ASTER: LEARNING TENSOR-NETWORK STRUCTURE FOR PDE SOLVING WITH A DIFFERENTIABLE QUANTUM–CLASSICAL MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Tensor networks offer compact representations of high-dimensional fields and are increasingly used for solving partial differential equations (PDEs) at resolutions where conventional discretisations become computationally prohibitive. However, most tensor-network PDE methods rely on fixed structures chosen in advance, while existing structure-search methods primarily optimise compression of individual fields rather than PDE dynamics. Although quantum circuits provide flexible trainable samplers, their use for tensor-network structure discovery is further limited by the lack of efficient mechanisms for generating only valid network structures. We introduce ASTER, a hybrid quantum–classical model that jointly learns tensor-network structures and PDE dynamics. A trainable quantum sampler generates valid network topologies without rejection and integrates them into a differentiable tensor-network solver. ASTER$^+$ further incorporates rank adaptation into the discovered network structure. Across three evaluation settings, ASTER$^+$ matches the accuracy of spectral Runge–Kutta on six of seven benchmark PDEs while using 2.9–10$\times$ fewer stored scalars on five. ASTER-SOLVER achieves the lowest error at every evaluated resolution on eight of twelve two-dimensional PDEs against state-of-the-art neural PDE solvers, while using substantially fewer parameters. As a structure-search method, ASTER$^+$ also achieves lower storage at matched accuracy than existing approaches at the three highest resolutions.

1 INTRODUCTION

Partial differential equations (PDEs) are fundamental to scientific and engineering computing, but their numerical solution becomes increasingly expensive as the spatial resolution grows. Conventional discretisations represent the solution on dense grids, causing memory and computational costs to increase rapidly with resolution and spatial dimension. This limits the resolution achievable in many large-scale simulations and motivates compact representations that can exploit the underlying structure of the solution.

Tensor networks (TNs) provide one such representation. By reorganising a high-dimensional field into a sequence or hierarchy of smaller tensors, they can represent structured solutions using substantially fewer parameters than dense arrays. This compression has enabled tensor-network methods to address a range of PDEs and scientific computing problems (Oseledets, 2011; Khoromskij, 2011; Grasedyck, 2010; Gourianov et al., 2022; Kiffner & Jaksch, 2023; Ye & Loureiro, 2022; Peddinti et al., 2024; Dolgov et al., 2012; Bachmayr et al., 2016). TNs are also well suited to dynamical simulation because PDE evolution can be constrained to remain within the represented low-dimensional manifold, enabling time integration directly in the compressed representation (Koch & Lubich, 2007; Lubich et al., 2013; Lubich & Oseledets, 2014).

Despite these advantages, most tensor-network PDE solvers prescribe the network structure before the simulation begins. These approaches are so called **fixed-topology TN (TN-FT)**. Common choices include matrix-product structures, hierarchical tree structures, and fixed orderings of the spatial degrees of freedom (Oseledets, 2011; Dolgov et al., 2012; Grasedyck, 2010; Hackbusch & Kühn, 2009; Khoromskij, 2011; Ritter et al., 2024). Such choices may be effective for problems

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

with well-understood structure, but the most suitable representation can depend on the PDE and on how its solution evolves. A topology that compresses an individual snapshot well is not necessarily the most effective representation of the full solution trajectory. This motivates adaptive structure discovery as part of the PDE-solving process rather than as a separate preprocessing step.

Several **tensor-network structure-search (TN-SS)** methods have therefore been developed, including evolutionary optimisation (TNGA (Li & Sun, 2020)), local permutation search (TNLS (Li et al., 2022)), alternating enumeration (TnALE (Li et al., 2023)), and greedy tree construction (Ballani & Grasedyck, 2014; Hashemizadeh et al., 2020). These methods demonstrate that changing the network topology can improve representation efficiency. However, they generally formulate structure selection as a compression problem and modify a predefined topology through local structural operations. *They therefore do not directly learn the network structure together with the PDE dynamics being solved*. A related line of work adapts tensor ranks during time integration while retaining a fixed topology (Ceruti & Lubich, 2022; Ceruti et al., 2022; 2021; 2023). Rank adaptation controls the size and flexibility of the tensors, but does not address whether the underlying connectivity of the network is appropriate for the evolving solution. Thus, two complementary aspects of tensor-network adaptation remain largely separated: discovering the network topology and adapting its representation capacity during evolution.

A second challenge arises when quantum methods are considered for TN-SS. The number of possible network topologies grows extremely rapidly with problem size, making the strategy used to generate candidate structures critical to efficient search. Parameterised quantum circuits provide a flexible family of trainable probability distributions and have been investigated as generative models through Born machines and related quantum sampling approaches (Boixo et al., 2018; Bouland et al., 2019; Arute et al., 2019; Aaronson & Arkhipov, 2011). Their use in structure discovery could provide a trainable mechanism for exploring large discrete spaces of candidate topologies.

Quantum circuits naturally produce bitstrings, whereas TN-SS requires valid network topologies. A straightforward decoding rule can map some circuit outputs to invalid structures, requiring rejected samples to be regenerated. This increases sampling cost and alters the effective distribution because the accepted samples are conditioned on validity. To the best of our knowledge, existing TN-SS methods have not addressed this representation problem or used a quantum sampler to generate tensor-network topologies directly. This creates a second research gap: an efficient mechanism is needed to translate quantum samples into valid network structures without rejection.

We address these gaps with ASTER, short for Adaptive Structure TEnsor-network Representation, a quantum–classical framework for adaptive tensor-network PDE solving. ASTER jointly learns the network topology and PDE dynamics rather than fixing the representation in advance. A trainable quantum circuit samples candidate topologies, while a differentiable tensor-network solver evaluates their ability to represent the evolving solution. We further introduce two models, ASTER$^+$ and ASTER-SOLVER. First, ASTER$^+$ extends ASTER with rank adaptation, allowing the tensor-network representation to adjust both its topology and local representational capacity during integration. Second, ASTER-SOLVER is an end-to-end differentiable hybrid quantum–classical PDE solver that integrates the tensor-network structure search of ASTER$^+$ into the PDE-solving framework.

The main contributions are as follows:

- We introduce a quantum topology sampler that generates valid tensor-network tree structures without rejection (§3.2). The proposed decoder provides a direct mapping from quantum bitstrings to network structures, eliminating invalid samples and avoiding the associated sampling overhead and distributional distortion.
- We develop a differentiable tensor-network PDE solver that jointly optimises network structure and dynamics (§3.3). The PDE evolution is represented on the sampled tensor-network manifold, while a learned correction compensates for the approximation introduced by the projection.
- We evaluate the framework in three settings: numerical integration, data-driven PDE solving, and TN-SS. The results demonstrate that jointly adapting topology and rank can achieve competitive accuracy with substantially reduced representation cost.

In §4, we first evaluate ASTER$^+$ as a numerical integrator on seven PDEs with known solutions. It matches the accuracy of spectral RK4 on six of the seven problems while using 2.9 to $10\times$ fewer stored scalars on five of them. We then evaluate ASTER-SOLVER as a learned PDE solver on twelve two-dimensional problems. It achieves the lowest error at every evaluated resolution on eight prob-

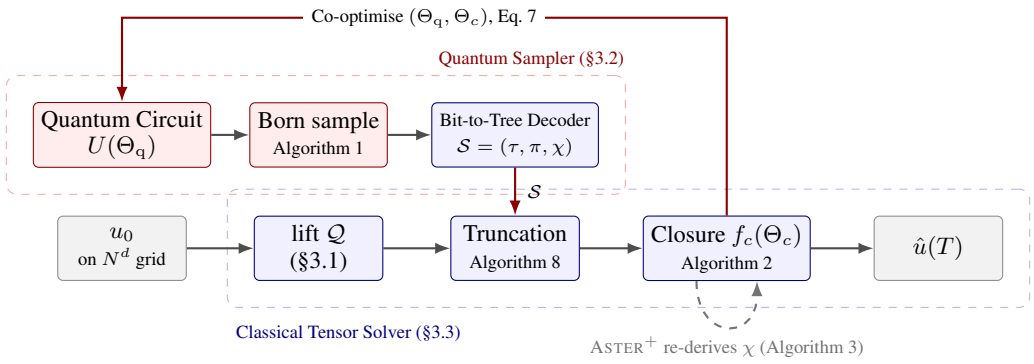


Figure 1: Overview of ASTER

lems compared with Fourier Neural Operator (FNO) (Li et al., 2020), DeepONet (Lu et al., 2021), and a latent neural ordinary differential equation model (Chen et al., 2018), while using substantially fewer parameters. Finally, as a structure-search method, ASTER$^+$ achieves lower storage at matched accuracy than TNGA, TNLS, and TnALE at the three highest evaluated resolutions.

2 Related works

Tensor networks for PDEs. The tensor-train (TT) (Oseledets, 2011) and hierarchical Tucker (HT) formats (Hackbusch & Kühn, 2009; Grasedyck, 2010) provide standard tree-structured tensor-network families, while quantics lifting (Khoromskij, 2011; Ritter et al., 2024) extends these representations to fields defined on fine grids. Time integration can be performed using dynamical low-rank approximation (Koch & Lubich, 2007; Lubich et al., 2013), with projector-splitting (Lubich & Oseledets, 2014) and time-dependent variational principle (TDVP) (Haegeman et al., 2016) integrators accounting for the curvature of the underlying tensor manifold.

Tensor-network structure search. Tensor-network topology selection has been formalised through evolutionary and local-search strategies. (Li & Sun, 2020) employ a genetic algorithm to search over tensor-ring topologies, while (Li et al., 2022) sample local neighbourhoods around the incumbent structure and (Li et al., 2023) alternate coordinate-wise exploration to reduce the number of evaluations. Related approaches construct tree-structured networks greedily (Ballani & Grasedyck, 2014; Hashemizadeh et al., 2020). However, these methods primarily target static tensor compression rather than time-dependent PDE integration, rely on local modifications from a manually specified initial topology, and perform the structure search entirely classically.

Quantum circuits as samplers. Shallow random quantum circuits provide a well-established setting for quantum sampling advantage, supported by both theoretical analyses (Boixo et al., 2018; Bouland et al., 2019; Aaronson & Arkhipov, 2011) and experimental demonstrations (Arute et al., 2019). Born machines make these distributions trainable through parameter-shift gradient estimators (Liu & Wang, 2018; Mitarai et al., 2018; Schuld et al., 2019; Wierichs et al., 2022), can capture correlations beyond those accessible to comparable classical generative models (Coyle et al., 2020; Gao et al., 2022), and have been studied alongside matrix-product-state (MPS) generative models (Han et al., 2018; Hibat-Allah et al., 2020). Those hardness results concern depths beyond the reach of matrix-product-state contraction; slightly entangled circuits are simulable in polynomial time (Vidal, 2003), and the configuration we run falls in that regime.

3 The proposed framework

We provide an overview of the proposed model in Figure 1.

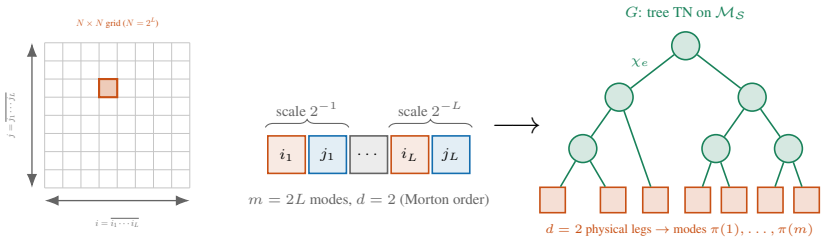


Figure 2: **Quantics lifting and the sampled tree.** The $N \times N$ field is isometrically reshaped into an order-$2L$ tensor whose modes encode the binary dyadic scales of the two spatial directions. A sampled structure $\mathcal{S}$ specifies the mode ordering π, tree topology τ, and bond dimensions χ.

