COREFLOW: LOW-RANK MATRIX GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning matrix-valued distributions from high-dimensional and possibly incomplete training data is challenging: ambient-space generative modeling is computationally expensive and statistically fragile when the matrix dimension is large but the sample size is limited. We propose *CoreFlow*, a geometry-preserving low-rank flow model that learns shared row/column subspaces across the matrix distribution, and then trains a continuous normalizing flow only on the induced low-dimensional core. CoreFlow is designed for settings where shared low-rank matrix geometry is present, especially in high-dimensional limited-sample regimes. This separates shared matrix geometry from sample-specific variation, preserves matrix structure, and substantially improves training efficiency. The same framework also handles incomplete training matrices through masked Riemannian updates and iterative completion. Across benchmarks spanning astrophysical imaging, atmospheric reanalysis, ocean climate, and various simulations, CoreFlow substantially improves spectral and moment-level generation quality in few-sample regimes while remaining competitive in data-rich settings, even under compression to 9% of the ambient dimension and with up to 40% missing training entries.

1 INTRODUCTION

Many modern scientific and engineered datasets are naturally *matrix-valued*: spatiotemporal fields in geophysics and climate reanalysis (Harries & O'Kane, 2020; Store et al., 2024), sensor grids (Ran et al., 2016; Wang et al., 2022), and medical imaging (Graff & Sidky, 2015; Suetens, 2017) are often represented as $M \in \mathbb{R}^{m_1 \times m_2}$ arrays whose two modes encode different physical axes (e.g., space$\times$time, sensor$\times$frequency). A core modeling goal is to learn a *generative distribution* over such matrices, enabling realistic synthesis, uncertainty-aware simulation, and downstream tasks such as augmentation, denoising, and imputation.

Diffusion models (Ho et al., 2020; Song et al., 2020) and flow-based generative models (Chen et al., 2018; Lipman et al., 2023) have become dominant for high-fidelity synthesis. However, for matrix-valued data, the ambient dimension can be tens of thousands or more, while the number of training samples may be limited. Therefore, training a diffusion or flow model in the original dimension can be not only *computationally expensive*, but also statistically *significantly underdetermined*: the model must estimate a complex high-dimensional distribution from too few examples.

Many matrix distributions concentrate near approximately low-rank sets: dependencies across the two modes induce shared row and column subspaces, while sample variability is expressed in a much smaller core space (Gupta & Nagar, 2018; Mak et al., 2017; Yuchi et al., 2023). Therefore, the intrinsic dimension can be far smaller than $m_1 m_2$. This motivates learning the shared low-rank geometry and fitting the generator only in the induced low-dimensional coordinates. Concretely, we may model each matrix as $M_i \approx U S_i V^\top$, where U and V are shared row/column subspaces, and S_i is a sample-specific low-dimensional core. Accordingly, we position our method for settings where shared low-rank matrix geometry is present.

Real-world matrix datasets may also be incomplete due to sensor dropouts, irregular sampling, or acquisition constraints. This makes shared-geometry recovery more challenging, and a practical low-rank generative method should remain stable even when training matrices are partially observed.

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.

To address these challenges, we propose *CoreFlow*, a two-stage geometry-preserving generator for matrix distributions with shared low-rank geometry. CoreFlow first learns shared row/column subspaces, and then learns a continuous normalizing flow (CNF) only on the induced low-dimensional core distribution. Our main *contributions* are: (i) *Geometry-preserving low-rank matrix generation.* CoreFlow learns a shared low-rank coordinate system for the matrix distribution and trains a CNF only in the induced core space. This substantially reduces the effective generative dimension while preserving matrix structure, yielding a strong quality–efficiency tradeoff, especially in high-dimensional few-sample settings. (ii) *Robust structure learning from incomplete matrices.* We develop a masked (U, V) subspace estimator with alternating completion, yielding stable subspace recovery and reliable core representations under substantial missingness.

1.1 Related Works

Ambient- and latent-space generative modeling. Diffusion (Ho et al., 2020; Song et al., 2020) and flow-based models (Chen et al., 2018; Lipman et al., 2023) support high-fidelity synthesis, statistical sampling (Tian et al., 2024; Wu & Xie, 2025; Wu et al., 2025), time-series generation (Lin et al., 2024; Wu et al., 2026), and image generation (Peebles & Xie, 2023; Li & He, 2025), but ambient-space training is expensive and statistically fragile in high dimensions with limited samples. Compressed or latent approaches improve efficiency (Rombach et al., 2022; Peebles & Xie, 2023; Zheng et al., 2025), but typically rely on a pre-trained VAE. CoreFlow instead learns a geometry-preserving representation directly from limited and possibly incomplete matrix observations. Concurrent Tucker Diffusion (Guo et al., 2026) embeds Tucker structure in an end-to-end tensor-space score model, whereas CoreFlow learns shared subspaces from complete or incomplete matrices and models only the R^2-dimensional cores with a CNF. No public implementation was available for comparison at submission. Closest to our setting, Guo et al. (2025) studies low-dimensional diffusion through compressed sensing, but does not exploit matrix geometry and may distort matrix structure.

Generative modeling with incomplete observations. MissDiff (Ouyang et al., 2023) masks the diffusion loss on observed entries, whereas MCFlow and DiffPuter (Richardson et al., 2020; Tashiro et al., 2021; Zheng & Charoenphakdee, 2022; Zhang et al., 2024) use diffusion or normalizing flows for iterative imputation. Unlike CoreFlow, these methods primarily target imputation and are generally trained on the full ambient distribution.

Generative imputation and matrix completion. Matrix completion exploits low-dimensional structure to recover missing entries of a partially observed matrix. Approaches include nuclear-norm and scalable optimization methods (Candès & Tao, 2010; Candes & Recht, 2012; Koltchinskii et al., 2011; Davenport & Romberg, 2016; Chi et al., 2019; Gu et al., 2024), and probabilistic factor models with uncertainty quantification (Alquier et al., 2014; Mak et al., 2017; Yuchi et al., 2023; Choi et al., 2024). The Singular Matrix-variate Gaussian family (Gupta & Nagar, 2018; Mak & Xie, 2018; Mak et al., 2017) uses a Gaussian core. These methods condition on a given matrix's observed entries and aim to recover that matrix. CoreFlow addresses a different task: it learns an unconditional distribution over complete matrices from possibly incomplete training samples, to generate entirely new matrices.

Factorized generators in other domains. Graph generators model discrete adjacency structures with graph-specific constraints, architectures, and metrics (You et al., 2018; Bojchevski et al., 2018; De Cao & Kipf, 2018; Simonovsky & Komodakis, 2018; Fan & Huang, 2019; Shiao & Papalexakis, 2021). BRGAN also uses low-rank factors, but to generate rank-controlled graphs evaluated by degree and clustering, a fundamentally different task from modeling continuous matrices. GenTen reduces the cost of 3D calorimeter simulation by factorizing each output tensor separately, rather than learning a common representation of the data distribution (Goulart et al., 2025). CoreFlow instead models a general *continuous* matrix distribution under a shared low-rank assumption.

2 Preliminaries

2.1 Neural ODE and continuous normalizing flow

A continuous normalizing flow (CNF) defines an invertible transformation through the Neural ODE (Chen et al., 2018)

$$\frac{dx(t)}{dt} = v_\theta(x(t), t), \qquad t \in [0, 1], \tag{1}$$

where v_θ is a time-dependent velocity field. We use the data-to-noise convention: forward integration transports the data distribution at $t = 0$ to a simple base distribution $q = \mathcal{N}(0, I)$ at $t = 1$. Generation reverses this process by drawing $x_1 \sim q$ and integrating the ODE backward to $t = 0$.

We train the velocity field by conditional flow matching (CFM) (Lipman et al., 2023). Given a data sample x_0, an independent base sample $x_1 \sim q$, and $t \sim \mathrm{Unif}(0, 1)$, CFM uses the linear path

$$x_t = \phi(x_0, x_1; t) = (1 - t)x_0 + tx_1, \qquad \frac{d\phi}{dt} = x_1 - x_0.$$

The model regresses $v_\theta(x_t, t)$ toward the tractable conditional velocity $x_1 - x_0$. Although this target is defined for individual pairs (x_0, x_1), its population regression recovers the marginal velocity that drives the corresponding distributional path. Section 3.3 applies this construction to CoreFlow by taking x_0 to be a learned core vector and x_1 to be Gaussian noise.

2.2 Optimization on the Stiefel manifold

Low-rank matrix methods often represent row and column subspaces by matrices with orthonormal columns. Since standard Euclidean updates do not preserve this constraint, we optimize on the Stiefel manifold (Absil et al., 2008; Edelman et al., 1998), $\mathrm{St}(m, R) = \{W \in \mathbb{R}^{m\times R} : W^\top W = I_R\}$. Given a Euclidean gradient G at $W \in \mathrm{St}(m, R)$, we project G onto the tangent space at W, take a first-order step, and retract back to the manifold. We use QR retraction, which is simple and numerically stable. Algorithm 2 summarizes this procedure as $\textsc{StiefelStep}(W, G, \eta)$.

3 Algorithm

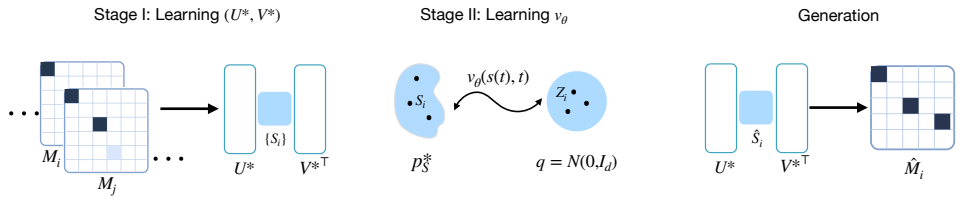


Figure 1: Overview of CoreFlow. *Left:* learning shared row/column subspaces (U^*, V^*) and constructing induced low-dimensional cores $\{S_i\}$. *Middle:* learning a continuous normalizing flow in the induced core space, matching the Stage-I cores $\{S_i\} \sim p_S^*$ to a Gaussian base. *Right:* generation starts from Gaussian noise, reverses the learned flow to obtain a generated core $\hat{S}_i$, and decodes it with (U^*, V^*) to produce a new matrix $\hat{M}_i$.

3.1 Settings and Goal

Let $M_i \in \mathbb{R}^{m_1\times m_2}$, $i = 1, \ldots, N$, be i.i.d. samples from an unknown distribution p_M^* of complete matrices. The observed training matrix may contain only the entries indexed by $\Omega_i \subseteq [m_1] \times [m_2]$:

$$M_i^{\mathrm{obs}} = \mathcal{P}_{\Omega_i}(M_i), \qquad (\mathcal{P}_\Omega(M))_{jk} = \begin{cases} M_{jk}, & (j, k) \in \Omega, \\ 0, & (j, k) \notin \Omega, \end{cases} \tag{2}$$

where $[m] = \{1, \ldots, m\}$; fully observed data correspond to $\Omega_i = [m_1] \times [m_2]$. We assume that the dominant variation across samples lies in shared rank-R row and column subspaces:

$$M_i \approx US_iV^\top, \qquad U \in \mathrm{St}(m_1, R), \quad V \in \mathrm{St}(m_2, R), \quad S_i \in \mathbb{R}^{R\times R}. \tag{3}$$

Here $R < \min(m_1, m_2)$, U and V capture row and column subspaces shared across the matrix distribution, and S_i is a sample-specific core. Let $s_i = \mathrm{vec}(S_i) \in \mathbb{R}^d$, where $d = R^2$, let $\mathrm{mat}(\cdot)$ be the inverse of $\mathrm{vec}(\cdot)$, and denote the distribution of s_i by p_s^*.

Goal. Given $\{(M_i^{\mathrm{obs}}, \Omega_i)\}_{i=1}^N$, the goal is to learn an unconditional distribution $\hat{p}_M$ that approximates p_M^* and generates complete matrices. By separating shared matrix geometry from sample-specific variation, CoreFlow reduces the generative dimension from m_1m_2 to $d = R^2$, significantly lowering computational cost and improving learning with limited data. Stage I estimates (U, V) and the sample-specific cores $\{s_i\}$ from the training observations, while Stage II fits a continuous normalizing flow (CNF) to $\{s_i\}$. To generate a matrix, the learned flow draws $\hat{s} \sim \hat{p}_s$, where $\hat{p}_s$ denotes the generated core distribution, and CoreFlow decodes it as $\hat{M} = U\,\mathrm{mat}(\hat{s})\,V^\top$, whose distribution is $\hat{p}_M$.

3.2 STAGE I: LEARNING SHARED MATRIX GEOMETRY ON THE STIEFEL MANIFOLDS

Complete matrices. In Stage I, we learn orthonormal factors $U \in \mathrm{St}(m_1, R)$ and $V \in \mathrm{St}(m_2, R)$ such that every sample of p_M^* can be well-approximated by the rank-R model $M_i = US_iV^\top$. Given (U, V), the core matrix can be re-written as $S_i := U^\top M_i V \in \mathbb{R}^{R\times R}$, so the reconstruction becomes $U\,S_i\,V^\top = UU^\top M_i VV^\top$. Therefore, we can estimate (U, V) by minimizing the reconstruction error:

$$\min_{U\in\mathrm{St}(m_1,R),\, V\in\mathrm{St}(m_2,R)} \mathcal{L}_{\mathrm{rec}}(U, V) := \frac{1}{N}\sum_{i=1}^{N}\left\|M_i - UU^\top M_i VV^\top\right\|_F^2. \tag{4}$$

For complete matrices, we start with a deterministic spectral (Tucker initialization) estimate of (U, V). It recovers the shared subspaces in the exact rank-R case. Otherwise, Stiefel updates are needed to refine the subspaces. These updates become essential in the incomplete setting below, where the masked loss has no one-shot SVD solution and must be optimized directly over the observed entries.

Incomplete matrices. When entries are missing, directly evaluating $\|M_i - US_iV^\top\|_F^2$ is impossible. Therefore, we introduce a latent *filled-in* matrix $\widetilde{M}_i$ for each training sample and optimize a masked objective on observed entries:

$$\mathcal{L}_{\mathrm{miss}}(U, V; \{\widetilde{M}_i\}) := \frac{1}{N}\sum_{i=1}^{N}\left\|\mathcal{P}_{\Omega_i}\left(M_i^{\mathrm{obs}} - UU^\top \widetilde{M}_i VV^\top\right)\right\|_F^2. \tag{5}$$

We then alternate: (i) update (U, V) using gradients of equation 5 (holding $\widetilde{M}_i$ fixed), and (ii) update $\widetilde{M}_i$ by keeping observed entries fixed and imputing the missing ones with the current low-rank reconstruction:

$$\widetilde{M}_i \leftarrow \mathcal{P}_{\Omega_i}(M_i^{\mathrm{obs}}) + \mathcal{P}_{\Omega_i^c}\left(UU^\top \widetilde{M}_i VV^\top\right).$$

Intuitively, Step (ii) produces a completed matrix consistent with observations while staying on the rank-R manifold induced by (U, V); Step (i) then refines the shared subspaces using only reliable observed entries. The full procedure is summarized in Alg. 3 of Appendix.

3.3 STAGE II: GEOMETRY-PRESERVING CORE-SPACE FLOW MATCHING

We denote by (U^*, V^*) the shared subspaces learned in Stage I. After Stage I, (U^*, V^*) are fixed and each (completed) training matrix is mapped to a low-dimensional core:

$$S := U^{*\top} M V^* \in \mathbb{R}^{R\times R}, \qquad s := \mathrm{vec}(S) \in \mathbb{R}^d.$$

Correspondingly, we denote by p_S^* the distribution of the induced core matrix S after learning the subspaces (U^*, V^*), and by $p_s^* := \mathrm{vec}_\# p_S^*$ the induced distribution of the core vector $s = \mathrm{vec}(S)$.

A continuous normalizing flow (CNF) defines a transformation between a data distribution and a base distribution via a Neural ODE indexed by time $t \in [0, 1]$. It transports samples from the data distribution at t=0 (i.e., $s(0) \sim p_s^*$) to the base distribution at t=1 (typically $\mathcal{N}(0, I_d)$) according to

$$\frac{ds(t)}{dt} = v_\theta(s(t), t), \quad t \in [0, 1], \tag{6}$$

where the trajectory is initialized at a data sample $s(0) = s$, and ends at $s(1) = z$ with $z \sim \mathcal{N}(0, I_d)$.

Flow matching trains a time-dependent velocity field $v_\theta(\cdot, t)$ on $\mathbb{R}^d$ by matching an analytic reference velocity along a simple coupling between Gaussian noise and data cores, as introduced in Section 2.1. Specifically, we sample $s = s(0) \sim p_s^*$, $z = s(1) \sim \mathcal{N}(0, I_d)$, $t \sim \mathrm{Unif}(0, 1)$, and then define the straight-line reference path and its derivative:

$$\phi(s, z; t) := (1 - t)s + tz, \qquad \frac{d\phi}{dt} = z - s.$$

The flow-matching objective regresses the model velocity to this target:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\; s\sim p_s^*,\; z\sim\mathcal{N}(0,I_d)} \left\|v_\theta(\phi(s, z; t),\, t) - (z - s)\right\|_2^2.$$

Algorithm 1: Two-Stage Training for COREFLOW

Require: Training set $\{M_i\}_{i=1}^N \subset \mathbb{R}^{m_1 \times m_2}$ (fully observed); rank R; iterative steps $T_{\text{sub}}, T_{\text{flow}}$; learning rates $\eta_U, \eta_V, \eta_\theta$.
Ensure: $U^* \in \mathbb{R}^{m_1 \times R}, V^* \in \mathbb{R}^{m_2 \times R}$ (orthonormal columns), flow parameters θ^*.
Initialize $U \in \mathrm{St}(m_1, R)$ and $V \in \mathrm{St}(m_2, R)$ via spectral initialization.
Stage I: learn shared subspaces (U^*, V^*)
for $t = 1$ to T_{sub} **do**
  Sample mini-batch $\mathcal{B} \subset \{1, \dots, N\}$.
  $\mathcal{L}_{\text{rec}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \|M_i - U U^\top M_i V V^\top\|_F^2$.
  Compute $G_U \leftarrow \nabla_U \mathcal{L}_{\text{rec}}, G_V \leftarrow \nabla_V \mathcal{L}_{\text{rec}}$.
  $U \leftarrow \text{STIEFELSTEP}(U, G_U, \eta_U)$; $V \leftarrow \text{STIEFELSTEP}(V, G_V, \eta_V)$.
end for
$U^* \leftarrow U$, $V^* \leftarrow V$.
Stage II: train CNF on core vectors under fixed (U^*, V^*)
for $t = 1$ to T_{flow} **do**
  Sample mini-batch $\mathcal{B} \subset \{1, \dots, N\}$.
  for all $i \in \mathcal{B}$ **do**
    $S_i \leftarrow {U^*}^\top M_i V^*$; $s_i \leftarrow \mathrm{vec}(S_i) \in \mathbb{R}^d$.
    Sample $z_i \sim \mathcal{N}(0, I_d)$, $t_i \sim \mathrm{Unif}(0, 1)$; set $\phi(s_i, z_i; t_i) \leftarrow (1 - t_i) s_i + t_i z_i$.
  end for
  $\mathcal{L}_{\text{FM}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \|v_\theta(\phi(s_i, z_i; t_i), t_i) - (z_i - s_i)\|_2^2$.
  $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}_{\text{FM}}$.
end for
$\theta^* \leftarrow \theta$; **return** (U^*, V^*, θ^*).

