
Kinetic Langevin Diffusion for Crystalline Materials Generation

François Cornet^{*1} Federico Bergamin^{*12}
Arghya Bhowmik¹ Juan Maria Garcia Lastra¹ Jes Frelsen¹² Mikkel N. Schmidt¹²

Abstract

Generative modeling of crystalline materials using diffusion models presents a series of challenges: the data distribution is characterized by inherent symmetries and involves multiple modalities, with some defined on specific manifolds. Notably, the treatment of fractional coordinates representing atomic positions in the unit cell requires careful consideration, as they lie on a hypertorus. In this work, we introduce Kinetic Langevin Diffusion for Materials (KLDM), a novel diffusion model for crystalline materials generation, where the key innovation resides in the modeling of the coordinates. Instead of resorting to Riemannian diffusion on the hypertorus directly, we generalize Trivialized Diffusion Model (TDM) to account for the symmetries inherent to crystals. By coupling coordinates with auxiliary Euclidean variables representing velocities, the diffusion process is now offset to a flat space. This allows us to effectively perform diffusion on the hypertorus while providing a training objective that accounts for the periodic translation symmetry of the true data distribution. We evaluate KLDM on both Crystal Structure Prediction (CSP) and De-novo Generation (DNG) tasks, demonstrating its competitive performance with current state-of-the-art models.

1. Introduction

The discovery of novel compounds with desired properties is critical to several scientific fields, such as molecular discovery (Bilodeau et al., 2022) and materials design (Merchant et al., 2023; Zeni et al., 2025). In the case of crystalline materials, the search space is vast, but only a fraction of it is physically plausible. The main challenges are to efficiently

search the space for *feasible* materials and to accurately estimate their properties. Conventional approaches usually combine random structure search methods with *ab-initio* Quantum Mechanics (QM) methods (Oganov et al., 2019), such as Density Functional Theory (DFT) (Kohn & Sham, 1965); however, structural optimization and property evaluation with DFT can be computationally expensive. Recently, the field has witnessed a paradigm shift where deep generative models have been introduced to supplement traditional search methods (Anstine & Isayev, 2023), such as random search (Pickard & Needs, 2011) or evolutionary algorithms (Glass et al., 2006; Wang et al., 2010). Deep generative models learn to approximate underlying probability distributions from existing material data, and in turn, can be sampled from to generate novel materials based on the learned patterns.

Among deep generative models, diffusion models have been successful on a variety of data modalities relevant to the sciences, ranging from Partial Differential Equations (PDE) simulations (Lippe et al., 2023; Rozet & Louppe, 2023; Shysheya et al., 2024) to molecule generation (Hoogeboom et al., 2022; Xu et al., 2023; Cornet et al., 2024). Unlike molecules, crystalline materials consist of a periodic arrangement of atoms, typically described by a *unit cell*, a parallelepiped that serves as the fundamental building block repeated to tile the entire space. A unit cell is typically represented by three vectors that define its edges and the angles between them, along with the coordinates and species of the atoms inside it. It can be considered as a multi-modal data type, specifically a geometric graph combining discrete and continuous features (Joshi et al., 2023). Notably, the atomic positions are described by coordinates that lie on a hypertorus. Dealing with non-Euclidean data in the diffusion setting requires careful consideration: for example, restricting Brownian motion to a manifold requires incorporating its geometric structure to ensure trajectories remain on the manifold, typically using projection or specialized approximations (Lou et al., 2023). Current diffusion models for crystals either handle this by working in real space, through multi-graph representations (Xie et al., 2022), or on fractional coordinates (Jiao et al., 2023). In addition to this, the data distribution is governed by inherent symmetries, including permutation invariance (swapping atom indices

^{*}Equal contribution ¹Technical University of Denmark, Kongens Lyngby, Denmark ²Pioneer Center for Artificial Intelligence, Copenhagen, Denmark. Correspondence to: François Cornet <frjc@dtu.dk>, Federico Bergamin <fedbe@dtu.dk>.

or lattice bases), translation invariance (shifting atom coordinates), and rotation invariance (rotating atom positions).

When operating on fractional coordinates, the main challenge is to ensure the periodic translation invariance of the learned distribution. This is usually enforced by parameterizing the score network with a periodic translation invariant architecture. However, while existing models have successfully demonstrated the potential of applying diffusion to crystalline materials generation, previous work has highlighted the existence of a mismatch between architecture and supervision signal, resulting in a suboptimal training objective (Lin et al., 2024). While the issue has been acknowledged in the literature, in this paper we show an alternative way to mitigate this mismatch that is effective at low levels of noise.

Contributions In this work, we introduce Kinetic Langevin Diffusion for Materials (KLDM), a novel diffusion model for crystalline materials generation, where the key innovation resides in the modeling of the fractional coordinates. Instead of resorting to Riemannian diffusion on the hyper-torus directly as in previous work (Jiao et al., 2023), we generalize Trivialized Diffusion Model (TDM) (Zhu et al., 2024) to model symmetric distributions over coordinates. Using the structure of the torus, the diffusion process is offset to auxiliary Euclidean variables representing velocities. We propose a specific parameterization of the resulting diffusion process leading to faster training convergence and improved performance while mitigating a mismatch in the training objective of previous diffusion models for materials. Finally, we show that KLDM offers competitive performance on Crystal Structure Prediction (CSP) and De-novo Generation (DNG).

2. Background

2.1. Crystalline materials and related symmetries

In this section, we introduce the data modality we are interested in, along with the relevant symmetries.

Unit cell We are interested in learning the distribution of crystalline materials, described as the repetition of a unit cell in 3D-space. We describe a unit cell containing K atoms as a fully connected geometric graph $\mathbf{x}$,

$$\mathbf{x} = (\mathbf{f}, \mathbf{l}, \mathbf{a}), \quad (1)$$

where $\mathbf{f} = (f_1, \dots, f_K) \in [0, 1)^{3 \times K}$ denotes the fractional coordinates, $\mathbf{l} = (l_1, l_2, l_3) \in \mathbb{R}^{3 \times 3}$ refers to the lattice vectors of the unit cell, and $\mathbf{a} = (a_1, \dots, a_K) \in \mathbb{Z}^K$ encodes the chemical composition.

The infinite periodic structure can be represented as,

$$\{(\mathbf{a}', \mathbf{f}') | \mathbf{a}' = \mathbf{a}; \mathbf{f}' = \mathbf{f} + \mathbf{k}\mathbf{1}^\top, \mathbf{k} \in \mathbb{Z}^3\}, \quad (2)$$

where $\mathbf{1}$ is a vector of ones with size K , and $\mathbf{k} = (k_1, k_2, k_3)$ translates the unit cell to tile the entire space. The lattice vectors $\mathbf{l}$ can also be compactly represented by a 6-dimensional vector consisting of the three lattice vector lengths and their interior angles, such that the representation is invariant to rotation.

Tasks Let $p(\mathbf{x})$ denote the true data distribution over the unit cells, and let $q(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \delta_{\mathbf{x}_i}$ be the empirical data distribution defined by the N available samples. The goal of generative models is to learn an approximate model $p_\theta(\mathbf{x})$ of $p(\mathbf{x})$ using $q(\mathbf{x})$. In the realm of crystalline materials generation, there are two tasks of interest.

Crystal Structure Prediction (CSP) aims at finding low-energy atomic arrangements $(\mathbf{f}, \mathbf{l})$, *i.e.* stable structures, for a given atomic composition $\mathbf{a}$. This is framed as a conditional generation task where a model $p_\theta(\mathbf{f}, \mathbf{l} | \mathbf{a})$ is trained using samples $\{(\mathbf{f}, \mathbf{l}, \mathbf{a})\}_{i=1}^N$ to approximate the true conditional distribution $p(\mathbf{f}, \mathbf{l} | \mathbf{a})$. Given a specific atomic composition $\mathbf{a}$, the model is used to generate possible coordinates $\mathbf{f}$ and lattice parameters $\mathbf{l}$.

De-novo Generation (DNG) aims at discovering novel and stable materials. The goal is to generate samples from $p(\mathbf{f}, \mathbf{l}, \mathbf{a})$. We approximate the distribution by training a generative model $p_\theta(\mathbf{f}, \mathbf{l}, \mathbf{a})$ on samples $\{(\mathbf{f}, \mathbf{l}, \mathbf{a})\}_{i=1}^N$ and evaluate the samples generated by p_θ .

Symmetries of crystalline materials Given a symmetry group G , a distribution is G -invariant if for any group element $g \in G$, $p(g \cdot \mathbf{x}) = p(\mathbf{x})$, with $\cdot$ denoting the group action. A conditional distribution is G -equivariant if for any $g \in G$, $p(g \cdot \mathbf{x} | g \cdot \mathbf{y}) = p(\mathbf{x} | \mathbf{y})$.

A number of transformations leave a material $\mathbf{x}$ unchanged, meaning the *true* data distribution has inherent symmetries:

Permutation of atom indices

$$p(\mathbf{f}, \mathbf{l}, \mathbf{a}) = p(g \cdot \mathbf{f}, \mathbf{l}, g \cdot \mathbf{a}), \quad \forall g \in S_K; \quad (3)$$

Rotation of lattice vectors

$$p(\mathbf{f}, \mathbf{l}, \mathbf{a}) = p(\mathbf{f}, g \cdot \mathbf{l}, \mathbf{a}), \quad \forall g \in \text{SO}(3); \quad (4)$$

Permutation of the lattice basis

$$p(\mathbf{f}, \mathbf{l}, \mathbf{a}) = p(g \cdot \mathbf{f}, g \cdot \mathbf{l}, \mathbf{a}), \quad \forall g \in S_3; \quad (5)$$

Periodic translation of fractional coordinates

$$p(\mathbf{f}, \mathbf{l}, \mathbf{a}) = p(g \cdot \mathbf{f}, \mathbf{l}, \mathbf{a}) \quad \forall g \in \mathbb{T}^3 \cong \mathbb{R}^3 / \mathbb{Z}^3. \quad (6)$$

We want our model, p_θ , to inherit these symmetries. Eq. (3) is naturally addressed by using graph neural networks, combined with a factorized prior distribution with no dependency on the index. The combination of fractional coordinates and rotation-invariant lattice representations addresses Eq. (4). Specific neural network architectures or additional

loss terms can address Eq. (5)—see *e.g.* (Lin et al., 2024). Remaining is to handle Eq. (6), we detail how it can be done in Section 2.3.

2.2. Diffusion models

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) are generative models that learn distributions through a hierarchy of latent variables, corresponding to corrupted versions of the data at increasing noise scales. Diffusion models consist of a forward and a reverse (generative) process. The forward process perturbs samples from the data distribution over time through noise injection, resulting in a trajectory of increasingly noisy latent variables $(\mathbf{x}_t)_{t \in [0, T]}$. Given an initial condition, $\mathbf{x}_0 \sim p_0(\mathbf{x}) = p_{\text{data}}(\mathbf{x})$, the conditional distribution of $(\mathbf{x}_t)_{t \in [0, T]}$ can be characterized by a Stochastic Differential Equation (SDE),

$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t)d\mathbf{w}_t, \quad (7)$$

where $\mathbf{x}_t \in \mathbb{R}^d$ denotes the latent variable at time t , $f(t)$ and $g(t)$ are scalar function of time t , and $\mathbf{w}_t$ is a standard Wiener process in $\mathbb{R}^d$. Due to the linearity of the drift term, for any $t \geq 0$, the corresponding transition kernel admits a closed-form expression (Särkkä & Solin, 2019). For instance, in the Variance-Preserving SDE (VP-SDE) setting (Song et al., 2021), where $f(t) = -\frac{1}{2}\beta(t)$ and $g(t) = \sqrt{\beta(t)}$ for a fixed schedule $\beta(t)$, the kernel writes $p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t|\alpha_t\mathbf{x}_0, \sigma_t^2\mathbb{I})$ with $\alpha_t = \exp(-0.5 \int_0^t \beta(s)ds)$ and $\sigma_t^2 = 1 - \exp(-\int_0^t \beta(s)ds)$, and the process defined by Eq. (7) converges geometrically from a low-variance Gaussian distribution centered around the data to the standard Gaussian distribution $p_T(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t|\mathbf{0}, \mathbb{I})$, which can be therefore interpreted as an uninformative prior distribution.

