

<https://doi.org/10.1038/s42005-024-01623-8>

Quantum-inspired framework for computational fluid dynamics

Check for updates

Raghavendra Dheeraj Peddinti^{1,2}✉, Stefano Pisoni^{1,3}, Alessandro Marini⁴, Philippe Lott⁴, Henrique Argentieri⁴, Egor Tiunov¹ & Leandro Aolita^{1,5}

Computational fluid dynamics is both a thriving research field and a key tool for advanced industry applications. However, the simulation of turbulent flows in complex geometries is a compute-power intensive task due to the vast vector dimensions required by discretized meshes. We present a complete and self-consistent full-stack method to solve incompressible fluids with memory and run time scaling logarithmically in the mesh size. Our framework is based on matrix-product states, a compressed representation of quantum states. It is complete in that it solves for flows around immersed objects of arbitrary geometries, with non-trivial boundary conditions, and self-consistent in that it can retrieve the solution directly from the compressed encoding, i.e. without passing through the expensive dense-vector representation. This framework lays the foundation for a generation of more efficient solvers of real-life fluid problems.

The Navier-Stokes equations remain one of the biggest open problems in physics¹, with closed-form solutions known only in restricted cases. This has given rise to the field of Computational Fluid Dynamics (CFD)². Most CFD methods rely on a problem-specific discretization of the spatial domain, a *mesh*. There, the velocity and pressure fields are represented by vectors whose dimension is given by the mesh size. However, the precision required to capture relevant dynamics often translates into prohibitively high dimensions. For instance, for an accurate description of turbulent flows, the ratio between the largest and the smallest length scale in the mesh must grow with the Reynolds number (Re)^{3,4}. The resulting mesh can then have sizes that render the problem intractable for standard methods. This is a manifestation in CFD of the infamous *curse of dimensionality*, a major limitation of state-of-the-art mesh-based solvers.

A similar limitation arises in simulations of quantum systems, described by vector spaces exponentially large in the number N of particles^{5,6}. There, sophisticated tensor-network techniques^{7–9} have been developed to tackle the problem under the assumption of low entanglement, i.e. non-factorability of the quantum state¹⁰. The best-known example is matrix-product states (MPSs)^{11,12}, also known as tensor trains¹³. These represent 2^N -dimensional quantum states as a set of $2N$ matrices whose maximal size (called *bond dimension*) depends on the amount of entanglement. For instance, low-entangled states of 1D systems can be exponentially compressed by MPSs^{5,7–9}. In fact, MPSs and their tensor-network

extensions provide the state-of-the-art framework for simulating complex quantum dynamics^{14,15}.

In view of this, MPSs have been applied to a variety of other high-dimensional problems^{16–22}. In particular, turbulent fluids in simple geometries have been observed to also admit an efficient MPS description^{23–26}. The rationale is that the energy cascade mechanism³, whereby energy transfer takes place only between adjacent spatial scales, may play a similar role to local interactions in 1D quantum systems. This may open a new arena for CFD solvers. Nevertheless, for this to happen, compression efficiency should be combined with versatility to describe immersed objects of diverse geometries, with non-trivial boundary conditions. Moreover, a practical solver should also allow for accessing the solution directly in the MPS encoding, without passing through the dense vector. Otherwise, potential speed-ups in solving the problem may be lost at evaluating the solution. It is an open question whether these requirements can be harmoniously met and, most importantly, if there are settings of practical relevance where this can be achieved with a moderate bond dimension.

Here, we answer these questions in the affirmative. We deliver an MPS-based toolbox for simulating incompressible fluids, with complexity logarithmic in the mesh size and polynomial in the bond dimension. This is achieved via three major contributions: First, approximate MPS masks to encode object shapes into MPSs of low bond dimension. Second, a native time-integration scheme incorporating the masks as built-in features.

¹Quantum Research Center, Technology Innovation Institute, Abu Dhabi, UAE. ²Department of Mathematics, ETH Zurich, Zurich, Switzerland. ³Institute for Quantum-Inspired and Quantum Optimization, Hamburg University of Technology, Hamburg, Germany. ⁴Propulsion and Space Research Center, Technology Innovation Institute, Abu Dhabi, UAE. ⁵Federal University of Rio de Janeiro, Caixa Postal 652, Rio de Janeiro RJ 21941-972, Brazil.

✉ e-mail: raghavendra.peddinti@tii.ae

This is based on finite-differences as in refs. 23,24, but it operates with open boundary conditions, as in ref. 25, crucial to treat immersed objects without very large spatial domains. Additionally, the proposed scheme embeds open boundary conditions into a matrix-product differential operator. Third, an MPS solution oracle for querying the solution directly from the MPS, bypassing dense-vector evaluations. This features two main operation modes: coarse-grained evaluation and pixel sampling. In particular, the latter generates random points weighted by the squared magnitude of a target property (vorticity, pressure, etc.), mimicking the measurement statistics on a quantum state. This is ideal for Monte-Carlo simulations.

We validate the framework by reproducing text-book regimes of flows around a cylinder and by comparing our solutions for flows around a square, and even a collection of squares, with those of the popular commercial solver Ansys Fluent. To our knowledge, this is the first MPS algorithm for differential-equation systems validated to such extents. Finally, we venture into flows around a real-life airfoil from the NACA data set. MPSs of 18,800 parameters are enough for an accurate simulation with a mesh of $2^{19} = 524288$ pixels, a significant 27.9X compression. Among others, we use pixel sampling to randomly explore the regions of highest vorticity around the wing, without evaluating a single component of the 2^{19} -dimensional vorticity vector field itself. These tools may be relevant for aero- and hydrodynamic design optimizations.

Results

MPS framework for CFD

Our toolbox is summarized in Fig. 1. The details of the Navier–Stokes equations as well as of the MPS encoding of velocity and pressure fields are given in the Methods. Our presentation is restricted to the 2D case for simplicity, but all components of the framework can be straightforwardly generalized to the 3D case.