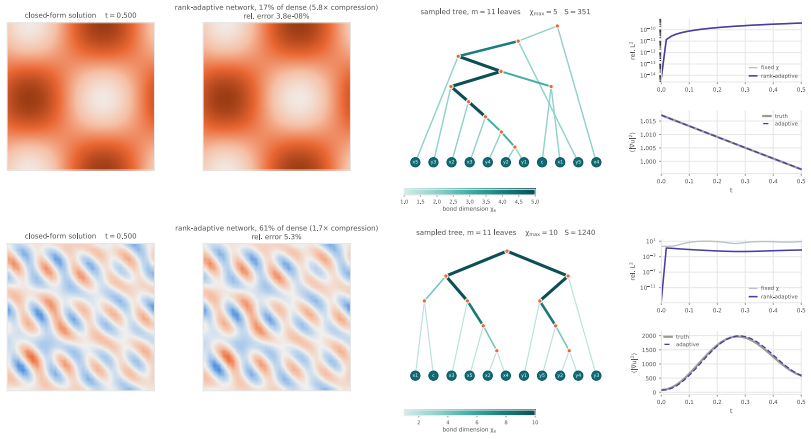


Figure 3: Adaptively discovered TN structures during PDE solving.

3.1 Preliminaries

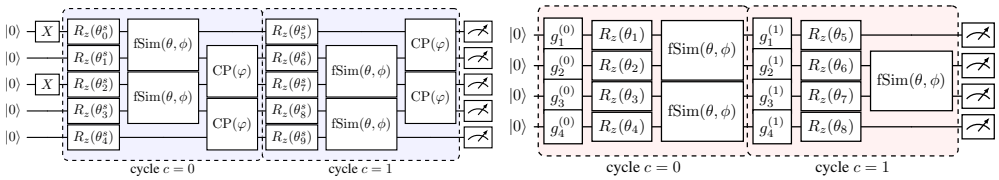
We show an illustration of using tensor networks to compress 2D field in Figure 2. Consider a PDE F and its field $\mathbf{u} : \mathbb{T}^d \to \mathbb{R}^{n_c}$ be sampled on a uniform periodic grid of $N = 2^L$ points per axis. We write each grid index in binary $i_k = \sum_{\ell=1}^{L} b_{k\ell} 2^{L-\ell}$. The quantics lifting $\mathcal{Q}$ reinterprets the array $u \in \mathbb{R}^{n_c \times N^d}$ as an order-$(dL{+}1)$ tensor $\mathcal{Q}u \in \mathbb{R}^{n_c \times \underbrace{2 \times \cdots \times 2}_{dL}}$, $(\mathcal{Q}u)_{c,\, b_{11},\ldots,b_{dL}} = u_c(i_1, \ldots, i_d)$. We write $m = 2L$ for the number of tensor modes, each of physical dimension $d = 2$. The total state space has dimension $d^m = N^2$. A tensor network's *structure* on m modes is a triple $\mathcal{S} = (\tau, \pi, \chi)$ where τ is a rooted binary tree with m leaves ($m - 1$ internal nodes, $2m - 2$ edges of which m are leaf edges and $m - 2$ are internal); permutation $\pi \in \mathfrak{S}_m$ assigns m tensor modes to m leaves; and $\chi \in \mathbb{N}^{m-2}$ assigns a bond dimension to each internal edge. Given $\mathcal{S}$, a tensor network $G = \{G^{(v)}\}$ places an order-3 core $G^{(v)} \in \mathbb{R}^{\chi_{e_1} \times \chi_{e_2} \times \chi_{e_3}}$ at each internal node v and an order-2 core $G^{(\ell)} \in \mathbb{R}^{d \times \chi_e}$ at each leaf. The contraction map $\mathcal{C}_\mathcal{S} : G \mapsto \mathcal{Q}\omega \in (\mathbb{R}^d)^{\otimes m}$ evaluates the network. We define the structure manifold $\mathcal{M}_\mathcal{S} = \{\, \mathcal{C}_\mathcal{S}(G) : G \text{ has exact ranks } \chi \,\} \subset (\mathbb{R}^d)^{\otimes m}$.

3.2 The quantum topology sampler

We now specify the map from circuit parameters Θ_q to a distribution over structures $\mathcal{S} = (\tau, \pi, \chi)$. Structures are drawn from the Born distribution of a parameterised circuit on three registers: a *shape* register decoded to a Catalan shape by the cycle lemma, a *permutation* register decoded to a leaf assignment, and a *rank* register decoded to bond dimensions, with parameters $\Theta_\mathrm{q} = (\theta^s, \theta^\pi, \theta^r)$,

$$p_\theta(\mathcal{S}) = \big|\langle \mathcal{S} \mid U(\Theta_\mathrm{q}) \mid 0 \rangle\big|^2. \tag{1}$$

The output bitstring is partitioned as $\boldsymbol{b} = (\boldsymbol{b}_s, \boldsymbol{b}_\pi, \boldsymbol{b}_r)$, where $\boldsymbol{b}_s$ is used to design the tree structure τ, $\boldsymbol{b}_\pi$ assigns m tensor modes into m leaves, and $\boldsymbol{b}_r$ assigns bond dimension to each internal edge.



(a) Shape register (b) Permutation and rank register

Figure 4: The proposed circuit architecture of quantum sampler

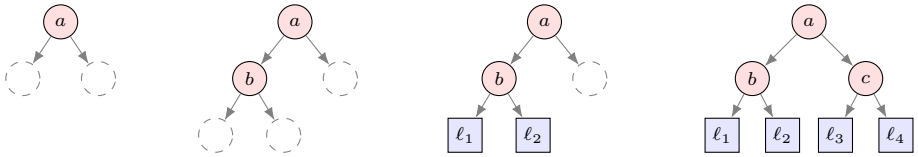
A straightforward bit-to-tree decoding $\boldsymbol{b}_s \mapsto \tau$ can be defined by assigning each two-bit symbol to a local branching operation: 00 terminates the current branch, 10 creates a left child, 01 creates a right child, and 11 creates both left and right children. This encoding does not guarantee that an arbitrary bitstring corresponds to a valid tree: some bitstrings create a leaf without a corresponding internal node (e.g. 0011). The circuit must then be sampled repeatedly until a valid bitstring appears, at substantial overhead.

Remark 1 *In our preliminary experiments, only* 0.70% *of samples yield valid trees for* $m = 12$*, corresponding to approximately* 143 *circuit executions per accepted sample, while the acceptance rate decreases to* 0.45% *for* $m = 16$*, requiring approximately* 222 *executions per accepted sample. This issue also adversely affects the sampler's distribution, as the probability mass assigned to valid trees is diluted across the larger search space containing invalid encodings.*

To address this issue, we propose to use *Łukasiewicz word* (Dvoretzky & Motzkin, 1947) which raises the acceptance rate to exactly 1, shown in Theorem 1.

Definition 1 *A word* $w \in \{0,1\}^{2m-1}$ *with exactly* $m-1$ *ones is a* Łukasiewicz word *if every proper prefix* $w_{1:k}$ *satisfies* $\#\{1\} - \#\{0\} \geq 0$*. Reading* w *as a preorder traversal that emits* 1 *at an internal node and* 0 *at a leaf reconstructs a unique rooted binary tree with* m *leaves, and this correspondence is a bijection onto the* C_{m-1} *such trees.*

Example 1 *Given* $m = 4$ *leaves, so the shape register has* $n_s = 2m - 1 = 7$ *qubits. Suppose the register measures* $\boldsymbol{b}_s = 0110010$*. A word is a valid Łukasiewicz code iff every proper prefix has non-negative partial sum. Enumerating all* 7 *cyclic rotations with their prefix sums yields unique Łukasiewicz code* 1100100 *(prefix sums is* $1, 2, 1, 0, 1, 0$*). The tree is built as follows:*



1100100: push a *1100100: push b* *1100100: b closes* *1100100: complete*

We show the quantum circuit used in the proposed quantum sampler in Figure 4. Panel 4a shows the shape register using $n_s = 2m - 1$ qubits with five qubits and 2 of the P cycles shown. The X layer prepares a computational basis state of Hamming weight $m - 1$ on a support drawn uniformly at circuit construction. A cycle applies R_z to every qubit, then a fSim brickwork on the pairs of parity $c \bmod 2$, then a CP brickwork on the complementary pairing. Specifically, $\sqrt{P} = \frac{1}{\sqrt{2}}\big(I - \mathrm{i}P\big)$, $W = \frac{X+Y}{\sqrt{2}}$, $R_z(\theta) = \begin{pmatrix} e^{-\mathrm{i}\theta/2} & 0 \\ 0 & e^{\mathrm{i}\theta/2} \end{pmatrix}$, and $\mathrm{fSim}(\theta,\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\mathrm{i}\sin\theta & 0 \\ 0 & -\mathrm{i}\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & e^{-\mathrm{i}\phi} \end{pmatrix}$ in the ordered basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Here, R_z, fSim and CP all commute with $\hat{n} = \frac{1}{2}\sum_j (I - Z_j)$. Panel 4b shows the permutation register ($n_\pi = \sum_{j=0}^{m-2} \lceil \log_2(m-j) \rceil$), and rank register ($n_r = b\,(m-2)$) with $b = 3$; four qubits shown. Each $g_q^{(c)} \in \{\sqrt{X}, \sqrt{Y}, \sqrt{W}\}$ is drawn uniformly subject to $g_q^{(c)} \neq g_q^{(c-1)}$ and is fixed at circuit construction.

3.3 CLASSICAL TENSOR SOLVER

The structure $\mathcal{S} = (\tau, \pi, \chi)$ emitted by the sampler is consumed by a classical solver that we develop in three parts: (1) a *truncation* that maps any dense field onto $\mathcal{M}_{\mathcal{S}}$, (2) a *projected step* that advances the network without leaving it, and (3) a *training protocol*. ASTER has two parameter sets. The closure parameters Θ_c parametrise a correction to the vector field, live in a continuous space, and are trained by back-propagation; the circuit parameters Θ_{q} parametrise a distribution over discrete structures. The trajectory error is governed by the normal defect $D = \|(\mathrm{I} - \mathrm{P})F\|$, an intrinsic property of $\mathcal{M}_{\mathcal{S}}$.

Truncation by hierarchical SVD. We adopt Grasedyck's hierarchical SVD (Grasedyck, 2010) as the truncation $\mathcal{T}_{\mathcal{S}} = \mathrm{HSVD}_{\mathcal{S}}$. For each non-root node v we matricise the lifted field $X = \mathcal{Q}u$ across the edge $e = (\mathrm{parent}(v), v)$, with the leaves below v indexing rows, and retain the leading $k_e = \min\{\chi_e, \mathrm{rank}\}$ left singular vectors as an isometric frame U_v, recording the discarded mass $t_e = \sum_{i>k_e} \sigma_i(e)^2$. Cores are then formed by projection onto those frames rather than by a second decomposition; Algorithm 10 states the construction in full.

The projected step. Let F be the semi-discrete right-hand side of the PDE. For a tree tensor network the projector admits the alternating sum (Lubich et al., 2013; Haegeman et al., 2016)

$$\mathrm{P}_{\mathrm{T}_G\mathcal{M}_{\mathcal{S}}} = \sum_{v \in V(\tau)} \mathrm{P}_v - \sum_{e \in E(\tau)} \mathrm{P}_e, \tag{2}$$

where P_v projects onto the span of variations of the core at v and P_e removes the gauge double-counting on edge e. The model is then

$$\dot{G} = \mathrm{P}_{\mathrm{T}_G\mathcal{M}_{\mathcal{S}}}\Big[\tilde{F}(G) + f_c(\Theta_c)(G; \mathcal{S})\Big], \qquad G(0) = \mathrm{HSVD}_{\mathcal{S}}(\mathcal{Q}u_0), \tag{3}$$

with $\tilde{F}$ a cheap tensor-format surrogate of F (Algorithm 8) and a learned closure $f_c(\Theta_c)$. We build the tangent projector from the network the previous truncation already produced, rather than re-decomposing the state. We form $G(y) = \mathcal{Q}^{-1}\mathrm{P}\,\mathcal{Q}\big[\tilde{F}(y) + f_c(\Theta_c)(y)\big]$, take four explicit Runge–Kutta stages of G, and truncate the result back onto $\mathcal{M}_{\mathcal{S}}$ (Algorithm 2). When $f_c(\Theta_c) \equiv 0$ the first stage is the projected right-hand side that the defect diagnostic has already computed, so we reuse it.

