statistical learning algorithms, namely multi-level Gaussian process regression and diffusion maps. Multi-level Gaussian process regression has to do with fine-resolution sub-domains, and can be done independent of projection. The other, Diffusion Maps, have to do with projection, and can be done independent of gaps in space. Here we will put them together but we first describe each “without the other”.

3.1. Gaussian process regression

The auxiliary simulation, which is relatively costless, provides a data set to “fill in” the missing data spatially, such as the boundary conditions of each sub-domain. In this framework there are 2-levels of data sets: (1) data from the main simulation, the local information with high accuracy and (2) data from the auxiliary simulation, the global information with low accuracy. Two data sets, from the main simulation and the auxiliary simulation, are modeled by two different Gaussian processes, \( y \) and \( y_a \), respectively. There is a scaling parameter (\( \rho \)), which represents a correlation between two Gaussian processes. Then, the auto-regressive scheme of Kennedy and O’Hagan [31] is defined by:

\[
y(x) = \rho(x)y_a(x) + \delta(x), \tag{1}
\]

where \( \delta(x) \) is a bias Gaussian field, which is independent of \( y \) and \( y_a \). \( \hat{y}(x^*) \) and \( \hat{y}_a(x^*) \) are the mean value of predicted field variables, i.e., temperature or velocities, at the local boundary \( x^* \) of each sub-domain. The mean \( (\hat{y}(x^*)) \) and variance \( (\hat{s}^2(x^*)) \) of predicted field variables are obtained by the following equation with \( \{\mu, \sigma^2\} \) and \( \{\mu_a, \sigma_a^2\} \), which represent means and variances of Gaussian processes of \( y \) and \( y_a \), respectively:

\[
\hat{y}(x^*) = \mu + \rho \hat{y}_a(x^*) + r^T \left( R + \sigma^2 I \right)^{-1} \left[ y(x) - \mu \right], \tag{2}
\]

\[
\hat{s}^2(x^*) = \rho^2 \hat{s}_a^2(x^*) + \sigma^2 \left[ 1 - r^T \left( R + \sigma^2 I \right)^{-1} r \right] + \left[ \frac{1 - r^T \left( R + \sigma^2 I \right)^{-1} r}{1^T \left( R + \sigma^2 I \right)^{-1} 1} \right], \tag{3}
\]

where \( R = \kappa(x, x', \theta) \) and \( r = \kappa(x, x^*, \theta) \) represent a correlation matrix and a correlation vector, respectively, and \( \sigma^2 \) is the variance of the normally distributed random field. The correlation kernel, \( \kappa(x, x', \theta) \), is constructed by the following Matérn kernel [32]:

\[
\kappa(x, x', \theta) = \left( 1 + \sqrt{3} \theta d(x, x') \right) \exp \left( -\sqrt{3} \theta d(x, x') \right), \quad x, x' \in y, \tag{4}
\]

where \( d(\cdot, \cdot) \) represents the Euclidean distance between two data.

An optimal hyperparameter set, \( \{\mu, \sigma, \sigma_a, \rho, \theta\} \), is obtained by maximum likelihood estimation (MLE) from the aforementioned data sets. For details of the formulation and hyper-parameters, see reference [21].

3.2. Diffusion maps

Diffusion maps, based on a discrete Laplace–Beltrami operator, provide a parametrization of the low-dimensional nonlinear manifold as well as a (diffusion) distance on it [23–25]. As shown in Fig. 2, diffusion maps obtained by temporal snapshots of field variables provide a “dynamics-related” temporal discretization on the diffusion coordinate. This observation can be used as a criterion of the computational redundancy of the original uniform discretization in time. Based on the diffusion distance, we can assign an appropriate projection time for the projective time integration. The diffusion distance, \( D(\cdot, \cdot) \), between adjacent snapshots is calculated on a one-dimensional slow manifold.

\[
D^2(t_{i+1}, t_i) = \lambda_1^2 \left[ \phi_1(t_{i+1}) - \phi_1(t_i) \right]^2, \tag{5}
\]

where \( (\lambda_1, \phi_1) \) represents the first nontrivial eigenvalue and eigenvector of the discrete Laplace–Beltrami operator via a diffusion kernel, \( W \).

\[
W_{i,j} = \exp \left( -\frac{\|y_i - y_j\|^2}{\epsilon^2} \right), \tag{6}
\]
where $y$ represents each data vector (temporal snapshot) and $\epsilon$ represents a characteristic distance between data vectors. In this framework, we choose $\epsilon$ to be the median distance between all data from the coarse but full simulation.