Approximate MPS masks. If a rigid body is immersed in a fluid at a fixed position, the velocity field must be zero both inside the object and on its boundary (no-slip condition). This is a crucial requirement that, to our knowledge, has not been addressed in the literature of quantum-inspired solvers. To incorporate it, we introduce the notion of *approximate MPS masks* (see Fig. 1b). These are MPSs encoding functions that approximate the target indicator function θ of the object – i.e. the function equal to zero within the object and to one outside it – up to arbitrary, tunable precision. We apply these masks on the MPS representing the field by simple element-wise multiplication.

To build such MPSs, we first consider a scalar function $q(x)$, with $x \in \mathcal{R}^D$, such that it *i*) vanishes on the boundary of the D -dimensional object in question, *ii*) takes negative values in the interior of the object, and *iii*) tends to infinity far away from the boundary. For example, for a circular object of unit radius in $D = 2$, $q(x, y) = x^2 + y^2 - 1$ does the job. Using such

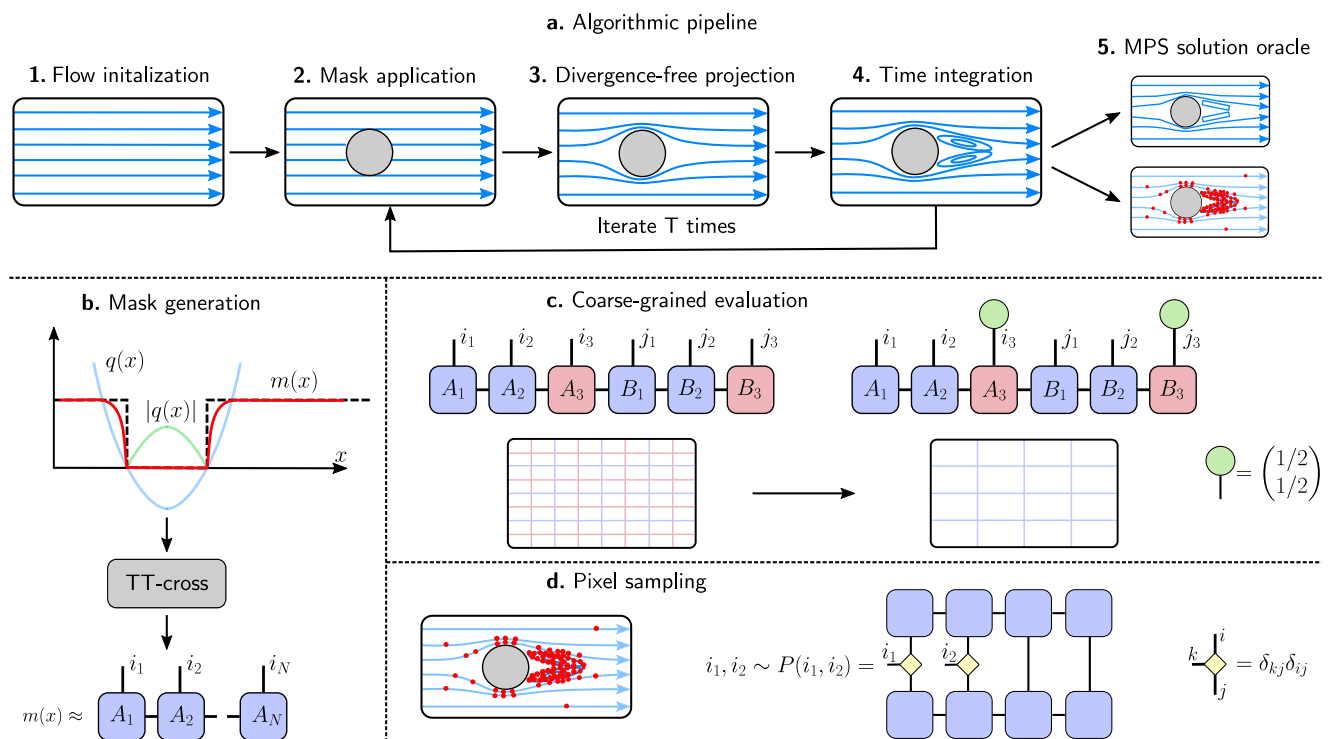
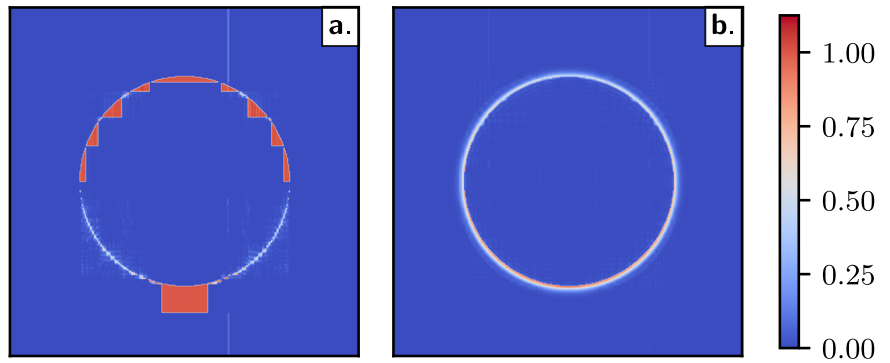