Algorithm 2: STIEFELSTEP(W, G, η) with QR retraction (generic)

Require: $W \in \mathbb{R}^{m \times R}$ with $W^\top W = I_R$; Euclidean gradient G; step size η.
$\mathrm{Sym}(A) \triangleq \frac{1}{2}(A + A^\top)$.
Tangent projection: $G^{\text{R}} \leftarrow G - W \,\mathrm{Sym}(W^\top G)$.
Retract via QR: $\widetilde{W} \leftarrow W - \eta G^{\text{R}}$; compute thin QR $\widetilde{W} = Q R_{\text{qr}}$; set $W \leftarrow Q$.
return W.

Generating a new matrix. To generate a new matrix, we draw $z \sim \mathcal{N}(0, I_d)$ and numerically solve the Neural ODE in equation 6 backward from $t = 1$ to $t = 0$, producing a generated core vector $\hat{s} \sim \hat{p}_s$. After that, we project the core vector $\hat{s}$ back to the ambient matrix space:

$$\widehat{M} \;=\; U^* \,\mathrm{mat}(\hat{s})\, {V^*}^\top.$$

Remark 3.1 (Optional patchification)**.** CoreFlow is most effective for approximately low-rank matrices. When global low-rank structure is weak but local patterns repeat, we optionally apply patchification before Stage I, inspired by patch-based vision methods (Buades et al., 2005; Dabov et al., 2007; Dosovitskiy et al., 2020). Rearranging matrices into patches can align local structure, increase row/column correlation, and reduce the effective rank. See Appendix C.3.

4 THEORETICAL PROPERTIES

We first analyze the complete matrix setting and show that the generation error is bounded by subspace error from empirical geometric recovery, core-generation error from flow matching, and a perturbation term. For incomplete matrices, Corollary B.1 in Appendix B.1 extends the same result with one additional term measuring how refilling changes the core distribution.

Assumption 4.1 (Approximate Shared Low-Rank Model)**.** The data-generating distribution p_M^* produces matrices $M \in \mathbb{R}^{m_1 \times m_2}$ of the form

$$M = U_0 S_0 V_0^\top + E,$$

where (i) *True Subspaces:* $U_0 \in \mathbb{R}^{m_1 \times R}$ and $V_0 \in \mathbb{R}^{m_2 \times R}$ are fixed unknown matrices with orthonormal columns. (ii) *Oracle Core Matrix:* $S_0 \in \mathbb{R}^{R \times R}$ is a random core matrix in the

oracle coordinates (U_0, V_0), with non-degenerate second moments, $\Sigma_L := \mathbb{E}[S_0 S_0^\top] \succ 0$ and $\Sigma_R := \mathbb{E}[S_0^\top S_0] \succ 0$, and $\mathbb{E}\|S_0\|_F^2 \le RC_S^2$. (iii) *Perturbation:* $E \in \mathbb{R}^{m_1 \times m_2}$ is a random perturbation satisfying $\mathbb{E}[E \mid S_0] = 0$ and $\mathbb{E}\|E\|_2^2 \le \tau^2$, where $\|\cdot\|_2$ denotes the spectral norm.

Next, the proposition below states that the Stage I objective (4) has the correct global minima.

Proposition 4.2. *Under Assumption 4.1, if the data is exactly low-rank, the global optimizers of the population reconstruction loss $\mathcal{L}(U, V) = \mathbb{E}\|M - UU^\top MVV^\top\|_F^2$ exactly recover the true subspaces up to orthogonal rotations:*

$$\underset{U^\top U = I,\ V^\top V = I}{\arg\min} \quad \mathcal{L}(U, V) = \{(U_0 R_1,\ V_0 R_2) : R_1 \text{ and } R_2 \text{ are orthonormal}\}.$$

The next proposition bounds the subspace error of the Tucker initializer for approximately low-rank data. When the matrices are exactly rank R and the observed cores span all R directions, the initializer recovers the true subspace exactly.

Proposition 4.3 (Tucker initialization)**.** *Suppose Assumption 4.1 holds. Define $C_L := \mathbb{E}[MM^\top]$, $\widehat{C}_L := N^{-1}\sum_{i=1}^N M_i M_i^\top$, and*

$$\delta_L := \|\widehat{C}_L - C_L\|_2 + \tau^2.$$

Let $U^{(0)}$ contain the top-R eigenvectors of $\widehat{C}_L$. If $\delta_L < \lambda_R(\Sigma_L)$, then

$$\sin\Theta(U^{(0)}, U_0) := \left\|\left(I - U_0 U_0^\top\right) U^{(0)}\right\|_2 \le \frac{\delta_L}{\lambda_R(\Sigma_L) - \delta_L}.$$

Moreover, when $\tau = 0$ and $N^{-1}\sum_{i=1}^N S_{0,i} S_{0,i}^\top \succ 0$, the initializer recovers $\mathrm{span}(U_0)$ *exactly.*

Remark 4.4. The same conclusion holds for the right subspace after replacing $(C_L, \widehat{C}_L, U_0, U^{(0)}, \Sigma_L)$ with $(C_R, \widehat{C}_R, V_0, V^{(0)}, \Sigma_R)$ in Proposition 4.3, where $C_R := \mathbb{E}[M^\top M]$ and $\widehat{C}_R := \frac{1}{N}\sum_{i=1}^N M_i^\top M_i$. Therefore, the initialized subspaces are close to the true ones when the empirical second moments are accurate and residual variation is small.

We analyze Stage II conditional on the subspaces learned in Stage I, so the induced core distribution p_s^* is fixed, and set $q := \mathcal{N}(0, I_d)$. Let X_t solve the ideal core-space ODE $\dot{X}_t = v(X_t, t)$, with $X_0 \sim p_s^*$ and $X_1 \sim q$, and let μ_t denote its distribution. Let $\hat{v}$ be the velocity learned in Stage II, and let $\hat{T}$ map each initial core at $t = 0$ to the endpoint of the learned ODE at $t = 1$. The regularity in Assumption 4.5 below makes $\hat{T}$ invertible. Then $\hat{\mu}_1 := \hat{T}_{\#} p_s^*$ is the learned forward distribution, whereas $\hat{p}_s := \hat{T}_{\#}^{-1} q$ is generated by running the learned ODE backward from Gaussian noise.

Assumption 4.5. There exist constants $\varepsilon > 0$ and $L_v > 0$ such that **(i)** the population velocity error satisfies

$$\int_0^1 \int_{\mathbb{R}^d} \|v(x,t) - \hat{v}(x,t)\|_2^2 \, d\mu_t(x)\, dt \le \varepsilon^2,$$

and **(ii)** for every x, the function $t \mapsto \hat{v}(x,t)$ is measurable; for every t, the map $x \mapsto \hat{v}(x,t)$ is L_v-Lipschitz, with the same constant L_v over $t \in [0,1]$; and $\int_0^1 \|\hat{v}(0,t)\|_2\, dt < \infty$.

Assumption 4.5 (i) treats ε as the population approximation error between the learned and ideal core-space velocity fields along the target probability path; it is not a finite-sample learning rate. Such a rate would depend on N, the core dimension R^2, the model class, and optimization error, and is not derived here. Given ε, Proposition 4.6 bounds the generated-core error, and Theorem 4.7 then propagates it to the matrix distribution together with the Stage-I error.

Proposition 4.6 (Core-space transport error)**.** *Let Assumption 4.5 hold. Then*

$$W_2(\hat{\mu}_1, q) \le e^{L_v}\varepsilon, \qquad W_2(p_s^*, \hat{p}_s) \le e^{2L_v}\varepsilon.$$

Theorem 4.7 (Ambient-space generation bound)**.** *Let Assumption 4.1 and Assumption 4.5 hold. Let (U^*, V^*) be the learned subspaces. Define the generated ambient distribution $\hat{p}_M$ by sampling $\hat{s} \sim \hat{p}_s$ and forming $\hat{M} = U^* \mathrm{mat}(\hat{s}) V^{*\top}$. Then the Wasserstein-2 distance between the true data distribution p_M^* and the generated distribution satisfies:*

$$W_2(\hat{p}_M, p_M^*) \le \underbrace{\sqrt{R} C_S (\sin\Theta(U^*, U_0) + \sin\Theta(V^*, V_0))}_{\text{(a)}} + \underbrace{e^{2L_v}\varepsilon}_{\text{(b)}} + \underbrace{\sqrt{\min(m_1, m_2)}\,\tau}_{\text{(c)}}.$$

Here, (a) *measures error in recovering the shared row and column subspaces,* (b) *measures the mismatch between the true and generated core distributions, and* (c) *bounds variation outside the learned rank-R representation.*

Remark 4.8 (Where the error comes from)**.** CoreFlow maps a core S to the matrix $U^* S V^{*\top}$. Since U^* and V^* are orthonormal, this mapping preserves Frobenius distances, so discrepancies between real and generated cores are not amplified. Let $\mathcal{E}_{\mathrm{rep}}$ be the sum of terms (a) and (c), accounting for subspace estimation and rank-R approximation error. If $\varepsilon \le e^{-2L_v}\mathcal{E}_{\mathrm{rep}}$, then term (b) is at most $\mathcal{E}_{\mathrm{rep}}$, and Theorem 4.7 gives $W_2(\hat{p}_M, p_M^*) \le 2\mathcal{E}_{\mathrm{rep}}$. Under this condition, reducing ε still decreases term (b), but improving the flow alone can only bring the bound to $\mathcal{E}_{\mathrm{rep}}$, not below it. Reducing $\mathcal{E}_{\mathrm{rep}}$ then requires more accurate subspace estimation or an R that better captures the data.

5 Experiments

Goals. Our experiments are designed to demonstrate that CoreFlow improves generative quality while training in a much lower-dimensional space than the original matrices, especially in high-dimensional and limited-sample settings. We also evaluate training matrices with missing entries as robustness tests of the same framework. Our code is included in the supplementary material on OpenReview.

Datasets. Our four real datasets span different scientific domains. Solar and Solar 2 use the 171 Å and 193 Å extreme-ultraviolet channels, respectively, of SDO/AIA (Lemen et al., 2012); each uses 300 200×200 images for training and a disjoint set of 300 for evaluation. LSPF contains 8,760 80×80 fields derived from the large-scale precipitation-fraction variable in the ERA5 single-level data (Store et al., 2024), providing a data-rich atmospheric benchmark. ERSSTv5 (Huang et al., 2017) contains 840 monthly 31×41 Pacific sea-surface-temperature anomaly fields. Together, they cover astrophysical imagery, atmospheric reanalysis, and ocean climate, with 1,271–40,000 ambient dimensions and 300–8,760 training samples.

To complement the real datasets, where the underlying low-rank structure is unknown, we use four synthetic cases with distinct spatial patterns: *Blobs* (Figure 6), *Bands* (Figure 7), *Waves* (Figure 8), and *Crosshatch* (Figure 9). Each contains $N = 1000$ training matrices of size 200×200 and enables controlled evaluation of distribution matching and subspace recovery across varied structures.

Rank selection. Let $\sigma_1 \ge \sigma_2 \ge \cdots$ denote the singular values averaged over the training matrices. We choose the smallest rank that keeps an α fraction of the spectral energy: $R_\alpha := \min\{r : \sum_{j \le r} \sigma_j^2 / \sum_j \sigma_j^2 \ge \alpha\}$. The real datasets are numerically full rank but spectrally concentrated: $R_{0.98}$ is 58 for Solar, 54 for Solar 2, and 56 for LSPF. We therefore use $R = 60$ for Solar/Solar 2 and $R = 56$ for LSPF, with $R = 120$ included to study the compression–fidelity tradeoff. For ERSSTv5, $R_{0.98} = 4$; we use $R = 12$ in the main comparisons and smaller ranks only for structured-missingness tests. Appendix C.2 provides full diagnostics.

The main comparisons mask training entries uniformly at random with $p_{\mathrm{miss}} \in \{0, 20, 40\}\%$ and evaluate generation against complete held-out matrices. Appendix D.4 also tests heterogeneous per-sample rates and contiguous block masks. CoreFlow remains robust across these patterns overall, although large contiguous holes are harder, especially for Bands. The neural network architecture and hyperparameter settings are provided in Appendix C.5.

Baselines. We compare against two baseline families. **(i)** *Ambient-space generative models with missingness handling*: *MissDiff* (Ouyang et al., 2023) and *MissFlow*, which train diffusion or flow directly in the original space using masked losses, but still operate in the full ambient dimension. **(ii)** *Low-dimensional generative models without shared matrix geometry*: *CSDM* (Guo et al., 2025), which uses a fixed compression operator and may distort matrix structure, and *SMG-Core*, a Gaussian core baseline adapted from the matrix completion method BayeSMG (Yuchi et al., 2023), where we fix the Stage I subspaces learned by CoreFlow and fit the core distribution using BayeSMG variance priors; see Appendix C.6 for details. We also report a PCA-Flow ablation in Appendix D.9, replacing Stage I with flattened PCA; since PCA requires complete vectors, it is evaluated only at $p_{\mathrm{miss}} = 0$.

Notably, VAE-based latent diffusion (Rombach et al., 2022; Peebles & Xie, 2023) is not well suited here, as it requires pre-training a deep VAE on the same limited and possibly incomplete matrices, introducing a substantially heavier pipeline that is often impossible in the few-sample regime.

Metrics. We quantify distributional similarity between generated and true matrices using: (i) entrywise moment discrepancies (mean and std), (ii) Frobenius-norm mean and std shifts, (iii) average singular-value discrepancy, and (iv) maximum mean discrepancy (MMD). For Simulation we also evaluate subspace recovery via principal angles. See formal definitions in Appendix C.1.

5.1 MAIN RESULTS

To keep the main text focused, we present representative metrics here and defer the remaining quantitative results to Appendix D (Tables 9–19), where the same overall trends continue to hold. Reported mean $\pm$ standard deviation is computed across five independent sets of 500 samples generated from the trained model. Appendix D.3 separately reports results from three independent full training runs on all four real datasets.

Table 1: Comparison with missing-data baselines on four real datasets using average singular-value discrepancy (lower is better). Columns show missing rate p_{miss}; entries are $\times 10^{-2}$ mean $\pm$ std. CoreFlow uses $\rho = 36\%$ for Solar/Solar 2, 49% for LSPF, and 11.33% for ERSSTv5, versus $\rho = 100\%$ for ambient-space baselines. Training sizes are $n = 300$ for Solar and Solar 2, 8760 for LSPF, and 840 for ERSSTv5.

	Solar			Solar 2			LSPF			ERSSTv5		
p_{miss}	0%	20%	40%	0%	20%	40%	0%	20%	40%	0%	20%	40%
MissDiff ($\rho = 100\%$)	$9.57_{\pm 0.29}$	$48.2_{\pm 0.31}$	$89.3_{\pm 0.29}$	$3.75_{\pm 0.12}$	$60.3_{\pm 0.15}$	$98.4_{\pm 0.25}$	$\mathbf{2.70_{\pm 0.20}}$	$36.2_{\pm 0.35}$	$48.7_{\pm 0.38}$	$24.67_{\pm 5.61}$	$38.37_{\pm 6.82}$	$164.00_{\pm 10.40}$
MissFlow ($\rho = 100\%$)	$7.63_{\pm 0.32}$	$37.3_{\pm 0.20}$	$75.7_{\pm 0.17}$	$2.05_{\pm 0.11}$	$41.7_{\pm 0.12}$	$86.2_{\pm 0.17}$	$2.71_{\pm 0.22}$	$33.9_{\pm 0.42}$	$45.2_{\pm 0.31}$	$23.32_{\pm 5.25}$	$28.40_{\pm 8.36}$	$86.72_{\pm 12.33}$
CoreFlow	$\mathbf{1.27_{\pm 0.49}}$	$\mathbf{1.57_{\pm 0.04}}$	$\mathbf{2.76_{\pm 0.05}}$	$\mathbf{0.77_{\pm 0.25}}$	$\mathbf{1.67_{\pm 0.18}}$	$\mathbf{1.99_{\pm 0.11}}$	$\mathbf{2.70_{\pm 0.19}}$	$\mathbf{3.73_{\pm 0.19}}$	$\mathbf{5.78_{\pm 0.10}}$	$\mathbf{21.84_{\pm 5.91}}$	$\mathbf{26.61_{\pm 4.43}}$	$\mathbf{26.71_{\pm 3.76}}$

Table 2: Comparison with low-dimensional baselines on Solar, Solar 2, and ERSSTv5, measured by average singular-value discrepancy ($\times 10^{-2}$; lower is better). Columns report p_{miss}, and entries are mean $\pm$ standard deviation. CoreFlow and SMG-Core use $\rho = 9\%$ for Solar/Solar 2 and $\rho = 11.33\%$ for ERSSTv5.

	Solar			Solar 2			ERSSTv5		
p_{miss}	0%	20%	40%	0%	20%	40%	0%	20%	40%
CSDM ($\rho = 9\%$)	$74.2_{\pm 1.1}$	$79.1_{\pm 1.0}$	$83.4_{\pm 0.90}$	$75.5_{\pm 1.8}$	$80.4_{\pm 1.4}$	$85.2_{\pm 1.1}$	$170.01_{\pm 1.13}$	$198.80_{\pm 3.03}$	$213.77_{\pm 4.59}$
CSDM ($\rho = 36\%$)	$51.5_{\pm 1.2}$	$61.8_{\pm 1.2}$	$71.5_{\pm 1.0}$	$54.0_{\pm 1.9}$	$64.3_{\pm 0.15}$	$73.5_{\pm 1.2}$	$138.30_{\pm 1.98}$	$156.69_{\pm 2.37}$	$160.83_{\pm 2.31}$
SMG-Core ($\rho = 9\%/11.33\%$)	$3.21_{\pm 0.90}$	$4.95_{\pm 1.1}$	$5.01_{\pm 1.2}$	$2.51_{\pm 0.98}$	$4.50_{\pm 1.0}$	$8.43_{\pm 1.3}$	$23.54_{\pm 5.70}$	$33.37_{\pm 6.04}$	$36.49_{\pm 3.95}$
CoreFlow ($\rho = 9\%/11.33\%$)	$\mathbf{0.87_{\pm 0.10}}$	$\mathbf{0.93_{\pm 0.07}}$	$\mathbf{1.01_{\pm 0.10}}$	$\mathbf{0.56_{\pm 0.16}}$	$\mathbf{0.64_{\pm 0.12}}$	$\mathbf{0.79_{\pm 0.07}}$	$\mathbf{21.84_{\pm 5.91}}$	$\mathbf{26.61_{\pm 4.43}}$	$\mathbf{26.71_{\pm 3.76}}$

Solar Flare (200×200, few-sample). Solar Flare is our most challenging real-data setting, with $40{,}000$ ambient dimensions but only 300 training matrices. Figures 10 and 11 show visual comparisons, while Tables 1, 9, and 10 show that CoreFlow, at $\rho = 36\%$, consistently outperforms full-space MissDiff and MissFlow on both Solar datasets. It also outperforms CSDM and SMG-Core at $\rho = 9\%$ (Tables 2, 17, and 18). One possible explanation is that fixed compression can distort spectral structure, while assuming a Gaussian core may be too restrictive.