A formulation for finding an appropriate projection time starts from a fundamental observation of the diffusion distance between temporal snapshots – the diffusion distance between adjacent snapshots in time corresponds to the data variability of the slow dynamics during that time interval. A large diffusion distance, for example, represents a large change in the dynamics, i.e., we have to employ the projective time integration with small timesteps, and vice versa. Hence, the projection time at time $t_i$ is generalized to an “adaptive timestep refinement” by the following equation:

$$J(t_i) = \alpha \log \frac{D_m}{D(t_{i+1}, t_i)},$$

(7)

where $\alpha$ is a tuned parameter and $D_m$ represents the maximum diffusion distance along the time series. Since the projection time should be an integer, $J(t_i)$ is rounded to the nearest integer. Then, we project and estimate field variables at time $t' = t + J(t) \cdot \tau \cdot \Delta t$ by a proper orthogonal decomposition (POD) [33] assisted projection [16], where $\tau$ is a non-interaction timestep number, which decides for how many timesteps ($\tau \cdot \Delta t$) to solve each fine-resolution subdomain independently (without communications between each other), see reference [1]. Finally, we can save computation during $J(t) \cdot \tau \cdot \Delta t$ and this leads to efficiency gain.

In this formulation, there is no projection when the diffusion distance is the maximum, while there is the biggest projection time when the diffusion distance is the minimum. There is a trade-off between the computational efficiency and accuracy via the tuned parameter $\alpha$. The tuned parameter $\alpha$ is obtained by prior knowledge about flow physics (proportional to $\alpha$). In this framework, $\alpha$ is set to 1 as a default, which assumes there is no prior knowledge about flow physics.

4. Multi-fidelity and heterogeneous auxiliary data

In order to demonstrate the capability of the information fusion with multi-fidelity and heterogeneous auxiliary data, we introduce two stochastic and particle-based auxiliary simulations, namely a two-dimensional random walk model for the heat equation and a dissipative particle dynamics (DPD) model for the Navier–Stokes equations in this section. In order to understand and characterize the capability for multi-fidelity and heterogeneous simulations, we used accurate and expensive models (but small number of particles) as the auxiliary simulation for analysis purpose. In the future, we will use “cheap and dirty but useful” auxiliary simulations. Details for simulation and problem setup are presented below.

4.1. Two-dimensional random walk model for the heat equation

The two-dimensional heat equation in a uniform rectangular grid is solved by a simple random walk model via the Monte Carlo method. Specifically, by Fick’s law, a particle with temperature $T(x, \tau)$ at a point $x$ can move north, south, east, or west with equal probability, $p(N) = p(S) = p(E) = p(W) = 0.25$, after a unit time $t = \tau + \Delta t$. Hence, after $t = \tau + n\Delta t$, all particles originally located at the point $x$ are distributed by the random walk model. Using those distributed data, we
calculate the updated temperature $T(x, t + n\Delta t)$ at every grid point. The equation for calculating a heat quantity in the arbitrary domain $D$ after time $n\Delta t$ is as follows:

$$T(x^*, \tau + n\Delta t) = \int_T T(x, \tau) d\mu(x) \quad \forall x \in D,$$

where $\mu(x)$ is a probability measure representing the probability that a particle moves from $x$ to $x^*$ after time $n\Delta t$ ($n$ steps).