The time-reversal of Eq. (7) is another diffusion process described by the following reverse-time SDE (Anderson, 1982),

$$d\mathbf{x}_t = [f(t)\mathbf{x}_t - g^2(t)\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x})] dt + g(t)d\hat{\mathbf{w}}_t, \quad (8)$$

with $p_t(\mathbf{x}_t)$ being the density of $\mathbf{x}_t$ and $d\hat{\mathbf{w}}_t$ is a time-reversed Wiener process. Sampling from the prior distribution $p_T(\mathbf{x}_t)$, and simulating Eq. (8) results in a sample from $p(\mathbf{x})$. In practice, the score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x})$ is not available and it is approximated using a *score network* $s_\theta(\mathbf{x}_t, t)$, whose parameters θ are trained via Denoising Score Matching (DSM),

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{\lambda(t)} \left[\left\| s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) \right\|_2^2 \right], \quad (9)$$

where $\lambda(t)$ is a positive time-dependent weighting function and the expectation is taken over the joint distribution $t \sim \mathcal{U}[0, 1]$, $\mathbf{x}_0 \sim p(\mathbf{x})$, and $\mathbf{x}_t \sim p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)$.

2.3. Existing diffusion models for fractional coordinates

As previously mentioned, the fractional coordinates define a hypertorus $\mathbf{f} \in [0, 1)^{3 \times K} \cong \mathbb{T}^{3 \times K}$. Most existing work has built upon DIFFCSP (Jiao et al., 2023), which leverages the score-based framework of Song et al. (2021) extended to Riemannian manifolds (De Bortoli et al., 2022; Jing et al., 2022).

In what follows, we present the key ingredients of DIFFCSP for modeling fractional coordinates.

Transition kernel DIFFCSP implements a specific case of Eq. (7)—*i.e.* Variance-Exploding SDE (VE-SDE), with $f(t) = 0$ and $g(t) = \sqrt{d\sigma^2(t)/dt}$, where $\sigma(t) = \sigma_{\min}^{1-t} \sigma_{\max}^t$ with $\sigma_{\min}$ and $\sigma_{\max}$ being hyperparameters.

In practice, as sampling noisy fractional coordinates $\mathbf{f}_t$ given $\mathbf{f}_0$ using a normal distribution does not capture the bounded and cyclical nature of $p(\mathbf{f})$, the solution consists in first sampling from the normal distribution $\tilde{\mathbf{f}}_t \sim \mathcal{N}(\mathbf{f}_0, \sigma^2(t))$ and then wrapping the samples $\mathbf{f}_t = w(\tilde{\mathbf{f}}_t)$ —where $w(\cdot) = \cdot - \lfloor \cdot \rfloor$ is the wrapping function.

The transition kernel corresponding to this two-step procedure corresponds to a wrapped normal distribution,

$$p_{t|0}(\mathbf{f}_t|\mathbf{f}_0) \propto \sum_{\mathbf{k} \in \mathbb{Z}^{3 \times K}} \exp\left(-\frac{\|\mathbf{f}_t - \mathbf{f}_0 + \mathbf{k}\|^2}{2\sigma^2(t)}\right), \quad (10)$$

whose gradient can then be approximated by truncating the series, allowing for training using denoising score-matching. Due to the wrapping operation, Eq. (10) converges to a uniform distribution over the hypertorus as $t \rightarrow T$.

Denoising score-matching Given the transition kernel $p_{t|0}(\mathbf{f}_t|\mathbf{f}_0)$, the approximate score function can be optimized by minimizing a denoising score-matching objective, which at time t writes

$$\mathcal{L}_{\mathbf{f}_t} = \mathbb{E}_{\mathbf{f}_0, \mathbf{f}_t} \left[\left\| \lambda(t) \left[\nabla_{\mathbf{f}_t} \log p_{t|0}(\mathbf{f}_t|\mathbf{f}_0) - s_\theta^{\mathbf{f}}(\mathbf{x}_t, t) \right] \right\|_2^2 \right], \quad (11)$$

where $\mathbf{f}_0 \sim p_0(\mathbf{f})$, $\mathbf{f}_t \sim p_{t|0}(\mathbf{f}_t|\mathbf{f}_0)$, $s_\theta^{\mathbf{f}}(\mathbf{x}_t, t)$ denotes the output of the score network corresponding to fractional coordinates, and $\lambda(t) = 1/\mathbb{E}_{\mathbf{f}_t \sim p_{t|0}(\mathbf{f}_t|\mathbf{f}_0)} [\|\nabla_{\mathbf{f}_t} \log p_{t|0}(\mathbf{f}_t|\mathbf{f}_0)\|_2^2]$ is a (pre-computable) scale factor ensuring that the loss magnitude is constant in expectation across time (Jing et al., 2022; Jiao et al., 2023).

Invariant approximate distribution As introduced in Eq. (6), the true data distribution $p(\mathbf{x})$, with $\mathbf{x} = (\mathbf{f}, \mathbf{l}, \mathbf{a})$ as defined in Section 2.1, is periodic translation invariant—*i.e.* unit cells equivalent up to periodic translation shall be equally likely. In DIFFCSP (Jiao et al., 2023), the learned distribution $p_\theta(\mathbf{x})$ is made invariant by combining an invariant prior - *i.e.* uniform distribution on the hypertorus - with an approximate reverse process that is equivariant to periodic

translation—i.e. $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = p_\theta(g \cdot \mathbf{x}_{t-1}|g \cdot \mathbf{x}_t), \forall g \in \mathbb{R}^3/\mathbb{Z}^3$. In practice, this is realized by parameterizing the score using a neural network that is invariant to periodic translations—i.e. $s_\theta^f(\mathbf{x}_t, t) = s_\theta^f(g \cdot \mathbf{x}_t, t), \forall g \in \mathbb{R}^3/\mathbb{Z}^3$. If the learned score $s_\theta^f(\mathbf{x}_t, t)$ correctly approximates the score of the true target distribution, then we are guaranteed to model a distribution $p_\theta(\mathbf{x})$ that exhibits the same symmetries as $p(\mathbf{x})$ from which new samples can be drawn.

3. Kinetic Langevin Diffusion for Materials generation

We now introduce Kinetic Langevin Diffusion for Materials (KLDM), and particularize our exposition to the fractional coordinates $\mathbf{f} \in [0, 1)^{3 \times K} \cong \mathbb{T}^{3 \times K}$, used to define the positions of atoms inside the unit cell.

Given the isomorphism between the torus $\mathbb{T}$ and $\text{SO}(2)$ ¹, the fractional coordinates can alternatively be described as a collection of 2×2 rotation matrices. We denote this alternative representation $\hat{\mathbf{f}}$. Since $\hat{\mathbf{f}}$ is defined on a direct product of connected compact Lie groups², we propose to generalize Trivialized Diffusion Model (TDM) (Zhu et al., 2024) to operate on geometric graphs similar to those defined in Eq. (1). While TDM in principle allows for proper treatment of the fractional coordinates *out-of-the-box*, we find its direct application to crystalline materials generation to result in slow convergence and subpar performance. We consequently propose a set of modifications designed to facilitate faster convergence and enhance empirical results.

3.1. Trivialized Diffusion Model (TDM) for fractional coordinates

Building upon previous work on momentum-based optimization (Tao & Ohsawa, 2020) and sampling (Kong & Tao, 2023), TDM is specifically tailored to data defined on Lie groups, and exploits the particular group structure, specifically the left-trivialization operation, to effectively perform diffusion on the manifold via the Lie algebra. The main idea is to couple the variables of interest defined on the manifold with auxiliary variables representing velocities defined on the Lie algebra. More precisely, the fractional coordinates $\mathbf{f}$, elements of the group $G = \text{SO}(2)^{3 \times K}$, are coupled with velocities $\hat{\mathbf{v}} \in \mathfrak{g}$ defined on the Lie algebra $\mathfrak{g}$. The latter corresponds to the tangent space $T_e G$ at the identity element of the group $e \in G$, and crucially can be thought of as a Euclidean space, $\mathfrak{g} \cong \mathbb{R}^{3 \times K}$. In the present setting, velocities

$\hat{\mathbf{v}}$ are 2×2 skew-symmetric matrices, whose anti-diagonal element, v , is real-valued.

Forward process As the Lie algebra is isomorphic to Euclidean space, a standard diffusion process can be defined for $\hat{\mathbf{v}}$. Given its coupling with $\hat{\mathbf{f}}$ determined by left-trivialization (Zhu et al., 2024), the resulting forward process is defined as,

$$\begin{cases} d\hat{\mathbf{f}}_t &= \hat{\mathbf{f}}_t \hat{\mathbf{v}}_t dt, \\ d\hat{\mathbf{v}}_t &= -\gamma \hat{\mathbf{v}}_t dt + \sqrt{2\gamma} d\mathbf{w}_t^{\mathfrak{g}}, \end{cases} \quad (12)$$

where the Ordinary Differential Equation (ODE) describes the time evolution of the fractional coordinates $\hat{\mathbf{f}}_t$ (that lie on a manifold) through a coupling with the velocity variables $\hat{\mathbf{v}}_t$. The latter evolve according to an SDE similar to that of Eq. (7), with constant drift $f(t) = -\gamma$ and constant volatility $g(t) = \sqrt{2\gamma}$, and where $\mathbf{w}_t^{\mathfrak{g}}$ denotes a standard Wiener process on the Lie algebra $\mathfrak{g}$. We note that, in principle, f and g could be generalized to time-dependent functions but that we consider constant here. In summary, Eq. (12) corrupts $\hat{\mathbf{f}}_t$ living on a hypertorus via a standard Euclidean diffusion process on the auxiliary variables, while guaranteeing that the trajectory $(\hat{\mathbf{f}}_t)_{t \in [0, T]}$ remains on the manifold at all times. Eq. (12) converges to a product of easy-to-sample distributions: a uniform distribution on G , and a standard Normal distribution on $\mathfrak{g}$.

Reverse (generative) process The time-reversal of Eq. (12) is given by

$$\begin{cases} d\hat{\mathbf{f}}_t &= \hat{\mathbf{f}}_t \hat{\mathbf{v}}_t dt, \\ d\hat{\mathbf{v}}_t &= \left[-\gamma \hat{\mathbf{v}}_t - 2\gamma \nabla_{\hat{\mathbf{v}}_t} \log p_t(\hat{\mathbf{f}}_t, \hat{\mathbf{v}}_t) \right] dt + \sqrt{2\gamma} d\hat{\mathbf{w}}_t^{\mathfrak{g}}, \end{cases} \quad (13)$$

where t flows backwards and $\nabla_{\hat{\mathbf{v}}_t} \log p_t(\hat{\mathbf{f}}_t, \hat{\mathbf{v}}_t)$ denotes the (true) score. The latter is unavailable, but can be approximated with a neural network $s_\theta(\hat{\mathbf{f}}_t, \hat{\mathbf{v}}_t, t)$ trained using DSM as in Eq. (11). However, we stress that the gradient is now with respect to $\hat{\mathbf{v}}_t$ —i.e. an Euclidean variable, in contrast to Eq. (11) where it was with respect to $\mathbf{f}_t$ —see Eq. (17). As in Eq. (12) there is no direct noise in the forward dynamics of $\hat{\mathbf{f}}_t$, its time-reversal in Eq. (13) does not need score-based correction and corresponds to the reverse ODE.

Practically, we split the vector field of the generative dynamics in Eq. (13) into two vector fields: one for $\hat{\mathbf{f}}_t$ where $\hat{\mathbf{v}}_t$ is considered fixed—this is a simple linear ODE; and one for $\hat{\mathbf{v}}_t$ where $\hat{\mathbf{f}}_t$ is considered fixed—this is a semi-linear SDE, as usually encountered in diffusion models.

