Data compression for turbulence databases using spatio-temporal sub-sampling and local re-simulation

Zhao Wu,[1] Tamer A. Zaki,[1] and Charles Meneveau[1]

Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218, USA

(Dated: November 30, 2020)

Motivated by specific data and accuracy requirements for building numerical databases of turbulent flows, data compression using spatio-temporal sub-sampling and local re-simulation is proposed. Numerical re-simulation experiments for decaying isotropic turbulence based on sub-sampled data are undertaken. The results and error analyses are used to establish parameter choices for sufficiently accurate sub-sampling and sub-domain re-simulation.

Keywords: data compression; sub-domain re-simulation; spatio-temporal sub-sampling; Navier-Stokes

I. INTRODUCTION

In the field of computational fluid dynamics, the study of turbulent flows based on data generated using Direct Numerical Simulations (DNS) has occupied a prominent place in the literature over the past several decades[1–5]. DNS provides spatial and temporal resolution down to the smallest and fastest eddies of a turbulent flow. Therefore, the Reynolds number achievable by DNS is limited by computing power and memory, and has been growing roughly at the rate expected from Moore's law. The amount of data generated by DNS is growing accordingly[6–10]. For instance, a simulation of turbulent flow outputting four field variables (e.g. the three velocity components and pressure) on 2000^3 spatial grid points and integrated over, say, 5×10^4 time-steps, will generate several Petabytes (PB) of data. Researchers thus store only a few selected snapshots of the flow during the simulations, and primarily rely on run-time analysis tools that are decided prior to the computation if time resolved phenomena are to be studied. As a result, when new questions and concepts arise, massive simulations must be performed over and over. Moreover, when storing snapshot data for later analysis, the traditional means of sharing available data after DNS, e.g. del Alamo and Jimenez [11], assumes that the data are downloaded as flat files and consequently a user has to worry about formats and provide the computational resources for analysis.

As a means to address these problems that challenge further growth of DNS and accessibility of data, modern database technologies have begun to be applied to DNS-based turbulence research. For instance, the Johns Hopkins Turbulence Database (JHTDB, http://turbulence.pha.jhu.edu)[12,13], has been constructed and has been in operation for about a decade, as an open public numerical laboratory. The system hosts about 1/2 petabyte (PB) DNS data including 5 space-time resolved data sets and several others with a few snapshots available. Users have Web-service facilitated access to the data, among others using a "virtual sensors" approach in which a user specifies the position and time at which data are requested and the system returns properly interpolated field data. Other derived quantities such as gradients[13] and fluid trajectories[14] are also available, typically delivered to within single-precision machine accuracy. A hallmark of the system is the ability of users to access very small targeted subsets of the data without having to download the entirety of the data. The system has been successful at democratizing access to some of the world's largest high-fidelity DNS of canonical turbulent flows. JHTDB data have been used in over

160 peer-reviewed journal articles since its inception, about 40 in 2019 alone.

In recent years, the scale of DNS data has continued to grow further. The largest simulations now generate data on about $O(10^4)$ grid points in each of the three directions, so storing multiple time steps to capture time evolution becomes very challenging, even in efficiently built databases. For example, storing even only one large-scale turnover time of the 8192^3 isotropic turbulence data set[6] would require storing about 80 PB. Over the next several years, it can be anticipated that even larger scale DNS will be performed, generating exabytes of data, far out of reach of anticipated facilities and the approaches on which JHTDB is currently based.

It is therefore necessary to explore innovative tools for compressing simulation data for use in conjunction with databases. Most of the general-purpose data compression algorithms are based on analyzing the data representation, and can generally be classified as lossless or lossy. Lossless data compression utilizes the statistical redundancy[15,16], while lossy data compression is to remove unnecessary data, e.g., JPEG[17] and MP3[18]. Lossless data compression tools are promising but for turbulence data where the flow's small-scale structures contain non-trivial information at each grid point, the compression ratios can be expected to be somewhat limited. While we continue current efforts along this direction and can expect further improvements, more aggressive tools will be required for the very large data sets envisioned in the near future. Regarding lossy compression, it is certainly appropriate for visualization and other applications where less fidelity is acceptable. However, if one wishes, e.g. to capture accurately velocity gradients, lossy compression algorithms in which the accuracy of primary variables is degraded, say, at the fourth decimal point, will already lead to significant errors in gradients and will thus be insufficient for the purposes of turbulence research.

It bears recalling that JHTDB enables users to receive interpolated data between spatial and temporal grid points, using polynomial functions (Lagrange, spline, Hermite). Far more aggressive data compression could be achieved if data could be stored more sparsely in both space and time. However, when a user requests localized pieces of data that fall between coarsely stored positions and/or times, one would need to revert to the dynamical equations (i.e. Navier-Stokes) to perform a physics-based rather than a polynomial based interpolation.

In this paper we explore and establish requirements for such a data compression method, named "Spatio-Temporal Sub-sampling and sub-domain Re-simulation" (STSR). The method aims at enabling users to recover data at close to machine accuracy (single-precision), based on very coarsely stored data. While the method can greatly compress the amount of data to be stored, such savings have to be balanced by the additional cost of processor (CPU or GPU) expense needed later on to accommodate user queries.

Initial efforts attempting to reproduce DNS data using local re-simulation (technical details to be provided below) have shown a surprisingly narrow and stringent range of conditions under which re-simulation in a sub-domain can generate data at the desired accuracy. That is to say, re-simulation that reproduces DNS at close to single-precision machine accuracy, the desired baseline accuracy level, is more difficult to achieve than one may expect. Any small deviations from the conditions to be developed can be shown to lead to significant errors. It will be observed that the errors do not arise due to chaotic dynamics as we do not observe exponential divergence of state-space trajectories or exponential growth of errors over time. The absence of chaotic divergence of dynamics may be due to the strong constraints introduced by boundary conditions prescribed around closed sub-domains i.e. that the ratio of sub-domain size to viscous length scale is sufficiently small for synchronization of chaos[19,20] to occur in the cases tested. Instead, errors are introduced due to small details of numerical implementation, discretization, and order of operations that at first glance may appear small and trivial but that can cause rather significant differences in results.

Therefore, the present paper aims to document the technical methodologies and tests performed with considerable attention to detail. Section II introduces the basic idea of data compression for turbulence databases using STSR. The desire to enable re-simulations over localized spatial domains precludes the use of spectral methods based on global basis functions. In this work, we explore the use of one of the most common discretization tools in CFD: second order finite differencing. The numerical scheme adopted in the present computations is described in Section III. The methodology is tested in the decaying isotropic turbulence, a well-understood and relatively simple flow described in Section III B. In Section IV, the influence of the boundary conditions on reproducibility of the simulations, up to the desired level of machine precision, is examined. The re-simulation errors are studied in Section V in more detail, and their dependence on artificially introduced noise in boundary conditions is established in order to better understand requirements for reaching desired levels of accuracy, which are slightly relaxed from machine accuracy down to relative errors at the order of $\sim 10^{-5}$ based on practical considerations. Section VI showcases an application using the recommended parameters. Finally, conclusions are presented in Section VII. The paper is limited to an account of the findings regarding methodology and requirements in the context of a simple flow at moderate computational scale. Construction of a large turbulence database system using the proposed STSR querying method is left as a future task.

II. SUB-SAMPLING AND LOCAL RE-SIMULATION

In this section, the basic concept of the proposed STSR approach is explained, together with an estimate of the data compression that can be achieved. Figure 1 is a two-domensional schematic of a DNS domain and the storage scheme of the data to enable later re-simulation. The flow domain inside the box in Figure 1(a) represents the entire, or global, domain of the original simulation, e.g. from a simulation of isotropic turbulence, channel flow, boundary layer, etc. The global domain consists of a large number of grid points; in 3D, say, $N^3 = N_x N_y N_z$. By enforcing initial and boundary conditions on the global domain boundaries, the simulation is advanced forward in time, at a time-step δt. The objective is to store a limited amount of data at each time-step in order to enable re-simulation of a sub-region of the global domain. For this purpose, the global domain is divided into small sub-volumes marked by the blue boundaries (Figure 1(b)) corresponding to planes in a 3D domain. For simplicity, the sub-volumes here have the same shape and dimensions but the discussion and general results to be presented can be considered quite general. While the main simulation is performed, the state vector (i.e. velocity and pressure fields for incompressible flow) is stored on these planes. If the size of an individual re-simulation sub-domain is M_s, in 3D there will be $3(N/M_s)$ such planes, each of size N^2.

Moreover, in order to limit the CPU cost of re-simulation, after a number of time-steps, the state vector data are stored at every grid point in the global domain. This occurs every M_t time steps, i.e. after a time equal to $M_t\,\delta t$ (see Figure 1(c)). In the rest of this paper, $t_n = n\,\delta t$ represents the physical time, while n represents the time step of the DNS. For a simulation lasting a total time T, the total number of full 3D fields to be stored is thus equal to $\sim T/(M_t \delta t)$.

After the direct simulation in the global domain has been completed and the sub-sampled data stored, data at a specific spatial and temporal location $(\boldsymbol{x}, t)$ may be required, for example to examine local flow states in particularly interesting sub-regions of the flow or to

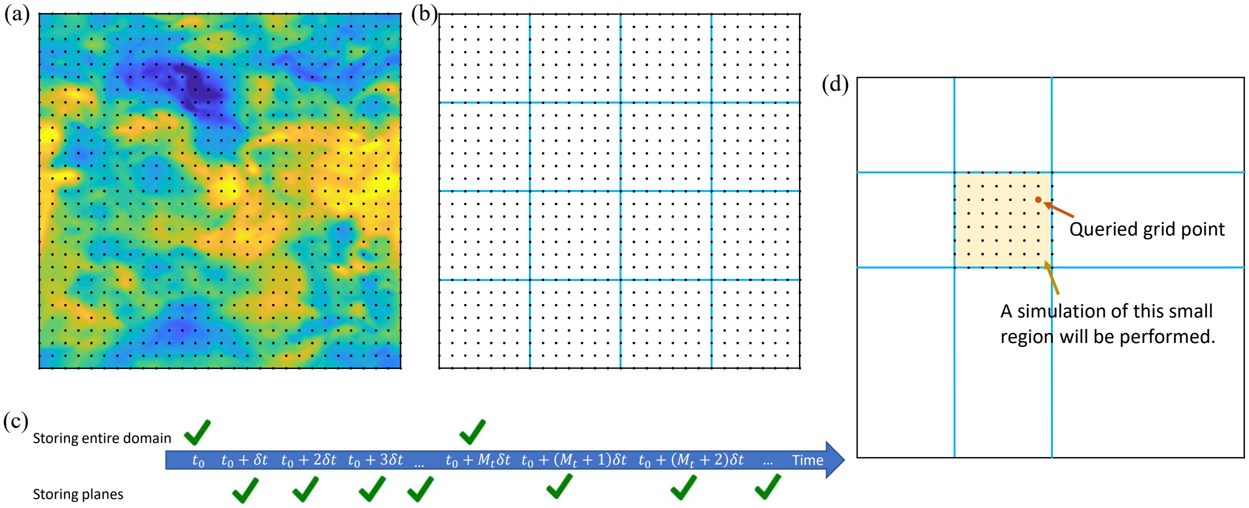

Figure 1. Schematic of STSR. (a) Entire DNS domain containing a large (N^3) number of grid points. (b) The entire DNS domain are divided into small cube regions by the blue lines. (c) The storage scheme of the spatio-temporal subsampling for re-simulation. The data in the entire domain are stored at every M_t time step. The data on the planes (blue lines) and on the outer planes (black lines), are stored at every time step. (d) When data are required on grid points and time steps that are not stored in the database, a re-simulation of a small region which includes the queried grid point is performed to obtain the data.

track particles through the flow. In general these locations do not correspond to stored data, and the data must be evaluated by re-evaluating the flow evolution in the host sub-volume and time interval (Figure 1(d)).

