

Computational Physics

Randomized tensor-train truncation on GPUs for quantum-inspired computational fluid dynamics

Kiet Tuan Pham^a, Gordon Euhyun Moon^b, Jinsung Kim^{a,*}^a Chung-Ang University, 84 Heukseok-Ro, Dongjak-Gu, Seoul, 06974, Republic of Korea^b Sogang University, 35 Baekbeom-Ro, Mapo-Gu, Seoul, 04107, Republic of Korea

ARTICLE INFO

The review of this paper was arranged by Prof. Andrew Hazel

Keywords:

Tensor trains (TT)
Quantum-inspired computational fluid dynamics (QIS-CFD)
Navier–Stokes
TT Truncation
Randomized SVD
GPU Computing

ABSTRACT

Tensor-network computational fluid dynamics solvers, often termed quantum-inspired CFD (QIS-CFD), evolve the incompressible Navier–Stokes equations in Tensor-Train (TT/MPS) form by repeatedly constructing nonlinear operators via tensor contractions and then recompressing intermediate tensors to control the bond dimension. In standard TT workflows, recompression is performed with deterministic truncated singular value decompositions (SVDs); for bond dimension χ and fixed local mode size, the per-bond $O(\chi^3)$ cost and limited GPU throughput of SVD routines make truncation a primary bottleneck in GPU-resident time stepping at moderate-to-large χ . We introduce a GPU-accelerated randomized TT truncation scheme (RTTD) based on randomized subspace iteration. RTTD replaces each deterministic truncated SVD with an oversampled randomized range finder and thin QR factorizations on low-dimensional projected matrices, shifting the truncation workload toward GEMM- and QR-dominated kernels while preserving the canonical constraints required by TT/MPS sweeps. We validate RTTD against direct numerical simulation (DNS) and an SVD-based baseline on two 2D benchmarks: a decaying turbulent jet ($Re = 10^5$ for short-run sweeps and $Re = 2 \times 10^5$ for long-horizon validation) and the Taylor–Green vortex ($Re = 10^3$). For $\chi \geq 32$, RTTD matches the SVD baseline and DNS in kinetic-energy evolution and divergence diagnostics. Computationally, RTTD accelerates the nonlinear Matrix-Product-Operator (MPO) construction phase by up to 7.8× and reduces the end-to-end wall-clock time per step by up to 2.2× relative to the deterministic baseline.

1. Introduction

Direct Numerical Simulation (DNS) of the incompressible Navier–Stokes equations provides the gold standard for analyzing turbulent flows, resolving all relevant scales from the integral length scale ℓ down to the Kolmogorov dissipative scale η [1,2]. However, the computational cost of DNS is prohibitive for high Reynolds numbers, as classical scaling estimates suggest the ratio of scales grows as $\ell/\eta \sim Re^{3/4}$ [3,4]. This necessitates rapidly increasing grid resolution and smaller time steps, making long-time integration for converged statistics extremely expensive [5–7]. Even with modern GPU acceleration and highly optimized solvers, large-scale DNS and finely resolved large eddy simulations (LES) remain resource-intensive for engineering-relevant regimes. Recent computational campaigns illustrate these demands across diverse regimes, from isotropic turbulence to wall-bounded and open-channel flows [8–11], and analyses of grid/time-step requirements further highlight the scaling burden [12]. Moreover, state-of-the-art GPU-

accelerated DNS/LES solvers and databases still require substantial resources despite significant speedups [13–17].

These limitations have motivated model reduction and compression strategies that exploit structure in the solution. Tensor-network methods, often termed quantum-inspired CFD (QIS-CFD), represent flow fields in formats originally developed for quantum many-body systems [18,19]. Specifically, the Tensor-Train (TT) decomposition, also known as the Matrix-Product State (MPS), can compactly encode high-dimensional data when correlations decay sufficiently fast along a suitable ordering of degrees of freedom [20,21]. While originating in quantum many-body physics, the TT/MPS format also admits a reduced-order modeling interpretation: it can be viewed as a structured generalization of projection-based ROMs (e.g., POD/Galerkin) that captures multiway correlations through low-rank tensor factors rather than a single global basis [22–24]. In QIS-CFD, this ordering is typically achieved via a Quantics (QTT) mapping [25–27], where grid-based fields are reshaped into higher-order tensors with small physical dimension (e.g., 2–4) that

* Corresponding author.

E-mail addresses: kietpt2099@gmail.com (K.T. Pham), ehmoon@sogang.ac.kr (G.E. Moon), kimsung@cau.ac.kr (J. Kim).

<https://doi.org/10.1016/j.cpc.2026.110259>

Received 22 January 2026; Received in revised form 18 May 2026; Accepted 8 June 2026

Available online 10 June 2026

0010-4655/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

are amenable to TT/MPS compression (Section 2.3). Foundational work by Gourianov et al. [28] demonstrated that local correlations in turbulence exhibit structures analogous to quantum entanglement, enabling velocity fields to be compressed significantly while retaining high variance at moderate TT bond dimension χ .

Building on these observations, recent studies have implemented end-to-end QIS-CFD solvers where the state evolves directly in the compressed TT/MPS format. Hölscher et al. [29] demonstrated a GPU-accelerated pipeline via NVIDIA cuQuantum that reproduced key turbulence diagnostics and reduced memory footprints, and related studies further explored end-to-end QIS-CFD workflows in the TT/MPS format [30,31]. Complementary work has developed high-performance GPU TT primitives (e.g., tensor-core accelerated kernels) for TT/MPS operations [32]. Beyond QIS-CFD, TT and QTT have been incorporated into broader PDE solvers and discretizations, and surveys provide a unified perspective on low-rank tensor methods for PDEs [33,34]. Representative applications include parabolic and elliptic PDE solvers and discretizations in TT/QTT form [35–37], as well as compressible-flow and hyperbolic discretizations such as TT-WENO [38]. A common algorithmic dependency across these TT/QTT pipelines is repeated TT/MPO truncation (rounding/recompression), which motivates kernel-level acceleration of truncation routines.

Despite the massive parallelism of modern GPUs, current QIS-CFD implementations suffer from a distinct performance bottleneck during time-dependent nonlinear operator (MPO) construction. Nonlinear time stepping repeatedly constructs and applies matrix-product operators (MPOs) to the TT/MPS state. These contractions increase intermediate bond dimensions, necessitating repeated rank truncation to enforce a prescribed maximum χ . In standard cuQuantum-based implementations, this per-bond truncation is performed using deterministic SVD-based routines. Although robust, the SVD scales as $O(\chi^3)$ and, in the moderate-to-large χ regime relevant to QIS-CFD, typically underutilizes the GPU relative to GEMM/QR kernels due to higher synchronization and memory-traffic demands. In contrast, the surrounding contractions are dominated by high-throughput GEMM-like operations. Profiling the open-source 2D QIS baseline of Hölscher et al. [29] on an NVIDIA RTX 4090, we find that this nonlinear MPO construction region accounts for approximately 75% of the total wall-clock time in typical regimes ($Re = 10^5$, $n = 10$, $\chi = 32$), as shown in Table B.1. Consequently, accelerating this truncation-heavy region is a high-leverage path to practical QIS-CFD.

To address this bottleneck, we propose a GPU-accelerated randomized TT truncation scheme (RTTD) designed for canonical TT/MPS workflows. We formulate RTTD as a randomized truncation primitive, based on randomized subspace iteration, that replaces the deterministic per-bond SVD with an oversampled randomized range finder and subsequent QR-based factorizations on projected matrices [39,40]. This approach shifts the dominant truncation workload from synchronization-heavy SVD routines to high-throughput GEMM/QR kernels that better match GPU execution characteristics. Crucially, our implementation outputs TT cores in canonical form compatible with the baseline pipeline, making RTTD a drop-in replacement for deterministic truncation in existing QIS workflows.

We evaluate RTTD on two benchmarks: an anisotropic turbulent decaying jet (DJ) at $Re \in \{10^5, 2 \times 10^5\}$ and a Taylor–Green vortex (TGV) at $Re = 10^3$. Our results show that RTTD maintains agreement with the deterministic baseline and DNS in the reported diagnostics, including fidelity/overlap (defined in Section 4), kinetic-energy evolution, and incompressibility measures. Performance profiling reveals that RTTD accelerates the nonlinear MPO construction region by 7.8 $\times$ and reduces total per-step wall-clock time by 2.2 $\times$ for typical settings. The main contributions of this work are:

1. **Bottleneck characterization for GPU QIS-CFD.** We identify that nonlinear MPO construction is dominated by SVD-based truncation due to the high cost of repeated per-bond SVDs in GPU-resident operator construction.

2. **Randomized TT truncation for canonical workflows.** We develop RTTD by combining randomized subspace iteration with QR/RQ factorizations, producing canonical TT cores compatible with standard QIS solvers and left- and right-orthonormal (partition) output requirements.
3. **GPU implementation.** We implement RTTD in C++/CUDA using cuBLAS and cuSOLVER, exposing it via a pybind11/CuPy interface as a drop-in replacement for cuQuantum’s `contract_decompose`.
4. **Validation on turbulence benchmarks.** On the 2D DJ and TGV benchmarks, we demonstrate that RTTD matches baseline diagnostics while delivering substantial end-to-end speedups.

The remainder of this paper is organized as follows: Section 2 presents the governing equations, the QTT/TT representation, MPO operators, and the deterministic SVD-based truncation primitive used for rank control; Section 3 describes the baseline QIS-CFD time-stepping loop and introduces RTTD as a drop-in replacement for per-bond truncation, including GPU implementation details and integration points; Section 4 presents the accuracy and performance results; and Section 5 summarizes limitations and future directions.

2. Preliminaries: TT/MPS, QTT mapping, and baseline truncation

2.1. Tensor trains, canonical form, and deterministic truncation

A tensor-train (TT) representation expresses a d -way tensor as a product of three-index cores with contracted bond indices [21,41]. Given $\mathcal{X} \in \mathbb{R}^{n_1 \times \dots \times n_d}$, the TT form is

$$\mathcal{X}(i_1, \dots, i_d) = \sum_{\alpha_0, \dots, \alpha_d} G_1(\alpha_0, i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \dots G_d(\alpha_{d-1}, i_d, \alpha_d), \quad (1)$$

where $\alpha_0 = \alpha_d = 1$ and $\alpha_k \in \{1, \dots, r_k\}$ are bond indices. Each core $G_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ is a 3-tensor with index ordering (left rank, physical, right rank). The TT ranks $(r_1, \dots, r_{d-1})$ determine compression and the cost of TT operations. We define the bond dimension as $\chi := \max_{1 \leq k \leq d-1} r_k$.

Canonical form. TT/MPS algorithms typically enforce orthogonality (canonicalization) for numerical stability and well-posed truncation [18, 19]. A core G_k is left-canonical if, after reshaping

$$\mathbf{G}_k^{(L)} \in \mathbb{R}^{(r_{k-1} n_k) \times r_k}, \quad \mathbf{G}_k^{(L)}[(\alpha_{k-1}, i_k), \alpha_k] = G_k(\alpha_{k-1}, i_k, \alpha_k), \quad (2)$$

it satisfies $(\mathbf{G}_k^{(L)})^\top \mathbf{G}_k^{(L)} = \mathbf{I}_{r_k}$. Similarly, G_k is right-canonical if, after reshaping

$$\mathbf{G}_k^{(R)} \in \mathbb{R}^{r_{k-1} \times (n_k r_k)}, \quad \mathbf{G}_k^{(R)}[\alpha_{k-1}, (i_k, \alpha_k)] = G_k(\alpha_{k-1}, i_k, \alpha_k), \quad (3)$$

it satisfies $\mathbf{G}_k^{(R)} (\mathbf{G}_k^{(R)})^\top = \mathbf{I}_{r_{k-1}}$. A TT is in mixed-canonical form when cores to the left of a chosen bond are left-canonical and cores to the right are right-canonical; equivalently, an orthogonality center is placed at a chosen site or bond.

Deterministic per-bond truncation (baseline). Per-bond truncation (TT rounding/recompression) is required after contractions (e.g., TT-MPO products) to control rank growth. At bond k , one merges adjacent cores (or an intermediate contraction tensor) and reshapes to a matrix

$$A \in \mathbb{R}^{M \times K}, \quad M = r_{k-1} n_k, \quad K = n_{k+1} r_{k+1}, \quad (4)$$

up to the specific contraction pattern. The baseline truncation enforces a prescribed maximum bond dimension χ by choosing

$$r = \min(\chi, \min(M, K)), \quad (5)$$

and computing the rank- r truncated SVD. Here, r denotes the truncated rank at the active bond (distinct from the TT ranks r_k). We use this hard rank cap (without tolerance-based adaptation) to match the baseline solver configuration and isolate truncation-kernel effects.

$$A = U \Sigma V^\top, \quad A_r := U_r \Sigma_r V_r^\top. \quad (6)$$

The approximation A_r is optimal in the Frobenius norm, i.e., it minimizes $\|A - A_r\|_F$ over all rank- r matrices. The factors are then reshaped back into updated TT cores in a standard TT-rounding sweep, restoring mixed-canonical form. Operationally, the truncation returns an isometric factor on one side of the split and transfers the remaining gauge (scaling) to the neighboring core; this is the interface that our randomized routine preserves. In NVIDIA cuQuantum, this “contract + truncate” pattern is exposed through `contract_decompose`, whose truncation step is SVD-based and typically delegates to GPU SVD routines (e.g., cuSOLVER).