ERSSTv5 (31×41, 840 samples). CoreFlow remains competitive with full-space models and outperforms the low-dimensional baselines while using only 11.33% of the ambient dimension (Tables 1 and 2). Its singular-value discrepancy increases only from 0.218 to 0.267 as i.i.d. missingness rises from 0% to 40%, whereas other full-space baselines deteriorate sharply at 40%. Corresponding qualitative comparisons are provided in Appendix D.1 (Figures 10 and 11).

LSPF (80×80, data-rich). On LSPF, where training data are abundant, the gap naturally narrows. Nevertheless, Table 1 shows that CoreFlow remains competitive while operating in only 49% of the ambient dimension, compared with ambient-space baselines trained at full dimension. As missingness increases, its spectral fidelity also degrades more gracefully, with the same pattern reflected in the Frobenius and entry-wise results in Tables 9 and 10.

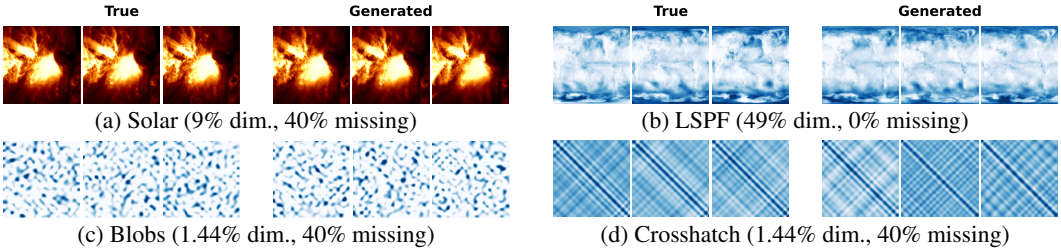
Simulation (200×200, extreme compression). As shown in Table 3, the simulation study considers an even harsher compression setting, where CoreFlow is trained in only 1.44% of the ambient dimension. Even under this extreme reduction, it preserves the matrix spectrum well across all datasets, while both CSDM and SMG-Core degrade noticeably at the same dimensionality. CSDM and SMG-Core improve only when allowed a much larger compressed space.

Robustness to missing entries. Across all benchmarks, CoreFlow degrades only mildly as p_{miss} increases, and substantially less than the baselines. This is because Stage I learns (U, V) with masked reconstruction and iterative completion (Alg. 3), while Stage II trains only in the lower-dimensional core space. Table 19 further confirms reliable subspace recovery under missingness in simulations.

Structure-aware Stage I matters. CoreFlow learns shared row/column subspaces and models only the induced core distribution, reducing dimension while preserving matrix geometry. This explains

Table 3: Comparison with low-dimensional generative baselines on simulation datasets, measured by average singular-value discrepancy (lower is better). Columns report the missing-entry rate p_{miss}. Entries are shown in $\times 10^{-2}$ as mean $\pm$ standard deviation. Parentheses indicate the learned-dimension ratio ρ.

	Blobs			Bands			Waves			Crosshatch		
p_{miss}	0%	20%	40%	0%	20%	40%	0%	20%	40%	0%	20%	40%
CSDM ($\rho = 1.44\%$)	$409_{\pm 7.4}$	$484_{\pm 7.9}$	$559_{\pm 9.1}$	$392_{\pm 7.2}$	$461_{\pm 8.3}$	$529_{\pm 9.6}$	$89.4_{\pm 3.2}$	$91.4_{\pm 2.5}$	$93.5_{\pm 1.9}$	$88.7_{\pm 2.8}$	$90.8_{\pm 3.2}$	$92.9_{\pm 4.8}$
CSDM ($\rho = 64\%$)	$197_{\pm 5.4}$	$230_{\pm 6.0}$	$261_{\pm 6.3}$	$196_{\pm 5.3}$	$226_{\pm 6.1}$	$254_{\pm 6.5}$	$43.3_{\pm 1.6}$	$57.8_{\pm 1.4}$	$69.6_{\pm 1.7}$	$41.7_{\pm 1.4}$	$56.1_{\pm 1.4}$	$67.8_{\pm 1.6}$
SMG-Core ($\rho = 1.44\%$)	$195_{\pm 4.0}$	$219_{\pm 5.1}$	$254_{\pm 5.7}$	$124_{\pm 3.3}$	$179_{\pm 4.0}$	$201_{\pm 4.9}$	$18.9_{\pm 1.0}$	$23.2_{\pm 1.2}$	$27.7_{\pm 1.4}$	$19.5_{\pm 0.90}$	$24.1_{\pm 1.1}$	$26.8_{\pm 1.3}$
SMG-Core ($\rho = 64\%$)	$20.7_{\pm 1.2}$	$28.3_{\pm 1.8}$	$34.8_{\pm 2.3}$	$16.8_{\pm 1.3}$	$19.7_{\pm 1.7}$	$24.1_{\pm 2.5}$	$6.20_{\pm 0.50}$	$10.8_{\pm 1.1}$	$15.6_{\pm 1.2}$	$8.12_{\pm 0.70}$	$12.9_{\pm 1.2}$	$17.8_{\pm 1.7}$
CoreFlow ($\rho = 1.44\%$)	$\mathbf{0.43}_{\pm 0.12}$	$\mathbf{0.63}_{\pm 0.14}$	$\mathbf{1.28}_{\pm 0.20}$	$\mathbf{4.45}_{\pm 0.74}$	$\mathbf{8.87}_{\pm 0.68}$	$\mathbf{9.09}_{\pm 0.32}$	$\mathbf{3.47}_{\pm 0.17}$	$\mathbf{5.25}_{\pm 0.07}$	$\mathbf{5.71}_{\pm 0.31}$	$\mathbf{3.23}_{\pm 0.50}$	$\mathbf{4.33}_{\pm 0.21}$	$\mathbf{4.70}_{\pm 0.26}$



(a) Solar (9% dim., 40% missing) (b) LSPF (49% dim., 0% missing)

(c) Blobs (1.44% dim., 40% missing) (d) Crosshatch (1.44% dim., 40% missing)

Figure 2: True samples (left) and CoreFlow-generated samples (right) on real and synthetic benchmarks. The close visual agreement shows that CoreFlow captures the matrix distributions well even under aggressive dimensionality reduction and substantial training missingness.

Computational efficiency. Figure 3 shows that CoreFlow achieves the best tradeoff between *generative-model cost* and generation quality by learning only in the geometry-aware core space. On Solar, it uses just 9% of the ambient dimension (about $11\times$ lower cost proxy than ambient-space models) while still achieving the best spectral fidelity. On LSPF, it reduces the training dimension to 49% ($2.1\times$ lower proxy cost) without sacrificing robustness. Full training times, including Stage I for CoreFlow and actual converged training times for the baselines, are reported in Appendix C.5.

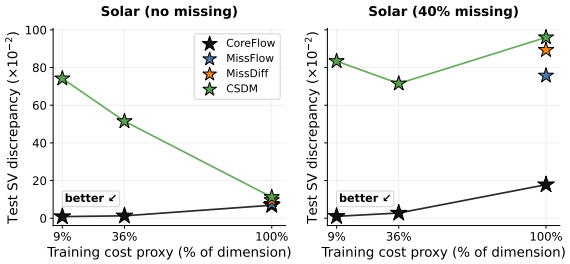


Figure 3: Performance–efficiency tradeoff on Solar among generative models. The horizontal axis is the learned-dimension ratio as a proxy for *generative-model cost*, and the vertical axis is singular-value discrepancy metric (lower is better). Full training times are reported in Appendix C.5.

its robustness under aggressive compression and missingness, where structure-agnostic compressors such as CSDM degrade. The PCA-Flow ablation in Appendix D.9 further shows that simply replacing Stage I with flattened PCA can be competitive on complete data, but is weaker overall and cannot handle missing entries. This highlights the benefit of our matrix-structured $USV^\top$ representation.

6 Conclusions

We presented CoreFlow, a geometry-preserving low-rank generative model for matrix generation that first learns shared row and column subspaces, and then fits a continuous normalizing flow only in the induced core space. This preserves matrix structure while substantially reducing the effective generative dimension. The PCA-Flow ablation further shows that low-dimensional flow modeling is useful, but that the matrix-structure-aware representation is crucial for matrix generation and for naturally handling incomplete training matrices. Theoretically, we establish an explicit separation between geometric error from subspace recovery and generative error from core-space modeling. Empirically, CoreFlow improves generation quality over ambient-space and geometry-agnostic baselines, especially in high-dimensional few-sample regimes, while remaining robust under substantial compression and missingness. A limitation is that CoreFlow is most effective when shared low-rank matrix geometry is present. While patchification can help reduce the effective rank, extending the method to more heterogeneous matrix distributions remains an important direction.

AI USE STATEMENT

We used generative AI tools only for language editing and compression. We did not use them to develop the method, derive the theory or conduct experiments. The authors reviewed all AI-assisted edits and take responsibility for the final content.

REFERENCES

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

Pierre Alquier, Vincent Cottet, Nicolas Chopin, and Judith Rousseau. Bayesian matrix completion: prior specification. *arXiv preprint arXiv:1406.1440*, 2014.

Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. NetGAN: Generating graphs via random walks. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 610–619, 2018.

Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pp. 60–65. Ieee, 2005.

Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

Emmanuel J Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

Yuejie Chi, Yue M Lu, and Yuxin Chen. Nonconvex optimization meets low-rank matrix factorization: An overview. *IEEE Transactions on Signal Processing*, 67(20):5239–5269, 2019.

Jungjun Choi, Hyukjun Kwon, and Yuan Liao. Inference for low-rank completion without sample splitting with application to treatment effect estimation. *Journal of Econometrics*, 240(1):105682, 2024.

Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16 (8):2080–2095, 2007.

Mark A Davenport and Justin Romberg. An overview of low-rank matrix recovery from incomplete observations. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):608–622, 2016.

Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. In *ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

Shuangfei Fan and Bert Huang. Labeled graph generative adversarial networks. *arXiv preprint arXiv:1906.03220*, 2019.

Paimon Goulart, Shaan Pakala, and Evangelos Papalexakis. Efficiently generating multidimensional calorimeter data with tensor decomposition parameterization. *arXiv preprint arXiv:2508.19443*, 2025.

Christian G Graff and Emil Y Sidky. Compressive sensing in medical imaging. *Applied optics*, 54(8): C23–C44, 2015.

Yuzhou Gu, Zhao Song, Junze Yin, and Lichen Zhang. Low rank matrix completion via robust alternating minimization in nearly linear time. In *The Twelfth International Conference on Learning Representations*, 2024.

Jianhua Guo, Xinbing Kong, Zeyu Li, and Junfan Mao. Tucker diffusion model for high-dimensional tensor generation. *arXiv preprint arXiv:2604.00481*, 2026.

Zhengyi Guo, Jiatu Li, Wenpin Tang, and David D Yao. Diffusion generative models meet compressed sensing, with applications to imaging and finance. *arXiv preprint arXiv:2509.03898*, 2025.

Arjun K Gupta and Daya K Nagar. *Matrix variate distributions*. Chapman and Hall/CRC, 2018.

Dylan Harries and Terence J O'Kane. Applications of matrix factorization methods to climate data. *Nonlinear Processes in Geophysics Discussions*, 2020:1–27, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Boyin Huang, Peter W Thorne, Viva F Banzon, Tim Boyer, Gennady Chepurin, Jay H Lawrimore, Matthew J Menne, Thomas M Smith, Russell S Vose, and Huai-Min Zhang. Extended reconstructed sea surface temperature, version 5 (ersstv5): upgrades, validations, and intercomparisons. *Journal of climate*, 30(20):8179–8205, 2017.

Vladimir Koltchinskii, Karim Lounici, and Alexandre B Tsybakov. Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *The Annals of Statistics*, 39(5):2302–2329, 2011.

James R Lemen, Alan M Title, David J Akin, Paul F Boerner, Catherine Chou, Jerry F Drake, Dexter W Duncan, Christopher G Edwards, Frank M Friedlaender, Gary F Heyman, et al. The atmospheric imaging assembly (aia) on the solar dynamics observatory (sdo). In *The solar dynamics observatory*, pp. 17–40. Springer, 2012.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

Lequan Lin, Zhengkun Li, Ruikun Li, Xuliang Li, and Junbin Gao. Diffusion models for time-series applications: a survey. *Frontiers of Information Technology & Electronic Engineering*, 25(1): 19–41, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Simon Mak and Yao Xie. Maximum entropy low-rank matrix recovery. *IEEE Journal of Selected Topics in Signal Processing*, 12(5):886–901, 2018.

Simon Mak, Henry Shaowu Yushi, and Yao Xie. Information-guided sampling for low-rank matrix completion. *arXiv preprint arXiv:1706.08037*, 2017.

Yidong Ouyang, Liyan Xie, Chongxuan Li, and Guang Cheng. Missdiff: Training diffusion models on tabular data with missing values. *arXiv preprint arXiv:2307.00467*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Bin Ran, Huachun Tan, Yuankai Wu, and Peter J Jin. Tensor based missing traffic data completion with spatial–temporal correlation. *Physica A: Statistical Mechanics and its Applications*, 446: 54–63, 2016.

Trevor W Richardson, Wencheng Wu, Lei Lin, Beilei Xu, and Edgar A Bernal. Mcflow: Monte carlo flow models for data imputation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14205–14214, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

William Shiao and Evangelos E. Papalexakis. Adversarially generating rank-constrained graphs. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–8. IEEE, 2021.

Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning – ICANN 2018*, pp. 412–422, 2018.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Copernicus Climate Data Store et al. Era5 hourly data on single levels from 1940 to present. *Datos recuperados entre noviembre*, 2024.

Paul Suetens. *Fundamentals of Medical Imaging*. Cambridge University Press, 2017.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.

Yifeng Tian, Nishant Panda, and Yen Ting Lin. Liouville flow importance sampler. *arXiv preprint arXiv:2405.06672*, 2024.

Zhiyuan Wang, Felix Bachofer, Jonas Koehler, Juliane Huth, Thorsten Hoeser, Mattia Marconcini, Thomas Esch, and Claudia Kuenzer. Spatial modelling and prediction with the spatio-temporal matrix: A study on predicting future settlement growth. *Land*, 11(8):1174, 2022.

Dongze Wu and Yao Xie. Annealing flow generative models towards sampling high-dimensional and multi-modal distributions. In *Forty-second International Conference on Machine Learning*, 2025.

Dongze Wu, David I Inouye, and Yao Xie. Po-flow: Flow-based generative models for sampling potential outcomes and counterfactuals. *arXiv preprint arXiv:2505.16051*, 2025.

Dongze Wu, Feng Qiu, and Yao Xie. Doflow: Flow-based generative models for interventional and counterfactual forecasting on time series. In *International Conference on Learning Representations*, 2026.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5708–5717, 2018.

Henry Shaowu Yuchi, Simon Mak, and Yao Xie. Bayesian uncertainty quantification for low-rank matrix completion. *Bayesian Analysis*, 18(2):491–518, 2023.

Hengrui Zhang, Liancheng Fang, Qitian Wu, and Philip S Yu. Diffputer: Empowering diffusion models for missing data imputation. *arXiv preprint arXiv:2405.20690*, 2024.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

Shuhan Zheng and Nontawat Charoenphakdee. Diffusion models for missing value imputation in tabular data. *arXiv preprint arXiv:2210.17128*, 2022.

A PROOFS

Proof of Proposition 4.2. Let $U \in \mathbb{R}^{m_1 \times R}$ and $V \in \mathbb{R}^{m_2 \times R}$ satisfy $U^\top U = I_R$ and $V^\top V = I_R$. Define the orthogonal projectors

$$P_U := UU^\top, \qquad P_V := VV^\top.$$

Note that $P_U^\top P_U = P_U$ and $P_V^\top P_V = P_V$. The population loss becomes

$$\mathcal{L}(U,V) = \mathbb{E}\|M - U(U^\top MV)V^\top\|_F^2 = \mathbb{E}\|M - P_U M P_V\|_F^2.$$

Expanding the Frobenius norm, we have

$$\|M - P_U M P_V\|_F^2 = \|M\|_F^2 - 2\langle M,\, P_U M P_V\rangle + \|P_U M P_V\|_F^2. \tag{7}$$

Using the cyclic property of the trace and the idempotence of projectors, the inner product term simplifies to

$$\begin{aligned}\langle M,\, P_U M P_V\rangle &= \operatorname{tr}\big(M^\top P_U M P_V\big)\\ &= \operatorname{tr}\big(M^\top P_U^\top P_U M P_V^\top P_V\big)\\ &= \operatorname{tr}\big((P_U M P_V)^\top (P_U M P_V)\big) = \|P_U M P_V\|_F^2.\end{aligned} \tag{8}$$

Substituting equation 8 into equation 7, we obtain

$$\mathcal{L}(U,V) = \mathbb{E}\|M\|_F^2 - \mathbb{E}\|P_U M P_V\|_F^2.$$

Since $\mathbb{E}\|M\|_F^2$ is independent of (U,V), minimizing $\mathcal{L}(U,V)$ is equivalent to maximizing

$$\mathcal{J}(U,V) := \mathbb{E}\|P_U M P_V\|_F^2.$$

Define matrices $A := U^\top U_0 \in \mathbb{R}^{R\times R}$ and $B := V^\top V_0 \in \mathbb{R}^{R\times R}$. Since $M = U_0 S_0 V_0^\top$ (under $\tau = 0$), we have

$$\mathcal{J}(U,V) = \mathbb{E}\|P_U U_0 S_0 V_0^\top P_V\|_F^2 = \mathbb{E}\|UAS_0B^\top V^\top\|_F^2 = \mathbb{E}\|AS_0B^\top\|_F^2,$$

where the last equality holds due to the orthogonal invariance of the Frobenius norm. Further defining $P := A^\top A = U_0^\top P_U U_0$ and $Q := B^\top B = V_0^\top P_V V_0$, we can rewrite the objective as:

$$\mathcal{J}(U,V) = \mathbb{E}\operatorname{tr}(AS_0B^\top BS_0^\top A^\top) = \mathbb{E}\operatorname{tr}(PS_0QS_0^\top) = \operatorname{tr}(P\,\mathbb{E}\,[S_0QS_0^\top]).$$

It is immediate that $P \succeq 0$ and $Q \succeq 0$. Furthermore, for any vector $x \in \mathbb{R}^R$, since multiplication by $U^\top$ (where U has orthonormal columns) cannot increase the ℓ_2-norm of a vector, we know that

$$x^\top P x = x^\top U_0^\top UU^\top U_0 x = \|U^\top(U_0x)\|_2^2 \le \|U_0x\|_2^2 = \|x\|_2^2.$$

The identical logic applies to Q. Therefore, the eigenvalues of P and Q are bounded:

$$0 \preceq P \preceq I_R, \qquad 0 \preceq Q \preceq I_R.$$

Using $P \preceq I_R$ and the fact that $\mathbb{E}[S_0QS_0^\top] \succeq 0$ it follows that

$$\mathcal{J}(U,V) = \operatorname{tr}(P\,\mathbb{E}[S_0QS_0^\top]) \le \operatorname{tr}(\mathbb{E}[S_0QS_0^\top]). \tag{9}$$

Using the cyclicity of the trace and recall that $\Sigma_R = \mathbb{E}[S_0^\top S_0]$, we have

$$\operatorname{tr}(\mathbb{E}[S_0QS_0^\top]) = \operatorname{tr}(Q\,\mathbb{E}[S_0^\top S_0]) = \operatorname{tr}(Q\Sigma_R).$$

Similarly, since $Q \preceq I_R$ and $\Sigma_R \succ 0$ (by Assumption 4.1), we further bound this by

$$\operatorname{tr}(Q\Sigma_R) \le \operatorname{tr}(\Sigma_R) = \mathbb{E}\operatorname{tr}(S_0^\top S_0) = \mathbb{E}\|S_0\|_F^2. \tag{10}$$

Combining equation 9 and equation 10 yields the global upper bound $\mathcal{J}(U,V) \le \mathbb{E}\|S_0\|_F^2$.