Similar to the global domain, the flow in the re-simulation sub-domain is governed by the continuity and Navier-Stokes equations. The numerical solution requires the initial and boundary conditions. Suppose there exists an integer n such that $(t_0 + nM_t\delta_{\rm dns} < t < t_0+(n+1)M_t\delta t_{\rm dns})$, i.e. the time at which data are sought t lies between two instances where the entire global domain was stored. The data stored at $(t_0 + nM_t\delta_{\rm dns})$ can then be used as the initial condition, and the plane data on the sub-domain boundary that was stored at every time step between times $(t_0 + nM_t\delta_{\rm dns})$ and t provide the boundary conditions needed for re-simulation. Unless otherwise stated, the original simulation and its re-simulation will adopt the same time step for forward integration of the governing equations.

To fix notation, in the rest of this work the “global domain” refers to the domain of the original simulation (the black enclosing box in Figure 1(d)); a “sub-domain” refers to the much smaller region containing a queried point or sets of points (the yellow region in Figure 1(d)); and "re-simulation" refers to numerical solution of the governing equation in this subdomain using initial and boundary conditions extracted during the original computation and stored in the STSR database.

With the proposed approach, only a small fraction of data is stored and the fields can be re-constructed on demand from simulations within the small sub-regions. The data compression (inverse) ratio c can be estimated as

$$c \approx \frac{N^3 + 3N^2(N/M_s)(M_t-1)}{N^3 M_t} = \frac{1}{M_t}\left(1 - \frac{3}{M_s}\right) + \frac{3}{M_s}, \tag{1}$$

where N is the number of grid points in each direction in the entire domain.

Hence, if for example $M_s = 128$ is used, and we store only every $M_t = 200$ full 3D fields, the total storage requirement is about 2.8% of the original data. Performing the re-simulation in the M_s^3 sub-domain is certainly much faster than doing a re-simulation in the original full 3D volume: the CPU cost of re-simulation is approximately $M_t(12M_s^3 + M_s^3 \log_2 M_s)$. Depending on the ratio of cost of storage and computation, as well as depending on patterns of data queries and usage, the optimal values of M_s and M_t could vary significantly. For now we simply observe that the 8192^3 grid database with $\sim 10^4$ time steps mentioned in the introduction requiring over 80 PB of storage, would require only about 2.2 PB if stored using sub-sampling with $M_s = 128$ and $M_t = 200$, and the computational cost of the re-simulation is only $\mathcal{O}(10^{-6})$ of the cost of the full simulation.

The approach becomes particularly attractive in studies where only small sub-regions of the flow need to be interrogated later on. For example, in particle tracking studies, one only needs velocities in the immediate vicinity of particles to be used for interpolation. In other studies, researchers may want to zoom into areas where extreme events such as core of vortices or high dissipation take place. Or, one may wish to obtain a one-dimensional spectrum along some representative lines through the flow requiring data only along those lines rather than the entire domain. In such scenarios, storing the entire data or having to perform re-simulation in the entire domain would be unnecessary and waste computational/storage resources.

One might consider the present methodology is similar to data assimilation[19,21,22] or "nudging"[23] to deal with incorporation of incomplete and/or imperfect (noisy) data. In nudging, a penalization term is included in the Navier-Stokes equations, so that the re-simulation result would be pulled towards the original (observation) data. In this case, deviation in initial condition is allowed, and the re-simulation result will match the original data after several time steps, depending on initial condition, the penalization term, flow condition etc. However, the number of time steps needed for the re-simulation to catch up to the original data is difficult to assess without using the correct initial condition. Therefore, for the purpose of reusing the DNS data, providing correct initial condition becomes a necessary condition in this study.

III. NUMERICAL SCHEME AND FLOW CONFIGURATION

Incompressible flow of a Newtonian fluid satisfies the continuity and Navier-Stokes equations written here in skew-symmetric form,

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

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{1}{2}(\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u}) = -\nabla p + \nu \nabla^2 \boldsymbol{u}, \tag{3}$$

where $\boldsymbol{u} = (u, v, w)^T$ is the velocity vector, t is time, and ν is the fluid kinematic viscosity. The three velocity components u, v and w correspond to the x, y and z directions, respectively, and p is pressure divided by density. The advection term in equation (3) is expressed in the skew-symmetric form which conserves kinetic energy and reduces aliasing errors[24]. However, other forms of the advection term can also be adopted.

A. Temporal and spatial discretization

A δp-form prediction-correction algorithm[25–27] is used to decouple the velocity and pressure:

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^{(n-1)}}{\delta t} = -\text{ Conv.} + \text{Diff.} - G(p^{(n-1)}), \tag{4}$$

$$DG\phi^{(n)} = \frac{D\boldsymbol{u}^*}{\delta t}, \tag{5}$$

$$\boldsymbol{u}^{(n)} = \boldsymbol{u}^* - \delta t G\phi^{(n)}, \tag{6}$$

$$p^{(n)} = p^{(n-1)} + \phi^{(n)}, \tag{7}$$

where δt is the time step, superscript $(\cdot)^n$ denotes the n-th step. Conv. is the discretized convective term, Diff. is the discretized diffusive term, G is the discretized gradient operator,

D is the discretized divergence operator, and ϕ is the pressure difference between two time steps. The advection term can be advanced in time explicitly using explicit Euler or second-order Adams-Bashforth (AB2) scheme; the viscous term can be advanced using Euler, AB2 or implicit Crank-Nicolson (CN) scheme.

A variant of the projection method referred to as the p-form[28,29] ignores the pressure gradient term in the prediction step (4), and therefore $\phi^{(n)}$ in the Poisson equation (5) is an approximation of the full pressure at the new time step, i.e. $p^{(n)} = \phi^{(n)}$. An notable difference between the herein adopted δp and the p forms is in the boundary conditions: (i) the boundary condition of the elliptic pressure equation is the pressure difference in the δp-form, and the pressure in the p-form; (ii) in terms of the velocity, in order to ensure second-order accuracy, one should enforce $\boldsymbol{u}^* = \boldsymbol{u}_\Gamma$ on the boundary of the computational domain Γ in the δp-form, but $\boldsymbol{u}^* = \boldsymbol{u}_\Gamma + \delta t\, G p_\Gamma^{(n-1)}$ in the p-form. In the present study, the δp-form is adopted throughout. Although not presented here, use of the p-form does not affect our results nor conclusions.

A staggered grid[30] is used in order to avoid checkerboard pressure oscillations. The spatial derivatives are approximated with second-order central finite differences. In light of the computational cost of the pressure equation (5), it is important to ensure that the re-simulation does not compromise any of the efficiency of the global solver. For instance, if the global domain is triply periodic, Fourier transform can be adopted in all three dimensions and the solution of (5) is inexpensive. The re-simulation sub-domain is, however, not periodic; we nonetheless adopt a fast Poisson solver using discrete sine and cosine transforms[31]. Details on the pressure Poisson solver used in re-simulations are provided in Appendix A.

B. Flow configuration: decaying isotropic turbulence

The flow adopted in this work as an example application of STSR is decaying isotropic turbulence in 3D. The global domain has dimensions $2\pi\times2\pi\times2\pi$, and is discretized uniformly using 256^3 grid points ($N = 256$); the grid spacing is $h = \Delta x = 0.02454$. The domain is periodic in all three spatial directions. Time integration of the viscous and convective terms starts with one Euler step at the initial condition, and is subsequently evolved using AB2. A snapshot from an 1024^3 isotropic turbulence data set (https://doi.org/10.7281/T1KK98XB) in JHTDB is used as the initial condition, sub-sampled every 4 grid points. After

Time	RMS vel.	Dissipation	Re-number	Kolmogorov scale	CFL	
t	u'	ε	R_λ	η	$u'\delta t/\Delta x$	$u_{\max}\delta t/\Delta x$
0	0.6024	0.0770	113.24	0.01795	0.0982	0.4013
2	0.5185	0.0645	91.67	0.01876	0.0845	0.3699

Table I. Statistics of decaying isotropic turbulence in the global domain (256^3). The statistics are the same to within four digits for the five different time steps used, except for the quoted CFL numbers which are based on the case $\delta t = 4 \times 10^{-3}$.

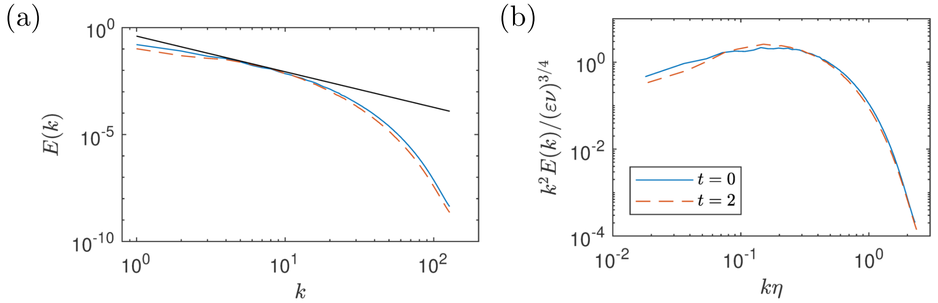

Figure 2. Radial (a) kinetic energy and (b) dissipation spectra at the start of the simulation $t = 0$ and the end of the simulation $t = 2$. The black straight line in (a) has a slope of -5/3.

a transient of a few hundred time steps, the entire velocity and pressure fields are stored and designated as the initial condition ($t = 0$, $n = 0$) of our set of numerical experiments.

The kinematic viscosity is set to $\nu = 2 \times 10^{-3}$ in order to provide appropriate resolution of the viscous scale at the initial time. Five different time steps will be used, $\delta t = \{4, 2, 1, 0.5, 0.25\} \times 10^{-3}$. Simulations are advanced from $t = 0$ to $t = 2$. Some basic statistics of the simulation of this decaying isotropic turbulence are listed in Table I. These were verified to be accurate to within four digits for the various choices of the time step; the reported CFL values are based on the largest $\delta t = 4 \times 10^{-3}$. The kinetic energy and dissipation spectra are shown in Figure 2. The dissipation spectra are displayed in Kolmogorov units, showing that the simulation is very well resolved in space (note that the spatial resolution is much better than than in the JHTDB original data even if using less points since here we simulate a much lower Reynolds number with a much higher ν).

While performing the simulation in the global domain, data are stored at every time step to be used for later analysis and comparison with re-simulation results. For the sample re-simulations and numerical experiments to be described in the next section, a sub-domain consisting of 32^3 grid points is selected (i.e. $M_s = 32$) located at a random location within the global domain. To compare results from re-simulation to the original global domain simulation, a normalized local error is defined according to

$$\epsilon_\varphi(x, y, z, t) = \frac{|\varphi_{os}(x, y, z, t) - \varphi_{rs}(x, y, z, t)|}{\mathrm{rms}(\varphi_{os})}, \tag{8}$$

where φ could be any quantities, such as u, v, w, p or vorticity components ω_i, $\mathrm{rms}(\cdot)$ is the root-mean-square (r.m.s) value within the sub-domain, "*os*" refers to the original simulation, and "*rs*" refers to the re-simulation. We focus on the L^∞ errors evaluated as function of time within the sub-domain, $\epsilon_{\varphi,\infty}(t)$, which is a stringent upper bound on the re-simulation errors.