2.2. Incompressible fluid dynamics

We consider the nondimensional two-dimensional incompressible Navier–Stokes equations:

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \frac{1}{\text{Re}} \nabla^2 \mathbf{u} - \nabla p, \quad (7)$$

$$\nabla \cdot \mathbf{u} = 0. \quad (8)$$

Here, $\mathbf{u} = (u_1, u_2)$ is the velocity field, p is the pressure, and $\text{Re} = UL/\nu$ is the Reynolds number. We take a periodic square domain of side length $L = 1$ with periodic boundary conditions (endpoints identified).

2.3. QTT/MPS Representation and MPO operators

We discretize each velocity component on a uniform $N \times N$ grid with $N = 2^n$, where n is the quantization depth, and periodic boundary conditions on a square domain of side length 1. Following Hölcher et al. [29], the grid spacing is defined as

$$\Delta x = \frac{1}{N-1}. \quad (9)$$

(We note that the standard periodic convention uses $\Delta x = 1/N$; the two definitions differ by $\mathcal{O}(N^{-2})$, and we retain the former here to ensure exact numerical comparability with the baseline study by Hölcher et al. [29].)

Writing the binary expansions of grid indices $x_1 = (x_1^1 \dots x_1^n)_2$ and $x_2 = (x_2^1 \dots x_2^n)_2$ with bits $x_j^k \in \{0, 1\}$, where x_j^1 denotes the most significant bit and x_j^n the least significant bit, we form the quantics index

$$\omega_k = (x_1^k x_2^k)_2 \in \{0, 1, 2, 3\}, \quad k = 1, \dots, n, \quad (10)$$

and reshape the grid tensor into the order- n tensor $U_i^{\omega_1 \dots \omega_n}$ [25,26,42]. We approximate $U_i^{\omega_1 \dots \omega_n}$ in MPS/TT form,

$$U_i^{\omega_1 \dots \omega_n} \approx \sum_{\alpha_1, \dots, \alpha_{n-1}} U_{i, \alpha_1}^{(1)}(\omega_1) U_{i, \alpha_1 \alpha_2}^{(2)}(\omega_2) \dots U_{i, \alpha_{n-1}}^{(n)}(\omega_n), \quad (11)$$

where α_k are bond indices and χ denotes the maximum bond dimension. We store u_1 and u_2 as two separate MPS/TT states using the same QTT mapping (10)–(11).

Although the physical benchmark problems considered in this work are two-dimensional, the computational state is not treated as a single dense matrix. Through the QTT mapping, each $N \times N$ field is reshaped into an order- n tensor with small local mode size and then represented in TT/MPS form. Thus, the matrix unfolding used in the truncation step is a local representation of one active bond, while the surrounding solver workflow remains TT/QTT-structured even in the present 2D setting.

Finite-difference operators as MPOs. Spatial operators are represented as matrix-product operators (MPOs) acting on the QTT/MPS state. For example, a central finite-difference operator in x_1 yields

$$\frac{\partial U_i^{x_1 x_2}}{\partial x_1} \approx \frac{U_i^{(x_1+1)x_2} - U_i^{(x_1-1)x_2}}{2\Delta x}, \quad (12)$$

where indices are taken modulo N due to periodicity. Using the same construction, we build MPOs for ∂_{x_2} and the Laplacian ∇^2 (and derived operators such as divergence/gradient).

Nonlinear term and recompression. The convection term requires products such as $u_j \partial_{x_j} u_i$. After MPO application, fields are multiplied pointwise using a copy-tensor construction, which increases the effective bond dimension (often up to χ^2). The result is recompressed back to the target χ by per-bond TT/MPO truncation (Section 2.1).

2.4. Randomized low-rank approximation building blocks

Randomized algorithms approximate the range of a matrix using random test vectors and then factorize a much smaller projected problem [39]. Given $A \in \mathbb{R}^{M \times K}$ and target rank r , one draws a Gaussian test matrix $\Omega \in \mathbb{R}^{K \times (r+p)}$ and forms

$$Y = A\Omega \in \mathbb{R}^{M \times (r+p)}, \quad (13)$$

where p is an oversampling parameter. Orthonormalizing Y yields $Q \in \mathbb{R}^{M \times (r+p)}$, and one forms the projected matrix

$$B = Q^T A \in \mathbb{R}^{(r+p) \times K}. \quad (14)$$

When singular values decay slowly, subspace iteration improves accuracy by alternating multiplications with A and A^T , with intermediate orthonormalization after each multiplication to maintain numerical stability.

2.5. GPU Execution model for TT/MPO linear algebra

On GPUs, the dominant kernels in TT/MPO workflows are dense linear algebra operations (general matrix multiplication, GEMM) and factorizations (QR/SVD) on moderate-sized matrices, together with data movement and reshaping. In cuQuantum-based implementations, contractions are typically expressed in terms of high-throughput GEMM-like kernels, whereas SVD-based truncation introduces synchronization-heavy factorizations whose performance and memory footprint can become limiting at moderate-to-large χ . This motivates truncation schemes that emphasize GEMM/QR-dominated building blocks and avoid repeated SVD-based truncations on large bond matrices when the benchmark accuracy can be maintained (Section 4).

3. Methods

3.1. Baseline QIS–CFD time stepping

We represent the discrete velocity components on an $N \times N$ periodic grid using the QTT/MPS encoding described in Section 2.3. We denote by $|u_i^s\rangle$ the MPS state representing the i -th velocity component ($i \in \{1, 2\}$) at physical time t_s . Spatial operators (finite-difference derivatives, Laplacian, and auxiliary contractions for the nonlinear term) are represented as MPOs and applied via tensor-network contractions as summarized in Section 2.3.

Time integration from t_s to t_{s+1} employs a fourth-order Runge–Kutta (RK4) scheme with $\Delta t = t_{s+1} - t_s$. The update for each velocity component i is given by

$$|u_i^{s+1}\rangle = |u_i^s\rangle + \frac{\Delta t}{6} (|k_{i,1}\rangle + 2|k_{i,2}\rangle + 2|k_{i,3}\rangle + |k_{i,4}\rangle), \quad i \in \{1, 2\}, \quad (15)$$

where $|k_{i,1}\rangle, \dots, |k_{i,4}\rangle$ denote the RK4 stage increments. While (15) defines the global update, the stage increments are computed sequentially by interpreting the scheme as four Euler-like substeps. Each substep computes an intermediate velocity by a variational minimization of the form

$$(|u_i^*\rangle, |u_j^*\rangle) = \arg \min_{|v_1\rangle, |v_2\rangle} \Theta(|v_1\rangle, |v_2\rangle, |a_1\rangle, |a_2\rangle, |b_1\rangle, |b_2\rangle, h), \quad (16)$$

where $|u_i^*\rangle$ denotes an intermediate velocity update in the RK4 procedure, $|v_1\rangle$ and $|v_2\rangle$ are trial MPS states, $(|a_1\rangle, |a_2\rangle)$ are reference states from the previous RK stage, $(|b_1\rangle, |b_2\rangle)$ are the states used to evaluate the nonlinear term, and h is the stage time step. The functional is

$$\Theta(|v_1\rangle, |v_2\rangle, |a_1\rangle, |a_2\rangle, |b_1\rangle, |b_2\rangle, h) = \mu \left\| \hat{D}_1 |v_1\rangle + \hat{D}_2 |v_2\rangle \right\|^2 + \sum_{i=1}^2 \left\| \frac{|v_i\rangle - |a_i\rangle}{h} + |B_i(|b_1\rangle, |b_2\rangle)\rangle \right\|^2, \quad (17)$$

with

$$|B_i(|b_1\rangle, |b_2\rangle)\rangle = \sum_{j=1}^2 \frac{1}{2} (|b_j\rangle \odot \hat{D}_j |b_i\rangle + \hat{D}_j (|b_j\rangle \odot |b_i\rangle)) - \frac{1}{\text{Re}} \hat{D}_j^2 |b_i\rangle, \quad i \in \{1, 2\}. \quad (18)$$

Incompressibility is enforced by the quadratic penalty term with parameter μ , fixed to $\mu = 25 \times 10^4$ in all experiments for fair comparison across truncation kernels. The RK4 substeps correspond to different stage-dependent choices of $(|a_i\rangle, |b_i\rangle, h)$ following Hölischer et al. [29].

The minimization in (16) is carried out via a DMRG-like sweep that updates one MPS core at a time while keeping the remaining cores fixed. Each local update involves solving a linear system, followed by shifting the canonical center along the chain. Concretely, each site update solves the coupled normal equations for the mixed-canonical center cores C_i (i.e., the active three-index TT/MPS tensor at the canonical center) obtained from $\partial\Theta/\partial C_i = 0$, using matrix-free conjugate gradients with tensor-network environment contractions. After each shift, the MPS is recompressed to a maximum bond dimension χ to control rank growth. In the baseline implementation, these recompression steps utilize cuQuantum's `contract_decompose` API in deterministic SVD truncation mode, i.e., deterministic truncated SVD-based TT/MPS truncation at each active bond. All DMRG stopping tolerances and sweep budgets (Table 1) are held fixed across all comparisons; the only modification introduced in this work is the per-bond truncation kernel used inside these contract-decompose-style recompression steps.

A TT/QTT projection formulation based on a pressure Poisson solve is a valid alternative to the present penalty-based variational formulation. Its main advantage is that incompressibility is enforced more directly at the discrete level by projecting the intermediate velocity onto a divergence-free space, subject to the accuracy of the Poisson solve, discretization, and tensor truncation. In the penalty formulation used here, incompressibility is instead enforced approximately through the quadratic penalty term in the variational objective. The advantage of the penalty route in the present context is that it matches the established QIS-CFD workflow inherited from Hölischer et al. [29], where the pressure variable is removed from the optimization problem and the update is performed within the variational sweep. By contrast, a projection-based TT/QTT formulation introduces an additional elliptic

subproblem, namely the pressure Poisson solve, inside each time step or stage; existing tensor-network CFD work reports this Poisson stage as the dominant computational cost in that solver family [31]. We therefore retain the baseline penalty formulation so that the effect of replacing deterministic SVD truncation with RTTD can be assessed without simultaneously changing the incompressibility solver.

Per-stage workflow and profiled regions. Fig. 1 summarizes the solver workflow and defines the high-level code regions used for runtime attribution (Table B.1). Starting from the current velocity MPSs, we (i) precompute environment tensors for time-independent linear operators, (ii) construct state-dependent nonlinear MPOs by contracting operator/state networks and recompressing intermediate bonds back to the target bond dimension χ , (iii) build the corresponding convection-diffusion environments for the current stage operator, and (iv) solve the variational RK stage by DMRG-like sweeps over the trial MPS tensors. Each sweep alternates between assembling the local linear system, performing CG updates, shifting the canonical center with recompression, and updating the environments, until the stage residual meets the stopping criterion or a maximum sweep/iteration budget is reached.

3.2. Randomized TT truncation (RTTD)

During QIS-CFD time stepping, intermediate TT/MPS tensors produced by MPO applications and Hadamard products may exceed the prescribed maximum bond dimension χ and must be recompressed. This recompression is performed repeatedly inside the RK4/DMRG sweeps of the baseline solver (Section 3.1), where cuTensorNet applies an SVD-based truncation at each active bond. We replace *only* this truncation primitive by a randomized subspace-iteration method applied to the single unfolding matrix associated with the active bond, while keeping the surrounding QIS-CFD workflow unchanged.

3.2.1. Single-bond truncation as a matrix approximation

At the local algorithmic level, RTTD is a randomized low-rank approximation of the unfolding matrix associated with one active TT bond. However, the method is not used here as an isolated matrix approximation procedure, but as a truncation primitive inside a canonical TT/MPS workflow, where the factors must be reshaped back into TT cores and must preserve the partition-dependent orthogonality structure required by the sweep.

Consider an active bond that separates indices (α_{k-1}, ω_k) from $(\omega_{k+1}, \alpha_{k+1})$, where ω_k denotes the local physical multi-index at site k

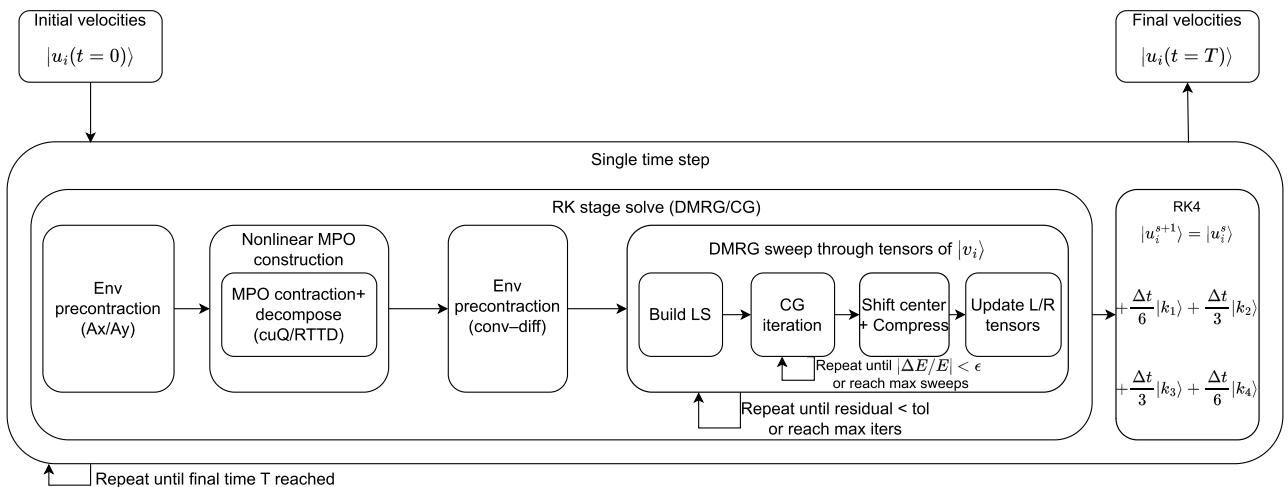


Fig. 1. Schematic overview of the QIS-CFD algorithm and the profiled regions used for the runtime breakdown (Table B.1). RTTD targets the *MPO contraction + decompose* operations inside nonlinear MPO construction, where RTTD replaces deterministic SVD-based truncation while preserving the canonical constraints required by the DMRG sweep.

