FLASHSINKHORN 2: BLOCK-SPARSE ENTROPIC OPTIMAL TRANSPORT ON GPU

Anonymous authors
Paper under double-blind review

ABSTRACT

Streaming GPU solvers for entropic optimal transport (EOT), such as FlashSinkhorn, avoid storing the dense kernel but still evaluate all $n \times m$ point pairs in every Sinkhorn iteration. We present **FlashSinkhorn 2** (FS2), a solver for squared-Euclidean cost on low-dimensional point clouds that solves large discrete EOT problems to a prescribed marginal residual on a single GPU by coupling two stages. A coarse stage solves on cell centroids, lifts the potentials to every point and, when a sampled marginal check rejects the lift, continues on the centroids, replacing most point-level updates. A block-sparse fine stage then removes the centroid error that coarse updates cannot. Its Morton-ordered blocks support screening and fused tensor-core execution, and a threshold set by the block masses bounds each omitted tile's contribution to every row and column. On synthetic benchmarks, FS2 reaches the target residual on all 32 problems and GeomLoss multiscale on 10. On one A100, FS2 solves discrete EOT between two 1.34×10^8-particle measures from a cosmological N-body simulation, at an entropic blur equal to the mean interparticle distance, to an all-particle marginal residual below 0.01 in under 2.5 hours. To our knowledge, it is the largest discrete EOT problem solved to this accuracy within hours.

1 INTRODUCTION

Optimal transport (OT) compares probability measures in applications ranging from generative modeling to cell dynamics and cosmological reconstruction (Arjovsky et al., 2017; Schiebinger et al., 2019; Frisch et al., 2002). Entropic regularization (Cuturi, 2013) yields a differentiable problem, entropic OT (EOT), solved by Sinkhorn iterations. Applications in computational biology, medical imaging, and cosmology now routinely produce point clouds with 10^5–10^8 points per measure (Bunne et al., 2024; Feydy et al., 2019a; Lévy et al., 2025). Solvers that store the dense $n \times m$ kernel exceed GPU memory beyond about 10^5 points. MOSCOT (Klein et al., 2025) couples a few hundred thousand cells per time point and suggests low-rank plans (Scetbon et al., 2021) for millions. Lévy et al. (2025) recover initial particle positions from a cosmological dataset with 1.34×10^8 target particles by semi-discrete transport, and note that a Sinkhorn iteration between unstructured point clouds costs quadratic time, "prohibiting very large-scale applications." Solvers that evaluate entries on the fly, GeomLoss (Feydy et al., 2019b) through KeOps (Charlier et al., 2021) and FlashSinkhorn (Ye et al., 2026) through one fused tiled kernel per update, avoid storing the $n \times m$ matrix but still perform $\mathcal{O}(nm)$ work per update.

At moderate regularization, much of this work is wasted, because the entropic plan concentrates on nearby pairs. Multiscale solvers first solve a coarser problem, which cuts the number of fine iterations, and then skip distant pairs whose contribution is bounded (Schmitzer, 2019) or estimated from the coarse potentials (Feydy et al., 2019a). GeomLoss multiscale combines both on the GPU but does not test the marginals it returns. Bringing a large problem to a prescribed residual with both ideas raises three practical issues. First, the solution of the coarser problem is only approximate, and checking whether it is accurate enough costs $\mathcal{O}(nm)$. Second, screening all point pairs to find the useful ones costs $\mathcal{O}(nm)$ as well. Third, the surviving pairs are distributed unevenly across rows, yet GPU tensor cores require fixed-size tiles.

We introduce **FlashSinkhorn 2** (FS2), which addresses these issues for squared-Euclidean costs in $d \leq 3$ by coupling a coarse stage with a block-sparse fine stage. FS2 first groups nearby points into

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.

cells and solves the much smaller problem between the cell centroids. It then lifts this solution to the original points and checks it on randomly drawn rows and columns instead of all nm pairs. If the check fails, FS2 refines the centroid solution further and, if the check still fails, removes the error of replacing points by centroids with sparse updates on the points. These updates skip negligible pairs by screening fixed-size blocks of nearby points instead of single points, and each block pair they keep maps onto whole tensor-core tiles. Every update runs FlashSinkhorn's streaming kernel, so FS2 stores block-pair indices but no kernel entries. Our contributions are:

- **Error guarantees.** Building on Schmitzer (2019), we bound each omitted tile's contribution to every row and column, and hence the full marginal residual. For the lift, we isolate the centroid error that only fine updates remove.
- **GPU co-design.** Cells and blocks are contiguous ranges of one Morton order. The fine stage screens block pairs, cutting the cost to $\mathcal{O}(nm/B^2)$, and runs each sparse half-step through FlashSinkhorn's fused kernel on the kept tiles.
- **Large-scale evaluation.** On one A100, FS2 solves discrete EOT between two 1.34×10^8-particle measures to a marginal residual below 0.01 in under 2.5 hours. To our knowledge, it is the largest discrete EOT problem solved to this accuracy within hours.

2 Background and Related Work

2.1 Entropic OT and FlashSinkhorn

For point clouds $\mathbf{X} = (x_i)_{i=1}^n$, $\mathbf{Y} = (y_j)_{j=1}^m$ in $\mathbb{R}^d$, strictly positive probability weights $\mathbf{a} \in \Delta^n$, $\mathbf{b} \in \Delta^m$, and squared Euclidean cost $C_{ij} = \|x_i - y_j\|^2$, EOT solves

$$\mathrm{OT}_\varepsilon(\mathbf{a}, \mathbf{b}) = \min_{\mathbf{P} \in \Pi(\mathbf{a}, \mathbf{b})} \langle \mathbf{P}, \mathbf{C} \rangle + \varepsilon \,\mathrm{KL}(\mathbf{P} \,\|\, \mathbf{a} \otimes \mathbf{b}), \qquad \varepsilon > 0, \tag{1}$$

where $\Pi(\mathbf{a}, \mathbf{b}) = \{\mathbf{P} \geq 0 : \mathbf{P}\mathbf{1} = \mathbf{a}, \mathbf{P}^\top \mathbf{1} = \mathbf{b}\}$. The optimal plan is $P_{ij} = a_i b_j \exp((f_i + g_j - C_{ij})/\varepsilon)$, with dual potentials $\mathbf{f} \in \mathbb{R}^n$, $\mathbf{g} \in \mathbb{R}^m$. With $\mathrm{LSE}_j(z_j) = \log \sum_j \exp(z_j)$ denoting the log-sum-exp over the index j, log-domain Sinkhorn uses the half-steps

$$f_i \leftarrow -\varepsilon \,\mathrm{LSE}_j[(g_j - C_{ij})/\varepsilon + \log b_j], \qquad g_j \leftarrow -\varepsilon \,\mathrm{LSE}_i[(f_i - C_{ij})/\varepsilon + \log a_i]. \tag{2}$$

Alternating updates apply the half-steps sequentially. Symmetric updates compute both from the same previous potentials, then average each updated potential with its previous value (Feydy, 2020). At converged balanced potentials, the source-coordinate gradient of EOT is $2a_i(x_i - T_\varepsilon(x_i))$, where $T_\varepsilon(x_i)$ is the barycentric projection of x_i (Li et al., 2025).

FlashSinkhorn (FS) rewrites each half-step as a row-wise LSE reduction of a biased dot-product score matrix (Ye et al., 2026). For squared Euclidean cost, $C_{ij} = \|x_i\|^2 + \|y_j\|^2 - 2x_i^\top y_j$, so with shifted potentials $\hat{f}_i = f_i - \|x_i\|^2$ and $\hat{g}_j = g_j - \|y_j\|^2$ the first half-step becomes $\hat{f}_i \leftarrow -\varepsilon \,\mathrm{LSE}_j\big[\big(2x_i^\top y_j + \hat{g}_j\big)/\varepsilon + \log b_j\big]$, and the second exchanges the points and weights. A fused Triton kernel (Tillet et al., 2019) computes each half-step in one pass over tiles, fixed-size submatrices of the score matrix. For each tile it forms the dot products $x_i^\top y_j$ as a matrix product (GEMM), adds the bias, and folds the tile's scores into a running LSE (Milakov & Gimelshein, 2018). The matrix products run on TF32 tensor cores, which offer higher throughput than FP32 at reduced mantissa precision. Scores and running statistics stay on chip and the kernel writes only the updated potentials to high-bandwidth memory (HBM), so FlashSinkhorn uses $\mathcal{O}((n+m)d)$ working memory without materializing the $n \times m$ score matrix. KeOps (Charlier et al., 2021) evaluates the same reduction pair by pair in scalar FP32 arithmetic and forms no matrix product, so it does not use the tensor cores. Each dense half-step still evaluates all nm point pairs. The reduction streams over tiles, so a block mask can select the tiles the kernel visits, as in block-sparse variants of FlashAttention (Dao et al., 2022; Dong et al., 2025).

2.2 Multiscale Sinkhorn and Related Approaches

The remaining question is which tiles to skip. Prior multiscale Sinkhorn methods answer it through spatial coarsening and potential-based truncation. In the plan, the factor $e^{-C_{ij}/\varepsilon}$ falls to $1/e$ at $\|x_i - y_j\| = \sqrt{\varepsilon}$, so $\sqrt{\varepsilon}$ is the spatial scale of the regularization, called the *blur*. In a locally uniform cloud, a ball whose radius is proportional to the blur holds few of the m target points when the blur

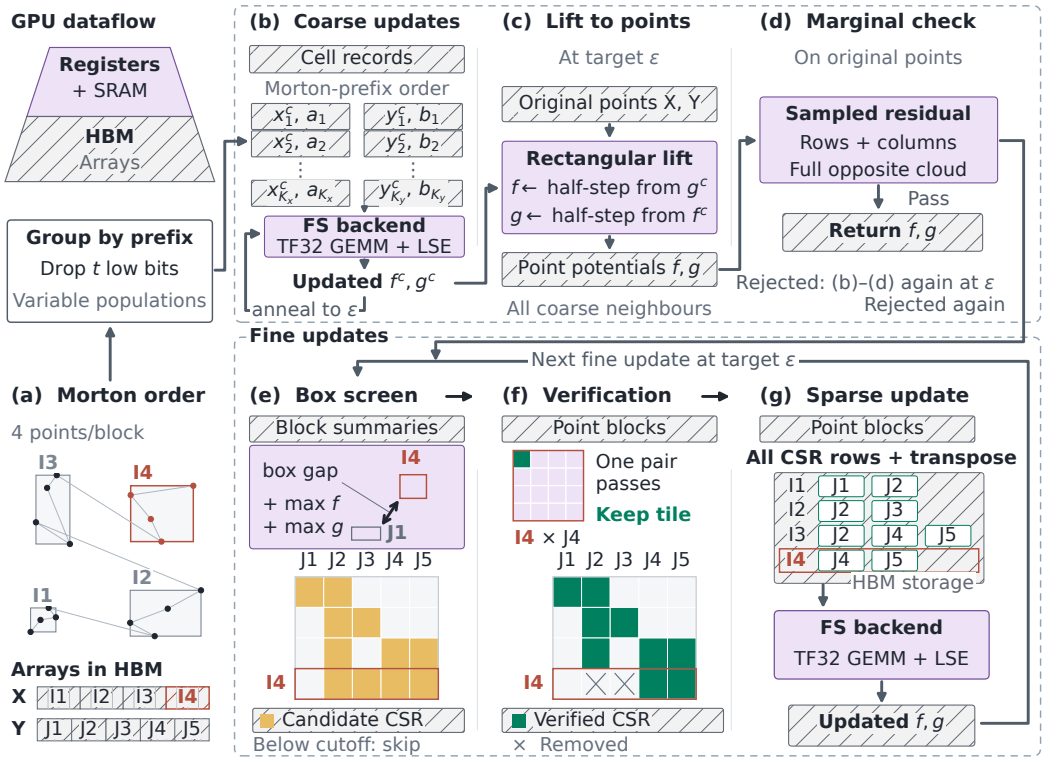


Figure 1: **FlashSinkhorn 2 overview** (Algorithm 1), drawn with $B{=}4$. Panel (a) is the Morton order both stages share. Panels (b)–(d) are the coarse stage (Section 3.1) and panels (e)–(g) the fine stage (Section 3.2). Red outlines trace source block I4. Yellow and green mark candidate and admitted tiles. Purple denotes on-chip work, and hatched grey denotes HBM storage.

spans a few nearest-neighbour spacings, so most point pairs carry negligible mass and screening pays off. The number of points in such a ball grows exponentially with the dimension d, as the d-th power of the blur over the spacing, and the methods below, including ours, therefore target $d \leq 3$.

Schmitzer (2019) combines coarsening and potential-based truncation with ε-annealing. It truncates the kernel at the current potentials, keeping only entries above a threshold, and bounds the resulting error in the primal–dual gap. To find these entries without testing all of them, it descends a hierarchy of cells and discards each cell pair that a bound shows to be negligible. GeomLoss multiscale (Feydy et al., 2019a; Feydy, 2020) anneals ε on weighted cell centroids and lifts the coarse potentials to the points without checking them. Before the fine updates, it builds a cell-pair mask from the last coarse potentials, keeping a pair of cells when their centroid cost is below the sum of their potentials plus a default margin of 5ε. KeOps (Charlier et al., 2021) then evaluates the fine updates on the kept pairs. Because the centroid cost does not bound the point-pair costs from below, this mask approximates Schmitzer's truncation but carries no bound on the mass it drops. Domain-decomposition methods solve local entropic subproblems in parallel, on CPUs (Bonafini & Schmitzer, 2021) or, for regular grids, on GPUs (Medina & Schmitzer, 2025).

3 FlashSinkhorn 2: Algorithm and Analysis

FS2 rests on two observations. First, coarse updates are cheap but cannot remove the error of replacing points by centroids. FS2 therefore stays on the centroids until coarse updates stall. It switches to fine updates (Sinkhorn updates on the points themselves) only if a sampled check of the lifted potentials fails (Section 3.1). Second, most block pairs carry negligible mass. Fine updates skip them using a threshold set by the block masses, with a bounded error (Section 3.2). Figure 1 illustrates the dataflow and Algorithm 1 summarizes the solver. Appendix C gives the proofs and Appendix D the implementation details.

Algorithm 1 FlashSinkhorn 2 forward solver. Appendix D.2 gives the full stopping rules.

Require: Source $\mathbf{X}$, target $\mathbf{Y}$, probability weights $\mathbf{a}, \mathbf{b}$, regularization ε, tolerance τ
Ensure: pointwise potentials (f, g) and the sampled-check outcome

Morton-sort, choose B and t, form blocks, boxes, and cells $(x_I^c, a_I), (y_J^c, b_J)$ *// (a)*
anneal on centroids with FS streaming kernels to reach ε *// (b)*
lift: f_i, g_j by one half-step of (2) against the coarse records *// (c)*
sample $i \sim a_i$, $j \sim b_j$, then **if** both means plus three standard errors $< \tau$, **return** *// (d)*
continue coarse updates at ε until they stall, lift, then **if** the check passes, **return**
for each fine update within budget **do**
  screen: keep (I, J) with $d^2_{\min}(I, J) < f_I^\vee + g_J^\vee - \varepsilon\, \rho_{IJ}$ *// (e)*
  verify: $\mathcal{A} \leftarrow \{(I, J) : m_{IJ} > \rho_{IJ}\}$ in CSR form *// (f)*
  update: both sparse half-steps on $\mathcal{A}$ from the same (f, g), then average *// (g)*
  if a scheduled sampled check passes, **return**
end for
return potentials and check outcome

Both stages need spatially compact groups of points stored contiguously for coalesced GPU access. We sort the points by Morton key (Figure 1a), which interleaves the bits of the coordinates, so points that share a key prefix are contiguous and lie in one axis-aligned box. Two partitions of the sorted order serve the two stages: cells, whose points share all but the lowest t key bits, for the coarse stage of Section 3.1, and blocks of B consecutive points for the fine stage of Section 3.2. We replace each cell by its total mass and mass-weighted centroid, giving K_x source records (x_I^c, a_I) and K_y target records (y_J^c, b_J). A finer partition reduces the distance from points to their centroids, but increases the $K_x K_y$ pairs evaluated by each coarse update. The automatic policy chooses $B \in \{64, 128, 256\}$ by comparing the estimated block-summary memory with the L2 budget, and takes the smallest t whose coarse state fits the same budget, with at most twice as many cells as blocks.

3.1 Coarse Sinkhorn Updates

The coarse stage solves the EOT problem between the cell records. We anneal with FlashSinkhorn's streaming TF32 kernels (Ye et al., 2026) (Figure 1b), running coarse updates along a decreasing ε-schedule, each from the potentials of the previous one. When the schedule first reaches the requested ε, we *lift* the coarse potentials to the original points (Figure 1c). The lifted f_i is the first half-step of (2) with the coarse target records (y_J^c, b_J, g_J^c) in place of (y_j, b_j, g_j), and g_j is the second with (x_I^c, a_I, f_I^c) in place of (x_i, a_i, f_i). The two lifts evaluate $nK_y + mK_x$ pairs.

The solver returns once a check finds both marginal residuals of the current plan, $\|\mathbf{P}\mathbf{1} - \mathbf{a}\|_1$ and $\|\mathbf{P}^\top \mathbf{1} - \mathbf{b}\|_1$, below a tolerance τ. Computing them exactly costs $\mathcal{O}(nm)$, so we check by sampling (Figure 1d). We draw rows i with probability a_i and average $|r_i/a_i - 1|$, where the row sum $r_i = (\mathbf{P}\mathbf{1})_i$ costs $\mathcal{O}(md)$. The average is an unbiased estimate of the row residual, and columns are sampled in the same way. The check passes when both estimates plus three standard errors are below τ. Every residual we report is evaluated at the returned potentials, independently of the check. A lifted plan can miss τ for two reasons, which the following bound separates.

Proposition 1. *Let $\mathbf{P}^c$ be the entropic plan induced by any finite coarse potentials, $\lambda_c = \max_J |\log(\sum_I P^c_{IJ}/b_J)|$ its largest log column-mass error, $R_Y = \max_J \max_{j \in J} \|y_j - y_J^c\|$ the largest target-cell radius, and $\theta = 2\,\mathrm{diam}(\mathbf{X}) R_Y/\varepsilon$. The plan $\mathbf{P}$ induced by the two lifts satisfies*

$$\|\mathbf{P}\mathbf{1} - \mathbf{a}\|_1 \le (e^{\lambda_c} - 1) + e^{\lambda_c}(e^\theta - 1 - \theta). \tag{3}$$

With the clouds exchanged, the same bound holds for the column residual $\|\mathbf{P}^\top \mathbf{1} - \mathbf{b}\|_1$.