Discretized update of the fractional coordinates The ODEs in Eqs. (12) and (13) describe the dynamics associated with the fractional coordinates. Since they lie on a manifold, integrating the dynamics from time t to $t + dt$ involves solving a Riemannian exponential map. Intuitively, this

¹We provide an intuitive explanation of this correspondence in Appendix B.1.

²A Lie group is a smooth manifold equipped with a group structure denoted by G and smooth group operations. We provide a short informal introduction to Lie groups and Lie algebra in Appendix A.

operation generalizes the concept of moving on a straight line to the manifold case. By considering an initial velocity $\hat{\mathbf{v}}_t$, an initial position $\hat{\mathbf{f}}_t$ and the step-size dt , the update of the positions can be written as

$$\hat{\mathbf{f}}_{t+dt} = \exp_{\hat{\mathbf{f}}_t}(\hat{\mathbf{f}}_t \hat{\mathbf{v}}_t dt) = \hat{\mathbf{f}}_t \expm(\hat{\mathbf{v}}_t dt), \quad (14)$$

where the matrix multiplication $\hat{\mathbf{f}}_t \hat{\mathbf{v}}_t dt$ returns the tangent vector in $\hat{\mathbf{f}}_t$, and the matrix exponential is the exact solution of the exponential map (Zhu et al., 2024).

The formalism previously introduced considered $\hat{\mathbf{f}}$ as a collection of rotation matrices. However, the usual neural networks used to parameterize the score have been designed to process fractional coordinates (Jiao et al., 2023; Lin et al., 2024). We therefore want to work on the hypertorus $\mathbb{T}^{3 \times K}$ directly, using the original representation of the fractional coordinates $\mathbf{f}$. Concretely, if we consider the exponential map defined in Eq. (14) and that $\hat{\mathbf{f}}^i$ is a single rotation matrix by an angle $\theta \in [-\pi, \pi)$, the matrix exponential corresponds to a periodic translation as follows,

$$\hat{\mathbf{f}}^i \expm(\hat{\mathbf{v}}^i dt) \rightarrow w(\theta^i + v^i dt), \quad (15)$$

where $\hat{\mathbf{v}}^i$ is a skew-symmetric matrix and v^i is its anti-diagonal element, while $w(\cdot) = \text{atan2}(\sin(\cdot), \cos(\cdot))$ is the wrapping function with atan2 denoting the signed atan function—see Appendix B.2 for more details. Since θ^i in Eq. (15) corresponds to a scaled version of the corresponding coordinate f^i , we continue the presentation with all operations expressed in terms of $\mathbf{f}$.

Transition kernel (Zhu et al., 2024) The transition kernel corresponding to the forward dynamics of Eq. (12) writes

$$p_{t|0}(\mathbf{f}_t, \mathbf{v}_t | \mathbf{f}_0, \mathbf{v}_0) = \text{WN}(\mathbf{r}_t | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I}) \cdot \mathcal{N}_{\mathbf{v}}(\mathbf{v}_t | \boldsymbol{\mu}_{\mathbf{v}_t}, \sigma_{\mathbf{v}_t}^2 \mathbf{I}), \quad (16)$$

where $\mathbf{r}_t = \logm(\mathbf{f}_0^{-1} \mathbf{f}_t)$, $\text{WN}(\cdot | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I})$ denotes the density of the Wrapped Normal distribution with mean $\boldsymbol{\mu}_{\mathbf{r}_t} = \frac{1-e^{-t}}{1+e^{-t}}(\mathbf{v}_t + \mathbf{v}_0)$ and variance $\sigma_{\mathbf{r}_t}^2 = 2t + \frac{8}{e^t+1} - 4$, while $\mathcal{N}_{\mathbf{v}}$ is a Gaussian distribution with $\boldsymbol{\mu}_{\mathbf{v}_t} = e^{-t} \mathbf{v}_0$ and $\sigma_{\mathbf{v}_t}^2 = 1 - e^{-2t}$.

Practically, noisy samples $\mathbf{f}_t$ can be obtained from the transition kernel in Eq. (16) as follows. First, we sample the auxiliary variables $\mathbf{v}_t$. We then sample $\mathbf{r}_t = \logm(\mathbf{f}_0^{-1} \mathbf{f}_t)$. Finally, $\mathbf{f}_t$ is obtained as $\mathbf{f}_t = \mathbf{f}_0 \expm(\mathbf{r}_t)$.

The joint distribution in Eq. (16) evolves from the product $p(\mathbf{f}) \cdot p(\mathbf{v}_0)$, to a tractable limiting distribution that is the product of a uniform distribution over $\mathbb{T}^{3 \times K}$ in $\mathbf{f}$ and a standard Normal in $\mathbf{v}$.

Denosing score-matching target As the diffusion process only acts on the (Euclidean) velocity variables, we only need to compute the score of Eq. (16) with respect to the velocity

variables, $\mathbf{v}_t$, unlike the objective presented in Eq. (11) where the gradient was instead computed with respect to the coordinates $\mathbf{f}_t$. It writes,

$$\begin{aligned} \nabla_{\mathbf{v}_t} \log p_{t|0} &= \nabla_{\boldsymbol{\mu}_{\mathbf{r}_t}} \log \text{WN}(\mathbf{r}_t | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I}) \frac{\partial \boldsymbol{\mu}_{\mathbf{r}_t}}{\partial \mathbf{v}_t} \\ &\quad + \nabla_{\mathbf{v}_t} \log \mathcal{N}_{\mathbf{v}}(\mathbf{v}_t | \boldsymbol{\mu}_{\mathbf{v}_t}, \sigma_{\mathbf{v}_t}^2 \mathbf{I}), \end{aligned} \quad (17)$$

where we shortened $p_{t|0}(\mathbf{f}_t, \mathbf{v}_t | \mathbf{f}_0, \mathbf{v}_0)$ as $p_{t|0}$. Eq. (17) is the target used to train our score network with the DSM objective defined in Eq. (9).

3.2. Modeling choices

Initial zero velocities Defining $p(\mathbf{v}_0)$ is a design choice, and we find that setting the initial velocities to zero, *i.e.* $p(\mathbf{v}_0) = \delta(\mathbf{v}_0)$, to be beneficial. This intuitively corresponds to considering coordinates $\mathbf{f}$ initially at rest, at time $t = 0$, with noise gradually propagating from the velocities to the coordinates over the course of the diffusion process. A similar benefit was also observed by Dockhorn et al. (2022), for small initial velocities.

Simplified score parameterization While predicting the full score in Eq. (17) directly is possible, alternative parameterizations exist—similar to what is done in standard diffusion models. By carefully inspecting the score expression, we note that the target writes as a sum of two terms, where some of the quantities are known,

$$\begin{aligned} \nabla_{\mathbf{v}_t} \log p_{t|0}(\mathbf{f}_t, \mathbf{v}_t | \mathbf{f}_0, \mathbf{v}_0) &= \frac{1 - e^{-t}}{1 + e^{-t}} \nabla_{\boldsymbol{\mu}_{\mathbf{r}_t}} \text{WN}(\mathbf{r}_t | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I}) - \frac{\boldsymbol{\varepsilon}_{\mathbf{v}}}{\sigma_{\mathbf{v}_t}}, \end{aligned} \quad (18)$$

with $\boldsymbol{\varepsilon}_{\mathbf{v}}$ denoting the reparameterization noise sampled to obtain $\mathbf{v}_t$. When considering zero initial velocities, *i.e.* $\mathbf{v}_0 = \mathbf{0}$, we have that $\boldsymbol{\mu}_{\mathbf{v}_t} = \mathbf{0}, \forall t$. Using the relationship, $\mathbf{v}_t = \boldsymbol{\mu}_{\mathbf{v}_t} + \sigma_{\mathbf{v}_t} \boldsymbol{\varepsilon}_{\mathbf{v}}$, the second term in Eq. (18) can be computed in closed-form—as we have $\boldsymbol{\varepsilon}_{\mathbf{v}} = \mathbf{v}_t / \sigma_{\mathbf{v}_t}$, and does not need to be learned. We are then left with a single unknown term, and the simplified parameterization writes,

$$s_{\theta}^{\mathbf{v}}(\mathbf{x}_t, t) = \frac{1 - e^{-t}}{1 + e^{-t}} s_{\theta}^{\mathbf{f}}(\mathbf{x}_t, t) - \frac{\mathbf{v}_t}{\sigma_{\mathbf{v}_t}^2}, \quad (19)$$

where superscript $\mathbf{f}$ in $s_{\theta}^{\mathbf{f}}(\mathbf{x}_t, t)$ refers to the output of the score network corresponding to fractional coordinates. More details are provided in Appendix C.

Empirically, we find the combination of the zero initial velocities and the simplified parameterization to be beneficial for convergence and performance, as discussed in Appendix E.

Architecture of the score network As imposed by the symmetries inherent to the target distribution, we parameterize our score network, $s_{\theta}(\mathbf{x}_t, t)$, using a graph neural network

architecture, with a backbone similar to that of previous work (Jiao et al., 2023). The sole difference is that the score network now takes the auxiliary variables, $\mathbf{v}_t$, as additional inputs. The network is made invariant to periodic translation of $\mathbf{f}_t$, by featurizing pairwise differences of fractional coordinates with periodic functions of different frequencies, or similarly with shortest distances on the flat torus. More details about the architecture are provided in Appendix I.

Velocity fields with zero net translation Due to the translational symmetry of the target distribution - and consequently the translation invariance of the score parametrization - we want to prevent velocities $(\mathbf{v}_t)_{t \in [0, T]}$ from inducing a net overall translation of the system at every time step. Since velocities are defined on the Lie algebra and can therefore be viewed as Euclidean variables, we consider velocity fields living in the mean-free linear subspace. This is, in essence, similar to subtracting the center of gravity when working with molecules in Euclidean space (Xu et al., 2022; Hoogeboom et al., 2022), except that it now takes place on the velocities defined on the Lie algebra.

In practice, this implies that the distribution of velocities, $\mathcal{N}_v$ in Eq. (16), corresponds to a projected Normal where all samples are mean-free—that is, they satisfy $\sum_k^K \mathbf{v}_k = \mathbf{0}$, to account for the constraint on the velocities. We also remove the mean from the sampled $\mathbf{r}_t$, and project the score in Eq. (17), to ensure that it does not introduce a net translation. Finally, the corresponding velocity score output $s_\theta^v(\mathbf{x}_t, t)$ in Eq. (19) - and hence $s_\theta^f(\mathbf{x}_t, t)$ - is constructed to be mean-free as well.

Ideally, we would like to preserve the group element g used to represent the clean sample $\mathbf{f}_0$, since the score network is insensitive to translation. While restricting the velocity fields to the mean-free subspace prevents global drift, it does not always ensure that the noisy coordinates, $\mathbf{f}_t$, maintain the *mean*³ of the original sample, $\mathbf{f}_0$.

However, we empirically observe that the mean tends to be preserved at moderate noise levels (see Figs. 2 and 3). This helps mitigate the potential mismatch that exists between the translation-invariant score parameterization and the non-invariant training target.

This issue has been pointed out in previous. Lin et al. (2024), for example, proposed a so-called *Periodic CoM-free Noising* scheme, in which the noise is carefully designed to ensure a translation-invariant training target. The resulting transition kernel is formulated as a von Mises distribution and estimated numerically via Monte Carlo simulations.

We refer the reader to Appendix D for a longer discussion

³In this case, we are referring to the mean defined on a manifold and not the Euclidean mean. See Appendix D for a more precise definition.

and a more detailed analysis.

3.3. Other modalities

Lattice vectors We represent lattices l as 6-dimensional vectors, collecting side lengths and angles. We follow previous work (Jiao et al., 2023; Lin et al., 2024) and rely on standard Euclidean diffusion (VP-SDE) as defined in Eqs. (7) and (8), with drift and diffusion functions defined by a linear schedule.

Compositions In the DNG setting, we employ three distinct representations for atomic compositions $\mathbf{a}$, as per previous work: continuous diffusion using one-hot encoding (Jiao et al., 2023); continuous diffusion on analog-bits (Chen et al., 2023; Miller et al., 2024); and discrete (absorbing) diffusion (Austin et al., 2021; Jiao et al., 2023).