(with $|\omega_k| = 4$ for 2D-QTT MPS states; for MPO cores we typically use a fused input/output physical index so $|\omega_k| = 16$) and $\alpha_{k-1}, \alpha_{k+1}$ are bond indices. The baseline truncation forms a local tensor $\mathcal{X}$ to be split across this bond and reshapes it into a matrix

$$A = \text{reshape}(\mathcal{X}, [M, K]), \quad M = r_{k-1} |\omega_k|, \quad K = |\omega_{k+1}| r_{k+1}. \quad (19)$$

The truncation problem is to compute a rank- r approximation with

$$r = \min(\chi, \min(M, K)), \quad A \approx UV^T, \quad (20)$$

where the `partition` option determines whether the left factor (U) or right factor (V^T) is returned orthonormal, matching `cuTensorNet`'s `contract_decompose` semantics.

This fixed-rank choice is primarily a controlled benchmarking decision. In the present work, our goal is to isolate the effect of replacing the deterministic SVD-based truncation kernel with RTTD while keeping the surrounding PDE model, discretization, and RK4/DMRG workflow unchanged. For this reason, we match the baseline solver configuration and use a hard rank cap rather than tolerance-based adaptive ranks.

In practical TT/QTT solvers, adaptive rank selection is also important because it balances accuracy and cost according to the local complexity of the solution. In principle, RTTD can be extended to such settings by selecting an effective rank from the projected low-dimensional problem instead of imposing a fixed cap χ in advance. However, unlike deterministic SVD truncation, RTTD does not directly provide SVD-optimal rank- r error control, so a tolerance-based RTTD variant would require an additional rank-selection or residual-based stopping criterion. The present paper therefore restricts attention to the fixed-rank setting used for the controlled comparison of truncation kernels.

3.2.2. Randomized subspace iteration

We apply a randomized subspace-iteration (power-iteration) range finder to the unfolding matrix A in (19). The method is summarized in [Algorithm 1](#). The oversampling parameter p and the number of power iterations q control the quality–cost trade-off of the randomized truncation.

First, p enlarges the sketch dimension from the target rank r to $\ell = r + p$. Its role is to provide a safety margin in the randomized range finder: with a larger sketch, the sampled subspace is more likely to capture the dominant singular directions of A . Thus, increasing p mainly improves reliability, at the price of a modest increase in the size of the projected problem and hence in runtime and memory use.

Second, q controls subspace iteration. In effect, the range finder is applied not only to A , but to a spectrally filtered form such as $(AA^T)^q A$. This suppresses trailing singular directions relative to dominant ones, so increasing q is most useful when the singular values decay slowly or when the rank- r spectral gap is weak. However, each increment of q requires additional passes over A and A^T , so its cost impact is more substantial than that of a modest increase in p .

Accordingly, we treat p and q differently in practice: p is chosen as a small positive oversampling parameter, typically in the range 5–20, while q is kept as small as possible. A practical default is therefore to use modest oversampling together with $q = 1$: the oversampling provides a safety margin for the sketch, while one power iteration often provides useful spectral refinement at relatively low extra cost. Larger q is reserved for cases where $q = 1$ remains insufficient.

Maintaining TT/MPS canonical form requires preserving the appropriate left- or right-isometry condition while transferring the orthogonality center across the active bond. We therefore apply a QR-based gauge transformation according to the selected orthogonality side, following our U/V side convention. Unlike the deterministic truncated-SVD baseline, which provides the Frobenius-norm optimal rank- r approximation of the active unfolding, the present RTTD variant does not provide SVD-optimal rank- r error control. We therefore evaluate RTTD in a fixed-rank setting, where both methods use the same prescribed bond-dimension cap χ . In this setting, RTTD approximation quality is

Algorithm 1: RTTD: Randomized TT Truncation.

Input: $A \in \mathbb{R}^{M \times K}$, max rank χ , oversampling p , power iterations q , `partition` $\in \{ 'U', 'V' \}$
Output: $U \in \mathbb{R}^{M \times r}$, $V^T \in \mathbb{R}^{r \times K}$ such that $A \approx UV^T$

```

1  $r \leftarrow \min(\chi, \min(M, K));$ 
2  $\ell \leftarrow \min(r + p, \min(M, K));$ 
3 if partition == 'U' then
    // Return right-orthonormal factor  $V^T$  (row-orthonormal)
4   Generate Gaussian  $\Omega_U \in \mathbb{R}^{K \times \ell};$ 
5    $Y_U \leftarrow A\Omega_U;$ 
6   Compute thin QR:  $Y_U = Q_U R_U;$ 
7   for  $j \leftarrow 1$  to  $q$  do
8      $Z_U \leftarrow A^T Q_U;$ 
9     Compute thin QR:  $Z_U = \hat{Q}_U \hat{R}_U;$ 
10     $Y_U \leftarrow A\hat{Q}_U;$ 
11    Compute thin QR:  $Y_U = Q_U R_U;$ 
12  end
13   $Q_{U,r} \leftarrow Q_U(:, 1:r);$ 
14   $B \leftarrow Q_{U,r}^T A \in \mathbb{R}^{r \times K};$ 
15  Compute thin QR:  $B^T = \tilde{V}_U R_B;$ 
16   $U \leftarrow Q_{U,r} R_B^T;$ 
17   $V^T \leftarrow \tilde{V}_U^T;$ 
18 end
19 else
    // partition == 'V': return left-orthonormal factor  $U$  (column-orthonormal)
20  Generate Gaussian  $\Omega_V \in \mathbb{R}^{M \times \ell};$ 
21   $Y_V \leftarrow A^T \Omega_V;$ 
22  Compute thin QR:  $Y_V = Q_V R_V;$ 
23  for  $j \leftarrow 1$  to  $q$  do
24     $Z_V \leftarrow A Q_V;$ 
25    Compute thin QR:  $Z_V = \hat{Q}_V \hat{R}_V;$ 
26     $Y_V \leftarrow A^T \hat{Q}_V;$ 
27    Compute thin QR:  $Y_V = Q_V R_V;$ 
28  end
29   $Q_{V,r} \leftarrow Q_V(:, 1:r);$ 
30   $L \leftarrow A Q_{V,r} \in \mathbb{R}^{M \times r};$ 
31  Compute thin QR:  $L = Q_L R_L;$ 
32   $U \leftarrow Q_L;$ 
33   $V^T \leftarrow R_L Q_{V,r}^T;$ 
34 end
35 return  $U, V^T;$ 

```

controlled indirectly through the rank cap and the quality of the randomized subspace approximation, which is influenced by the oversampling parameter p and the power-iteration count q . Accordingly, the present study compares deterministic and randomized truncation under the same fixed-rank budget, rather than claiming SVD-style tolerance control for RTTD.

3.2.3. Canonical form and `partition` semantics

[Algorithm 1](#) enforces orthonormality on the factor dictated by `partition`. Here, `partition` specifies which output factor is returned as an isometry (QR gauge), rather than explicitly partitioning singular values as in an SVD-based decompose.

For `partition`='V', we form $L = A Q_{V,r}$ and compute a thin QR factorization $L = Q_L R_L$; we set $U = Q_L$ so that $U^T U = I_r$, while the scaling is carried by R_L through $V^T = R_L Q_{V,r}^T$ and absorbed into the neighboring core.

For partition='U', we compute $B = Q_{U,r}^T A$ and perform a thin QR of B^T to obtain a right-orthonormal factor V^T with $V^T(V^T)^T = I_r$, while the triangular factor is absorbed into $U = Q_{U,r} R_B^T$ as a gauge transformation.

3.2.4. Computational cost and GPU realization

With $\ell = \min(r + p, \min(M, K))$, RTTD is dominated by GEMMs and thin QR factorizations on moderate-sized matrices. In our GPU implementation, all steps in Algorithm 1 are executed on GPU in double precision: cuRAND generates Ω_U or Ω_V , cuBLAS forms matrix products (e.g., $A\Omega_U$, $A^T\Omega_V$, A^TQ_U , $A\hat{Q}_U$, AQ_V , $A^T\hat{Q}_V$, and projections such as $Q_{U,r}^T A$ or $AQ_{V,r}$), and cuSOLVER computes thin QR factorizations. In the current implementation we use $p = 5$ and $q = 1$ unless stated otherwise, and fix the RNG seed for reproducibility.

3.2.5. Integration into the baseline solver

RTTD replaces only the SVD-based truncation step inside `contract_decompose`-style “contract + truncate” operations during nonlinear MPO construction and canonical-center shifts. Inputs/outputs (target bond dimension χ , partition choice, and canonical constraints) match the baseline API so RTTD can be used as a drop-in replacement. The PDE model, discretization, and RK4/DMRG workflow remain unchanged.

We additionally provide isolated kernel micro-benchmarks comparing RTTD against existing GPU tensor libraries in Appendix A.

4. Results: accuracy and performance

The experiments in this section validate the proposed RTTD truncation on the two-dimensional decaying jet (DJ) benchmark at $Re \in \{10^5, 2 \times 10^5\}$ and the Taylor–Green vortex (TGV) at $Re = 10^3$. All runs are performed on a single NVIDIA GeForce RTX 4090 GPU (24 GB device memory), paired with an AMD Ryzen 9 7950X (16-core) and 64 GB of host RAM. The software environment comprises CUDA 12.6, cuBLAS 12, cuSOLVER 11, cuQuantum 24.08, CuPy 13.6, and Python 3.9.

We compare RTTD against two baselines following Hölscher et al. [29]: (i) the quantum-inspired CFD solver (QIS), which differs from RTTD only in that it employs deterministic SVD-based truncation in the decompose step; and (ii) a direct numerical simulation (DNS) using finite differences for spatial derivatives and Fast Fourier Transforms for the periodic divergence-free projection. Unless stated otherwise, DNS, QIS, and RTTD are run on the same $N \times N$ grid for the reported diagnostics. Common discretization and solver settings shared across all runs are summarized in Table 1. Across the reported cases, RTTD yields diagnostics that closely track the deterministic QIS baseline and agree with DNS / analytical references up to the resolution and rank limits of the underlying solver.

4.1. Diagnostics

All diagnostics are computed on the uniform $N \times N$ grid with spacing $\Delta x = 1/(N - 1)$. For grid functions $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{2 \times N \times N}$, define

$$\langle \mathbf{a}, \mathbf{b} \rangle_{\Delta x} = \Delta x^2 \sum_{i,j} \mathbf{a}_{ij} \cdot \mathbf{b}_{ij}, \quad \|\mathbf{a}\|_{\Delta x}^2 = \langle \mathbf{a}, \mathbf{a} \rangle_{\Delta x}. \quad (21)$$

The kinetic energy is:

$$E_M(t) = \frac{1}{2} \|\mathbf{u}^M(t)\|_{\Delta x}^2, \quad M \in \{\text{DNS, QIS, RTTD}\}. \quad (22)$$

We monitor numerical stability and incompressibility via the maximum and mean discrete divergence:

$$D_M^{\max}(t) = \|\nabla_{\Delta x} \cdot \mathbf{u}^M(\cdot, t)\|_{\infty}, \quad D_M^{\text{mean}}(t) = \frac{1}{N^2} \sum_{i,j} |\nabla_{\Delta x} \cdot \mathbf{u}_{ij}^M(t)|. \quad (23)$$

4.2. Verification: 2D Taylor–Green vortex

We use the 2D Taylor–Green vortex (TGV) as a benchmark to verify the accuracy, stability, and computational efficiency of the proposed RTTD scheme. We fix the Reynolds number $Re = 10^3$, $n = 10$, and compare bond dimensions $\chi \in \{32, 64\}$.

The initial velocity field is given by:

$$\begin{aligned} u_1(x, y, 0) &= u_0 \sin(2\pi x) \cos(2\pi y), \\ u_2(x, y, 0) &= -u_0 \cos(2\pi x) \sin(2\pi y), \end{aligned} \quad (24)$$

with $u_0 = 1$, which corresponds to the exact analytical decay of kinetic energy:

$$E_{\text{ana}}(t) = E(0) \exp(-16\pi^2 \nu t). \quad (25)$$

where $\nu = 1/Re$ is the nondimensional kinematic viscosity.