Fig. 1 | Schematics of the framework. **a** Main steps of the solver, featuring the paradigmatic steady flow around a cylinder. The first step is the flow initialization with constant velocity. The second one is the Matrix Product State (MPS) mask application, which nullifies the velocity field inside the object. This produces an artificial field that is not divergence-free. To correct this, in the third step, we project the field back onto the divergence-free manifold. The fourth step is the time integration over a small evolution time, using an explicit Euler method. The last three steps are then iterated T times to get the final MPS field at time T . Finally, the fifth step is the MPS solution oracle. The two boxes there represent the main operation modes: coarse-grained evaluation and pixel sampling. **b** Construction of approximate MPS masks (for a 1D object for clarity). Given a smooth function $q(x)$ (in light blue, its absolute value in green) that vanishes on the boundary of the object, we define $m(x) = 1 - e^{-\alpha(q(x)+|q(x)|)}$ (red solid). This approximates the object's

indicator function $\theta(x)$ (black dashed) up to a tunable small error (through α). Next, we apply the TT-cross algorithm to produce an MPS approximation to $m(x)$ from a few queries of it. **c** Coarse-grained evaluation: a 4-bit MPS, encoding an average over the smallest (pink) spatial scale, is obtained directly from a 6-bit MPS through simple contractions with constant single-site gadget tensors in green. **d** Pixel sampling: mesh cells (red points) are randomly chosen with probabilities proportional to the squared vorticity field at those points. The key tool are the marginals of the joint probability distributions over the binary representation of the cell positions. These allow one to sample the entire string bit by bit and can be efficiently obtained directly from the MPS. In the figure, a 2-bit marginal is obtained through simple single-site contractions of two copies of 4-bit MPS and the gadget tensors in yellow. Neither operation mode of the oracle requires field evaluations in the dense-vector representation (see Section MPS solution oracle).

Fig. 2 | Reconstruction error of the MPS mask. The figure shows the absolute error between the exact indicator function $\theta(\mathbf{x})$ and the MPS masks obtained from TT-cross approximation of (a) $\theta(\mathbf{x})$ and (b) $m(\mathbf{x})$. The domain discretization is $N_x = N_y = 11$ bits both horizontally and vertically (we show only a 6X close-up of the object); and the bond dimension is $\chi = 30$. Clearly, the deviations seen in panel a are much larger, not only close to the object's boundary but also far from it (note the thin vertical lines). That is, the intermediate step through m leads to masks of significantly higher accuracy. This is crucial for the solver, which applies the mask at every time iteration.



boundary functions, we define the smooth function

$$m(\mathbf{x}) := 1 - e^{-\alpha(q(\mathbf{x})+|q(\mathbf{x})|)}, \quad (1)$$

with $\alpha > 0$. By construction, m approximates θ increasingly better for growing values of α . In particular, the l_2 distance between $m(\mathbf{x})$ and $\theta(\mathbf{x})$ decreases with α as $1/\sqrt{\alpha}$ (see Supplementary Note 1).

The next step is to obtain an MPS representation of $m(\mathbf{x})$. If $m(\mathbf{x})$ consists of elementary functions, the MPSs can be constructed analytically^{27,28}. However, in general, one must resort to numerical approximations. In this work, we use the standard TT-cross algorithm²⁹ to find an MPS approximation of $m(\mathbf{x})$. This approximation can be found, up to a controllable l_2 distance ϵ , from few evaluations of the function. TT-cross is a classical tensor sketching algorithm, and more details can be found in Supplementary Note 1. We observe that $\epsilon = 10^{-3}$ is enough to generate high-quality masks of the cylinder up to meshes of 2^{30} cells (see Supplementary Note 1). The 30-bit MPS masks generated there have $\chi = 30$ and a total of 24,200 parameters, corresponding to a compression factor of more than 44,360. Our observations consistently show that the TT-cross approximation directly on the discontinuous function θ produces significantly worse masks than via the intermediate approximation m . This is shown in full-resolution in Fig. 2 for a 22-bit mesh. (The 30-bit case is too large for full resolution evaluation, see Supplementary Note 1.)

Time evolution. We discretize the differential operators in Eqs. (2) using finite differences with an 8-th order stencil³⁰. We use open boundary conditions (BCs), necessary to treat immersed objects using limited spatial domains. More precisely, we apply inlet-outlet BCs³¹ at the left- and right-hand edges, where the fluid enters and exits the domain, respectively; and use closed BCs at the upper and lower edges. We note that, in ref. 25, inlet-outlet BCs are enforced via application of a mask. Instead, we embed them directly into the differential operator. This operator can in turn be analytically recast as a matrix product operator (MPO), as shown in refs. 28,32,33 for simpler BCs. As a result, we obtain differential MPOs with inlet-outlet BCs as built-in features, with low bond dimensions (see Supplementary Note 2). With the differential MPOs in place, we can perform the time evolution.

The first step (see Fig. 1a) is to initialize the velocity MPSs [see Eq. (3)] as a constant vector field (flow far away from the object). Next, we loop T times over steps 2–4, with T the desired evolution time. Step 2 is the object mask application, described above. Steps 3 and 4 consist respectively of projecting the resulting MPS velocity field onto the divergence-free manifold, defined by Eq. (2b), and integrating over a small time step Δt . We implement these two steps with Chorin's projection method^{34,35}. That is, we first solve for the pressure from a Poisson equation, and use that pressure to project the velocity back onto the divergence-free manifold, enforcing the incompressibility condition (see Supplementary Note 2 for details). Then, we solve for the velocity at the next time instant by integrating the momentum equation [Eq. (2a)] without its pressure term. This iterative splitting procedure is equivalent to directly solving Eqs. (2) (see

Supplementary Note 2). The resulting velocity field satisfies neither the no-slip nor the incompressibility conditions, but this is corrected for in (steps 2 and 3 of) the next iteration (and, after the last iteration, we run one extra round of steps 2 and 3). The fifth and final step is to retrieve the solution from its MPS encoding, explained in Sec. MPS solution oracle.

The Poisson equation is the most computationally-expensive task in each iteration (see Section Computational complexity analysis). We tackle this with a DMRG-type algorithm that solves an equivalent linear system³⁶. As for the time integration, we use the Euler explicit time stepping, with time and spatial steps chosen according to known numerical-stability criteria (see Supplementary Note 2). Finally, every operation on an MPS typically increases its bond dimension. We keep it under a chosen threshold by periodic truncation (see Supplementary Note 3).

MPS solution oracle. Our solver outputs the solution in the MPS form, requiring up to exponentially fewer parameters than the dense-vector representation. This is specially relevant for turbulent flows (large Re), where the separation between the Kolmogorov scale (η) and integral scale (l) grows with Reynolds number, given by $\eta/l \sim Re^{-3/4}$ ³⁴. For fields in 3D, this means a mesh size which scales as $Re^{9/4}$. This raises the question of how to retrieve the solution without mapping the MPS to the dense vector, which can eliminate potential speed-ups gained in solving the problem. We propose two methods to extract the main features of the solution directly from the MPS: *coarse-grained evaluation* and *pixel sampling*.