4. Related work

Early deep generative models for crystal generation leveraged image representations (Hoffmann et al., 2019; Court et al., 2020), ad-hoc frequency space representations (Ren et al., 2022) or 3D coordinates without considering their geometric nature (Nouira et al., 2018; Kim et al., 2020; Yang et al., 2024b).

Since then, several works have leveraged diffusion models operating on geometric graphs. The seminal approach of Xie et al. (2022) initially worked in *real space*, and resorted to multi-graphs to account for periodicity (Xie & Grossman, 2018). The diffusion process was limited to coordinates, with fixed lattice parameters and composition predicted by a VAE. While such a model has shown practical usefulness, *e.g.* to find novel 2D materials (Lyngby & Thygesen, 2022), more recent diffusion models instead defined a (more flexible) joint diffusion process for coordinates, lattice structure, and atom types (Jiao et al., 2023; Zeni et al., 2025), and accounted for periodicity by operating on fractional coordinates. Miller et al. (2024) generalized Riemannian flow matching (Lipman et al., 2023; Chen & Lipman, 2024) to the same setup. Sriram et al. (2024) further leveraged the flexibility of flow matching and used a fine-tuned LLM as a base distribution. Others (Flam-Shepherd & Aspuru-Guzik, 2023; Gruver et al., 2024; Antunes et al., 2024) also trained or fine-tuned LLMS on text representation of materials and demonstrated the ability of such models to generate valid descriptions, sometimes outperforming domain-specific methods.

Finally, a recent line of work (Jiao et al., 2024; Levy et al., 2024) has sought to exploit space-group information to restrict generation to the smallest asymmetric part of the unit cell only, including for disordered materials (Petersen et al., 2025).

Table 1: **Crystal Structure Prediction (CSP) results.** Baseline results are extracted from the respective papers. @ indicates the number of samples considered to evaluate the metrics, *e.g.* @20 indicates the best of 20. Error bars for @1 represent the standard deviation over the mean at sampling time across 20 different seeds. Most notably, KLDM compares favorably to the competing models, achieving performance comparable to or better than state-of-the-art (SOTA) methods.

MODEL	PEROV-5		MP-20		MPTS-52	
	MR [%] $\uparrow$	RMSE $\downarrow$	MR [%] $\uparrow$	RMSE $\downarrow$	MR [%] $\uparrow$	RMSE $\downarrow$
METRICS @ 1						
CDVAE	45.31	0.1138	33.90	0.1045	5.34	0.2106
DIFFCSP (PC)	52.02	0.0760	51.49	0.0631	12.19	0.1786
EQUICSP (PC)	52.02	0.0707	57.59	<u>0.0510</u>	14.85	0.1169
FLOWMM	53.15	0.0992	61.39	0.0566	17.54	0.1726
KLDM- ε (EM)	<u>53.14</u> $\pm .6$	0.0758 $\pm .002$	61.72 $\pm .2$	0.0686 $\pm .001$	17.71 $\pm .3$	0.2023 $\pm .005$
KLDM- ε (PC)	52.72 $\pm .8$	<u>0.0678</u> $\pm .002$	<u>65.37</u> $\pm .1$	0.0455 $\pm .001$	<u>21.46</u> $\pm .2$	0.1339 $\pm .002$
KLDM- x_0 (EM)	52.44 $\pm .7$	0.0698 $\pm .002$	62.92 $\pm .2$	0.0833 $\pm .002$	21.13 $\pm .2$	0.1800 $\pm .003$
KLDM- x_0 (PC)	52.14 $\pm .9$	0.0647 $\pm .002$	65.83 $\pm .2$	0.0517 $\pm .001$	23.93 $\pm .2$	<u>0.1276</u> $\pm .002$
METRICS @ 20						
CDVAE	88.51	0.0464	66.95	0.1026	20.79	0.2085
DIFFCSP (PC)	98.60	0.0128	77.93	0.0492	34.02	0.1749
FLOWMM	98.60	0.0328	75.81	0.0479	34.05	0.1813
KLDM- ε (EM)	99.97	<u>0.0152</u>	83.68	0.0532	<u>39.04</u>	0.1865
KLDM- ε (PC)	<u>99.94</u>	0.0226	81.08	0.0440	39.81	<u>0.1462</u>
KLDM- x_0 (EM)	99.89	0.0186	<u>82.94</u>	0.0575	37.77	0.1673
KLDM- x_0 (PC)	99.92	0.0255	80.18	<u>0.0453</u>	37.10	0.1394

5. Experimental results

5.1. Settings

Tasks and Datasets We now evaluate KLDM on the two tasks outlined in Section 2.1: Crystal Structure Prediction (CSP) and De-novo Generation (DNG).

We follow previous work (Jiao et al., 2023) and evaluate KLDM across 4 datasets: PEROV-5 (Castelli et al., 2012) including perovskite materials with 5 atoms per unit cell (ABX_3), all sharing the same structure but differing in composition; MP-20 including almost all experimentally stable materials from the Materials Project (Jain et al., 2013), with unit cells containing at most 20 atoms; and MPTS-52 also extracted from the Materials Project (Jain et al., 2013), with unit cells containing up to 52 atoms.

Sampling schemes As DIFFCSP (Jiao et al., 2023) and EQUICSP (Lin et al., 2024) both rely on a Predictor-Corrector (PC) integrator (Song et al., 2021), we consider two different integration schemes for KLDM: the first combines Euler–Maruyama (EM) for the lattice parameters with an exponential integrator for the velocities, while the second applies a PC scheme to the velocities while retaining EM for the other modalities. Details are provided in Algorithms 3 and 4.

5.2. Crystal Structure Prediction (CSP) task

Model setup We present two different models that differ in terms of the score parameterization for the lattice parameters l . We consider a model that uses the ε -parameterization (KLDM- ε) and one using the x_0 parameterization (KLDM- x_0). For the latter, we also standardize the value of the lattice parameters following (Miller et al., 2024).

Metrics We follow the evaluation procedure of Xie et al. (2022) to assess the quality of the structures generated by KLDM. We report Match Rate (MR), measuring the proportion of reconstructions from $q_\theta(f, l|a)$ that are satisfactorily close to the ground truth structures as per StructureMatcher (Ong et al., 2013); and Root-Mean-Square-Error (RMSE), quantifying the RMSE between coordinates of matching reconstructions and ground truth structures.

Baselines We compare KLDM to the following recent generative models: CDVAE (Xie et al., 2022), DIFFCSP (Jiao et al., 2023), EQUICSP (Lin et al., 2024), and FLOWMM (Miller et al., 2024).

Results We perform CSP on all datasets and present the corresponding results in Table 1. We note that this task constitutes an ideal test bench for measuring the effect of the novel treatment of the fractional coordinates specific to

Table 2: **De-novo Generation (DNG) results on MP-20**. Stability metrics are evaluated using MATTERGEN’s pipeline (Zeni et al., 2025) with MatterSim-v1-1M (Yang et al., 2024a). Baseline results are taken from MATTERGEN’s own benchmark. KLDM was trained on the original MP-20 dataset, whereas MATTERGEN-MP* and DIFFCSP* were trained on a re-optimized version. We report average and standard deviation across 3 sampling seeds. Notably, KLDM with analog-bit or discrete diffusion outperforms DIFFCSP in terms of RMSD, energy above the hull, and stability, while being slightly subpar on RMSD.

	RMSD [$\text{\AA}$] $\downarrow$	AVG. ABOVE HULL [eV/atom] $\downarrow$	STABLE [%] $\uparrow$	S.U.N. [%] $\uparrow$
MATTERGEN-MP*	0.147	0.201	47.05	25.76
DIFFCSP*	0.413	0.189	41.25	20.13
KLDM-x_0 (C)	0.371 $\pm_{.01}$	0.269 $\pm_{.01}$	38.62 $\pm_{.1}$	16.67 $\pm_{.1}$
KLDM-x_0 (C-AB)	0.296 $\pm_{.01}$	0.187 $\pm_{.01}$	49.84 $\pm_{.1}$	17.91 $\pm_{.1}$
KLDM-x_0 (D)	0.283 $\pm_{.01}$	0.155 $\pm_{.01}$	59.21 $\pm_{.1}$	18.52 $\pm_{.1}$

KLDM. On the simpler PEROV-5 dataset, KLDM performs on par with the compared models for the @1 experiment, and yields improved results for @20. Notably, the PC sampler does not enhance sample quality on this dataset. In contrast, on the more realistic datasets MP-20 and MPTS-52, KLDM already yields competitive MR for the EM sampler. The PC sampler further improves performance, improving MR and reducing RMSE. Additional gains are achieved on these datasets by standardizing the lattice parameters and adopting the x_0 -parameterization for the lattice parameters.

For completeness, we report results on the CARBON-24 dataset in Appendix F, Table 4. The @1 task is inherently ill-defined, as the dataset consists exclusively of carbon atoms, allowing multiple distinct structures to satisfy the same composition. On the @20 task, KLDM compares favorably to baselines.

5.3. De-novo Generation (DNG) task

Model setup As described in Section 3.3, the CSP model formulation can be readily extended to the DNG task, by introducing an additional diffusion process operating on the atom types, α . To sample from $p_\theta(x)$, we first sample the number of atoms K in the unit cell from the empirical distribution observed in the training set, *i.e.* from $p_\theta(x|K)p(K)$ (Hoogeboom et al., 2022).

For the lattice parameters l , we adopt the KLDM- x_0 variant, as it demonstrated the best performance in the CSP task on the larger datasets. Similarly, we use the PC sampler as it consistently yielded improved results.

Regarding the atom types α , we compare three approaches for modeling diffusion over these discrete attributes: (C) continuous diffusion on one-hot encoded atom types (Jiao et al., 2023), (C-AB) continuous diffusion on analog bits (Chen et al., 2023), and (D) discrete diffusion with absorbing state (Austin et al., 2021; Shi et al., 2024).

Metrics We evaluate generated samples using a machine-learning interatomic potential, based on the open-source pipeline from MATTERGEN (Zeni et al., 2025)—*i.e.* using MatterSim-v1-1M (Yang et al., 2024a). Sample quality is assessed in terms of the following metrics: RMSD between the generated samples and their relaxed structure, where lower values mean generated structures are closer to equilibrium; the average energy above the hull, with lower values meaning that generated materials are closer to thermodynamic (meta-)stability; stability, measured as the proportion of samples with an energy above the convex hull below 0.1 eV/atom; and S.U.N. (stable, unique, novel) measuring the percentage of generated samples that satisfy all three criteria, identifying promising candidates. For each variant of KLDM, we generated 10000 samples, from which we discarded samples that contained elements not supported by the validation pipeline.

Baselines We compare KLDM to DIFFCSP and MATTERGEN-MP on MP-20. For the baselines, we report numbers borrowed from MATTERGEN’s own benchmark (Zeni et al., 2025). Notably, the baseline models were trained on a re-optimized version of MP-20 in which certain chemical elements were removed, specifically noble gases, radioactive elements, and elements with atomic number greater than 84. Additionally, samples with energy above the hull bigger than 0.1 eV/atom were also filtered out. In contrast, our model was trained on the original, unfiltered MP-20.

Results The results are summarized in Table 2. Notably, when relying on analog-bits or discrete diffusion to model the atom types, KLDM outperforms DIFFCSP in terms of RMSD, energy above the hull and stability while being slightly subpar on S.U.N..

Compared to MATTERGEN-MP, KLDM generates structures that are more thermodynamically stable and closer to the convex hull on average, but with slightly higher RMSD, indicating larger deviations from the relaxed geometries.