The proof is in Appendix C.1. The first term comes from inexact coarse marginals and vanishes at coarse convergence. The second bounds the error of replacing points by centroids. It grows like $\theta^2/2$ for small θ, and at coarse convergence it reduces to $e^\theta - 1 - \theta$, which no coarse update changes. When the check rejects a lift, the solver therefore shrinks the first term with further coarse updates at the target ε, which cost only $K_x K_y$ centroid pairs each, until its potentials stall or 2048 extra updates have run, and then lifts and checks again. If this check also fails, the solver turns to fine updates on the points, repeats the check on a schedule, and reports that the tolerance was not confirmed if the fine-update budget runs out first. On all our problems the bound of Proposition 1 exceeds τ, so the sampled check, not the bound, decides whether to accept a lift.

Outside the forward solve, the coarse records also yield a surrogate gradient for optimization. Holding the coarse target records (y_J^c, b_J, g_J^c) fixed makes each lifted f_i a function of x_i alone, and the surrogate $\widetilde{\mathcal{L}} = \sum_i a_i f_i$ has gradient $\widetilde{G}_i = 2a_i(x_i - \widetilde{T}_i)$, where $\widetilde{T}_i$ is the barycentric projection of x_i against these records. The gradient costs the lift's nK_y pairs and needs no fine update. Appendix C.2 gives a condition on direction and step size sufficient to decrease the original objective.

3.2 Sparse Sinkhorn Updates

A dense fine update evaluates all nm point pairs, so FS2 restricts each fine update to the block pairs in a mask. We build the mask over the $N_B = \lceil n/B \rceil$ source and $M_B = \lceil m/B \rceil$ target blocks of at most B points by potential-dependent truncation of the current entropic plan $P_{ij} = a_i b_j e^{(f_i + g_j - C_{ij})/\varepsilon}$. From here on, I and J index blocks rather than cells. Both half-steps share the mask, so its threshold must account for both block masses. The following bound makes this precise.

Theorem 1. *Fix the potentials and $\delta_{\mathrm{rel}} > 0$. Write $a_I = \sum_{i \in I} a_i$, $b_J = \sum_{j \in J} b_j$, and define*

$$m_{IJ} = \max_{i \in I, j \in J} \frac{f_i + g_j - C_{ij}}{\varepsilon}, \qquad \rho_{IJ} = \log \frac{\delta_{\mathrm{rel}}}{\max(a_I, b_J)}. \tag{4}$$

If $m_{IJ} \le \rho_{IJ}$, then $\sum_{j \in J} P_{ij} \le \delta_{\mathrm{rel}} a_i$ for every $i \in I$, and $\sum_{i \in I} P_{ij} \le \delta_{\mathrm{rel}} b_j$ for every $j \in J$.

The proof is in Appendix C.3: bound each plan entry by $e^{m_{IJ}}$, then sum the weights in the opposite block. Testing every m_{IJ} directly would evaluate all nm point pairs, so we screen in two stages (Figure 1e–f). First, a *box screen* rejects block pairs whose axis-aligned bounding boxes are far apart, using only block-level potential maxima $f_I^\vee = \max_{i \in I} f_i$, $g_J^\vee = \max_{j \in J} g_j$ and the minimum squared distance $d^2_{\min}(I, J)$:

$$\mathcal{A} := \{(I, J) : m_{IJ} > \rho_{IJ}\} \subseteq \{(I, J) : d^2_{\min}(I, J) < f_I^\vee + g_J^\vee - \varepsilon \rho_{IJ}\}. \tag{5}$$

This reduces the scan from nmd to $N_B M_B d$. Second, a *verification* pass evaluates the exact m_{IJ} over all B^2 point pairs of each *candidate* (a block pair that passes the box screen) and rejects those with $m_{IJ} \le \rho_{IJ}$. We store the *admitted* tiles $\mathcal{A}$ in CSR form, which lists, for each source block I, the target blocks J with $(I, J) \in \mathcal{A}$. The f half-step reads this index and the g half-step reads its transpose. If a block has no admitted tile, its sparse half-step would have no terms, so the solver computes it densely, against every point of the other cloud. The other blocks stay sparse.

Both the truncation principle, which rejects negligible pairs with the current potentials, and the block relaxation behind Equation (5) come from Schmitzer (2019), but FS2 differs from his solver in three ways. First, Schmitzer stores the truncated kernel, one entry per retained point pair, and iterates by sparse matrix–vector products. FS2 stores one index per admitted tile instead of its B^2 entries, because each sparse half-step runs FlashSinkhorn's streaming kernel (Section 2.1) over these tiles and recomputes their scores on chip. Second, Schmitzer refines a cell hierarchy recursively down to point pairs, whereas we screen all block pairs in parallel at a single level and keep whole $B \times B$ tiles, since the kernel evaluates every pair of an admitted tile. Each kept tile spans two contiguous ranges of the Morton order, so the kernel loads both ranges with coalesced reads and forms their scores on tensor cores. Third, Schmitzer's threshold is uniform, whereas ours is set by the block masses. Because Theorem 1 certifies a mask only at the potentials used to build it, we rebuild the mask at every fine update. The mask also determines the error in each sparse update and in the gradient.

Proposition 2. *Let ν_i be the fraction of row i's plan mass omitted by the mask, with positive retained mass. Then* (i) *the sparse and full row updates differ by $|f_i^{+,\mathcal{A}} - f_i^+| = \varepsilon \log \frac{1}{1-\nu_i}$, and* (ii) *the barycentric projection over the admitted columns satisfies $\|T_\varepsilon^{\mathcal{A}}(x_i) - T_\varepsilon(x_i)\| \le \nu_i \operatorname{diam}(\mathbf{Y})$.*

The proof is in Appendix C.4. Both errors vanish with ν_i, which Theorem 1 bounds. Each omitted tile contributes at most $\delta_{\mathrm{rel}} a_i$ to row i, and a source block omits at most M_B tiles, so $\nu_i \le M_B \delta_{\mathrm{rel}}\, a_i / r_i$, where a_i / r_i tends to 1 as the marginals converge. This bound guides the choice of δ_{rel} for a tolerance τ. The automatic policy takes $\delta_{\mathrm{rel}} = \tau / (4 \max(N_B, M_B))$, so $\nu_i \le (\tau/4)\, a_i / r_i$ in every row and the mass the mask omits, summed over all rows or over all columns, is at most $\tau/4$. This also certifies the full residual. Let $\mathbf{P}^{\mathcal{A}}$ keep only the admitted tiles of $\mathbf{P}$, and let $\Omega = \sum_{(I,J) \notin \mathcal{A}} a_I b_J e^{\bar{m}_{IJ}}$, where $\bar{m}_{IJ}$ is m_{IJ} for a rejected candidate and $(f_I^\vee + g_J^\vee - d^2_{\min}(I, J))/\varepsilon$ for a pair the box screen rejects. Then Ω, at most $\tau/4$, bounds the

Table 1: FS2 against GeomLoss (GL) and FlashSinkhorn (FS) on the clustered $n{=}m{=}2^{21}$, $\varepsilon{=}0.001$ problem. Bold marks solves that meet $\tau{=}0.005$. In (a), stage times are GPU kernel time from a separate profiled call, with sampled checks counted in the fine stage. GL$^+$ is GL at scaling 0.99, continued at the target ε until its full residual first meets τ.

(a) Complete solve

	FS2	GL	GL$^+$	FS
Time (s)	**15.44**	21.45	**421.10**	311.98
coarse stage	2.94	0.005	0.022	–
fine stage	12.30	21.40	420.67	311.98
Fine updates	18	12	225	44
per update (s)	0.68	1.78	1.87	7.09
Memory (GiB)	2.09	0.26	0.26	0.20
Residual	**0.00483**	0.05416	**0.004997**	0.05972

(b) One fine update per direction

	FS2	GL	FS
Point pairs (10^{11})	2.58	2.58	44.0
Kernel time (ms)	251	394	3,600
ms per 10^9 pairs	0.97	1.53	0.82
Warp instr. per pair	0.28	0.63	0.34
HBM read (B/10^3 pairs)	14.9	0.32	4.35
Tensor pipe (%)	21.1	0.07	–
Math-pipe stall (%)	2.3	17.4	9.0

omitted mass. At the potentials used to build the mask, $\|\mathbf{P}\mathbf{1}-\mathbf{a}\|_1 \le \|\mathbf{P}^{\mathcal{A}}\mathbf{1}-\mathbf{a}\|_1 + \Omega$, and likewise for the columns. The sparse half-steps give the marginals of $\mathbf{P}^{\mathcal{A}}$ at no extra cost, since $(\mathbf{P}^{\mathcal{A}}\mathbf{1})_i/a_i = e^{(f_i - f_i^{+,\mathcal{A}})/\varepsilon}$, and Ω costs one reduction over values the two passes already form.

When an application needs an accurate gradient rather than the surrogate, FS2 evaluates the gradient $2a_i(x_i - T_\varepsilon(x_i))$ on the points. It builds a mask at the returned potentials by the same rule and replaces $T_\varepsilon(x_i)$ by $T_\varepsilon^{\mathcal{A}}(x_i)$, the barycentric projection over the admitted tiles of the source block containing i. A kernel computes it by streaming $\sum_j P_{ij}y_j$ alongside the row sum, and the transposed index gives the target-side gradient in the same way. Proposition 2(ii) bounds the error that the mask adds, and the potentials' distance from convergence and the kernel's arithmetic add the rest.

Each fine update has three costs: the box screen ($\Theta(N_B M_B d)$), the verification ($\Theta(B^2 d)$ per candidate), and both sparse half-steps ($\Theta(B^2 d\,|\mathcal{A}|)$). As the blur $\sqrt{\varepsilon}$ falls to a few nearest-neighbour spacings, $|\mathcal{A}|$ drops well below $N_B M_B$ and sparse updates become cheaper than dense ones. Beyond the points and potentials, working memory holds only the indices of the candidate and admitted tiles. Appendices C.5 and D.4 give the IO model and the complete operation count.

4 Experiments

We evaluate FS2 on 32 synthetic problems and on the Quijote cosmological N-body simulation, measuring the time each solver takes to return potentials whose full two-sided marginal residual, $R = \max(\|\mathbf{P}\mathbf{1}-\mathbf{a}\|_1, \|\mathbf{P}^\top\mathbf{1}-\mathbf{b}\|_1)$, is at most a tolerance τ. On the synthetic problems (Section 4.1), we compare FS2 with GeomLoss multiscale and FlashSinkhorn and isolate its three ingredients: the coarse stage, the sparse fine stage and the surrogate gradient for optimization. On Quijote (Section 4.2), we scale FS2 from 2^{24} particles to the full 2^{27}-particle box.

Every solve runs on one A100, and FS2 receives centered, TF32-rounded coordinates. Any potentials induce the plan $P_{ij} = a_i b_j e^{(f_i+g_j-C_{ij})/\varepsilon}$, which is the exact EOT solution for its own marginals $\mathbf{P}\mathbf{1}$ and $\mathbf{P}^\top\mathbf{1}$ (Peyré & Cuturi, 2019). Its error therefore lies entirely in these marginals, and R measures how far they are from $\mathbf{a}$ and $\mathbf{b}$.

4.1 Benchmarks

Setup. We benchmark 32 problems in $d{=}3$ with uniform weights, 16 on clustered clouds of 32 Gaussian clusters and 16 on heterogeneous clouds that mix a compact Gaussian, a thin needle and a 1% uniform background. Both families span $n = m$ from 2^{19} to 2^{23} and ε from 0.001 to 0.02. This range of ε puts the blur $\sqrt{\varepsilon}$ between 0.032 and 0.14, or 3–14% of the side of the unit cube of the clustered clouds. All methods receive the same coordinates and the full squared-Euclidean cost, GeomLoss always runs its multiscale solver, and the target is $\tau = 0.005$. Appendix E gives the data, the protocol, the endpoint of every run and a sweep of GeomLoss's scaling and cluster scale.

Speed. FS2 meets τ on all 32 problems and GeomLoss on 10, all heterogeneous (Figure 2a–d). At the returned potentials, the certificate of Section 3.2 confirms $R \le \tau$ for every return we evaluated on the 32 problems. Across 20 further sampling seeds on each of the seven problems closest to

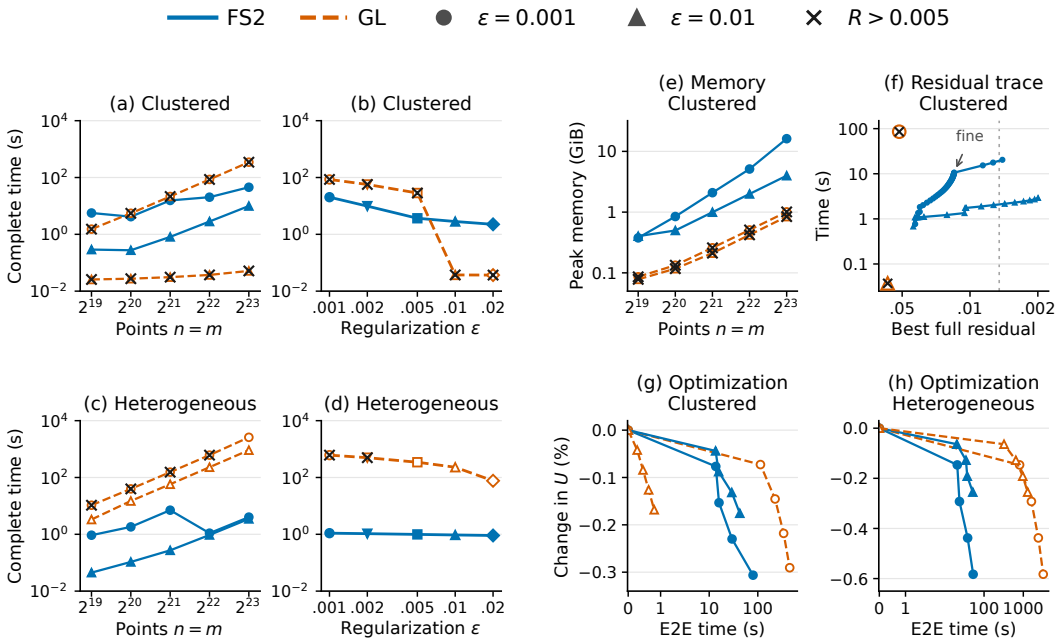


Figure 2: **FS2 vs GeomLoss (GL).** (a–d) Time of a complete solve on 26 problems: n from 2^{19} to 2^{23} (a,c) and ε from 0.001 to 0.02 at $n{=}2^{22}$ (b,d). A cross marks a GL return with residual $R > \tau{=}0.005$. (e) Peak GPU memory allocated by PyTorch. (f) On clustered $n{=}2^{22}$, the time at which each new best full residual R is reached, for FS2's potentials along the solve and for GL's native returns. The residual decreases to the right. The dotted line marks τ and the arrow marks the first fine update. (g,h) Change in the grouped objective U over twenty optimization steps at $n{=}2^{21}$, against cumulative end-to-end (E2E) time, including compilation and autotuning.

τ, the check never accepted potentials whose residual exceeds τ (Appendix E.4). On clustered problems GeomLoss misses τ at every size and regularization. Table 1a shows one such problem, $n{=}2^{21}$, in full: FS2 meets τ in 15.4 s, GeomLoss ends its schedule after 21.45 s at residual 0.054, and FlashSinkhorn needs 312 s to reach 0.060, because each of its 44 updates evaluates all n^2 pairs. With its native schedule, GeomLoss misses τ on this problem at every scaling and cluster scale we tried, and the best setting needs 421 s when continued at the target ε until it meets τ (GL$^+$). Figure 2(f) shows how FS2 reaches τ on clustered $n{=}2^{22}$. At $\varepsilon{=}0.001$, FS2 lifts after less than a second, because a coarse update evaluates $K_x K_y$ centroid pairs rather than n^2 point pairs, and the lifted residual is already below GeomLoss's final one. Further coarse updates then lower the residual, and the fine updates remove the remaining centroid error, bringing the residual below τ in about a quarter of GeomLoss's time. At $\varepsilon{=}0.01$ the blur is $\sqrt{10}$ times wider relative to the cells, so the centroid error is smaller and the coarse stage alone reaches τ. GeomLoss returns sooner but misses it.

On the ten problems where both solvers meet τ, FS2 is 74–645$\times$ faster than GeomLoss at its default cluster scale. The largest gap is on heterogeneous $n{=}2^{23}$ at $\varepsilon{=}0.001$, where FS2 returns in 4.026 s and GeomLoss takes 2595 s. On all ten, FS2's first lift already passes the check, so its only work on the points is that lift and the check. GeomLoss's default gives far fewer cells, wider than the final blur, so its schedule ends with levels on the points, and it never checks its lift. FS2 picks its cell count from the input. We tested that count on one of these ten problems, where GeomLoss also stops after its coarse stage and comes close to FS2 but remains slower.

Ablations. Each row of Table 2 removes or shortens one component of FS2. FS2 itself runs 2196 coarse updates and then needs just 15 fine updates. Without the coarse stage, the solve anneals on the points and misses τ even at the fine-update cap. Stopping the coarse stage after 404 updates instead leaves 1115 fine updates and makes the solve 51.6$\times$ slower. With the same 15 fine updates, dense execution evaluates every tile rather than the 6.3% the mask admits and is 8.00$\times$ slower. The box screen alone, without tile verification, recovers almost all of this gain and is only 4% slower.

Table 2: Ablations on clustered $n{=}m{=}2^{20}$, $\varepsilon{=}0.001$.

Control	Coarse	Fine	Time (s)	Time/FS2
FS2	2196	15	4.17	1
No coarse stage	0	130	153.33	$R{>}\tau$
Early coarse stop	404	1115	215.67	51.59
Dense fine	2196	15	33.33	8.00
Box only	2196	15	4.34	1.04

Coarse, Fine: updates. Early coarse stop: fine cap 8192.

Table 3: FS2 peak memory at $\varepsilon{=}0.001$.