Co-optimisation. We train Θ_{q} and Θ_c in a single loop and a single backward pass, which makes ASTER trainable end to end across the quantum–classical boundary. We fix u_0 and compute reference snapshots $u_1, \ldots, u_n$ at $t_k = k\Delta t$. A structure $\mathcal{S}$ determines a rollout $\hat{u}_k = \mathcal{Q}^{-1}G_k$ with $G_0 = \mathcal{T}_{\mathcal{S}}(\mathcal{Q}u_0)$, and we score it by

$$\ell(\mathcal{S}; \Theta_c) = \mathcal{L}_{\mathrm{data}} + \lambda_r\,\mathcal{L}_{\mathrm{pde}} + \lambda_c\,\mathcal{L}_{\mathrm{cons}} + \lambda_\kappa\, n\, K(\mathcal{S}) + \rho\,\mathcal{E}(\mathcal{S}), \tag{4}$$

whose five terms are: $\mathcal{L}_{\mathrm{data}} = \Big(\sum_{k=1}^n w_k\Big)^{-1} \sum_{k=1}^n w_k \frac{\|\hat{u}_k - u_k\|^2}{\|u_k\|^2}$, $w_k = \frac{k}{n}, \sum_k w_k = \frac{n+1}{2}$; $\mathcal{L}_{\mathrm{pde}} = \frac{1}{n-1}\sum_{k=2}^n \Big\langle \Big(\frac{\hat{u}_k - \hat{u}_{k-1}}{\Delta t} - F(\hat{u}_{k-1})\Big)^2 \Big\rangle$ where $\langle\cdot\rangle$ is the mean over grid points and components; $\mathcal{L}_{\mathrm{cons}} = \frac{1}{n}\sum_{k=1}^n \sum_{I \in \mathcal{I}} \frac{|I(\hat{u}_k) - I(u_k)|}{|I(u_k)|}$ where $\mathcal{I}$ is the set of invariants listed per PDE; $K(\mathcal{S}) = \frac{\Phi_1(\mathcal{S})}{\dim \mathcal{M}_{\mathcal{S}}}$ where Φ_1 is analytic FLOPs of one step; and $\mathcal{E}(\mathcal{S}) = \sum_{k=1}^n \Big(\sum_e t_{e,k}\Big)^{1/2}$ where $t_{e,k}$ = mass discarded at edge e, step k. The objective is the expectation of equation 4 under the Born distribution the circuit induces,

$$\mathcal{L}(\Theta_{\mathrm{q}}, \Theta_c) = \mathbb{E}_{\mathcal{S} \sim p_{\Theta_{\mathrm{q}}}}\big[\ell(\mathcal{S}; \Theta_c)\big], \qquad \mathcal{S} = \mathcal{D}(\boldsymbol{b}),\ \ \boldsymbol{b} \sim p_{\Theta_{\mathrm{q}}}, \tag{5}$$

with $\mathcal{D}$ the total decoder of the quantum sampler. We replace the rollout by a proxy on the same scale and keep the cost functional identical:

$$\ell_0(\mathcal{S}) = \hat{\varepsilon}(\mathcal{S})^2 + \lambda_\kappa K(\mathcal{S}), \qquad \hat{\varepsilon}(\mathcal{S}) = \frac{\|\mathcal{Q}u_0 - \mathcal{T}_{\mathcal{S}}\mathcal{Q}u_0\|}{\|\mathcal{Q}u_0\|}. \tag{6}$$

We use the REINFORCE identity (Williams, 1992) $\nabla_{\Theta_{\mathrm{q}}}\mathbb{E}[\ell] = \mathbb{E}[\ell\,\nabla_{\Theta_{\mathrm{q}}} \log p_{\Theta_{\mathrm{q}}}]$, which needs only that $\log p_{\Theta_{\mathrm{q}}}$ be evaluable and differentiable. For a Born machine $\partial p_{\Theta_{\mathrm{q}}}(\boldsymbol{b})/\partial\theta_j$ is a finite

combination of the same circuit at shifted parameters (Mitarai et al., 2018; Schuld et al., 2019; Wierichs et al., 2022), so a parametrised circuit returns this exactly. The estimator is hybrid: score function on the categorical part $\zeta = (\tau, \pi)$, pathwise on the ranks via a Gumbel–softmax (Maddison et al., 2016; Jang et al., 2016) with a straight-through gradient (Bengio et al., 2013). All three gradients are terms of one surrogate,

$$\widehat{J} = \frac{1}{B}\sum_{i=1}^{B}\Big[\underbrace{\ell(\mathcal{S}_i;\Theta_c)}_{\nabla_{\Theta_c}\text{: pathwise}} + \underbrace{a_i\,\log \tilde{p}_{\Theta_q,\epsilon}(\zeta_i)}_{\nabla_{\Theta_q}\text{: score function}} + \underbrace{\lambda_\kappa\, n\, K(\tau_i,\pi_i,\tilde{\chi}_i)}_{\nabla_{\Theta_q}\text{: pathwise, via }\tilde{\chi}}\Big], \tag{7}$$

so a single backward pass yields $\nabla_{\Theta_c}\mathcal{L}$ and $\nabla_{\Theta_q}\mathcal{L}$ together. Here, B is the number of structures drawn per update, $a_i = (\ell_i - \beta_i)/\hat{\sigma}$ is the standardised advantage, $\beta_i = (B-1)^{-1}\sum_{j\neq i}\ell_j$ the leave-one-out baseline (Kool et al., 2019), and $\hat{\sigma}$ the sample standard deviation of $\ell - \beta$. We write B for this count throughout to keep it distinct from the storage metric S of §4.

3.4 Algorithms

Algorithm 1 performs structure sampling with the quantum sampler, Algorithm 8 constructs the surrogate in equation 3, Algorithm 2 performs the projection step, and Algorithm 3 implements the rank adaptation of ASTER$^+$, which lowers approximation error against a parameter cost. Lines marked ★ are novel components.

Input: circuit parameters $\Theta_q = (\theta^s, \theta^\pi, \theta^r)$; Schmidt rank R; field spec (d, L, n_c)
Output: structure $\mathcal{S} = (\tau, \pi, \chi)$ distributed as p_θ

1: ★ $\lambda \leftarrow \mathrm{softmax}(w)$ and $w \in \mathbb{R}^R$. $z \sim \mathrm{Categorical}(\lambda)$
2: **for all** registers $w \in \{s, \pi, r\}$ **do**
3: ★ $\Theta_q^w \leftarrow \Theta_q^w + \Delta\Theta_q^w$
4: $q^w \leftarrow \left|U(\Theta_q^w)|0\rangle\right|^2$; $b^w \sim q^w$ ▷ Born rule
5: **end for**
6: $\tau \leftarrow$ CycleLemmaDecode(b^s), $\pi \leftarrow$ LehmerDecode(b^π); $\chi \leftarrow$ ClipFeasible(b^r)
7: **return** $\mathcal{S} = (\tau, \pi, \chi)$

Algorithm 1: Structure sampling from the Born distribution

Input: network $G \in \mathcal{M}_\mathcal{S}$; right-hand side F; step Δt; closure $f_c(\Theta_c)$
Output: $G' \in \mathcal{M}_\mathcal{S}$ advanced by Δt

1: $V \leftarrow \mathcal{Q}\big(\tilde{F}(\mathcal{Q}^{-1}G)\big) + f_c(G)$ ▷ lift the flow to quantics form
2: **for all** leaves ℓ of τ **do**
3: $Y \leftarrow$ Unfold(V, ℓ)
4: ★ $Y \leftarrow Q_\ell\,(Q_\ell^\top Y)$
5: $V \leftarrow$ Fold(Y, ℓ)
6: **end for**
7: $PV \leftarrow \sum_v \mathrm{P}_v V - \sum_e \mathrm{P}_e V$ ▷ alternating sum equation 2
8: $G' \leftarrow \mathrm{RK4}(G, PV, \Delta t)$; $G' \leftarrow \mathrm{HSVD}_\mathcal{S}(G')$
9: ★ reuse G' as the next step's network; reuse PV as the next k_1 when $f_c \equiv 0$
10: **return** G'

Algorithm 2: One projected step using closure $f_c(\Theta_c)$

4 Experimental Results

4.1 Experimental design

ASTER as an integrator. Every method receives the same semi-discrete right-hand side F and the same $u(0)$. Seven PDEs with closed-form solutions are integrated to $T = 0.5$ on $N^2 = 32^2$ (2D) and $N^3 = 16^3$ (3D) grids, 30 unseen initial conditions each. ASTER$^+$ adds rank adaptivity, trading storage for error. Baselines are pseudo-spectral RK4, pseudo-spectral ETD-RK4 (Cox & Matthews,

2002; Kassam & Trefethen, 2005) and a MUSCL finite-volume scheme (Van Leer, 1979; Rusanov, 1962). The error ε is the relative L^2 error against the closed-form solution, and the storage S is the number of real numbers a method retains to represent the state, that is $N^d n_c$ for a dense solver and $\sum_v \prod(\text{core shape})$ for a tensor network.

We report the storage–accuracy trade-off two ways. The *primary* comparison is *matched accuracy*: we identify the problems on which ASTER$^+$ reproduces a baseline's relative error to three significant figures and compare storage at that fixed accuracy, which is like-for-like and needs no scalarisation. As a *secondary* summary we retain the *accuracy density* $\eta = 100 \log_{10}(1/\varepsilon)/S$, correct digits per 100 numbers stored. It is a metric of our own construction, it rewards a small denominator superlinearly, and S counts tensor-network cores against a dense solver's full field, so it reads as memory footprint rather than model capacity.

ASTER-SOLVER as a PDE solver. NN-based PDE solvers use training trajectories instead of the equations F. To compare against them we remove ASTER's access to F, replacing its right-hand side by a surrogate fitted from the same trajectories the operators train on, so at test time no method evaluates the governing equations. We name this variant ASTER-SOLVER. The task is to predict $u(T)$ from $u(T/2)$ of an unseen initial condition, with the operator stride set equal to the integrator step count. Twelve 2D PDEs are swept over grids $\{8, 16, 32\}$, giving 36 problem instances, with 80 training, 20 validation and 30 test unseen initial conditions. We compare ASTER-SOLVER against FNO (Li et al., 2020), DeepONet (Lu et al., 2021) and a latent neural ODE (Chen et al., 2018).

4.2 EFFECTIVENESS ANALYSIS

ASTER as an integrator. In Figure 5 (left panel), on six of the seven PDEs, ASTER$^+$ reproduces the relative error of spectral RK4 to three significant figures. On five of those it does so from between $2.9\times$ and $10.2\times$ fewer stored scalars; on `burgers2d` it requires $1.3\times$ more, that is 2752 scalars against a dense field of 2048, so there the tensor network is larger than the field it represents. On `ABC3d` the errors are close but not matched, 6.58×10^{-14} against 5.31×10^{-14}, from $32.6\times$ fewer scalars. Spectral ETD-RK4 attains the lowest error of any method on all seven problems.

In this evaluation, we emphasise on the method's compression ability at fixed error. ASTER$^+$ has the highest accuracy density on six of seven, losing `burgers2d` to both spectral methods (0.12 against 0.16), and ASTER has the highest on six of seven, losing `heat3d` to all three baselines (0.13 against 0.20, 0.36 and 0.23). The two variants fail on different problems, so the better of the two leads everywhere but neither does so alone. The largest margin is `convTGV2d`, where ASTER$^+$ attains $\eta = 3.92$ at $S = 201$ against $\eta = 0.38$ at $S = 2048$ for the best RK4-based competitor. For `ABC3d` the three-component velocity field needs $3 \times 16^3 = 12{,}288$ dense parameters, against 649 and 377 for ASTER and ASTER$^+$.

ASTER-SOLVER as a PDE solver. Figure 5 (right panel) shows that ASTER-SOLVER consistently requires fewer parameters than the NN-based solvers, with 11 of the 36 settings falling in the low-error, low-parameter region. ASTER-SOLVER attains the lowest relative L^2 error at all three resolutions on eight of the twelve PDEs: `heat2d`, `reactdiff2d`, `aniso2d`, `hyper2d`, `burgers2d`, `convTGV2d`, `tgvhi2d` and `stokes2d`. On three others, it wins at some resolutions only, losing `advdiff2d` and `wave2d` at $N = 8$ and `schrod2d` at $N = 8$ and 16; on `wave2d` at $N = 8$ its error of 1.04 exceeds the zero predictor, so no method in that column is usable. It loses `linkdv2d` at every resolution and diverges there at $N = 16$ and 32, which we attribute to a mismatch between tree manifolds and the dispersive third-derivative term.