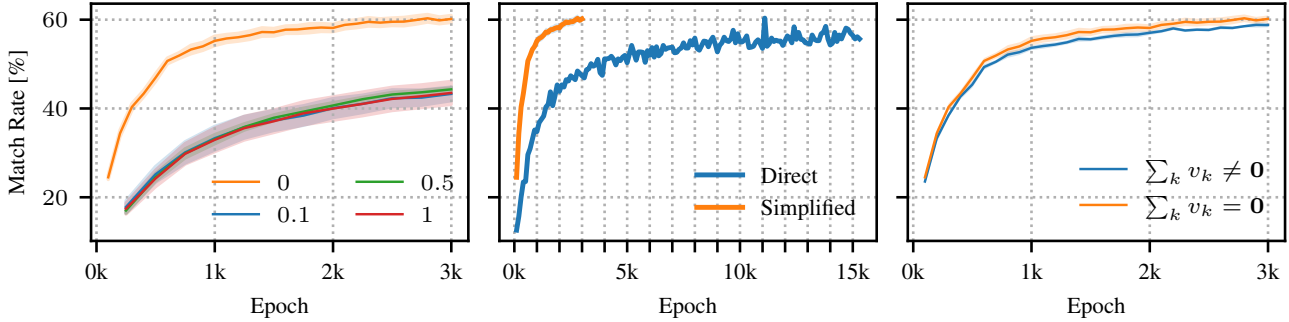


Figure 1: **Ablation study results.** We report the mean and standard error across 6 seeds of the validation Match Rate (MR). **(Left)** Impact of the initial velocity distribution variance. A variance of 0 corresponds to the zero-initial velocity case (i.e., a delta distribution). **(Center)** Effect of the score parameterization on convergence. **(Right)** Impact of enforcing zero-net translation in velocity fields under zero-initial velocities. Notably, all design choices shown contribute to performance improvements.

It also produces slightly fewer unique and stable samples, as reflected by a lower S.U.N. score. We attribute this remaining performance gap to several factors: (1) the more expressive denoiser architecture used by MATTERGEN-MP, which operates in real space, (2) its uses of PC sampler for the lattice parameters, and (3) the impact of dataset pre-processing in the re-optimized MP-20 used for training.

Additional results using standard proxy metrics are provided in Appendix G.

5.4. Ablation study

In this section, we analyze the impact of three key design choices: (1) the distribution of the initial velocities, (2) the simplified score parameterization in Eq. (19) compared to the direct parameterization of Eq. (18), and (3) the enforcement of a zero-net translation velocity field.

We find that using zero-initial velocities significantly improves both performance and convergence speed, as shown in Fig. 1 (Left). This improvement may also stem from the ability to use the simplified parameterization from Eq. (19), which is only compatible with zero-initial velocities. As shown in Table 3, this simplified parameterization consistently yields better final performance across different sampling schemes and datasets, while also accelerating convergence (Fig. 1, Center).

Finally, enforcing a zero-net translation velocity field provides additional improvements in terms of MR on the validation set, albeit to a smaller extent. These gains are observed both when using the direct parameterization (Fig. 4) and when combining zero-initial velocities with the simplified parametrization (Fig. 1, Right).

We refer the reader to Appendix E for a more detailed analysis.

6. Conclusion

We introduced KLDM, a novel diffusion model for periodic crystal structure generation, whose key innovation lies in the modeling of the fractional coordinates through a coupling with auxiliary variables representing velocities. We evaluated KLDM on both the CSP and DNG tasks, where it performed on par with, or outperformed, several state-of-the-art (SOTA) models. Notably, on the CSP task, KLDM demonstrated significant improvements on the real-scale datasets MPTS-20 and MPTS-52, at a similar computational cost to the compared models. On the DNG task, it matched current SOTA performance as evaluated by machine-learning interatomic potentials. Further validation with DFT simulations is a natural next step to confirm these findings.

Despite its strong performance, there are opportunities for further improvement of KLDM. Exploring alternative processes to noise the velocity variables in Eq. (12) could lead to better results. Additionally, optimizing the score network architecture to better account for the periodic nature of crystal structures (Lin et al., 2023) and investigating modality-specific noise schedules (Qiu et al., 2025) may result in even further gains. We also envision that incorporating space-group information (Jiao et al., 2024) or Wyck-off positions (Levy et al., 2024) could improve practical performance, particularly on the DNG task.

We believe that KLDM provides a strong foundation for diffusion-based crystal structure generation. Future work will focus on targeted material generation with specific properties, as well as extending KLDM to larger systems such as metal-organic frameworks (MOFs) by incorporating rotational frame modeling (Kim et al., 2025).

Impact statement

Our paper presents a generative model for novel crystalline materials generation. As mentioned in the introduction, the ability to accurately generate novel compounds with targeted properties has significant implications across multiple scientific domains, particularly in molecular discovery and materials design. While advances in materials discovery might benefit society, from more efficient energy storage to improved catalysts, we acknowledge the potential for unintended applications. However, the considerable gap between the model prediction to the successful material synthesis acts as a natural safeguard. Therefore, in the immediate term, our work mostly impacts researchers working on the same topic.

Acknowledgements

The authors would like to thank the anonymous reviewers for their feedback that helped improve the paper. They would also like to thank Johanna Marie Gegenfurtner for insightful discussions on Lie groups.

FC, AB, and JMGL are supported by the Independent Research Fund Denmark (DELIGHT, Grant No. 0217-00326B; "TeraBatt: Data-driven quest for TWh scalable Na-ion battery", Grant No. 2035-00232B). FB and JF are supported by the Novo Nordisk Foundation through the Center for Basic Machine Learning Research in Life Science (MLLS, grant No. NNF20OC0062606). MNS acknowledges support from the Novo Nordisk Foundation ("BNNs for molecular discovery", Grant No. NNF22OC0076658).

References

- Anderson, B. D. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.
- Anstine, D. M. and Isayev, O. Generative models as an emerging paradigm in the chemical sciences. *Journal of the American Chemical Society*, 145(16):8736–8750, 2023.
- Antunes, L. M., Butler, K. T., and Grau-Crespo, R. Crystal structure generation with autoregressive large language modeling. *Nature Communications*, 15(1):1–16, 2024.
- Arvanitogeorgos, A. *An introduction to Lie groups and the geometry of homogeneous spaces*, volume 22. American Mathematical Soc., 2003.
- Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.
- Bilodeau, C., Jin, W., Jaakkola, T., Barzilay, R., and Jensen, K. F. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 12(5):e1608, 2022.
- Castelli, I. E., Landis, D. D., Thygesen, K. S., Dahl, S., Chorkendorff, I., Jaramillo, T. F., and Jacobsen, K. W. New cubic perovskites for one-and two-photon water splitting using the computational materials repository. *Energy & Environmental Science*, 5(10):9034–9043, 2012.
- Chen, R. T. Q. and Lipman, Y. Flow matching on general geometries. In *The Twelfth International Conference on Learning Representations*, 2024.
- Chen, T., Zhang, R., and Hinton, G. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations*, 2023.
- Cornet, F. R., Bartosh, G., Schmidt, M. N., and Naesseth, C. A. Equivariant neural diffusion for molecule generation. *Advances in Neural Information Processing Systems*, 37, December 2024.
- Court, C. J., Yildirim, B., Jain, A., and Cole, J. M. 3-d inorganic crystal structure generation and property prediction via representation learning. *Journal of Chemical Information and Modeling*, 60(10):4518–4535, 2020.
- Davies, D. W., Butler, K. T., Jackson, A. J., Skelton, J. M., Morita, K., and Walsh, A. Smact: Semiconducting materials by analogy and chemical theory. *Journal of Open Source Software*, 4(38):1361, 2019.
- De Bortoli, V., Mathieu, E., Hutchinson, M., Thornton, J., Teh, Y. W., and Doucet, A. Riemannian score-based generative modelling. *Advances in Neural Information Processing Systems*, 35:2406–2422, 2022.
- Dockhorn, T., Vahdat, A., and Kreis, K. Score-based generative modeling with critically-damped langevin diffusion. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.
- Flam-Shepherd, D. and Aspuru-Guzik, A. Language models can generate molecules, materials, and protein binding sites directly in three dimensions as xyz, cif, and pdb files. *arXiv preprint arXiv:2305.05708*, 2023.
- Fréchet, M. Les éléments aléatoires de nature quelconque dans un espace distancié. In *Annales de l'institut Henri Poincaré*, volume 10, pp. 215–310, 1948.
- Glass, C. W., Oganov, A. R., and Hansen, N. Uspeh—evolutionary crystal structure prediction. *Computer physics communications*, 175(11-12):713–720, 2006.

- Gruver, N., Sriram, A., Madotto, A., Wilson, A. G., Zitnick, C. L., and Ulissi, Z. W. Fine-tuned language models generate stable inorganic materials as text. In *The Twelfth International Conference on Learning Representations*, 2024.
- Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- Hoffmann, J., Maestrati, L., Sawada, Y., Tang, J., Sellier, J. M., and Bengio, Y. Data-driven approach to encoding and decoding 3-d crystal structures. *arXiv preprint arXiv:1909.00949*, 2019.
- Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022.
- Jain, A., Ong, S. P., Hautier, G., Chen, W., Richards, W. D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1), 2013.
- Jiao, R., Huang, W., Lin, P., Han, J., Chen, P., Lu, Y., and Liu, Y. Crystal structure prediction by joint equivariant diffusion. *Advances in Neural Information Processing Systems*, 36, 2023.
- Jiao, R., Huang, W., Liu, Y., Zhao, D., and Liu, Y. Space group constrained crystal generation. In *The Twelfth International Conference on Learning Representations*, 2024.
- Jing, B., Corso, G., Chang, J., Barzilay, R., and Jaakkola, T. S. Torsional diffusion for molecular conformer generation. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.
- Joshi, C. K., Bodnar, C., Mathis, S. V., Cohen, T., and Lio, P. On the expressive power of geometric graph neural networks. In *International conference on machine learning*, pp. 15330–15355. PMLR, 2023.
- Kim, N., Kim, S., Kim, M., Park, J., and Ahn, S. Moflow: Flow matching for structure prediction of metal-organic frameworks. In *The Thirteenth International Conference on Learning Representations, ICLR*, 2025.
- Kim, S., Noh, J., Gu, G. H., Aspuru-Guzik, A., and Jung, Y. Generative adversarial networks for crystal structure prediction. *ACS central science*, 6(8):1412–1420, 2020.
- Kohn, W. and Sham, L. J. Self-consistent equations including exchange and correlation effects. *Physical review*, 140(4A):A1133, 1965.
- Kong, L. and Tao, M. Convergence of kinetic langevin monte carlo on lie groups. In Agrawal, S. and Roth, A. (eds.), *The Thirty Seventh Annual Conference on Learning Theory (COLT)*, volume 247, pp. 3011–3063. PMLR, 2023.
- Levy, D., Panigrahi, S. S., Kaba, S.-O., Zhu, Q., Galkin, M., Miret, S., and Ravanbakhsh, S. SymmCD: Symmetry-preserving crystal generation with diffusion models. In *AI for Accelerated Materials Design - NeurIPS 2024*, 2024.
- Lin, P., Chen, P., Jiao, R., Mo, Q., Jianhuan, C., Huang, W., Liu, Y., Huang, D., and Lu, Y. Equivariant diffusion for crystal structure prediction. In *Forty-first International Conference on Machine Learning*, 2024.
- Lin, Y., Yan, K., Luo, Y., Liu, Y., Qian, X., and Ji, S. Efficient approximations of complete interatomic potentials for crystal property prediction. In *International conference on machine learning*, pp. 21260–21287. PMLR, 2023.
- Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.
- Lippe, P., Veeling, B. S., Perdikaris, P., Turner, R. E., and Brandstetter, J. PDE-refiner: Achieving accurate long rollouts with neural PDE solvers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- Lou, A., Xu, M., Farris, A., and Ermon, S. Scaling riemannian diffusion models. *Advances in Neural Information Processing Systems*, 36:80291–80305, 2023.
- Lyngby, P. and Thygesen, K. S. Data-driven discovery of 2d materials by deep generative models. *npj Computational Materials*, 8(1):232, 2022.
- Merchant, A., Batzner, S., Schoenholz, S. S., Aykol, M., Cheon, G., and Cubuk, E. D. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023.
- Miller, B. K., Chen, R. T., Sriram, A., and Wood, B. M. Flowmm: Generating materials with riemannian flow matching. In *Forty-first International Conference on Machine Learning*, 2024.
- Nouira, A., Sokolovska, N., and Crivello, J.-C. Crystalgan: learning to discover crystallographic structures with generative adversarial networks. *arXiv preprint arXiv:1810.11203*, 2018.
- Oganov, A. R., Pickard, C. J., Zhu, Q., and Needs, R. J. Structure prediction drives materials discovery. *Nature Reviews Materials*, 4(5):331–348, 2019.