IV. PRELIMINARY RESULTS

As a first test we consider a re-simulation starting from the initial condition at $t = 0$. One can use the velocity and pressure fields at $n = 0$ as the re-simulation initial condition. The boundary conditions at time step n are $\boldsymbol{u}^{(n)}_{rs,\Gamma} = \boldsymbol{u}^{(n)}_{os,\Gamma}$ for velocity and $\left(\partial\phi^{(n)}_{rs}/\partial\boldsymbol{n}\right)_\Gamma = \left(\partial\phi^{(n)}_{os}/\partial\boldsymbol{n}\right)_\Gamma$ for pressure increment. Above, $\boldsymbol{n}$ denotes the outward pointing normal unit vector to the boundary Γ (distinct from time step n).

Using these initial and boundary conditions, the re-simulation is integrated in time between $t = 0$ all the way to $t = 2$ (i.e. for 500 time-steps for the case $\delta t = 4 \times 10^{-3}$). Figure 3(a) shows a comparison of the pressure distribution on a representative plane and time. While overall the agreement may appear good, there are some noticeable differences especially near the lower left and upper right boundaries.

More quantitatively, the maximum error (L^∞) and r.m.s error over the sub-domain are shown as functions of time in Figures 3(b-d). The error is large already at the first re-simulation time-step and then remains at similar order of magnitude. The L^∞ and r.m.s errors of velocity and pressure are of order 10^{-3}, and vorticity errors are about one order of magnitude higher and could reach near 10%; these errors are too large compared to our stated desired level of accuracy. (We have found the vorticity errors are typically one order of magnitude higher than velocity, so we only show vorticity results towards the end when showing results of acceptable levels of errors.)

An interesting observation is that the errors do not grow exponentially, suggesting that

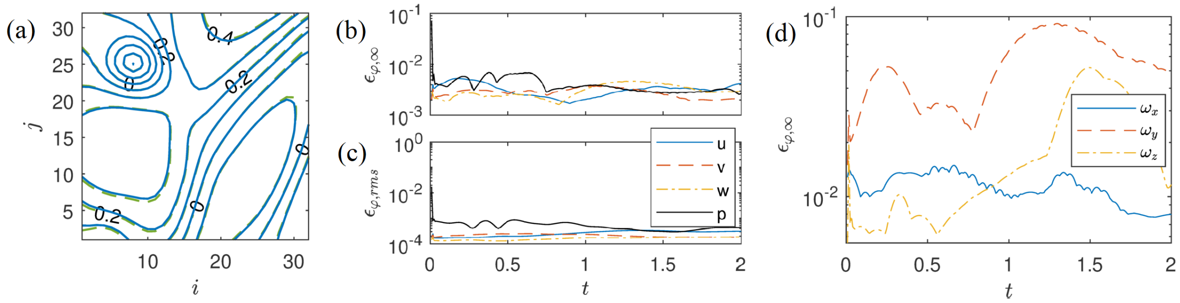

Figure 3. (a) Contour plot of pressure distribution on a randomly selected slice in the 32^3 sub-domain re-simulation at a randomly selected, representative, time step. The dash contour lines are the original simulation, while the solid contour lines are the re-simulation. (b)$\varepsilon_{\varphi,\infty}$ as function of time t. (c) r.m.s error $\varepsilon_{\varphi,rms}$ as function of time t. In the re-simulation, the velocity boundary conditions are $\boldsymbol{u}$ and the pressure boundary condition is Neumann type. (d) L^{∞} vorticity errors as a function of time. All plots are for the case $\delta t = 4 \times 10^{-3}$.

the observed errors are not caused by chaotic dynamics as one may have initially suspected based on the non-linear character of the governing equations. The reason might be that the sub-domain size is relatively small so that even if there are differences between the two fields, the re-simulation dynamics are slaved to the original dynamics by the imposed boundary conditions. Naturally we anticipate that if the sub-domain was large enough, simply providing boundary conditions would not guarantee that the two trajectories would not diverge eventually in time due to chaotic dynamics in the domain interior. Regardless of the origin of the observed errors, we have experimented with a number of parameters such as the time step and spatial resolution, and the basic conclusion remains that the errors are significant and far from the desired accuracy for our database application. Aiming to reduce these errors, we analyze the source of the discrepancy and identify the appropriate choice of implementing initial and boundary conditions in order to greatly reduce these errors.

A. Re-simulation boundary conditions: $\boldsymbol{u}$ versus $\boldsymbol{u}^*$

Consider the re-simulation procedure from the initial condition $n = 0$ to the first time step $n = 1$. At time step $n = 0$, the initial conditions are based on $\boldsymbol{u}_{os}^{(0)}$ and $p_{os}^{(0)}$ of the global computation, and therefore the re-simulation matches that state exactly. Since $\boldsymbol{u}_{rs}^{(0)}$ and $p_{rs}^{(0)}$ match the global simulation, the convective, diffusive and pressure gradient terms

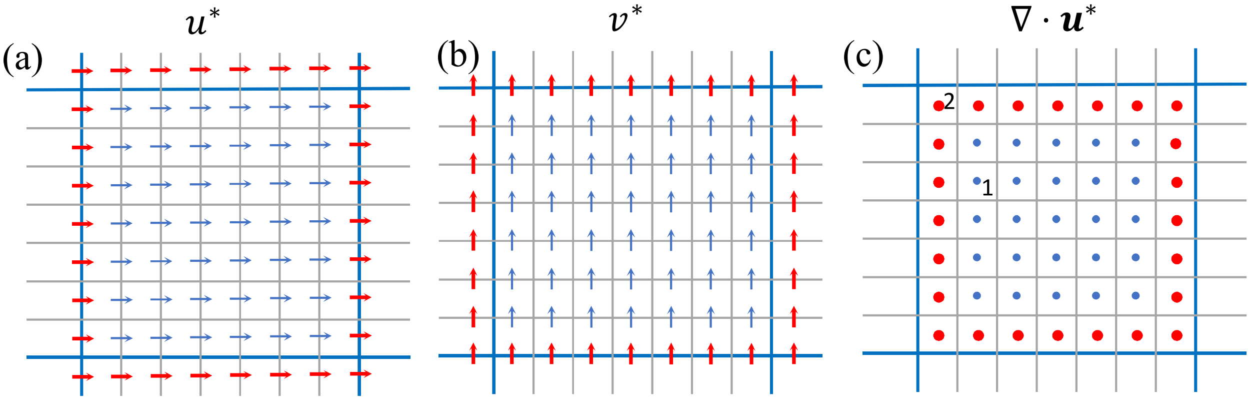

Figure 4. Comparisons of (a) u^*, (b) v^* and (c) $\nabla\cdot\boldsymbol{u}^*$ between the re-simulation and the original simulation. Blue (thin) and red (thick) symbols denote the quantities in re-simulation match/mismatch to the original simulation data.

inside the re-simulation sub-domain are correct. Because the momentum equations are both integrated with an Euler method in the original simulation and the re-simulation to the first time step $n = 1$, $\boldsymbol{u}^*$ *inside* the sub-domain is the same as in the original simulation (blue thin arrows in Figure 4(a,b)). Meanwhile, $\boldsymbol{u}_{os}^{(1)}$ on Γ are applied as the velocity boundary conditions. However, the data on and outside the sub-domain boundary are also $\boldsymbol{u}^*$ in the original simulation, since they lie within the global domain. Thus, the re-simulation does not match the original computation on and outside the sub-domain boundary (red thick arrows in Figure 4(a,b)). The source term of the Poisson equation is then computed, and the comparison with the original simulation is shown in Figure 4(c). Considering two grid points as examples, the source term at point 1 is calculated from surrounding values of $\boldsymbol{u}_{rs}^*$, all of which are identical to the original simulation. Thus the source term is correct (blue small dots). However, at point 2, the values of u^* at left and v^* above are different from the original simulation, thus the source term at this grid point differs from the global solver (red big dots). The Poisson equation with perturbed source term is solved and $\boldsymbol{u}_{rs}$ and ϕ_{rs} therefore contain errors.

The above discussion shows that the choice of velocity boundary conditions leads to errors in the re-simulation outcome, as reported in Figure 3. The remedy is to adopt $\boldsymbol{u}_{os}^*$ as the velocity boundary condition in the re-simulation procedure.

Thus switching procedure, now the values of $\boldsymbol{u}_{os}^*$ at the boundaries of the sub-domains were stored during the global simulation. These were subsequently used for boundary con-

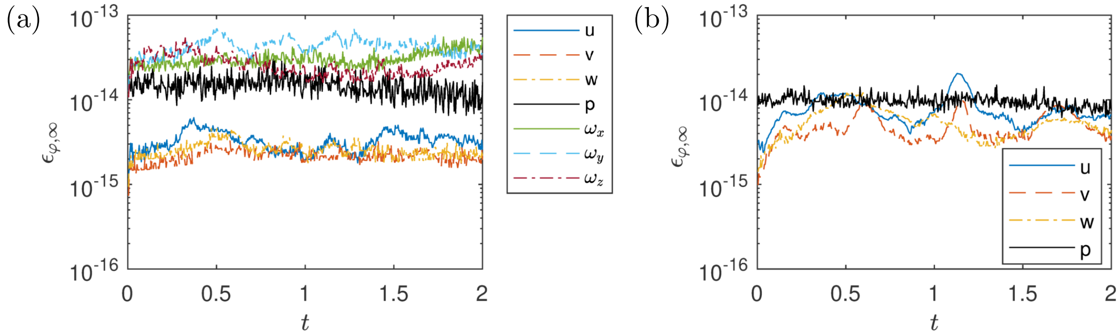

Figure 5. Errors $\varepsilon_{\varphi,\infty}$ as function of time (for the case $\delta t = 4 \times 10^{-3}$). In the re-simulation, the velocity boundary conditions are $\boldsymbol{u}^*$, and the pressure boundary condition is (a) Neumann type and (b) Dirichlet type.

ditions in the local re-simulation procedure. The resulting L^∞ errors are reported in Figure 5(a). Indeed the re-simulation velocities and pressure agree with the global computation results exactly, to within machine precision.

Gresho and Sani[32] and Abdallah and Dreyer[33] showed that Dirichlet and Neumann pressure boundary conditions are equivalent, to within a constant. We confirmed the same behavior for the re-simulations by performing a test with pressure Dirichlet boundary conditions $\phi_{rs} = \phi_{os}$ on the sub-domain boundary Γ. The re-simulation errors, shown in Figure 5(b), are still at machine accuracy, the same as those in the re-simulations with the pressure Neumann boundary conditions (note that in both cases $\boldsymbol{u}^*_{rs} = \boldsymbol{u}^*_{os}$ is enforced on Γ).