We assess the solver performance using four quantitative metrics. Energy $E(t)$ and the divergence measure $D^{\max}(t)$ are defined in Section 4.1. To measure accuracy against the analytical solution $\mathbf{u}_{\text{ana}}(t)$, we define the relative discrete L_2 error

$$\varepsilon(t) = \frac{\|\mathbf{u}(t) - \mathbf{u}_{\text{ana}}(t)\|_{\Delta x}}{\|\mathbf{u}_{\text{ana}}(t)\|_{\Delta x}}. \quad (26)$$

We quantify the computational efficiency via the wall-clock speedup factor S :

$$S = \frac{T_{\text{baseline}}}{T_{\text{RTTD}}}, \quad (27)$$

where T represents the average wall-clock time per time step.

Fig. 2 presents a quantitative comparison between the proposed RTTD method and the deterministic QIS baseline.

As shown in Fig. 2(a), the relative error of RTTD is visually indistinguishable from the deterministic baseline at the plot resolution for both $\chi = 32$ and $\chi = 64$. The error increases over time due to accumulated discretization and truncation effects but remains bounded at $\mathcal{O}(10^{-2})$ over the shown window, indicating accuracy comparable to the SVD baseline for this benchmark and the tested bond dimensions. In Fig. 2(b), the kinetic energy decay follows the analytical reference initially; deviations at later times are comparable for RTTD and the baseline and are consistent with numerical dissipation at fixed χ and grid resolution. Fig. 2(c) shows that the divergence diagnostic remains bounded (on the order of 10^{-1}) and comparable to the baseline throughout the run.

Fig. 2(d) plots the energy decay against cumulative wall-clock time. The RTTD curves exhibit a significantly steeper slope compared to the baseline, indicating a faster time-to-solution. Quantitatively, the RTTD scheme achieves a speedup of $\approx 2.15\times$ for $\chi = 32$ and $\approx 2.67\times$ for $\chi = 64$. This performance gain is attributed to the reduced complexity of the randomized range finder compared to the deterministic SVD, particularly as the bond dimension increases.

To further assess whether randomized truncation produces visually apparent artifacts, Fig. 3 shows the RTTD flow field and corresponding error fields at $t = 0.3$. A common concern with randomized range finding is that the sampling step may introduce small-scale, noise-like perturbations or disrupt flow symmetries. Panel (a) shows that the expected Taylor–Green vortex symmetry is qualitatively preserved, with no obvious symmetry-breaking patterns or high-frequency jitter at the

Table 1
Solver and parameters settings used in all experiments.

Parameter	Value
Grid exponent	n
Grid size	$N = 2^n$
Domain length	$L = 1$
Grid spacing	$\Delta x = \frac{1}{N-1}$
Time step (RK4)	$\Delta t = \frac{0.1}{2^{n-1}}$
Penalty weight	$\mu = 2.5 \times 10^5$
Stage solver	Conjugate gradient (CG)
Max CG iterations	100
CG tolerance	10^{-5}
Max sweeps per time step	20
Arithmetic	Double precision

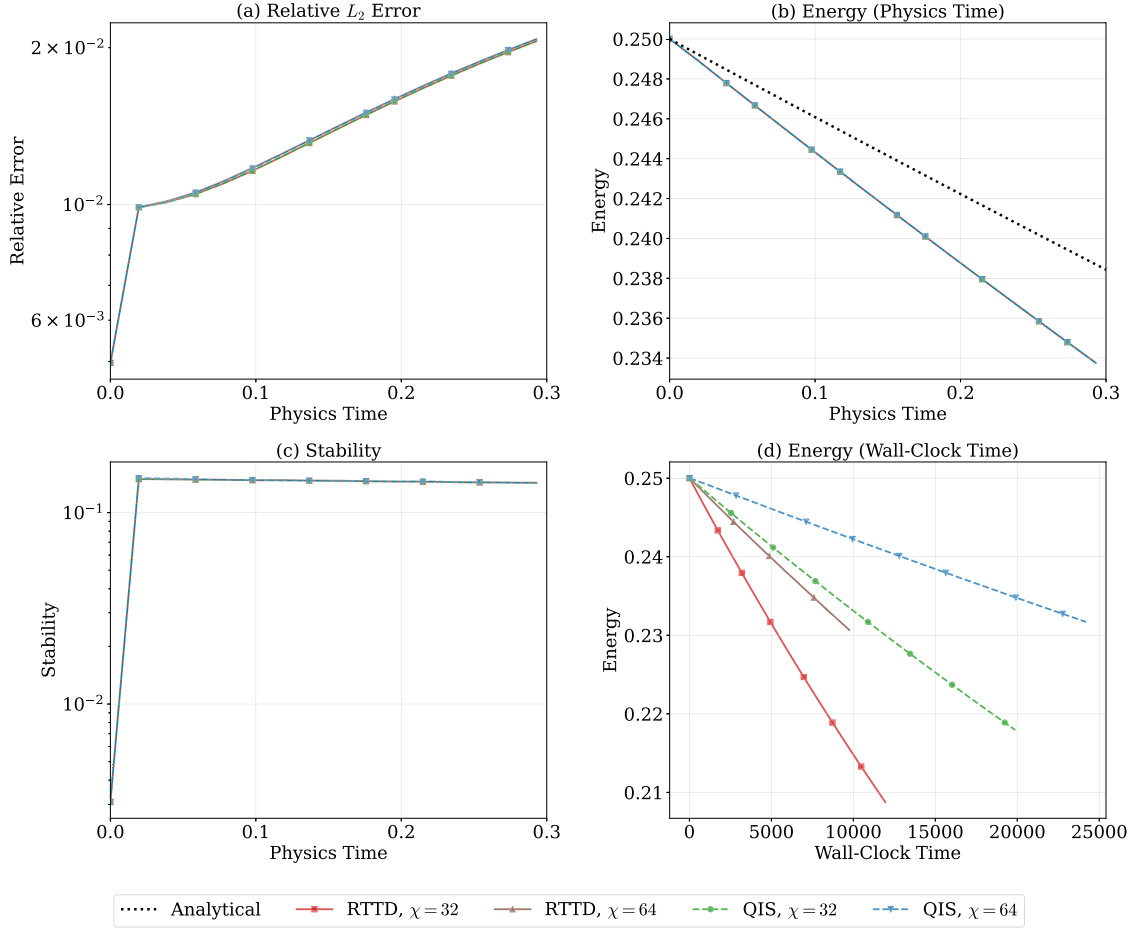


Fig. 2. Verification of the RTTD scheme against the deterministic QIS baseline and analytical solution for the 2D Taylor–Green vortex at $\text{Re} = 10^3$. (a) Relative L_2 error $\varepsilon(t)$. (b) Decay of total kinetic energy $E(t)$ in physical time (c) Stability ($D^{\max}$) (d) Kinetic energy decay versus wall-clock time.

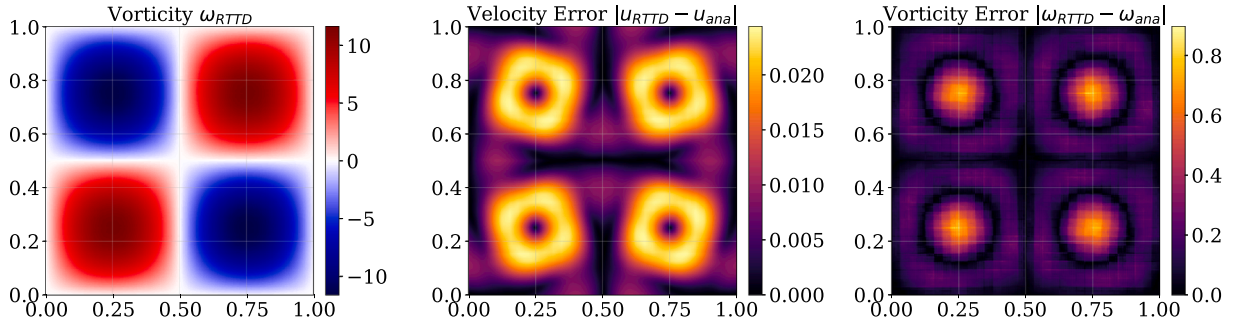


Fig. 3. Snapshot of the flow field and error distribution at $t = 0.3$ ($\chi = 32$, $n = 10$). (a) Vorticity field ω_{RTTD} (b) Pointwise velocity-magnitude error $|\mathbf{u}_{\text{RTTD}} - \mathbf{u}_{\text{ana}}|$ (c) Vorticity error magnitude $|\omega_{\text{RTTD}} - \omega_{\text{ana}}|$.

resolution shown. Panel (b)–(c) show spatially structured errors, with larger magnitudes localized near shear layers and vortex cores where gradients are largest, consistent with errors dominated by discretization and finite-rank truncation.

4.3. Decaying jet (DJ) benchmark

We consider the 2D decaying jet benchmark on the periodic unit square of side length 1, which initializes a horizontal shear layer that undergoes Kelvin–Helmholtz roll-up and develops progressively smaller scales, providing a demanding nonlinear test case.

The initial velocity is prescribed as

$$\mathbf{u}_{\text{init}}(x_1, x_2) = \mathbf{J}(x_2) + \mathbf{D}(x_1, x_2), \quad (28)$$

where the mean jet profile is

$$\mathbf{J}(x_2) = \hat{\mathbf{e}}_1 \frac{u_0}{2} \left[\tanh\left(\frac{x_2 - x_{\min}}{h}\right) - \tanh\left(\frac{x_2 - x_{\max}}{h}\right) - 1 \right], \quad (29)$$

with $u_0 = 1$, $x_{\min} = 0.4$, $x_{\max} = 0.6$, and $h = 1/200$. The disturbance is

$$\mathbf{D}(x_1, x_2) = \delta(\hat{\mathbf{e}}_1 d_1(x_1, x_2) + \hat{\mathbf{e}}_2 d_2(x_1, x_2)), \quad (30)$$

with $\delta = u_0/(40A)$ and

$$A = \max_{(x_1, x_2) \in \Omega} \sqrt{d_1(x_1, x_2)^2 + d_2(x_1, x_2)^2}, \quad (31)$$

where A is evaluated on the discrete $N \times N$ grid.

Define $\xi_k = (x_2 - x_k)/h$ for $k \in \{\min, \max\}$ and

$$\Sigma_s = \sin(8\pi x_1) + \sin(24\pi x_1) + \sin(6\pi x_1), \quad (32)$$

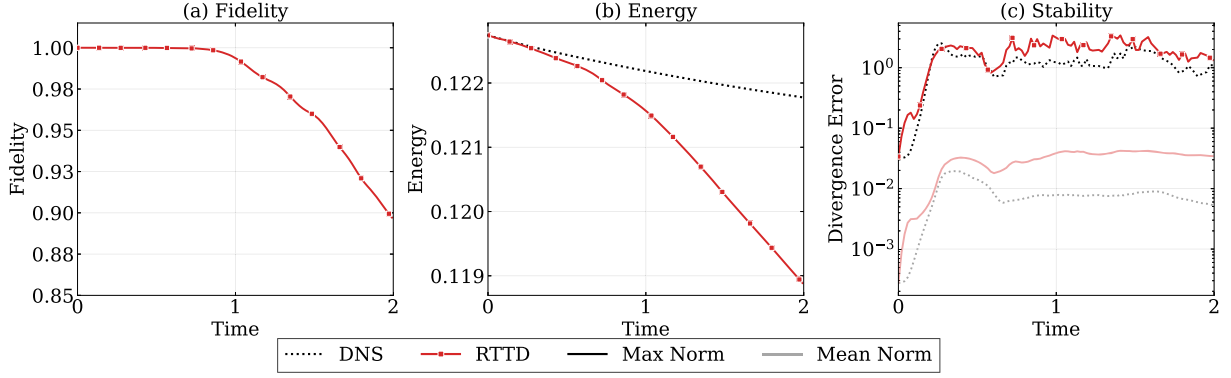


Fig. 4. DJ at $Re = 2 \times 10^5$: comparison of RTTD against the deterministic QIS baseline and DNS. **(a)** Fidelity $F_M(t)$ relative to DNS. **(b)** Kinetic energy $E_M(t)$. **(c)** Stability indicator: Maximum and mean discrete divergence, $D_M^{\max}(t)$ and $D_M^{\text{mean}}(t)$.

$$\Sigma_c = 8 \cos(8\pi x_1) + 24 \cos(24\pi x_1) + 6 \cos(6\pi x_1). \quad (33)$$

Then

$$d_1(x_1, x_2) = \frac{2}{h^2} \left[(x_2 - x_{\max})e^{-\frac{\epsilon^2}{2}} + (x_2 - x_{\min})e^{-\frac{\epsilon^2}{2}} \right] \Sigma_s, \quad (34)$$

$$d_2(x_1, x_2) = \pi \left[e^{-\frac{\epsilon^2}{2}} + e^{-\frac{\epsilon^2}{2}} \right] \Sigma_c. \quad (35)$$

This specification is deterministic and fully reproducible, and matches the DJ setup used in [29].

Let $\mathbf{u}^{DNS}(t)$ denote the discrete DNS velocity field at time t , and $\mathbf{u}^M(t)$ the corresponding field obtained with method $M \in \{QIS, RTTD\}$. Using the discrete inner product in Eq. (21), the fidelity of method M is

$$F_M(t) = \frac{|\langle \mathbf{u}^{DNS}(t), \mathbf{u}^M(t) \rangle_{\Delta x}|^2}{\|\mathbf{u}^{DNS}(t)\|_{\Delta x}^2 \|\mathbf{u}^M(t)\|_{\Delta x}^2}, \quad M \in \{QIS, RTTD\}. \quad (36)$$

We also report the energy $E_M(t)$ and divergence measures $D_M^{\max}(t)$ and $D_M^{\text{mean}}(t)$ as defined in Section 4.1.