Comparison with existing TN-SS methods. We compare ASTER$^+$ with TNGA (Li & Sun, 2020), TNLS (Li et al., 2022) and TnALE (Li et al., 2023) on seven two-dimensional flow fields at five resolutions $N \in \{16, 32, 64, 128, 256\}$, giving 35 configurations. Stored scalars S are reported at relative accuracy $\varepsilon \leq 10^{-5}$ with $\chi \leq 30$, at the median of three seeds. ASTER$^+$ requires a geometric mean of 162, 221 and 272 scalars at $N = 64, 128, 256$. TNGA is the most compact baseline at every resolution, and against it ASTER$^+$ stores $0.93\times$, $0.63\times$ and $0.39\times$ as much at those resolutions, equivalently $1.1\times$, $1.6\times$ and $2.6\times$ more compact. Against TNLS and TnALE at $N = 256$, it is $22.6\times$ and $52.7\times$ more compact. The advantage does not hold at small N: TNGA

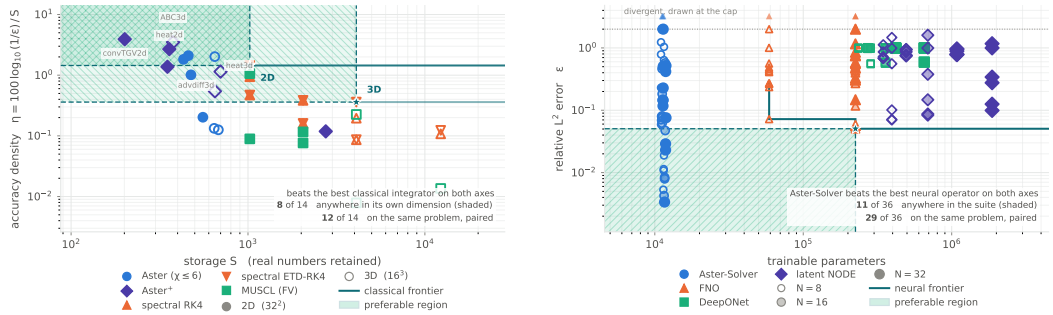


Figure 5: Benchmarking results: **(Left)** ASTER and ASTER$^+$ as integrators. **(Right)** ASTER-SOLVER against neural-network-based solvers. The shaded region marks preferable results with low storage and low error.

is more compact at $N = 16$ (44 against 58) and $N = 32$ (88 against 101), so the crossover is at $N = 64$.

5 DISCUSSION AND CONCLUSION

Ablation study. Table 9 compares the quantum sampler with a classical uniform sampler and with two hand-chosen fixed structures. The trained quantum sampler improves on the balanced tree with blocked ordering in accuracy density on six of the seven PDEs and ties on `heat3d`. The uniform sampler does not: it improves accuracy density on only three, losing on `heat2d` (0.39 against 1.84), `advdiff2d` (0.38 against 0.92) and `advdiff3d` (0.08 against 0.09), though it does reduce storage on five. Against the classical sampler, the quantum sampler reduces S on four of seven PDEs, ties on `heat3d` where the two differ by 0.01 in η, and stores more on `burgers2d` (556 against 512) while attaining higher η there (0.20 against 0.19).

Advantages of bit-to-tree decoder. The bit-to-tree decoder of §3.2 guarantees valid tree structures and so removes the rejection cost, which we formalise as follows:

Theorem 1 *Let $m \geq 2$, let $n_s = 2m - 1$ be the width of the shape register, let $B_m = \{\, \boldsymbol{b}_s \in \{0,1\}^{n_s} : |\boldsymbol{b}_s| = m - 1 \,\}$ be the fixed-Hamming-weight subspace, and let $\mathcal{T}_m$ denote the rooted binary trees with m leaves. Write $\mathcal{D}_\tau =$ CYCLELEMMADECODE for the shape decoder of Algorithm 1. Then:*

(i) (Totality) *For every $\boldsymbol{b}_s \in B_m$ there is exactly one $s^\star(\boldsymbol{b}_s) \in \{0, \ldots, n_s - 1\}$ such that $\sigma^{s^\star}(\boldsymbol{b}_s)$ is a Łukasiewicz word. Hence $\mathcal{D}_\tau$ is a total, surjective map $B_m \to \mathcal{T}_m$ and never rejects.*

(ii) (Uniform fibres) *$\mathcal{D}_\tau$ is exactly n_s-to-one: $|\mathcal{D}_\tau^{-1}(\tau)| = 2m - 1$ for every $\tau \in \mathcal{T}_m$, and consequently $|B_m| = \binom{2m-1}{m-1} = (2m-1)\, C_{m-1}$.*

(iii) (Zero rejection) *If the shape-register circuit $U(\Theta_{\mathrm{q}}^s)$ is noiseless, commutes with the total number operator $\hat{n} = \frac{1}{2}\sum_j (I - Z_j)$ and is initialised in the weight-$(m-1)$ sector, then $\operatorname{supp} p_\theta \subseteq B_m$. Algorithm 1 therefore accepts with probability 1 and yields one structure per circuit execution.*

Conclusion. We presented ASTER, an end-to-end differentiable quantum–classical framework that discovers tensor-network topologies for PDE solving rather than fixing them a priori. A surjective bit-to-tree decoder yields one valid structure per circuit execution with exact differentiability, and a projected integrator with learned closure jointly optimises structure and dynamics. Across seven closed-form PDEs, ASTER$^+$ reproduces the accuracy of spectral RK4 on six from 2.9× to 10× fewer stored scalars. Across twelve two-dimensional problems it attains the lowest error at every resolution on eight; and as a compressor it is the most compact structure-search method at the three highest resolutions.

AI USE STATEMENT

In this work, we used generative AI tools for grammar and language editing, code and theorem auditing, and simulated peer review of the original draft. We did not use generative AI tools for method formulation, implementation planning, or the formulation or proof of theorems. We have carefully reviewed all AI-assisted work and take full responsibility for the final content of this work, including all text, claims, analyses, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

We declare that there are no conflicts of interest.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed method and its implementation in the main text and appendices. The source code implementing our method is also submitted with the manuscript to facilitate reproducibility.

REFERENCES

Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pp. 333–342, 2011.

Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *nature*, 574(7779):505–510, 2019.

Markus Bachmayr, Reinhold Schneider, and André Uschmajew. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. *Foundations of Computational Mathematics*, 16(6):1423–1472, 2016.

Jonas Ballani and Lars Grasedyck. Tree adaptive approximation in the hierarchical tensor format. *SIAM journal on scientific computing*, 36(4):A1415–A1431, 2014.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J Bremner, John M Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595–600, 2018.

Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. On the complexity and verification of quantum random circuit sampling. *Nature Physics*, 15(2):159–163, 2019.

Gianluca Ceruti and Christian Lubich. An unconventional robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 62(1):23–44, 2022.

Gianluca Ceruti, Christian Lubich, and Hanna Walach. Time integration of tree tensor networks. *SIAM Journal on Numerical Analysis*, 59(1):289–313, 2021.

Gianluca Ceruti, Jonas Kusch, and Christian Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 62(4):1149–1174, 2022.

Gianluca Ceruti, Christian Lubich, and Dominik Sulz. Rank-adaptive time integration of tree tensor networks. *SIAM Journal on Numerical Analysis*, 61(1):194–222, 2023.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Steven M Cox and Paul C Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

Brian Coyle, Daniel Mills, Vincent Danos, and Elham Kashefi. The born supremacy: quantum advantage and training of an ising born machine. *npj Quantum Information*, 6(1):60, 2020.

Sergey V Dolgov, Boris N Khoromskij, and Ivan V Oseledets. Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the fokker–planck equation. *SIAM Journal on Scientific Computing*, 34(6):A3016–A3038, 2012.

Aryeh Dvoretzky and Th Motzkin. A problem of arrangements. 1947.

Xun Gao, Eric R Anschuetz, Sheng-Tao Wang, J Ignacio Cirac, and Mikhail D Lukin. Enhancing generative models via quantum correlations. *Physical Review X*, 12(2):021037, 2022.

Nikita Gourianov, Michael Lubasch, Sergey Dolgov, Quincy Y van den Berg, Hessam Babaee, Peyman Givi, Martin Kiffner, and Dieter Jaksch. A quantum-inspired approach to exploit turbulence structures. *Nature Computational Science*, 2(1):30–37, 2022.

Lars Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM journal on matrix analysis and applications*, 31(4):2029–2054, 2010.

Wolfgang Hackbusch and Stefan Kühn. A new scheme for the tensor representation. *Journal of Fourier analysis and applications*, 15(5):706–722, 2009.

Jutho Haegeman, Christian Lubich, Ivan Oseledets, Bart Vandereycken, and Frank Verstraete. Unifying time evolution and optimization with matrix product states. *Physical Review B*, 94(16): 165116, 2016.

Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised generative modeling using matrix product states. *Physical Review X*, 8(3):031012, 2018.

Meraj Hashemizadeh, Michelle Liu, Jacob Miller, and Guillaume Rabusseau. Adaptive tensor learning with tensor networks. *arXiv preprint arXiv:2008.05437*, pp. 2018, 2020.

Mohamed Hibat-Allah, Martin Ganahl, Lauren E Hayward, Roger G Melko, and Juan Carrasquilla. Recurrent neural network wave functions. *Physical Review Research*, 2(2):023358, 2020.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Aly-Khan Kassam and Lloyd N Trefethen. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

Boris N Khoromskij. O (d log n)-quantics approximation of n-d tensors in high-dimensional numerical modeling. *Constructive Approximation*, 34(2):257–280, 2011.

Martin Kiffner and Dieter Jaksch. Tensor network reduced order models for wall-bounded flows. *Physical Review Fluids*, 8(12):124101, 2023.

Othmar Koch and Christian Lubich. Dynamical low-rank approximation. *SIAM Journal on Matrix Analysis and Applications*, 29(2):434–454, 2007.

Wouter Kool, Herke Van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

Chao Li and Zhun Sun. Evolutionary topology search for tensor network decomposition. In *International conference on machine learning*, pp. 5947–5957. PMLR, 2020.

Chao Li, Junhua Zeng, Zerui Tao, and Qibin Zhao. Permutation search of tensor network structures via local sampling. In *International conference on machine learning*, pp. 13106–13124. PMLR, 2022.

Chao Li, Junhua Zeng, Chunmei Li, Cesar F Caiafa, and Qibin Zhao. Alternating local enumeration (tnale): Solving tensor network structure search with fewer evaluations. In *International conference on machine learning*, pp. 20384–20411. PMLR, 2023.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Jin-Guo Liu and Lei Wang. Differentiable learning of quantum circuit born machines. *Physical Review A*, 98(6):062324, 2018.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Christian Lubich and Ivan V Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 54(1):171–188, 2014.

Christian Lubich, Thorsten Rohwedder, Reinhold Schneider, and Bart Vandereycken. Dynamical approximation by hierarchical tucker and tensor-train tensors. *SIAM Journal on Matrix Analysis and Applications*, 34(2):470–494, 2013.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Raghavendra Dheeraj Peddinti, Stefano Pisoni, Alessandro Marini, Philippe Lott, Henrique Argentieri, Egor Tiunov, and Leandro Aolita. Quantum-inspired framework for computational fluid dynamics. *Communications Physics*, 7(1):135, 2024.

Marc K Ritter, Yuriel Núñez Fernández, Markus Wallerberger, Jan Von Delft, Hiroshi Shinaoka, and Xavier Waintal. Quantics tensor cross interpolation for high-resolution parsimonious representations of multivariate functions. *Physical Review Letters*, 132(5):056501, 2024.

Vladimir Vasil'evich Rusanov. The calculation of the interaction of non-stationary shock waves and obstacles. *USSR Computational Mathematics and Mathematical Physics*, 1(2):304–320, 1962.

Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

Bram Van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of computational Physics*, 32(1):101–136, 1979.

Guifré Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical review letters*, 91(14):147902, 2003.

David Wierichs, Josh Izaac, Cody Wang, and Cedric Yen-Yu Lin. General parameter-shift rules for quantum gradients. *Quantum*, 6:677, 2022.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Erika Ye and Nuno FG Loureiro. Quantum-inspired method for solving the vlasov-poisson equations. *Physical Review E*, 106(3):035208, 2022.