As shown in Fig. 3, the continuous model is rewritten as a two-dimensional $10 \times 10$ grid discrete model with $N$ sample paths (particles) of the Monte Carlo method as:

$$T(x^*, \tau + n\Delta t) = \sum_{x \in D} T(x, \tau) P(x),$$

where $x$ is the position of each node $(i, j)$ and $P(x)$ is a discrete probability measure calculated by the Monte Carlo method:

$$P(x) = \frac{n_p}{N_p},$$

where $N_p$ is the number of total sample paths originating from $x$, and $n_p$ is the number of those paths that land at $x^*$ at time $t + n\Delta t$.

In this work, we choose 20, 200 and 2000 sample paths at each grid point (totaling 2000, 20000 and 200000 computations, respectively) for calculating the updated temperature at each point through equations (9) and (10). By the equivalent numerical discretization corresponding to the random walk model, the diffusivity $\kappa = h^2/4\Delta t$ is determined by temporal and spatial discretization ($\Delta t$ and $h$) of the random walk model. Temperature contours of different auxiliary data are shown in Fig. 4.

4.2. Dissipative Particle Dynamics (DPD) for the Navier–Stokes equations

Next, we introduce a dissipative particle dynamics (DPD) model, a particle-based mesoscale simulation, as auxiliary data-producing simulator for the Navier–Stokes equations. The DPD method uses virtual particles, which represent “molecular clusters” moving together in Lagrangian fashion. In DPD systems, we assume that particles, whose mass $m_i$, position $\mathbf{r}_i$, and velocity $\mathbf{v}_i$, interact with each other by pairwise-additive forces, which consist of three terms: (1) conservative force ($F_{ij}^C$), (2) dissipative force ($F_{ij}^D$), and (3) random force ($F_{ij}^R$):

$$F_{ij}^C = F_{ij}^C(r_{ij}) \mathbf{r}_{ij},$$

$$F_{ij}^D = -\gamma \omega^D(r_{ij})(\mathbf{v}_{ij} \cdot \mathbf{r}_{ij}) \mathbf{r}_{ij},$$

$$F_{ij}^R = \sigma \omega^R(r_{ij}) \xi_{ij} \mathbf{r}_{ij},$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, $r_{ij}$ is a distance between $i$ and $j$ particles; $\gamma$ and $\sigma$ are hyper-parameters for dissipative and random force, respectively; $\omega^D$ and $\omega^R$ are distance dependent weight functions, and $\xi_{ij}$ is a normally distributed random variable. The conservative force $F_{ij}^C(r_{ij})$ with a cutoff radius $r_c$ is obtained by

$$F_{ij}^C(r_{ij}) = \begin{cases} a_{ij}(1 - r_{ij}/r_c) & \text{if } r_{ij} \leq r_c; \\ 0 & \text{if } r_{ij} > r_c; \end{cases}$$

where $a_{ij} = \sqrt{\sigma^D}$ and $a_i, a_j$ represent conservative force coefficient for particle $i$ and $j$, respectively. For details of DPD systems, see reference [34].
Fig. 4. Temperature contours of different auxiliary data for the heat equation at time $t = 15$ (a transient period). (a)–(c): the finite difference method. (d)–(f): the random walk model by the Monte Carlo method.

Fig. 5. Streamwise velocity contours of different auxiliary data for the Navier–Stokes equations at time $t = 5$ (a transient period). (a)–(c): the finite difference method. (d)–(f): the DPD model.

In this paper, the kinematic viscosity, $\nu$, of the DPD fluid is equal to 2.86 (in DPD units), corresponding to Reynolds number $Re = 35$. This value was obtained by fitting a double parabola to the DPD results using the reverse Poiseuille flow method described in reference [35]. For boundary condition, we employ boundary particles with particle reflection rules, for details, see reference [34]. The field variables of the DPD results are obtained by a spatiotemporal averaging. In space, the DPD results are averaged on $10 \times 10$ rectangular bins, total 100 bins. In particular, we take average of all field variables in each bin and set this mean value as a representative field variable of each bin. In time, the DPD results are averaged over every 1000 time steps with $t = 0.1$ with a same process in space. The accuracy of averaged values is dependent on the number of particles, which is controlled by the length in the $z$-direction. Due to our definition of the auxiliary data as relatively costless, we employ small numbers of particles like 1000, 4000, and 8000 DPD particles to obtain field variables. Streamwise velocity contours of different auxiliary data are shown in Fig. 5.
5. Simulation setup