4.3.1. Long-term accuracy and stability ($Re = 2 \times 10^5$)

To validate robustness in a turbulent-like regime, we perform a long-time integration for $t \in [0, 2.0]$ at $Re = 2 \times 10^5$.

Fig. 4 summarizes the long-term behavior. The fidelity (Fig. 4a) remains high throughout the transition to turbulence, and the kinetic energy decay (Fig. 4b) closely tracks the DNS dissipation trend. Fig. 4(c) confirms that both the maximum and mean divergence remain bounded over the full horizon.

Qualitative validation is provided in Fig. 5. The randomized solver successfully captures the formation of large-scale coherent structures ($t = 1.0$) and their subsequent breakdown into fine-scale turbulence ($t = 2.0$).

To verify scale-wise accuracy, we examine the instantaneous isotropic kinetic energy spectrum computed from the velocity snapshot at time t . Let $\hat{u}(k, t)$ be the discrete Fourier transform of the velocity field at time t . The 2D spectral energy density is defined as:

$$E(k_1, k_2) = \frac{1}{2} \sum_{i=1}^2 |\hat{u}_i(k_1, k_2, t)|^2. \quad (37)$$

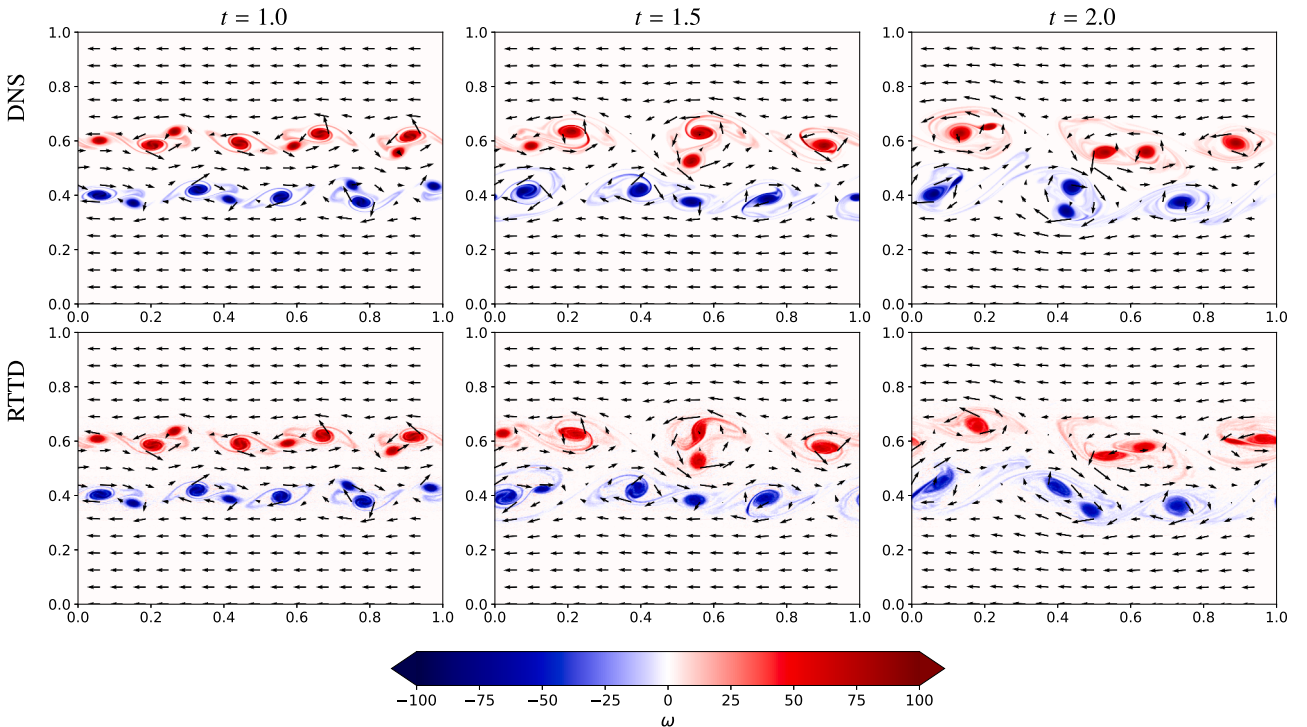


Fig. 5. DJ at $Re = 2 \times 10^5$. Snapshots of scalar vorticity ω (colors) and velocity field $\mathbf{u}$ (arrows) at $t = 1.0, 1.5, 2.0$. **Top:** DNS. **Bottom:** RTTD.

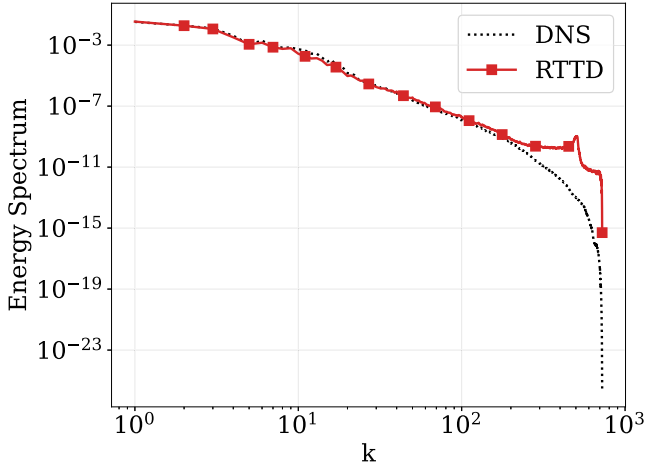


Fig. 6. Instantaneous isotropic kinetic energy spectrum $E(k)$ at $t = 2.0$ comparing RTTD and DNS.

The isotropic spectrum $E(k)$ is obtained by summing over annuli of constant wavenumber magnitude $k = \sqrt{k_1^2 + k_2^2}$:

$$E(k) = \sum_{(q_1, q_2): k - \frac{1}{2} < \sqrt{q_1^2 + q_2^2} \leq k + \frac{1}{2}} E(q_1, q_2). \quad (38)$$

Fig. 6 presents $E(k)$ at the final time $t = 2.0$. RTTD matches the DNS spectrum closely across the energy-containing and intermediate wavenumbers. A slight deviation is visible only at the highest wavenumbers, consistent with the implicit small-scale filtering expected from fixed-rank tensor truncation.

4.3.2. Algorithmic sensitivity

We fix $\text{Re} = 10^5$ and vary (n, χ) . All results in this subsection use the same simulation horizon, integrating the decaying jet over $t \in [0, 0.25]$. The velocity field is represented in tensor-train form with quantization depth $n \in \{10, 11, 12\}$ and maximum bond dimensions $\chi \in \{16, 32, 64, 100, 128\}$. For each (n, χ) configuration, we

- execute 20 warm-up time steps to amortize initialization and just-in-time compilation overheads,
- measure wall-clock runtimes over the subsequent 100 time steps, and
- report the mean wall-clock time per time step.

We compare QIS and RTTD against DNS using the diagnostics defined in Eqs. (23)–(36). **Fig. 7** shows the resulting time histories for several bond dimensions.

Fig. 7(a) reports the fidelity $F(t)$. For the lowest rank $\chi = 16$, both QIS and RTTD exhibit a gradual loss of overlap with DNS, reflecting an

under-resolved TT representation. In contrast, for $\chi \geq 32$ the RTTD and deterministic curves are visually indistinguishable and remain above $F \approx 0.999$ throughout the simulation window. This indicates that, at these ranks, the additional approximation introduced by RTTD is small relative to the discretization and truncation errors already present in the solver.

The energy evolution in **Fig. 7(b)** shows that RTTD preserves the energy decay curve. Across all tested ranks, the RTTD trajectories track the DNS reference closely; for $\chi \geq 64$ the discrepancy is within line thickness at the scale of the plot.

Fig. 7(c) shows the divergence norm $D(t)$. As expected, $D(t)$ increases slowly over time due to accumulated splitting and truncation errors. Crucially, the RTTD curves remain of the same order as the deterministic QIS curves for all χ , and we observe no systematic drift or blow-up attributable to the randomized truncation. Overall, the RTTD and QIS trajectories remain closely aligned across the tested (n, χ) settings.

We also analyze the effect of grid resolution by holding the bond dimension at $\chi = 32$ and varying the grid as $n \in \{10, 11, 12\}$ (**Fig. 8**). Across all tested resolutions, the RTTD curves are visually indistinguishable from the deterministic baseline in both fidelity and energy, and they collapse onto the DNS reference at each n . Fidelity remains ≥ 0.9995 throughout the window, while the divergence norm stays bounded with no systematic drift attributable to RTTD, confirming robust physical fidelity for state spaces up to 2^{2n} .

4.3.3. Runtime scaling with n and χ

Fig. 9 summarizes the impact of RTTD on wall-clock time per time step. RTTD removes the dominant truncation bottleneck in nonlinear MPO construction. On an RTX 4090, the nonlinear MPO construction time decreases from 4.72 s to 0.61 s per step (7.8 $\times$), yielding a 2.2 $\times$ reduction in end-to-end per-step runtime (from 6.33 s to 2.89 s). The remaining cost is dominated by contraction-heavy variational sweeps, consistent with Amdahl-limited scaling. A detailed instrumented breakdown is provided in **Appendix B (Table B.1)**.

For fixed $\chi = 32$ and increasing n from 10 (1024² grid) to 12 (4096² grid), see **Fig. 9(a)**, the baseline QIS runtime grows from 7.42 s to 8.90 s per time step, while RTTD scales from 4.35 s to 5.08 s. The end-to-end speedup therefore increases slightly with problem size and reaches $S \approx 1.75$ at $n = 12$, indicating that the QR-based randomized truncation continues to mitigate the cost of large TT operators as the grid is refined.

At fixed $n = 10$ and varying χ , **Fig. 9(b)** shows that both solvers become more expensive as the bond dimension grows, but RTTD consistently remains faster. For example, at $\chi = 32$ the mean step time drops from 7.42 s (QIS) to 4.35 s (RTTD), a 1.7 $\times$ speedup. Even at $\chi = 100$, where the tensors are substantially less compressible, RTTD reduces runtime from 44.42 s to 33.23 s ($S \approx 1.3$). Thus, the benefits of avoiding large-scale SVDs persist across the practically relevant range of ranks.

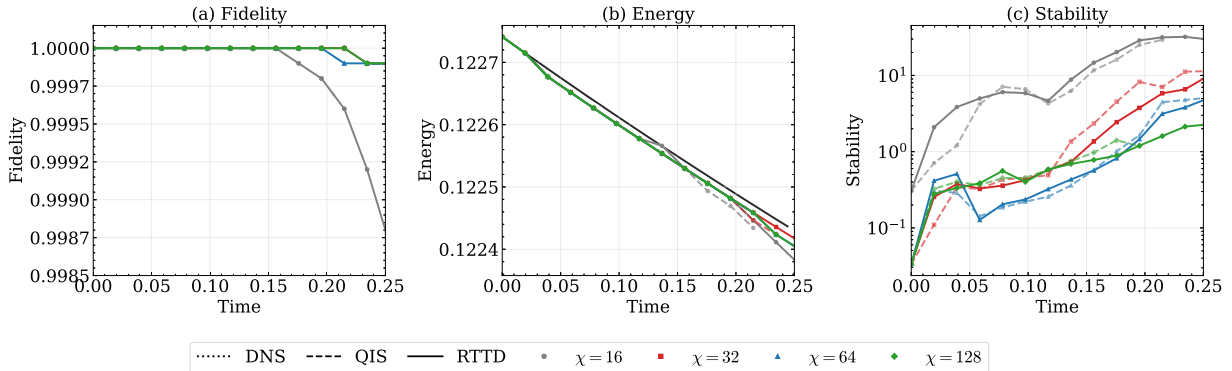


Fig. 7. Accuracy of RTTD compared to the deterministic QIS baseline for the decaying jet at $\text{Re} = 10^5$ and various bond dimensions χ . **(a)** Fidelity. **(b)** Energy. **(c)** Stability.

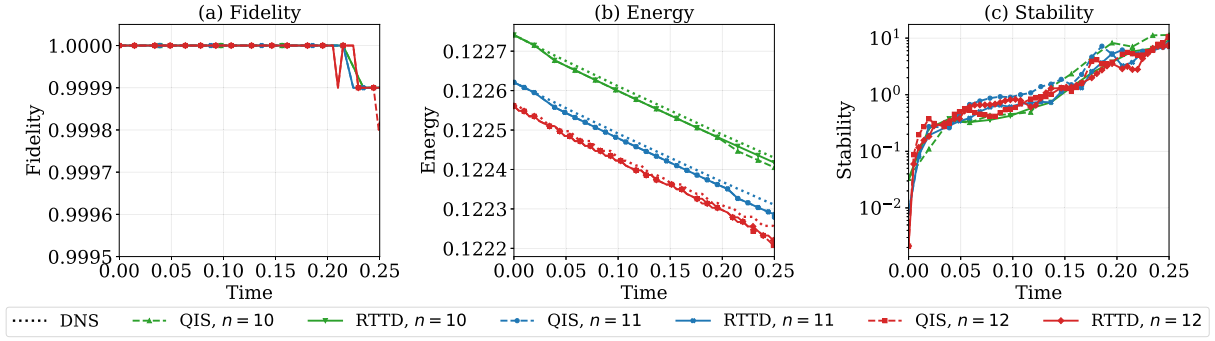


Fig. 8. Accuracy of RTTD compared to the deterministic QIS baseline for the decaying jet at $\text{Re} = 10^5$ and varying grid resolutions n (fixed $\chi = 32$). (a) Fidelity $F(t)$. (b) Energy $E(t)$. (c) Maximum divergence norm $D(t)$.