A SUPPLEMENTAL DETAILS OF METHOD

A.1 QUANTUM CIRCUITS

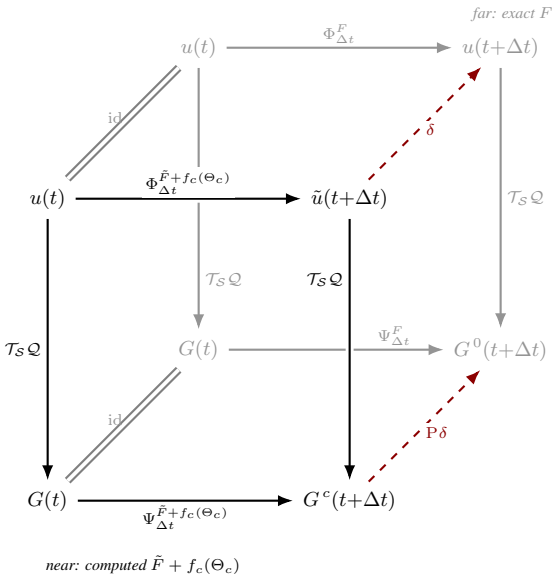
The three registers are disjoint and the joint state is a product, $U(\Theta_{\mathrm{q}}) = U(\theta^s) \otimes U(\theta^\pi) \otimes U(\theta^r)$, so the Born law of equation 1 factorises as $p_\theta(\boldsymbol{b}) = p_{\theta^s}(\boldsymbol{b}_s)\, p_{\theta^\pi}(\boldsymbol{b}_\pi)\, p_{\theta^r}(\boldsymbol{b}_r)$, and the three bitstrings are decoded independently by CYCLELEMMADECODE, LEHMERDECODE and CLIPFEASIBLE. Experiments use $P = 6$ cycles per register; two are drawn. Measurement is in the computational basis and is exact, by perfect sampling from the matrix-product state, with the realised truncation weight asserted below 10^{-12}. Angles are drawn as $\theta \sim \frac{\pi}{2} + \mathcal{N}(0, 0.35^2)$ and $\phi \sim \frac{\pi}{6} + \mathcal{N}(0, 0.35^2)$.

Remark 2 *The acceptance rates quoted in Remark 1 are measured as follows. The naive decoder reads $\boldsymbol{b}_s$ as two-bit symbols under the branching convention of §3.2 and accepts a bitstring when the induced traversal closes exactly one rooted binary tree with m leaves and consumes the register without underflow or leftover symbols. We draw bitstrings uniformly from $\{0,1\}^{2m-1}$ and report the fraction accepted, which is the ratio of valid encodings to the full register volume and does not depend on the circuit parameters.*

Remark 3 *We evaluate equation 2 using $2m-1$ node terms minus $2m-2$ edge terms in the factored form. P_v is the product, over the edges incident to v, of the orthogonal projector onto the frame of the* far *side of that edge. P_e is the product of the two projectors either side of e. The factors of each term act on disjoint leaf groups, so they commute and the term reduces to a short sequence of small matrix products. We apply $Y \mapsto Q\,(Q^\top Y)$ at $\mathcal{O}(k\,n)$ against the $\mathcal{O}(d_\ell\, n)$ of the materialised projector, where $k = \chi_e$ is a bond dimension and the far-group dimension d_ℓ grows to as much as half the ambient tensor (Algorithm 2, line 4). The two forms are algebraically identical but our approach gives $7.9\times$ speed-up at $\leq 8.3 \times 10^{-16}$ deviation.*

A.2 CLOSURE LEARNING

We demonstrate the concept of closure learning for dynamical simulation in Figure 6.



Notation.

$\mathcal{H}$	ambient quantics space
$\mathcal{M}_{\mathcal{S}}$	tensor manifold of $\mathcal{S} = (\tau, \pi, \chi)$, $\mathcal{M}_{\mathcal{S}} \subset \mathcal{H}$
$\mathcal{Q}$	quantics lift; exact, invertible
$\mathcal{T}_{\mathcal{S}}$	HSVD truncation onto $\mathcal{M}_{\mathcal{S}}$; the identity on $\mathcal{M}_{\mathcal{S}}$
P	tangent projector $\mathrm{P}_{\mathrm{T}_G\mathcal{M}_{\mathcal{S}}}$
F	exact semi-discrete right-hand side
$\tilde F,\ f_c(\Theta_c)$	cheap surrogate; learned closure
$\Phi^V_{\Delta t}$	time-Δt flow of V in $\mathcal{H}$
$\Psi^V_{\Delta t}$	$= \mathcal{T}_{\mathcal{S}} \circ (\mathrm{id} + \Delta t\, \mathrm{P} V)$, one step on $\mathcal{M}_{\mathcal{S}}$
G^0, G^c	one step of F; of $\tilde F + f_c(\Theta_c)$
δ	$= \Delta t\big[(\tilde F + f_c(\Theta_c)) - F\big] + \mathcal{O}(\Delta t^2)$
D	$= (\mathrm{I} - \mathrm{P})F(G) \in (\mathrm{T}_G\mathcal{M}_{\mathcal{S}})^\perp$

Figure 6: **The closure cube.** Axes: time (horizontal), truncation onto $\mathcal{M}_{\mathcal{S}}$ (vertical), and model (depth). The exact F on the far grey face, the computed $\tilde F + f_c(\Theta_c)$ on the near one. The two agree at t, so the left depth edges are identities. The depth displacement is δ along the ambient edge but only $\mathrm{P}\delta$ along the manifold edge, because $\Psi^V_{\Delta t}$ projects *before* advancing. First-order steps are drawn.

A.3 TRAINING SCHEDULE

We propose a three-stage training schedule.

1. **Warm start.** Draw B structures and score each by the proxy equation 6, which costs $\mathcal{O}(L\chi^4)$ and requires no time stepping, then take the score-function step of equation 7. The sampler reaches a sensible region of (τ, π) before any rollout is paid for.
2. **Closure.** Train Θ_c against equation 4 with Θ_{q} frozen at a short horizon, so the closure is fitted on a structure distribution that has stopped moving.
3. **Joint.** Alternate, updating Θ_c every epoch and Θ_{q} every n_{alt}-th epoch ($n_{\mathrm{alt}} = 20$), with Adam at 10^{-3} and 10^{-2} respectively. With $t \in [0, 1]$ the normalised epoch we anneal three schedules together: the horizon geometrically from 4 to 64 steps, the smoothing ϵ geometrically from 10^{-2} to 10^{-4}, and the Gumbel–softmax temperature linearly from 1.0 to 0.1.

We write n_{alt} for the alternation period to keep it distinct from the number of field components n_c of §3.1. In the cost function presented in §3.3, $\mathcal{L}_{\mathrm{data}}$ represents a weighted mean of squared relative errors, where w_k increases as k increases. $\mathcal{L}_{\mathrm{pde}}$ evaluates the reconstructed trajectory with respect to the governing equations, while $\mathcal{L}_{\mathrm{cons}}$ assesses conformity to the invariants inherent in the formulation. K is the cost per degree of freedom of the manifold the structure defines, and the factor n makes the total cost of the rollout explicit. $\mathcal{E}$ charges for the information that the truncation throws away, thereby pricing gratuitous topology switching when temporal adaptation is enabled.

Rank-adaptive dynamical low-rank integration. The closest numerical-analysis counterpart to ASTER$^+$ is rank-adaptive dynamical low-rank approximation. The unconventional basis-update and Galerkin integrator (Ceruti & Lubich, 2022) is robust to small singular values, and (Ceruti et al., 2022) adds a rank-adaptive step driven by a tolerance on the discarded mass; both have been lifted to tree tensor networks, at prescribed tree rank (Ceruti et al., 2021) and rank-adaptively (Ceruti et al., 2023). ASTER$^+$ adapts bond dimensions by the same principle (Algorithm 3) but additionally samples and learns the tree and the leaf ordering, so rank adaptivity moves within a tree while the sampler moves between trees.

A.4 HYPERPARAMETERS

Table 1: Protocol hyperparameters of the three studies.

Group	Symbol	Value
Integrator study	grids	32^2 (2D), 16^3 (3D)
	horizon T, step Δt	0.5; Δt set per PDE by the advective and diffusive stability limits, then shrunk so an integer number of steps lands on T — $\Delta t \in [1.85\times10^{-2},\ 1.25\times10^{-1}]$, i.e. 4 to 27 steps
	rank cap $\chi_{\max}$	6 (ASTER); 32 (ASTER$^+$)
	ASTER$^+$ tolerance	searched over $\{10^{-4}, 10^{-6}, 10^{-8}\}$; 10^{-4} selected on every PDE
	ASTER$^+$ rank update	every step, from the *unprojected* predictor $u + h\,F(u)$
	validation / test initial conditions	5 / 30, disjoint; the search sees validation only
	seeds reported	1
Solver study	grids	$\{8, 16, 32\}$
	train / validation / test	80 / 20 / 30 initial conditions
	task	predict $u(T)$ from $u(T/2)$; training trajectories cover $[0, T/2]$ only
	operator training budget	100 optimiser steps on minibatches of 8 trajectories, K-step pushforward loss, graph detached every 2 steps
	closure / surrogate $f_c(\Theta_c)$	circular-padded CNN, width 24, depth 3; 250 Adam steps at 10^{-3}, minibatch 128
	baseline capacities	FNO width 24, depth 3, 8 modes (Adam 10^{-3}); DeepONet width $= p = 192$, depth 4 (Adam 3×10^{-4}); latent neural ODE width 384, latent 96
	matched stride	operator rollout length K equals the integrator's step count, asserted at run time
	seeds reported	1
Compression study	resolutions	$\{16, 32, 64, 128, 256\}$
	accuracy target, rank cap	$\varepsilon \le 10^{-5}$, $\chi \le 30$
	seeds reported	3; median over seeds, then geometric mean over the seven two-dimensional fields
	search budget per baseline	ASTER$^+$ and APTN: 48 structure evaluations. TNGA, TNLS, TnALE: evaluation budget 20, 6 sweeps, 1 restart (the baselines' own published settings)

B SUPPLEMENTAL DETAILS OF RESULTS

B.1 DETAILS OF EVALUATED PDES

Table 2: The seven benchmark PDEs of ASTER and ASTER$^+$ as PDE integrators.

Name	d	n_c	ν	Governing equation on $\mathbb{T}^d$
`heat2d`	2	1	0.05	$\partial_t u = \nu \Delta u$
`advdiff2d`	2	1	0.02	$\partial_t u + c \cdot \nabla u = \nu \Delta u$
`burgers2d`	2	2	0.05	$\partial_t u + (u \cdot \nabla) u = \nu \Delta u$
`convTGV2d`	2	2	0.05	$\partial_t u + (u \cdot \nabla) u = -\nabla p + \nu \Delta u$, $\nabla \cdot u = 0$
`heat3d`	3	1	0.05	$\partial_t u = \nu \Delta u$
`advdiff3d`	3	1	0.02	$\partial_t u + c \cdot \nabla u = \nu \Delta u$
`ABC3d`	3	3	0.05	$\partial_t u + (u \cdot \nabla) u = -\nabla p + \nu \Delta u$, $\nabla \cdot u = 0$; Beltrami $\nabla \times u = k\, u$, $k = 1$

Table 3: The twelve two-dimensional problems of the ASTER-SOLVER dynamics.

Name	n_c	ν	Governing equation on $\mathbb{T}^2$
`heat2d`	1	0.05	$\partial_t u = \nu \Delta u$
`advdiff2d`	1	0.02	$\partial_t u + c \cdot \nabla u = \nu \Delta u$
`reactdiff2d`	1	0.05	$\partial_t u = \nu \Delta u - \lambda u$, $\lambda = 0.5$
`aniso2d`	1	0.02	$\partial_t u = \nu \big(\partial_x^2 + r\, \partial_y^2\big) u$, $r = 8$
`hyper2d`	1	10^{-3}	$\partial_t u = -\nu \Delta^2 u$
`schrod2d`	2	—	$\mathrm{i}\, \partial_t \psi = -\frac{1}{2} \Delta \psi$
`linkdv2d`	1	10^{-2}	$\partial_t u = -\partial_x^3 u + \nu \Delta u$
`wave2d`	2	—	$\partial_t u = v$, $\partial_t v = c^2 \Delta u$, $c = 1$
`burgers2d`	2	0.05	$\partial_t u + (u \cdot \nabla) u = \nu \Delta u$
`convTGV2d`	2	0.05	$\partial_t u + (u \cdot \nabla) u = -\nabla p + \nu \Delta u$, $\nabla \cdot u = 0$
`tgvhi2d`	2	0.01	as `convTGV2d`, five times the Reynolds number
`stokes2d`	2	0.05	$\partial_t u = \nu \Delta u - \nabla p$, $\nabla \cdot u = 0$

Table 4: The seven two-dimensional fields used to compare ASTER$^+$ with the structure-search baselines.