Coarse-grained evaluation (Fig. 1c) produces an MPS with fewer spatial scales, encoding an averaged version of the solution. The averaging is done directly on the MPSs in a highly efficient manner: one simply contracts the physical indices of the fine scales one wishes to average out with single-site constant gadget tensors (in light-green in the figure). To evaluate the averaged field in full resolution, one then maps the obtained, smaller MPS to its corresponding dense vector, contracting the virtual indices between all $N - 1$ nearest-neighbors. Clearly, the finest-scales details are lost in this operation mode. A convenient, complementary alternative is pixel sampling.

Pixel sampling (Fig. 1d) is an exact sampling method that generates random points (x_i, y_j) chosen with probability $P(\mathbf{i}, \mathbf{j}) = |v_{\mathbf{i}, \mathbf{j}}|^2 / \|\mathbf{v}\|_2$, where $\|\mathbf{v}\|_2$ is the l_2 -norm of \mathbf{v} . Next, one can efficiently query the MPS to assess the encoded solution at the sampled points. This reproduces the measurement statistics of a quantum state that encodes \mathbf{v} in its amplitudes, hence being potentially relevant also to future quantum solvers³⁷. We note that related sampling methods have been used for training mesh-free neural-network solvers³⁸. However, there, the points are sampled uniformly, whereas here according to their relevance to the field. Our approach relies on standard techniques for l_2 -norm sampling physical-index values from an MPS^{7,8}. The key ingredient is the marginals of $P(\mathbf{i}, \mathbf{j})$, which can be computed by simple single-site tensor contractions on two copies of the MPS (with the gadget tensors in yellow in the figure). All N marginals are calculated in this fashion, without a single evaluation of \mathbf{v} at any point (\mathbf{i}, \mathbf{j}) . With the marginals, the conditional distribution for each bit given the previous ones is obtained,

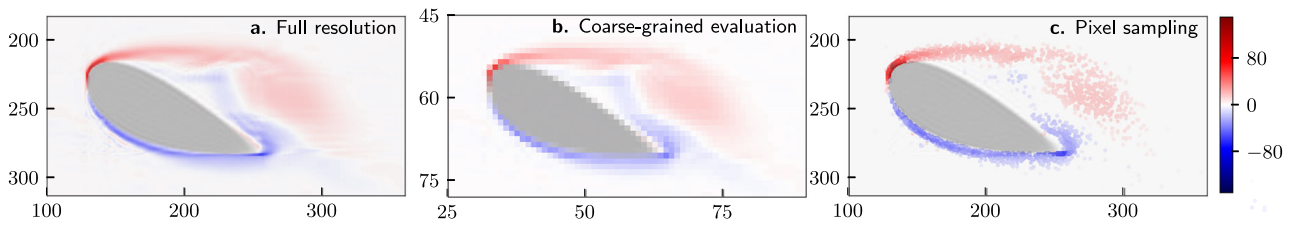


Fig. 3 | Vorticity field around a NACA 0040 airfoil, for different evaluation modes of the MPS solution oracle. The mesh size is $2^{19} = 524,288$, the number of time steps $T = 5000$, and Reynolds number $Re = 750$. We compress the corresponding fields into 19-bit MPSs of 18,800 parameters, a 27.9X compression. **a** Vorticity field in full resolution. This requires mapping the MPS to the dense-

vector representation. **b** Coarse-grained evaluation, with the two smallest scales in both x and y directions averaged out. The resulting 15-bit MPS has 16 times fewer parameters than the 19-bit MPS solution. **c** Pixel sampling, featuring 2500 pixels randomly selected according to their squared vorticity. No evaluation of the vorticity field itself is required for that.

allowing one to sample the entire bit string. Importantly, the technique applies to any MPS^{39,40}. For instance, one can obtain the MPS encoding the vorticity field from that of the velocity.

In Fig. 3 below, we showcase the real-life applicability of coarse-grained evaluation and pixel sampling for assessing the vorticity field around a wing in a mesh of 2^{19} cells. Moreover, in Supplementary Note 1, these evaluation modes allow us to benchmark the MPS masks generated for a mesh of 2^{30} cells. The hardware used for that is a standard Intel i9 CPU with 16 GB RAM, where such high-dimensional benchmarks would be impossible using dense vectors.

Finally, these oracle modes can optionally be combined by *close-ups*, i.e. zoomed-in evaluations of specifically chosen subdomains. Close-ups are obtained fixing a number of large-scale bits of an MPS solution which results in a smaller MPS on the remaining bits, that corresponds to the solution on the target subdomain. Depending on the use case, one can perform either full-resolution evaluation, coarse-grained evaluation or pixel sampling of the smaller MPS. For example, one can extend pixel sampling to sample densely from several subdomains using close-ups.

Computational complexity analysis. We perform a full time-complexity analysis of the solver and its components. Table 1 shows the asymptotic (worst-case) upper bounds as well as the numerically observed runtimes, per subroutine. For the tasks within the time integration loop, the estimates are per time iteration. The complexity of the mask generation is due to the TT-cross approximation²⁹. Those of the other tasks are dominated by their corresponding tensor-network operations (contractions and truncation of χ)^{7,39}. As for the numerical estimates, they correspond to an average over ten time iterations (see Supplementary Note 4). Note that the projection onto the divergence-free manifold is the most expensive subroutine, with a worst-case

performance $\mathcal{O}(N\chi^6)$. The latter is dominated by a sub-task involving N linear systems of size $4\chi^2 \times 4\chi^2$, which we solve exactly since χ is low in our cases. However, this can also be solved approximately via variational approaches (see Supplementary Note 4), which render the complexity $\mathcal{O}(N\chi^4)$ and are therefore a convenient alternative for high χ ²³. Finally, the last row shows the total numerically observed runtime per time iteration. The scaling there is a conservative estimate from numerical fits to the runtimes obtained for fixed N (between 15 and 23) as function of χ (varying from 20 to 50). Those fits are consistent with scalings $\alpha N\chi^{4.1}$, with $10^{-7} \leq \alpha \leq 10^{-6}$ (see Supplementary Note 4 for details). This is significantly below the worst-case bound $\mathcal{O}(N\chi^6)$.