Cloud	n	B	Admitted	Peak (GiB)
Clustered	2^{20}	128	6.3%	0.85
Heterog.	2^{20}	128	60.6%	2.72
Heterog.	2^{21}	128	59.8%	9.89
Clustered	2^{23}	128	5.2%	16.17
Clustered	2^{23}	256	5.6%	6.88
Heterog.	2^{23}	128	no mask	3.40

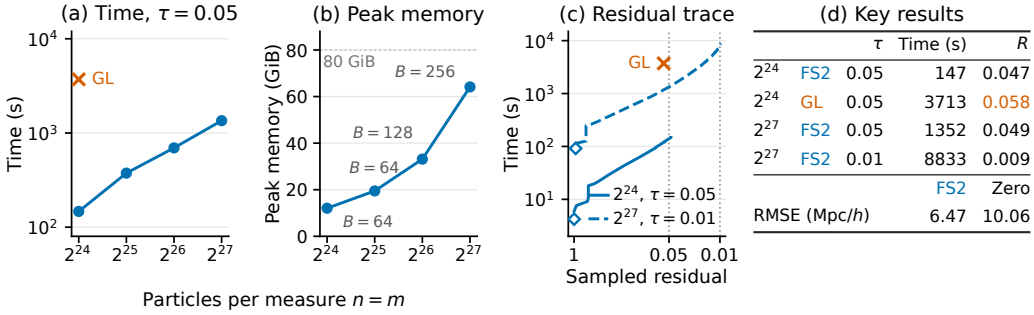
Profiling insight. To see why FS2's fine update is fast, Table 1b profiles one, in both directions, with NCU on an A100, next to a dense FlashSinkhorn update and GeomLoss's KeOps executor on FS2's mask and potentials. FS2's update evaluates only 5.9% of the point pairs (x_i, y_j) of the dense update, and NCU measures a $14\times$ shorter kernel time. Per pair, the sparse kernel is nearly as cheap as the dense one, 0.97 against 0.82 ms per 10^9 pairs, and executes fewer warp instructions. This holds although it gathers each admitted target block through the CSR index, while the dense kernel streams every block in order (Proposition 5). On the same mask, GeomLoss's KeOps executor takes $1.57\times$ longer and executes $2.3\times$ more instructions per pair, because FS2 computes the squared distances as tiled dot products on the tensor cores. FS2 even reads more HBM data per pair than GeomLoss, so on the same mask its speed comes from fewer instructions, not fewer bytes.

Memory. FS2 trades memory for speed by design. The selector keeps the coarse state and the block summaries within the L2 budget and, within it, picks the finest cells and blocks, so the coarse stage stays cheap and the mask screens at fine granularity. The price is the mask's index, 24 bytes per screened or admitted tile, which puts FS2's peak allocation at 8–10$\times$ that of either baseline in Table 1a. The peak follows the mask (Table 3). At $n{=}2^{20}$ the heterogeneous mask admits ten times the clustered share and needs three times the memory. At $n{=}2^{23}$ the heterogeneous solve returns its first lift and builds no mask, so its peak lies below the one at 2^{21}. Without a mask, as at $\varepsilon{=}0.01$ for $n{\geq}2^{20}$, the peak grows only as GeomLoss's does. At fixed ε the mask grows faster than n, because each point has more neighbours within the blur, so over the clustered sweep at $\varepsilon{=}0.001$ the peak rises 40-fold while n rises 16-fold (Figure 2e). Larger blocks cut this cost. At $n{=}2^{23}$, $B = 256$ instead of the selected 128 lowers the peak from 16.2 to 6.9 GiB for slightly more time, and the selector itself takes $B = 256$ for the 2^{27} Quijote run, whose peak stays within the 80 GiB card.

Optimization. Many applications move points along an OT gradient and need a descent direction rather than an exact gradient. We test FS2's direction and its end-to-end cost by moving the source points to decrease their EOT cost OT_ε (Equation (1)) to a fixed target cloud, with no other term in the objective. FS2 steps along the surrogate gradient $\widetilde{G}$, which takes the barycentric projection over the K_y coarse target centroids instead of the m targets. A step therefore costs one coarse solve and one pass over the centroids, and needs no fine update. A small step along any direction whose relative error to the exact gradient is below one decreases the objective (Proposition 4). On each of the 32 inputs, both methods take twenty normalized steps $\mathbf{X} \leftarrow \mathbf{X} - 0.03\, H/\|H\|_F$, with $H = \widetilde{G}$ for FS2 and, for GeomLoss, the autograd gradient with respect to the source points, which costs about one forward solve per step. Each step moves the points by $0.03/\sqrt{n}$ RMS, so the twenty steps measure the direction at the initial points rather than following a long flow. Both methods' iterates are scored by the same grouped objective U, which fixes K groups of the initial source points and of the targets and adds the EOT cost between the mass-weighted group centroids, solved accurately, to the mean squared distance of the points to their group centroids. This score ranks the iterates without a full-point solve, and the surrogate lowers it at every checkpoint on all 32 inputs. At $n{=}2^{21}$ (Figure 2g,h), FS2 is 25–61$\times$ faster end to end on heterogeneous data, with tied final scores. On clustered data it reaches a lower final U at both ε, and is also faster at $\varepsilon{=}0.001$. GeomLoss is faster only at clustered $\varepsilon{=}0.01$, where its forward solve takes 0.03 s but misses τ.

4.2 Quijote Particle Transport

Problem and data. We test FS2 on one fiducial realization of the Quijote cosmological N-body suite, 512^3 dark-matter particles in a periodic $1\,h^{-1}\mathrm{Gpc}$ box (Villaescusa-Navarro et al., 2020). The sources are the redshift-zero positions x_i and the targets are the Lagrangian lattice sites q_i, both with uniform weights. The particle IDs define the reference displacements but are hidden from the



(d) Key results

		τ	Time (s)	R
2^{24}	FS2	0.05	147	0.047
2^{24}	GL	0.05	3713	0.058
2^{27}	FS2	0.05	1352	0.049
2^{27}	FS2	0.01	8833	0.009

	FS2	Zero
RMSE (Mpc/h)	6.47	10.06

Figure 3: **FS2 on Quijote.** (a) Time of a complete warm call of FS2 at τ=0.05. The cross marks GL's instrumented call at 2^{24}. (b) Peak GPU memory allocated by PyTorch, and the selected block size B. (c) Best sampled residual versus time. Diamonds mark the first fine update and dotted lines the targets. (d) Key results. Orange marks a return with residual above τ, and RMSE is the displacement error of the 2^{27} solution at τ=0.01 and of a zero-displacement predictor.

solve. Nested subdomains centered in the box give $n = m$ from 2^{24} to 2^{27}, the full box. We fix the blur $\sqrt{\varepsilon} \approx 1.95\,\mathrm{Mpc}/h$ at one lattice spacing, the mean interparticle distance and 0.2–0.4% of the domain width, and use the plain squared-Euclidean cost, which ignores the periodic wrap of the box. The evolved cloud is strongly clustered and far from the lattice. In the 2^{24} subdomain the median particle lies 4.6 blurs from its lattice site. At this blur the coarse stage alone does not reach the target, so every solve's fine updates start from the first lift. We set the cell level t and the fine-update cap for each size. Appendix F gives the data, the protocol and every run.

Scaling. At $\tau = 0.05$, FS2 meets the target at all four sizes of Figure 3, verified on sampled points at the two intermediate sizes. From 2^{24} to 2^{27} particles, the time grows 9.2-fold for an 8-fold increase in n, and the peak memory stays within the 80 GiB card. As on the clustered benchmarks, the fine stage does most of the work. It begins within the first 3% of each run's time and lowers the sampled residual from about 1 to the target. GeomLoss multiscale takes 25 times as long as FS2 at 2^{24} and still misses τ. Its time grows almost quadratically with n on the synthetic benchmarks, so we run it only at 2^{24}. At $\tau = 0.01$, a first complete call of FS2 solves the full box, 2^{54} particle pairs, on one A100 in under 2.5 hours. Its full marginal residual is below 0.01, even after an FP64 correction estimated on sampled points. The certificate of Section 3.2 gives the same bound from one mask build at the returned potentials, in minutes rather than the hours of the full evaluation. To our knowledge, the full box is the largest discrete EOT problem solved to this accuracy within hours, and it matches the particle count of the semi-discrete reconstruction of Lévy et al. (2025).

Displacement reconstruction. The plan also estimates each particle's displacement. On 4096 particles sampled from the 2^{27}, $\tau = 0.01$ solution, the barycentric projection $T_\varepsilon(x_i)$ of Section 2.1, taken over all 2^{27} targets, gives displacements $x_i - T_\varepsilon(x_i)$ whose RMSE against the particle-ID displacements $x_i - q_i$, both wrapped to the periodic box, is 35.7% below that of predicting no displacement (Figure 3d). The three components correlate equally well with the reference, and particles whose reference trajectories cross a periodic face have larger errors than the rest.

5 Conclusion and Limitations

We introduced FlashSinkhorn 2 (FS2), which solves large discrete EOT problems to a prescribed marginal residual by coupling a coarse stage on cell centroids with a block-sparse fine stage that removes the centroid error. On synthetic benchmarks, FS2 reaches the target residual on all 32 problems and GeomLoss on 10. On one A100 it solves EOT between two 1.34×10^8-particle measures, at a blur of one lattice spacing, to a marginal residual below 0.01 in under 2.5 hours.

FS2 targets squared-Euclidean cost in $d \leq 3$, where a blur-sized ball holds few points and screening pays off. Reaching the 10^{10} points that Lévy et al. (2025) set as the goal for cosmology needs blocks spread over several GPUs, because the coordinates of one measure alone take 112 GiB in FP32, more than an A100's 80 GiB. It also needs a sub-quadratic pair search, such as a dual-tree traversal (Lee et al., 2011), because the box screen tests every block pair.

AI USE STATEMENT

In this work, we used generative AI tools (Anthropic's Claude through Claude Code, and OpenAI's Codex) to generate synthetic data sets, discuss theoretical models and conceptual frameworks, propose and refine hypotheses, design and provide feedback on the methodology and experiments, implement methods, clean and reformat the Quijote data, and interpret results. We have not used generative AI tools to formulate mathematical claims, to write their proofs or provide critical ingredients for proving them, or for translation, and qualitative and thematic data analysis is not applicable to this work. Additionally, we used generative AI tools to create and modify figures, suggest experimental parameters, create and edit software code, draft and edit parts of the paper, suggest its structure, search for and summarize related literature, and format references. We have reviewed all AI-assisted work: the authors revised all text and checked every reported number against the experiment records. Generated code was tested, including parity tests against GeomLoss, and every reported marginal residual comes from an independent evaluation of the returned potentials, over all marginals except for two intermediate Quijote sizes, checked on sampled points. Cited works were checked against their sources. We take responsibility for all content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This paper presents work whose goal is to advance the field of machine learning. It involves no human subjects or personal data, and the cosmological simulation data it uses are public (Villaescusa-Navarro et al., 2020). There are many potential societal consequences of our work, none of which we feel must be highlighted here.

REPRODUCIBILITY STATEMENT

We will release the code upon acceptance. The paper specifies everything needed to reproduce the results without it. Algorithms 1 to 3 give the solver, the mask construction and the sparse kernel. Theorem 1 and Propositions 1–5 are proved in Appendix C. Appendix D gives the block-size and cell selection, the stopping rules and the operation count, and Appendices E and F give the data generators, hardware and software versions, baseline configurations, timing protocol and the independent residual evaluation behind every reported number. FS2 builds on the public FlashSinkhorn kernels (Ye et al., 2026), and both baselines are public releases.

REFERENCES

Mokhtar Z. Alaya, Maxime Bérar, Gilles Gasso, and Alain Rakotomamonjy. Screening Sinkhorn algorithm for regularized optimal transport. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019. URL https://arxiv.org/abs/1906.08540.

Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Niles-Weed. Massively scalable Sinkhorn distances via the Nyström method. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019. URL https://arxiv.org/abs/1812.05189.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.

R. R. Bahadur and L. J. Savage. The nonexistence of certain statistical procedures in nonparametric problems. *The Annals of Mathematical Statistics*, 27(4):1115–1122, 1956.

Mauro Bonafini and Bernhard Schmitzer. Domain decomposition for entropy regularized optimal transport. *Numerische Mathematik*, 149(4):819–870, 2021. doi: 10.1007/s00211-021-01245-0.

Charlotte Bunne, Geoffrey Schiebinger, Andreas Krause, Aviv Regev, and Marco Cuturi. Optimal transport for single-cell and spatial omics. *Nature Reviews Methods Primers*, 4(1):58, 2024.

Olivier Catoni. Challenging the empirical mean and empirical variance: A deviation study. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 48(4):1148–1185, 2012.

Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the GPU, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(74):1–6, 2021.

Samir Chowdhury, David Miller, and Tom Needham. Quantized Gromov-Wasserstein. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, volume 12977 of *Lecture Notes in Computer Science*, pp. 811–827, 2021. doi: 10.1007/978-3-030-86523-8_49. URL https://arxiv.org/abs/2104.02013.

Marco Cuturi. Sinkhorn Distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 7, 2025. arXiv:2412.05496.

Jean Feydy. *Geometric data analysis, beyond convolutions*. PhD thesis, Université Paris-Saclay, 2020.

Jean Feydy, Pierre Roussillon, Alain Trouvé, and Pietro Gori. Fast and scalable optimal transport for brain tractograms. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pp. 636–644, 2019a. doi: 10.1007/978-3-030-32248-9_71.

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and MMD using Sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pp. 2681–2690, 2019b.

Uriel Frisch, Sabino Matarrese, Roya Mohayaee, and Andrei Sobolevski. A reconstruction of the initial conditions of the Universe by optimal mass transportation. *Nature*, 417:260–262, 2002. doi: 10.1038/417260a.

Johannes Gasteiger, Marten Lienen, and Stephan Günnemann. Scalable optimal transport in high dimensions for graph distances, embedding alignment, and more. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. URL https://arxiv.org/abs/2107.06876.

Samuel Gerber and Mauro Maggioni. Multiscale strategies for computing optimal transport. *Journal of Machine Learning Research*, 18(72):1–32, 2017.

Peter Halmos, Julian Gold, Xinhao Liu, and Benjamin J. Raphael. Hierarchical refinement: Optimal transport to infinity and beyond. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21629–21661, 2025.

Mete Kemertas, Amir-massoud Farahmand, and Allan D. Jepson. A truncated Newton method for optimal transport. In *International Conference on Learning Representations (ICLR)*, 2025a. URL https://arxiv.org/abs/2504.02067.

Mete Kemertas, Allan D. Jepson, and Amir-massoud Farahmand. Efficient and accurate optimal transport with mirror descent and conjugate gradients. *Transactions on Machine Learning Research*, 2025b. URL https://openreview.net/forum?id=FVFqrxeF8e.

Dominik Klein, Giovanni Palla, Marius Lange, Michal Klein, Zoe Piran, Manuel Gander, Laetitia Meng-Papaxanthos, Michael Sterr, Lama Saber, Changying Jing, Aimée Bastidas-Ponce, Perla Cota, Marta Tarquis-Medina, Shrey Parikh, Ilan Gold, Heiko Lickert, Mostafa Bakhti, Mor Nitzan, Marco Cuturi, and Fabian J. Theis. Mapping cells through time and space with moscot. *Nature*, 638(8052):1065–1075, 2025. doi: 10.1038/s41586-024-08453-2.

Rajmadan Lakshmanan, Alois Pichler, and Daniel Potts. Nonequispaced fast Fourier transform boost for the Sinkhorn algorithm. *Electronic Transactions on Numerical Analysis*, 58:289–315, 2023. doi: 10.1553/etna_vol58s289.

Dongryeol Lee, Alexander G. Gray, and Andrew W. Moore. Dual-tree fast Gauss transforms. *arXiv preprint arXiv:1102.2878*, 2011. URL https://arxiv.org/abs/1102.2878.

Tobias Lehmann, Max-K. von Renesse, Alexander Sambale, and André Uschmajew. A note on overrelaxation in the Sinkhorn algorithm. *Optimization Letters*, 16(8):2209–2220, 2022.

Bruno Lévy, Nicolas Ray, Quentin Mérigot, and Hugo Leclerc. Large-scale semi-discrete optimal transport with distributed Voronoi diagrams. *Journal of Computational Physics*, 542:114374, 2025. doi: 10.1016/j.jcp.2025.114374. arXiv:2406.04192.

Xingjie Li, Fei Lu, Molei Tao, and Felix X-F Ye. Robust first- and second-order differentiation for regularized optimal transport. *SIAM Journal on Scientific Computing*, 47(3):C630–C654, 2025. doi: 10.1137/24M1674030.

Ismael Medina and Bernhard Schmitzer. Flow updates for domain decomposition of entropic optimal transport. *ESAIM: Mathematical Modelling and Numerical Analysis*, 59(3):1239–1270, 2025. doi: 10.1051/m2an/2025022.

Maxim Milakov and Natalia Gimelshein. Online Normalizer Calculation for Softmax. *arXiv preprint arXiv:1805.02867*, 2018.

Gabriel Peyré and Marco Cuturi. Computational Optimal Transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

Yixuan Qiu. cuRegOT: A GPU-Accelerated Solver for Entropic-Regularized Optimal Transport. *arXiv preprint arXiv:2605.08793*, 2026. URL https://arxiv.org/abs/2605.08793.

Meyer Scetbon, Marco Cuturi, and Gabriel Peyré. Low-rank Sinkhorn factorization. In *International Conference on Machine Learning*, pp. 9344–9354, 2021.

Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, Brian Cleary, Vidya Subramanian, Aryeh Solomon, Joshua Gould, Siyan Liu, Stacie Lin, Peter Berube, Lia Lee, Jenny Chen, Justin Brumbaugh, Philippe Rigollet, Konrad Hochedlinger, Rudolf Jaenisch, Aviv Regev, and Eric S. Lander. Optimal-Transport analysis of single-cell Gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943.e22, 2019.

Marwin Schindler. Shape shifting: A multiscale optimal transport approach to 3D point cloud comparison. Master's thesis, TU Wien, 2025. URL https://repositum.tuwien.at/handle/20.500.12708/216205.