This bound is tight and strictly achievable. If we set $(U,V) = (U_0,V_0)$, then $P_{U_0} M P_{V_0} = U_0U_0^\top MV_0V_0^\top = M$. Thus

$$\mathcal{J}(U_0,V_0) = \mathbb{E}\|P_{U_0} M P_{V_0}\|_F^2 = \mathbb{E}\|M\|_F^2 = \mathbb{E}\|U_0S_0V_0^\top\|_F^2 = \mathbb{E}\|S_0\|_F^2.$$

If a pair (U, V) is a global optimizer, it must achieve the maximal upper bound. Consequently, the inequalities in equation 9 and equation 10 must hold as equalities, meaning:

$$\mathrm{tr}((I_R - Q)\Sigma_R) = 0 \quad \text{and} \quad \mathrm{tr}((I_R - P)\mathbb{E}[S_0 Q S_0^\top]) = 0.$$

Since $\Sigma_R \succ 0$ and $I_R - Q \succeq 0$, the trace of their product is zero if and only if $I_R - Q = 0$, implying $Q = I_R$. Consequently, $\mathbb{E}[S_0 Q S_0^\top] = \mathbb{E}[S_0 S_0^\top] = \Sigma_L \succ 0$. By the same logic, $I_R - P \succeq 0$ combined with $\Sigma_L \succ 0$ implies $P = I_R$.

The conditions $P = I_R$ and $Q = I_R$ explicitly translate to

$$U_0^\top P_U U_0 = I_R \quad \text{and} \quad V_0^\top P_V V_0 = I_R.$$

Focusing on U, we can rewrite $U_0^\top P_U U_0 = I_R$ as $U_0^\top (I_{m_1} - P_U) U_0 = 0$. Since $I_{m_1} - P_U$ is a positive semi-definite projection matrix, this requires $(I_{m_1} - P_U)U_0 = 0$, or equivalently, $P_U U_0 = U_0$. This means $U(U^\top U_0) = U_0$, which strictly implies that $\mathrm{span}(U_0) \subseteq \mathrm{span}(U)$. Since both U and U_0 have exactly rank R, their column spaces must be identical: $\mathrm{span}(U_0) = \mathrm{span}(U)$. By symmetric reasoning, $\mathrm{span}(V_0) = \mathrm{span}(V)$. Thus, the global optimizers are equivalent to the true subspaces up to orthogonal rotations. □

Proof of Proposition 4.3. Let $A_L := U_0 \Sigma_L U_0^\top$. Expanding $MM^\top$ and using $\mathbb{E}[E \mid S_0] = 0$ gives

$$C_L = A_L + \Sigma_{E,L}, \qquad \Sigma_{E,L} := \mathbb{E}[EE^\top].$$

Moreover, $\|\Sigma_{E,L}\|_2 \le \mathbb{E}\|E\|_2^2 \le \tau^2$. Hence

$$\|\widehat{C}_L - A_L\|_2 \le \|\widehat{C}_L - C_L\|_2 + \|\Sigma_{E,L}\|_2 \le \delta_L.$$

The matrix A_L has rank R, its leading eigenspace is $\mathrm{span}(U_0)$, and its R-th eigenvalue is $\lambda_R(\Sigma_L)$. Weyl's inequality therefore gives

$$\lambda_R(\widehat{C}_L) \ge \lambda_R(\Sigma_L) - \delta_L > 0.$$

Let $\widehat{C}_L U^{(0)} = U^{(0)} \Lambda^{(0)}$, where $\lambda_{\min}(\Lambda^{(0)}) = \lambda_R(\widehat{C}_L)$, and let $U_{0,\perp}$ be an orthonormal complement of U_0. Since $U_{0,\perp}^\top A_L = 0$,

$$U_{0,\perp}^\top U^{(0)} \Lambda^{(0)} = U_{0,\perp}^\top (\widehat{C}_L - A_L) U^{(0)}.$$

Taking spectral norms yields

$$\sin\Theta(U^{(0)}, U_0) = \|U_{0,\perp}^\top U^{(0)}\|_2 \le \frac{\delta_L}{\lambda_R(\Sigma_L) - \delta_L}.$$

If $\tau = 0$, then $\widehat{C}_L = U_0 \widehat{\Sigma}_L U_0^\top$, where $\widehat{\Sigma}_L = N^{-1} \sum_i S_{0,i} S_{0,i}^\top$. When $\widehat{\Sigma}_L \succ 0$, $\widehat{C}_L$ has rank R and range exactly $\mathrm{span}(U_0)$, proving exact recovery. □

Proof of Proposition 4.6. Let X_t be the ideal trajectory defined above. Starting from the same core X_0, let Y_t follow the learned ODE:

$$\dot{Y}_t = \hat{v}(Y_t, t), \qquad Y_0 = X_0.$$

Then $X_t \sim \mu_t$, $Y_1 \sim \hat{\mu}_1$, and $X_1 \sim q$. Subtracting the two ODEs and applying the spatial Lipschitz condition and Grönwall's inequality gives

$$\|X_1 - Y_1\|_2 \le e^{L_v} \int_0^1 \|v(X_t, t) - \hat{v}(X_t, t)\|_2 \, dt.$$

Jensen's inequality and Assumption 4.5(i) give

$$\mathbb{E}\|X_1 - Y_1\|_2^2 \le e^{2L_v} \int_0^1 \mathbb{E}\|v(X_t, t) - \hat{v}(X_t, t)\|_2^2 dt \le e^{2L_v} \varepsilon^2.$$

The pair (X_1, Y_1) is therefore a coupling of q and $\hat{\mu}_1$, so $W_2(\hat{\mu}_1, q) \le e^{L_v} \varepsilon$.

The inverse flow $\hat{T}^{-1}$ is e^{L_v}-Lipschitz by the same Grönwall argument. Since $p_s^* = \hat{T}_\#^{-1} \hat{\mu}_1$ and $\hat{p}_s = \hat{T}_\#^{-1} q$,

$$W_2(p_s^*, \hat{p}_s) \le e^{L_v} W_2(\hat{\mu}_1, q) \le e^{2L_v} \varepsilon.$$

□

Proof of Theorem 4.7. Condition on the fixed Stage-I output (U^*, V^*), and define

$$P_U := U^* U^{*\top}, \qquad P_V := V^* V^{*\top}.$$

For $M \sim p^*_M$, let

$$S := U^{*\top} M V^*, \qquad s := \mathrm{vec}(S),$$

so that $s \sim p^*_s$. Let $\hat{S} := \mathrm{mat}(\hat{s})$, where $\hat{s} \sim \hat{p}_s$, and define

$$\hat{M} := U^* \hat{S} V^{*\top}.$$

We couple $(s, \hat{s})$ via the optimal W_2-coupling between p^*_s and $\hat{p}_s$, so that

$$\mathbb{E}\|S - \hat{S}\|_F^2 = \mathbb{E}\|s - \hat{s}\|_2^2 = W_2^2(p^*_s, \hat{p}_s).$$

Complete this coupling by drawing (M, S_0, E) from the model in Assumption 4.1, conditional on the sampled core $S = U^{*\top} M V^*$. Pairing this draw with $\hat{S}$ gives a joint distribution of $(M, \hat{M})$ with marginals p^*_M and $\hat{p}_M$. Therefore,

$$W_2^2(\hat{p}_M, p^*_M) \le \mathbb{E}\|M - \hat{M}\|_F^2.$$

Using $S = U^{*\top} M V^*$, we have

$$U^* S V^{*\top} = U^* U^{*\top} M V^* V^{*\top} = P_U M P_V.$$

To avoid overloading notation, write the oracle low-rank representation in Assumption 4.1 as

$$M = U_0 S_0 V_0^\top + E,$$

where S_0 is the oracle core. Adding and subtracting the intermediate term $U^* S V^{*\top} = P_U M P_V$ gives

$$\begin{aligned} M - \hat{M} &= \big(M - U^* S V^{*\top}\big) + \big(U^* S V^{*\top} - \hat{M}\big) \\ &= U_0 S_0 V_0^\top + E - P_U (U_0 S_0 V_0^\top + E) P_V + U^* S V^{*\top} - U^* \hat{S} V^{*\top} \\ &= \underbrace{\big(U_0 S_0 V_0^\top - P_U U_0 S_0 V_0^\top P_V\big)}_{=:\ \Delta_{\mathrm{sub}}} + \underbrace{U^*(S - \hat{S}) V^{*\top}}_{=:\ \Delta_{\mathrm{core}}} + \underbrace{\big(E - P_U E P_V\big)}_{=:\ \Delta_{\mathrm{noise}}}. \end{aligned}$$

Applying Minkowski's inequality in L^2 over the joint distribution of $(M, S, \hat{S})$,

$$\big(\mathbb{E}\|M - \hat{M}\|_F^2\big)^{1/2} \le \big(\mathbb{E}\|\Delta_{\mathrm{sub}}\|_F^2\big)^{1/2} + \big(\mathbb{E}\|\Delta_{\mathrm{core}}\|_F^2\big)^{1/2} + \big(\mathbb{E}\|\Delta_{\mathrm{noise}}\|_F^2\big)^{1/2}. \tag{11}$$

We first bound the subspace mismatch term. Since

$$\Delta_{\mathrm{sub}} = U_0 S_0 V_0^\top - P_U U_0 S_0 V_0^\top P_V,$$

we decompose

$$\Delta_{\mathrm{sub}} = (I - P_U) U_0 S_0 V_0^\top + P_U U_0 S_0 V_0^\top (I - P_V).$$

Principal angles are symmetric in the two subspaces, so $\|(I - P_U) U_0\|_2 = \sin\Theta(U^*, U_0)$ and $\|V_0^\top (I - P_V)\|_2 = \sin\Theta(V^*, V_0)$. By the triangle inequality and mixed-norm submultiplicativity, and the fact that $\|V_0\|_2 = 1$ and $\|P_U U_0\|_2 \le 1$,

$$\begin{aligned} \|\Delta_{\mathrm{sub}}\|_F &\le \|(I - P_U) U_0 S_0 V_0^\top\|_F + \|P_U U_0 S_0 V_0^\top (I - P_V)\|_F \\ &\le \|(I - P_U) U_0\|_2 \|S_0\|_F \|V_0\|_2 + \|P_U U_0\|_2 \|S_0\|_F \|V_0^\top (I - P_V)\|_2 \\ &\le \big(\sin\Theta(U^*, U_0) + \sin\Theta(V^*, V_0)\big) \|S_0\|_F. \end{aligned}$$

Taking the L^2-norm gives

$$\begin{aligned} \big(\mathbb{E}\|\Delta_{\mathrm{sub}}\|_F^2\big)^{1/2} &\le \big(\mathbb{E}\|S_0\|_F^2\big)^{1/2} \big(\sin\Theta(U^*, U_0) + \sin\Theta(V^*, V_0)\big) \\ &\le \sqrt{R}\, C_S \big(\sin\Theta(U^*, U_0) + \sin\Theta(V^*, V_0)\big), \end{aligned} \tag{12}$$

where we used Assumption 4.1(ii): $\mathbb{E}\|S_0\|_F^2 \le R C_S^2$.

Next, since U^* and V^* have orthonormal columns,

$$\mathbb{E}\|\Delta_{\text{core}}\|_F^2 = \mathbb{E}\|U^*(S-\hat{S})V^{*\top}\|_F^2 = \mathbb{E}\|S-\hat{S}\|_F^2 = W_2^2(p_s^*, \hat{p}_s).$$

Applying Proposition 4.6 gives

$$W_2(p_s^*, \hat{p}_s) \le e^{2L_v}\varepsilon. \tag{13}$$

Finally, the noise term is bounded directly by Assumption 4.1. Since $E \mapsto P_U E P_V$ is an orthogonal projection in Frobenius norm,

$$\|E - P_U E P_V\|_F \le \|E\|_F.$$

Hence

$$\begin{aligned}\big(\mathbb{E}\|\Delta_{\text{noise}}\|_F^2\big)^{1/2} &= \big(\mathbb{E}\|E - P_U E P_V\|_F^2\big)^{1/2} \\ &\le \big(\mathbb{E}\|E\|_F^2\big)^{1/2} \\ &\le \sqrt{\min(m_1,m_2)}\cdot\big(\mathbb{E}\|E\|_2^2\big)^{1/2} \le \sqrt{\min(m_1,m_2)}\,\tau.\end{aligned} \tag{14}$$

Substituting equation 12, equation 13, and equation 14 into equation 11 gives

$$W_2(\hat{p}_M, p_M^*) \le \sqrt{R}C_S\big(\sin\Theta(U^*,U_0) + \sin\Theta(V^*,V_0)\big) + e^{2L_v}\varepsilon + \sqrt{\min(m_1,m_2)}\,\tau.$$

This proves the theorem. □

Remark A.1 (Error metric alignment)**.** Proposition 4.6 starts the ideal and learned ODEs from the same core. Comparing their endpoints gives the first factor e^{L_v}, and mapping both endpoint distributions back through the learned inverse flow gives the second. Since U^* and V^* are orthonormal, converting cores to matrices preserves Frobenius distances and introduces no additional factor.

B Stage I: Learn (U, V) from Incomplete Matrices

The Stage I training procedure for incomplete matrices is provided in Algorithm 3.

Algorithm 3: Stage I for CoreFlow: Learn (U, V) from Incomplete Matrices

Require: Observations $\{(M_i^{\text{obs}}, \Omega_i)\}_{i=1}^N$; rank R; outer iterative steps E; inner iterative steps T_{sub}; learning rates η_U, η_V.

Initialize filled-in states $\widetilde{M}_i \leftarrow \mathcal{P}_{\Omega_i}(M_i^{\text{obs}})$ for all i.
Initialize $U \in \text{St}(m_1, R)$, $V \in \text{St}(m_2, R)$ via spectral initialization.
for $e = 1$ to E **do**
  (a) Subspace update with masked loss
  for $t = 1$ to T_{sub} **do**
    Sample mini-batch $\mathcal{B} \subset \{1, \dots, N\}$.
    $\mathcal{L}_{\text{miss}} \leftarrow \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \Big\|\mathcal{P}_{\Omega_i}\Big(M_i^{\text{obs}} - UU^\top \widetilde{M}_i V V^\top\Big)\Big\|_F^2$.
    Compute $G_U \leftarrow \nabla_U \mathcal{L}_{\text{miss}}$, $G_V \leftarrow \nabla_V \mathcal{L}_{\text{miss}}$.
    $U \leftarrow \text{StiefelStep}(U, G_U, \eta_U)$; $V \leftarrow \text{StiefelStep}(V, G_V, \eta_V)$.
  end for
  (b) Completion update: keep observed entries, impute missing ones
  for $i = 1$ to N **do**
    $\widehat{M}_i \leftarrow U\,U^\top \widetilde{M}_i V\,V^\top$.
    $\widetilde{M}_i \leftarrow \mathcal{P}_{\Omega_i}(M_i^{\text{obs}}) + \mathcal{P}_{\Omega_i^c}(\widehat{M}_i)$.
  end for
end for
return $U^* \leftarrow U,\ V^* \leftarrow V,\ \{\widetilde{M}_i\}_{i=1}^N$.

B.1 Extension to incomplete observations

Under independent entrywise missingness, suppose each entry is observed with probability π. Let A be any residual matrix fixed before the random mask Ω is drawn.

$$\mathbb{E}_\Omega\left[\pi^{-1}\|\mathcal{P}_\Omega(A)\|_F^2\right] = \|A\|_F^2. \tag{15}$$

For a fixed residual, Equation 15 says that the observed squared error, divided by the observation probability, equals the full squared error on average over masks. This explains the masked Stage I objective. In Algorithm 3, however, refilling makes the residual depend on the observed mask. The identity alone therefore does not guarantee that the alternating procedure recovers the correct subspaces or cores.

We next isolate how cores computed from refilled matrices change the complete-data generation bound.

Corollary B.1 (Effect of refilling on generation)**.** *Fix the learned subspaces (U^*, V^*). For a new matrix M and mask Ω, let $\widetilde{M}$ be the matrix obtained by applying the refilling procedure in Algorithm 3 to the observed entries $\mathcal{P}_\Omega(M)$. Define the complete and refilled cores by*

$$s = \mathrm{vec}({U^*}^\top M V^*), \qquad \widetilde{s} = \mathrm{vec}({U^*}^\top \widetilde{M} V^*),$$

and let p_s^* and $p_{\widetilde{s}}^*$ denote their distributions. Suppose Assumption 4.1 holds and that Assumption 4.5 is satisfied with $p_{\widetilde{s}}^*$ in place of p_s^*. Then

$$\begin{aligned} W_2(\hat{p}_M, p_M^*) \le \sqrt{R} C_S(\sin\Theta(U^*, U_0) + \sin\Theta(V^*, V_0)) + e^{2L_v}\varepsilon \\ + \sqrt{\min(m_1, m_2)}\,\tau + W_2(p_s^*, p_{\widetilde{s}}^*), \end{aligned}$$

where the last term measures the distributional change caused by refilling and satisfies

$$W_2(p_s^*, p_{\widetilde{s}}^*) \le \left(\mathbb{E}\|{U^*}^\top (M - \widetilde{M}) V^*\|_F^2\right)^{1/2} \le \left(\mathbb{E}\|M - \widetilde{M}\|_F^2\right)^{1/2}.$$

Proof. Theorem 4.7 contains the core term $W_2(p_s^*, \hat{p}_s)$. Since Stage II is trained on refilled rather than complete cores, we insert $p_{\widetilde{s}}^*$ and apply the triangle inequality:

$$W_2(p_s^*, \hat{p}_s) \le W_2(p_s^*, p_{\widetilde{s}}^*) + W_2(p_{\widetilde{s}}^*, \hat{p}_s).$$

Proposition 4.6, applied to the refilled cores, bounds the second term by $e^{2L_v}\varepsilon$; the other terms in Theorem 4.7 are unchanged. We bound the first term by pairing each complete matrix with its own refilled version:

$$\begin{aligned} W_2^2(p_s^*, p_{\widetilde{s}}^*) &\le \mathbb{E}\|s - \widetilde{s}\|_2^2 \\ &= \mathbb{E}\|{U^*}^\top (M - \widetilde{M}) V^*\|_F^2 \le \mathbb{E}\|M - \widetilde{M}\|_F^2, \end{aligned}$$

where the last inequality uses the orthonormality of U^* and V^*. □

Remark B.2 (Scope of the incomplete-data result)**.** Once (U^*, V^*) are learned, Stage II uses only the projected $R \times R$ cores, so refilling need not recover every missing entry exactly. What matters is how much it changes the core distribution. Corollary B.1 captures this change separately from subspace error. Showing that both errors are small for Algorithm 3 requires stronger assumptions than we make here: the masks must reveal enough information about each shared direction, the subspaces must be identifiable (e.g., through incoherence and a spectral gap), the residual variation must be controlled, and the alternating updates must be suitably initialized. We do not prove such a finite-sample recovery result here. The corollary shows how any Stage I recovery guarantee would translate into the final generation bound.