Numerical results

To showcase the versatility of our framework with three classes of immersed objects. First, we show how our simulations reproduce well-known regimes of flows around a cylinder at different Reynolds numbers. Second, we benchmark our results for flows around squared cylinders against those of the popular commercial solver Ansys Fluent. Third, we tackle the challenging case of flows around an realistic airfoil, where we also showcase the different operation modes of the MPS solution oracle.

The flow around a circular cylinder is one of the paradigmatic problems in fluid dynamics⁴¹. We simulate the three characteristic dynamics of laminar flow at different $Re \leq 150$ (see Fig. 4). We solve for the velocity and pressure fields on a mesh with $N_x = 8$ and $N_y = 7$, encoded in a 15-bit MPS with $\chi = 30$ (see Supplementary Table S1 for details). The solutions obtained reproduce the textbook sub-regimes as Re increases: from a flow without downstream circulation to the well-known Kármán vortex street⁴².

Next, we simulate flows around a squared cylinder and a collection of four squared cylinders at $Re = 127$ and $Re = 141$, respectively. Both simulations are performed on a mesh with $N_x = 8$ and $N_y = 7$ using MPSs of $\chi = 30$. As shown in Fig. 5, we compare our solutions with the ones obtained using Ansys Fluent. We note that comparison between conceptually different solvers are meaningful only in the fully developed phase of the flow, after the initial transient behavior. The evolution times used for Fig. 5 were carefully chosen as to guarantee this condition (see Supplementary Note 5). This rules out possible mismatches in the transient regimes due to different numerical approaches. In addition, in Supplementary Note 5, we also report simulations for the squared cylinder at $Re = 630$ and $Re = 2230$.

Finally, we consider the simulation of flows around a realistic airfoil in a mesh of $N_x = 10$ and $N_y = 9$, shown in Fig. 3. The flow inclination angle used is 22° , $Re = 750$, and $\chi = 45$. We take advantage of this complex geometry to showcase the different evaluation modes of the MPS solution oracle. As a specific example, we consider the vorticity field $\omega = \nabla \times \mathbf{v}$. We obtain its associated MPS directly from the velocity MPSs and the matrix-product form of ∇ . The vorticity MPS is then taken as the solution oracle for the plots in Fig. 3. We stress that neither coarse-grained evaluation nor pixel sampling require the vector representation of ω , only its MPS.

The observed versatility for flows around realistic geometries, together with the built-in ability to access the solution directly from the MPS, renders

Table 1 | Time complexities

Algorithmic task	Complexity	Runtime %
Mask generation	$\mathcal{O}(N\chi^3)$	offline
Mask application	$\mathcal{O}(\chi^6)$	2.0
Divergence-free projection	$\mathcal{O}(N\chi^6)$	80.1
Euler time stepping	$\mathcal{O}(\chi^6)$	17.9
Coarse-grained evaluation	$\mathcal{O}(N\chi^3)$	offline
Pixel sampling (per pixel)	$\mathcal{O}(\chi^3)$	offline
Numerically observed total runtime $\approx N\chi^{4.1}$		

The first column shows the main subroutines. Their corresponding asymptotic worst-case time complexity scaling and percentages of CPU runtime per time iteration observed are shown in the second and third columns, respectively. In the third column, we highlight, with bold text, the most expensive task of the pipeline. The numerical run times were measured for the flow around a squared cylinder, using a 15-bit MPS encoding with bond dimension $\chi = 30$. We refer by offline to the tasks performed outside the time evolution loop. Finally, the last row shows the scaling of the numerical runtime observed for an entire time iteration (see Supplementary Note 4 in the SI for more details).

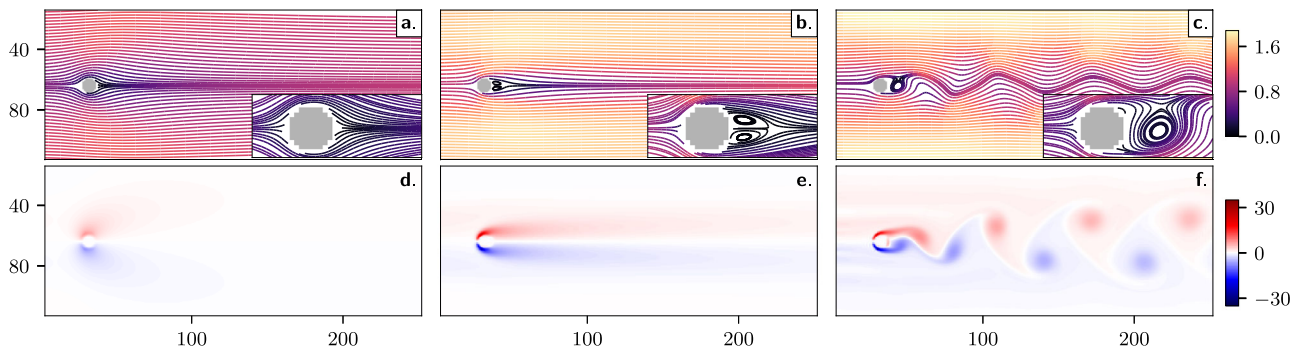
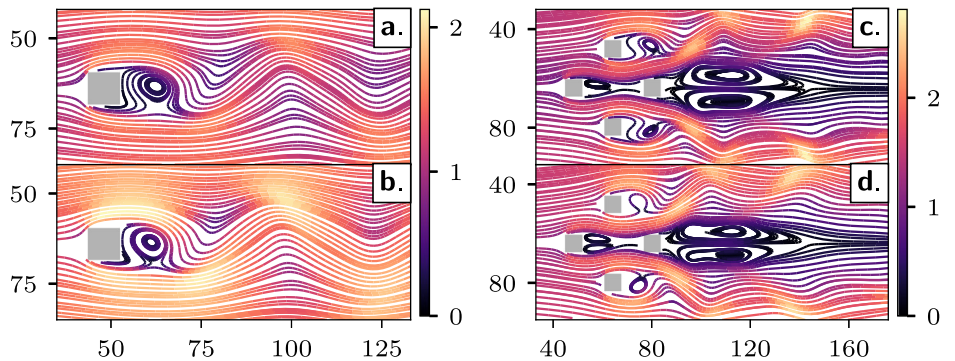