Bernhard Schmitzer. A sparse multiscale algorithm for dense optimal transport. *Journal of Mathematical Imaging and Vision*, 56(2):238–259, 2016. doi: 10.1007/s10851-016-0653-9. URL https://arxiv.org/abs/1510.05466.

Bernhard Schmitzer. Stabilized sparse scaling algorithms for entropy regularized transport problems. *SIAM Journal on Scientific Computing*, 41(3):A1443–A1481, 2019.

Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional Wasserstein Distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics*, 34(4):66:1–66:11, 2015.

Xun Tang, Michael Shavlovsky, Holakou Rahmanian, Elisa Tardini, Kiran Koshy Thekumparampil, Tesi Xiao, and Lexing Ying. Accelerating Sinkhorn Algorithm with Sparse Newton Iterations. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Kuj5gVp5GQ.

Alexis Thibault, Lénaïc Chizat, Charles Dossal, and Nicolas Papadakis. Overrelaxed Sinkhorn–Knopp algorithm for regularized optimal transport. *Algorithms*, 14(5):143, 2021. doi: 10.3390/a14050143.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Francisco Villaescusa-Navarro, ChangHoon Hahn, Elena Massara, et al. The Quijote simulations. *The Astrophysical Journal Supplement Series*, 250(1):2, 2020. doi: 10.3847/1538-4365/ab9d82.

Felix X.-F. Ye, Xingjie Li, An Yu, Ming-Ching Chang, Linsong Chu, and Davis Wertheimer. FlashSinkhorn: IO-aware entropic optimal transport on GPU. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025. doi: 10.18653/v1/2025.acl-long.1126.

A Notation and Conventions

Capital indices I, J name source and target groups: spatial cells before Section 3.2 and execution blocks from Section 3.2 on, with set masses $a_I = \sum_{i\in I} a_i$ and $b_J = \sum_{j\in J} b_j$ in both cases. The mark $\vee$ takes the maximum over the block, so $f_I^\vee = \max_{i\in I} f_i$ and $g_J^\vee = \max_{j\in J} g_j$. A superscript c names the coarse measure, so x_I^c, y_J^c are centroids and f_I^c, g_J^c the potentials solved on them. On a boldface array the subscript slices, so $\mathbf{Y}_J$ and $\hat{\mathbf{g}}_J$ are that block's rows.

Table 4: Notation used across sections. Symbols local to one proof or appendix section are defined where they appear.

Symbol	Meaning
Problem and Sinkhorn	
$\mathbf{X}, \mathbf{Y}$	$\mathbf{X} = (x_i)_{i=1}^n$, $\mathbf{Y} = (y_j)_{j=1}^m$, $x_i, y_j \in \mathbb{R}^d$
$\mathbf{a}, \mathbf{b}$	$\mathbf{a} \in \Delta^n$, $\mathbf{b} \in \Delta^m$, $a_i, b_j > 0$
C_{ij}	$\|x_i - y_j\|^2$
ε	entropic regularization, $\varepsilon > 0$, in cost units
$\mathbf{f}, \mathbf{g}$	dual potentials, $\mathbf{f} \in \mathbb{R}^n$, $\mathbf{g} \in \mathbb{R}^m$
$\hat{\mathbf{f}}, \hat{\mathbf{g}}$	$\hat{f}_i = f_i - \|x_i\|^2$, $\hat{g}_j = g_j - \|y_j\|^2$
$\mathbf{P}$	$P_{ij} = a_i b_j \exp\big((f_i + g_j - C_{ij})/\varepsilon\big)$
$T_\varepsilon(x_i)$	$\sum_j P_{ij} y_j \big/ \sum_j P_{ij}$
R, τ	$R = \max(\|\mathbf{P}\mathbf{1} - \mathbf{a}\|_1, \|\mathbf{P}^\top \mathbf{1} - \mathbf{b}\|_1)$ at the returned potentials, with target $R \le \tau$
Coarse stage	
t	cell level, the number of low Morton-key bits dropped to form cells
K_x, K_y	numbers of source and target cells
r_i	$(\mathbf{P}\mathbf{1})_i$
f_i^+	$-\varepsilon \mathrm{LSE}_j\big[(g_j - C_{ij})/\varepsilon + \log b_j\big]$
$\widetilde{\mathcal{L}}, \widetilde{G}_i$	$\widetilde{\mathcal{L}} = \sum_i a_i f_i$ with (y_J^c, b_J, g_J^c) fixed, and $\widetilde{G}_i = 2a_i(x_i - \widetilde{T}_i)$
$\widetilde{T}_i$	barycentric projection of x_i against (y_J^c, b_J, g_J^c)
U	grouped score of the optimization experiment
Blocks and the mask	
B, N_B, M_B	$B \in \{64, 128, 256\}$, $N_B = \lceil n/B \rceil$, $M_B = \lceil m/B \rceil$
$d^2_{\min}(I, J)$	$\min\{\|x - y\|^2 : x \in \mathrm{box}(I),\ y \in \mathrm{box}(J)\}$
m_{IJ}	$\max_{i\in I, j\in J}(f_i + g_j - C_{ij})/\varepsilon$
$\rho_{IJ}, \delta_{\mathrm{rel}}$	$\rho_{IJ} = \log\big(\delta_{\mathrm{rel}}/\max(a_I, b_J)\big)$, with $\delta_{\mathrm{rel}} = \tau/(4\max(N_B, M_B))$ by default
$\mathcal{A}$	$\{(I, J) : m_{IJ} > \rho_{IJ}\}$, written $\mathcal{A}(\mathbf{f}, \mathbf{g}, \delta_{\mathrm{rel}})$ when the arguments vary
ν_i	$r_i^{-1} \sum_{J:(I,J)\notin\mathcal{A}} \sum_{j\in J} P_{ij}$ for $i \in I$
$f_i^{+,\mathcal{A}}, T_\varepsilon^{\mathcal{A}}(x_i)$	f_i^+ and $T_\varepsilon(x_i)$ with j restricted to $\bigcup_{J:(I,J)\in\mathcal{A}} J$

Potential conventions. The relation $f_i = \hat{f}_i + \|x_i\|^2$ in Section 2.1 maps the stored potentials to those used in Theorem 1. The block maxima are taken on $\mathbf{f}$, so the builder adds the squared norms back before reducing. Both stages carry the block mass explicitly and neither folds it into a potential, so both admit arbitrary positive weights and both use the same ρ_{IJ}.

B Additional Related Work

Other approaches replace the Sinkhorn half-steps with gradient-based or Newton-type optimization to reduce iteration counts. Kemertas et al. (2025b) combine mirror descent with conjugate gradients, and Kemertas et al. (2025a) achieve locally quadratic convergence with a truncated Newton method. Like Sinkhorn-Newton-Sparse (Tang et al., 2024) and cuRegOT (Qiu, 2026), these solvers retain $\mathcal{O}(nm)$ work per iteration and become impractical at the problem sizes we target. Over-relaxed Sinkhorn iterations (Thibault et al., 2021; Lehmann et al., 2022) reduce the iteration count but also retain $\mathcal{O}(nm)$ work per iteration. Nyström (Altschuler et al., 2019) and low-rank (Scetbon et al., 2021) methods approximate the kernel or plan globally rather than exploiting spatial locality. Hierarchical Refinement (Halmos et al., 2025) refines low-rank couplings recursively into a bijective Monge map in log-linear time, a different output from an EOT plan at a prescribed marginal residual.

The sparse scaling method of Schmitzer (2019) bounds the omitted pairs' contribution to the primal–dual gap (Eq. (3.7) and Propositions 10–11). Its hierarchical search combines potential maxima and cost lower bounds with entrywise refinement (Definition 13 and Algorithm 4). The reported experiments use a single CPU core (Section 5.1). Other multiscale methods address unregularized OT (Schmitzer, 2016; Gerber & Maggioni, 2017), and octree extensions apply the GeomLoss centroid test at successive resolutions (Schindler, 2025). Chowdhury et al. (2021) scale a quantized Gromov–Wasserstein variant to more than a million points. Sparse local corrections (Gasteiger et al., 2021) extend the Nyström approximation of Altschuler et al. (2019). Convolutional heat kernels on grids (Solomon et al., 2015) and nonequispaced fast Fourier transforms accelerate kernel application. Lakshmanan et al. (2023) report matrix-free Sinkhorn computations with 10^7 points in low dimension. Screenkhorn fixes selected dual variables before solving the reduced problem (Alaya et al., 2019). Native Sparse Attention (Yuan et al., 2025) uses compressed representations to score, rank and select key/value blocks. Each selected block contains consecutive tokens, allowing contiguous loads and tiled matrix computations even when selected blocks are separated in the sequence. FS2 uses spatial blocks of points and selects tiles through the potential-dependent, mass-based rule in Section 3.2.

C Proofs for Section 3

C.1 Proof of Proposition 1

Let f^c, g^c be finite coarse potentials, not necessarily converged, with coarse plan and target marginal ratios

$$P^c_{IJ} = a_I b_J e^{(f^c_I + g^c_J - C(x^c_I, y^c_J))/\varepsilon}, \qquad c_J = \frac{\sum_I P^c_{IJ}}{b_J}.$$

The first lift defines weights $w_{iJ} = b_J e^{(f_i + g^c_J - C(x_i, y^c_J))/\varepsilon}$ with $\sum_J w_{iJ} = 1$, and $\pi_{I|J} = P^c_{IJ}/(b_J c_J)$ is the conditional distribution over source centroids for target cell J, with mean $u_J = \sum_I \pi_{I|J} x^c_I$. For $j \in J$ write $z_j = y_j - y^c_J$ and

$$\phi_{iJ}(z) = \Big[\sum_I \pi_{I|J} e^{2(x^c_I - x_i)^\top z/\varepsilon}\Big]^{-1}.$$

Substituting both lifts into the fine plan and expanding the squared distances, the two copies of $e^{-\|z_j\|^2/\varepsilon}$ cancel, and

$$\frac{(\mathbf{P1})_i}{a_i} = \sum_J \frac{w_{iJ}}{c_J} \sum_{j \in J} \frac{b_j}{b_J}\, \phi_{iJ}(z_j). \tag{6}$$

Proof. Fix J and $z = z_j$, and put $\ell_I = 2(x^c_I - x_i)^\top z/\varepsilon$ and $F(s) = (\sum_I \pi_{I|J} e^{s\ell_I})^{-1}$, so that $\phi_{iJ}(z) = F(1)$. Exact centroids are convex combinations of points, so $|\ell_I| \le 2\,\mathrm{diam}(\mathbf{X}) R_Y/\varepsilon = \theta$.

With μ_s and v_s the mean and variance of ℓ_I under probabilities proportional to $\pi_{I|J}e^{s\ell_I}$, we have $F'(s) = -F(s)\mu_s$ and $F''(s) = F(s)(\mu_s^2 - v_s)$. Since $\mu_s^2 + v_s \le \theta^2$ and $F(s) \le e^{s\theta}$ on $[0,1]$, $|F''(s)| \le e^{s\theta}\theta^2$, and

$$|F(1) - 1 - F'(0)| \le \int_0^1 (1-s)e^{s\theta}\theta^2\, ds = e^\theta - 1 - \theta.$$

The cell average of $F'(0) = -2(u_J - x_i)^\top z/\varepsilon$ vanishes because y_J^c is the exact mass-weighted centroid, so $|\sum_{j\in J}(b_j/b_J)\phi_{iJ}(z_j) - 1| \le e^\theta - 1 - \theta$. Substituting into Equation (6) with $\sum_J w_{iJ} = 1$, $c_J^{-1} \le e^{\lambda_c}$ and $|c_J^{-1} - 1| \le e^{\lambda_c} - 1$ gives, for every row,

$$\Big|\frac{(\mathbf{P1})_i}{a_i} - 1\Big| \le (e^{\lambda_c} - 1) + e^{\lambda_c}(e^\theta - 1 - \theta),$$

and multiplying by a_i and summing proves Equation (3). The column bound exchanges the roles of the two clouds. □

Exact coarse convergence does not remove the second term. Let both measures put mass $1/2$ at $-h$ and h, aggregated into one cell at zero. Then $f^c = g^c = 0$ is optimal, the lifted potentials are $f_i = g_j = h^2$, and $\|\mathbf{P1} - \mathbf{a}\|_1 = \|\mathbf{P}^\top\mathbf{1} - \mathbf{b}\|_1 = \cosh(2h^2/\varepsilon) - 1 > 0$.

Size of the bound on the grid. On the 32 grid problems of Section E.3, computed on the solver's own partitions, θ ranges from 2.1 to 143 for rows and from 2.1 to 145 for columns, so the second term of Equation (3) exceeds 4.8 everywhere and the bound never accepts a lift. On the 20 problems where FS2 returned a lifted pair, the measured residual is at most 0.0041. The two terms still separate the two ways a lift fails. On clustered problems with $\varepsilon \ge 0.002$ the coarse problem is not yet converged when the lift is first rejected, and coarse updates alone lower the lifted residual. On heterogeneous problems at $\varepsilon = 0.001$ and $n \le 2^{21}$, and on clustered $n = 2^{19}$ at $\varepsilon = 0.001$, the lifted residual stays flat under extra coarse updates, and fine updates are required.

Stored centroids. With a stored representative $\widetilde{y}_J^c$ in place of the exact centroid, z_j is measured from $\widetilde{y}_J^c$ and the cell average of $F'(0)$ no longer vanishes: it adds $2(x_i - u_J)^\top \sum_{j\in J}(b_j/b_J)z_j/\varepsilon$. FS2 rounds the aggregated coordinates once to TF32, so that the same stored coordinates supply the norms, the dot products and the lift. FP32 norms of unrounded centroids combined with TF32 dot products would perturb the cost by source-only and target-only terms. Rounding at formation changes the potentials by about 1.6×10^{-4} and the lift's residual by 0.5%.

C.2 Coordinate Gradients from Coarse Potentials (Propositions 3 and 4)

At each outer step the optimization rebuilds the geometry at the current iterate $\mathbf{X}$ and anneals on the cells until it first reaches ε. The target centroids y_J^c, masses b_J and potentials g_J^c are then held fixed, so gradients do not pass through the partition, the centroids or the coarse updates. With the source coordinates replaced by $Z = (z_i)_i$, let

$$w_{iJ}(Z) = \frac{b_J \exp((g_J^c - \|z_i - y_J^c\|^2)/\varepsilon)}{\sum_L b_L \exp((g_L^c - \|z_i - y_L^c\|^2)/\varepsilon)}, \qquad \widetilde{T}_i(Z) = \sum_J w_{iJ}(Z)\, y_J^c.$$

At $Z = \mathbf{X}$ these are the lift weights of Section C.1 and the projection $\widetilde{T}_i$ of Section 3.1, and direct differentiation gives $\nabla_{z_i}\widetilde{\mathcal{L}}(Z) = 2a_i(z_i - \widetilde{T}_i(Z))$, the gradient formula of Section 3.1.

The surrogate gradient can differ from a gradient built on a fine barycentric projection in two ways. Let p_{ij} be any normalized conditional distribution over the target points for source row i, $p_{iJ} = \sum_{j\in J} p_{ij}$, and, when $p_{iJ} > 0$, $\bar{y}_{iJ} = p_{iJ}^{-1}\sum_{j\in J} p_{ij}y_j$. With $w_{iJ} = w_{iJ}(\mathbf{X})$, let $r_J = \max_{j\in J}\|y_j - y_J^c\|$ and $\mathrm{TV}_i = \frac{1}{2}\sum_J |p_{iJ} - w_{iJ}|$.

Proposition 3. *For $T_i = \sum_j p_{ij}y_j$ and $\widetilde{T}_i = \sum_J w_{iJ}y_J^c$,*

$$\|T_i - \widetilde{T}_i\| \le \sum_J p_{iJ}r_J + \mathrm{diam}(\mathbf{Y}^c)\ \mathrm{TV}_i\,. \tag{7}$$

Consequently, for $G_i = 2a_i(x_i - T_i)$ and $\widetilde{G}_i = 2a_i(x_i - \widetilde{T}_i)$,

$$\|G - \widetilde{G}\|_F \le 2\left[\sum_i a_i^2 \left(\sum_J p_{iJ} r_J + \operatorname{diam}(\mathbf{Y}^c)\,\mathrm{TV}_i\right)^2\right]^{1/2}. \tag{8}$$

Proof. Insert the intermediate barycentre $T_i^c = \sum_J p_{iJ} y_J^c$. The within-cell term satisfies

$$\|T_i - T_i^c\| = \left\|\sum_J p_{iJ}(\bar{y}_{iJ} - y_J^c)\right\| \le \sum_J p_{iJ} r_J.$$

The differences $p_{iJ} - w_{iJ}$ sum to zero, so their positive and negative parts both have total mass TV_i. If $\mathrm{TV}_i = 0$, the two centroid averages coincide. Otherwise

$$T_i^c - \widetilde{T}_i = \mathrm{TV}_i \left(\sum_J \frac{(p_{iJ} - w_{iJ})_+}{\mathrm{TV}_i} y_J^c - \sum_J \frac{(w_{iJ} - p_{iJ})_+}{\mathrm{TV}_i} y_J^c\right).$$

The two sums in parentheses are convex combinations of the same centroid set and their distance is at most its diameter. This proves $\|T_i^c - \widetilde{T}_i\| \le \operatorname{diam}(\mathbf{Y}^c)\,\mathrm{TV}_i$. The first claim follows by another triangle inequality. Multiplying row i by $2a_i$, squaring, and summing proves Equation (8). □

The first term of Equation (7) is quantization and the second the difference in conditional weights, which includes coarse solve error. With $p_{ij} = P_{ij}^\star / a_i$, G is the gradient of the converged balanced OT value.

Proposition 4. *Let $G = \nabla\mathcal{L}(\mathbf{X}) \neq 0$, let $H \neq 0$ be a supplied direction, and write $c = \langle G, H\rangle / (\|G\|_F \|H\|_F)$. Suppose the gradient of $\mathcal{L}$ is β-Lipschitz, with $\beta > 0$, on the segment from $\mathbf{X}$ to $\mathbf{X}^+ = \mathbf{X} - \eta_{\mathrm{opt}} H/\|H\|_F$. Then*

$$\mathcal{L}(\mathbf{X}^+) \le \mathcal{L}(\mathbf{X}) - \eta_{\mathrm{opt}} \|G\|_F c + \frac{\beta \eta_{\mathrm{opt}}^2}{2}. \tag{9}$$

If $c > 0$ and $0 < \eta_{\mathrm{opt}} < 2\|G\|_F c/\beta$, the objective decreases. A sufficient condition for $c > 0$ is $\|H - G\|_F \le \delta \|G\|_F$ with $0 \le \delta < 1$, which implies $c \ge \sqrt{1 - \delta^2}$.