C Experimental Details

Real-data evaluation protocol. Solar and Solar 2 each use 300 matrices for training and a disjoint set of 300 complete matrices for evaluation. LSPF is trained on 8,760 fields and evaluated on a disjoint set of 500 test fields. ERSSTv5 is split by time into 840 training fields from 1940–2009, 72 validation fields from 2010–2015, and 84 test fields from 2016–2022.

Missingness is applied only to the training matrices, and every model generates complete matrices. Within each dataset and missingness setting, all methods use the same training data, masks, complete reference set, number of generated samples, and evaluation code. Thus, comparisons are matched within each dataset and use held-out references. For ERSSTv5, we use monthly NOAA anomalies relative to the 1971–2000 climatology on the native 2° Pacific grid over 30°S–30°N and 160°E–240°E. Temperatures in °C are divided by 5, with no additional interpolation, smoothing, imputation, or rank truncation.

For each of five sampling seeds, every trained model generates 500 samples to compute the entrywise, Frobenius, and singular-value metrics. Each batch is compared with the same complete reference set, and we report the mean and standard deviation across the five batches.

C.1 METRICS

Let $\{M_b^{\text{true}}\}_{b=1}^{B_{\text{true}}}$ denote the set of ground-truth matrices and $\{M_b^{\text{gen}}\}_{b=1}^{B_{\text{gen}}}$ denote the set of generated matrices, where each $M \in \mathbb{R}^{m_1 \times m_2}$. We calculate the metrics below that compare the empirical distributions induced by these two sets.

Entrywise moment discrepancy (mean). For each entry $j \in \{1, \dots, m_1 m_2\}$ (after flattening the matrix), we define the per-entry empirical means as:

$$\mu_j^{\text{true}} \;=\; \frac{1}{B_{\text{true}}} \sum_{b=1}^{B_{\text{true}}} M_{b,j}^{\text{true}}, \qquad \mu_j^{\text{gen}} \;=\; \frac{1}{B_{\text{gen}}} \sum_{b=1}^{B_{\text{gen}}} M_{b,j}^{\text{gen}},$$

and the absolute per-entry mean difference is $\Delta_j^{\mu} = |\mu_j^{\text{gen}} - \mu_j^{\text{true}}|$. We report the average of $\{\Delta_j^{\mu}\}_{j=1}^{m_1 m_2}$ across all entries:

$$\text{AbsEntryMeanDiff} \;=\; \frac{1}{m_1 m_2} \sum_{j=1}^{m_1 m_2} \Delta_j^{\mu}.$$

Entrywise moment discrepancy (standard deviation). Analogously, we define the per-entry empirical standard deviations as:

$$\sigma_j^{\text{true}} \;=\; \text{Std}\Big(\{M_{b,j}^{\text{true}}\}_{b=1}^{B_{\text{true}}}\Big), \qquad \sigma_j^{\text{gen}} \;=\; \text{Std}\Big(\{M_{b,j}^{\text{gen}}\}_{b=1}^{B_{\text{gen}}}\Big),$$

and the absolute per-entry standard-deviation difference $\Delta_j^{\sigma} = |\sigma_j^{\text{gen}} - \sigma_j^{\text{true}}|$. We report the average of it across all entries:

$$\text{AbsEntryStdDiff} \;=\; \frac{1}{m_1 m_2} \sum_{j=1}^{m_1 m_2} \Delta_j^{\sigma}.$$

These entrywise metrics quantify how well the generator matches the first two moments *at each spatial location*.

Frobenius-norm distribution shift. For each matrix, we compute its Frobenius norm:

$$r_b^{\text{true}} \;=\; \|M_b^{\text{true}}\|_F, \qquad r_b^{\text{gen}} \;=\; \|M_b^{\text{gen}}\|_F.$$

Let $\bar{r}^{\text{true}}$ and $\bar{r}^{\text{gen}}$ denote the empirical means of $\{r_b^{\text{true}}\}_{b=1}^{B_{\text{true}}}$ and $\{r_b^{\text{gen}}\}_{b=1}^{B_{\text{gen}}}$, and we let s^{true} and s^{gen} denote the empirical standard deviations. We report absolute differences

$$\text{FrobMeanDiff} \;=\; |\bar{r}^{\text{gen}} - \bar{r}^{\text{true}}|, \qquad \text{FrobStdDiff} \;=\; |s^{\text{gen}} - s^{\text{true}}|.$$

These statistics summarize distributional shifts in overall matrix energy and its variability.

Average singular-value spectrum distance. Let $\sigma_i(M)$ denote the i-th singular value of M, for $i = 1, \dots, k$ with $k = \min(m_1, m_2)$. We define the average singular-value spectra as:

$$\bar{\boldsymbol{\sigma}}^{\text{true}} \;=\; \frac{1}{B_{\text{true}}} \sum_{b=1}^{B_{\text{true}}} \big(\sigma_1(M_b^{\text{true}}), \dots, \sigma_k(M_b^{\text{true}})\big), \qquad \bar{\boldsymbol{\sigma}}^{\text{gen}} \;=\; \frac{1}{B_{\text{gen}}} \sum_{b=1}^{B_{\text{gen}}} \big(\sigma_1(M_b^{\text{gen}}), \dots, \sigma_k(M_b^{\text{gen}})\big).$$

We then compute the relative ℓ_2 discrepancy as:

$$\mathrm{SVRelL2} \;=\; \frac{\|\bar{\boldsymbol{\sigma}}^{\text{true}} - \bar{\boldsymbol{\sigma}}^{\text{gen}}\|_2}{\|\bar{\boldsymbol{\sigma}}^{\text{true}}\|_2 + \varepsilon}, \qquad \varepsilon = 10^{-8}.$$

This metric compares the global spectral structure of the two sets of matrices.

We also evaluate how good the estimates are about U, V, as well as generated low-rank matrix M.

Principal Angles. For simulation cases where U_{true} and V_{true} are available, we also measure subspace recovery for both U and V. Given the ground-truth and estimated factors $U_{\text{true}}, U^* \in \mathbb{R}^{m\times R}$ (analogously V_{true}, V^*), we first orthonormalize the columns via reduced QR:

$$Q_{\text{true}} = \mathrm{qr}(U_{\text{true}}), \qquad Q^* = \mathrm{qr}(U^*),$$

and form the overlap matrix $M = Q_{\text{true}}^\top Q^* \in \mathbb{R}^{R\times R}$. As in our implementation, we compute singular values $\{\sigma_r\}_{r=1}^R = \mathrm{svdvals}(M)$, clamp them to $[-1, 1]$, and define principal angles by

$$\theta_r = \arccos(\sigma_r) \in \left[0, \tfrac{\pi}{2}\right] \quad (\text{equivalently } \theta_r \in [0, 90^\circ]).$$

We report the mean principal angle (in degrees) for U and V separately:

$$\overline{\theta}_U = \frac{1}{R}\sum_{r=1}^{R} \theta_r^{(U)}, \qquad \overline{\theta}_V = \frac{1}{R}\sum_{r=1}^{R} \theta_r^{(V)}.$$

Maximum Mean Discrepancy (MMD). To compare the generated and true matrix distributions directly in ambient matrix space, we also report the maximum mean discrepancy (MMD) with an RBF kernel on Frobenius distance. Given true and generated samples $\{M_i^{\text{true}}\}_{i=1}^n$ and $\{\widehat{M}_j\}_{j=1}^m$, we define

$$k(M, \widetilde{M}) = \exp\left(-\frac{\|M - \widetilde{M}\|_F^2}{2\sigma^2}\right),$$

where the bandwidth is chosen by the pooled-sample median,

$$\sigma^2 = \mathrm{median}\Big\{\|Z_a - Z_b\|_F^2 :\; a < b,\; Z \in \{M_i^{\text{true}}\}_{i=1}^n \cup \{\widehat{M}_j\}_{j=1}^m\Big\}.$$

As in our implementation, we use the unbiased U-statistic estimator

$$\widehat{\mathrm{MMD}}_u^2 = \frac{1}{n(n-1)}\sum_{i\neq i'} k(M_i^{\text{true}}, M_{i'}^{\text{true}}) + \frac{1}{m(m-1)}\sum_{j\neq j'} k(\widehat{M}_j, \widehat{M}_{j'}) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k(M_i^{\text{true}}, \widehat{M}_j),$$

and report $\mathrm{MMD} = \sqrt{\max(\widehat{\mathrm{MMD}}_u^2, 0)}$, where smaller values indicate better distributional alignment.

C.2 Rank selection and spectral concentration

For an explained-energy level α, Table 4 reports R_α from the rule in the main text. The 98% rule closely matches the primary experimental ranks; the larger Solar/Solar 2 rank is included to evaluate the compression–fidelity tradeoff.

Table 5 distinguishes exact from approximate low-rank structure. The simulations generally have a sharp cutoff near rank 24, whereas the real datasets have gradual tails and full numerical rank despite concentrating 98% of their energy in far fewer directions.

C.3 Patchification details and illustrations

Many real-world matrices are not globally low-rank, yet they often exhibit strong *local* redundancy. Motivated by patch-based computer vision methods that exploit self-similarity through patch extraction and grouping (e.g., (Buades et al., 2005; Dabov et al., 2007; Dosovitskiy et al., 2020)), we can

Table 4: Ranks selected by explained spectral energy. "Paper R" lists the ranks used in the reported experiments.

Dataset	**Paper** R	$R_{0.90}$	$R_{0.95}$	$R_{0.98}$
Blobs	24	22	23	24
Bands	24	21	23	24
Waves	24	20	22	24
Crosshatch	24	20	22	24
Solar	60/120	29	43	58
Solar 2	60/120	24	37	54
LSPF	56	31	47	56
ERSSTv5	6/8/12	2	3	4

Table 5: Spectral-tail diagnostics. The count is the number of singular values exceeding 1% of σ_1.

Dataset	σ_{25}/σ_1	$\#\{j : \sigma_j > 0.01\sigma_1\}$	**Interpretation**
Blobs	4.37×10^{-7}	24	sharp cutoff
Bands	3.93×10^{-7}	24	sharp cutoff
Waves	1.52×10^{-3}	24	sharp cutoff
Crosshatch	1.96×10^{-2}	25	mild tail
Solar	2.23×10^{-1}	181	gradual decay
Solar 2	1.34×10^{-1}	182	gradual decay
LSPF	1.46×10^{-1}	80	gradual decay
ERSSTv5	1.64×10^{-3}	11	concentrated spectrum

apply a simple *patchification* operator before learning shared subspaces, which yields an effectively lower-rank representation even when the original matrix is not inherently low-rank.

Specifically, given a matrix $M \in \mathbb{R}^{m_1 \times m_2}$, we choose a patch size $p_1 \times p_2$ and extract n_p local submatrices (patches), denoted by $\{P_k(M) \in \mathbb{R}^{p_1 \times p_2}\}_{k=1}^{n_p}$. We then form the *patch matrix* by vectorizing each patch and stacking the resulting row vectors:

$$M_{\text{patch}} \triangleq \begin{bmatrix} \text{vec}(P_1(M))^\top \\ \vdots \\ \text{vec}(P_{n_p}(M))^\top \end{bmatrix} \in \mathbb{R}^{n_p \times (p_1 p_2)}. \tag{16}$$

Because repeated local patterns induce highly correlated rows or columns in M_{patch}, it typically has a much faster singular-value decay (i.e., a lower effective rank) than M. See Figures 4 and 5 for an illustrative spectrum comparison demonstrating the rank reduction by patchification.

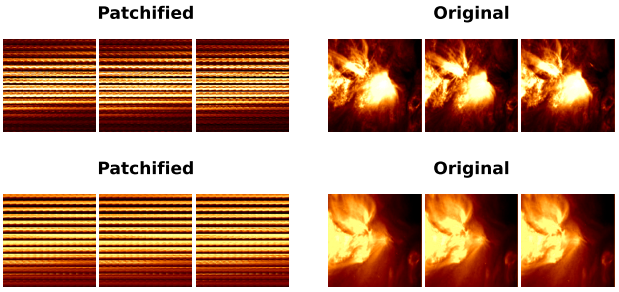
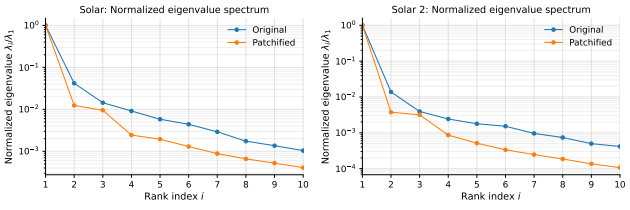


Figure 4: **Patchified vs. original matrices (Solar).** Patchification (Eq. equation 16) reshapes local patches into rows of a patch-matrix. In the patchified view (left), many rows/columns become visually near-repetitions due to recurring local textures, while the original matrices (right) remain globally diverse.



(a) Solar. (b) Solar 2.

Figure 5: **Consistent rank reduction via patchification.** Normalized spectra (λ_i/λ_1) show that the patchified representation decays faster: for $i \geq 2$, the patchified curve lies consistently below the original, indicating that more energy concentrates in the leading components and the representation has a lower effective rank.

By reorganizing local patches into a patch-matrix via Eq. equation 16, patchification can convert matrices that are not *globally* low-rank into an *effectively* lower-rank representation by leveraging local redundancy. As shown in Fig. 4, stacking vectorized patches makes recurring local structures (e.g., horizontal bands) appear as near-repeated rows/columns, which increases correlation and reduces the effective rank. Figure 5 reports the normalized spectrum λ_i/λ_1 for $i = 1, \ldots, 10$. Across both Solar and Solar 2, the patchified curve is consistently below the original, indicating faster spectral decay and a lower effective rank.

Patchification for Solar datasets. For Solar Flare experiments, we apply deterministic non-overlapping patchification before Stage I. Given $X_i \in \mathbb{R}^{H\times W}$, we set

$$p = \mathrm{round}\big((HW)^{1/4}\big), \qquad H_c = \lfloor H/p \rfloor p, \qquad W_c = \lfloor W/p \rfloor p.$$

Each matrix is cropped to $H_c \times W_c$, divided into non-overlapping $p \times p$ patches, and rearranged into a matrix of size

$$m_1 = \frac{H_c W_c}{p^2}, \qquad m_2 = p^2.$$

For the 200×200 Solar/Solar 2 data, this gives $p = 14$, $H_c = W_c = 196$, and patched matrices of size 196×196. Generated samples are mapped back to the cropped image domain by the inverse patch operation before visualization and metric computation. Patchification is not used for LSPF or synthetic datasets. In all reported comparisons where patchification is used, CoreFlow and all baselines use the same patch size, cropping rule, inverse mapping, and evaluation domain.

C.4 Synthetic data generation

In all simulated data synthesis, we set $m_1 = m_2 = 200$ and rank $R = 24$. We generate all synthetic benchmarks from a common form

$$M \;=\; USV^\top, \qquad U, V \in \mathbb{R}^{200\times 24},\; S \in \mathbb{R}^{24\times 24}$$

where U and V are fixed within each case and the core S is sampled independently per matrix. We generate $N = 1000$ matrices per case and save (U, V) as ground truth for subspace recovery evaluation.

Discrete Cosine Transform (DCT) basis. When a case uses a DCT basis, we build an *orthonormal* discrete cosine transform matrix $B \in \mathbb{R}^{m_1\times m_2}$ and take its first R columns. Concretely, for indices $i \in \{0, \ldots, m_1-1\}$ and $k \in \{0, \ldots, m_2-1\}$,

$$B_{k,i} \;=\; \alpha_k \cos\Big(\frac{\pi}{n}(i+\tfrac{1}{2})k\Big), \qquad \alpha_0 = \sqrt{\tfrac{1}{n}},\;\; \alpha_k = \sqrt{\tfrac{2}{n}}\;\; (k \geq 1),$$

using the corresponding $n = m_1$ or $n = m_2$. And we set:

$$U = B_{[:,\,0:R]},\; V = B_{[:,\,0:R]}.$$

The DCT-II columns correspond to cosine modes with increasing spatial frequency. Also, restricting to the first R modes biases M toward smooth and low-frequency structure.

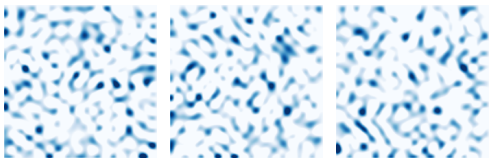


Figure 6: Case Blobs



Figure 7: Case Bands

Case Blobs. We use DCT bases for U, V and sample an i.i.d. Gaussian core

$$S \;=\; 1.5\, Z, \quad Z \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I).$$

Because energy is expressed in low-frequency DCT modes, the resulting matrices appear as smooth blob-like fields. See Figure 6.

Case Bands. We use DCT bases for U, V and sample a *diagonal* core with random mode activation, resulting in structured band patterns and varying effective rank. For each sample, we draw strengths

$$s_r \sim 1.5\,\mathcal{N}(0,1) + 3.0, \qquad r = 1, \ldots, R,$$

and independent activations $a_r \sim \mathrm{Bern}(p_r)$ with p_r increasing linearly from 0.2 to 0.9 across modes. We then set

$$S = \mathrm{diag}(s_1 a_1, \ldots, s_R a_R).$$

This yields coherent banded and diagonal structure. Also, the number of active modes (i.e., effective rank) varies across samples. See Figure 7.