- Ong, S. P., Richards, W. D., Jain, A., Hautier, G., Kocher, M., Cholia, S., Gunter, D., Chevrier, V. L., Persson, K. A., and Ceder, G. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319, 2013.
- Petersen, M. H., Zhu, R., Dai, H., Aggarwal, S., Wei, N., Chen, A. P., Bhowmik, A., Lastra, J. M. G., and Hippalgaonkar, K. Dis-CSP: Disordered crystal structure predictions. In *AI for Accelerated Materials Design - ICLR 2025*, 2025.
- Pickard, C. J. Airss data for carbon at 10gpa and the c+n+h+o system at 1gpa. *Materials Cloud Archive*, doi: 10.24435/materialscloud:2020.0026/v1, 2020.
- Pickard, C. J. and Needs, R. Ab initio random structure searching. *Journal of Physics: Condensed Matter*, 23(5): 053201, 2011.
- Qiu, K., Song, Y., Fan, Z., Liu, P., Zhang, Z., Zheng, M., Zhou, H., and Ma, W.-Y. Piloting structure-based drug design via modality-specific optimal schedule. *arXiv preprint arXiv:2505.07286*, 2025.
- Ren, Z., Tian, S. I. P., Noh, J., Oviedo, F., Xing, G., Li, J., Liang, Q., Zhu, R., Aberle, A. G., Sun, S., et al. An invertible crystallographic representation for general inverse design of inorganic crystals with targeted properties. *Matter*, 5(1):314–335, 2022.
- Rozet, F. and Louppe, G. Score-based data assimilation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- Särkkä, S. and Solin, A. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.
- Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.
- Shi, J., Han, K., Wang, Z., Doucet, A., and Titsias, M. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.
- Shysheya, A., Diaconu, C., Bergamin, F., Perdikaris, P., Hernández-Lobato, J. M., Turner, R. E., and Mathieu, E. On conditional diffusion models for PDE simulations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.
- Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.
- Sriram, A., Miller, B. K., Chen, R. T. Q., and Wood, B. M. FlowLLM: Flow matching for material generation with large language models as base distributions. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- Tao, M. and Ohsawa, T. Variational optimization on lie groups, with examples of leading (generalized) eigenvalue problems. In *International Conference on Artificial Intelligence and Statistics*, pp. 4269–4280. PMLR, 2020.
- Wang, Y., Lv, J., Zhu, L., and Ma, Y. Crystal structure prediction via particle-swarm optimization. *Physical Review B—Condensed Matter and Materials Physics*, 82(9): 094116, 2010.
- Ward, L., Agrawal, A., Choudhary, A., and Wolverton, C. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2(1):1–7, 2016.
- Xie, T. and Grossman, J. C. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018.
- Xie, T., Fu, X., Ganea, O.-E., Barzilay, R., and Jaakkola, T. S. Crystal diffusion variational autoencoder for periodic material generation. In *International Conference on Learning Representations*, 2022.
- Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S., and Tang, J. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022.
- Xu, M., Powers, A. S., Dror, R. O., Ermon, S., and Leskovec, J. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pp. 38592–38610. PMLR, 2023.
- Yang, H., Hu, C., Zhou, Y., Liu, X., Shi, Y., Li, J., Li, G., Chen, Z., Chen, S., Zeni, C., et al. Mattersim: A deep learning atomistic model across elements, temperatures and pressures. *arXiv preprint arXiv:2405.04967*, 2024a.
- Yang, S., Cho, K., Merchant, A., Abbeel, P., Schuurmans, D., Mordatch, I., and Cubuk, E. D. Scalable diffusion for materials generation. In *The Twelfth International Conference on Learning Representations*, 2024b.
- Zeni, C., Pinsler, R., Zügner, D., Fowler, A., Horton, M., Fu, X., Wang, Z., Shysheya, A., Crabbé, J., Ueda, S., Sordillo,

R., Sun, L., Smith, J., Nguyen, B., Schulz, H., Lewis, S., Huang, C.-W., Lu, Z., Zhou, Y., Yang, H., Hao, H., Li, J., Yang, C., Li, W., Tomioka, R., and Xie, T. A generative model for inorganic materials design. *Nature*, Jan 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-08628-5.

Zhu, Y., Chen, T., Kong, L., Theodorou, E. A., and Tao, M. Trivialized momentum facilitates diffusion generative modeling on lie groups. *arXiv preprint arXiv:2405.16381*, 2024.

Zimmermann, N. E. and Jain, A. Local structure order parameters and site fingerprints for quantification of coordination environment and crystal structure similarity. *RSC advances*, 10(10):6063–6081, 2020.

Organization of the Supplementary Material

Appendix A We provide a short informal introduction to Lie groups and manifolds. We provide intuition behind key equations Eqs. (12) and (13), and the TDM and KLDM models more broadly.

Appendix B We discuss why data on a torus can be represented either as angles or rotation matrices. We also show that, in this case, the matrix exponential simplifies to a rotation matrix, enabling a coordinate update that corresponds to a translation following by a wrapping operation.

Appendix C We derive, step-by-step, the target used to train the score network in KLDM.

Appendix D We analyze how enforcing a zero-net translation constraint affects the mean of the system.

Appendix E We provide an additional discussion and analysis of the ablation study.

Appendix F and Appendix G We present additional results for the CSP and DNG tasks, respectively.

Appendix H We present pseudo-code for all the core algorithms in KLDM.

Appendix I Experimental settings required for reproducibility, along with a brief discussion of the differences between KLDM and the baseline methods.

A. A primer on Lie groups and manifolds

In this section, we provide an intuitive overview of key concepts used in the main text. For crystalline materials, the fractional coordinates $\mathbf{f}$, which define atomic positions within the unit cell, lie on a hypertorus: *i.e.* $\mathbf{f} \in [0, 1)^{3 \times K} \cong \mathbb{T}^{3 \times K}$. The hypertorus is a direct product of Lie groups. In the main paper, we introduced Lie groups as smooth manifolds equipped with a group structure G and smooth group operations. Here, we revisit some of the building concepts underlying KLDM in more detail, from a Riemannian geometry perspective rather than from the Lie group one.

Lie group In mathematics, a *group* is denoted by $(G, \cdot)$, with G being the non-empty set of elements that belong to the group and $\cdot$ being the group operation that combines any two elements a, b in the group G and results in an element of G . The group operation $\cdot$ has to satisfy three different properties:

- $(a \cdot b) \cdot c = a \cdot (b \cdot c) \quad \forall a, b, c \in G$ (*associativity property*)
- For every element $a \in G$, there should be a unique element $e \in G$, such that $e \cdot a = a$ and $a \cdot e = a$ (existence of a unique *identity element*)
- For every element $a \in G$ there is an element $b \in G$ such that $a \cdot b = e$ and $b \cdot a = e$. (existence of an *inverse element*)

If the group operation $\cdot$ is also *commutative*, *i.e.* $a \cdot b = b \cdot a \quad \forall a, b \in G$, then the group is called an Abelian group. This property holds in the case of the hypertorus $\mathbb{T}^{3 \times K}$ and allows us to define the transition kernel in Eq. (16) in closed-form (and hence a simulation-free training procedure). For a complete derivation of this, we refer to [Zhu et al. \(2024\)](#).

We can formalize two different operations that can be done with elements of the group G that will be useful to give some intuition about why data on a manifold can be modeled through quantities defined in Euclidean space. Let us consider two elements $a, g \in G$ of the group. The *left translation* of g by a is defined as the operation $L_a : G \rightarrow G$ which takes g as input and returns $a \cdot g$, or equivalently $g \mapsto a \cdot g$. In the same way, we can define the right translation as $R_a : G \rightarrow G, g \mapsto g \cdot a$.

Tangent space Before describing other key elements of a Lie group, we introduce the concept of *tangent space*, which stems from the fact that a Lie group is also a manifold. Possible ways to understand the notion of a manifold involve thinking about it as a hypersurface embedded in a higher-dimensional space or as a collection of points that are somewhat connected to generate a surface. A tangent space is a vector space containing all the vectors that are tangential to a specific point in the manifold, *i.e.* any element $g \in G$ in the case of a Lie group. It is usually denoted as $\mathcal{T}_g \mathcal{M}$ for manifolds and $\mathcal{T}_g G$ for Lie Groups. The tangent space offers a linearized view of the manifold in the neighborhood of the point g and therefore it is usually considered as Euclidean space—*i.e.* $\mathcal{T}_g G \cong \mathbb{R}^d$, with d being the dimension of g . If we consider all the elements in the group G , the set of all the tangent spaces associated with these elements form the *tangent bundle*, denoted with $\mathcal{T}\mathcal{M}$ or $\mathcal{T}G$, depending if we are working with a manifold or a Lie group.

Lie algebra, left-invariant vector fields and exponential maps The Lie algebra $\mathfrak{g}$ is defined as the tangent space at the identity element of the group, $e \in G$: $\mathcal{T}_e G$. Since tangent spaces are approximately Euclidean, we treat the Lie algebra $\mathfrak{g}$ as Euclidean as well.

To understand how to define the dynamics in Eq. (12) and introduced by Zhu et al. (2024), we first introduce *left-invariant vector fields*.

A vector field, $X : G \rightarrow \mathcal{T}G$, is a function that associates every element of the group $g \in G$ to a vector $X_g \in \mathcal{T}_g G$ of the tangent space at g .

Recall the left-translation operation $L_a : G \rightarrow G$, which maps a group element $g \in G$ to $a \cdot g \in G$ for any group element $a \in G$. The differential of this operation, denoted as $(dL_a)_g : \mathcal{T}_g G \rightarrow \mathcal{T}_{a \cdot g} G$, acts on tangent vectors instead of group elements. A vector field is *left-invariant* if:

$$(dL_a)_g(X_g) = X_{a \cdot g}, \quad \text{for any two group elements } a, g \in G. \quad (20)$$

Intuitively, this means that the vector field behaves predictably under left-translations. Specifically, if g is translated to $a \cdot g$, the vector $X_g \in \mathcal{T}_g G$ will be mapped to $X_{a \cdot g}$. Thus, a left-invariant vector looks the same at all points in the group.

In particular, it can be shown that there exists a linear isomorphism between the Lie algebra $\mathfrak{g}$ and the tangent space of the identity element $e \in G$, i.e. $\mathfrak{g} \cong \mathcal{T}_e G$ (Arvanitogeorgos, 2003). This means that an invariant vector field is determined entirely by its value on the tangent space of the identity element $e \in G$. Using this, we can rewrite Eq. (20) in terms of the identity element $e \in G$ and its tangent space $\mathcal{T}_e G$, which is $\mathfrak{g}$:

$$\begin{aligned} (dL_a)_e : \mathcal{T}_e G &\rightarrow \mathcal{T}_{a \cdot e} G = \mathcal{T}_a G \\ (dL_a)_e(X_e) &= X_{a \cdot e} = X_a \end{aligned} \quad (21)$$

where $a \cdot e = a$. Eq. (21) shows that any vector $X_e \in \mathcal{T}_e G$ can be translated to a tangent space $X_a \in \mathcal{T}_a G$ using $(dL_a)_e(X_e)$. Now, with both the group element $a \in G$ and the tangent vector $X_a \in \mathcal{T}_a G$, we can use the *Riemannian exponential map*⁴ $\exp_a : \mathcal{T}_a G \rightarrow G$ to compute the group element $a' \in G$ reached by starting from a with velocity X_a in one unit time. This map extends the Euclidean notion of motion along a straight line to the manifold.