Name	Field $u(x, y)$ on $\mathbb{T}^2 = [0, 2\pi)^2$
TGV vorticity	$\cos x \, \cos y$
TGV velocity-u	$1 + \sin x \, \cos y$
Shear layer	$\tanh\big((y - \pi)/0.4\big)$
Double shear	$\tanh\big(3 \sin(x - \pi)\big)$
Gaussian vortex	$\exp\big(-\big[(x - \pi)^2 + (y - \pi)^2\big]/0.6\big)$
Vortex pair	$\exp\big(-\big[(x - 2)^2 + (y - \pi)^2\big]/0.4\big) - \exp\big(-\big[(x - 4)^2 + (y - \pi)^2\big]/0.4\big)$
Multiscale	$\sin x \, \cos y + \frac{1}{2} \sin 4x \, \cos 4y + \frac{1}{4} \sin 8x \, \cos 2y$

B.2 HYPERPARAMETERS OF COMPARED NN-BASED PDE SOLVERS

Two configurations per architecture, selected on the validation split by rollout MSE: FNO {width 24, depth 3, modes 8} and $\{32, 4, 8\}$; DeepONet {width 192, p=192} and $\{256, 256\}$; latent neural ODE {width 384, latent 96} and $\{512, 128\}$. The optimiser used is Adam with learning rate $\eta = 10^{-3}$ (3×10^{-4} for DeepONet, whose unnormalised branch–trunk inner product over p sensors diverges at 10^{-3} until scaled by $1/\sqrt{p}$). All methods are trained in 150 epochs.

B.3 COMPARISON RESULTS IN §4

Table 5: Evaluation of ASTER and ASTER$^+$ as an integrator. Every structure is a Born draw from the random-circuit sampler, selected on validation initial conditions disjoint from the test set.

	$N = 32$ (2D)				$N = 16$ (3D)		
Method	heat2d	advdiff2d	burgers2d	convTGV2d	heat3d	advdiff3d	ABC3d
(a) relative L^2 error ε against the closed-form solution							
ASTER ($\chi \le 6$)	2.21×10^{-10}	1.52×10^{-5}	0.0747	1.34×10^{-8}	0.139	0.14	7.51×10^{-14}
ASTER$^+$	2.21×10^{-10}	1.52×10^{-5}	0.000544	1.34×10^{-8}	9.71×10^{-9}	0.000292	6.58×10^{-14}
spectral RK4	2.21×10^{-10}	1.52×10^{-5}	0.000544	1.34×10^{-8}	9.71×10^{-9}	0.000292	5.31×10^{-14}
spectral ETD-RK4	$\mathbf{1.60\times10^{-15}}$	$\mathbf{1.49\times10^{-5}}$	**0.000519**	$\mathbf{1.32\times10^{-8}}$	$\mathbf{1.74\times10^{-15}}$	**0.000285**	$\mathbf{7.20\times10^{-16}}$
MUSCL (FV)	1.37×10^{-11}	0.122	0.0273	0.00416	5.95×10^{-10}	0.476	0.0215
(b) storage S							
ASTER ($\chi \le 6$)	460	476	**556**	432	**644**	680	649
ASTER$^+$	**359**	**350**	2752	**201**	700	**650**	**377**
spectral RK4	1024	1024	2048	2048	4096	4096	12288
spectral ETD-RK4	1024	1024	2048	2048	4096	4096	12288
MUSCL (FV)	1024	1024	2048	2048	4096	4096	12288
(c) accuracy density $\eta = 100\log_{10}(1/\varepsilon)/S$							
ASTER ($\chi \le 6$)	2.10	1.01	**0.20**	1.82	0.13	0.13	2.02
ASTER$^+$	**2.69**	**1.38**	0.12	**3.92**	**1.15**	**0.54**	**3.50**
spectral RK4	0.94	0.47	0.16	0.38	0.20	0.09	0.11
spectral ETD-RK4	1.44	0.47	0.16	0.38	0.36	0.09	0.12
MUSCL (FV)	1.06	0.09	0.08	0.12	0.23	0.01	0.01

Table 6: Evaluation of ASTER-SOLVER. Relative L^2 error against the closed-form solution. The tensor-network structure is a Born draw from the random-circuit sampler, selected on training and validation trajectories only.

	heat2d			advdiff2d			reactdiff2d			aniso2d		
N	8	16	32	8	16	32	8	16	32	8	16	32
ASTER-SOLVER	**0.23**	**0.183**	**0.228**	1.23	**0.462**	**0.507**	**0.059**	**0.0759**	**0.145**	**0.0798**	**0.0622**	**0.117**
FNO	0.429	0.399	1.16	1	0.51	0.748	0.458	0.474	0.854	0.269	0.478	0.756
DeepONet	0.997	0.992	1	**1**	0.998	0.999	0.991	0.999	0.997	0.991	0.992	0.995
latent NODE	0.722	0.866	0.887	1.12	0.938	0.965	0.678	0.759	0.792	0.683	0.742	0.778
	hyper2d			burgers2d			schrod2d			wave2d		
N	8	16	32	8	16	32	8	16	32	8	16	32
ASTER-SOLVER	**0.0689**	**0.081**	**0.145**	**0.0188**	**0.0473**	**0.0758**	0.639	0.529	**0.418**	1.04	**0.574**	**0.518**
FNO	0.486	0.429	0.58	0.235	0.118	0.148	**0.409**	**0.319**	0.565	2.63	3.67	2.07
DeepONet	0.992	0.992	0.996	1	1	1	0.993	0.995	0.998	**1.01**	1	0.999
latent NODE	0.688	0.756	0.785	0.557	0.377	0.341	0.989	0.995	0.999	1.47	1.6	1.17
	convTGV2d			tgvhi2d			stokes2d			linkdv2d		
N	8	16	32	8	16	32	8	16	32	8	16	32
ASTER-SOLVER	**0.0108**	**0.00838**	**0.023**	**0.0114**	**0.00795**	**0.0289**	**0.00532**	**0.00426**	**0.00337**	0.787	div	div
FNO	0.264	0.315	0.33	0.271	0.318	0.304	0.0721	0.0505	0.057	**0.563**	**0.27**	**0.475**
DeepONet	0.549	0.572	0.589	0.564	0.591	0.595	0.97	0.983	0.987	0.993	1	0.998
latent NODE	0.071	0.0871	0.125	0.07	0.0835	0.0993	0.101	0.147	0.276	0.863	0.945	0.95

B.4 ON THE ACCURACY-DENSITY METRIC

The accuracy density $\eta = 100\log_{10}(1/\varepsilon)/S$ used as a secondary summary in §4 is our own construction and carries three properties a reader should hold in mind. First, it is non-positive whenever $\varepsilon \ge 1$, so it orders only methods that are at least as good as the zero predictor. Second, it divides by S without a floor, so it rewards a small denominator superlinearly and can prefer a much less accurate method that stores a little less; the ranking of `heat2d` in Table 5(c), where ASTER leads spectral ETD-RK4 despite being five orders of magnitude less accurate, is an instance. Third, S is not the same kind of quantity across the rows: for a tensor network it is the number of fitted core entries, and for a spectral or finite-volume solver it is the full field $N^d n_c$, which is a memory footprint rather than a parameter budget. The matched-accuracy comparison of Table 7 avoids all three

Table 7: Matched-accuracy comparison against spectral RK4, derived from Table 5(a) and (b). *Matched* marks the problems where the two relative errors agree to three significant figures. The ratio is storage of spectral RK4 divided by storage of ASTER$^+$; values above 1 favour ASTER$^+$.

	heat2d	advdiff2d	burgers2d	convTGV2d	heat3d	advdiff3d	ABC3d
Errors matched to 3 s.f.	yes	yes	yes	yes	yes	yes	no
ε (ASTER$^+$)	2.21×10^{-10}	1.52×10^{-5}	5.44×10^{-4}	1.34×10^{-8}	9.71×10^{-9}	2.92×10^{-4}	6.58×10^{-14}
S (ASTER$^+$)	359	350	2752	201	700	650	377
S (spectral RK4)	1024	1024	2048	2048	4096	4096	12288
Storage ratio	**2.9**×	**2.9**×	0.7×	**10.2**×	**5.9**×	**6.3**×	**32.6**×

Table 8: Comparison with existing tensor network structure search methods

Method	$N = 16$	$N = 32$	$N = 64$	$N = 128$	$N = 256$	wins
TNGA (Li & Sun, 2020)	**44**	**88**	175	350	701	16/35
TNLS (Li et al., 2022)	384	768	1536	3072	6144	0/35
TnALE (Li et al., 2023)	896	1792	3584	7168	14336	0/35
ASTER$^+$	58	101	**162**	**221**	**272**	27/35

issues by fixing ε and reading off S, and it is the comparison we ask readers to weigh. We report η because it summarises seven problems compactly, not because it is a standard of the field.

B.5 PER-PROBLEM SAMPLER ABLATION

Measured against the balanced tree with blocked ordering in Table 9, the accuracy-density gain of the quantum sampler is 14% on heat2d (2.10 against 1.84), 10% on advdiff2d (1.01 against 0.92), 5% on burgers2d (0.20 against 0.19), 49% on convTGV2d (1.82 against 1.22), 0% on heat3d (0.13 against 0.13), 44% on advdiff3d (0.13 against 0.09) and 39% on ABC3d (2.02 against 1.45). The uniform classical sampler, by contrast, is behind the same balanced tree on heat2d (0.39 against 1.84), advdiff2d (0.38 against 0.92) and advdiff3d (0.08 against 0.09), and level on burgers2d (0.19). Against the uniform sampler the quantum sampler attains smaller S on four of seven problems, ties on heat3d, and is behind on η for heat3d (0.13 against 0.14) and on S for burgers2d (556 against 512). All of these are single-run differences and several are within a plausible seed-to-seed range, which is the reason we do not treat them as establishing a quantum effect.

Table 9: Ablation study comparing the quantum sampler with classical uniform sampling strategies. Each cell is the storage S and the accuracy density η of the same pipeline under a different source for the structure. Per column, **smallest** S and **largest** η are highlighted. The last row is the accuracy-density gain of the quantum sampler over the hand-chosen structure the original benchmark used.

	$N = 32$ (2D)				$N = 16$ (3D)		
Structure *(S / η)*	heat2d	advdiff2d	burgers2d	convTGV2d	heat3d	advdiff3d	ABC3d
ASTER (quantum sampler)	**460**/**2.10**	**476**/**1.01**	556/**0.20**	**432**/**1.82**	**644**/0.13	680/**0.13**	**649**/**2.02**
Classical Uniform Sampling	556/0.39	556/0.38	**512**/0.19	472/1.67	**644**/**0.14**	**620**/0.08	697/1.88
Balanced Tree / blocked ordering	524/1.84	524/0.92	648/0.19	648/1.22	772/0.13	772/0.09	913/1.45
Balanced Tree / Morton ordering	524/0.19	524/0.18	648/0.13	648/1.22	772/0.10	772/0.08	913/1.45

C SUPPLEMENTAL DETAILS OF ALGORITHMS

Throughout, τ has m leaves, $2m-1$ nodes and $2m-2$ edges, of which $m-2$ are internal; $\ell(v)$ is the set of leaves below v; (B_e, A_e) is the bipartition of the leaves induced by cutting edge e; and $d_1, \ldots, d_m$ are the leaf dimensions under the ordering π.