B. Crank-Nicolson scheme

In simulations of non-homogeneous flows such as wall-bounded turbulent flows, the viscous term may limit the time step due to the stability restriction. Therefore, this term is often discretized in time using Crank-Nicolson (CN) scheme in order to mitigate the stability restriction. Using CN, equation (4) is approximated with the alternating direction implicit (ADI) method according to,

$$(1 - A_x)(1 - A_y)(1 - A_z)\boldsymbol{u}^* = \delta t[-\text{Conv.} + \frac{1}{2}\nu L(\boldsymbol{u}^{(n-1)}) - G(p^{(n-1)})] + \boldsymbol{u}^{(n-1)}, \qquad (9)$$

where $A_x = \frac{1}{2}\nu\delta t L_x$, $A_y = \frac{1}{2}\nu\delta t L_y$, $A_z = \frac{1}{2}\nu\delta t L_z$, Conv. $= \alpha_c C(\boldsymbol{u}^{(n-1)}) + \beta_c C(\boldsymbol{u}^{(n-2)})$ is the integrated advection term, L is the discretized Laplacian operator, and L_x, L_y and L_z are the discretized Laplacian operators in the x, y and z directions. The procedure for solving

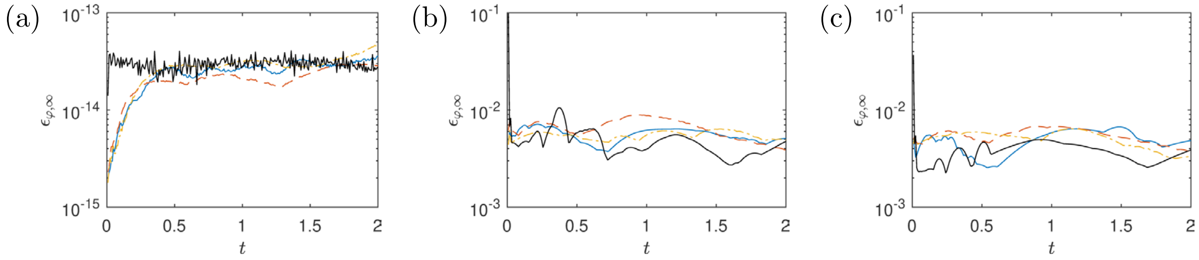

Figure 6. Re-simulations errors with different velocity boundary conditions, which are (a) $\boldsymbol{u}^{*1}$, $\boldsymbol{u}^{*2}$ and $\boldsymbol{u}^{*3}$ in the corresponding directions, (b) $\boldsymbol{u}$ in all directions and (c) $\boldsymbol{u}^{*} = \boldsymbol{u}^{*3}$ in all directions. In all plots, $\Delta t = 4 \times 10^{-3}$. See Figure 5 for legend.

the above equation consists of evaluating $\boldsymbol{u}^{*}$ in each of the three directions successively: (i) solve for $\boldsymbol{u}^{*1}$ in the x direction, where $(1 - A_x)\boldsymbol{u}^{*1}$ =right hand side of equation (9) with x boundary conditions; (ii) solve for $\boldsymbol{u}^{*2}$ in the y direction, where $(1 - A_y)\boldsymbol{u}^{*2} = \boldsymbol{u}^{*1}$ with y boundary conditions; (iii) solve for $\boldsymbol{u}^{*} = \boldsymbol{u}^{*3}$ in the z direction, where $(1 - A_z)\boldsymbol{u}^{*3} = \boldsymbol{u}^{*2}$ with z boundary conditions. In Section IV A, it was demonstrated that $\boldsymbol{u}^{*}$ should be the velocity boundary condition if both the original and re-simulation algorithms are explicit Euler/AB2. When CN/ADI is adopted however, different intermediate velocity boundary conditions are required. Specifically, $\boldsymbol{u}^{*1}$ should be applied on the boundaries during the inversion of the x-diffusion term, $\boldsymbol{u}^{*2}$ should be applied on the boundaries during the solution in the y direction, and $\boldsymbol{u}^{*} = \boldsymbol{u}^{*3}$ should be applied on the boundaries in the final z direction.

We demonstrate this requirement by performing the original/global simulation and the re-simulation using the CN scheme as described above, and compare the results with cases in which some of the specific directional requirements for $\boldsymbol{u}^{*}$ are relaxed. The re-simulation errors with the correct boundary condition implementation are shown in Figure 6(a). The re-simulation errors remain near 10^{-14} for all velocities and pressure. As comparison, the re-simulations with either $\boldsymbol{u}$ or $\boldsymbol{u}^{*} = \boldsymbol{u}^{*3}$ (the last step of the ADI) velocity boundary conditions are also performed. Both produce significant error levels, between 10^{-3} and 10^{-2} (Figure 6(b,c)).

V. ANALYSIS OF DOMINANT SOURCES OF ERRORS

In Section IV A, the correct velocity boundary conditions for re-simulation was shown to be $\boldsymbol{u}^{*}$. It was shown that using $\boldsymbol{u}^{*}$ on the boundaries based on surface data stored at every DNS time step, and replicating the precise time advancement scheme at every time step between the original DNS and the re-simulation, yielded machine-accuracy from re-simulation. However, in practical applications of STSR, one may wish to relax some of these requirements. For example, one may wish to store the boundary values not at every time-step and use moderate sub-sampling (e.g. snapshots of the 1024^3 isotropic turbulence data set in JHTDB are stored only every 10 simulation steps, and temporal polynomial interpolation is used to find data between stored time steps). Or, one may wish to use a different time-advancement scheme during the initial time stepping of the re-simulation. Each of these approaches will induce some additional error and prevent the re-simulation to reach machine precision. In order to establish a clear understanding of these errors, it is useful to quantify the amplification of errors by the re-simulation procedures.

In order to lay the foundation for the subsequent discussions, we intentionally add noise to the boundary condition values $\boldsymbol{u}^*$. We use zero-mean Gaussian white noise and define the contaminated boundary condition on the boundary Γ, for example for the u-component, as

$$u^*_\sigma = u^*(1 + \sigma\mathcal{N}(0,1)), \tag{10}$$

where σ represents the r.m.s. of the added noise as multiple of the original signal. Moreover, $\mathcal{N}(0,1)$ is the standard normal distribution with zero mean and unit variance. Similar noise perturbations are added to the two other components v^* and w^*, and pressure increment ϕ, at all time steps $n > 0$.

Re-simulation experiments are performed for four different levels of σ ($10^{-4} - 10^{-10}$) using $\boldsymbol{u}^*_\sigma$ and $\partial\phi_\sigma/\partial\boldsymbol{n}$ as boundary conditions. The re-simulation errors $\varepsilon_{u,\infty}$ are shown in Figure 7(a) as a function of t with different noise levels σ; only u errors are plotted for clarity. Although the noise levels are different, the errors are qualitatively similar at different values of σ and only differ in magnitude. Figure 7(b) shows the scaling of $\max_t[\varepsilon_{\varphi,\infty}]$ with σ. The results clearly show that re-simulation errors grow linearly with the magnitude of the added noise level in the boundary conditions.

It should be noted that, in the above analysis, the noise is added to the boundary conditions at all time steps after the initial condition, i.e. $n \geqslant 1$, and the re-simulation errors are proportional to the input errors. If the noise is added at the initial condition across the entire re-simulation domain at $n = 0$, similar results are obtained (not shown here).

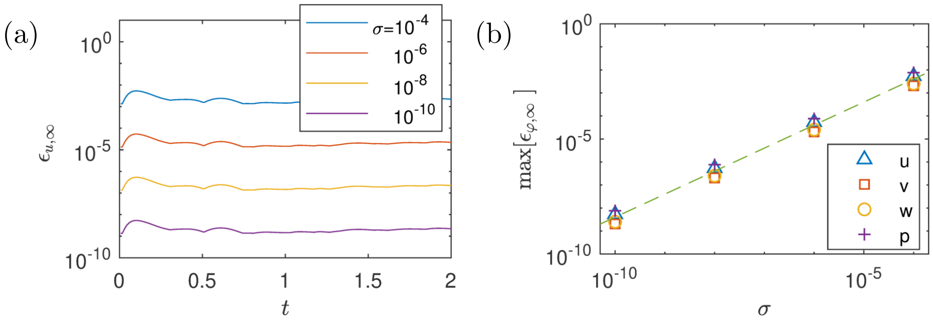

Figure 7. Re-simulations with different levels of noises added to the velocity boundary conditions $\boldsymbol{u}^*$. (a) Re-simulation errors $\varepsilon_{u,\infty}$ against t . Only u errors are plotted for clarity. It has been checked that v, w and p errors behave similarly. (b) $\max[\varepsilon_{\varphi,\infty}(t>0)]$ as function of σ. The dashed line has a slope of 1. In both plots, $\delta t = 4\times 10^{-3}$.

A. Re-examination of $\boldsymbol{u}$ boundary condition errors

We have seen that the re-simulation errors are proportional to the input errors. We now revisit the errors discussed in Section IV A, where we first naively applied $\boldsymbol{u}$ as the velocity boundary conditions, to explain the observed errors based on the findings that errors are linearly proportional to boundary condition errors.

From equation (6), one can easily show that the difference between $\boldsymbol{u}^{(n)}$ and $\boldsymbol{u}^*$ is second order in time,

$$\boldsymbol{u}^{(n)} - \boldsymbol{u}^* = -\delta t \nabla \phi^{(n)} = -\delta t \nabla (p^{(n)} - p^{(n-1)}) \sim -(\delta t)^2\, \nabla (\frac{\partial p}{\partial t}). \tag{11}$$

Based on the results in Figure 7, one would then expect that applying $\boldsymbol{u}$ as boundary conditions in the re-simulation would lead to second order errors in δt. This expectation was tested by performing the global and re-simulations with different values of δt and prescribing $\boldsymbol{u}$ as the velocity boundary condition in the re-simulations. The resulting re-simulation errors are plotted in Figure 8(a,b). Same as in Figure 7(a), $\varepsilon_{\varphi,\infty}$ behave qualitatively similar for different values of δt. The maximum errors, $\max_t[\varepsilon_{\varphi,\infty}]$, are reported in Figure 8(c). Surprisingly, the pressure errors are only first order in δt, while the velocity errors are second order, as expected. In addition, we find that the pressure errors recover second order accuracy at $n>1$ (Figure 8(d)). In fact, figures 8(c) and (d) show that the maximum pressure errors are first order in δt for $n \geqslant 1$, but second order for $n>1$. This observation suggests that the pressure errors are of first order at $n=1$ but second order afterwards. The insert of Figure 8(b) shows the pressure errors near $n=0$, while Figure 9 shows the u

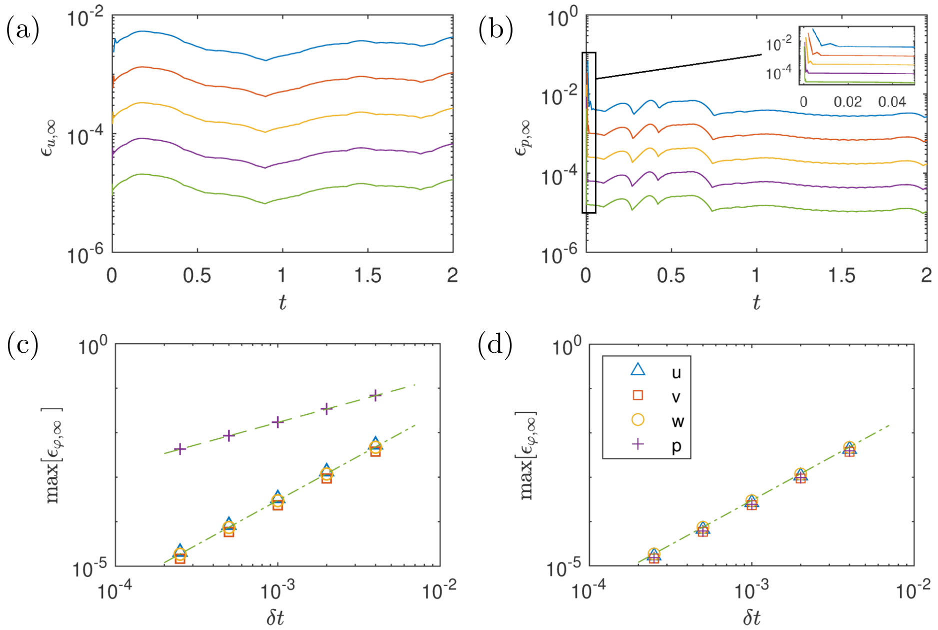

Figure 8. Re-simulations with $\boldsymbol{u}$ as the velocity boundary conditions using different time steps. (a) u errors $\varepsilon_{u,\infty}$ as function of time, t . (b) Pressure errors $\varepsilon_{p,\infty}$ as function of time t. The insert is a zoom near $t = 0$. In (a) and (b), lines from top to bottom represent simulations with $\delta t = 4\times10^{-3}$, 2×10^{-3}, 1×10^{-3}, 5×10^{-4} and 2.5×10^{-4} respectively. (c) $\max[\varepsilon_{\varphi,\infty}(n \geqslant 1)]$ as function of δt. (d) $\max[\varepsilon_{\varphi,\infty}(n > 1)]$ as function of δt. In (c) and (d), the dashed line has a slope of 1 and the dashed-dotted line has a slope of 2.