Proof. Let $v = H/\|H\|_F$. The descent lemma for a β-smooth function gives

$$\mathcal{L}(\mathbf{X} - \eta_{\mathrm{opt}} v) \le \mathcal{L}(\mathbf{X}) - \eta_{\mathrm{opt}} \langle G, v\rangle + \frac{\beta \eta_{\mathrm{opt}}^2}{2}.$$

Since $\langle G, v\rangle = \|G\|_F c$, this is Equation (9), and the stated step-size condition makes its right-hand side strictly smaller than $\mathcal{L}(\mathbf{X})$.

For the relative-error claim, set $u = G/\|G\|_F$, $h = H/\|G\|_F$, and $r = \|h\|_F > 0$. If $c = \langle u, h\rangle / r$, then $\|h - u\|_F^2 = r^2 + 1 - 2rc \le \delta^2$, and therefore

$$c \ge \frac{r^2 + 1 - \delta^2}{2r} = \frac{1}{2}\left(r + \frac{1 - \delta^2}{r}\right) \ge \sqrt{1 - \delta^2},$$

where the last step is the arithmetic–geometric mean inequality. □

One surrogate-gradient evaluation visits all K_y target cells per source point and runs no fine update. With sorting, aggregation and S_c coarse updates, an outer step costs

$$\mathcal{O}((n+m)\log(n+m) + S_c K_x K_y d + n K_y d), \tag{10}$$

where $K_x \le 2\lceil n/B\rceil$ and $K_y \le 2\lceil m/B\rceil$.

C.3 Proof of Theorem 1

Proof. Write the plan at the current potentials, with no convergence assumed, as

$$P_{ij}(\mathbf{f},\mathbf{g}) = a_i\, b_j \exp\big((f_i + g_j - C_{ij})/\varepsilon\big). \tag{11}$$

Every exponent in the tile is at most m_{IJ}, so $P_{ij} \le a_i b_j e^{m_{IJ}}$ for $i \in I$, $j \in J$, and summing over the opposite block gives

$$\sum_{j' \in J} P_{ij'} \le a_i\, b_J\, e^{m_{IJ}}, \qquad \sum_{i' \in I} P_{i'j} \le b_j\, a_I\, e^{m_{IJ}}. \tag{12}$$

Since $e^{\rho_{IJ}} = \delta_{\mathrm{rel}}/\max(a_I, b_J)$ and both block masses are at most $\max(a_I, b_J)$, the hypothesis $m_{IJ} \le \rho_{IJ}$ bounds the two right-hand sides by $\delta_{\mathrm{rel}} a_i$ and $\delta_{\mathrm{rel}} b_j$. □

One bound thus serves both directions: the row bound carries the factor $b_J/\max(a_I, b_J)$ and the column bound $a_I/\max(a_I, b_J)$, and one of the two is always 1. The box screen is conservative, because $f_i \le f_I^\vee$, $g_j \le g_J^\vee$ and $C_{ij} \ge d^2_{\min}(I, J)$ give

$$m_{IJ} \le \big(f_I^\vee + g_J^\vee - d^2_{\min}(I, J)\big)/\varepsilon, \tag{13}$$

so a pair that fails Equation (5) has $m_{IJ} \le \rho_{IJ}$. Both bounds use only pointwise inequalities at the current potentials and hold at every level of the schedule. The screen depends on the potentials only through $f_I^\vee + g_J^\vee$, so it is invariant under the shift $\mathbf{f} \mapsto \mathbf{f} + \kappa_0$, $\mathbf{g} \mapsto \mathbf{g} - \kappa_0$, which leaves the plan unchanged. A rule built on $\max(f_I^\vee, g_J^\vee)$ would give different masks for these equivalent potential pairs.

Non-uniform weights. Every benchmark uses uniform weights, for which ρ_{IJ} is the constant $\log(\tau/4)$ whenever B divides $n = m$. To exercise the dependence on block mass, we rerun the two problems of Table 5 with the non-uniform weights of FlashSinkhorn's public benchmark code (Ye et al., 2026), $a_i \propto u_i + 0.1$ with u_i uniform on $[0, 1]$ and $\mathbf{b}$ drawn independently, and with a rank$^{-1/2}$ power law. We compare the mass-dependent rule with a scalar rule that uses the largest block mass for every pair, which also certifies. FS2 meets τ in all four cases. Every fine update is sparse, the per-point bound that Theorem 1 implies, at most $\delta_{\mathrm{rel}} a_i$ per omitted tile in row i and likewise for columns, holds everywhere, and an FP64 exact test on sampled block rows agrees with the refined masks tile for tile. The mass-dependent rule admits 0.3–0.5% fewer block pairs than the scalar rule under the first weighting, whose block masses differ by at most a factor 1.4, and 2.9–5.3% fewer under the power law (factor 6.4–7.0). Both rules take the same number of fine updates, and the mass-dependent solve is at most 3.3% faster.

C.4 Proof of Proposition 2

Proof. (i) Write $S_i = \sum_j b_j e^{(g_j - C_{ij})/\varepsilon}$. The common factor $a_i e^{f_i/\varepsilon}$ cancels from the omitted mass ratio, so the restricted sum is $S_i^{\mathcal{A}} = (1 - \nu_i) S_i$ and

$$f_i^{+,\mathcal{A}} - f_i^{+} = -\varepsilon \log S_i^{\mathcal{A}} + \varepsilon \log S_i = -\varepsilon \log(1 - \nu_i).$$

The column update is identical.

(ii) The masked projection $T_\varepsilon^{\mathcal{A}}(x_i)$ restricts both sums of $T_\varepsilon(x_i) = \sum_j P_{ij} y_j / \sum_j P_{ij}$ to the admitted columns. Subtracting it from the full projection gives the exact identity

$$T_\varepsilon(x_i) - T_\varepsilon^{\mathcal{A}}(x_i) = \frac{\sum_{J:(I,J)\notin\mathcal{A}} \sum_{j\in J} P_{ij}\big(y_j - T_\varepsilon^{\mathcal{A}}(x_i)\big)}{\sum_j P_{ij}}, \tag{14}$$

whose denominator sums over all targets, and the triangle inequality gives

$$\|T_\varepsilon^{\mathcal{A}}(x_i) - T_\varepsilon(x_i)\| \le \frac{\sum_{J:(I,J)\notin\mathcal{A}} \sum_{j\in J} P_{ij} \|y_j - T_\varepsilon^{\mathcal{A}}(x_i)\|}{\sum_j P_{ij}} \le \nu_i \operatorname{diam}(\mathbf{Y}). \tag{15}$$

□

Table 5: Proposition 2 at FS2's returned potentials, $n = m = 2^{20}$, $\varepsilon = 0.001$: largest values over all rows. The ν_i column divides by the Theorem 1 bound $M_B \delta_{\mathrm{rel}} a_i / r_i$, the error column by $\nu_i \operatorname{diam}(\mathbf{Y})$, and the gradient compares $2a_i(x_i - T_\varepsilon^{\mathcal{A}}(x_i))$ with $2a_i(x_i - T_\varepsilon(x_i))$.

Cloud	δ_{rel} factor	Admitted	Max ν_i	ν_i over Thm. 1 bound	Max error	$\|T_\varepsilon^{\mathcal{A}} - T_\varepsilon\|$ over $\nu_i \operatorname{diam}(\mathbf{Y})$	Gradient rel. ℓ_2
Clustered	1	6.33%	6.7×10^{-6}	0.005	2.7×10^{-6}	0.48	1.6×10^{-7}
Clustered	64	5.32%	3.7×10^{-4}	0.005	1.4×10^{-4}	0.48	1.3×10^{-5}
Heterog.	1	60.54%	2.9×10^{-5}	0.023	2.4×10^{-6}	0.23	9.1×10^{-6}
Heterog.	64	49.24%	3.0×10^{-3}	0.038	1.7×10^{-4}	0.23	1.8×10^{-3}

The middle term of Equation (15) keeps the distance of each omitted contribution, so it is much smaller than the diameter bound when the omitted mass lies near the masked projection. The diameter bound is nevertheless sharp. Take two identical one-dimensional clouds, each with B points at 0 and B at $D > 0$, uniform weights and one block per location. At the balanced optimum every row sends the fraction $\nu_i = (1 + e^{D^2/\varepsilon})^{-1}$ to the opposite location, so when $\nu_i \le \delta_{\mathrm{rel}} < 1 - \nu_i$ the admission rule keeps only the two same-location tiles. For $x_i = 0$, $T_\varepsilon^{\mathcal{A}}(x_i) = 0$ while $T_\varepsilon(x_i) = \nu_i D$. By (ii), the source gradient $2a_i(x_i - T_\varepsilon(x_i))$ changes by at most $2a_i \nu_i \operatorname{diam}(\mathbf{Y})$ when the masked projection is used.

Measured error. Table 5 evaluates Proposition 2 at the potentials FS2 returns on the clustered and heterogeneous $n = m = 2^{20}$, $\varepsilon = 0.001$ problems. A mask is built at those potentials by the admission rule with the production δ_{rel}, and with $64\delta_{\mathrm{rel}}$ as a positive control. One FP64 pass over all $n \times m$ pairs splits each row's mass and first moment into admitted and omitted sums, so no reported quantity is a difference of two large numbers, and $\operatorname{diam}(\mathbf{Y})$ comes from the convex hull of $\mathbf{Y}$. No row exceeds either bound. The looser mask raises the largest error 50–70-fold, so the production values are not a floor of the measurement. The block-sparse gradient kernel, run on the production mask, adds its own arithmetic error: against the FP64 masked gradient it is within 1.0×10^{-4} (clustered) and 1.0×10^{-2} (heterogeneous) in relative ℓ_2 in FP32, and within 2.1×10^{-3} and 0.13 in TF32, far above the masking error of the table.

C.5 IO Model (Proposition 5)

The dense FlashSinkhorn kernel moves $\Theta(nd + md + nmd^2/M)$ scalars between HBM and fast memory of size M (Ye et al., 2026). For the sparse traversal of Algorithm 3, the count is governed by neighbour-record misses.

Proposition 5. *Let a neighbour block record occupy $\sigma = \Theta(B(d+1))$ scalars. Suppose one source block's queries and accumulators stay resident in fast memory of size $M \ge c_0\sigma$ for a fixed constant c_0, costs are computed online, CSR indices are read once per traversal, and caches are cold at the start of each half-step. If L_f and L_g count neighbour-record misses along the CSR and its transpose, one iteration evaluating both directions moves*

$$Q_{\mathrm{iter}} = \Theta\big((n+m)(d+1) + |\mathcal{A}| + \sigma[L_f + L_g]\big) \tag{16}$$

scalars between slow and fast memory. Each miss count lies between the number of active neighbour blocks in its traversal and $|\mathcal{A}|$.

Proof. We count the $\hat{\mathbf{f}}$ half-step, and the $\hat{\mathbf{g}}$ half-step is the same count on the transposed index. Each of the N_B source blocks moves its queries $\mathbf{X}_I$ and writes back $\hat{\mathbf{f}}_I$ once, $\Theta(B(d+1))$ scalars, because both stay resident for the whole row. Over both directions this is $\Theta((n+m)(d+1))$. The traversal reads crow once and each entry of col once, $\Theta(N_B + |\mathcal{A}|)$ scalars, and $N_B \le n$ is absorbed by the first term. An empty row reads no neighbour record. Its repair is a dense reduction, counted with the dense updates in Equation (20). Visiting an admitted pair needs $\mathbf{Y}_J$, $\hat{\mathbf{g}}_J$ and $\boldsymbol{\delta}_J$, $\sigma = \Theta(B(d+1))$ scalars, and moves nothing when the record is already resident. The L_f visits that miss contribute $\Theta(\sigma L_f)$. Summing gives Equation (16). Each half-step starts cold, so every active neighbour block misses at least once. At most one miss is charged per admitted pair, so L_f lies between the number of active neighbour blocks and $|\mathcal{A}|$, and so does L_g. □

On-chip memory is private to each program, and within one row the CSR entries are distinct, so a neighbour record can be reused across source blocks only through L2. The upper end, one gather per admitted pair, is therefore the bound that always holds, and unlike the dense case, enlarging M beyond one record does not reduce it: the admitted columns of a row form an arbitrary subset, with no Cartesian block of queries and keys to stage. How much L2 retains depends on the traversal order.

D Implementation Details

D.1 Block Size and Coarse Cells

The automatic policy chooses the block size from $B \in \{64, 128, 256\}$ by estimating the memory of the block summaries. Each summary holds a bounding box, a potential maximum and a log block mass, 32 bytes for $d = 3$, so the summaries of both clouds take $32(N_B + M_B)$ bytes, with $N_B = \lceil n/B \rceil$ and $M_B = \lceil m/B \rceil$. The chosen B keeps this estimate within $0.8C_{\mathrm{L2}}$, where C_{L2} is the physical L2 cache size, so the budget forces larger blocks on larger clouds.

For each $t \in \{0, \dots, 62\}$, we count the occupied prefixes of the sorted 63-bit Morton keys. Smaller t gives a finer partition. We choose the smallest t satisfying

$$\begin{gathered} K_x \le 2N_B, \qquad K_y \le 2M_B, \qquad 112(K_x + K_y) \le 0.8C_{\mathrm{L2}}, \\ 16\max(K_x, K_y, BN_B, BM_B) \le 2^{31} - 1. \end{gathered} \tag{17}$$

The first two conditions limit the coarse point count relative to the execution blocks. The third limits estimated storage for coarse arrays, and the fourth checks signed 32-bit offsets for coordinates padded to width 16. The fourth errs by one on the safe side, so it excludes $n = 2^{27}$, whose largest offset $16n - 1 = 2^{31} - 1$ still fits. Cell counts come from the sorted keys, without solving a coarse problem for each t. The budget constants are empirical. They make the cells and blocks as fine as memory allows, which raises memory traffic.

Morton-prefix cells have geometric extents determined by the retained bits. Dropping a multiple of d bits gives cubes in normalized coordinates, while other truncations give boxes whose side lengths differ by at most a factor of two. Fixed-population blocks have no comparable diameter guarantee.

D.2 Forward Solver and Stopping Rules

Sampled check. In exact arithmetic $(\mathbf{P1})_i / a_i = \exp((f_i - f_i^+)/\varepsilon)$, where f_i^+ is the unrestricted row update, so for potentials fixed independently of the sample, drawing i with probability a_i gives

$$\mathbb{E}_{i\sim\mathbf{a}} \left|\exp((f_i - f_i^+)/\varepsilon) - 1\right| = \sum_i a_i \left|(\mathbf{P1})_i/a_i - 1\right| = \|\mathbf{P1} - \mathbf{a}\|_1, \tag{18}$$

and columns drawn with $j \sim \mathbf{b}$ estimate $\|\mathbf{P}^\top \mathbf{1} - \mathbf{b}\|_1$ in the same way. The check draws 1024 indices per side with seed 0, reduces each sampled marginal over all opposite points, and bounds each residual by its sample mean plus three standard errors. It passes when the larger bound is below τ and fails when a mean minus three standard errors exceeds τ. Otherwise it evaluates 8192 further indices and passes only if their bound is below τ. No sample size gives a distribution-free guarantee on a mean (Bahadur & Savage, 1956). With uniform weights, S draws miss a given row with probability $(1 - 1/n)^S$, so the check can accept potentials whose full residual exceeds τ. The solver draws the indices once per solve, reuses them at every check and stops at the first check that passes, so the potentials at the stopping check depend on the sample, whereas Equation (18) assumes they do not. A finite-sample guarantee would need a certified variance bound (Catoni, 2012). Every reported residual therefore comes from a full evaluation, except at two intermediate Quijote sizes. Section E.4 measures how often the check accepts such potentials on the grid and evaluates the certificate of Section 3.2 at returned potentials.

Coarse stage. The coarse solve anneals on the Morton-prefix cells of Section 3 with GeomLoss's ε-scaling schedule, from the diagonal of the joint bounding box down to the blur $\sqrt{\varepsilon}$ with scaling 0.95 and averaged updates, and lifts when it first reaches the requested ε. If the sampled check rejects the lift, the solver completes the 64-entry target-ε tail, lifts and checks again, and then continues in

batches of 64 coarse updates. After each batch it evaluates, from the shifted coarse potentials before and after the last averaged update,

$$D_c = \frac{2}{\varepsilon} \max\Big\{ \sum_I a_I \big|\hat{f}_I^{c+} - \hat{f}_I^{c}\big|,\ \sum_J b_J \big|\hat{g}_J^{c+} - \hat{g}_J^{c}\big| \Big\}, \tag{19}$$

accumulated in FP64. In exact arithmetic, because an averaged update moves each potential halfway to its full update, D_c is the larger of the coarse log-marginal errors $\sum_I a_I |\log(\sum_J P^c_{IJ}/a_I)|$ and $\sum_J b_J |\log c_J|$ before that update, with P^c and c_J as in Section C.1. These are weighted means, whereas λ_c in Proposition 1 is a maximum over target cells, so the stop does not bound the first term of Equation (3). Extra coarse work stops after two consecutive batches with $D_c \le \tau/2$, three consecutive relative improvements below 1%, or 2048 extra updates. A non-finite D_c discards the extra batches. The refined state is lifted and checked.

Fine stage. The lifted pair initializes the fine updates without another annealing schedule. Because Theorem 1 certifies a mask only at the potentials that built it, the mask is rebuilt before every fine update. The two symmetric half-steps start from the same incoming potentials, so one mask serves both. The first two ordinary updates are checked one by one. Afterwards the check interval is the smallest of 1, 2, 4, 8 and 16 updates whose combined time is at least ten times that of a check, and 1 once the 1024-index bound is at most 2τ. Checks also fall at every 16th update and at the cap. A terminal preview forms unaveraged potentials from the same incoming pair after the re-lift that ends the coarse continuation, after the second ordinary update, at every 16th ordinary update once the 1024-index bound is at most 2τ, and at the cap. A rejected preview does not enter the ordinary trajectory. Each preview is a complete fine update with its own mask, and the fine-update counts of Section 4.1 include previews.