Input: u_0; structure (τ, π); tolerance ε; cap $\chi_{\max}$; steps n
1: ★ $\chi \leftarrow \textsc{ToleranceRanks}(u_0, \tau, \pi, \varepsilon, \chi_{\max})$ ▷ adequate from step *zero*, not grown
2: $G \leftarrow \mathrm{HSVD}_{(\tau,\pi,\chi)}(\mathcal{Q}u_0)$
3: **for** $k = 0, \ldots, n-1$ **do**
4: ★ $G \leftarrow \textsc{ProjectedStep}(G, F, \Delta t)$ ▷ Algorithm 2
5: ★ $\hat{u} \leftarrow \mathcal{Q}^{-1}G + \Delta t\, F(\mathcal{Q}^{-1}G)$ ▷ *unprojected* predictor
6: ★ $\chi \leftarrow \textsc{ToleranceRanks}(\hat{u}, \tau, \pi, \varepsilon, \chi_{\max})$
7: $G \leftarrow \mathrm{HSVD}_{(\tau,\pi,\chi)}(\mathcal{Q}\,\mathcal{Q}^{-1}G)$ ▷ re-truncate to the new ranks
8: **end for**
9: **return** $\mathcal{Q}^{-1}G$

Algorithm 3: ASTER^{+}: tolerance-driven rank adaptation

Input: field u; tree τ; ordering π; tolerance ε; cap $\chi_{\max}$
Output: $\chi : E(\tau) \to \mathbb{N}$, feasible, adequate for u at tolerance ε
1: $X \leftarrow \mathcal{Q}u$
2: **for all** $e \in E(\tau)$ **do**
3: $(B_e, A_e) \leftarrow$ bipartition at e; $\sigma \leftarrow \mathrm{svdvals}\big(\mathrm{Mat}_{B_e}(X)\big)$
4: ★ $r_e \leftarrow \#\big\{i : \sigma_i > \varepsilon \cdot \max(\sigma_1, 10^{-300})\big\}$ ▷ numerical rank at relative ε
5: $f_e \leftarrow \min\big\{\chi_{\max},\ \prod_{i\in B_e} d_i,\ \prod_{i\in A_e} d_i\big\}$
6: $\chi_e \leftarrow \min\big\{\max(r_e, 1),\ f_e\big\}$
7: **end for**
8: **return** χ

Algorithm 4: TOLERANCERANKS: bond dimensions from a relative tolerance.

Input: bitstring $b^s \in \{0,1\}^{2m-1}$ of Hamming weight $m-1$
Output: rooted binary tree τ with m leaves
1: $n \leftarrow 2m-1$
2: Assert $|b^s| = m-1$
3: **for** $s = 0, \ldots, n-1$ **do**
4: $w \leftarrow \sigma^s(b^s)$ ▷ cyclic rotation by s
5: $p \leftarrow 0$; $ok \leftarrow$ TRUE
6: **for** $k = 1, \ldots, n-1$ **do** ▷ *proper* prefixes only
7: $p \leftarrow p + (2w_k - 1)$ ▷ $1 \mapsto +1$ internal, $0 \mapsto -1$ leaf
8: **if** $p < 0$ **then**
9: $ok \leftarrow$ FALSE
10: **end if**
11: **end for**
12: **if** ok **then**
13: **return** $\textsc{Preorder}(w)$
14: **end if**
15: **end for**

16: **function** $\textsc{Preorder}(w)$ ▷ consumes w left to right
17: **if** next symbol of w is 1 **then**
18: $a \leftarrow \textsc{Preorder}(w)$; $b \leftarrow \textsc{Preorder}(w)$
19: **return** internal node with children (a, b)
20: **end if**
21: **return** leaf
22: **end function**

Algorithm 5: CYCLELEMMADECODE: shape register $\to$ Catalan tree.

Input: bitstring $b^\pi \in \{0,1\}^{n_\pi}$, $n_\pi = \sum_{j=0}^{m-2} \lceil \log_2(m-j) \rceil$
Output: a permutation π of $(0, \ldots, m-1)$; **never fails**

$A \leftarrow (0, 1, \ldots, m-1)$; $\pi \leftarrow ()$; $i \leftarrow 1$
for $j = 0, \ldots, m-2$ **do**
  $k_j \leftarrow \max\{1, \lceil \log_2(m-j) \rceil\}$ ▷ digit j is a contiguous k_j-qubit block
  $c_j \leftarrow \mathrm{Int}_2\left(b^\pi_i, \ldots, b^\pi_{i+k_j-1}\right)$; $i \leftarrow i + k_j$ ▷ a short read decodes to 0
  ★ $\pi \leftarrow \pi \frown A_{c_j \bmod |A|}$; remove that entry from A ▷ total, but biased: Proposition 1
end for
$\pi \leftarrow \pi \frown A_0$ ▷ the last mode is forced
return π

Algorithm 6: LEHMERDECODE: permutation register → mode ordering.

Input: bitstring $b^r \in \{0,1\}^{n_r}$, $n_r = b\,(m-2)$; tree τ; ordering π; cap $\chi_{\max}$
Output: $\chi : E(\tau) \to \mathbb{N}$, feasible on every edge; **never fails**

for all $e \in E(\tau)$ **do** ▷ maximal feasible rank first
  $(B_e, A_e) \leftarrow$ bipartition of τ at e
  $\chi_e \leftarrow \min\left\{\chi_{\max}, \prod_{i \in B_e} d_i, \prod_{i \in A_e} d_i\right\}$
end for
$i \leftarrow 1$
for all $e \in E_{\mathrm{int}}(\tau)$ **do** ▷ $b = 3$ qubits per internal edge
  $v_e \leftarrow \mathrm{Int}_2\left(b^r_i, \ldots, b^r_{i+b-1}\right)$; $i \leftarrow i + b$
  ★ $\chi_e \leftarrow \min\left\{\chi_e, \max(1, 2^{v_e})\right\}$ ▷ *exponent* encoding: b qubits span $\{1, 2, \ldots, 2^{2^b-1}\}$
end for
return χ

Algorithm 7: CLIPFEASIBLE: rank register → feasible bond dimensions.

Input: state u; structure (τ, π); cheap cap χ_{cheap}
Output: $\tilde{F}(u)$, and implicitly the closure target $F(u) - \tilde{F}(u)$

$\mathcal{L} \leftarrow \mathrm{LINEAR}(u)$
★ $\mathcal{N} \leftarrow F(u) - \mathcal{L}$
★ $\mathcal{N} \leftarrow \mathcal{Q}^{-1}\,\mathrm{HSVD}_{\chi_{\text{cheap}}}(\mathcal{Q}\mathcal{N})$ ▷ re-truncate the product: the sole source of $F - \tilde{F}$
return $\mathcal{L} + \mathcal{N}$

Algorithm 8: Surrogate modelling of right-hand side $\tilde{F}$

Input: state u; time t; PDE exposing $F(\cdot, t)$ and, when available, its linear part $\mathcal{L}(\cdot)$; structure (τ, π); cheap cap χ_{cheap}
Output: $\tilde{F}(u, t)$, and the closure target $F(u, t) - \tilde{F}(u, t)$

$\chi_e^{\text{cheap}} \leftarrow \min\left\{\chi_{\text{cheap}}, \prod_{i \in B_e} d_i, \prod_{i \in A_e} d_i\right\}$ for every $e \in E(\tau)$
if the PDE exposes a linear part **then**
  $\mathcal{L} \leftarrow \mathcal{L}(u)$ ▷ diffusion and the Leray projection: mode-wise, rank-preserving, exact
  ★ $\mathcal{N} \leftarrow F(u, t) - \mathcal{L}$ ▷ advective part only
  ★ $\tilde{F} \leftarrow \mathcal{L} + \mathcal{Q}^{-1}\mathrm{HSVD}_{(\tau, \pi, \chi^{\text{cheap}})}\left(\mathcal{Q}\mathcal{N}\right)$
else
  ★ $\tilde{F} \leftarrow \mathcal{Q}^{-1}\mathrm{HSVD}_{(\tau, \pi, \chi^{\text{cheap}})}\left(\mathcal{Q}F(u, t)\right)$ ▷ cruder fallback; the branch taken is recorded, never inferred
end if
return $\tilde{F}$

Algorithm 9: SURROGATESPLIT: the linear/nonlinear split behind $\tilde{F}$.

Input: $X \in \mathcal{H}$; structure $\mathcal{S} = (\tau, \pi, \chi)$; leaf dimensions d
Output: $G \in \mathcal{M}_{\mathcal{S}}$; realised ranks k_e; discarded masses t_e

for all $v \in V(\tau) \setminus \{\text{root}\}$ **do** ▷ one thin SVD per edge
  $e \leftarrow (\text{parent}(v), v)$; $(B_e, A_e) \leftarrow$ bipartition at e
  $U\Sigma W^\top \leftarrow \text{SVD}\big(\text{Mat}_{B_e}(X)\big)$ ▷ rows indexed by the leaves below v
  $k_e \leftarrow \min\{\chi_e,\ \text{rank Mat}_{B_e}(X)\}$; $U_v \leftarrow U[:, 1{:}k_e]$
  $t_e \leftarrow \sum_{i>k_e} \sigma_i(e)^2$ ▷ logged; $(\sum_e t_e)^{1/2}$ is the truncation error of the step
end for
for all $v \in V(\tau)$ **do** ▷ cores by projection, not by a second decomposition
  if v is a leaf **then**
    $C_v \leftarrow U_v$ ▷ shape (d_v, k_e)
  else
    $(a, b) \leftarrow \text{children}(v)$
    if $v = \text{root}$ **then**
      $C_v \leftarrow (U_a \otimes U_b)^\top \text{vec}_{\ell(a)\frown\ell(b)}(X)$, reshaped $(k_a, k_b, 1)$
    else
      $C_v \leftarrow (U_a \otimes U_b)^\top \Pi\, U_v$, reshaped (k_a, k_b, k_v) ▷ Π reorders the rows of U_v from $\ell(v)$ into $\ell(a) \frown \ell(b)$
    end if
  end if
end for
return $G = \big(\tau, \{k_e\}, \{C_v\}\big), \{t_e\}$

Algorithm 10: $\text{HSVD}_{\mathcal{S}}$: hierarchical-SVD truncation onto $\mathcal{M}_{\mathcal{S}}$.

Input: $G \in \mathcal{M}_{\mathcal{S}}$; ambient tensor $V \in \mathcal{H}$
Output: $\text{P}V \in \text{T}_G\mathcal{M}_{\mathcal{S}}$

for all $e \in E(\tau)$ **do** ▷ edge frames; cached across steps, tree-only plan
  $Q_e^{\downarrow} \leftarrow \text{orth}\,\big(\text{mat}(\textsc{Subtree}(G, e))\big)$; $Q_e^{\uparrow} \leftarrow \text{orth}\,\big(\text{mat}(\textsc{Env}(G, e))\big)$ ▷ thin QR, first χ_e columns
end for
$Y \leftarrow 0$
for all $v \in V(\tau)$ **do** ▷ $+\sum_v \text{P}_v$, all $2m-1$ nodes
  $W \leftarrow V$
  for all (e, S, side) incident to v **do** ▷ $S =$ *far*-side leaf group
    ★ $W \leftarrow \textsc{ApplyOnLeaves}(W, S, Q_e^{\text{side}})$
  end for
  $Y \leftarrow Y + W$
end for
for all $e \in E(\tau)$ **do** ▷ $-\sum_e \text{P}_e$, all $2m-2$ edges
  $W \leftarrow \textsc{ApplyOnLeaves}(V, B_e, Q_e^{\downarrow})$
  $W \leftarrow \textsc{ApplyOnLeaves}(W, A_e, Q_e^{\uparrow})$
  $Y \leftarrow Y - W$
end for
return Y

Algorithm 11: TANGENTPROJECT: the Dirac–Frenkel projector $\text{P}_{\text{T}_G\mathcal{M}_{\mathcal{S}}}$ (Lubich et al., 2013; Haegeman et al., 2016).