Case Waves. We use DCT bases for U, V but sample a structured core that concentrates energy on a few low-frequency interactions and adds a mild "band blur" in core space. Specifically, let $k_{\max} = \max(4, \lfloor R/3 \rfloor)$ and $n_{\text{comp}} = 4$. For each sample we initialize $S = 0$, then: (i) draw (k_j, ℓ_j) uniformly from $\{0, \ldots, k_{\max}-1\}^2$ for $j = 1, \ldots, n_{\text{comp}}$ and add amplitudes $A_j \sim \mathcal{N}(0, 1.2^2)$ via $S_{k_j \ell_j} \mathrel{+}= A_j$; (ii) for band width $b = 2$, add Gaussian noise to the first two super- and sub-diagonals:

$$S_{p,p+d} \mathrel{+}= 0.15\,\xi_{p,d}, \quad S_{p+d,p} \mathrel{+}= 0.15\,\tilde{\xi}_{p,d}, \qquad d = 1, 2,$$

where $\xi_{p,d}, \tilde{\xi}_{p,d} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$. This produces smooth wave-like patterns in the generated matrices. See Figure 8.



Figure 8: Case Waves



Figure 9: Case crosshatch

Case Crosshatch. We use DCT bases for U, V and sample diagonal cores with stochastic activation (similar to Case Bands, but with different parameters), yielding strong crosshatched textures and varying effective rank. Specifically, we draw

$$s_r \sim \mathcal{N}(2.5, 1.1^2), \qquad a_r \sim \mathrm{Bern}(p_r), \qquad p_r \text{ increases linearly from } 0.15 \text{ to } 0.85,$$

and set $S = \mathrm{diag}(s_1 a_1, \ldots, s_R a_R)$. See Figure 9.

C.5 Neural Network Architecture and Computing Resources

Our generative model in Stage II parameterizes the continuous-time velocity field $v_\theta(x, t)$ on the low-dimensional core space using a lightweight time-conditioned U-Net. Stage I does not use a neural network: the shared row and column subspaces $U \in \mathrm{St}(m_1, R)$ and $V \in \mathrm{St}(m_2, R)$ are

Table 6: Neural Networks parameter counts across different datasets and rank settings. Notably, all models compared share the exact same network structure.

Setting	R	Parameters
Case A (Blobs)	24	195,169
Case B (Bands)	24	195,169
Case C (Waves)	24	195,169
Case D (Crosshatch)	24	195,169
LSPF	56	564,481
LSPF	80	614,657
Solar / Solar 2	60	564,481
Solar / Solar 2	120	564,481
Solar / Solar 2	196	614,657

optimized directly on the Stiefel manifold, and each matrix is represented in the core space by $s = \mathrm{vec}(U^\top MV) \in \mathbb{R}^{R^2}$. Therefore, only Stage II velocity field is implemented using a Neural Network.

Velocity network. We use a compact convolutional residual network with U-Net-style skip connections as the Stage-II velocity backbone. This design is an empirical architectural choice, not an assumption that cores have natural image-like locality. Reshaping the core vector provides a convenient way to impose parameter sharing and multiscale smoothing. In preliminary comparisons, same-scale MLP backbones led to higher flow-matching loss and poorer samples. We observed the same trend for the neural baselines. We therefore use this backbone for all neural methods.

Given a core vector $x \in \mathbb{R}^d$ with $d = R^2$ and time $t \in [0, 1]$, we reshape x into a single-channel square feature map and predict the velocity field $v_\theta(x, t) \in \mathbb{R}^d$ using a compact U-Net. Time is encoded by a two-layer MLP and injected into every residual block through an additive bias. Each residual block uses GroupNorm, SiLU activations, two 3×3 convolutions, and a residual skip connection. When the input and output channel dimensions differ, the skip path uses a 1×1 convolution. Downsampling is implemented by average pooling followed by a 3×3 convolution, and upsampling uses nearest-neighbor interpolation followed by a 3×3 convolution. The network includes encoder skip connections, two bottleneck residual blocks, and a final GroupNorm–SiLU–3×3 output layer. If the reshaped core image size is not divisible by the downsampling factor, we pad symmetrically before the U-Net and crop back to the original size at the output.

Architecture hyperparameters. The architecture is determined by the following quantities:

- *Core rank* R, which sets the core dimension $d = R^2$.
- *U-Net base width*: `base_channels` $= 32$.
- *Channel multipliers*: either `[1,1]` for smaller core grids or `[1,1,2]` for larger ones.
- *Residual blocks per resolution*: `num_res_blocks` $= 1$.
- *Time embedding width*: `time_emb_dim` $= 128$.

Optimization. Stage I uses QR-retracted Riemannian gradient updates (Algorithm 2), with no separate optimizer or weight decay. Stage II uses Adam with zero weight decay. Both stages use early stopping, so the iteration and epoch counts in Table 7 are configured maxima. The exact setting for each experiment is provided in the accompanying `configs/paper` files.

Sampling. To generate samples, we solve the learned ODE $\dot{x}(t) = v_\theta(x, t)$ from a Gaussian initial state in the core space using an RK4 solver with 101 time steps, and then map the generated core back to the matrix space through the learned subspaces.

Baseline architectures. All neural methods use the same lightweight U-Net backbone and are trained until convergence. This controls neural capacity and focuses the comparison on the representation: ambient-space inputs versus learned low-dimensional cores. Scaling an ambient-space model could

Table 7: Optimization settings used across the reported experiments. Each row lists all distinct values used for that hyperparameter; the exact dataset-, rank-, and missingness-specific setting for every experiment is provided in the supplementary `configs/paper` files.

Stage	Hyperparameter	Values used
Stage I	Step size $\eta_U = \eta_V$	$\{10^{-5}, 5\times10^{-5}, 10^{-4}, 5\times10^{-4}, 10^{-2}\}$
	Batch size	$\{1000, 5000\}$
	Refill iterations	$\{1, 20, 30, 35, 40, 50, 80\}$
	Maximum inner epochs	$\{50, 100, 500, 1000, 1500, 2000, 3000, 4000\}$
	Early-stopping patience	$\{10, 20, 50, 100\}$
Stage II	Optimizer	Adam with zero weight decay
	Learning rate	$\{10^{-4}, 2\times10^{-4}, 2.5\times10^{-4}, 3\times10^{-4}, 5\times10^{-4}, 10^{-3}, 2\times10^{-3}\}$
	Batch size	$\{200, 300, 1000\}$
	Maximum epochs	$\{700, 1000, 1500, 2000, 6000\}$
	Early-stopping patience	$\{200, 250, 1000, 1500\}$

improve its accuracy, but would also increase its parameter and computational cost; Table 8 reports the measured end-to-end training times.

Compute. All experiments were conducted on a single NVIDIA V100 GPU. Because the velocity network is shared across methods, Figure 3 in the main text uses the learned-dimension ratio only as a proxy for *Stage II CNF training cost*. However, CoreFlow is a two-stage method, so its full training cost also includes Stage I subspace learning. Table 8 reports end-to-end training times.

Table 8: Measured total training time on a single NVIDIA V100 GPU. For CoreFlow, the total includes both Stage I subspace training and Stage II core-space CNF training. For MissDiff, MissFlow, and CSDM, we report the actual generative model's training time until validation-loss convergence.

Dataset	Method	ρ	p_{miss}	Stage I (min)	Stage II to convergence (min)	Total (min)
Solar / Solar 2	CoreFlow	9%	0%	4.42	4.05	8.47
	CoreFlow	9%	20%	8.85	4.05	**12.90**
	CoreFlow	9%	40%	8.85	4.05	**12.90**
	CoreFlow	36%	0%	3.17	13.23	**16.40**
	CoreFlow	36%	20%	4.76	13.23	**17.99**
	CoreFlow	36%	40%	6.35	13.23	**19.58**
	MissDiff / MissFlow	100%	all	–	37.70	37.70
	CSDM	9%	all	–	8.29	**8.29**
	CSDM	36%	all	–	23.81	23.81
LSPF	CoreFlow	49%	0%	11.36	30.20	**41.56**
	CoreFlow	49%	20%	22.73	30.20	**52.93**
	CoreFlow	49%	40%	22.73	30.20	**52.93**
	MissDiff / MissFlow	100%	all	–	83.20	83.20

On Solar and Solar 2, the measured Stage-I training speed is 10.5 epochs/s for $R = 120$ ($\rho = 36\%$) and 11.3 epochs/s for $R = 60$ ($\rho = 9\%$). Based on the actual number of Stage-I epochs required under each compression and missingness setting, this corresponds to Stage-I costs of 3.17/4.76/6.35 minutes at $p_{\text{miss}} = 0/20/40\%$ for $\rho = 36\%$, and 4.42/8.85/8.85 minutes for $\rho = 9\%$. Adding the corresponding measured Stage-II times then gives the total CoreFlow training time.

On LSPF, Stage I runs at 2.5 epochs/s on average and takes 11.36/22.73/22.73 minutes for $p_{\text{miss}} = 0/20/40\%$. Adding the 30.20-minute Stage-II training gives total times of 41.56/52.93/52.93 minutes. For the other baselines, Table 8 reports their measured time to convergence. For the simulations, Stage I is negligible at $p_{\text{miss}} = 0\%$ because the spectral initializer is used directly and takes about 8.62 minutes under missingness (1500 epochs at 2.9 epochs/s).

C.6 MissFlow, SMG-Core, and CSDM implementation details

MissFlow. Let $x_i^{\text{obs}} = \text{vec}(M_i^{\text{obs}})$, with unobserved entries set to zero, and let m_i be the corresponding binary mask. For $z_i \sim \mathcal{N}(0, I)$ and $t \sim \text{Unif}(0, 1)$, MissFlow uses $x_t = (1-t)x_i^{\text{obs}} + tz_i$

and minimizes

$$\mathcal{L}_{\text{MissFlow}}(\theta) = \mathbb{E}\left[\frac{\|m_i \odot [v_\theta(x_t, t) - (z_i - x_i^{\text{obs}})]\|_2^2}{\max\{\|m_i\|_1, 1\}}\right].$$

Thus, only observed entries contribute to training. To generate a complete matrix, we solve the learned ambient-space ODE backward from $z \sim \mathcal{N}(0, I)$ over all entries.

SMG-Core. The original BayeSMG framework is designed for *matrix completion*: it performs conditional inference for a partially observed matrix under an SMG prior, using the observed entries to infer posterior uncertainty about the unobserved ones. Our task here is different: we study *unconditional generation* of new matrices from a training distribution. Therefore, the original BayeSMG procedure is not directly applicable as a generative baseline, since at generation time there is no partially observed test matrix to condition on. To obtain a controlled low-dimensional baseline for our setting, we fix the shared subspaces learned in Stage I and use the BayeSMG Gaussian variance-prior structure only to model the core distribution. This yields SMG-Core, which isolates the effect of replacing CoreFlow's nonlinear core generator by a Gaussian core model under the same shared geometry.

Normal–Inverse–Wishart prior. We place a conjugate Normal–Inverse–Wishart (NIW) prior on the unknown mean $\mu \in \mathbb{R}^d$ and covariance $\Sigma \in \mathbb{R}^{d\times d}$:

$$\Sigma \sim \text{InvWishart}(\nu_0, \Psi_0), \qquad \mu \mid \Sigma \sim \mathcal{N}\left(\mu_0, \frac{1}{\kappa_0}\Sigma\right),$$

where $\kappa_0 > 0$, $\nu_0 > d - 1$, and $\Psi_0 \succ 0$.

Posterior. Given data $\{x_i\}_{i=1}^N$, define the empirical mean $\bar{x} = \frac{1}{N}\sum_i x_i$ and scatter matrix

$$S = \sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^\top.$$

The NIW posterior is again NIW with parameters

$$\kappa_N = \kappa_0 + N, \qquad \mu_N = \frac{\kappa_0\mu_0 + N\bar{x}}{\kappa_N},$$
$$\nu_N = \nu_0 + N, \qquad \Psi_N = \Psi_0 + S + \frac{\kappa_0 N}{\kappa_N}(\bar{x} - \mu_0)(\bar{x} - \mu_0)^\top.$$

Posterior predictive sampling. A Bayesian posterior predictive sample can be drawn in either of the following equivalent ways:

(i) Two-stage sampling:

$$\Sigma \sim \text{InvWishart}(\nu_N, \Psi_N), \qquad \mu \sim \mathcal{N}\left(\mu_N, \frac{1}{\kappa_N}\Sigma\right), \qquad x_{\text{new}} \sim \mathcal{N}(\mu, \Sigma).$$

(ii) Collapsed predictive distribution: Integrating out (μ, Σ) yields a multivariate Student-t distribution:

$$x_{\text{new}} \sim t_{\nu_N - d + 1}\left(\mu_N, \frac{\kappa_N + 1}{\kappa_N(\nu_N - d + 1)}\Psi_N\right).$$

We then reshape x_{new} into a core matrix and decode:

$$T_{\text{new}} = \text{unvec}(x_{\text{new}}), \qquad M_{\text{new}} = U\, T_{\text{new}}\, V^\top.$$

Missingness handling. CSDM uses empirical entrywise-mean filling before compression. Specifically, for each entry location (j, k), missing values are replaced by the average of all observed training values at that location. The completed matrices are then used for CSDM compression. SMG-Core instead uses the same fitted subspaces and core representations produced by CoreFlow's Stage I, replacing the CNF with the NIW model above.

D ADDITIONAL RESULTS

This appendix provides additional qualitative and quantitative results that complement the main paper.

D.1 QUALITATIVE CROSS-METHOD SAMPLE COMPARISONS

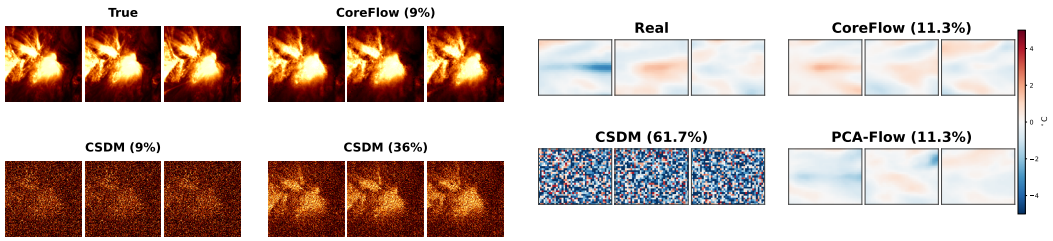


Figure 10: *Left:* Comparisons of generated samples on Solar Flare with $p_{\text{miss}} = 0\%$ (training on complete matrices). CoreFlow is trained at $\rho = \mathbf{9}\%$ of the original dimension, and is compared against CSDM trained at $\rho \in \{9\%, 36\%\}$. *Right:* ERSSTv5 with complete training matrices. CoreFlow and PCA-Flow use $\rho = 11.33\%$; the visualization additionally shows CSDM at $\rho = 61.68\%$. The fixed real and generated fields are unpaired and use a common color scale.

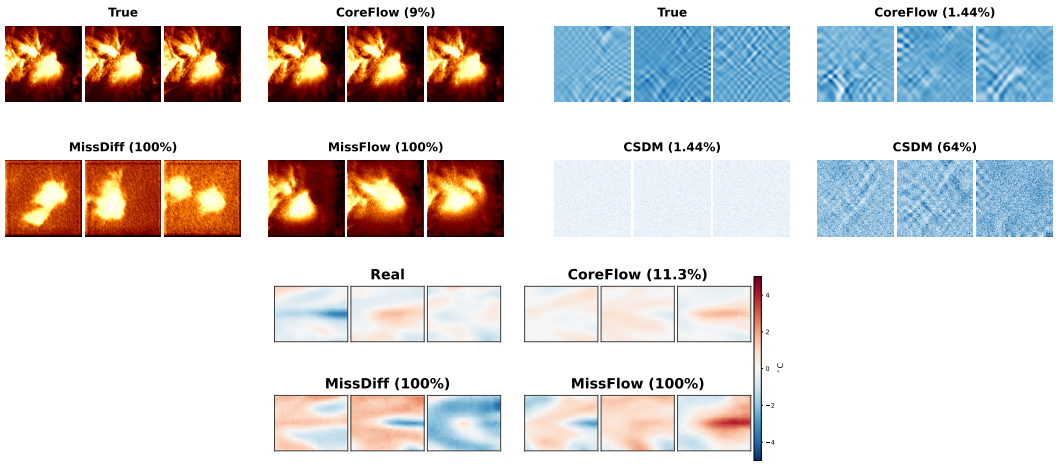


Figure 11: Comparison of generated samples across methods. *Top Left:* Solar Flare with $p_{\text{miss}} = 40\%$. CoreFlow is trained at $\rho = 9\%$ of the original dimension, and is compared against MissDiff and MissFlow trained at $\rho = 100\%$. *Top Right:* Case Waves with $p_{\text{miss}} = 0\%$ (complete training matrices). CoreFlow is trained at $\rho = 1.44\%$ of the original dimension, and is compared against CSDM trained at $\rho \in \{1.44\%, 64\%\}$. *Bottom:* ERSSTv5 with $p_{\text{miss}} = 40\%$ i.i.d. missing entries. CoreFlow uses $\rho = 11.33\%$ and is compared against full-space MissDiff and MissFlow ($\rho = 100\%$); the fixed real and generated fields are unpaired and use a common color scale.

Figures 10 and 11 visualize representative generated matrices under both complete training ($p_{\text{miss}} = 0\%$) and missing-entry training. For ERSSTv5, Figure 10 *Right* compares CoreFlow with CSDM and the matched-dimensional PCA-Flow ablation under complete training: CoreFlow and PCA-Flow retain coherent large-scale fields, whereas CSDM loses spatial coherence. Figure 11 *Bottom* compares CoreFlow with ambient-space MissDiff and MissFlow under 40% i.i.d. missingness. On Solar Flare with complete training (Figure 10), CoreFlow at $\rho = 9\%$ reproduces the salient flare morphology, matching the visual statistics of real samples. In contrast, CSDM at the same compression ($\rho = 9\%$) loses the global structure. This gap is primarily due to information loss induced by CSDM's compression operator: its compressed measurements may not preserve spatial locality and can discard geometry information at aggressive ρ.

Under missing entries (Figure 11 *Top Left*), CoreFlow maintains coherent spatial patterns and realistic intensity variation despite operating in a reduced core space, while full-space baselines (MissDiff and MissFlow) fail to learn the matrix distribution in the 200×200 ambient dimension with limited samples. Finally, in the Waves simulation (Figure 11, *Top Right*), CoreFlow preserves the characteristic smooth oscillatory textures even at extreme compression ($\rho = 1.44\%$).

D.2 Real data: robustness to missing entries

Table 9: Comparisons to baselines with *missingness handling* on real data (Frobenius-norm distribution shift: FrobMeanDiff and FrobStdDiff; lower is better). Column headers denote missing-entry rate. Parentheses report each method's learned-dimension ratio (CoreFlow: ρ=36% for Solar and ρ=49% for LSPF; all ambient-space baselines use ρ=100%). Solar is a few-sample regime (300 samples) and LSPF a standard regime (8760 samples).