5.1. Heat equation

First, we solve the two-dimensional heat equation with diffusivity $\kappa$,

$$\frac{\partial T}{\partial t} = \kappa \nabla^2 T.$$  (15)

In order to perform a patch simulation and a reference simulation on the complete domain, we employ the second-order finite difference method. We employ Dirichlet boundary condition for physical boundary conditions, which are described in Fig. 6(a). The grid resolution of the patch simulation is $11 \times 11$ per each sub-domain (total of 5 sub-domains), with a total of 605 grid points in the patch simulation, while the grid resolution of the reference simulation is $31 \times 31$. In order to investigate the influence of multi-fidelity auxiliary data on the overall accuracy, we employ not only a coarse grid with resolution $4 \times 4$, $6 \times 6$, and $10 \times 10$, but also the random walk model with different numbers of sample paths as 20, 200, and 2000 on a $10 \times 10$ grid. We use an ensemble average of the data by 30 trials of the random walk model. Boundary conditions for each fine-resolution sub-domain are of the Dirichlet type estimated by multi-level Gaussian process regression with auxiliary data. The temporal discretization corresponds to time step $\Delta t = 0.01$ with heat diffusivity $\kappa = 1/60$ given by the random walk model. The non-interaction timestep number $\tau$ is 15. The discretized equations are integrated from $t = 0$ to $t = 30$ (the steady-state is established around $t = 25$).

5.2. Navier–Stokes equations

Next, we consider incompressible flow for the lid-driven cavity described by the divergence-free Navier–Stokes equations:
6. Results

The contour plots of field variables compared to reference simulations (no spatial gaps or projective time integration) qualitatively are shown in Fig. 7. Also, the results of the patch simulation with different auxiliary data and projection time, compared to the results of the reference gappy simulation (no projective time integration), are shown in Figs. 8–11. The green line represents the result of the gappy simulation, which has no projective time integration as in previous work [1]. The blue dots represent the result of the patch simulation, which employ projective time integration with different projection time. The quantitative comparisons among different auxiliary data are shown in Fig. 12.

In order to compare results of patch simulations with different auxiliary data, we employ a measure of RMS error, namely the “total” RMS error (RMSET) for temporal accuracy. The “total” RMS error at time $t$ is calculated by the following formula:

$$
\text{RMSET}(t) = \sqrt{\frac{1}{N_t N_g n_i} \sum_{j=1}^{N_g} \sum_{i=1}^{n_i} (u_{r,j}(i,t) - u_{p,j}(i,t))^2},
$$

where $\mathbf{v}$ is the velocity vector, $p$ is the pressure, and $\nu$ is the kinematic viscosity of the fluid. We employ a second-order finite difference method. The physical boundary conditions are of Dirichlet type, as shown in Fig. 6(b). The resolution of a rectangular cell for each sub-domain is $8 \times 8$, with a total of 320 cells for the patch simulation. In order to compare errors from different auxiliary data quantitatively, we employ not only coarse simulations with resolution $4 \times 4$, $8 \times 8$, and $16 \times 16$ for the entire domain, but also DPD simulations with different numbers of DPD particles as 1000, 4000, and 8000. For the DPD result, we use an ensemble average data of 30 trials. We employ Dirichlet boundary conditions for each fine-resolution sub-domain estimated by multi-level Gaussian process regression with auxiliary data. The time step is $\Delta t = 0.005$ with Reynolds number $Re = 35$ given by the DPD model. The non-interaction timestep number $\tau$ is 20. The discretized equations are integrated from $t = 0$ to $t = 9$ (the steady-state is established around $t = 5$).
Fig. 8. The time history of time-averaged RMS error for different projection time in the heat equation. (b) and (d) have different RMSE scales. The green line represents a solution of the gappy simulation, which has no projective time integration. The blue dots represent a solution of the patch simulation with fixed projection time. (a)–(b): The auxiliary data comes from a finite difference model with resolution $6 \times 6$. (c)–(d): The auxiliary data comes from the random walk model with 2000 sample paths. We employ 10 snapshots in the projective time integration. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