Return. The solver returns as soon as a sampled check passes. At the fine-update cap, 128 ordinary updates unless stated otherwise, it returns the checked potentials with the best finite sample score and reports the tolerance as not confirmed. If no score is finite, it returns the retained lift.

D.3 Mask Construction and Sparse Half-Steps

Algorithm 2 builds the candidates of the box screen Equation (5) in CSR form, and verification then evaluates m_{IJ} on each candidate tile. Algorithm 3 gives the sparse $\hat{\mathbf{f}}$ half-step over the admitted blocks, and the $\hat{\mathbf{g}}$ half-step runs the same kernel on the transposed index.

All forward solves form the dot products on TF32 tensor cores with FP32 accumulation and keep the potentials in FP32. The bias $(\hat{g}_j + \varepsilon \log b_j)/\varepsilon$ is added in FP32 after the TF32 dot product. Folding it into the dot as an extra coordinate would round an $\mathcal{O}(1/\varepsilon)$ operand to TF32's ten mantissa bits, an absolute logit error of about 5 at $\varepsilon = 10^{-4}$, and would not reduce the MMA instruction count.

D.4 Operation Count

Let S_c count all coarse updates, including annealing and additional iterations at the requested ε, and let L_{lift} count paired lifts. Each contributes respectively $\mathcal{O}(K_x K_y d)$ and $\mathcal{O}((nK_y + mK_x)d)$ operations. For sampled checks, define $Q_{\text{check}} = \sum_\ell (mS_{x,\ell} + nS_{y,\ell})$, where $S_{x,\ell}, S_{y,\ell}$ are the actual sample sizes in check ℓ. Both stages of an enlarged check contribute to this sum.

For fine update k, let $\mathcal{C}_k$ be the bounding-box candidates and $\mathcal{A}_k$ the verified tiles. Computing potential maxima and scanning block pairs costs $\mathcal{O}(n + m + N_B M_B d)$. Verification and the two sparse half-steps cost $\mathcal{O}(B^2 d(|\mathcal{C}_k| + 2|\mathcal{A}_k|))$. Writing and transposing the indices adds $\mathcal{O}(N_B + M_B + E_k \log(2 + E_k))$ as a conservative bound including sorting, where $E_k = |\mathcal{C}_k| + |\mathcal{A}_k|$. Let Q_{dense} count point pairs evaluated by dense updates and empty-row repairs. The complete arithmetic and index work is bounded by

$$\begin{aligned} \mathcal{O}\Big(& (n+m)\log(n+m) + (n+m)d \\ & + d\{S_c K_x K_y + L_{\text{lift}}(nK_y + mK_x) + Q_{\text{check}} + Q_{\text{dense}}\} \\ & + \sum_k \{n + m + N_B M_B d + B^2 d(|\mathcal{C}_k| + 2|\mathcal{A}_k|) + E_k \log(2 + E_k)\}\Big). \end{aligned} \tag{20}$$

Algorithm 2 BUILDMASK: Equation (5) as two Triton passes over block rows, one counting and one writing. Both scan the full $N_B \times M_B$ grid with the predicate in registers. Each tile test costs about $6d + 3$ operations, about 0.010 per point pair at $d = 3$ and $B = 64$.

Require: boxes $[\mathbf{x}^I_{\text{lo}}, \mathbf{x}^I_{\text{hi}}]$, $[\mathbf{y}^J_{\text{lo}}, \mathbf{y}^J_{\text{hi}}]$, block maxima $f^\vee, g^\vee$, log masses $\log a, \log b$, ε, δ_{rel}
Ensure: CSR arrays crow, col

for each block row I **in parallel do**
  $c_I \leftarrow 0$ *// pass 1: count*
  for each tile $\mathcal{J}$ of column blocks **do**
    $W \leftarrow \max\big(\mathbf{x}^I_{\text{lo}} - \mathbf{y}^{\mathcal{J}}_{\text{hi}},\ \mathbf{y}^{\mathcal{J}}_{\text{lo}} - \mathbf{x}^I_{\text{hi}},\ 0\big)$ *// per-axis box separation*
    $d^2_{\min} \leftarrow \text{rowsum}(W \odot W)$ *// lower-bounds every pair in the tile*
    $\text{keep} \leftarrow \big[\, d^2_{\min} < f^\vee_I + g^\vee_{\mathcal{J}} - \varepsilon\, \rho_{I\mathcal{J}} \,\big]$ *// Equation (5), $\log a_I$ is hoisted*
    $c_I \leftarrow c_I + \text{sum}(\text{keep})$
  end for
  write c_I to HBM
end for
$\text{crow} \leftarrow \text{ExclusiveScan}(c)$ *// offsets exist before any row writes*
if $\text{crow}[N_B]$ exceeds the index type **then raise** *// refuse rather than wrap silently*
for each block row I **in parallel do**
  $\text{base} \leftarrow \text{crow}[I],\ w \leftarrow 0$ *// pass 2: fill*
  for each tile $\mathcal{J}$ of column blocks **do**
    recompute keep in registers *// cheaper than storing a dense Boolean mask*
    $\text{pos} \leftarrow \text{ExclusiveScan}(\text{keep})$ *// slot inside this tile*
    $\text{col}[\text{base} + w + \text{pos}] \leftarrow \mathcal{J}$ **where** keep *// one program per row, so ascending*
    $w \leftarrow w + \text{sum}(\text{keep})$
  end for
end for

Algorithm 3 SPARSELSE: the streaming $\hat{\mathbf{f}}$-update over the admitted blocks. The online max/sum-exp recurrence is FlashSinkhorn's, run here over a CSR row rather than over every column block. The cost is never formed either way.

Require: $\mathbf{X} \in \mathbb{R}^{n\times d}$, $\mathbf{Y} \in \mathbb{R}^{m\times d}$, $\hat{\mathbf{g}} \in \mathbb{R}^m$, $\mathbf{b} \in \Delta^m$, CSR $(\text{crow}, \text{col})$, ε
Ensure: $\hat{\mathbf{f}} \in \mathbb{R}^n$

$\boldsymbol{\delta} \leftarrow \varepsilon \log \mathbf{b}$ *// precompute the weight bias*
for each block row I of size B **do**
  load $\mathbf{X}_I$ to SRAM, $\mathbf{m}_I \leftarrow -\infty,\ \mathbf{s}_I \leftarrow 0$ *// running max and sum-exp*
  $r_0 \leftarrow \text{crow}[I],\ \text{nnz}_I \leftarrow \text{crow}[I{+}1] - r_0$
  for $t = 0, \ldots, \text{nnz}_I - 1$ **do**
    $J \leftarrow \text{col}[r_0 + t]$, load $\mathbf{Y}_J, \hat{\mathbf{g}}_J, \boldsymbol{\delta}_J$ to SRAM *// admitted blocks only*
    $\mathbf{Z} \leftarrow (2\,\mathbf{X}_I\mathbf{Y}_J^\top + \hat{\mathbf{g}}_J + \boldsymbol{\delta}_J)/\varepsilon$ *// one kernel tile, the cost is never formed*
    $\mathbf{m}_{\text{new}} \leftarrow \max(\mathbf{m}_I,\ \text{rowmax}(\mathbf{Z}))$
    $\mathbf{s}_I \leftarrow e^{\mathbf{m}_I - \mathbf{m}_{\text{new}}} \odot \mathbf{s}_I + \text{rowsum}\big(e^{\mathbf{Z} - \mathbf{m}_{\text{new}}}\big),\ \mathbf{m}_I \leftarrow \mathbf{m}_{\text{new}}$
  end for
  $\hat{\mathbf{f}}_I \leftarrow -\varepsilon\big(\mathbf{m}_I + \log \mathbf{s}_I\big)$, write to HBM *// $-\varepsilon\,\text{LSE}_{\text{row}}$ over the admitted blocks*
end for

The sum runs over ordinary updates and terminal previews, rejected ones included, and each dense update adds its nm evaluations to Q_{dense}. No score matrix is stored: beyond the point and coarse arrays, memory holds the candidate and admitted CSR indices, and an index with z nonzeros is allocated only if it fits signed 32-bit offsets and $24z + 8(N_B + M_B + 2) \le 0.8M_{\text{free}}$ bytes, with M_{free} the free GPU memory. The selector's limits give $K_xK_y \le 4N_BM_B$ and $nK_y + mK_x \le 2(nM_B + mN_B)$. Both remain quadratic in n at fixed B.

E SYNTHETIC BENCHMARKS

After the setup and the GeomLoss baseline, the subsections follow the paragraphs of Section 4.1 in order, each with its own protocol.

E.1 SETUP

Unless stated otherwise, every run uses one A100 with PyTorch 2.8.0+cu128, Triton 3.4.0, GeomLoss 0.3.1 and PyKeOps 2.3. The kernel profiles of Section E.6 use a separate A100. Inputs are three-dimensional with uniform weights and seed zero, and all methods receive the same centered, TF32-rounded coordinates and the full squared-Euclidean cost, without a factor of $1/2$, so GeomLoss's default blur of 0.05 is the grid point $\varepsilon = 0.005$. Times are warmed complete calls, from preparation and parameter selection to return, and exclude input loading, compilation and validation. Grid endpoints are single calls, and comparisons and ablations report medians of three. Accuracy is the full residual R of the returned potentials (Table 4), evaluated on all points, against $\tau = 0.005$. Peak memory is the largest PyTorch allocation, inputs included, in a separate warmed call.

Methods compared.

- *FS2*: the automatic policy of Section D, which chooses $B = 128$ over 64 on every problem and, for $n = 2^{19}, \dots, 2^{23}$, the cell levels $t = 46$, 45, 44, 43 and 42 on clustered and 43, 42, 41, 40 and 39 on heterogeneous clouds.
- *GeomLoss multiscale*: its native multiscale solver with scaling 0.9, default cluster scale and truncation 5, with KeOps for the fine reductions.
- *FlashSinkhorn*: its native dense solver with scaling 0.9.

Neither baseline gets added refinement or a residual-based stopping rule.

Data. A clustered cloud draws 32 centers uniformly in $[0, 1]^3$ and places each point at a uniformly chosen center plus isotropic Gaussian noise of standard deviation 0.02. Source and target draw their centers independently. A heterogeneous cloud mixes three components: 80% of the points in a compact Gaussian at $(-0.35, -0.15, 0)$ with standard deviations $(0.04, 0.025, 0.02)$, 19% in a thin needle at $(0.35, 0.10, 0)$ with standard deviations $(0.20, 0.008, 0.008)$, rotated by 0.6 rad about the z-axis, and 1% uniform background in $[-0.9, 0.9]^3$. The target shifts the two Gaussian centers by $(0.025, -0.01, 0.008)$ and $(-0.018, 0.018, -0.006)$ and is drawn independently. Both families are centered and rounded once to TF32 before any solve.

E.2 GEOMLOSS MULTISCALE WORKFLOW

The evaluated GeomLoss 0.3.1 point-cloud backend (Feydy et al., 2019a; Feydy, 2020) uses two spatial resolutions: weighted cell centroids and the original points. Its default grid clustering supports $d \leq 3$. Each cell of side ℓ_{cell} is represented by its mass-weighted centroid and total mass. Sorting fine points by cell gives contiguous memory ranges whose lengths vary with local density.

The solver follows $\varepsilon_1 \geq \cdots \geq \varepsilon_{N_\varepsilon}$, initializing each level with the preceding potentials. It starts on the centroid measures and switches to the original points just before the first level whose blur falls below ℓ_{cell}. At this transition, an LSE against the coarse records lifts the potentials. If the schedule never reaches a blur below the cell scale, GeomLoss completes the coarse solve and lifts its final potentials to the original points.

When fine updates are needed, the mask is built at the last coarse regularization level, indexed by $p^\star$:

$$\mathcal{A}^{\text{GL}} = \{(I, J) : C(x_I^c, y_J^c) < f_I^c + g_J^c + 5\varepsilon_{p^\star}\}, \tag{21}$$

where x_I^c, y_J^c are cell centroids, f_I^c, g_J^c are coarse potentials, and 5 is GeomLoss's default truncation parameter. The retained cell pairs become KeOps index ranges (Charlier et al., 2021). Subsequent fine LSE updates reuse these ranges without materializing the cost matrix, while the lift sums over all coarse neighbours independently of the mask. At a terminal coarse-to-fine transition, the implementation only lifts the final coarse potentials and takes no fine update.

Omitted mass. One FP64 pass over all pairs of the two problems of Table 5 measures the plan mass each mask omits, with admitted and omitted sums accumulated apart. At FS2's returned potentials

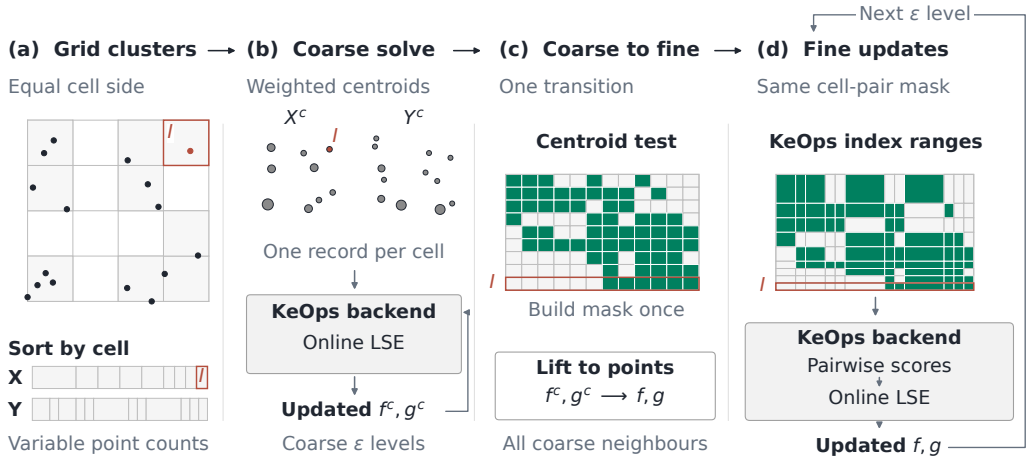


Figure 4: **GeomLoss multiscale workflow** (Feydy et al., 2019a; Feydy, 2020; Charlier et al., 2021). The input points match Figure 1. A single centroid test fixes the ranges for fine KeOps updates, while the lift uses all coarse neighbours. Red traces cell I, and green marks retained pairs.

on clustered data, the GeomLoss mask, built at its switch to the points and reused, omits more than half the plan mass of 964 of the 2^{20} rows and nearly all of it in 12, while FS2's mask omits at most 6.7×10^{-6} of any row and keeps half as many point pairs. This difference includes the stage at which GeomLoss builds its mask, not only its criterion. At GeomLoss's own returned potentials its mask omits 9.8×10^{-6} of the mass, 0.02% of its residual of 0.054, so that residual is not mainly omitted mass. On heterogeneous data GeomLoss's mask omits at most 7×10^{-18} of any row, and FS2's at most 2.9×10^{-5}.

Figure 4 illustrates the balanced cross-measure forward path. The cell side is chosen for illustration, using the same input coordinates and weights as Figure 1.

E.3 Speed

The grid covers clustered and heterogeneous clouds: a size sweep over $n = m \in \{2^{19}, \dots, 2^{23}\}$ at $\varepsilon \in \{0.001, 0.01\}$ and a regularization sweep over $\varepsilon \in \{0.001, 0.002, 0.005, 0.01, 0.02\}$ at $n = m \in \{2^{21}, 2^{22}\}$. Eight problems are shared, which leaves 32 inputs (Table 6). Figure 2(a–d) shows the size sweep and the $n = 2^{22}$ regularization sweep, 26 problems. FS2 runs its automatic policy (Sections D.1 and D.2) with one fixed kernel calibration and no per-problem tuning.

Trajectories. Figure 2(f) plots, on clustered $n = m = 2^{22}$, the full residual of FS2's potentials each time they improve on the best so far (40 times at $\varepsilon = 0.001$, 18 at $\varepsilon = 0.01$). The potentials are lifted to the points every 64 coarse updates and taken after every fine update, and each is placed at the time in the solve at which it is available, its own lift included. GeomLoss's native returns are markers, and FS2 enters the fine stage only at $\varepsilon = 0.001$.

GeomLoss settings. The grid runs GeomLoss at scaling 0.9 with its default cluster scale and truncation 5. Table 7 varies the scaling and the cluster scale on two $n = m = 2^{21}$ problems and reruns FS2 in the same session, reproducing the grid's FS2 and GeomLoss endpoints. The matched cluster scale gives GeomLoss as many source cells as FS2's coarse stage uses, to within one. The count is taken from FS2's record and not charged to GeomLoss. At this scale, GeomLoss's schedule stays on the centroids and ends with a lift. On clustered data at $\varepsilon = 0.001$ none of the six settings meets τ. The best, scaling 0.99 at the default cluster scale, returns 0.0069 after 200 s, 13 times FS2's time, and the matched coarse exits return 0.017–0.060. FS2's own coarse stage also misses τ on this problem, so here the target needs fine updates, and GeomLoss's fine stage in these settings does not reach it. On heterogeneous data at $\varepsilon = 0.01$ all six settings meet τ. At the default cluster scale GeomLoss switches to the points for the last six of its 36 levels and takes 57.8 s, 212 times FS2. Given FS2's cell count, GeomLoss's coarse exit meets τ in 0.39 s, 1.4 times FS2, with residual 7.6×10^{-4} against FS2's 9.5×10^{-4}. The two scaling-0.9 runs differ only in the cluster scale, so on this problem the

Table 6: The 32-problem forward grid. A dagger marks $R > 0.005$. FS2 meets the target on all 32 problems and GeomLoss on 10, all heterogeneous. Each entry is one call. Table 1a and Table 7 remeasure clustered 2^{21}, $\varepsilon = 0.001$ as medians of three calls.