	Solar			**LSPF**		
p_{miss}	0%	20%	40%	0%	20%	40%
MissDiff ($\rho = 100\%$)	$(0.85_{\pm.13}, 0.62_{\pm.11})$	$(37.4_{\pm.19}, 0.72_{\pm.24})$	$(71.2_{\pm.30}, 0.91_{\pm.26})$	$(\mathbf{0.21}_{\pm.11}, 0.18_{\pm.08})$	$(13.5_{\pm.17}, 0.61_{\pm.16})$	$(16.7_{\pm.21}, 0.82_{\pm.19})$
MissFlow ($\rho = 100\%$)	$(\mathbf{0.13}_{\pm.10}, 0.53_{\pm.10})$	$(21.2_{\pm.13}, 0.32_{\pm.11})$	$(54.3_{\pm.14}, 0.41_{\pm.13})$	$(\mathbf{0.20}_{\pm.12}, \mathbf{0.19}_{\pm.09})$	$(11.9_{\pm.15}, 0.44_{\pm.16})$	$(15.9_{\pm.11}, 0.72_{\pm.12})$
CoreFlow ($\rho = 36\%/49\%$)	$(0.56_{\pm.28}, \mathbf{0.39}_{\pm.21})$	$(\mathbf{0.30}_{\pm.14}, \mathbf{0.45}_{\pm.09})$	$(\mathbf{0.84}_{\pm.15}, \mathbf{0.40}_{\pm.08})$	$(0.24_{\pm.09}, 0.12_{\pm.06})$	$(\mathbf{0.67}_{\pm.09}, \mathbf{0.20}_{\pm.06})$	$(\mathbf{0.98}_{\pm.08}, \mathbf{0.44}_{\pm.05})$

Table 10: Comparisons to baselines with *missingness handling* on real data (entry-wise moment discrepancies: AbsEntryMeanDiff and AbsEntryStdDiff; lower is better). Column headers denote missing-entry rate. Parentheses report each method's learned-dimension ratio (CoreFlow: ρ=36% for Solar and ρ=49% for LSPF; all ambient-space baselines use ρ=100%). Solar is a few-sample regime (300 samples) and LSPF a standard regime (8760 samples). All values are in $\times 10^{-2}$.

	Solar			**LSPF**		
p_{miss}	0%	20%	40%	0%	20%	40%
MissDiff ($\rho = 100\%$)	$(12.3_{\pm.28}, 5.91_{\pm.29})$	$(29.3_{\pm.20}, 6.00_{\pm.24})$	$(39.1_{\pm.25}, 6.96_{\pm.28})$	$(\mathbf{1.69}_{\pm.22}, 1.74_{\pm.20})$	$(15.6_{\pm.18}, 2.53_{\pm.20})$	$(21.7_{\pm.14}, 3.07_{\pm.19})$
MissFlow ($\rho = 100\%$)	$(7.16_{\pm.39}, 5.41_{\pm.38})$	$(16.4_{\pm.18}, 5.69_{\pm.25})$	$(33.4_{\pm.19}, 6.68_{\pm.27})$	$(1.71_{\pm.35}, \mathbf{1.70}_{\pm.23})$	$(14.8_{\pm.17}, 2.65_{\pm.28})$	$(20.4_{\pm.12}, 3.23_{\pm.21})$
CoreFlow ($\rho = 36\%/49\%$)	$(\mathbf{1.02}_{\pm.17}, \mathbf{0.92}_{\pm.33})$	$(\mathbf{0.87}_{\pm.08}, \mathbf{1.19}_{\pm.13})$	$(\mathbf{0.97}_{\pm.03}, \mathbf{1.32}_{\pm.04})$	$(1.87_{\pm.11}, 1.89_{\pm.12})$	$(\mathbf{1.69}_{\pm.10}, \mathbf{1.96}_{\pm.09})$	$(\mathbf{1.62}_{\pm.07}, \mathbf{2.38}_{\pm.05})$

Tables 9 and 10 report distribution-shift metrics on two real datasets (Solar and LSPF) under $p_{\text{miss}} \in \{0, 20, 40\}\%$. On Solar (few-data regime), full-space baselines can be competitive at $p_{\text{miss}} = 0\%$ but degrade sharply as missingness increases, with FrobMeanDiff growing by orders of magnitude at 20%. On LSPF (standard regime), all methods are closer at $p_{\text{miss}} = 0\%$, but CoreFlow achieves clearly smaller shifts once missingness is introduced (20%/40%).

D.3 Sensitivity to training randomness

We additionally train CoreFlow independently with seeds 42, 123, and 2026. Table 11 reports the singular-value discrepancy for complete training data and 40% missingness. The final column gives the mean and sample standard deviation across the three trained models, rather than across repeated samples from one model.

D.4 Robustness across missingness patterns

For each dataset, we keep the average missing rate shown in parentheses fixed and vary only the mask structure. Under the i.i.d. mask, every entry is dropped independently at the same rate. Under the heterogeneous mask, each matrix draws its own missing rate and its entries are then dropped independently at that rate, while the dataset-wide average is unchanged. Block-10 and block-20 masks instead remove contiguous square blocks with side length 10 and 20, respectively.

Table 11: CoreFlow across three independent training seeds. Entries are SVRelL2 in $\times 10^{-2}$ (lower is better). Solar/Solar 2 use $\rho = 9\%$, LSPF uses $\rho = 49\%$, and ERSSTv5 uses $\rho = 11.33\%$. Solar/Solar 2 and LSPF use complete training matrices as the reference for this diagnostic; ERSSTv5 uses its held-out test set.

Dataset	p_{miss}	**Seed 42**	**Seed 123**	**Seed 2026**	**Mean $\pm$ std.**
Solar	0%	0.855	1.164	0.966	$0.995_{\pm 0.156}$
Solar	40%	0.814	0.951	1.032	$0.933_{\pm 0.110}$
Solar 2	0%	0.633	0.749	1.219	$0.867_{\pm 0.310}$
Solar 2	40%	0.607	0.708	1.092	$0.802_{\pm 0.256}$
LSPF	0%	2.753	2.904	2.880	$2.846_{\pm 0.081}$
LSPF	40%	4.801	4.962	5.123	$4.962_{\pm 0.161}$
ERSSTv5	0%	21.835	20.925	23.307	$22.022_{\pm 1.202}$
ERSSTv5	40%	26.708	22.505	25.100	$24.771_{\pm 2.121}$

Table 12: CoreFlow under different missingness patterns. Parentheses give the average missing rate; Solar results use $\rho = 9\%$. Entries are mean SVRelL2 in $\times 10^{-2}$ (lower is better).

Dataset	**i.i.d.**	**Heterogeneous**	**Block-10**	**Block-20**
Blobs (20%)	0.63	1.15	1.16	4.58
Bands (40%)	9.09	8.14	19.12	26.58
Waves (20%)	5.25	4.55	6.76	6.50
Crosshatch (20%)	4.33	5.22	5.91	7.42
Solar (20%)	0.93	0.78	1.14	1.31
Solar 2 (20%)	0.64	0.59	0.75	0.94

The heterogeneous-rate results stay close to their i.i.d. counterparts, showing that CoreFlow does not require every training matrix to have the same observation rate. It also remains accurate for most block-mask settings, but contiguous gaps can be substantially harder, especially for Bands. The missing rate alone therefore does not determine performance: recovery depends on whether the observed entries still reveal the shared matrix directions. If a mask systematically hides those directions, they cannot be recovered without stronger assumptions.

D.5 Simulations: distribution matching under controlled structure

Table 13: Comparisons to *low-dimensional* generative baselines on Blobs and Bands simulations (Frobenius-norm distribution shift: FrobMeanDiff and FrobStdDiff; lower is better). Column headers denote missing-entry rate. Parentheses report each method's learned-dimension ratio.

	Blobs			**Bands**		
p_{miss}	0%	20%	40%	0%	20%	40%
CSDM ($\rho = 1.44\%$)	$(130_{\pm 6.9}, 0.70_{\pm .38})$	$(155_{\pm 5.1}, 0.76_{\pm .36})$	$(178_{\pm 4.1}, 0.84_{\pm .31})$	$(40.5_{\pm 1.3}, 1.61_{\pm .18})$	$(48.7_{\pm 2.3}, 1.61_{\pm .13})$	$(55.9_{\pm 1.8}, 1.62_{\pm .18})$
CSDM ($\rho = 36\%$)	$(76.1_{\pm 5.5}, 0.69_{\pm .44})$	$(93.4_{\pm 4.3}, 0.72_{\pm .34})$	$(108_{\pm 3.9}, 0.76_{\pm .34})$	$(23.4_{\pm 1.5}, 1.57_{\pm .33})$	$(28.9_{\pm 1.8}, 1.56_{\pm .22})$	$(33.7_{\pm 2.4}, 1.56_{\pm .29})$
CSDM ($\rho = 64\%$)	$(50.6_{\pm 5.1}, 0.71_{\pm .51})$	$(64.1_{\pm 4.4}, 0.74_{\pm .25})$	$(75.8_{\pm 5.3}, 0.77_{\pm .24})$	$(15.6_{\pm 1.1}, 1.56_{\pm .29})$	$(19.9_{\pm 1.3}, 1.55_{\pm .37})$	$(23.6_{\pm 1.4}, 1.53_{\pm .34})$
SMG-Core ($\rho = 1.44\%$)	$(70.2_{\pm 6.0}, 2.30_{\pm .94})$	$(79.7_{\pm 6.3}, 2.15_{\pm .90})$	$(80.2_{\pm 6.2}, 2.03_{\pm .88})$	$(14.2_{\pm .80}, 2.28_{\pm .34})$	$(16.3_{\pm 0.8}, 2.14_{\pm .37})$	$(19.9_{\pm 1.0}, 2.37_{\pm .36})$
CoreFlow ($\rho = 1.44\%$)	$(\mathbf{0.09}_{\pm .06}, \mathbf{0.20}_{\pm .05})$	$(\mathbf{0.19}_{\pm .06}, \mathbf{0.12}_{\pm .06})$	$(\mathbf{0.44}_{\pm .07}, \mathbf{0.15}_{\pm .05})$	$(\mathbf{0.08}_{\pm .06}, \mathbf{0.12}_{\pm .06})$	$(\mathbf{0.85}_{\pm .08}, \mathbf{0.23}_{\pm .06})$	$(\mathbf{0.98}_{\pm .07}, \mathbf{0.47}_{\pm .04})$

Tables 13 and 14 report Frobenius-norm distribution shift on synthetic families; Tables 15 and 16 report entry-wise moment discrepancies. Across Blobs/Bands/Waves/Crosshatch, CoreFlow at $\rho = 1.44\%$ achieves near-zero discrepancies and remains small as p_{miss} increases. In contrast, low-dimensional generative baselines exhibit substantially larger shifts.

D.6 Compression-matched comparison on Solar Flare

Tables 17 and 18 directly compare CoreFlow and low-dimensional baselines at the *same* compression ratio $\rho = 9\%$ on Solar and Solar 2, isolating the contribution of the modeling pipeline beyond

Table 14: Comparisons to *low-dimensional* generative baselines on Waves and Crosshatch simulations (Frobenius-norm distribution shift: FrobMeanDiff and FrobStdDiff; lower is better). Column headers denote missing-entry rate. Parentheses report learned-dimension ratio.

	Waves			**Crosshatch**		
p_{miss}	0%	20%	40%	0%	20%	40%
CSDM ($\rho = 1.44\%$)	$(83.5_{\pm 5.6}, 9.84_{\pm 1.3})$	$(85.4_{\pm 5.0}, 10.1_{\pm 1.3})$	$(87.4_{\pm 4.3}, 10.3_{\pm 1.2})$	$(62.4_{\pm 3.4}, 7.67_{\pm 1.2})$	$(63.8_{\pm 3.0}, 7.82_{\pm 1.1})$	$(65.31_{\pm 2.5}, 8.00_{\pm 1.2})$
CSDM ($\rho = 36\%$)	$(34.8_{\pm 2.2}, 4.23_{\pm .96})$	$(41.8_{\pm 2.9}, 5.03_{\pm .85})$	$(49.6_{\pm 2.7}, 5.94_{\pm .72})$	$(25.6_{\pm 1.3}, 3.32_{\pm .64})$	$(30.9_{\pm 1.2}, 3.94_{\pm .66})$	$(36.8_{\pm 1.2}, 4.63_{\pm .56})$
CSDM ($\rho = 64\%$)	$(17.8_{\pm 2.7}, 2.22_{\pm .32})$	$(26.5_{\pm 3.4}, 3.23_{\pm .51})$	$(36.4_{\pm 2.1}, 4.38_{\pm .72})$	$(12.88_{\pm 1.6}, 1.79_{\pm .45})$	$(19.46_{\pm 1.5}, 2.56_{\pm .43})$	$(26.9_{\pm 1.6}, 3.45_{\pm .51})$
SMG-Core ($\rho = 1.44\%$)	$(11.6_{\pm 1.3}, 4.10_{\pm .47})$	$(14.8_{\pm 1.9}, 3.79_{\pm .29})$	$(15.7_{\pm 2.0}, 4.97_{\pm .87})$	$(5.94_{\pm .90}, 2.14_{\pm .40})$	$(7.28_{\pm 1.1}, 2.31_{\pm .44})$	$(9.75_{\pm 1.0}, 2.76_{\pm .48})$
CoreFlow ($\rho = 1.44\%$)	$(\mathbf{0.55}_{\pm .38}, \mathbf{0.53}_{\pm .32})$	$(\mathbf{0.47}_{\pm .38}, \mathbf{2.87}_{\pm .37})$	$(\mathbf{3.53}_{\pm .63}, \mathbf{0.87}_{\pm .39})$	$(\mathbf{0.96}_{\pm .31}, \mathbf{0.75}_{\pm .30})$	$(\mathbf{1.07}_{\pm .39}, \mathbf{0.95}_{\pm .38})$	$(\mathbf{1.60}_{\pm .48}, \mathbf{0.96}_{\pm .36})$

Table 15: Comparisons to *low-dimensional* generative baselines on Blobs and Bands simulations (entry-wise moment discrepancies: AbsEntryMeanDiff and AbsEntryStdDiff; lower is better). Column headers denote missing-entry rate. Parentheses report each method's learned-dimension ratio. All values are in $\times 10^{-2}$.

	Blobs			**Bands**		
p_{miss}	0%	20%	40%	0%	20%	40%
CSDM ($\rho = 1.44\%$)	$(63.6_{\pm 2.0}, 34.1_{\pm 1.8})$	$(84.8_{\pm 2.4}, 24.3_{\pm 1.3})$	$(106_{\pm 2.2}, 15.5_{\pm 1.2})$	$(20.5_{\pm 1.2}, 12.2_{\pm .14})$	$(27.2_{\pm 1.1}, 9.10_{\pm .09})$	$(34.0_{\pm 1.2}, 3.67_{\pm .12})$
CSDM ($\rho = 36\%$)	$(37.5_{\pm 1.9}, 17.2_{\pm 1.1})$	$(50.0_{\pm 1.7}, 12.3_{\pm 1.3})$	$(62.6_{\pm 1.5}, 6.25_{\pm 1.6})$	$(12.3_{\pm 1.2}, 6.37_{\pm .18})$	$(16.2_{\pm 1.1}, 4.70_{\pm .19})$	$(20.2_{\pm 1.2}, 1.53_{\pm .20})$
CSDM ($\rho = 64\%$)	$(26.9_{\pm .75}, 11.0_{\pm 1.4})$	$(35.8_{\pm .81}, 7.87_{\pm 1.3})$	$(37.9_{\pm .98}, 8.57_{\pm 1.0})$	$(8.98_{\pm .91}, 4.21_{\pm .08})$	$(11.7_{\pm .04}, 3.12_{\pm .13})$	$(14.5_{\pm 1.3}, 0.87_{\pm .05})$
SMG-Core ($\rho = 1.44\%$)	$(34.9_{\pm 1.2}, 3.92_{\pm .82})$	$(36.5_{\pm 1.2}, 3.84_{\pm .86})$	$(41.3_{\pm 1.6}, 4.02_{\pm 1.0})$	$(8.26_{\pm .57}, 1.02_{\pm .07})$	$(10.3_{\pm .63}, 1.74_{\pm .11})$	$(12.9_{\pm .89}, 1.55_{\pm .12})$
CoreFlow ($\rho = 1.44\%$)	$(\mathbf{0.98}_{\pm .04}, \mathbf{0.74}_{\pm .02})$	$(\mathbf{0.99}_{\pm .03}, \mathbf{0.72}_{\pm .01})$	$(\mathbf{1.01}_{\pm .04}, \mathbf{0.72}_{\pm .02})$	$(\mathbf{0.29}_{\pm .03}, \mathbf{0.19}_{\pm .01})$	$(\mathbf{0.45}_{\pm .03}, \mathbf{0.67}_{\pm .03})$	$(\mathbf{0.52}_{\pm .02}, \mathbf{0.86}_{\pm .02})$

Table 16: Comparisons to *low-dimensional* generative baselines on Waves and Crosshatch simulations (entry-wise moment discrepancies: AbsEntryMeanDiff and AbsEntryStdDiff; lower is better). Column headers denote missing-entry rate. Parentheses report each method's learned-dimension ratio. All values are in $\times 10^{-2}$.

	Waves			**Crosshatch**		
p_{miss}	0%	20%	40%	0%	20%	40%
CSDM ($\rho = 1.44\%$)	$(44.1_{\pm 1.8}, 8.22_{\pm .75})$	$(45.2_{\pm 1.4}, 7.14_{\pm .54})$	$(46.2_{\pm 1.0}, 6.91_{\pm .45})$	$(31.9_{\pm 1.1}, 7.97_{\pm .17})$	$(32.7_{\pm 1.0}, 7.21_{\pm .15})$	$(33.5_{\pm 1.5}, 7.00_{\pm .16})$
CSDM ($\rho = 36\%$)	$(26.0_{\pm .86}, 3.78_{\pm .68})$	$(29.2_{\pm 1.1}, 6.40_{\pm .75})$	$(33.7_{\pm .99}, 7.96_{\pm .81})$	$(18.7_{\pm 1.1}, 10.9_{\pm .36})$	$(20.9_{\pm 1.2}, 4.72_{\pm .11})$	$(24.2_{\pm 1.2}, 5.67_{\pm .15})$
CSDM ($\rho = 64\%$)	$(18.5_{\pm .79}, 2.36_{\pm .53})$	$(21.9_{\pm 1.1}, 8.06_{\pm .92})$	$(27.8_{\pm 1.1}, 10.6_{\pm .95})$	$(13.2_{\pm 1.4}, 2.23_{\pm .08})$	$(15.7_{\pm 1.4}, 5.27_{\pm .13})$	$(20.0_{\pm 1.4}, 6.84_{\pm .15})$
SMG-Core ($\rho = 1.44\%$)	$(13.1_{\pm .40}, 1.70_{\pm .21})$	$(17.6_{\pm .47}, 1.99_{\pm .28})$	$(20.6_{\pm .57}, 2.04_{\pm .39})$	$(12.7_{\pm .85}, 1.53_{\pm .05})$	$(14.3_{\pm .94}, 1.87_{\pm .11})$	$(17.3_{\pm 1.2}, 1.84_{\pm .19})$
CoreFlow ($\rho = 1.44\%$)	$(\mathbf{0.97}_{\pm .09}, \mathbf{1.11}_{\pm .13})$	$(\mathbf{0.82}_{\pm .09}, \mathbf{2.65}_{\pm .13})$	$(\mathbf{1.59}_{\pm .28}, \mathbf{1.37}_{\pm .14})$	$(\mathbf{0.93}_{\pm .22}, \mathbf{0.74}_{\pm .07})$	$(\mathbf{0.85}_{\pm .23}, \mathbf{0.96}_{\pm .09})$	$(\mathbf{0.89}_{\pm .14}, \mathbf{0.97}_{\pm .07})$

Table 17: Comparisons to *low-dimensional* baselines on Solar, Solar 2, and ERSSTv5 (Frobenius-norm distribution shift: FrobMeanDiff and FrobStdDiff; lower is better). Column headers denote missing-entry rate. Parentheses report each method's learned-dimension ratio.