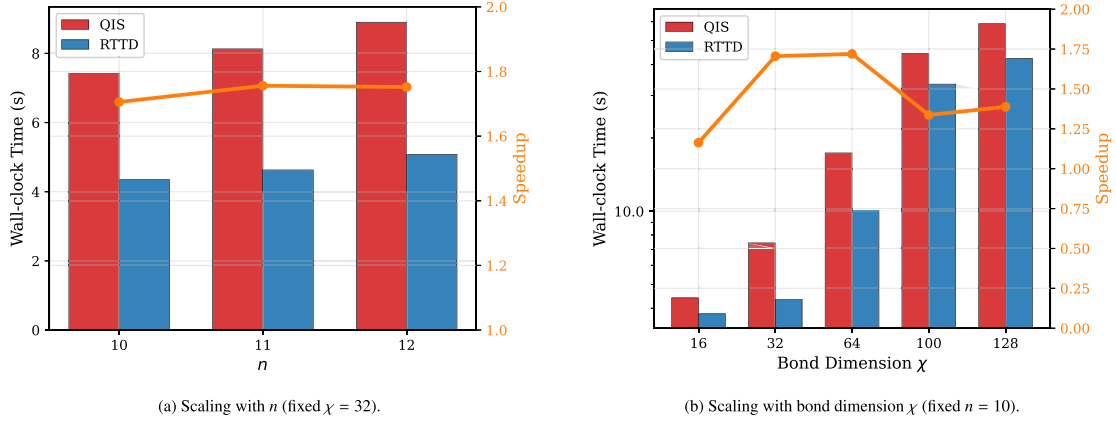


Fig. 9. Runtime and speedup of RTTD versus the cuQuantum baseline (QIS). Bars (left axis) show mean wall-clock time per step; the red line (right axis) shows the speedup $S = T_{\text{QIS}}/T_{\text{RTTD}}$. (a) Scaling with grid resolution: RTTD achieves up to $1.75\times$ speedup at $n = 12$ (4096^2 grid). (b) Scaling with bond dimension: RTTD maintains a $1.3\times$ – $1.7\times$ advantage across $\chi \in [16, 128]$, including the high-accuracy regime ($\chi \geq 64$). Runtime is plotted on a logarithmic scale. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

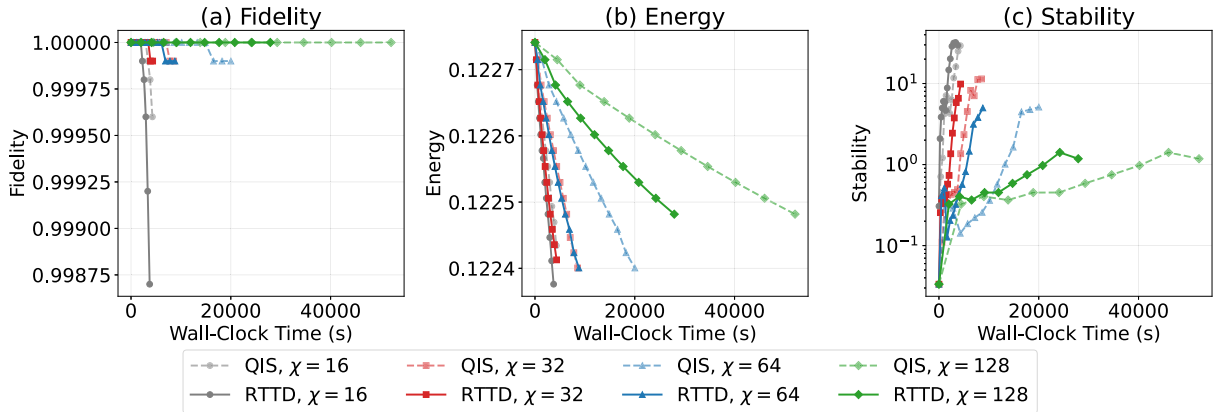


Fig. 10. Time-to-solution analysis for the decaying jet at $\text{Re} = 10^5$, varying χ . RTTD is compared to the deterministic QIS baseline. (a) Fidelity. (b) Energy. (c) Stability.

We analyze time-to-solution by plotting key diagnostics against accumulated wall-clock time, rather than physical time. In this representation, a faster method reaches the same diagnostic values earlier, appearing as a left-shift of the trajectory.

Time-to-solution at fixed resolution. In Fig. 10, RTTD trajectories are consistently left-shifted relative to the baseline across χ . For $\chi = 128$, RTTD reaches the final simulated state ($t = 0.25$) in $\approx 3.0 \times 10^4$ s, compared to $\approx 5.0 \times 10^4$ s for the baseline, while exhibiting comparable fidelity and energy decay over the simulated interval.

Scaling with grid resolution. Fig. 11 shows the same effect for increasing resolution at fixed $\chi = 32$: for each $n \in \{10, 11, 12\}$, RTTD reaches comparable diagnostic levels with reduced wall-clock latency.

4.4. Memory usage

To verify that the runtime speedups of the RTTD kernel do not come at the expense of excessive memory usage, we analyze the peak GPU memory consumption of the solver. We instrument two critical phases: (i) the MPO construction phase, where the truncation is

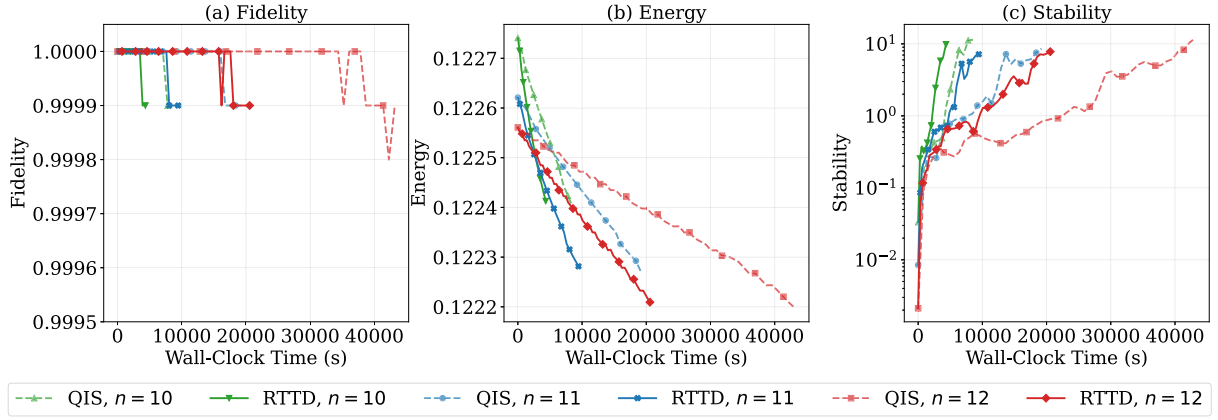


Fig. 11. Time-to-solution analysis for the decaying jet at $Re = 10^5$, varying n (fixed $\chi = 32$). RTTD is compared to the deterministic QIS baseline. (a) Fidelity $F(t)$. (b) Energy $E(t)$. (c) Maximum divergence norm $D(t)$.

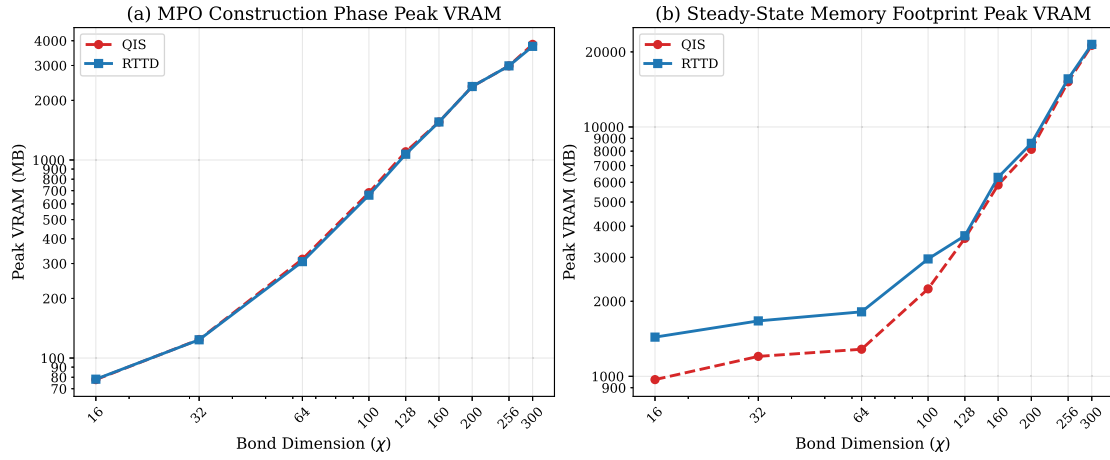


Fig. 12. Memory scaling with bond dimension χ (fixed $n = 10$). (a) MPO construction phase (b) Steady-state memory footprint.

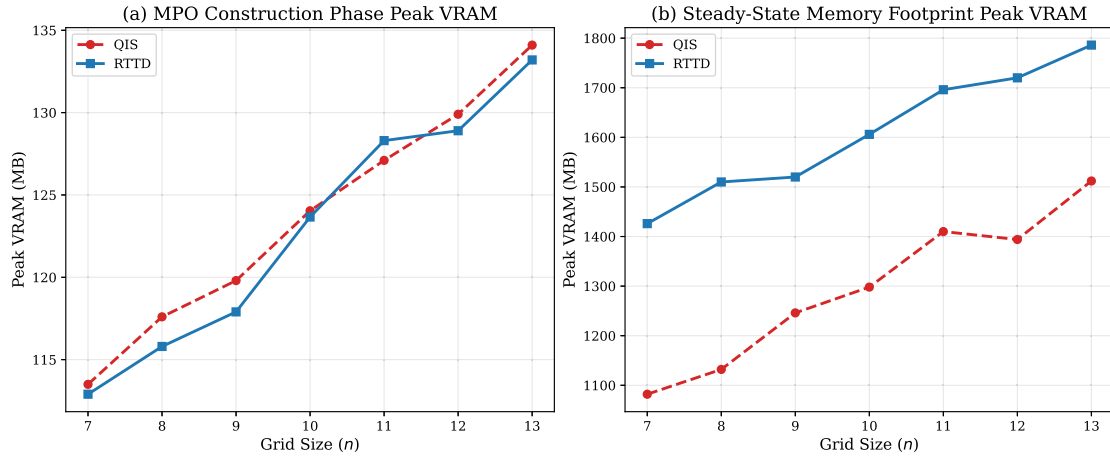


Fig. 13. Memory scaling with n (fixed $\chi = 32$). (a) MPO construction phase (b) Steady-state memory footprint.

performed, and (ii) the steady-state memory footprint, representing the GPU memory required to store the persistent simulation state (velocity tensors), operators, and environments during the time-stepping loop (see Fig. 13).

Fig. 12 illustrates memory consumption as the bond dimension increases from $\chi = 16$ to $\chi = 300$. Panel (a) shows that the peak VRAM

usage during the MPO construction phase is nearly identical for RTTD and the deterministic baseline. This indicates that the auxiliary random matrices used in the sketching procedure do not impose a prohibitive memory overhead compared to the baseline truncation workspaces. In the steady-state memory footprint (panel b), both methods converge to the same memory usage at large χ (approximately 21 GB at $\chi = 300$),

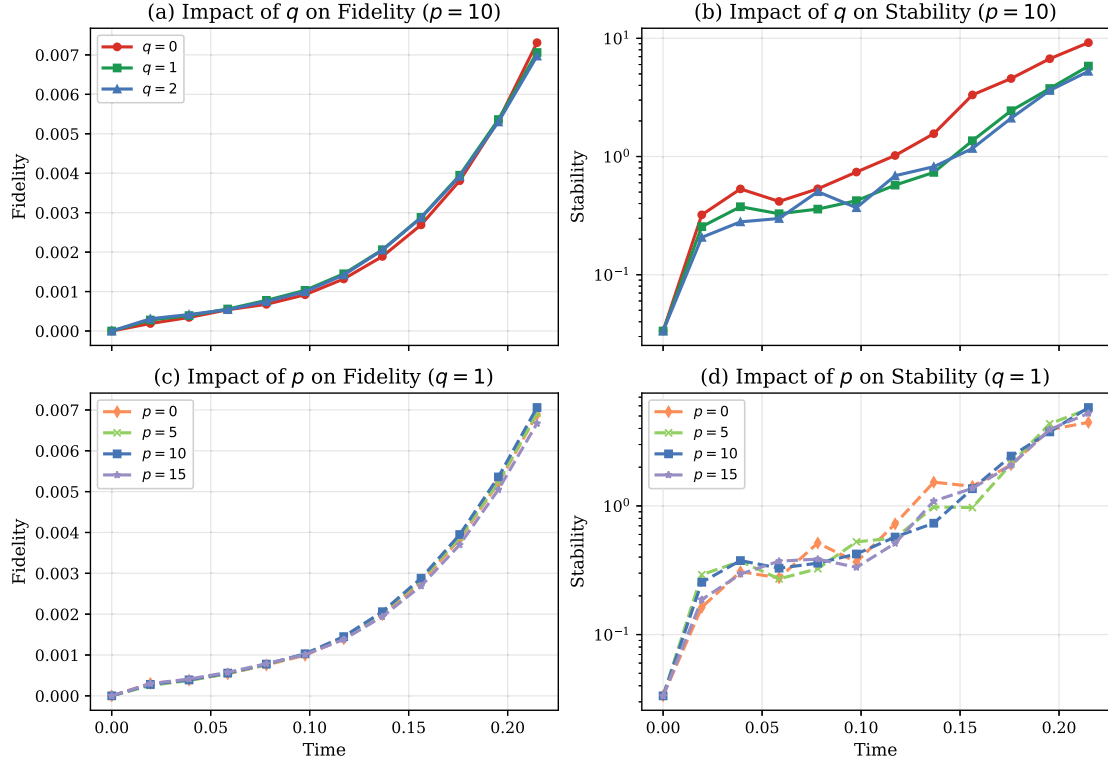


Fig. 14. Sensitivity of solution accuracy and stability to RTTD hyperparameters. (a, b) Impact of power iterations q (fixed $p = 10$). (c, d) Impact of oversampling p (fixed $q = 1$).