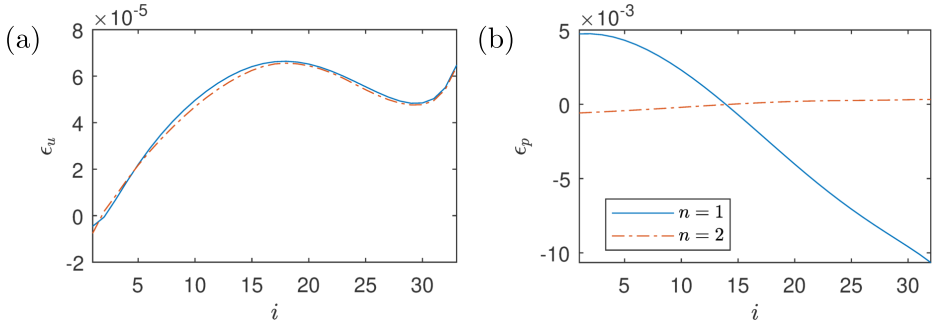

Figure 9. Relative errors of (a) u and (b) p along a line in the center of the sub-domain at the first (solid) and second (dashed) time step in the $\delta t = 4\times10^{-3}$ case.

and p errors along a line in the centre of the sub-domain.

A brief explanation follows: assume the initial field of the re-simulation matches the original global computation. In the first time step, if $\boldsymbol{u}_{os}^{(1)}$ is used as the velocity boundary condition, i.e., $\boldsymbol{u}_\Gamma^* = \boldsymbol{u}_{os}^{(1)}$, sub-domain now contain $\mathcal{O}(\delta t^2)$ errors at the boundaries,

$$\epsilon(\boldsymbol{u}^*) = \begin{cases} \boldsymbol{u}^{(1)} - \boldsymbol{u}^* = (\delta t)^2 \frac{\partial}{\partial x}(\frac{\partial p}{\partial t})|_{n=1} = \delta t^2 \zeta_{n=1} & \text{on the boundaries} \\ 0 & \text{inside the sub-domain} \end{cases}, \tag{12}$$

where $\zeta = \frac{\partial}{\partial x}(\frac{\partial p}{\partial t})$. From the right hand side of equation (5) and Figure 4, the source term of the Poisson equation will therefore have $\mathcal{O}(\delta t)$ errors due to the errors at the sub-domain boundaries,

$$DG\epsilon(\phi^{(1)}) = \frac{D\epsilon(\boldsymbol{u}^*)}{\delta t} = \begin{cases} \delta t^2 \zeta_{n=1}/h\delta t = \delta t \zeta_{n=1}/h & \text{on the boundaries} \\ 0 & \text{inside the sub-domain} \end{cases}. \tag{13}$$

Even though the non-zero source terms only exist at the boundary nodes in equation (13), the errors in ϕ contaminate the entire sub-domain due to the ellipticity of the Poisson operator. Thus ϕ errors, as well as p errors, are $h\delta t \zeta_{n=1} = \mathcal{O}(\delta t)$ at $n = 1$. It is important to note here that $\epsilon(\phi^{(1)})$ is linearly distributed in the sub-domain (can be verified analytically to be a solution of equation (13) , or refer to Figure 9(b)). As a result, the gradient of $\epsilon(\phi^{(1)})$ is uniform in the correction step, leading to a uniform $\delta t^2 \zeta_{n=1}$ error in the velocity within the sub-domain:

$$\epsilon(\boldsymbol{u}^1) = \epsilon(\boldsymbol{u}^*) - \delta t G\epsilon(\phi^1) = \delta t^2 \zeta_{n=1} = \mathcal{O}(\delta t^2). \tag{14}$$

At the second time step $n = 2$, $\boldsymbol{u}^*$ have uniform $\mathcal{O}(\delta t^2)$ errors both inside the sub-domain and on the boundaries: the errors inside the sub-domain, $\delta t^2 \zeta_{n=1}$, come from $\boldsymbol{u}^{(1)}$ (see above), while the errors on the boundaries, $\delta t^2 \zeta_{n=2}$, come from the new velocity boundary conditions. The leading $\mathcal{O}(\delta t^2)$ errors of $\boldsymbol{u}^*$ are cancelled out during the calculation of the divergence of $\boldsymbol{u}^*$,

$$D\epsilon(\boldsymbol{u}^*) = \begin{cases} \delta t^2 \zeta_{n=2} - \delta t^2 \zeta_{n=1} = \delta t^3 \frac{\partial \zeta}{\partial t}|_{n=1} & \text{on the boundaries} \\ \delta t^2 \zeta_{n=1} - \delta t^2 \zeta_{n=1} = \mathcal{O}(\delta t^3) & \text{inside the sub-domain} \end{cases}, \tag{15}$$

leading to second order errors in the source term of the Poisson equation, also in the pressure field at $n = 2$. In addition, the velocity errors remain at second order,

$$\epsilon(\boldsymbol{u}^{(2)}) = \epsilon(\boldsymbol{u}^*) - \delta t G\epsilon(\phi^{(2)}) = \mathcal{O}(\delta t^2) - \delta t \mathcal{O}(\delta t^2) = \mathcal{O}(\delta t^2). \tag{16}$$

The preceding analysis thus demonstrates that the observed errors when using $\boldsymbol{u}$ instead of $\boldsymbol{u}^*$ as boundary conditions for re-simulation scale in expected ways with the size of time-step. If one wanted to use $\boldsymbol{u}$ instead of $\boldsymbol{u}^*$ for re-simulation, however, the required time steps would be too small to be practical for purposes of the STSR.

B. Errors from mismatch in temporal discretization

The above results all assumed that the re-simulation starts from an Euler scheme, same as the original computation which at $n = 0$ also began using an Euler step. This ensures that the re-simulation could calculate the intermediate velocity inside the sub-domain correctly as seen in Figure 4, and reproduce the original simulation data precisely, when using the $\boldsymbol{u}^*_{os}$ boundary conditions.

However, in applications of STSR, the re-simulation will typically start at any of the stored original simulation time steps, i.e. when n equals any integer multiple of $M_t\delta t$. Recall that the original simulation used AB2 time-stepping at those times, not Euler. As a result, for the re-simulation to reproduce the original computation, it must adopt an AB2 scheme from its start. However, this requirement can only be met if two consecutive time steps are stored to be used as initial condition. Otherwise, with a single field, the re-simulation must adopt a first Euler step and will therefore deviate from the original AB2-based computation.

In order to demonstrate the errors incurred by an initial Euler step, we perform the following experiment: The data on the entire domain is stored at $t = 1$, meaning the initial condition for the re-simulation is now $\boldsymbol{u}_{os}$ and p_{os} at $t = 1$. The re-simulation starts there with a single Euler scheme and then continues with AB2.

At the first time step after the initial condition, the Euler scheme will introduce local truncation errors of $\mathcal{O}(\delta t^2)$ into the re-simulation. The re-simulation errors are shown in Figure 10. Similar to the case which uses $\boldsymbol{u}$ as the velocity boundary condition (Section V A), the p errors are first order in δt at the first time step, but second order afterwards. On the other hand, velocity errors are always second order.

In addition, we considered another case to explore errors incurred if the time stepping scheme used in the re-simulation is always different from that in the original one. We performed re-simulation with Euler scheme from $t = 1$ and for all time steps, rather than for the first step only. In this case, the Euler scheme has global errors of $\mathcal{O}(\delta t)$ compared to AB2. The errors are shown in Figure 11. The $\boldsymbol{u}$ errors increase over t. This is due to the cumulative effect of the local truncation errors committed in each step from the Euler scheme. As a result, the velocity errors grow from second order to first order (see Figure 11(c-d)). On the other hand, the p errors are already first order at the first time step, and retain that scaling, consistent with Euler's global truncation errors $\mathcal{O}(\delta t)$.

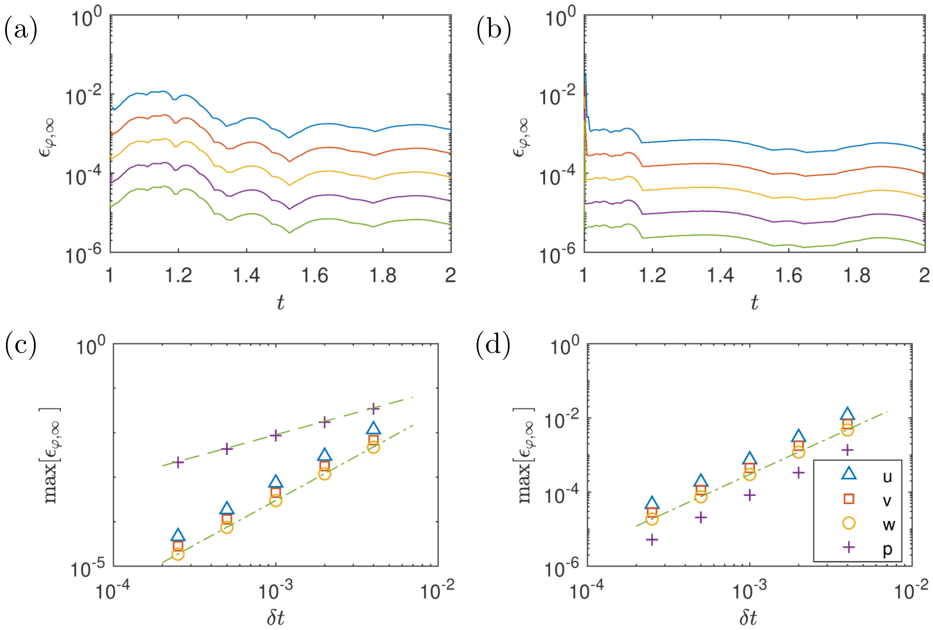

Figure 10. Re-simulation error evolution when using an Euler scheme at the first time step and then continuing with AB2 ($1 \le t \le 2$). The original simulation used the AB2 scheme. (a) u error $\varepsilon_{u,\infty}$ against t. (b) p error $\varepsilon_{p,\infty}$ against t. In (a) and (b), lines from above to bottom represent simulations with $\delta t = \{4, 2, 1, 0.5, 0.25\} \times 10^{-3}$ respectively. (c) $\varepsilon_{\varphi,\infty}$ against δt at the first time step. (d) $\max[\varepsilon_{\varphi,\infty}]$ against δt after the first time step. In (c) and (d), the dashed line has a slope of 1 and the dashed-dotted line has a slope of 2.