	Solar			**Solar 2**			**ERSSTv5**		
p_{miss}	0%	20%	40%	0%	20%	40%	0%	20%	40%
CSDM ($\rho = 9\%$)	$(52.9_{\pm 1.8}, 0.98_{\pm .30})$	$(56.4_{\pm 1.6}, 1.03_{\pm .40})$	$(60.3_{\pm 1.5}, 1.08_{\pm .62})$	$(65.9_{\pm 3.8}, 1.81_{\pm 1.0})$	$(70.2_{\pm 3.3}, 1.92_{\pm 1.1})$	$(75.1_{\pm 3.1}, 2.04_{\pm 1.1})$	$(69.7_{\pm 4.1}, 1.99_{\pm .44})$	$(73.7_{\pm 3.9}, 2.07_{\pm .58})$	$(79.6_{\pm 4.8}, 2.53_{\pm .50})$
CSDM ($\rho = 36\%$)	$(26.6_{\pm 3.5}, 0.52_{\pm .24})$	$(32.8_{\pm 3.3}, 0.62_{\pm .25})$	$(39.9_{\pm 3.0}, 0.72_{\pm .29})$	$(33.7_{\pm 6.6}, 1.06_{\pm .48})$	$(41.4_{\pm 6.1}, 1.25_{\pm .47})$	$(50.2_{\pm 5.5}, 1.43_{\pm .58})$	$(36.5_{\pm 2.2}, 1.25_{\pm .21})$	$(42.3_{\pm 2.8}, 1.46_{\pm .30})$	$(49.1_{\pm 2.8}, 1.84_{\pm .35})$
SMG-Core ($\rho = 9\%/11.33\%$)	$(2.15_{\pm .87}, 0.20_{\pm .06})$	$(3.02_{\pm .90}, 0.28_{\pm .08})$	$(3.79_{\pm 1.0}, 0.36_{\pm .18})$	$(1.03_{\pm .31}, 1.09_{\pm .32})$	$(1.92_{\pm .34}, 1.30_{\pm .37})$	$(2.03_{\pm .45}, 1.54_{\pm .58})$	$(1.08_{\pm .28}, 0.34_{\pm .23})$	$(1.79_{\pm .30}, 0.61_{\pm .19})$	$(1.82_{\pm .34}, 0.66_{\pm .20})$
CoreFlow ($\rho = 9\%/11.33\%$)	$(\mathbf{0.33}_{\pm .19}, \mathbf{0.14}_{\pm .11})$	$(\mathbf{0.53}_{\pm .08}, \mathbf{0.05}_{\pm .04})$	$(\mathbf{0.65}_{\pm .10}, \mathbf{0.11}_{\pm .06})$	$(\mathbf{0.44}_{\pm .29}, \mathbf{0.93}_{\pm .22})$	$(\mathbf{0.61}_{\pm .13}, \mathbf{0.40}_{\pm .11})$	$(\mathbf{0.70}_{\pm .13}, \mathbf{0.74}_{\pm .12})$	$(1.03_{\pm .28}, 0.40_{\pm .24})$	$(1.27_{\pm .21}, 0.40_{\pm .25})$	$(1.28_{\pm .18}, 0.36_{\pm .26})$

Table 18: Comparisons to *low-dimensional* baselines on Solar, Solar 2, and ERSSTv5 (entry-wise moment discrepancies: AbsEntryMeanDiff and AbsEntryStdDiff; lower is better). Column headers denote missing-entry rate. Parentheses report each method's learned-dimension ratio. All values are in $\times 10^{-2}$.

	Solar			**Solar 2**			**ERSSTv5**		
p_{miss}	0%	20%	40%	0%	20%	40%	0%	20%	40%
CSDM ($\rho = 9\%$)	$(26.8_{\pm 1.3}, 2.91_{\pm .40})$	$(28.1_{\pm 1.4}, 4.18_{\pm .22})$	$(29.6_{\pm 1.4}, 4.82_{\pm .29})$	$(34.9_{\pm 1.7}, 2.21_{\pm .38})$	$(36.6_{\pm 1.9}, 4.64_{\pm .45})$	$(38.6_{\pm 1.6}, 5.74_{\pm .57})$	$(76.3_{\pm 2.2}, 3.79_{\pm .42})$	$(88.2_{\pm 2.9}, 4.93_{\pm .50})$	$(91.5_{\pm 3.7}, 5.68_{\pm .57})$
CSDM ($\rho = 36\%$)	$(19.6_{\pm 1.4}, 2.30_{\pm .31})$	$(21.4_{\pm 1.5}, 6.53_{\pm .35})$	$(23.9_{\pm 1.6}, 8.41_{\pm .47})$	$(25.3_{\pm 1.6}, 1.76_{\pm .50})$	$(27.9_{\pm 1.1}, 8.55_{\pm .71})$	$(31.4_{\pm 1.1}, 11.4_{\pm .92})$	$(53.0_{\pm 1.1}, 2.04_{\pm .23})$	$(58.9_{\pm 1.9}, 2.26_{\pm .29})$	$(60.3_{\pm 2.0}, 2.97_{\pm .28})$
SMG-Core ($\rho = 9\%/11.33\%$)	$(1.54_{\pm .49}, 1.44_{\pm .26})$	$(1.76_{\pm .46}, 1.47_{\pm .21})$	$(1.81_{\pm .51}, 1.36_{\pm .34})$	$(1.30_{\pm .15}, 1.27_{\pm .19})$	$(1.34_{\pm .16}, 1.32_{\pm .19})$	$(1.48_{\pm .19}, 1.38_{\pm .20})$	$(9.69_{\pm .82}, 1.86_{\pm .49})$	$(10.3_{\pm .92}, 1.99_{\pm .57})$	$(11.0_{\pm .90}, 2.34_{\pm .53})$
CoreFlow ($\rho = 9\%/11.33\%$)	$(\mathbf{0.77}_{\pm .21}, \mathbf{0.83}_{\pm .10})$	$(\mathbf{0.80}_{\pm .08}, \mathbf{0.86}_{\pm .03})$	$(\mathbf{0.91}_{\pm .10}, \mathbf{0.93}_{\pm .15})$	$(\mathbf{0.86}_{\pm .05}, \mathbf{0.71}_{\pm .06})$	$(\mathbf{0.98}_{\pm .07}, \mathbf{0.69}_{\pm .04})$	$(\mathbf{1.05}_{\pm .09}, \mathbf{0.78}_{\pm .04})$	$(9.51_{\pm .70}, 1.93_{\pm .49})$	$(9.56_{\pm .65}, 1.69_{\pm .32})$	$(9.49_{\pm .78}, 1.65_{\pm .30})$

dimensionality reduction. At matched ρ, CoreFlow dramatically reduces both Frobenius-norm distribution shift (Table 17) and entry-wise moment discrepancies (Table 18) across all missing rates.

D.7 STAGE-I SUBSPACE RECOVERY DIAGNOSTIC VIA PRINCIPAL ANGLES

Finally, Table 19 shows that CoreFlow recovers the ground-truth row/column subspaces accurately, with small mean and maximum principal angles even at $p_{\text{miss}} = 40\%$.

Table 19: Simulation: **CoreFlow** principal-angle discrepancy in degrees. Each entry reports $(\overline{\theta}_U/\theta_{U,\max},\ \overline{\theta}_V/\theta_{V,\max})$. All values lie in $[0, 90]$; lower is better. Columns denote the missing-entry rate p_{miss}.

	p_{miss}		
Case	0%	20%	40%
Blobs ($\rho = 1.44\%$)	(0.00/0.00, 0.00/0.00)	(0.01/0.07, 0.04/0.13)	(0.04/0.14, 0.03/0.15)
Bands ($\rho = 1.44\%$)	(0.00/0.00, 0.00/0.00)	(0.28/0.74, 0.27/0.74)	(0.58/1.98, 0.58/1.97)
Waves ($\rho = 1.44\%$)	(0.00/0.00, 0.00/0.00)	(0.04/0.33, 0.05/0.89)	(0.05/0.40, 0.08/0.89)
Crosshatch ($\rho = 1.44\%$)	(0.00/0.00, 0.00/0.00)	(0.20/4.63, 0.21/4.62)	(0.24/4.63, 0.22/4.63)

D.8 MAXIMUM MEAN DISCREPANCY (MMD) METRICS

Table 20: Comparison with baselines that explicitly handle missing data on real datasets, measured by MMD (lower is better). Columns report the missing-entry rate p_{miss}. Entries report MMD values. CoreFlow uses a learned-dimension ratio of $\rho = 36\%$ for Solar and Solar 2 and $\rho = 49\%$ for LSPF, while all ambient-space baselines use $\rho = 100\%$. Solar and Solar 2 correspond to few-sample settings ($n = 300$), whereas LSPF is a standard-sample setting ($n = 8760$).

	Solar			**Solar 2**			**LSPF**		
p_{miss}	**0%**	**20%**	**40%**	**0%**	**20%**	**40%**	**0%**	**20%**	**40%**
MissDiff ($\rho = 100\%$)	0.893	0.917	1.044	0.593	0.915	0.976	**0.116**	0.685	0.811
MissFlow ($\rho = 100\%$)	0.882	0.786	0.866	0.517	0.932	0.923	**0.118**	0.661	0.814
CoreFlow ($\rho = 36\%/49\%$)	**0.177**	**0.193**	**0.249**	**0.187**	**0.267**	**0.360**	0.122	**0.115**	**0.120**

Table 21: Comparison with low-dimensional baselines on the Solar datasets, measured by MMD (lower is better). Columns report the missing-entry rate p_{miss}. Entries report MMD values. Parentheses indicate the learned-dimension ratio ρ.

	Solar			**Solar 2**		
p_{miss}	**0%**	**20%**	**40%**	**0%**	**20%**	**40%**
CSDM ($\rho = 9\%$)	1.197	1.314	1.367	1.206	1.351	1.373
CSDM ($\rho = 36\%$)	0.842	0.873	0.905	0.857	0.913	0.970
SMG-Core ($\rho = 36\%$)	0.224	0.265	0.293	0.219	0.285	0.426
CoreFlow ($\rho = 36\%$)	**0.177**	**0.193**	**0.249**	**0.187**	**0.267**	**0.360**

Table 22: Comparison with low-dimensional generative baselines on simulation datasets, measured by MMD (lower is better). Columns report the missing-entry rate p_{miss}. Entries report MMD values. Parentheses indicate the learned-dimension ratio ρ.

	Blobs			**Bands**			**Waves**			**Crosshatch**		
p_{miss}	**0%**	**20%**	**40%**	**0%**	**20%**	**40%**	**0%**	**20%**	**40%**	**0%**	**20%**	**40%**
CSDM ($\rho = 1.44\%$)	1.206	1.389	1.402	1.071	1.293	1.335	0.913	1.136	1.252	0.988	0.993	1.102
CSDM ($\rho = 64\%$)	0.836	0.897	0.943	0.851	0.906	0.988	0.495	0.561	0.607	0.503	0.599	0.684
SMG-Core ($\rho = 1.44\%$)	0.814	0.970	0.995	0.702	0.747	0.804	0.276	0.314	0.380	0.220	0.289	0.319
SMG-Core ($\rho = 64\%$)	0.312	0.364	0.411	0.323	0.324	0.365	0.127	0.249	0.299	0.100	0.192	0.237
CoreFlow ($\rho = 1.44\%$)	**0.004**	**0.010**	**0.017**	**0.006**	**0.117**	**0.212**	**0.084**	**0.158**	**0.101**	**0.073**	**0.082**	**0.095**

D.9 ABLATION STUDIES: STAGE-I REPRESENTATION WITH PCA-FLOW

To isolate the effect of our matrix-structured Stage-I representation, we further consider a PCA-Flow ablation that follows the same two-stage low-dimensional flow idea as CoreFlow, but replaces the learned row/column subspaces with a standard PCA subspace on flattened matrices. Given complete

training matrices $M_i \in \mathbb{R}^{m_1 \times m_2}$, let $x_i = \mathrm{vec}(M_i) \in \mathbb{R}^D$, where $D = m_1 m_2$, and define

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad X_c = \begin{bmatrix} (x_1 - \mu)^\top \\ \cdots \\ (x_N - \mu)^\top \end{bmatrix}.$$

We compute the economy SVD $X_c = A\Sigma V^\top$ and retain the leading d_{PCA} right singular vectors

$$W_{\mathrm{PCA}} = [v_1, \ldots, v_{d_{\mathrm{PCA}}}] \in \mathbb{R}^{D \times d_{\mathrm{PCA}}}, \qquad d_{\mathrm{PCA}} \le \min\{R^2, N-1\},$$

with an additional rounding to the nearest perfect square when using the U-Net velocity field. Each matrix is then encoded as a PCA score

$$z_i = W_{\mathrm{PCA}}^\top (x_i - \mu) \in \mathbb{R}^{d_{\mathrm{PCA}}},$$

normalized as $\tilde{z}_i = (z_i - \bar{z})/\sigma_z$, and the same CNF architecture is trained in this PCA-score space by flow matching:

$$\epsilon \sim \mathcal{N}(0, I), \qquad t \sim \mathrm{Unif}(0,1), \qquad x_t = (1-t)\epsilon + t\tilde{z}_i,$$

$$\mathcal{L}_{\text{PCA-Flow}}(\theta) = \mathbb{E}_{i,t,\epsilon}\left[\|v_\theta(x_t, t) - (\tilde{z}_i - \epsilon)\|_2^2\right].$$

At generation time, we sample $\epsilon \sim \mathcal{N}(0, I)$, solve the learned ODE from $t = 0$ to $t = 1$ to obtain a generated normalized PCA score $\hat{\tilde{z}}$, de-normalize it by $\hat{z} = \sigma_z \hat{\tilde{z}} + \bar{z}$, and reconstruct the matrix by

$$\hat{M} = \mathrm{mat}\left(\mu + W_{\mathrm{PCA}}\hat{z}\right).$$

Table 23: Ablation comparison on LSPF between CoreFlow and PCA-Flow using three metrics: SVRelL2 ($\times 10^{-2}$), FrobMeanDiff, and AbsEntryMeanDiff ($\times 10^{-2}$). This table is for the complete-data case ($p_{\mathrm{miss}} = 0$). CoreFlow uses the matrix-structured representation $M \approx USV^\top$ with $\rho = 49\%$, while PCA-Flow replaces Stage I with flattened PCA and trains the same CNF on PCA scores.

	LSPF		
Method	**SVRelL2**	**FrobMeanDiff**	**AbsEntryMeanDiff**
PCA-Flow ($\rho = 49\%$)	$2.74_{\pm 0.20}$	$0.34_{\pm 0.15}$	$2.14_{\pm .08}$
CoreFlow ($\rho = 49\%$)	$\mathbf{2.70_{\pm 0.19}}$	$\mathbf{0.24_{\pm .09}}$	$\mathbf{1.87_{\pm .11}}$

Table 24: Ablation comparison on simulation datasets between CoreFlow and PCA-Flow using three metrics: SVRelL2 ($\times 10^{-2}$), FrobMeanDiff, and AbsEntryMeanDiff ($\times 10^{-2}$). This table is for the complete-data case ($p_{\mathrm{miss}} = 0$). CoreFlow uses the matrix-structured representation $M \approx USV^\top$ with $\rho = 1.44\%$, while PCA-Flow replaces Stage I with flattened PCA and trains the same CNF on PCA scores.

	Blobs			Bands			Waves			Crosshatch		
Method	**SVRelL2**	**FrobMeanDiff**	**AbsEntryMeanDiff**	**SVRelL2**	**FrobMeanDiff**	**AbsEntryMeanDiff**	**SVRelL2**	**FrobMeanDiff**	**AbsEntryMeanDiff**	**SVRelL2**	**FrobMeanDiff**	**AbsEntryMeanDiff**
PCA-Flow ($\rho = 1.44\%$)	$1.77_{\pm 0.60}$	$0.55_{\pm 0.25}$	$2.94_{\pm 0.08}$	$24.45_{\pm 1.26}$	$0.33_{\pm 0.21}$	$0.71_{\pm 0.12}$	$7.74_{\pm 0.98}$	$2.39_{\pm 2.16}$	$1.76_{\pm 0.76}$	$8.07_{\pm 0.92}$	$1.71_{\pm 1.29}$	$1.56_{\pm 0.37}$
CoreFlow ($\rho = 1.44\%$)	$\mathbf{0.43_{\pm 0.12}}$	$\mathbf{0.09_{\pm .06}}$	$\mathbf{0.98_{\pm .04}}$	$\mathbf{4.45_{\pm 0.74}}$	$\mathbf{0.08_{\pm .06}}$	$\mathbf{0.29_{\pm .03}}$	$\mathbf{3.47_{\pm 0.17}}$	$\mathbf{0.55_{\pm .38}}$	$\mathbf{0.97_{\pm .09}}$	$\mathbf{3.23_{\pm 0.50}}$	$\mathbf{0.96_{\pm .31}}$	$\mathbf{0.93_{\pm .22}}$

Key findings. PCA-Flow is a useful ablation because it follows the same high-level principle as CoreFlow. In complete-data settings, PCA-Flow can obtain similar, though slightly weaker generation quality, suggesting that low-dimensional flow modeling itself is important. However, PCA-Flow is a flattened and geometry-agnostic representation: it treats each matrix as a vector and reconstructs only through a global linear PCA subspace. Consequently, it does not preserve the row/column factorization structure $USV^\top$ and does not explicitly separate shared row and column subspaces from sample-specific core variation. Besides, it cannot naturally handle incomplete training matrices because PCA requires complete vectors before the projection $W_{\mathrm{PCA}}^\top(x - \mu)$ is well-defined. By contrast, CoreFlow learns a matrix-structure-aware representation $M \approx USV^\top$, trains the CNF on the induced core $S = U^\top MV$, and naturally supports missing entries through the masked Stage-I objective and alternating completion. Overall, the ablation study PCA-Flow confirms the value of low-dimensional flow modeling, while the stronger CoreFlow architecture is essential for geometry-preserving generation and robustness under missingness.