$n = m$	ε	FS2 time (s)	FS2 R	GL time (s)	GL R
Clustered					
2^{19}	0.001	5.638	0.004879	1.521	0.054122†
2^{19}	0.01	0.290	0.002918	0.026	0.072730†
2^{20}	0.001	4.176	0.004791	5.500	0.053802†
2^{20}	0.01	0.275	0.003245	0.027	0.071918†
2^{21}	0.001	15.593	0.004828	21.448	0.054161†
2^{21}	0.002	5.380	0.004094	14.305	0.055280†
2^{21}	0.005	2.318	0.003198	7.167	0.056265†
2^{21}	0.01	0.812	0.002673	0.031	0.073064†
2^{21}	0.02	0.646	0.001066	0.029	0.046715†
2^{22}	0.001	20.289	0.004639	85.484	0.053612†
2^{22}	0.002	9.843	0.004871	56.868	0.054748†
2^{22}	0.005	3.676	0.004129	28.455	0.055743†
2^{22}	0.01	2.825	0.001989	0.037	0.070512†
2^{22}	0.02	2.248	0.000810	0.037	0.046228†
2^{23}	0.001	45.405	0.004694	346.051	0.054192†
2^{23}	0.01	9.914	0.001276	0.052	0.071911†
Heterogeneous					
2^{19}	0.001	0.929	0.003605	10.547	0.012514†
2^{19}	0.01	0.045	0.001944	3.314	0.002151
2^{20}	0.001	1.824	0.004342	39.628	0.013687†
2^{20}	0.01	0.106	0.000970	14.886	0.000886
2^{21}	0.001	7.060	0.003875	154.122	0.013793†
2^{21}	0.002	0.302	0.001911	125.213	0.007890†
2^{21}	0.005	0.285	0.000962	86.684	0.002550
2^{21}	0.01	0.272	0.000949	57.820	0.000841
2^{21}	0.02	0.261	0.000548	19.300	0.001679
2^{22}	0.001	1.096	0.003059	612.415	0.013734†
2^{22}	0.002	1.053	0.001056	497.391	0.007850†
2^{22}	0.005	0.992	0.000644	344.334	0.002531
2^{22}	0.01	0.947	0.000880	229.573	0.000846
2^{22}	0.02	0.908	0.000531	76.566	0.001682
2^{23}	0.001	4.026	0.001624	2595.417	0.003906
2^{23}	0.01	3.484	0.000885	915.975	0.000883

gap under the default setting comes from the cluster scale. FS2 chooses its resolution from the input and stops when its sampled check accepts, while the matched runs receive the resolution and carry no test.

GeomLoss continued at the target ε. The grid gives GeomLoss no residual-based stopping rule, so here we give it one that favours it. We append K copies of the target ε to its native schedule, an option its interface does not expose, and leave its jump rule and truncation unchanged. We then stop at K^*, found by scoring $K = 0, 1, 2, 4, \ldots, 2048$ and bisecting the first bracket that meets τ. This oracle stop charges GeomLoss for no residual test, whereas FS2's times include its sampled checks. On clustered $\varepsilon = 0.001$ the setting that came closest above is scaling 0.99 at the default cluster scale. It reaches τ after 118 extra entries in 421 s, 27.3 times FS2's 15.4 s (Table 8). The grid's setting, scaling 0.9 at the default cluster scale, reaches τ at none of the scored K. Its residual falls to 0.020 after 512 entries and then rises to 75 and 3.5×10^3 at 1024 and 2048. Given FS2's cell count, GeomLoss's coarse exit continued for 2048 entries reaches 0.016 and is still falling. At $\varepsilon = 0.01$ the matched setting reaches τ after 183 entries in 1.13 s, 1.38 times FS2's 0.82 s, while both default-cell settings level off at 0.037. Both results are specific to these two problems.

Table 7: GeomLoss settings on two $n{=}m{=}2^{21}$ problems, truncation 5, rerun with FS2 in the same session. Bold marks solves that meet $\tau{=}0.005$. Default is GeomLoss's cluster scale (430 and 2033 source cells), and matched gives as many source cells as FS2's coarse stage, to within one (23,799 against 23,800, and 28,332). FS2 and matched times are medians of three calls, the others single calls. R is the full two-sided residual.

		Clustered, $\varepsilon{=}0.001$		Heterogeneous, $\varepsilon{=}0.01$	
Solver	Cells	Time (s)	R	Time (s)	R
FS2	auto	15.37	**.00483**	0.273	**.00095**
GL, scaling .90	default	21.46	.0542	57.80	**.00084**
GL, scaling .95	default	39.48	.0310	86.59	**.00094**
GL, scaling .99	default	200.38	.00687	404.03	**.00020**
GL, scaling .90	matched	0.361	.0597	0.391	**.00076**
GL, scaling .95	matched	0.548	.0373	0.567	**.00083**
GL, scaling .99	matched	2.023	.0171	2.012	**.00036**

Table 8: GeomLoss continued at the target ε on clustered $n{=}m{=}2^{21}$, truncation 5, on the GPU of Table 7. K^* is the smallest number K of appended target-ε entries whose full residual meets $\tau{=}0.005$. Time is the median of three clean calls at K^*, which at $\varepsilon{=}0.001$ are the GL$^+$ calls of Table 1a. Bold marks solves that meet τ. Without a K^*, the last column gives the lowest scored residual. FS2 takes 15.44 s with $R{=}0.00483$ at $\varepsilon{=}0.001$ (Table 1a) and, in the run of this table, 0.817 s with $R{=}0.00267$ at $\varepsilon{=}0.01$.

		$K{=}0$	Continued at the target ε		
Problem	GL setting	R	K^*	Time (s)	R
$\varepsilon{=}0.001$	.90, default	.0542	–	–	.0203 at $K{=}512$
	.99, default	.00687	118	421.10	**.004997**
	.90, matched	.0597	–	–	.0160 at $K{=}2048$
$\varepsilon{=}0.01$	.90, default	.0731	–	–	.0372 at $K{=}512$
	.99, default	.0373	–	–	.0372 at $K{=}512$
	.90, matched	.0591	183	1.13	**.004976**

E.4 Residual Certificate and Sampled Check

Certificate. At each evaluated return we build the mask by the production rule at the returned pair, whatever the return kind, and compute $R_{\mathcal{A}}$ with the solver's sparse kernel from the same incoming pair in both directions. The solver as run stops on its sampled check, and the certificate is computed afterwards. We evaluated the clustered returns up to $n = 2^{22}$ at $\varepsilon = 0.001$ and up to 2^{21} at the other regularizations, the heterogeneous returns at $\varepsilon = 0.001$ up to 2^{21}, and both 2^{27} Quijote solutions, but not the other grid returns. All 15 are certified (Table 9). On the grid, Ω is at most 1% of τ and $R_{\mathcal{A}}$ agrees with the recorded full residual to within 1.2×10^{-6}, so the certificate exceeds the full evaluation by at most 1% of τ, at the cost of one mask build and one sparse update. The heterogeneous returns take the dense update, so their certificate recomputes the full residual rather than bounding it through the mask. At 2^{27}, Ω is below 10^{-7}, so FP32 arithmetic rather than the mask sets the certificate. At $\tau = 0.01$ it gives 0.009108, slightly below the recorded full residual of 0.009154. Both are FP32 estimates. The same admitted tiles evaluated through the tensor cores give 0.00859, and the FP64 correction of Section F.3 gives .00882 and .00869 for rows and columns, all below τ. The certificate takes minutes on one A100, where the full evaluation takes some 38 hours.

Sampled check. Of the 40 reported returns with a full residual, none exceeds τ, and the largest is 0.98τ. We re-solved the seven grid problems closest to τ, clustered $\varepsilon = 0.001$ at $n = 2^{19}$, 2^{20} and 2^{21}, clustered 2^{21} at $\varepsilon = 0.002$, and heterogeneous $\varepsilon = 0.001$ at 2^{19}, 2^{20} and 2^{21}, with the check's two index seeds set to $(s, 159015{+}s)$ for $s = 1, \ldots, 20$. With $s = 0$ each solve reproduces its record bitwise. All 140 solves returned confirmed pairs, and none has a full residual above τ, which bounds the rate of false acceptance at 2.6% (Clopper–Pearson, 95%), or at 6.0% over the 60 solves on the three problems whose stopping update moved with the seed. On those three problems the stop varied

Table 9: The certificate $R_{\mathcal{A}} + \Omega$ of Section 3.2 at returned potentials, with $R_{\mathcal{A}}$ the larger of the row and column residuals of $\mathbf{P}^{\mathcal{A}}$ in IEEE FP32 and Ω the tile-sum bound on the omitted mass in FP64. Where the mask would admit more than half of the tiles (dense), the update is dense, $\Omega = 0$ and $R_{\mathcal{A}}$ is the full residual. R is the recorded full residual, and bold certifies $R \le \tau$.

Problem	n	ε	τ	Mask	R	$R_{\mathcal{A}}$	Ω	$R_{\mathcal{A}} + \Omega$
Clustered	2^{19}	0.001	0.005	sparse	0.004879	0.004879	4.7×10^{-6}	**0.004884**
Clustered	2^{19}	0.01	0.005	sparse	0.002918	0.002918	2.7×10^{-5}	**0.002945**
Clustered	2^{20}	0.001	0.005	sparse	0.004791	0.004791	4.7×10^{-6}	**0.004796**
Clustered	2^{20}	0.01	0.005	sparse	0.003245	0.003246	2.5×10^{-5}	**0.003271**
Clustered	2^{21}	0.001	0.005	sparse	0.004828	0.004828	4.8×10^{-6}	**0.004833**
Clustered	2^{21}	0.002	0.005	sparse	0.004094	0.004094	7.5×10^{-6}	**0.004101**
Clustered	2^{21}	0.005	0.005	sparse	0.003198	0.003198	1.4×10^{-5}	**0.003211**
Clustered	2^{21}	0.01	0.005	sparse	0.002673	0.002674	2.3×10^{-5}	**0.002697**
Clustered	2^{21}	0.02	0.005	sparse	0.001066	0.001067	4.9×10^{-5}	**0.001115**
Clustered	2^{22}	0.001	0.005	sparse	0.004639	0.004639	4.8×10^{-6}	**0.004644**
Heterog.	2^{19}	0.001	0.005	dense	0.003605	0.003605	0	**0.003605**
Heterog.	2^{20}	0.001	0.005	dense	0.004342	0.004342	0	**0.004342**
Heterog.	2^{21}	0.001	0.005	dense	0.003875	0.003875	0	**0.003875**
Quijote	2^{27}	3.81	0.01	sparse	0.009154	0.009108	1.1×10^{-8}	**0.009108**
Quijote	2^{27}	3.81	0.05	sparse	0.048965	0.048953	5.4×10^{-8}	**0.048953**

over updates 70–74, 12–14 and 15–16 and the returned residual over 0.93–0.98τ. On the other four, every seed stopped at the same first passing check. At the accepting check, $(R - \text{mean})/\text{SE}$ over the 280 sides has mean 0.03 and standard deviation 1.01, so reusing one sample across checks leaves no detectable bias at the stop.

Concentrated error. On the accepted pair of heterogeneous 2^{20} at $\varepsilon = 0.01$ ($R = 0.19\tau$), we shift f on a random fraction q of the rows by $\varepsilon \log c$, with c set so that the row residual equals $\kappa\tau$, and run the check with 1000 index seeds per setting. Using the row side alone, the check accepts 95.8% of the seeds at $q = 0.01\%$ for $\kappa = 1.2$ and 2 (predicted 94.6% from the exact per-row values), 38.9% and 38.0% at $q = 0.1\%$ (predicted 36.9% and 36.1%), and 0.6% and 0.1% at $q = 1\%$ (predicted 0.8% and 0.04%). The production check tests both sides and accepted none of the 6000 perturbed states. Shifting f adds mass to the column sums near the perturbed rows, which raises the column residual to 1.15–1.96τ, so the column side rejects. The $(1-1/n)^S$ argument of Section D.2 describes these row-side rates and bounds them from below, since a sample containing a few perturbed rows can still pass when each carries a small excess. These measurements cover the seven problems and one perturbed state. They do not extend to $n \ge 2^{22}$, to other ε, to non-uniform weights or to the Quijote blur.

E.5 Ablations

The ablations use clustered and heterogeneous clouds with $n = m = 2^{20}$ at $\varepsilon \in \{0.001, 0.01\}$, and pair each control with an FS2 reference measured in the same run on the same GPU. Each of three controls removes one component. *No coarse stage* anneals on the points instead of the centroids, with no lift and no coarse continuation. *Dense fine* keeps the coarse stage and replaces every fine update by a dense streaming update. *Box only* keeps the box screen and sparse execution but skips tile verification.

Coarse stage. Removing the coarse stage slows the solve by 154–1643× on the three problems where both arms meet the target (Table 10). On clustered $\varepsilon = 0.001$ the control stops at the 128-update fine cap with $R = 0.02135$. A shorter coarse continuation has the same effect on a smaller scale (Table 11a): after 64 or 256 extra coarse updates the solve misses τ within the fine cap, and with the cap raised to 8192 it needs 1305 or 1115 fine updates, 51.6–60.0× the FS2 time. On three data seeds the 256-update control is 10.8–54.9× slower (Table 11b).

Sparse fine execution. On clustered $\varepsilon = 0.001$ both arms run 13 ordinary updates and two terminal previews, and sparse execution cuts the time from 33.327 to 4.165 s (8.00×). Heterogeneous $\varepsilon =$

Table 10: Component ablations. C and H denote clustered and heterogeneous inputs, followed by ε. The ratio (control/FS2) is given for active controls that meet the target.

Case	Control	FS2 (s)	Control (s)	Full R	Ratio	State
C, 0.001	No coarse stage	4.165	153.330	0.02135	–	misses target
	Dense fine	4.165	33.327	0.00479	8.001	qualified
	Box only	4.167	4.335	0.00479	1.040	qualified
C, 0.01	No coarse stage	0.275	183.132	0.00477	664.818	qualified
	Dense fine	0.276	0.275	0.00325	–	inactive
	Box only	0.276	0.275	0.00325	–	inactive
H, 0.001	No coarse stage	1.825	281.133	0.00039	154.023	qualified
	Dense fine	1.824	2.423	0.00434	1.328	qualified
	Box only	1.824	1.576	0.00434	0.864	qualified
H, 0.01	No coarse stage	0.089	145.599	0.00318	1643.418	qualified
	Dense fine	0.089	0.089	0.00097	–	inactive
	Box only	0.089	0.107	0.00097	–	inactive

Table 11: Coarse-continuation and threshold controls at $n = m = 2^{20}$, $\varepsilon = 0.001$. Fine updates count ordinary updates plus terminal previews. (a) Clustered, with the number of extra coarse updates fixed. Ratios divide by the FS2 reference of the same run. (b) The 256-update control with fine cap 8192 on three data seeds. (c) Every threshold control meets the target.

(a) Extra coarse updates, clustered

Extra coarse	Fine cap	Fine updates	Time (s)	R	Ratio
2048 (native)	128	13 + 2	4.169	0.00479	1
64	128	128 + 3	24.355	0.01896	–
256	128	128 + 3	24.817	0.01421	–
64	8192	1267 + 38	250.235	0.00478	60.0
256	8192	1077 + 38	215.674	0.00477	51.6

(b) 256 extra coarse updates, fine cap 8192, three data seeds

Seed	FS2 (s), updates	Control (s), updates	Ratio
0	4.181, 13 + 2	215.674, 1077 + 38	51.6
1	4.798, 16 + 2	263.227, 1308 + 47	54.9
2	7.185, 28 + 3	77.685, 349 + 19	10.8

(c) δ_{rel} scaled by a factor; time relative to the native threshold

Factor	Clustered	Heterogeneous
10^{-2}	1.080	1.039
10^{-1}	1.068	1.027
10^{1}	0.965	0.945
10^{2}	0.908	1.530

0.001 takes one terminal preview over a mask that admits 60.6% of the tiles and gains 1.33×. At $\varepsilon = 0.01$ neither geometry takes a fine update.

Tile verification. On clustered $\varepsilon = 0.001$, verification cuts the padded point-pair work from 3.049×10^{12} to 2.087×10^{12} and the time from 4.335 to 4.167 s, while on heterogeneous data it removes 3.95% of the work and costs time, 1.824 against 1.576 s. The extra screening pays when the reduction work it saves exceeds its own cost, and the two geometries fall on either side. Scaling δ_{rel} by 10^{-2} to 10^{2} changes the clustered time by at most 10%, and the largest factor slows the heterogeneous solve by 53% (Table 11c).

E.6 Profiling

The profiles use the clustered $n = m = 2^{21}$ problem of Table 1 on an A100 at $\varepsilon \in \{0.01, 0.001\}$: Nsight Systems for the phase timeline and NCU for kernel counters, each on a separate call. Unprofiled on this GPU, the $\varepsilon = 0.001$ solve takes 16.23 s with FS2 and 21.49 s with GeomLoss,

against 15.44 and 21.45 s on the separate A100 of Table 1a. That table's stage times come from one Nsight Systems call per solver on that card. FS2's coarse stage is its coarse kernels (2.663 s) and lifts (0.277 s), and its fine stage its fine kernels (8.274 s), mask construction (3.118 s) and sampled checks (0.909 s). GeomLoss's coarse stage is its centroid updates and lift.

Phase timeline. Nsight Systems gives, per phase, the union of kernel intervals. At $\varepsilon = 0.001$, GeomLoss spends 2.029 ms in its coarse phase and 21.400 s in its fine phase, and FS2 spends 2.674 s on coarse kernels, 0.283 s on lifts, 9.064 s on fine kernels, 3.117 s on mask construction and 0.923 s on sampled checks. GeomLoss's fine phase is 96 kernels averaging 0.22 s, so kernel launches do not explain its time. At $\varepsilon = 0.01$ neither solver runs a fine phase.

Table 12: Kernel counters for one fine update per direction on clustered $n = m = 2^{21}$, $\varepsilon = 0.001$, on the profiling A100. FS replays one dense reduction per direction from its native incoming state, with its counters averaged over the two directions. GL and FS2 run FS2's first ordinary fine update on the same $B = 128$ mask, and f/g gives the two directions. Per-pair counts divide totals by the point pairs they cover, instructions are executed warp instructions, and dashes mark counters not collected.