C. Errors from temporal sub-stepping

In the previous section, it was shown that the re-simulation has $\mathcal{O}(\delta t^2)$ errors if started with an Euler scheme at an arbitrary time. These errors are too large for reproducing a DNS database using realistic values of δt. For example, when $\delta t = 4 \times 10^{-3}$, even if we discard the results at the first time step, the relative errors between the original and re-simulation are approximately $10^{-3}-10^{-2}$ in subsequent time steps. Using an initial Euler step in the re-simulation compared to AB2 in the original computation results in an initial error that persists in time—consistent to the behaviour when artificial errors were included in the initial conditions. Although one could store an extra snapshot so that the re-simulation starts with AB2 and obtain error-free data, this approach would appreciably increase the storage requirements.

Rather than storing two time steps, we examine a different approach that does not increase the required storage but only increases CPU cost during re-simulation: temporal

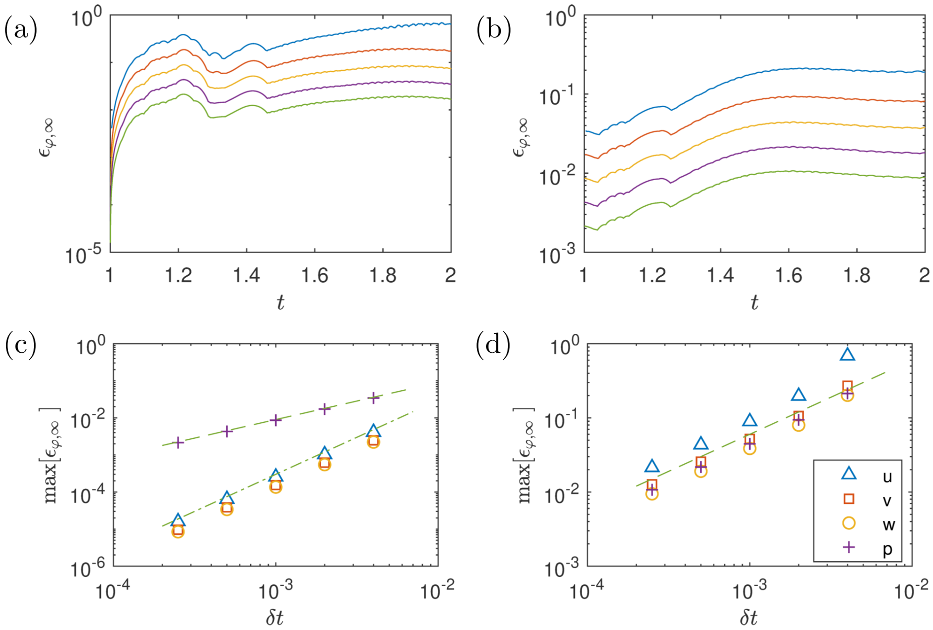

Figure 11. Re-simulations using and Euler time advancement throughout ($1 \leq t \leq 2$). The original simulation always uses the AB2 scheme. (a) u error, $\varepsilon_{u,\infty}$, against t. (b) p error, $\varepsilon_{p,\infty}$, against t. In (a) and (b), lines from above to bottom represent simulations with $\delta t = \{4, 2, 1, 0.5, 0.25\} \times 10^{-3}$ respectively. (c) $\varepsilon_{\varphi,\infty}$ against δt at the first time step. (d) $\max[\varepsilon_{\varphi,\infty}]$ against δt after the first time step. In (c) and (d), the dashed line has a slope of 1 and the dashed-dotted line has a slope of 2.

sub-stepping. This idea aims to minimize the error between the original single AB2 step and many smaller steps the first of which is Euler followed by AB2.

Consider integration from t to $t + \delta t$. The analytic integration could be approximated by an AB2 scheme or an Euler scheme both with a time-step size δt. We have already seen in the previous section that the differences between AB2 and Euler schemes lead to re-simulation errors. Usually, an AB2 scheme produces smaller errors than Euler compared with analytic (true) values. On the other hand, the time step from t to $t + \delta t$ could also be divided into, say, k sub-time steps: the size of each sub-time step is thus $\delta t/k$ (see Figure 12 for an example with $k = 4$). Integration from t to $t+\delta t$ would then be computed using Euler in the first sub-time step, then AB2 in the remaining $(k - 1)$ sub-time steps. The numerical integration results will approach the true value with increasing number of sub-steps k. The single full-time-step Euler integration is the special case with $k = 1$. Thus, one could expect that the errors between the single full-time-step AB2 integration and the integration with temporal sub-stepping would decrease first, then increase, and finally reach an asymptotic

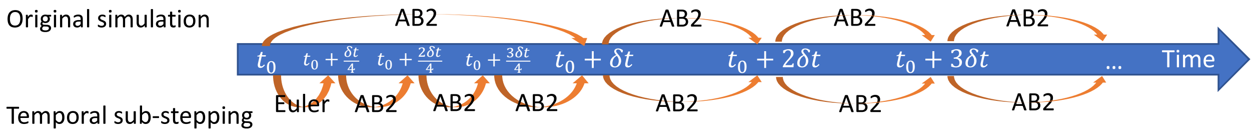

Figure 12. Schematic of temporal sub-stepping with four sub-time steps.

value as the number of time sub-steps k increases: the asymptotic value is the errors of the AB2 scheme itself. Ideally, there will be a k with which the re-simulation errors are minimized, even though this optimized k, if it exists, would be different from one simulation to another.

Beyond $t + \delta t$, the re-simulation can proceed with AB2 using the original time step δt. For example, the solution at $t + 2\delta t$ can be computed from information at t and $t + \delta t$; similarly the solution at $t + 3\delta t$ can use the information at $t + \delta t$ and $t + 2\delta t$ and so on.

The boundary conditions on Γ at the sub-time steps can be approximated from temporal interpolation of $\boldsymbol{u}^*_{os}$ from the original simulation data. For instance, in the example below, the boundary conditions between t and $t + \delta t$ are obtained by applying piecewise cubic Hermite interpolating polynomial (PCHIP) on stored boundary conditions (plane data) at $t - \delta t$, t, $t + \delta t$ and $t + 2\delta t$.

For demonstration, we perform a re-simulation of the original computation with $\delta t = 4 \times 10^{-3}$, starting from $t = 1$ and advancing the simulation until $t = 2$. Re-simulations with different numbers of temporal sub-steps k, as well as the original AB2 scheme, are compared. Just a reminder, $k = 1$ is equivalent to the re-simulation performing the entire first step with Euler scheme. In this example, the results from a re-simulation with $k = 1000$ sub-time steps are used as the reference data to approximate the “true, exact” values which are unknown. We discard the first few δt to avoid including the pressure jump as seen in the previous examples.

Figure 13(a) shows the maximum relative errors compared with the reference data for $1 < t < 2$. The symbols denote the errors between the re-simulation and the reference data, which decrease as k increases. In fact, the errors are proportional to k^{-2}, or the square of the size of the time sub-step $(\delta t/k)^2$, since the temporal scheme is AB2 in the re-simulation. The horizontal lines represent the errors between the original AB2 simulation and the reference data. The errors of the AB2 scheme itself are about $10^{-5} - 10^{-4}$. Also from Figure 13(a), it is clear that the errors between the re-simulations (symbols) and the original DNS (lines)

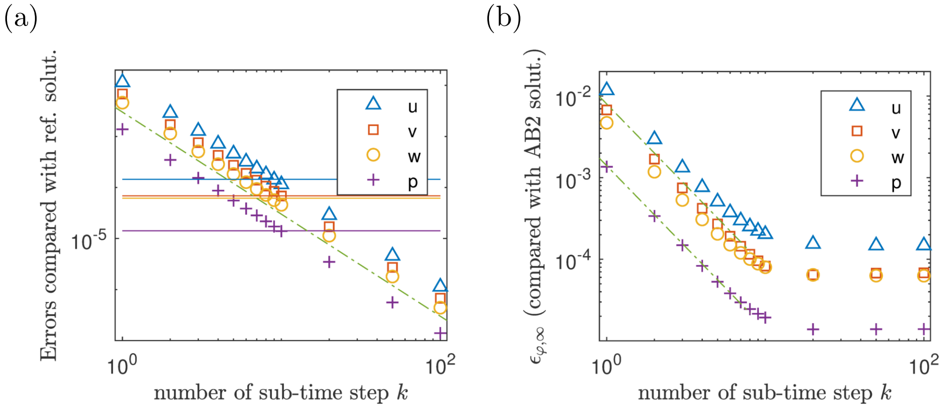

Figure 13. (a) The L^∞ relative errors compared with the reference re-simulation ($k = 1000$). The symbols represent the re-simulations with sub-time steps, while the lines represent the original simulation with the AB2 scheme. The colours of the horizontal lines represent the same variables as the symbols. (b) Re-simulation errors compared with the original simulation data, $\varepsilon_{\varphi,\infty}$. In both plots, the dash-dot line has a slope of 2.

decrease and then increase as k increases. However, it should be noted that the differences between the symbols and lines do not equal to the actual errors between the re-simulations and the original DNS, $\varepsilon_{\varphi,\infty}$.

The re-simulation errors $\varepsilon_{\varphi,\infty}$, shown in Figure 13(b), decrease at a rate of second order in k before about $k = 6$, and then become nearly constant. Although an optimal k is not observed, the drop of the errors is about two orders of magnitude in the current example. The asymptotic values of $\varepsilon_{\varphi,\infty}$ are also the AB2 errors shown in Figure 13(a). This example shows that the re-simulation errors could decrease by two orders of magnitude with only 10 additional time sub-steps within the first δt from the initial condition, and the minimum errors are bound by those of the AB2 integration scheme in the original simulation.

D. Temporal sub-sampling for the boundary conditions

In all previous examples, the re-simulations adopted boundary conditions data that were stored at every time step during the original DNS . This may not be necessary or feasible. As mentioned before, the snapshots of the 1024^3 isotropic turbulence data set in JHTDB are stored only every 10 simulation steps. When data is queried between the two stored time steps, they are obtained with temporal interpolation and the errors are approximately 10^{-6} (we could not determine whether the interpolation errors are lower than 10^{-6}, because the

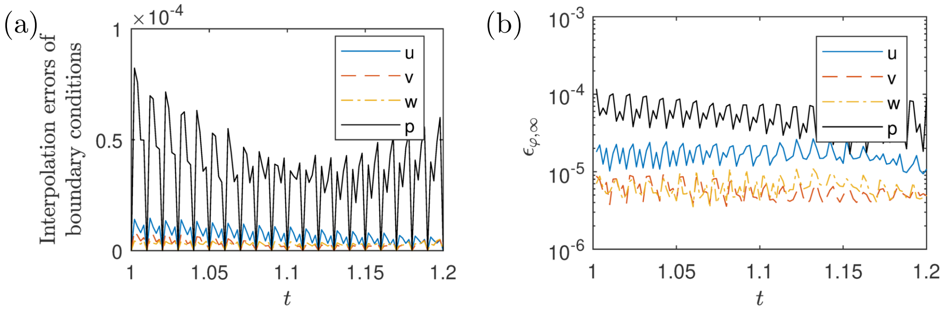

Figure 14. (a) The interpolation errors of boundary conditions. (b) $\epsilon_{\varphi,\infty}$ with interpolated boundary conditions. The re-simulation is from $t = 1$ to 2, but only $t = [1, 1.2]$ is plotted here to more clearly display the oscillations of the errors. The re-simulation starts with the AB2 scheme using an extra snapshot provided. The time step size is $\delta t = 2 \times 10^{-3}$.

data on JHTDB are stored in single precision). Here we examine the impact of temporal interpolation of temporally sub-sampled boundary data for re-simulation.