where $N_g$ is the number of grid points of each fine-resolution sub-domain and $n_s$ is the number of fine-resolution sub-domains, while $u_{r,j}(i, t)$ and $u_{p,j}(i, t)$ represent the fine-resolution reference solution and the patch solution on $j$th fine-resolution sub-domain, respectively.

As shown in Figs. 8 and 10, for a small fixed projection time, the difference between RMSE$_t$ of the gappy simulation and the patch simulation is relatively small. However, too many computations are needed near the steady-state when the change of dynamics is negligible. For a large fixed projection time, on the other hand, a large difference in RMSE$_t$ is observed during the transient period. For projection time by diffusion maps in Figs. 9 and 11, the gap between results of the gappy simulation and the patch simulation is small enough during the transient period (due to a small projection time), and the computation redundancy is also small enough near the steady-state (due to a large projection time).

In order to quantify these observations, we introduce three different measures: namely “a scaled maximum difference of RMSE$_t$ ($D_M$)” for quantifying the accuracy, “a normalized computation time ($T_C'$)” for the efficiency, and “an overall accuracy ($\zeta$) compared to a reference gappy simulation. As shown in Fig. 12, the RMS errors converge as the accuracy of auxiliary data increases for both the finite difference method and the particle based method. In the patch simulation, we add the projective time integration to the gappy simulation for acceleration of simulation. Since spatial gaps are preassigned by software or hardware error, the RMS error and computational time of the gappy simulation, which has same spatial gaps of
However, as accuracy and gives projection accuracy (the heat paper, the 2000 maps. The simulation, Fig. 9.

In the patch simulation, are used to measure “accuracy loss” and “efficiency gain” of the patch simulation. Hence, we employ the gappy simulation as the reference simulation to measure accuracy and efficiency in the same circumstance. In this paper, we choose as auxiliary data simulations from the finite difference method with a 24 × 24 grid resolution for both the heat equation and the Navier–Stokes equations in the reference simulations. Then, the $D_M$ is calculated by the difference between RMSE of two simulations:

$$D_M = \max \left( \left| \text{RMSE}_p^g(t) - \text{RMSE}_p^g(t) \right| \cdot R, \quad t \in (0, T) \right),$$

where superscripts “p” and “g” represent the patch simulation and the gappy simulation, respectively, while $R$ is the ratio of the mean RMSE^g between each auxiliary data and the reference auxiliary data. Then, the RMS error of all auxiliary data can be scaled with respect to the reference gappy simulation. Another measure $T_C$ is calculated by the ratio of total computation time of the patch simulation to the reference gappy simulation. There are inverse correlations between the two aforementioned measures, i.e., if $D_M$ becomes larger, then $T_C$ becomes smaller. Hence, we introduce the “overall accuracy ($\zeta$)”: $\zeta = 1 - (D_M \cdot T_C).$

6.1. Accuracy versus efficiency

In the heat equation, for cases of fixed projection time only, $T_C$ decreases linearly while $D_M$ grows exponentially as projection time increases (see Fig. 13). Hence, in the projective time integration, a small projection time requires a large computation time linearly while a large projection time loses accuracy exponentially. That is, a small projection time has higher overall accuracy compared to a large projection time. However, varying the projection time by the diffusion distance gives a value for $D_M$ that is similar to the value for a fixed projection time of 2τ, and a value for $T_C$ that is similar to the value for a fixed projection time of 6τ. This leads to a better overall accuracy ($\zeta$). The patch simulation with the random walk model (stochastic and particle based auxiliary data) has the same trend as the finite difference method (deterministic and grid based auxiliary data), i.e., varying projection time by the diffusion distance always provides the highest overall accuracy for any auxiliary data. The random walk model with 2000 sample paths is found to be the best auxiliary data with respect to the overall accuracy, although the finite difference model with a 10 × 10 grid is comparable. Since the random walk model solves the heat equation on a 10 × 10 grid resolution, $D_M$ appears to be similar to the finite difference method with a 10 × 10 grid.