How does this relate to modeling data living on the torus? We can take advantage of the torus being a Lie group and use the operations from the previous section. Specifically, we use the Lie algebra $\mathfrak{g}$ (a Euclidean vector space) and the operation mapping vectors in $\mathfrak{g}$ to the tangent space of any point of the manifold (via Eq. (21)). In the forward dynamics of KLDM (Eq. (12)), we couple fractional coordinates (elements of our Lie group) with velocity vectors in $\mathfrak{g}$. Since the torus is a Lie group, we can map any vector in $\mathfrak{g}$ the tangent space of any group element. This allows us to define a standard Euclidean diffusion model over these velocities in $\mathfrak{g}$ and use Eq. (20) to get the corresponding tangential velocities. By applying the exponential map, we can update the fractional coordinates, explaining the coupled process in Eq. (12).

The computation of Eq. (21) for the torus is straightforward, as both the vectors in the Lie algebra $\mathfrak{g}$ and group elements can be represented as $d \times d$ matrices. The differential operation in Eq. (21) simplifies to matrix multiplication (Zhu et al., 2024). Given a vector $X_e \in \mathfrak{g}$ and an element $a \in G$, the tangent vector in a is simply $X_a = aX_e \in \mathcal{T}_a G$.

Additionally, the Riemannian exponential map, involving constant velocity, has a closed-form solution using the matrix exponential. The group element a' reached from a with velocity X_a is given by $a' = \exp_a(aX_e) = \exp_a(X_a) = a \expm(X_e)$, where $\expm(X_e)$ denotes the matrix exponential.

B. Sampling from the noising and denoising processes

B.1. Intuition about the correspondence between $\mathbb{T}$ and $\text{SO}(2)$

In the following, we provide an intuitive explanation of the isomorphism between $\mathbb{T}$ and $\text{SO}(2)$. Let us consider a variable $x \in [a, b]$, we are going to work on an *alternative representation* thereof, that we will call g – a representation in $\text{SO}(2)$.

⁴Note that the exponential map operation can be defined from a Riemannian manifold perspective and from the Lie group one. In the Riemannian case, we have that it is defined as $\exp_a : \mathcal{T}_a G \rightarrow G$, i.e. it takes a velocity living in the tangent space of an element of the manifold (which in this case is also a group) and map it to another element of the manifold. The Lie group exponential map instead is a function $\exp_e : \mathfrak{g} \rightarrow G$ that takes a vector in the Lie algebra $\mathfrak{g} \cong \mathcal{T}_e G$ and maps it to the group.

First, we are going to map x to an angle θ as follows,

$$\theta = 2\pi \left(\frac{x}{b-a} - \frac{1}{2} \right),$$

such that $\theta \in [-\pi, \pi)$.

Then, we can construct the representation g as,

$$g = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}.$$

From g , we can readily recover θ ,

$$\theta = \text{sign}(g_{0,1}) \cdot \arccos g_{0,0},$$

and in turn x ,

$$x = \left(\frac{\theta}{2\pi} + \frac{1}{2} \right) (b-a).$$

B.2. Update of positions is equivalent to periodic translation

As we have mentioned in the previous section, the update of the coordinate in the forward process can be expressed as $\mathbf{f}_t = \mathbf{f}_0 \expm(\mathbf{r}_t)$ where then $\mathbf{r}_t \sim \text{WN}(\mathbf{r}_t | \mu_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2)$. Therefore, the update involves a matrix exponential operation. In the case of the torus, we have that the structure of $\mathbf{r}_t$ can be written as a 2×2 skew-symmetric matrix of the form:

$$\mathbf{r}_t = \begin{bmatrix} 0 & r_t \\ -r_t & 0 \end{bmatrix}, \quad r_t \in \mathbb{R}$$

In our case, we are interested in computing the matrix exponential of $\mathbf{r}_t$. For simplicity, if we assume that $r_t = 1$, we are interested in computing a matrix exponential with the following structure:

$$\expm(At) = \expm \left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} t \right)$$

This corresponds to the following infinite sum

$$\expm(At) = \expm \left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} t \right) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & t \\ -t & 0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} -t^2 & 0 \\ 0 & -t^2 \end{bmatrix} + \frac{1}{3!} \begin{bmatrix} 0 & -t^3 \\ t^3 & 0 \end{bmatrix} + \frac{1}{4!} \begin{bmatrix} t^4 & 0 \\ 0 & t^4 \end{bmatrix} + \dots$$

which can be rewritten as follows:

$$\expm \left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} t \right) = \begin{bmatrix} 1 - \frac{t^2}{2} + \frac{t^4}{4!} - \dots & t - \frac{t^3}{3!} + \frac{t^5}{5!} - \dots \\ -t + \frac{t^3}{3!} - \frac{t^5}{5!} + \dots & 1 - \frac{t^2}{2} + \frac{t^4}{4!} - \dots \end{bmatrix}$$

where we can recognize the Maclaurin series for $\sin(t)$ and $\cos(t)$. Therefore, we can conclude that:

$$\expm \left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} t \right) = \begin{bmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{bmatrix}$$

In forward and backward integration of the dynamics described by Eq. (7) and Eq. (13) the update of the coordinates involves the computation of a matrix exponential. In the following, we are going to derive step-by-step the closed-form solution for the matrix exponential, highlighting how this will be equivalent to a translation and a wrap of the current position.

Forward process In the forward process, assuming a discretization step denoted by dt , we are interested in computing the following update:

$$\begin{aligned}
 \mathbf{f}_t &= \mathbf{f}_{t-1} \expm(dt \mathbf{v}_t) \\
 &= \mathbf{f}_{t-1} \expm\left(dt \begin{bmatrix} 0 & v_t \\ -v_t & 0 \end{bmatrix}\right) = \mathbf{f}_{t-1} \expm\left(\begin{bmatrix} 0 & v_t dt \\ -v_t dt & 0 \end{bmatrix}\right) \\
 &= \mathbf{f}_{t-1} \begin{bmatrix} \cos v_t dt & \sin v_t dt \\ -\sin v_t dt & \cos v_t dt \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \cos v_t dt & \sin v_t dt \\ -\sin v_t dt & \cos v_t dt \end{bmatrix} \\
 &= \begin{bmatrix} \cos \theta \cos v_t dt - \sin \theta \sin v_t dt & \cos \theta \sin v_t dt + \sin \theta \cos v_t dt \\ -\sin v_t dt \cos \theta - \cos v_t dt \sin \theta & -\sin \theta \sin v_t dt + \cos \theta \cos v_t dt \end{bmatrix} \\
 &= \begin{bmatrix} \cos(\theta + v_t dt) & \sin(\theta + v_t dt) \\ -\sin(\theta + v_t dt) & \cos(\theta + v_t dt) \end{bmatrix}
 \end{aligned}$$

where we used the rotation matrix representation for the fractional coordinates on the torus, as we explained at the beginning of this section.

Reverse process In the backward process, instead, we are interested in computing the following update:

$$\begin{aligned}
 \mathbf{f}_t &= \mathbf{f}_{t-1} \expm(-dt \mathbf{v}_t) \\
 &= \mathbf{f}_{t-1} \expm\left(-dt \begin{bmatrix} 0 & v_t \\ -v_t & 0 \end{bmatrix}\right) = \mathbf{f}_{t-1} \expm\left(\begin{bmatrix} 0 & -v_t dt \\ v_t dt & 0 \end{bmatrix}\right) \\
 &= \mathbf{f}_{t-1} \begin{bmatrix} \cos v_t dt & -\sin v_t dt \\ \sin v_t dt & \cos v_t dt \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \cos v_t dt & -\sin v_t dt \\ \sin v_t dt & \cos v_t dt \end{bmatrix} \\
 &= \begin{bmatrix} \cos \theta \cos v_t dt + \sin \theta \sin v_t dt & -\cos \theta \sin v_t dt + \sin \theta \cos v_t dt \\ -\cos v_t dt \sin \theta + \sin v_t dt \cos \theta & \sin \theta \sin v_t dt + \cos \theta \cos v_t dt \end{bmatrix} \\
 &= \begin{bmatrix} \cos(\theta - v_t dt) & \sin(\theta - v_t dt) \\ -\sin(\theta - v_t dt) & \cos(\theta - v_t dt) \end{bmatrix}
 \end{aligned}$$

C. Derivation of the target of KLDM

We recall that the transition kernel of our KLDM forward process defined in Eq. (12) can be obtained in closed form and it is given by (Zhu et al., 2024):

$$p_{t|0}(\mathbf{f}_t, \mathbf{v}_t | \mathbf{f}_0, \mathbf{v}_0) = \text{WN}(\logm(\mathbf{f}_0^{-1} \mathbf{f}_t) | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I}) \cdot \mathcal{N}_{\mathbf{v}}(\mathbf{v}_t | \boldsymbol{\mu}_{\mathbf{v}_t}, \sigma_{\mathbf{v}_t}^2 \mathbf{I}),$$

where the mean and the variance of the two distributions are the following:

$$\boldsymbol{\mu}_{\mathbf{r}_t} = \frac{1 - e^{-t}}{1 + e^{-t}} (\mathbf{v}_t + \mathbf{v}_0) \quad \boldsymbol{\mu}_{\mathbf{v}_t} = e^{-t} \mathbf{v}_0 \quad (22)$$

$$\sigma_{\mathbf{r}_t}^2 = 2t + \frac{8}{e^t + 1} - 4 \quad \sigma_{\mathbf{v}_t}^2 = 1 - e^{-2t} \quad (23)$$

Intuitively, the normal distribution describes how we should noise the initial velocity $\mathbf{v}_0$ to get a sample $\mathbf{v}_t$, while the wrapped-normal distribution implicitly defines how to get a noisy sample for the fractional coordinates $\mathbf{f}_t$ starting from $\mathbf{f}_0$. The trick is to define $\mathbf{r}_t = \logm(\mathbf{f}_0^{-1} \mathbf{f}_t)$, and by taking the matrix-exponential on both sides, we can get the following update rule $\mathbf{f}_t = \mathbf{f}_0 \expm(\mathbf{r}_t)$ where then $\mathbf{r}_t \sim \text{WN}(\mathbf{r}_t | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I})$

The target score used in the DSM loss (Eq. (9)) for KLDM is $\nabla_{\mathbf{v}_t} \log p_{t|0}(\mathbf{f}_t, \mathbf{v}_t | \mathbf{f}_0, \mathbf{v}_0)$. In the following, we derive it step-by-step, highlighting at the end the parameterization used to train our *score network*. The target score can therefore be

computed as follows:

$$\begin{aligned}\nabla_{\mathbf{v}_t} \log p_{t|0}(\mathbf{f}_t, \mathbf{v}_t | \mathbf{f}_0, \mathbf{v}_0) &= \nabla_{\mathbf{v}_t} \log [\text{WN}(\log(\mathbf{f}_0^{-1} \mathbf{f}_t) | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I}) \cdot \mathcal{N}_{\mathbf{v}}(\mathbf{v}_t | \boldsymbol{\mu}_{\mathbf{v}}, \sigma_{\mathbf{v}}^2 \mathbf{I})] \\ &= \underbrace{\nabla_{\mathbf{v}_t} \log \text{WN}(\log(\mathbf{f}_0^{-1} \mathbf{f}_t) | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I})}_{\mathbf{s}_c \text{ coordinates term}} + \underbrace{\nabla_{\mathbf{v}_t} \log \mathcal{N}_{\mathbf{v}}(\mathbf{v}_t | \boldsymbol{\mu}_{\mathbf{v}}, \sigma_{\mathbf{v}}^2 \mathbf{I})}_{\mathbf{s}_v \text{ velocity term}}\end{aligned}\quad (24)$$

We start by deriving the score for the second term $\mathbf{s}_v$, which can be expressed in multiple way:

$$\mathbf{s}_v = \frac{-\mathbf{v}_t + \boldsymbol{\mu}_{\mathbf{v}}}{\sigma_{\mathbf{v}}^2} = -\frac{\boldsymbol{\varepsilon}}{\sigma_{\mathbf{v}}} \quad (25)$$