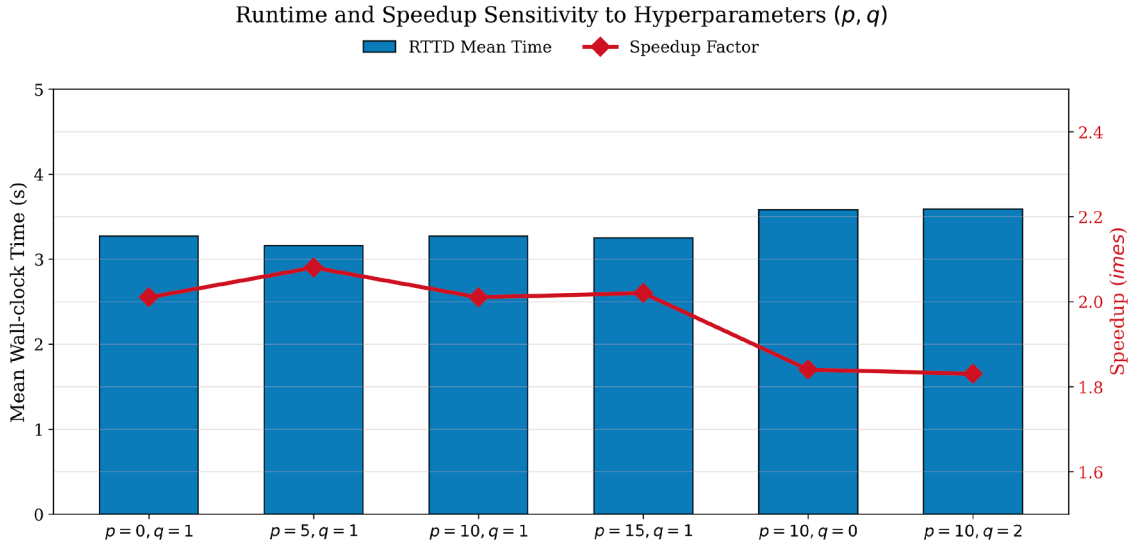


Fig. 15. Sensitivity of RTTD performance to hyperparameters (oversampling p , power iterations q). Bars (left axis) show mean wall-clock time per step; the line (right axis) shows the speedup relative to the deterministic QIS baseline.

which fits within the 24 GB device memory of the RTX 4090 used here. RTTD exhibits a slightly higher baseline usage at small χ , attributed to pre-allocation of persistent workspace buffers for randomized linear-algebra routines. This overhead becomes negligible in relative terms as χ increases.

Figure 13 further examines the memory scaling with grid size n at fixed bond dimension $\chi = 32$. For the MPO construction phase, Fig. 13(a), RTTD and the deterministic baseline show nearly identical peak VRAM usage, with only a modest increase as n grows. This suggests that the randomized truncation does not add significant tempo-

rary memory overhead during construction. The steady-state footprint in Fig. 13(b) is moderately higher for RTTD. This offset, on the order of a few hundred megabytes, comes from persistent workspace buffers used by the randomized linear-algebra routines. Since the offset remains small relative to the 24 GB GPU memory budget, the memory results indicate that RTTD preserves the practical memory profile of the baseline while providing the runtime benefits reported above.

Overall, Figs. 12 and 13 show that RTTD preserves the memory scalability of the deterministic baseline while adding only a modest fixed workspace overhead.

4.5. Hyperparameter study: oversampling and power iterations

The randomized RTTD algorithm introduces two key hyperparameters: the oversampling parameter p and the number of power iterations q . These parameters control the trade-off between the quality of the randomized range approximation and the computational cost of the truncation step. To identify an effective configuration for turbulent flow simulations, we conducted a sensitivity analysis on the decaying-jet benchmark ($N = 1024$, $\chi = 32$).

Fig. 14 summarizes the evolution of the fidelity $F_{RTTD}(t)$ and the maximum divergence $D_{RTTD}^{\max}(t)$ for several (p, q) configurations.

Impact of power iterations (q). As shown in Fig. 14(b), the power-iteration count q plays a critical role in stability. The configuration with $q = 0$ exhibits the fastest growth in the divergence diagnostic, reaching $D_M^{\max}(t) \approx 9.2$ at $t = 0.21$. Increasing to $q = 1$ significantly suppresses this growth, reducing the final value to ≈ 5.8 , while a further increase to $q = 2$ yields only marginal improvement. Since each power iteration requires additional applications of the MPOs to the sketch vectors, we select $q = 1$ as the best trade-off between improved stability and added cost.

Impact of oversampling (p). Fig. 14(c,d) shows that the solver is largely insensitive to the oversampling parameter p for this benchmark. For $p \in \{0, 5, 10, 15\}$ with fixed $q = 1$, the trajectories of both the fidelity and the divergence diagnostic remain tightly clustered, suggesting that modest oversampling is adequate at the tested rank.

Fig. 15 reports per-step runtimes for several (p, q) settings. The runtime is approximately invariant with respect to p : configurations with $p \in \{0, 5, 10, 15\}$ at fixed $q = 1$ cluster around ~ 3.2 s per step, corresponding to a speedup of about $2.0\times$ relative to the deterministic baseline. In contrast, increasing q from 1 to 2 increases the runtime to ~ 3.6 s per step, reducing the speedup to about $1.8\times$.

Based on this trade-off, we select $(p, q) = (5, 1)$ as the default configuration. The q sweep in Fig. 14(a,b) uses $p = 10$ to reduce sensitivity to undersampling while isolating the effect of q ; in practice, performance at fixed $q = 1$ is similar for $p = 5$ (Fig. 15).

5. Conclusion

We addressed a key performance bottleneck in GPU-resident quantum-inspired CFD: repeated SVD-based truncation during MPO construction. To resolve this, we introduce RTTD, a randomized tensor-train truncation scheme that replaces per-bond truncated SVD with oversampled randomized range finding followed by QR-based factorizations. This redesign shifts the dominant truncation work toward GEMM- and QR-dominated kernels while preserving the orthogonality structure required by TT workflows.

Integrated into a cuTensorNet-compatible QIS-CFD pipeline, RTTD delivers substantial acceleration. Across the 2D decaying jet and Taylor-Green vortex benchmarks, the method preserves the physical diagnostics reported in this work—including energy evolution, divergence control, and state fidelity—while remaining consistent with DNS and the deterministic QIS baseline. On an RTX 4090, RTTD achieves up to $7.8\times$ acceleration in the dominant truncation kernel and a $2.2\times$ reduction in end-to-end wall-clock time. At practical bond dimensions (e.g., $\chi \geq 32$), the additional approximation introduced by RTTD remains smaller than the discretization error of the underlying Navier-Stokes scheme in our experiments.

Discussion and future work. This study focused on single-GPU execution of two-dimensional flows to isolate the impact of the truncation redesign. Future work will extend RTTD to three-dimensional turbulence and higher Reynolds numbers, where rank growth and contraction costs are more pronounced, and will investigate adaptive truncation control using online spectral indicators to adjust (p, q) without manual tuning. Finally, extending RTTD to distributed multi-GPU settings and

co-optimizing it with communication-aware contraction schedules is a priority for scaling QIS-CFD workflows to larger state spaces.

As an additional stress test, we also evaluated a harder decaying-turbulence (DT) initial condition following Hölscher et al. [29]. Under the same solver configuration, the deterministic SVD-based baseline and RTTD showed comparable early-step behavior. Over subsequent steps, both methods degraded rapidly, indicating that this case is not adequately resolved by the present fixed-rank QIS-CFD configuration rather than exposing an RTTD-specific failure. This behavior is consistent with prior observations that decaying turbulence requires larger bond dimensions than the decaying-jet case and that QIS-CFD can struggle to accurately approximate the turbulent kinetic energy spectrum of DT under highly turbulent conditions. These results further motivate adaptive-rank RTTD, where the effective bond dimension is selected from a tolerance criterion rather than prescribed by a fixed hard cap.

CRedit authorship contribution statement

Kiet Tuan Pham: Writing – review & editing, Supervision, Project administration; **Gordon Euhyun Moon:** Writing – review & editing, Funding acquisition; **Jinsung Kim:** Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jinsung Kim reports financial support was provided by Chung-Ang University. Jinsung Kim reports financial support was provided by National Research Foundation. Kiet Pham reports financial support was provided by Chung-Ang University. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-RS-2024-00359076). This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) under the Leading Generative AI Human Resources Development(IITP-2026-25545512) grant funded by the Korea government(MSIT). This research was supported by the Chung-Ang University Young Scientist Scholarship in 2022.

Data availability

Data will be made available on request. Full code repository: <https://github.com/CAU-HPCL/RTTD>.

Appendix A. Kernel micro-benchmarks and stability

We evaluate the performance of the isolated RTTD truncation kernel against state-of-the-art tensor libraries. Fig. A.1 compares the execution time of RTTD against cuQuantum (deterministic SVD and randomized SVD), TNTorch, and Tensorly for truncating matrices of size $M \times K$, where $K = 23328$ is fixed (typical for $n = 10$ simulations) and M varies.

Fig. A.1 compares the execution time of RTTD against NVIDIA cuQuantum, TNTorch, and Tensorly. For fixed target bond dimension χ , the truncation step requires an orthonormal basis for projection and a canonical (gauge-fixed) representation, and any residual scaling can be absorbed into neighboring TT cores. Although cuTensorNet exposes a randomized SVD option, in our setup it was not a practical baseline for the matrix sizes and bond shapes arising in the QIS-CFD loop: it exhibited excessive runtime and, in memory-stress regimes, failed with device

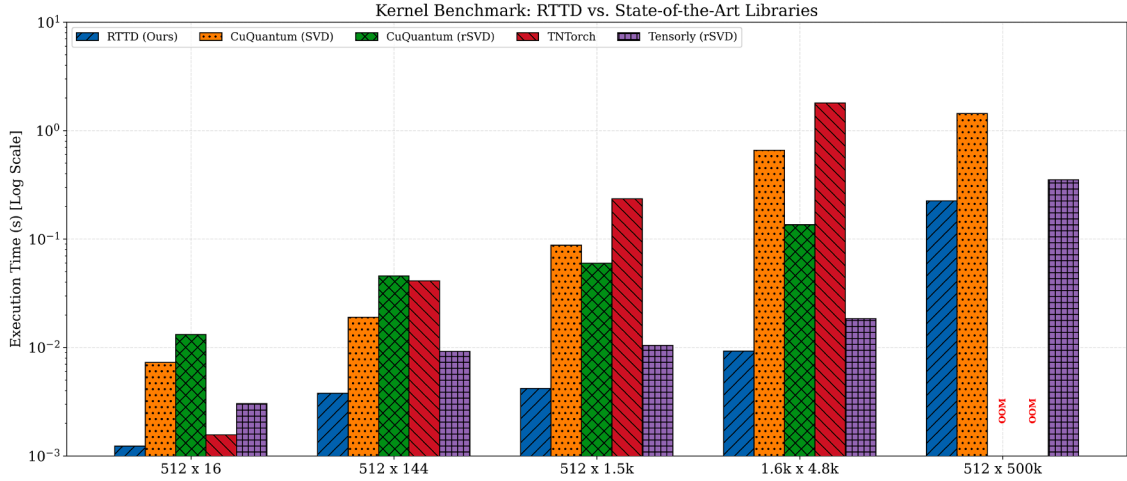


Fig. A.1. Kernel-level benchmark of RTTD vs. existing GPU-accelerated libraries.

out-of-memory errors. We therefore report cuQuantum rSVD only as a kernel-level reference in Fig. A.1 and focus the end-to-end comparisons on the deterministic cuQuantum contract-decompose path and DNS.

1. Adaptive Low-Rank Efficiency: For small boundary matrices (512×16), RTTD dispatches to a direct SVD routine (PyTorch `svd_lowrank`) to avoid randomized sketching overhead; otherwise RTTD uses the QR-based randomized path. This dispatch matches the performance of lightweight wrappers like Tensorly and outperforms the standard cuQuantum path. In the end-to-end QIS-CFD solver runs Section 4, we use the QR-only RTTD path; the SVD dispatch is used only in the standalone microbenchmarks on very small matrices to remove sketching overhead.