In the Navier–Stokes equations, for cases of fixed projection time with a finite difference method only, results are same as the heat equation, i.e., varying projection time by the diffusion distance provides the highest overall accuracy; $D_M$ is at a similar level as for a fixed projection time of 2τ, and $T_C$ is at a similar level as for fixed projection time of 4τ or 6τ. However, the DPD model with 1000 particles (too coarse) cannot provide an adequate projection time due to its incorrect slow manifold, which results in a decrease in overall accuracy even though we employ the varying projection time. As the number of DPD particles increases from 1000 to 8000, the diffusion map provides the appropriate projection time, which
leads to large improvement of both the accuracy and the efficiency (see Fig. 14). Furthermore, since differences in $T_C$ for all cases are negligible (except the finite difference with $4 \times 4$ grid), $D_M$ determines the overall accuracy. Finally, the finite difference model with a $16 \times 16$ grid (the lowest $D_M$) is found to be the best auxiliary data with respect to the overall accuracy.

From the results of all simulations, we conclude that varying projection time at every projection step, based on the rate of temporal variation of the local leading diffusion maps coordinate, is the best method to achieve the highest overall accuracy for any auxiliary data. Moreover, the heterogenous models (based on particles) are comparable as the number of particles increases. This result shows that the patch simulation provides high quality even when using heterogeneous information fusion.

6.2. Projection time and auxiliary data

Diffusion maps provide a non-linear low-dimensional manifold associated with the slow dynamics of the original problem. The diffusion maps from accurate auxiliary data draw a smooth coordinate which contains slow dynamics precisely and this results in increasing projection time compared to inaccurate auxiliary data (see Fig. 15). Hence, $T_C$ decreases as the
Fig. 11. The time history of time-averaged RMS error for different projection time in the Navier-Stokes equations. The green line represents a solution of the gappy simulation, which has no projective time integration. The blue dots represent a solution of the patch simulation with varying projection time by Diffusion maps. (a): The auxiliary data comes from a finite difference model with resolution $8 \times 8$. (b): The auxiliary data comes from the DPD result with 4000 DPD particles. We employ 5 snapshots in the projective time integration. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

Fig. 12. RMSE$_{\tau}$ for different auxiliary data. The RMS errors are converged as the accuracy of the auxiliary data increases in both cases. The accuracy of the auxiliary data also affects computation time. For example, finite difference methods with high accuracy require a higher order formula or a more refined grid resolution, which results in consuming extra computation time to perform the auxiliary simulation. If we employ more refined grid resolution, we need additional computational time to obtain optimal hyper-parameters for the multi-level Gaussian process regression by the maximum likelihood estimation. Hence, more accurate auxiliary data cannot guarantee smaller computation time. However, more accurate auxiliary data provides more accurate snapshots, which results in a smaller $D_M$ with the same projection time. Finally, we show that more accurate auxiliary data gives higher overall accuracy ($\zeta$).

Next, we focus on projection time of heterogeneous models (stochastic and particle-based models). As shown in Fig. 15, projection time is bounded above and has wiggles near the steady-state. As the system approaches the steady-state, the change of dynamics is very small (or negligible). This, in turn, means that the diffusion distance between adjacent snapshots at that time is very small. Thus, projection time (inverse logarithm scale of the diffusion distance) becomes very large and it can be affected strongly by a small perturbation of each snapshot. Thus, intrinsic perturbations of both stochastic models, the random walk model and the DPD model, cause the oscillation of projection time near the steady-state in both benchmark problems.
6.3. Number of snapshots

The number of saved snapshots, \( N \), determines the number of terms of a POD expansion for reconstructing field variables in the projective time integration. Generally, more snapshots for the projective time integration with same auxiliary data and projection time reduce the truncated error of POD expansion and this leads to gaining accuracy [16] while the extent of simulation “saved” becomes less, leading to less efficiency.