Fig. 4 | Flow around a circular cylinder at various Reynolds numbers. Each column shows one of the three paradigmatic phases of a laminar flow around a cylinder⁴². **a–c** Streamlines, originating at the inlet (left side), are shown. The color code indicates the magnitude of velocity. The insets show the features of the flow near the boundary of the object. **d–f** Vorticity field $\omega = \nabla \times \mathbf{v}$ is shown. The first column corresponds to Reynolds number $Re = 1.7$: in panel **a**, we observe that the streamlines do not detach from the edge of the cylinder. In turn, in panel **d**, no vorticity is observed behind the object. These are the signatures of a potential flow, where \mathbf{v} is given by the gradient of a scalar field. The second column corresponds to

$Re = 25$. As seen in panel **b**, streamlines detach from the object’s boundary. In turn, in panel **e**, a pair of counter-rotating vortices appears behind the object. These are the characteristics of a steady laminar flow. The third column corresponds to $Re = 125$. In this regime, the flow becomes unsteady and develops the well-known Kármán vortex street⁴¹. This can be observed in panels **c** and **f**, where clear oscillations in the streamlines and in the vortices behind the object are shown. A 15-bit MPS with $\chi = 30$ is used for all cases. In all of them, the simulated flow reproduces the expected behavior.

Fig. 5 | Comparison with Ansys Fluent. Panels **a** and **b** show the flows around a square cross-section for Reynolds number $Re = 127$ and at time step $T = 19,900$, obtained by our solver (in **a**) and by the commercial solver Ansys Fluent (in **b**). Panels **c** and **d** show a similar comparison for a collection of four squares, for $Re = 141$ and at $T = 2500$, where panel **c** corresponds to our solver and panel **d** corresponds to Ansys Fluent. For both cases, we have MPSs with $N_x = 8$ and $N_y = 7$ and $\chi = 30$. Importantly, the values of T were chosen high enough for the flows to be in the fully-developed regime (see Supplementary Note 5). In both cases the agreement between our solver and Ansys Fluent is excellent.



our framework potentially interesting to hydro-dynamic design problems. In particular, pixel sampling is well-suited for Monte-Carlo estimations of objective functions that may be relevant to industrial optimizations.

Discussion

We have presented a quantum-inspired framework for CFD based on matrix-product states (MPSs). Our solver supersedes previous works in that it incorporates immersed objects of diverse geometries, with non-trivial boundary conditions, and can retrieve the solution directly from the compressed MPS encoding, i.e. without passing through the expensive dense-vector representation. For a mesh of size 2^N , both memory and runtime scale linearly in N and polynomially in the bond dimension χ . The latter grows with the complexity of the geometry in question. Nevertheless, we showed that our toolbox can handle highly non-trivial geometries using not only moderate χ but also very few parameters. For instance, the most challenging flow simulation considered is for a wing airfoil in a mesh of 2^{19} pixels. Our machinery accurately captures the flows with MPSs of $N = 19$ bits, $\chi = 45$, and a total of 18,800 parameters (27.9X compression). In turn, it can also accurately represent a cylinder in a mesh of 2^{30} cells with an MPS of $N = 30$, $\chi = 30$, and a total of 24,200 parameters: a compression of more than 44,360X.

A core technical contribution to enable such high efficiencies is the approximate MPS masks. In fact, we have provided a versatile mask generation recipe exploiting standard tensor-network tools such as the TT-cross algorithm. Another key ingredient is the alternative methods we have proposed to query the solution directly from its MPS: coarse-grained

evaluation and pixel sampling. In particular, pixel sampling explores random mesh points distributed according to the square of a chosen property, such as for instance vorticity or pressure (or, through an equation of state, also temperature). This mimics the measurement statistics of a quantum state encoding the property in its amplitudes, hence being relevant also to future quantum solvers³⁷. It does so by efficiently computing marginals of the target distribution via local tensor contractions directly on the MPS, without evaluating the property itself at any mesh point. Pixel sampling is ideal for Monte-Carlo simulations, relevant to optimizations via stochastic gradient descent, e.g. This can be interesting to aero- or hydro-dynamic design problems as well as to training neural-network models.

Our work offers several directions for future research. In particular, the extension to the 3D case will enable probing the framework in truly turbulent regimes. This will also require in-depth studies of the dependence of the bond dimension with the evolution time. We anticipate that further optimization of the tensor-network Ansatz will play a crucial role for that, as well as extensions to non-uniform meshes. Another possibility is the exploration of tensor networks with other natural function bases for turbulent phenomena, such as Fourier⁴³ or wavelets⁴⁴. In turn, an interesting opportunity is the application of our framework to other partial differential equations^{21,22,45}. This may be combined with extensions to finite elements or finite volumes, which can in principle also be formulated with tensor networks²⁵. Moreover, our method may be relevant to mesh-free solvers too. For instance, pixel sampling on small meshes could be explored to speed-up the training of mesh-free neural-network schemes³⁸. Finally, a further prospect could be to combine the framework with quantum sub-routines, in

view of hybrid classical-quantum solvers. For example, our main computational bottleneck is the Poisson equation, and future quantum linear solvers may offer significant speed-ups for that^{37,46,47}.

All in all, our findings open a playground with potential to build radically more efficient solvers of real-life fluid dynamics problems as well as other high-dimensional partial differential equation systems, with far-reaching research and development implications.