2. Dominance in Production Workloads: In the most frequent operating regime (512×144), the RTTD C++ kernel becomes active and highly efficient. It completes in 3.8 ms, achieving a $5\times$ speedup over cuQuantum and an $11\times$ speedup over Tensorly, which suffers from framework overheads at this scale.

3. Extreme Scalability: For high-fidelity simulations ($1.6k \times 4.8k$), RTTD exhibits near-constant scaling, delivering a $6.5\times$ speedup over the deterministic baseline. Furthermore, in the extreme memory stress test ($512 \times 500k$), RTTD is the most robust randomized method; while cuQuantum’s randomized SVD and TNTorch fail with OOM errors, RTTD finishes $1.59\times$ faster than the surviving Tensorly implementation.

This kernel-level efficiency directly translates to the application-level speedups discussed in Section 4.

Appendix B. Performance analysis

We examine how RTTD changes the runtime distribution of the QIS-CFD time-stepping loop. We instrument wall-clock timers for the four

high-level regions defined in Fig. 1 and report per-step averages for the decaying-jet benchmark ($n = 10$, $Re = 10^5$). Table B.1 summarizes timings averaged over 100 steps.

In the cuQuantum baseline, nonlinear MPO construction dominates the per-step runtime ($\approx 75\%$), reflecting the cost of repeated SVD-based truncations inside contract-decompose. Replacing these truncations with RTTD reduces this region from 4.72 s to 0.61 s ($7.8\times$), yielding a $2.2\times$ end-to-end speedup per time step.

With RTTD, the cost of the nonlinear construction phase is reduced from 4.72 s to 0.61 s, corresponding to a local speedup of $7.8\times$. This optimization reduces the share of MPO construction to roughly 29% of the per-step runtime. Consequently, the relative cost of DMRG sweeps increases to 58%, even though the solver achieves an end-to-end wall-clock speedup of $2.2\times$.

After this bottleneck is reduced, the DMRG sweep phase becomes the dominant cost ($\approx 58\%$). We observe that sweep time increases from 1.03 s (SVD) to 1.68 s (RTTD). A likely contributor is truncation optimality at fixed rank: truncated SVD yields the Frobenius-optimal rank- χ approximation, whereas RTTD produces an approximate rank- χ subspace. At the same cap χ , this can yield slightly less compressible intermediates (e.g., less favorable effective-rank profiles after canonicalization), increasing contraction work during sweeps. Despite this shift, the overall runtime decreases due to the large reduction in nonlinear MPO construction.

Finally, once MPO construction is accelerated, the achievable end-to-end speedup is limited by the remaining contraction- and solve-dominated regions; additional gains would require reducing DMRG sweep cost (e.g., fewer sweeps/iterations or cheaper contractions), rather than further optimizing the already-shortened truncation kernel.

Table B.1

Per-step runtime breakdown for the decaying-jet case ($n = 10$, $Re = 10^5$), averaged over 100 time steps. RTTD replaces cuQuantum’s deterministic SVD-based truncation in contract-decompose operations during nonlinear MPO construction.

Region	cuQ [s]	RTTD [s]	Speedup [$\times$]	cuQ [%]	RTTD [%]
Env precontraction (Ax/Ay)	0.002	0.003	0.67	0.07	0.12
Nonlinear MPO construction (contract + decompose)	4.715	0.608	7.75	74.54	29.36
Env precontraction (conv-diff)	0.579	0.602	0.96	9.05	20.69
DMRG sweeps (total)	1.034	1.677	0.62	16.34	57.83
TOTAL	6.330	2.890	2.19	100.0	100.0

References

- [1] I. Tiselj, C. Flageul, J. Oder, Direct numerical simulation and wall-resolved large eddy simulation in nuclear thermal hydraulics, *Nucl. Technol.* 206 (2) (2020) 164–178.
- [2] P. Moin, K. Mahesh, Direct numerical simulation: a tool in turbulence research, *Annu. Rev. Fluid Mech.* 30 (1998) 539–578. <https://doi.org/10.1146/annurev.fluid.30.1.539>
- [3] M. Chiarini, G. Tandurella, M.E. Rosti, Kolmogorov-size particles in homogeneous and isotropic turbulence, *J. Fluid Mech.* 1007 (2025) A81. <https://doi.org/10.1017/jfm.2025.122>
- [4] L. Schneiders, M. Meinke, W. Schröder, Direct particle–fluid simulation of Kolmogorov-length-scale size particles in decaying isotropic turbulence, *J. Fluid Mech.* 819 (2017) 188–227.
- [5] C. Hirsch, *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics*, Elsevier, 2007.
- [6] G.K. Batchelor, *An Introduction to Fluid Dynamics*, Cambridge University Press, 2000.
- [7] S.B. Pope, *Turbulent Flows*, Cambridge University Press, Cambridge, 2000.
- [8] M. Oberlack, S. Hoyas, F. Alcántara-Ávila, A. Kraheberger, J. Laux, Wall turbulence at high friction Reynolds numbers, *Phys. Rev. Fluids* 7 (2022) 014602. <https://doi.org/10.1103/PhysRevFluids.7.014602>
- [9] J. Yao, X. Chen, F. Hussain, Direct numerical simulation of turbulent open channel flows at moderately high Reynolds numbers, *J. Fluid Mech.* 953 (2022) A19. <https://doi.org/10.1017/jfm.2022.942>
- [10] T. Ishihara, T. Gotoh, Y. Kaneda, Study of high-Reynolds number isotropic turbulence by direct numerical simulation, *Annu. Rev. Fluid Mech.* 41 (2009) 165–180. <https://doi.org/10.1146/annurev.fluid.010908.165203>
- [11] M. Lee, R.D. Moser, Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$, *J. Fluid Mech.* 774 (2015) 395–415. <https://doi.org/10.1017/jfm.2015.268>
- [12] X. Yang, K.P. Griffin, Grid-point and time-step requirements for direct numerical simulation and Large-Eddy simulation, *Phys. Fluids* 33 (1) (2021) 015108. <https://doi.org/10.1063/5.0036515>
- [13] F. De Vanna, F. Avanzi, M. Cogo, S. Sandrin, M. Bettencourt, F. Picano, E. Benini, GPU-Acceleration of Navier-Stokes solvers for compressible wall-bounded flows: the case of URANOS, in: *AIAA SCITECH 2023 Forum*, 2023, pp. 1129.
- [14] G. Dang, S. Liu, T. Guo, J. Duan, X. Li, Direct numerical simulation of compressible turbulence accelerated by graphics processing unit: an open-access database of high-resolution direct numerical simulation, *AIP Adv.* 12 (12) (2022).
- [15] Y. Wang, G. Cao, L. Pan, Multiple-GPU accelerated high-order gas-kinetic scheme for direct numerical simulation of compressible turbulence, *J. Comput. Phys.* 476 (2023) 111899.
- [16] P. Costa, E. Phillips, L. Brandt, M. Fatica, GPU Acceleration of CaNS for massively-parallel direct numerical simulations of canonical fluid flows, *Comput. Math. Appl.* 81 (2021) 502–511.
- [17] Z.-M. Liao, L.-B. Chen, Z.-H. Wan, N.-S. Liu, X.-Y. Lu, GPU Acceleration of four-way coupled PP-DNS for compressible particle-laden wall turbulence, *Int. J. Multiphase Flow* 176 (2024) 104840.
- [18] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, *Ann. Phys.* 326 (1) (2011) 96–192. <https://doi.org/10.1016/j.aop.2010.09.012>
- [19] R. Orús, A practical introduction to tensor networks: matrix product states and projected entangled pair states, *Ann. Phys.* 349 (2014) 117–158. <https://doi.org/10.1016/j.aop.2014.06.013>
- [20] D. Perez-Garcia, F. Verstraete, M.M. Wolf, J.I. Cirac, Matrix product state representations, *Quant. Inf. Comput.* 7 (5-6) (2007) 401–430. <https://doi.org/10.26421/QIC7.5-6-1>
- [21] I.V. Oseledets, Tensor-train decomposition, *SIAM J. Sci. Comput.* 33 (5) (2011) 2295–2317. <https://doi.org/10.1137/090752286>
- [22] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, *SIAM Rev.* 57 (4) (2015) 483–531. <https://doi.org/10.1137/130932715>
- [23] P. Holmes, J.L. Lumley, G. Berkooz, C.W. Rowley, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge University Press, 2nd ed., 2012. <https://doi.org/10.1017/CBO9780511919701>
- [24] M. Kiffer, D. Jaksch, Tensor network reduced order models for wall-bounded flows, *Phys. Rev. Fluids* 8 (2023) 124101. <https://doi.org/10.1103/PhysRevFluids.8.124101>
- [25] B.N. Khoromskij, $O(d \log n)$ -Quantics approximation of $n-d$ tensors in high-dimensional numerical modeling, *Constr. Approx.* 34 (2) (2011) 257–280. <https://doi.org/10.1007/s00365-011-9131-1>
- [26] V.A. Kazeev, B.N. Khoromskij, Low-rank explicit QTT representation of the Laplace operator and its inverse, *SIAM J. Matrix Anal. Appl.* 33 (3) (2012) 742–758. <https://doi.org/10.1137/100820479>
- [27] I. Oseledets, E. Tyrtysnikov, TT-Cross approximation for multidimensional arrays, *Linear Algebra Appl.* 432 (1) (2010) 70–88. <https://doi.org/10.1016/j.laa.2009.07.024>
- [28] N. Gourianov, M. Lubasch, S. Dolgov, Q.Y. van den Berg, H. Babae, P. Givi, M. Kiffer, D. Jaksch, A quantum-inspired approach to exploit turbulence structures, *Nat. Comput. Sci.* 2 (1) (2022) 30–37. <https://doi.org/10.1038/s43588-021-00181-1>
- [29] L. Hölscher, P. Rao, L. Müller, J. Klepsch, A. Luckow, T. Stollenwerk, Quantum-inspired fluid simulation of two-dimensional turbulence with GPU acceleration, *Phys. Rev. Res.* 7 (1) (2025) 013112. <https://doi.org/10.1103/PhysRevResearch.7.013112>
- [30] S. Pisoni, R.D. Peddinti, S.E. Guzman, E. Tiunov, L. Aolita, Compression, simulation, and synthesis of turbulent flows with tensor trains, 2025, <https://doi.org/10.48550/arXiv.2506.05477>
- [31] R.D. Peddinti, S. Pisoni, A. Marini, P. Lott, H. Argenterio, E. Tiunov, L. Aolita, Quantum-inspired framework for computational fluid dynamics, *Commun. Phys.* 7 (2024) 135. <https://doi.org/10.1038/s42005-024-01623-8>
- [32] X.-Y. Liu, H. Hong, Z. Zhang, W. Tong, J. Kossaifi, X. Wang, A. Walid, High-performance tensor-train primitives using GPU tensor cores, *IEEE Trans. Comput.* 73 (11) (2024) 2634–2648. <https://doi.org/10.1109/TC.2024.3441831>
- [33] M. Bachmayr, R. Schneider, A. Uschmajew, Low-rank tensor methods for partial differential equations, *Acta Numer.* 32 (2023) 1–121. <https://doi.org/10.1017/S0962492922000125>
- [34] S.V. Dolgov, B.N. Khoromskij, I.V. Oseledets, Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker–Planck equation, *SIAM J. Sci. Comput.* 34 (6) (2012) A3016–A3038. <https://doi.org/10.1137/120864210>
- [35] L. Richter, L. Sallandt, N. Nüsken, Solving high-dimensional parabolic PDEs using the tensor train format, in: *International Conference on Machine Learning, PMLR*, 2021, pp. 8998–9009.
- [36] E. Kornev, S. Dolgov, M. Perelshtein, A. Melnikov, TetraFEM: numerical solution of partial differential equations using tensor train finite element method, *Mathematics* 12 (20) (2024) 3277.
- [37] Q.T. Tran, D.P. Truong, K.Ø. Rasmussen, B. Alexandrov, A Tensor Train-Based Isogeometric Solver for Large-Scale 3D Poisson Problems on Complex Geometries, 2025, Also available at SSRN: <https://doi.org/10.2139/ssrn.5497561> (accessed 2026-01-19), <https://doi.org/10.48550/arXiv.2509.13224>
- [38] M.E. Danis, D.P. Truong, I.D. Boureima, O. Korobkin, K.O. Rasmussen, B. Alexandrov, Tensor-train WENO scheme for compressible flows, *J. Comput. Phys.* 529 (2025) 113891. <https://doi.org/10.1016/j.jcp.2025.113891>
- [39] N. Halko, P.-G. Martinsson, J.A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, *SIAM Rev.* 53 (2) (2011) 217–288. <https://doi.org/10.1137/090771806>
- [40] G. Yu, J. Feng, Z. Chen, X. Cai, L. Qi, A randomized block Krylov method for tensor train approximation, 2023, <https://doi.org/10.48550/arXiv.2308.01480>
- [41] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, *SIAM Rev.* 51 (3) (2009) 455–500. <https://doi.org/10.1137/07070111X>
- [42] I.V. Oseledets, Constructive representation of functions in low-rank tensor formats, *Constr. Approx.* 37 (1) (2013) 1–18. <https://doi.org/10.1007/s00365-012-9175-x>