We have seen that the re-simulation errors are proportional to the errors in the boundary conditions. Thus, if the boundary conditions are stored every few ($M_{t,bc}$) time steps and temporal interpolation is used during re-simulation, the errors in the re-simulation will be directly proportional to the interpolation errors. Figure 14 shows an example: the time step of the simulation is $\delta t = 2 \times 10^{-3}$. The boundary data are stored at every $M_{t,bc} = 5$ time steps, actually close to the time step requirement based on CFL (based on maximum velocity) equaling to unity. Cubic spline interpolation with three points before and after the query point is used for temporal interpolation. The L^∞ relative errors of the interpolated boundary condition fields on the Γ planes are shown in Figure 14(a). The oscillations of the errors are apparent, vanishing at each of the $5\delta t$ time instants in which boundary data are known exactly. The re-simulation starts at $t = 1$ using the AB2 scheme with an extra snapshot provided, and runs until $t = 2$. As a result, no other errors are introduced in the re-simulation, except those due to the temporal interpolation of the boundary conditions. The maximum interpolation errors over time for $\{u, v, w, p\}$ are $\{1.47, 1.54, 1.64, 9.66\}\times 10^{-5}$ (Figure 14(a)). The re-simulation errors $\epsilon_{\varphi,\infty}$ (Figure 14(b)) for $\{u, v, w, p\}$ are $\{2.66, 2.08, 2.30, 24.2\}\times 10^{-5}$: all are only slightly higher than the interpolation errors. The oscillations of the re-simulation errors are caused by the oscillatory errors of the temporal interpolation of the boundary conditions.

VI. SUMMARY: RECOMMENDED CHOICES FOR STSR

The previous section has documented separately errors to be expected from various parameter choices for STSR. Here we now combine the various choices that may be expected in an actual implementation of STSR: we use $\boldsymbol{u}^*$ on the boundaries stored at every $M_{t,bc} = 5$ DNS time steps, use $k = 10$ for the initial temporal sub-sampling during the first time-step of re-simulation, use cubic polynomial temporal interpolation of the stored $\boldsymbol{u}^*$ and p boundary values to interpolate to the re-simulation time-step δt, and integrate between $t = 1$ and $t = 2$.

Figure 15 compares two fields at $t = 2$ from the re-simulation to the original simulation: (a) u-velocity and (b) $z-$component vorticity ω_z (computed using centered finite differencing). The contour lines of re-simulation fields and the original ones are on top of each other.

Figure 15(c) shows the corresponding evolution of the L^∞ errors. The vorticity errors are about one order of magnitude higher than velocity errors and is about 10^{-4}. This level of difference between re-simulation and original DNS is acceptable and falls within the desired guidelines.

VII. CONCLUSIONS

In the present paper, we propose an idea of data compression for numerical simulation results of incompressible fluid flow. The entire simulation domain of the original simulation is divided into multiple small sub-regions by planes. The data in the entire domain are stored, say, at every few hundred or thousand time steps, while data on the dividing planes are stored at every time step, or sub-sampled every few time steps. Once data at an arbitrary position and time is needed, a re-simulation of the small cube region (sub-domain) which includes that point is performed. The data stored in the entire domain are used as the initial condition, while the planar data surrounding the sub-domain are used as the boundary conditions.

It is found that if the numerical scheme in the re-simulation matches the original simulation exactly, the re-simulation will produce error-free results. On the other hand, any mismatch between the re-simulation and the original one can produce significant errors,

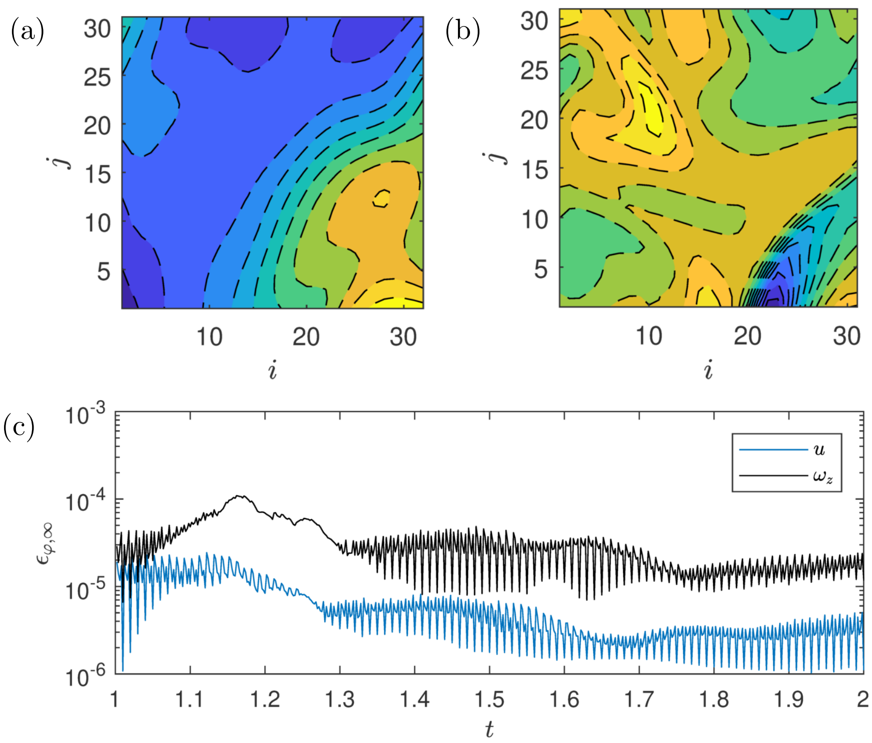

Figure 15. (a) Contour plot of u on a randomly selected slice. (b) Contour plot of z-component vorticity on a randomly selected slice. In (a) and (b), colour contours are the original simulation, while the black dash contour lines are the re-simulation. (c) L^{∞} errors of u and $z-$ component vorticity.

exceeding the minimum error levels one would like to enforce for a database that contains spatially and temporally sub-sampled data.

For example, it was found that re-simulation errors are too high when using velocity and pressure differences (or pressure) for the boundary condition. It was shown that the correct velocity boundary conditions for the re-simulation should be the intermediate velocity after the projection step: this is because the boundaries of the sub-domain are still the internal part of the entire domain of the original simulation.

Another example is that the re-simulation should use the same time integration scheme as the original simulation. This poses a challenge if only one snapshot of the initial field is provided: the re-simulation must start with an Euler scheme while the original simulation has been advanced with an AB2 scheme. The challenge can be resolved by storing an extra snapshots so that the re-simulation could start with the AB2 scheme as well, or could be improved using Euler-AB2 integration with several sub-time steps to approximate the first

AB2 integration in the original simulation. We have shown the latter approach saves storage space, and can also reduce the re-simulation errors by two orders of magnitude with only 10 sub-time steps added in the first original time step.

The findings also imply that if the original simulation contains source terms in the Navier-Stokes equations, such as in forced isotropic flow these source terms must also be recorded together with the original simulation and included in the re-simulation.

Tests using boundary data with added noise show that re-simulation errors remain linearly proportional to the errors in the boundary conditions. This observation helps explain several trends in re-simulation errors. Also, it provides a guideline about how much temporal sub-sampling of the boundary data may be used. The resulting errors in re-simulation will be proportional to the errors caused by temporal interpolation on the boundary data. Experiment shows the re-simulation error is similar to the interpolation errors of the boundary conditions. Thus, in a real application, one could carefully control the interval of two stored plane data and achieve further compression of the simulation data.

A sample application combining all of the recommended sub-sampling parameters and re-simulation strategies shows that relative maximum errors in velocity on the order of 10^{-5} to 10^{-4}, which is acceptable and leads to errors of less that 0.1% in velocity gradients. These levels are acceptable for applications of building numerical turbulence databases like JHTDB. It is worth reiterating that the errors in the numerical experiments performed here did not reveal exponential growth in time, at least not over the tested time horizons. From the viewpoint of data assimilation[19,21,22], synchronization of chaos[20] and nudging[23] of Navier-Stokes turbulence, the present results have implications on how effective the time-evolving boundary conditions are at constraining and effectively synchronizing or nudging the dynamics. In prior work[19] it was shown that providing the correct large-scale Navier-Stokes dynamics at all wavenumbers down to $\sim 0.2k_\eta$ (i.e. corresponding to grid spacings of $\sim 15\eta$) leads to eventual slaving (synchronization) of the smaller scales, while coarser truncations lead to chaotic divergence of trajectories at the small scales (similar results were obtained later in[20]). Here we show something different: that domains of significantly larger size $(30\eta)^3$ can still remain slaved to the dynamics at all scales provided the data at the boundaries contain scales down to the smallest viscous scales (DNS resolution). A more systematic analysis, such as testing how large the re-simulation sub-domain can be made before the boundary information is no longer able to synchronize the dynamics in the core of the sub-domain, is beyond the scope of the present study.

The sub-sampling and local re-simulation technique described in this paper could also be applied on unstructured meshes, as long as the correct information is stored during the original simulation and the resimulation uses exactly the same method as the original simulation. If a spectral method is used in the original simulation (such as in several of the existing JHTDB datasets), using local resimulation with (e.g.) finite differencing will lead to significant errors. If the spectral method is used only in one or two directions, like channel flow, good accuracy can be achieved if the resimulation domain consists of the entire 1D "pencils" or 2D "slabs". However, if the spectral method is used in all three directions, the present technique cannot reproduce error-free data unless the resimulation is done on the entire (large) domain, which is expected to be prohibitive.

We remark that alternative re-simulation methods e.g. based on machine learning tools instead of grid-based CFD methods could be considered. For instance, one could apply Physics Informed Neural Network (PINNs) methods[34] to train an Artificial Neural Network constrained by Navier-Stokes equations to predict field data at desired points and time using similar types of initial and bounding surface data as used in the present method as inputs (see also Ref.[35] for a recent example). The present results documenting errors to be expected from Navier-Stokes based re-simulation can serve as useful reference or benchmark to which to compare such alternative methodologies.

Finally, although this work is focused on turbulence in incompressible flows, extensions of the basic idea and methodological requirements to other fields of computational physics appear possible. Also, other compression tools can be applied on top of the present technique. For example, one can use wavelet methods[36] to further compress the planar and volumetric data.

ACKNOWLEDGMENTS

The authors acknowledge funding from the National Science Foundation (grant # OCE-1633124). Computations were made possible by the Maryland Advanced Research Computing Center (MARCC) and the SciServer platform. Useful discussions and conversations with Profs. A. Szalay, R. Burns, G. Eyink and Dr. C. Lalescu are gratefully appreciated.

APPENDIX A: FAST POISSON SOLVER FOR RE-SIMULATION

In this appendix, details about a spectral fast Poisson solver for equation (5) used in re-simulations are described. Since the re-simulation sub-domain is in general not periodic, a fast Poisson solver using discrete sine and cosine transforms[31] is implemented.

Consider a one-dimension Poisson equation,

$$\nabla^2 \psi = b \tag{17}$$

on a uniform grid $x_i = ih$ $(i = 1, \dots, N)$, where $h = \Delta x$ is the constant grid spacing. The Poisson equation discretized with second-order central finite differences is

$$\frac{\psi_{i-1} - 2\psi_i + \psi_{i+1}}{h^2} = b_i, \qquad i = 1, \dots, m, \tag{18}$$

and can be represented in Fourier space as

$$\lambda_j \hat{\psi}_j = \hat{b}_j, \qquad j = 1, \dots, N, \tag{19}$$

where $\lambda = -k'^2$ is the eigenvalue and k' is the modified wavenumber. Thus, the Poisson equation can be solved in three steps: (i) calculate $\hat{b}_j$ from the forward Fourier/sine/cosine transform of b; (ii) find $\hat{\psi}_j = \hat{b}_j / \lambda_j$ from equation (19); (iii) calculate ψ from the inverse transform of $\hat{\psi}_j$. The transforms used in (i,iii) and the eigenvalues λ_j depend on the boundary conditions and are listed in tables II and III. In table II, "DFT" refers to the discrete Fourier transform, "DST-II" to type-II discrete sine transform, and "DCT-II" to type-II discrete cosine transform. For non-homogeneous boundary conditions, b_1 and b_n can be modified in order to absorb the values at the boundaries.