In the heat equation, as shown in Figs. 16(a)–(c), accuracy gains with respect to \( D_M \) are very small even though \( T_C \) increases. This result shows that 5 snapshots, i.e., 5 terms of POD expansion, are enough to reconstruct and estimate field variables since the governing equation is linear. On the other hand, for the Navier–Stokes equations (which are much more complex than the heat equation and nonlinear equations), \( D_M \) decreases as the number of snapshots increases (see Fig. 16(d)). The additional snapshots can reconstruct the field variables precisely, which leads to high accuracy for the patch simulation by using more terms of POD expansion. However, additional snapshots require more time steps to perform fine simulations for each projective time integration, which make \( T_C \) increase (see Fig. 16(e)). From our observations, \( D_M \) has a more pronounced effect than \( T_C \) on the overall accuracy. Hence, as shown in Fig. 16(f), more snapshots are suitable for this problem setup. Generally, the nonlinear governing equation requires more snapshots to gain overall accuracy.

7. Summary

We formulated and implemented a new framework to extend gappy simulation with a “patch dynamics” flavor and tested simulations of two benchmark problems, namely the heat equation and flow in a lid-driven cavity in two dimen-
Fig. 14. Three computational quality measures in the Navier–Stokes equations: the x-axis represents a fixed projection time for the projective time integration and “DMAP” represents a varying projection time by the diffusion maps. Square and triangle markers represent auxiliary data from a finite difference model with a coarse grid and a DPD model, respectively.

Fig. 15. Projection time with respect to time by the diffusion maps. The dashed lines represent projection time for the finite difference model in both cases. The solid lines represent projection time for the random walk model in the heat equation and the DPD model in the Navier–Stokes equations, respectively.
Fig. 16. Results of three computational quality measures with different number of snapshots. x-axis represents the number of snapshots for the projective time integration. (a)–(c): in the heat equation, the blue and the red represent the coarse grid (10 × 10) and the random walk model with 2000 sample paths, respectively. (d)–(f): in the Navier–Stokes equations, the blue and the red represent the coarse grid (8 × 8) and the DPD model with 4000 particles, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
sions. Furthermore, we demonstrate the new capability that statistical learning tools help achieve with heterogeneous and multi-fidelity data. By learning from the auxiliary data, the new algorithmic framework can achieve not only high accuracy via the Gaussian process regression, but also high temporal integration efficiency via the diffusion maps. From results of simulations, we observed important trends about the patch simulation. Accurate auxiliary data provides accurate data sets in the Gaussian process regression and the appropriate projection time from the diffusion map. Moreover, the number of snapshots we use for the projective time integration totally depends on the dynamics of the problem. Generally, more snapshots increase the accuracy but decrease the efficiency.

We demonstrated that this framework can be extended to enable multi-fidelity and multiscale parallel simulations in a resilient way. In future work, we will further extend this framework to address multi-rate dynamics. For example, we can employ velocity field data as the auxiliary data for simulating concentrations of species driven by a convection–diffusion equation. Then, multi-species concentrations in the same velocity can be solved much more accurately and efficiently by considering only one velocity problem (as the auxiliary data) in this framework. It is also highly desirable to investigate the numerical stability of this framework with respect to auxiliary data, gap size and projection time, and we are currently pursuing such an investigation.

Acknowledgments

S. Lee and G. E. Karniadakis gratefully acknowledge the financial support of the Army Research Office (ARO W911NF-14-1-0425), and also the Air Force Office of Scientific Research (AFOSR FA-9550-12-1-0463). I. G. Kevrekidis gratefully acknowledges the partial support of the National Science Foundation. All DPD simulations were performed at the facilities of Center for Computation and Visualization in Brown University.

References