Methods

Incompressible Navier–Stokes equations

We solve the incompressible Navier–Stokes equations in the absence of external forces (for simplicity):

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{v}, \quad (2a)$$

$$\nabla \cdot \mathbf{v} = 0, \quad (2b)$$

where $\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$ and $p = p(\mathbf{x}, t)$ are respectively the velocity and pressure fields at position \mathbf{x} and time t , ρ is the density, and ν the kinematic viscosity. Eqs. (2a) and (2b) follow respectively from momentum and mass conservations^{4,48,49}. They give rise to a variety of behaviors, ranging from laminar to turbulent flows. The flow regime is determined by the Reynolds number, $\text{Re} = \frac{UL}{\nu}$, where U and L are characteristic velocity and length scales of the problem⁴. This dimensionless parameter quantifies the ratio between the inertial $[(\mathbf{v} \cdot \nabla) \mathbf{v}]$ and dissipative $[\nu \nabla^2 \mathbf{v}]$ force terms. At high Re , the nonlinear inertial term dominates and the flow is highly chaotic and turbulent. Instead, at low Re , the dissipative term dominates and the flow is stable and laminar.

Field encoding with matrix product states

We encode scalar functions – such as the velocity components $v_x(\mathbf{x})$ and $v_y(\mathbf{x})$, or the pressure $p(\mathbf{x})$ – into matrix-product states (MPSs)^{7–9}. We discretize the 2D domain into a mesh of 2^N points, specified by $N = N_x + N_y$ bits. There, we represent a continuous function $v(x, y)$ by a vector of elements $v_{\mathbf{i}, \mathbf{j}} := v(x_{\mathbf{i}}, y_{\mathbf{j}})$, where the strings $\mathbf{i} := (i_1, i_2, \dots, i_{N_x})$ and $\mathbf{j} := (j_1, j_2, \dots, j_{N_y})$ give the binary representation of $x_{\mathbf{i}}$ and $y_{\mathbf{j}}$, respectively. Then, we write each $v_{\mathbf{i}, \mathbf{j}}$ as a product of N matrices:

$$v_{\mathbf{i}, \mathbf{j}} = A_1^{(i_1)} A_2^{(i_2)} \dots A_{N_x}^{(i_{N_x})} B_1^{(j_1)} B_2^{(j_2)} \dots B_{N_y}^{(j_{N_y})}. \quad (3)$$

This is the MPS form (see also Fig. 1c). The indices i_k and j_k are referred to as *physical indices* of the matrices A_j and B_j , respectively. Note that, the first N_x matrices correspond to $x_{\mathbf{i}}$ and the remaining N_y ones to $y_{\mathbf{j}}$, as in refs. 21, 24. However, other arrangements are possible^{16,23,50}. The bond dimension χ is defined as the maximum dimension over all $2N$ matrices used. Importantly, the total number of parameters is at most $2N\chi^2$. Hence, if χ constant, the MPS provides an exponentially compressed representation of the 2^N -dimensional vector. Moreover, instead of fixing all virtual dimensions at χ , we dynamically adapt each site's dimension to the amount of inter-scale correlations. This typically reduces the number of parameters to well below the bound $2N\chi^2$.

Data availability

The data sets generated during the current study are available from the corresponding author upon request.

Code availability

The software used in this work can be found at the [accompanying CodeOcean capsule](#).

Received: 18 December 2023; Accepted: 9 April 2024;

Published online: 27 April 2024

References

1. Fefferman, C. L. Existence and smoothness of the Navier–Stokes equation. *Millennium Prize Probl.* **57**, 67 (2000).
2. Orszag, S. A. & Patterson Jr, G. Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Phys. Rev. Lett.* **28**, 76 (1972).
3. Kolmogorov, A. N. The local structure of turbulence in incompressible viscous fluid for very large Reynolds number. *Dokl. Akad. Nauk. SSSR*, **30**, 301–303 (1941).
4. Pope, S. B. *Turbulent Flows* (Cambridge University Press, New York, 2000).
5. Eisert, J., Cramer, M. & Plenio, M. B. Colloquium: Area laws for the entanglement entropy. *Rev. Mod. Phys.* **82**, 277 (2010).
6. Poulin, D., Qarry, A., Somma, R. & Verstraete, F. Quantum simulation of time-dependent hamiltonians and the convenient illusion of Hilbert space. *Phys. Rev. Lett.* **106**, 170501 (2011).
7. Schollwöck, U. The density-matrix renormalization group in the age of matrix product states. *Ann. Phys.* **326**, 96–192 (2011).
8. Orús, R. A practical introduction to tensor networks: matrix product states and projected entangled pair states. *Ann. Phys.* **349**, 117–158 (2014).
9. Verstraete, F., Murg, V. & Cirac, J. I. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Adv. Phys.* **57**, 143–224 (2008).
10. Horodecki, R., Horodecki, P., Horodecki, M. & Horodecki, K. Quantum entanglement. *Rev. Mod. Phys.* **81**, 865 (2009).
11. White, S. R. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.* **69**, 2863–2866 (1992).
12. Vidal, G. Efficient classical simulation of slightly entangled quantum computations. *Phys. Rev. Lett.* **91**, 147902 (2003).
13. Oseledets, I. V. Tensor-train decomposition. *SIAM J. Sci. Comput.* **33**, 2295–2317 (2011).
14. Zhou, Y., Stoudenmire, E. M. & Waintal, X. What limits the simulation of quantum computers? *Phys. Rev. X* **10**, 041038 (2020).
15. Tindall, J., Fishman, M., Stoudenmire, E. M. & Sels, D. Efficient Tensor Network Simulation of IBM's Eagle Kicked Ising Experiment. *PRX Quantum* **5**, 010308 (2024).
16. Latorre, J. Image compression and entanglement. Preprint at <https://arxiv.org/abs/quant-ph/0510031> (2005).
17. Glasser, I., Sweke, R., Pancotti, N., Eisert, J. & Cirac, I. Expressive power of tensor-network factorizations for probabilistic modeling. *Adv. Neural Inf. Process. Syst.* **32**, 134 (2019).
18. Torlai, G. et al. Quantum process tomography with unsupervised learning and tensor networks. *Nat. Commun.* **14**, 2858 (2023).
19. Kurmapu, M. K. et al. Reconstructing Complex States of a 20-Qubit Quantum Simulator. *PRX Quantum* **4** (2023).
20. Kastoryano, M. & Pancotti, N. A highly efficient tensor network algorithm for multi-asset fourier options pricing. Preprint at <https://arxiv.org/abs/2203.02804> (2022).
21. Shinaoka, H. et al. Multiscale space-time ansatz for correlation functions of quantum systems based on Quantics Tensor Trains. *Phys. Rev. X* **13**, 021015 (2023).
22. Truong, D. P. et al. Tensor networks for solving the time-independent Boltzmann neutron transport equation. *J. Comput. Phys.* **507**, 112943 (2024).
23. Gourianov, N. et al. A quantum-inspired approach to exploit turbulence structures. *Nat. Comput. Sci.* **2**, 30–37 (2022).
24. Kiffner, M. & Jaksch, D. Tensor network reduced order models for wall-bounded flows. *Phys. Rev. Fluids* **8**, 124101 (2023).
25. Kornev, E. et al. Numerical solution of the incompressible Navier–Stokes equations for chemical mixers via quantum-inspired Tensor Train Finite Element Method. Preprint at <https://arxiv.org/abs/2305.10784> (2023).
26. Gourianov, N. *Exploiting the structure of turbulence with tensor networks*. Ph.D. thesis (University of Oxford, 2022).