Input: tensor X; leaf group S; orthonormal basis $Q \in \mathbb{R}^{d_S \times k}$, $d_S = \prod_{i \in S} d_i$
Output: X with $QQ^\top$ applied to the modes in S

if $S = \emptyset$ **then**
  return X
end if
$\rho \leftarrow \text{sort}(S) \frown \big(\{1, \ldots, m\} \setminus S\big)$
$Y \leftarrow \text{reshape}\big(\text{transpose}(X, \rho),\ (d_S, \cdot)\big)$ ▷ $\textsc{Unfold}(X, S)$
★ $Y \leftarrow Q\,\big(Q^\top Y\big)$ ▷ $\mathcal{O}(k\,n)$; forming $QQ^\top$ costs $\mathcal{O}(d_S\,n)$
return $\text{transpose}\big(\text{reshape}(Y, \text{dims}), \rho^{-1}\big)$ ▷ $\textsc{Fold}(Y, S)$

Algorithm 12: APPLYONLEAVES

D PROOF OF THEOREM

Convention. We set index words from 1. For $\boldsymbol{b} \in \{0,1\}^n$, we write $\sigma^s(\boldsymbol{b})$ for the left rotation $(b_{s+1}, \dots, b_n, b_1, \dots, b_s)$, and define the walk

$$S_0(\boldsymbol{b}) = 0, \qquad S_j(\boldsymbol{b}) = \sum_{i=1}^{j} \big(2b_i - 1\big) = \#\{1\text{'s in } b_{1:j}\} - \#\{0\text{'s in } b_{1:j}\}, \quad 1 \le j \le n. \tag{8}$$

Following Definition 1, $\boldsymbol{b}$ is a *Łukasiewicz word* when $S_j(\boldsymbol{b}) \ge 0$ for every *proper* prefix, $1 \le j \le n-1$. For $\boldsymbol{b} \in B_m$, we have $n = n_s = 2m-1$ and $S_n(\boldsymbol{b}) = (m-1) - m = -1$, so the walk ends one step below where it started. The constraint is on the proper prefixes only, and it is exactly what IsLukasiewicz tests. Reading such a word in preorder, emitting an internal node on a 1 and a leaf on a 0, is a bijection onto $\mathcal{T}_m$.

Lemma 1 *Let $\boldsymbol{b} \in B_m$, $n = 2m-1$, and let $S_j = S_j(\boldsymbol{b})$ be as in equation 8. Then $\sigma^s(\boldsymbol{b})$ is a Łukasiewicz word if and only if*

$$S_s \le S_r \text{ for all } r > s, \quad \text{and} \qquad S_s < S_r \text{ for all } r < s, \quad r, s \in \{0, \dots, n-1\}, \tag{9}$$

that is, if and only if s is the first *index at which $\min_{0 \le j \le n-1} S_j$ is attained.*

Proof: Extend S to all of $\mathbb{Z}_{\ge 0}$ by reading $\boldsymbol{b}$ periodically. Since one period contributes $S_n = -1$, the extension satisfies

$$S_{j+n} = S_j - 1 \qquad \text{for every } j \ge 0. \tag{10}$$

The walk of the rotated word is a shifted copy of the same extension: for $1 \le j \le n$, $S_j\big(\sigma^s(\boldsymbol{b})\big) = S_{s+j} - S_s$. Hence, $\sigma^s(\boldsymbol{b})$ is Łukasiewicz precisely when

$$S_{s+j} \ge S_s \text{ for } j = 1, \dots, n-1; \tag{11}$$

i.e. when S_s is the minimum of S over the window $\{s, s+1, \dots, s+n-1\}$ of one full period. That window contains each residue exactly once. The indices $s, \dots, n-1$ contribute the values S_r with $r \ge s$, and the indices $n, \dots, s+n-1$ contribute, by equation 10, the values $S_r - 1$ with $r < s$. The window condition therefore reads $S_s \le S_r$ for $r > s$ and $S_s \le S_r - 1$, equivalently $S_s < S_r$, for $r < s$, which is equation 9. Conversely equation 9 gives the window condition, so the two are equivalent.

Proof of Theorem 1. *(i) Totality.* The set $\{S_0, \dots, S_{n-1}\}$ is a finite set of integers, so its minimum is attained and the first index attaining it is unique. By Lemma 1 that index, and only that index, gives a Łukasiewicz rotation; write it $s^\star(\boldsymbol{b})$. Hence, $\mathcal{D}_\tau$ is defined on all of B_m and never rejects. Its value is the tree obtained by reading $\sigma^{s^\star}(\boldsymbol{b})$ in preorder, which has m leaves because the word has $m-1$ ones and m zeros. *Surjectivity*: given $\tau \in \mathcal{T}_m$, its preorder word w_τ has $m-1$ ones among $2m-1$ letters, so $w_\tau \in B_m$, and w_τ is Łukasiewicz, so $s^\star(w_\tau) = 0$ and $\mathcal{D}_\tau(w_\tau) = \tau$. *(ii) Uniform fibres.* First, every rotation orbit in B_m is free. Suppose $\sigma^d(\boldsymbol{b}) = \boldsymbol{b}$ for some $0 < d < n$. The stabiliser of $\boldsymbol{b}$ is a subgroup of $\mathbb{Z}_n$, hence generated by $g = \gcd(d, n)$, so $\boldsymbol{b}$ consists of n/g repetitions of its first g letters and therefore (n/g) divides $|\boldsymbol{b}| = m-1$. It also divides $n = 2m-1$, so it divides

$$\gcd(2m-1, m-1) = \gcd\big(m-1, (2m-1) - 2(m-1)\big) = \gcd(m-1, 1) = 1, \tag{12}$$

forcing $g = n$ and contradicting $g \le d < n$. Hence every orbit has exactly $n = 2m-1$ elements. Next, the fibres of $\mathcal{D}_\tau$ are exactly the orbits. Rotation preserves the Hamming weight, so orbits lie in B_m; two words in one orbit have the same set of rotations, hence by (i) the same Łukasiewicz representative and the same image. Conversely, if $\mathcal{D}_\tau(\boldsymbol{b}) = \mathcal{D}_\tau(\boldsymbol{b}') = \tau$ then, the preorder reading being a bijection, both have w_τ as their Łukasiewicz rotation, so both lie in the orbit of w_τ. Therefore $|\mathcal{D}_\tau^{-1}(\tau)| = 2m-1$ for every $\tau \in \mathcal{T}_m$. Counting B_m in two ways now gives, using surjectivity from (i),

$$\binom{2m-1}{m-1} = |B_m| = \sum_{\tau \in \mathcal{T}_m} \big|\mathcal{D}_\tau^{-1}(\tau)\big| = (2m-1)\,|\mathcal{T}_m| = (2m-1)\,C_{m-1}, \tag{13}$$

which is also the identity $\binom{2m-1}{m-1} = \frac{(2m-1)!}{(m-1)!\,m!} = (2m-1) \cdot \frac{1}{m}\binom{2m-2}{m-1}$. Because the fibres are equinumerous, the pushforward of the uniform law on B_m is exactly uniform on $\mathcal{T}_m$: the encoding

contributes no bias of its own, and any structure in $p_\theta(\tau)$ is attributable to $U(\Theta_{\mathrm{q}})$. This statement concerns the shape register only; the permutation register is treated in Proposition 1. *(iii) Zero rejection.* On a computational-basis state, $\hat{n}\,|\boldsymbol{b}\rangle = |\boldsymbol{b}|\,|\boldsymbol{b}\rangle$, since $\frac{1}{2}(I - Z)$ annihilates $|0\rangle$ and fixes $|1\rangle$. Its eigenspaces are therefore $\mathcal{H}_w = \mathrm{span}\{|\boldsymbol{b}\rangle : |\boldsymbol{b}| = w\}$, and $\mathcal{H}_{m-1}$ has computational-basis support exactly B_m. If $[\,U(\Theta^s_{\mathrm{q}}), \hat{n}\,] = 0$ then $U(\Theta^s_{\mathrm{q}})$ leaves every $\mathcal{H}_w$ invariant. The state is prepared as $\left|\boldsymbol{b}^{(0)}\right\rangle$ with $|\boldsymbol{b}^{(0)}| = m - 1$, hence $U(\Theta^s_{\mathrm{q}})\left|\boldsymbol{b}^{(0)}\right\rangle \in \mathcal{H}_{m-1}$ and

$$p_\theta(\boldsymbol{b}) = |\left\langle \boldsymbol{b}\,|\,U(\Theta^s_{\mathrm{q}})\,|\,\boldsymbol{b}^{(0)}\right|\rangle^2 = 0 \qquad \text{whenever } |\boldsymbol{b}| \neq m-1; \tag{14}$$

so $\mathrm{supp}\, p_\theta \subseteq B_m$. The hypothesis holds for the circuit of Figure 4a: R_z and CP are diagonal, and $\mathrm{fSim}(\theta, \phi)$ fixes $|00\rangle$, maps $\mathrm{span}\{|01\rangle, |10\rangle\}$ to itself and multiplies $|11\rangle$ by a phase, so each commutes with the local number operator and hence with $\hat{n}$. Combining with (i), every measurement outcome lies in the domain on which $\mathcal{D}_\tau$ is total: Algorithm 1 accepts with probability 1 and returns one structure per circuit execution. The argument uses unitarity of the evolution; under a noise channel that does not commute with $\hat{n}$ the support statement fails.

Bias of the permutation and rank decoders. Theorem 1 establishes that the shape decoder is unbiased. The permutation decoder is not, and the following makes the discrepancy exact.

Proposition 1 (Modulo bias of LEHMERDECODE) *Fix a digit $j \in \{0, \ldots, m-2\}$, write $a = |A| = m - j$ for the number of modes still unassigned, $k_j = \max\{1, \lceil \log_2 a \rceil\}$ for the width of the qubit block read at that digit, and $M = 2^{k_j}$. Suppose the block is uniform on $\{0, \ldots, M-1\}$. Then $a \leq M < 2a$, and line 5 of Algorithm 6 selects index $c \bmod a$ with probability*

$$\mathbb{P}[\,\textit{index } i\,] = \frac{1}{M}\big(1 + \mathbf{1}[\,i < M - a\,]\big), \qquad i \in \{0, \ldots, a-1\}, \tag{15}$$

so the selection is uniform if and only if a is a power of two, and otherwise the total-variation distance from uniform is

$$d_{\mathrm{TV}} = \frac{(M-a)\,(2a-M)}{a\,M}. \tag{16}$$

Proof. By definition of k_j we have $2^{k_j - 1} < a \leq 2^{k_j}$, hence $a \leq M < 2a$, so $\lfloor M/a \rfloor = 1$ and $M \bmod a = M - a =: r$. Reduction modulo a therefore maps exactly two of the M block values onto each of the indices $0, \ldots, r-1$ and exactly one onto each of the indices $r, \ldots, a-1$, which gives the stated probabilities; they sum to $(2r + (a-r))/M = (a+r)/M = 1$. For the distance,

$$d_{\mathrm{TV}} = \tfrac{1}{2}\Big[r\Big|\tfrac{2}{M} - \tfrac{1}{a}\Big| + (a-r)\Big|\tfrac{1}{M} - \tfrac{1}{a}\Big|\Big] = \tfrac{r}{2aM}\big[(2a-M) + (a-r)\big] = \tfrac{r\,(2a-M)}{aM}, \tag{17}$$

using $a - r = 2a - M$ in the last step, which is equation 16. It vanishes exactly when $r = 0$, that is when $a = M$ is a power of two. □

For $m = 10$, the digits with $a \in \{10, 9, 7, 6, 5, 3\}$ are biased and those with $a \in \{8, 4, 2\}$ are not; the largest single-digit distance is $d_{\mathrm{TV}} = 1/6$ at $a = 3$. Composing digits, the distance between the induced law on $\mathfrak{S}_m$ and the uniform law is bounded by the sum of the per-digit distances. Two remedies are available and we have not implemented either: decode the Lehmer code in factorial base by arithmetic decoding, which is total and measure preserving, or widen each block and use rejection-free interval mapping. We note that the bias is a property of the encoding and not of the circuit, so it applies equally to the classical uniform sampler of Table 9, and the ablation therefore compares like with like.

The rank decoder is likewise total but coarse. With $b = 3$ qubits per internal edge, Algorithm 7 reaches $\chi_e \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ before the feasibility clip. Under the cap $\chi_{\max} = 6$ of Table 5 the reachable set per edge is $\{1, 2, 4, 6\}$, and under $\chi_{\max} = 30$ of Table 8 it is $\{1, 2, 4, 8, 16, 30\}$. The rank register searches a geometric grid, not the feasible integer range, and the effective search is correspondingly narrower than the $\mathbb{N}^{m-2}$ of §3.1.