When $\lambda_1 = 0$, an additional equation is required, e.g. with the periodic or Neumann boundary conditions in all directions one could simply set $\hat{\psi}_1 = 0$ leading to a zero-mean solution. It should also be noted that this algorithm gives the least square solution for the discretized Poisson equation if the compatibility condition $\sum b_i = 0$ is not satisfied.

The discrete Fourier, sine and cosine transforms are included in various libraries, including FFTW and FFTPACK. If a DST-II or DCT-II is not implemented, e.g. in the Intel Math Kernel Library (MKL), it can be computed via a DCT-III combined with $\mathcal{O}(2n)$ pre- and post-processing.

Extension of the algorithm to 3D is straightforward: (i) calculate $\hat{b}_{j_1 j_2 j_3}$ from the forward transform of b; (ii) find $\hat{\psi}_{j_1 j_2 j_3} = \hat{b}_{j_1 j_2 j_3} / \lambda_{j_1 j_2 j_3}$, where $\lambda_{j_1 j_2 j_3} = \lambda_{j_1} + \lambda_{j_2} + \lambda_{j_3}$; (iii) calculate ψ from the backward transform of $\hat{\psi}_{j_1 j_2 j_3}$.

Boundary conditions	Forward	Backward
Periodic $(x_0 = x_m,\ x_{m+1} = x_1)$	DFT	Inverse of DFT
Dirichlet on cell faces $(x_1 + x_0 = 0,\ x_{m+1} + x_m = 0)$	DST-II	Inverse of DST-II
Neumann on cell faces $(x_1 - x_0 = 0,\ x_{m+1} - x_m = 0)$	DCT-II	Inverse of DCT-II

Table II. The transforms used in steps 1 and 3 in the fast Poisson solver.

Boundary conditions	Eigenvalues
Periodic $(x_0 = x_n,\ x_{m+1} = x_1)$	$\lambda_k = -\frac{4}{h^2}\sin^2\frac{(k-1)\pi}{m}$
Dirichlet on cell faces $(x_1 + x_0 = 0,\ x_{m+1} + x_m = 0)$	$\lambda_k = -\frac{4}{h^2}\sin^2\frac{k\pi}{2m}$
Neumann on cell faces $(x_1 - x_0 = 0,\ x_{m+1} - x_m = 0)$	$\lambda_k = -\frac{4}{h^2}\sin^2\frac{(k-1)\pi}{2m}$

Table III. The eigenvalues used in step 2 in the fast Poisson solver.

If the grid is non-uniform in only one direction, e.g. in channel or boundary-layer flows, the spectral approach is adopted in all dimensions where the grid is uniform, and a tri-diagonal solver is adopted in the direction of grid stretching (see Moin[37], Section 6.2.1 for an example). In fact, solving a tri-diagonal linear system is faster than Fourier transforms, since the former has a computational cost $\mathcal{O}(N)$, which is less than that of fast Fourier transform, $\mathcal{O}(N \log N)$.

The current fast Poisson solver is faster in time and saves the memory compared with a Poisson solver implementing sparse matrix solver. Table IV compares the time spent in solving the discrete Poisson equation using sparse matrix LU decomposition, FFT and DST/DCT. When the gird comprises 128^3 points, the LU decomposition requires extensive memory and in our tests using limited resources (as one would like to use during re-simulation), it runs out of memory. The solution using DST/DCT requires approximately twice the time of the DFT, and only one-dimensional DST/DCT are available in the major-

Grid points	LU decomposition	DFT	DST/DCT
32^3	0.0082 s	$< 10^{-3}$ s	$< 10^{-3}$ s
48^3	0.0357 s	$< 10^{-3}$ s	0.0016 s
64^3	0.1137 s	0.0019 s	0.0035 s
96^3	0.5451 s	0.0052 s	0.0101 s
128^3	-	0.0109 s	0.0234 s

Table IV. Time spent in solving the discrete Poisson equation with a sparse matrix solver, FFT or DST/DCT. The timing has a resolution of 10^{-3} s, and is averaged over 100 runs. In the LU decomposition method, only the solution phase (i.e. forward and backward substitutions after the LU decomposition) is timed. The hardware is Intel Core i5-7500 (4 Cores, 3.4GHz) and 16GB memory. The code uses Intel Fortran compiler, Intel MKL and OpenMP in Windows. The parallelization of the sparse matrix solver and the DFT is implemented in Intel MKL, while that of DST/DCT is implemented by authors using OpenMP. In the 128^3 case, the LU decomposition runs out of memory.

ity of numerical libraries. Nevertheless, DST/DCT outperforms the direct solver based on the sparse matrix LU decomposition, and its scalability is superior.

REFERENCES

[1] J. Kim, P. Moin, and R. Moser, "Turbulence statistics in fully developed channel flow at low Reynolds number," Journal of Fluid Mechanics **177**, 133–166 (1987).

[2] P. Moin and K. Mahesh, "DIRECT NUMERICAL SIMULATION: A Tool in Turbulence Research," Annual Review of Fluid Mechanics **30**, 539–578 (1998).

[3] D. Livescu and J. R. Ristorcelli, "Variable-density mixing in buoyancy-driven turbulence," Journal of Fluid Mechanics **605**, 145–180 (2008).

[4] P. K. Yeung, D. A. Donzis, and K. R. Sreenivasan, "Dissipation, enstrophy and pressure statistics in turbulence simulations at high Reynolds numbers," Journal of Fluid Mechanics **700**, 5–15 (2012).

[5] I. Bermejo-Moreno, J. Bodart, J. Larsson, B. M. Barney, J. W. Nichols, and S. Jones, "Solving the compressible Navier-Stokes equations on up to 1.97 million cores and 4.1 trillion grid points," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13* (ACM Press, New York, New York, USA, 2013) pp. 1–10.

[6] P. K. Yeung, X. M. Zhai, and K. R. Sreenivasan, "Extreme events in computational turbulence," Proceedings of the National Academy of Sciences **112**, 12633–12638 (2015).

[7] M. Lee and R. D. Moser, "Direct numerical simulation of turbulent channel flow up to Reτ=5200," Journal of Fluid Mechanics **774**, 395–415 (2015), arXiv:1410.7809.

[8] J. Lee and T. A. Zaki, "Detection algorithm for turbulent interfaces and large-scale structures in intermittent flows," Computers & Fluids **175**, 142–158 (2018).

[9] Y. Yamamoto and Y. Tsuji, "Numerical evidence of logarithmic regions in channel flow at R eτ=8000," Physical Review Fluids **3**, 012602 (2018).

[10] J. You and T. A. Zaki, "Conditional statistics and flow structures in turbulent boundary layers buffeted by free-stream disturbances," Journal of Fluid Mechanics **866**, 526–566 (2019).

[11] J. C. del Alamo and J. Jimenez, "Spectra of the very large anisotropic scales in turbulent channels," Physics of Fluids **15**, L41 (2003).

[12] E. Perlman, R. Burns, Y. Li, and C. Meneveau, "Data exploration of turbulence simulations using a database cluster," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing - SC '07* (ACM Press, New York, New York, USA, 2007) p. 1.

[13] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink, "A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence," Journal of Turbulence **9**, N31 (2008), arXiv:0804.1703.

[14] H. Yu, K. Kanov, E. Perlman, J. Graham, E. Frederix, R. Burns, A. Szalay, G. Eyink, and C. Meneveau, "Studying Lagrangian dynamics of turbulence using on-demand fluid particle tracking in a public turbulence database," Journal of Turbulence **13**, N12 (2012).

[15] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," Proceedings of the IRE **40**, 1098–1101 (1952).

[16] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," IEEE Transactions on Information Theory **23**, 337–343 (1977).

[17] ISO., "ISO/IEC 11172-3:1993 - Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio," (1993).

[18] ISO., "ISO/IEC 10918-1:1994 - Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines," (1994).

[19] K. Yoshida, J. Yamaguchi, and Y. Kaneda, "Regeneration of small eddies by data assimilation in turbulence," Physical review letters **94**, 014501 (2005).

[20] C. C. Lalescu, C. Meneveau, and G. L. Eyink, "Synchronization of chaos in fully developed turbulence," Physical review letters **110**, 084102 (2013).

[21] C. Foias, C. F. Mondaini, and E. S. Titi, "A discrete data assimilation scheme for the solutions of the two-dimensional navier–stokes equations and their statistics," SIAM Journal on Applied Dynamical Systems **15**, 2109–2142 (2016).

[22] J. L. Callaham, K. Maeda, and S. L. Brunton, "Robust flow reconstruction from limited measurements via sparse representation," Physical Review Fluids **4**, 103907 (2019).

[23] P. C. Di Leoni, A. Mazzino, and L. Biferale, "Synchronization to big data: Nudging the navier-stokes equations for data assimilation of turbulent flows," Physical Review X **10**, 011023 (2020).

[24] A. Kravchenko and P. Moin, "On the Effect of Numerical Errors in Large Eddy Simulations of Turbulent Flows," Journal of Computational Physics **131**, 310–322 (1997).

[25] J. van Kan, "A Second-Order Accurate Pressure-Correction Scheme for Viscous Incompressible Flow," SIAM Journal on Scientific and Statistical Computing **7**, 870–891 (1986).

[26] J. B. Bell, P. Colella, and H. M. Glaz, "A second-order projection method for the incompressible navier-stokes equations," Journal of Computational Physics **85**, 257–283 (1989).

[27] L. Nicolaou, S. Jung, and T. Zaki, "A robust direct-forcing immersed boundary method with enhanced stability for moving body problems in curvilinear coordinates," Computers & Fluids **119**, 101–114 (2015).

[28] A. J. Chorin, "Numerical solution of the Navier-Stokes equations," Mathematics of Computation **22**, 745–745 (1968).

[29] J. Kim and P. Moin, "Application of a fractional-step method to incompressible Navier-Stokes equations," Journal of Computational Physics **59**, 308–323 (1985).

[30] F. H. Harlow and J. E. Welch, "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface," Physics of Fluids **8**, 2182 (1965).

[31] U. Schumann and R. A. Sweet, "Fast Fourier transforms for direct solution of poisson's equation with staggered boundary conditions," Journal of Computational Physics **75**, 123–137 (1988).

[32] P. M. Gresho and R. L. Sani, "On pressure boundary conditions for the incompressible Navier-Stokes equations," International Journal for Numerical Methods in Fluids **7**, 1111–1145 (1987).

[33] S. Abdallah and J. Dreyer, "Dirichlet and Neumann boundary conditions for the pressure poisson equation of incompressible flow," International Journal for Numerical Methods in Fluids **8**, 1029–1036 (1988).

[34] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," Journal of Computational Physics **378**, 686–707 (2019).

[35] K. Fukami, K. Fukagata, and K. Taira, "Super-resolution reconstruction of turbulent flows with machine learning," Journal of Fluid Mechanics **870**, 106–120 (2019).

[36] K. Schneider and O. V. Vasilyev, "Wavelet Methods in Computational Fluid Dynamics," Annual Review of Fluid Mechanics **42**, 473–503 (2010).

[37] P. Moin, *Fundamentals of Engineering Numerical Analysis*, 2nd ed. (Cambridge University Press, 2010) p. 256.