Metric	FS (dense)	GL (KeOps, FP32)	FS2 (TF32)
Point pairs per direction	4.40×10^{12}	2.58×10^{11}	2.58×10^{11}
NCU time per direction (ms)	3,600	393.957	251.435
Warm update, both directions (ms)	–	791.073	504.392
Warp instructions / pair	0.34	0.628315	0.278530
Shared-load wavefronts / pair	–	0.0625001	0.0246587
Shared-store wavefronts / pair	–	0.0009806	0.0053709
HBM read bytes / pair	0.0043542	0.0003161	0.0149489
HBM write bytes / pair	0.0000025	0.0001135	0.0000558
Tensor-pipe instructions / pair	–	0.0003344	0.015625
Tensor activity, f/g (%)	–	0.07 / 0.07	21.11 / 21.08
Occupancy, f/g (%)	37.08	91.4 / 91.14	12.48 / 12.47
Math-pipe throttle, f/g (%)	8.995	17.39 / 17.41	2.32 / 2.31
Long scoreboard, f/g (%)	1.21	0.51 / 0.51	6.05 / 6.03

Kernel counters. Table 12 compares the GL and FS2 executors on the same mask. We freeze FS2's first ordinary fine update on the clustered $n = m = 2^{21}$, $\varepsilon = 0.001$ problem, with a $B = 128$ mask of 15,760,133 admitted block pairs and 5.16×10^{11} point pairs over both directions. FS2's TF32 kernels and GeomLoss's FP32 KeOps executor receive the same sorted points, incoming potentials, weights and transposed support, KeOps with each run of contiguous target blocks as one range. Over 20 warmed repetitions the update takes 504.392 ms with FS2 and 791.073 ms with KeOps, $1.568\times$ longer. Per pair, FS2 executes 55.7% fewer warp instructions and 60.5% fewer shared-load wavefronts, with tensor-pipe activity of 21% against 0.07%, despite more HBM reads and shared-store wavefronts, lower occupancy and more long-scoreboard stall. The extra HBM traffic follows from the policy's block size $B = 128$ (Section D.1), which trades memory traffic for time. The dense FlashSinkhorn reduction evaluates all 4.40×10^{12} pairs of one direction in 3.6 s, $14\times$ longer than FS2's reduction on its mask.

E.7 Memory

Figure 5 reports two memory measures for the twenty clustered solves of Figure 2(e): the process peak sampled by NVML, which includes allocator pools and external CUDA allocations but can miss short-lived ones, and the PyTorch allocator peak, which excludes external allocations.

The same protocol on the five heterogeneous FS2 solves at $\varepsilon = 0.001$, on the same GPU, gives PyTorch peaks of 0.80, 2.72, 9.89, 1.70 and 3.40 GiB for $n = 2^{19}$ to 2^{23}. Up to 2^{21}, FS2 takes one or two fine updates over 60–61% of the tiles. From 2^{22} it returns the lifted coarse potentials without a fine update and builds no mask.

Block size. We hold the clustered $n = m = 2^{23}$, $\varepsilon = 0.001$ problem and its coarse partition fixed ($t = 42$, with 89,140 and 89,657 centroids) and change only the block size. Raising B from 128 to 256 cuts the admitted tiles from 2.25×10^8 to 6.0×10^7. It lowers the PyTorch peak from 16.17

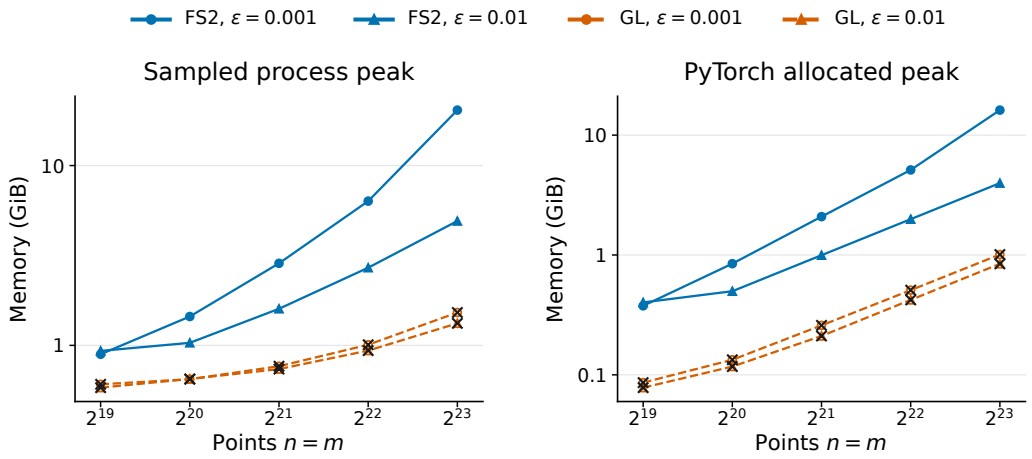


Figure 5: Peak memory of the twenty clustered solves: sampled process peak (left) and PyTorch allocator peak (right), inputs included. Crosses mark GeomLoss returns with $R > 0.005$.

to 6.88 GiB and the NVML-sampled process peak from 20.34 to 8.05 GiB, for 7% more time, 48.52 against 45.38 s. At $B = 256$ the kernel evaluates each block pair in 64×64 point tiles. Both arms take one terminal preview after 2048 extra coarse updates and return $R = 0.004694$. The automatic policy chooses B from the L2 budget of the block summaries, not from this mask memory.

E.8 Optimization

The optimization grid reuses the 32 forward inputs of Section E.3. Each method runs one trajectory of twenty normalized updates (Section 4.1) and saves steps 0, 5, 10, 15 and 20. Here the timer includes compilation and autotuning, and a run stops at 7200 s. FS2 steps along the surrogate gradient of Section 3.1, with the target-cell moment in TF32 and the norm in FP64, while GeomLoss differentiates its native multiscale loss with autograd. Table 13 summarizes the $n = m = 2^{21}$ runs of Figure 2(g,h). The panels plot the five saved checkpoints of these four runs as the percentage change of U from its initial value, on a time axis that is linear to one second and logarithmic beyond. GeomLoss reaches the time limit on six problems: clustered 2^{23} at $\varepsilon = 0.001$, and heterogeneous 2^{22} at $\varepsilon \in \{0.001, 0.002, 0.005\}$ and 2^{23} at $\varepsilon \in \{0.001, 0.01\}$.

Score. The evaluator fixes K nested groups of the initial source cloud and of $\mathbf{Y}$, with FP64 centroids and within-group variances. At a saved iterate $\mathbf{X}$, $U_K(\mathbf{X})$ is the entropic cost between the source and target group centroids, solved to row and column residuals of at most 10^{-4}, plus the within-group squared-distance averages $V_x(\mathbf{X})$ and $V_y(\mathbf{Y})$. We report $K = 131072$ at $\varepsilon \le 0.002$, 65536 at $\varepsilon = 0.005$ and 16384 at $\varepsilon \ge 0.01$. Every decrease must agree with the one at a coarser count (65536, 16384 and 4096) within $\max(10^{-6}, 0.02|\Delta U_K|)$, and final scores within 3×10^{-6} of each other count as ties. The score U_K ranks trajectories and is not the full-point OT value. On heterogeneous problems the decreases along the two directions agree to 0.02–0.15%. At about 2×10^{-5} RMS displacement per point and step, the twenty steps measure the direction at $\mathbf{X}_0$ rather than a long flow.

Direction at $\mathbf{X}_0$. At the initial points of the four $n = m = 2^{21}$ runs of Figure 2(g,h), we compare $\widetilde{G}$ and GeomLoss's autograd direction with $G_i = 2a_i(x_i - T_\varepsilon(x_i))$ built from two numerical references: the forward grid's returned pair ($\tau = 0.005$) and a separate solve at $\tau = 5 \times 10^{-4}$. Against the second reference, $\widetilde{G}$ has cosine 0.92 and 0.96 with G on clustered data ($\varepsilon = 0.001$ and 0.01) and at least 0.9997 on heterogeneous data. Its relative error is 0.39 and 0.28 on clustered data and, from 4096 rows evaluated in FP64, 0.023 and 0.012 on heterogeneous data. Against the first reference these values change by at most 0.007. They measure alignment with numerical references, not the error to the exact gradient, because the two references differ by 0.2–3%, and on clustered data at $\varepsilon = 0.001$ the tighter solve stopped at residual 1.2×10^{-3}, above its target. GeomLoss's direction is

farther from both references on clustered data, with relative error 0.49 and 0.40. On heterogeneous data the difference between the two directions lies within the uncertainty of the references. On the same 4096 rows, the right-hand side of Equation (8) exceeds $\|G\|_F$ on three of the four problems, so Proposition 3 does not certify a relative error below one there. The IEEE-FP32 kernel that evaluates G loses 2% on heterogeneous data at $\varepsilon = 0.001$, because each row's plan spreads over some 10^5 targets and accumulating $\sum_j p_{ij} y_j$ in FP32 loses about 6×10^{-4} against a displacement $\|x_i - T_\varepsilon(x_i)\|$ of about 0.03.

Table 13: Twenty-step optimization at $n = m = 2^{21}$. A decrease is the percentage drop of U from the common initial score U_0 over the twenty steps, and final scores within 3×10^{-6} tie. Times cover the twenty updates, including compilation and autotuning.

Cloud	ε	U_0	FS2 s	GL s	FS2 decrease	GL decrease
Heterogeneous	.001	.004297	52.80	3215.67	.5829%	.5826%
Heterogeneous	.01	.01501	51.66	1290.01	.2551%	.2550%
Clustered	.001	.05230	79.81	453.22	.3064%	.2906%
Clustered	.01	.08166	42.58	0.80	.1756%	.1682%

F Quijote Particle Transport

F.1 Problem and Data

The sources are redshift-zero positions matched by particle ID to their Lagrangian lattice sites, the targets. The 2^{24} instance is the central 256^3 subcube and the 2^{27} instance the full 512^3 box, with $\varepsilon = 3.814697265625$, so that the blur $\sqrt{\varepsilon} = 1.953125\,\mathrm{Mpc}/h$ equals the lattice spacing, and a nonperiodic cost. FS2 solves on TF32-rounded coordinates, which move each particle by at most $0.125\,\mathrm{Mpc}/h$, 6% of the blur. In the 2^{24} subcube the particle-ID displacement $\|x_i - q_i\|$ has median 9.013, 99th percentile 20.668 and maximum $39.135\,\mathrm{Mpc}/h$.

F.2 Configuration and Timing

Table 14: Quijote forward solves, with the $\tau = 0.05$ size sweep in the first four rows. $B/t/c/h$: block size, cell level, fine cap and returned ordinary updates. Peak: PyTorch allocator peak of the size sweep. Light validation: mean absolute relative marginal error on 4096 sampled indices per direction, 65,536 at 2^{25} and 2^{26}, with the mean plus three standard errors in brackets. Full max: R, with dashes where it is not enumerated. W: warm call, F: first call, I: instrumented call. MS/.95/default/5: multiscale, scaling 0.95, default cluster scale, truncation 5.

method	$n = m$	τ	configuration	time (s)	peak (GiB)	light validation $[U_{3\mathrm{SE}}]$	full max
FS2	2^{24}	.05	$B64/t47/c128/h80$	146.905^{W}	12.053	.046836 [.048531]	.046631 ✓
FS2	2^{25}	.05	$B64/t45/c128/h48$	374.839^{W}	19.472	.049304 [.049758]	—
FS2	2^{26}	.05	$B128/t45/c128/h72$	694.630^{W}	33.131	.048340 [.048783]	—
FS2	2^{27}	.05	$B256/t45/c128/h80$	1351.589^{W}	64.129	.049009 [.050797]	.048965 ✓
FS2	2^{24}	.01	$B64/t47/c800/h432$	736.393^{W}	—	.009538 [.009924]	.009433 ✓
FS2	2^{27}	.01	$B256/t45/c800/h560$	8832.994^{F}	—	.009585 [.009954]	.009154 ✓
native GL	2^{24}	.05	MS/.95/default/5	3713.300^{I}	—	.058976 [.060909]	.058463 ✗

Configuration.

- 2^{24}: $t = 47$, the selector's $B = 64$ and a sampled check every 16 fine updates. The fine cap is 128 at $\tau = 0.05$ and 800 at $\tau = 0.01$.
- 2^{27}: the selector's $B = 256$, processed as 64×64 point tiles, with $t = 45$ set explicitly. The fine caps are as at 2^{24}, and the check interval is chosen automatically.
- Coarse stage: at every size the solver completes the 64 entries at ε before its first lift. In place of the continuation of Appendix D.2, it runs at most one batch of 64 to 384 coarse

updates after the second fine update and lifts again. In every run the sampled check scored that lift worse than the fine iterate, so the fine updates continue from the first lift. The time of the batch is included.

- GeomLoss, at 2^{24} and $\tau = 0.05$ only: scaling 0.95, default cluster scale and truncation 5.

We set t by hand at both sizes. At 2^{27} the rule would select $t = 46$ if its fourth condition admitted that size. With $t = 46$ the sampled check became non-finite in the first fine updates, so we use $t = 45$, whose coarse state exceeds the L2 budget of Equation (17). The cap is raised to 800 at $\tau = 0.01$ because the two sizes return after 432 and 560 ordinary updates there, and fixed checks every 16 fine updates returned sooner at 2^{24} than fixed checks every 32. With the continuation of Appendix D.2 in place of the single batch, the 2^{27}, $\tau = 0.05$ solve took 2030.4 s and 119 ordinary updates, against 1351.6 s and 80 in Table 14. Both are single warm calls that passed the sampled check.

Mask size. At 2^{27} and $\tau = 0.01$, in the update that produced the returned potentials, the box screen passed 107,397,067 of the 2^{38} block pairs and verification admitted 24,451,822, which hold 1.60×10^{12} point pairs. A truncated kernel with one FP32 value and one 32-bit index per pair would take 11.7 TiB. FS2's index for the candidate and admitted tiles, at 24 bytes per tile, takes 2.9 GiB.

Timed calls. Every time in Table 14 is a single call.

- FS2 at 2^{24}: warm calls. A second call returns identical potentials (147.858 s at $\tau = 0.05$, 749.711 s at $\tau = 0.01$), so its all-particle validation applies.
- FS2 at 2^{27}: a warm call at $\tau = 0.05$, and at $\tau = 0.01$ the first complete call, because its warm repeat was stopped.
- GeomLoss: the instrumented call whose potentials are scored. An earlier uninstrumented call (3706.982 s) returned different potentials.

Validation is not timed. At 2^{27} the IEEE-FP32 evaluator of the full residual processes 1024 chunks of 2^{18} rows or columns at about 135 s each on an A100, some 38 hours for the $\tau = 0.01$ solution. The light validation of Table 14 uses seed 1, and each sampled marginal is reduced against every point of the opposite measure.

F.3 Evaluator Precision

The full residuals come from an IEEE-FP32 evaluator that forms each row's log marginal ratio as the difference of two potentials shifted by $\|x_i\|^2$. Near the box corners $\|x_i\|^2$ reaches 7.5×10^5, whose FP32 spacing is 1.6% of ε, so a single row's mass is resolved only to 0.1–1.6%, and the full residual is an aggregate over every particle rather than a per-particle bound. On 4096 or 16384 sampled rows and columns per entry, we recompute each row's error $|r_i/a_i - 1|$ in FP64 from coordinate differences and subtract the mean paired difference between the FP32 and FP64 errors from the full FP32 residual. This gives .04653/.04661, .04883/.04889, .00939/.00942 and .00882/.00869 (row/column) for the rows of Table 14 with a full residual, in table order. Three standard errors of the paired mean are at most .0002, so every entry stays below its τ. The certificate of Section E.4 bounds both 2^{27} entries below their τ as well.

F.4 Size Sweep

Figure 3(a,b) and the first four rows of Table 14 fix $\tau = 0.05$ and use FS2 only.

- Domains: nested cuboids centered in the box, from $256 \times 256 \times 256$ to $512 \times 512 \times 512$ lattice sites (2^{24} to 2^{27} points per measure), with the lattice spacing and ε fixed.
- Endpoints: the 2^{24} and 2^{27} runs above. At 2^{24} the time and the memory come from two calls with identical potentials.
- Middle sizes: single warm calls with the 2^{27} configuration ($t = 45$, fine cap 128) and the selector's B. Both runs pass the solver's sampled check.
- Validation: at the middle sizes, 65,536 sampled outputs per side. The full residual is not enumerated.

F.5 Displacement Reconstruction

We take the 2^{27}-particle, $\tau = 0.01$ solution of Table 14 and a uniform sample of 4096 source indices, drawn without replacement with seed 20260919. Each projection sums over all 2^{27} targets without a mask, using the plan of the TF32-rounded input and the unrounded physical target positions. With $L = 1000\,\mathrm{Mpc}/h$ and the componentwise minimum image $w_L(v) = v - L\lfloor v/L + 1/2 \rfloor$, we evaluate

$$T_i = \frac{\sum_j P_{ij} q_j^{\mathrm{phys}}}{\sum_j P_{ij}}, \qquad \hat{\psi}_i = w_L(x_i^{\mathrm{phys}} - T_i), \qquad \psi_i^* = w_L(x_i^{\mathrm{phys}} - q_i^{\mathrm{phys}}).$$

The wrapping enters only the scoring, and the solver's cost stays nonperiodic. Over the sample $\mathcal{S}$, displacement RMSE is $[|\mathcal{S}|^{-1} \sum_{i \in \mathcal{S}} \|\hat{\psi}_i - \psi_i^*\|^2]^{1/2}$, and normalized RMSE divides it by the RMS reference displacement (Table 15). The component correlations are 0.797, 0.789 and 0.802, and the unwrapped position RMSE is 122.268 Mpc/h.

Table 15: Displacement reconstruction from the 2^{27}-particle FS2 solution at $\tau = 0.01$ on 4096 sampled particles, with minimum-image scoring. The interval comes from 500 bootstrap resamples over particles at 95% confidence. The subsets split the sample by whether the reference trajectory crosses a periodic face.

Predictor	Particles	RMSE (Mpc/h)	Normalized RMSE	95% interval
Zero displacement	4096	10.064	1.000	–
FS2 barycentric displacement	4096	6.473	0.643	[0.625, 0.662]
no periodic-face crossing	4034	6.333	–	–
periodic-face crossing	62	12.603	–	–

Cost and checks.

- The 4096-query projection takes 9.303 s on one A100, after 43.474 s of loading and preparation.
- FP64 direct-distance projections of eight sampled queries agree with the FP32 stream within 0.002281 Mpc/h.
- Joining the raw source and target ID arrays reproduces the sampled reference displacements to $3.1 \times 10^{-5}\,\mathrm{Mpc}/h$.