27. Oseledets, I. V. Constructive representation of functions in low-rank tensor formats. *Constr. Approx.* **37**, 1–18 (2012).
28. García-Ripoll, J. J. Quantum-inspired algorithms for multivariate analysis: from interpolation to partial differential equations. *Quantum* **5**, 431 (2021).
29. Oseledets, I. & Tyrtshnikov, E. TT-cross approximation for multidimensional arrays. *Linear Algebra its Appl.* **432**, 70–88 (2010).
30. Fornberg, B. Generation of finite difference formulas on arbitrarily spaced grids. *Math. Comput.* **51**, 699–706 (1988).
31. Weinan, E. & Liu, J.-G. Projection method I: Convergence and numerical boundary layers. *SIAM J. Numer. Anal.* **32**, 1017–1057 (1995).
32. Oseledets, I. V. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM J. Matrix Anal. Appl.* **31**, 2130–2145 (2010).
33. Kazeev, V. A. & Khoromskij, B. N. Low-rank explicit QTT representation of the laplace operator and its inverse. *SIAM J. Matrix Anal. Appl.* **33**, 742–758 (2012).
34. Chorin, A. J. The numerical solution of the Navier-Stokes equations for an incompressible fluid. *Bull. Am. Math. Soc.* **73**, 928–931 (1967).
35. Chorin, A. J. Numerical solution of the Navier-Stokes equations. *Math. Comput.* **22**, 74–762 (1968).
36. Oseledets, I. V. & Dolgov, S. V. Solution of linear systems and matrix inversion in the TT-format. *SIAM J. Sci. Comput.* **34**, A2718–A2739 (2012).
37. Bagherimehrab, M., Nakajji, K., Wiebe, N. & Aspuru-Guzik, A. Fast quantum algorithm for differential equations. Preprint at <https://arxiv.org/abs/2306.11802> (2023).
38. Sirignano, J. & Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **375**, 1339–1364 (2018).
39. Ferris, A. J. & Vidal, G. Perfect sampling with unitary tensor networks. *Phys. Rev. B* **85**, 165146 (2012).
40. Han, Z.-Y., Wang, J., Fan, H., Wang, L. & Zhang, P. Unsupervised generative modeling using matrix product states. *Phys. Rev. X* **8**, 031012 (2018).
41. Kačmarčík, V. T. *Aerodynamics: Selected topics in the light of their historical development* (Dover Publications, 2004).
42. Feynman, R. P., Leighton, R. B. & Sands, M. *The Feynman lectures on physics; New millennium ed.* (Basic Books, 2010). Originally published 1963–1965.
43. Li, Z. et al. Fourier neural operator for parametric partial differential equations. Preprint at <https://arxiv.org/abs/2010.08895> (2021).
44. Farge, M. Wavelet transforms and their applications to turbulence. *Annu. Rev. Fluid Mech.* **24**, 395–458 (1992).
45. Lubasch, M., Moinier, P. & Jaksch, D. Multigrid renormalization. *J. Comput. Phys.* **372**, 587–602 (2018).
46. Tosta, A., de Lima Silva, T., Camilo, G. & Aolita, L. Randomized semi-quantum matrix processing. Preprint at <https://arxiv.org/abs/2307.11824> (2023).
47. Lubasch, M., Joo, J., Moinier, P., Kiffner, M. & Jaksch, D. Variational quantum algorithms for nonlinear problems. *Phys. Rev. A* **101**, 010301 (2020).
48. Monin, A. S. & Yaglom, A. M. *Statistical fluid mechanics*, volume I (Courier Corporation, 2007).
49. Monin, A. S. & Yaglom, A. M. *Statistical fluid mechanics*, volume II: mechanics of turbulence (Courier Corporation, 2013).
50. Ye, E. & Loureiro, N. F. G. Quantum-inspired method for solving the Vlasov-Poisson equations. *Phys. Rev. E* **106**, 035208 (2022).

Author contributions

E.T. and L.A. conceived the project. L.A. supervised its execution. R.P., S.P., and E.T. developed the tensor-network theoretical framework and software. A.M., P.L. and H.A. provided CFD theoretical support and numerical benchmarks. R.P., S.P., E.T. and L.A. wrote the manuscript. All authors participated in discussions and contributed with ideas.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s42005-024-01623-8>.

Correspondence and requests for materials should be addressed to Raghavendra Dheeraj Peddinti.

Peer review information *Communications Physics* thanks the anonymous reviewers for their contribution to the peer review of this work. A peer review file is available.

Reprints and permissions information is available at <http://www.nature.com/reprints>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2024