We can now focus on the $\mathbf{s}_c$ term, where for simplicity we define $\mathbf{r}_t = \log(\mathbf{f}_0^{-1} \mathbf{f}_t)$ as above:

$$\begin{aligned}\mathbf{s}_c &= \nabla_{\mathbf{v}_t} \log \text{WN}(\log(\mathbf{f}_0^{-1} \mathbf{f}_t) | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I}) = \nabla_{\mathbf{v}_t} \log \text{WN}(\mathbf{r}_t | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I}) \\ &= \nabla_{\mathbf{v}_t} \log \left(\frac{1}{\sqrt{2\pi}\sigma_{\mathbf{r}_t}} \sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \right) \\ &= \nabla_{\mathbf{v}_t} \log \left(\sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \right) \\ &= \frac{1}{\underbrace{\sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right)}_C} \nabla_{\mathbf{v}_t} \left[\sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \right] \\ &= C \cdot \left[\sum_{k=-\infty}^{+\infty} \nabla_{\mathbf{v}_t} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \right] \\ &= C \cdot \left[\sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \nabla_{\mathbf{v}_t} \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \right] \\ &= C \cdot \left[\sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \nabla_{\mathbf{v}_t} \left(-\frac{(\mathbf{r}_t - \frac{1-e^{-t}}{1+e^{-t}}(\mathbf{v}_t + \mathbf{v}_0) + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \right] \\ &= C \cdot \left[\sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \frac{1-e^{-t}}{1+e^{-t}} \frac{1}{\sigma_{\mathbf{r}_t}^2} \left(\mathbf{r}_t - \frac{1-e^{-t}}{1+e^{-t}}(\mathbf{v}_t + \mathbf{v}_0) + 2\pi k \right) \right] \\ &= \frac{1}{\sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right)} \left[\sum_{k=-\infty}^{+\infty} \frac{1-e^{-t}}{1+e^{-t}} \frac{1}{\sigma_{\mathbf{r}_t}^2} \left(\mathbf{r}_t - \frac{1-e^{-t}}{1+e^{-t}}(\mathbf{v}_t + \mathbf{v}_0) + 2\pi k \right) \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \right] \\ &= \frac{1}{\sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right)} \left[\sum_{k=-\infty}^{+\infty} \frac{1-e^{-t}}{1+e^{-t}} \frac{1}{\sigma_{\mathbf{r}_t}^2} \left(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k \right) \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \right] \\ &= \frac{1-e^{-t}}{1+e^{-t}} \frac{1}{\sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right)} \left[\sum_{k=-\infty}^{+\infty} \frac{1}{\sigma_{\mathbf{r}_t}^2} \left(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k \right) \exp \left(-\frac{(\mathbf{r}_t - \boldsymbol{\mu}_{\mathbf{r}_t} + 2\pi k)^2}{2\sigma_{\mathbf{r}_t}^2} \right) \right] \\ &= \frac{1-e^{-t}}{1+e^{-t}} \nabla_{\boldsymbol{\mu}_{\mathbf{r}_t}} \log \text{WN}(\mathbf{r}_t | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I}) \\ &= \frac{\partial \boldsymbol{\mu}_{\mathbf{r}_t}}{\partial \mathbf{v}_t} \nabla_{\boldsymbol{\mu}_{\mathbf{r}_t}} \log \text{WN}(\mathbf{r}_t | \boldsymbol{\mu}_{\mathbf{r}_t}, \sigma_{\mathbf{r}_t}^2 \mathbf{I})\end{aligned}\quad (26)$$

Therefore, the target score is given by summing Eq. (25) and Eq. (26) together. If we now assume that the velocities are

zeros at time $t = 0$, we can notice that the score s_v can be rewritten as follows:

$$s_v = \frac{-v_t + \mu_v}{\sigma_v^2} = \frac{-v_t + e^{-t}v_0}{\sigma_v^2} = -\frac{v_t}{\sigma_v^2} \quad (27)$$

where we used the definition of $\mu_v = e^{-t}v_0$ and the fact that we assumed that $v_0 = \mathbf{0}$, resulting in $\mu_v = \mathbf{0}$.

D. Effect of the zero-net translation velocity field

Mismatch between translation-invariant score parameterization and non-invariant training target As noted by Lin et al. (2024), there is a potential mismatch between the translation-invariant parameterization of the score and the non-invariant training target. For instance, a noisy point cloud and its periodic translation are considered equivalent by the network, but the target scores, as computed in Eq. (18) and used by DIFFCSP (Jiao et al., 2023), will differ. While this mismatch is averaged out during training and does not prevent the model from learning a useful score (as DIFFCSP performs well on the tasks), it still affects gradient variance during training.

Constraining the velocity field In Section 3.2, we enforce a velocity field with zero net translation by imposing zero-mean velocities ($\sum_k^K v_k = \mathbf{0}$). Since our score network is periodic translation-invariant, the goal is to remove any translation component from the dynamics, as the network cannot distinguish between translations.

To illustrate this, consider the simple case of a one-dimensional system with a single atom (*i.e.* f_0 is just one coordinate). In this scenario, any noisy f_t can be seen as a periodic translation of any other f'_t , and hence even of the clean sample f_0 itself. Without constraining the velocity field, the dynamics defined in Eq. (12) produce noisy samples f_t corresponding to periodic translations of f_0 , which the network cannot distinguish. By imposing a zero-mean velocity field, the velocity coupled with f_0 must, in this case, be zero, preventing the forward dynamics from introducing an overall translation of the crystal.

Effect of the constraint on forward dynamics To understand the impact of this constraint on the forward dynamics in the general case, we track the *mean* of the noisy samples generated by using the transition kernel defined in Eq. (16). Since our data resides on a torus, the appropriate generalization of the concept of arithmetic mean to non-Euclidean spaces is the Fréchet mean (Fréchet, 1948). The Fréchet mean ensures that the resulting point lies on the manifold but requires solving an optimization problem. Given N datapoints $\{f_0, \dots, f_N\} \in \mathcal{M}$ on a manifold, we define the *Karcher means* as the set of solutions to:

$$f_{\text{mean}} = \arg \min_{p \in \mathcal{M}} \sum_{i=1}^N d^2(p, f_i), \quad (28)$$

where $d(\cdot, \cdot)$ is the geodesic distance on the manifold. When the optimization yields a unique solution, it is the Fréchet mean.

We consider a one-dimensional crystal composed of 10 atoms and two different noise levels, namely $\sigma^2 = 0.1$ and $\sigma^2 = 0.7$. We sample 1000 points from the transition kernel defined in Eq. (16). For each sample, we check whether the original Fréchet mean is preserved when enforcing a zero-net velocity field. As shown in Fig. 2, at low noise levels, the Fréchet mean is generally preserved, indicating that explicitly ensuring a zero-net velocity field prevents the translation of the whole system. However, at higher levels of noise, we note that the Fréchet mean jumps between n different values, each of them separated by $\frac{2\pi}{n}$. Despite this, the noisy samples typically share the same mean as the noiseless reference. In contrast, when no constraint is imposed on the velocity field (Fig. 3), the Fréchet mean is never preserved for both noise levels, and it varies continuously. Our approach, which constrains the velocity field, limits the Fréchet mean to a discrete set of possible values, as opposed to the continuous variation observed without the constraint.

The above analysis shows that sampling from the transition kernel during training can introduce translations, meaning the mismatch may not be fully mitigated across all noise levels. However, we argue that this mismatch is significantly reduced at low noise levels when the vector field constraint is enforced, which benefits training (as shown in Appendix E). Enforcing the constraint also turns the issue of shifting the mean of the system into a discrete problem, rather than a continuous one.

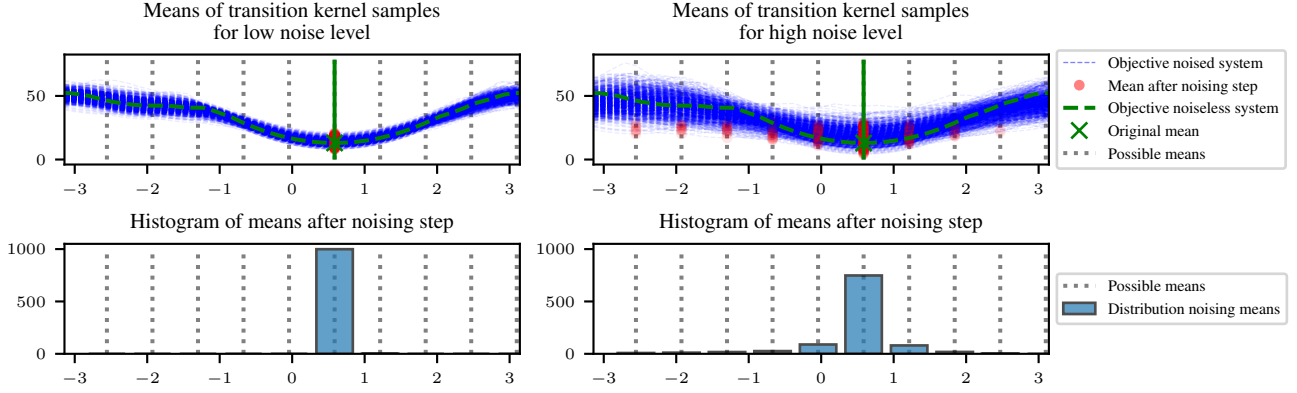


Figure 2: Effect of sampling from the transition kernel in Eq. (16) with a zero-net translation velocity field. We consider a 1D crystal made of 10 atoms and two different noise levels: low noise (sampled from $\mathcal{N}(0, 0.1)$, left) and high noise (sampled from $\mathcal{N}(0, 0.7)$, right). For each noise level, we generate 1000 noisy samples. At low noise, enforcing a zero-net translation velocity field preserves the original mean, preventing any system translation. At higher noise levels, while the mean is typically conserved, it occasionally shifts between 10 distinct values, each separated by $\frac{2\pi}{10}$.

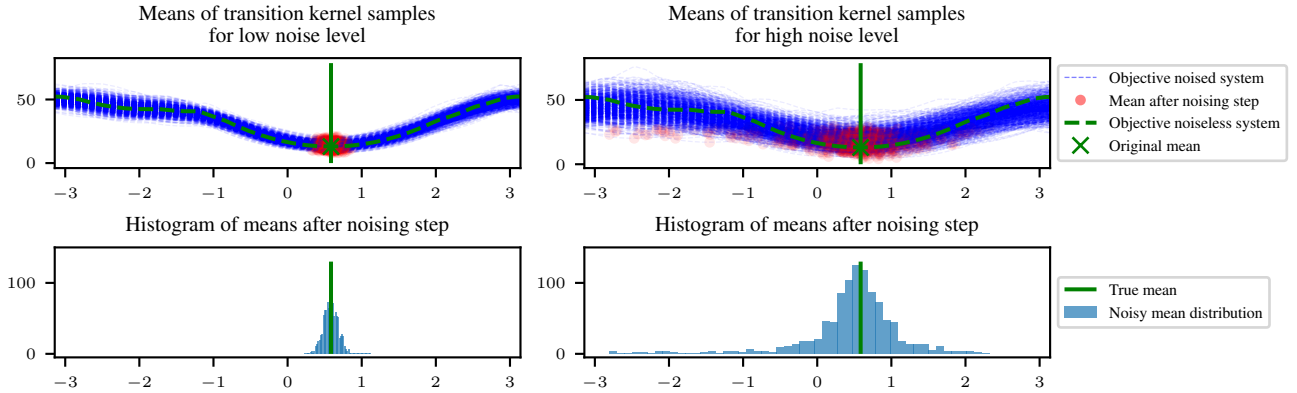


Figure 3: Effect of sampling from the transition kernel in Eq. (16) without a zero-net translation velocity field. We consider a 1D crystal made of 10 atoms and two different noise levels: low noise (sampled from $\mathcal{N}(0, 0.1)$, left) and high noise (sampled from $\mathcal{N}(0, 0.7)$, right). For each noise level, we generate 1000 noisy samples. We observe that the mean is not preserved at either low or high noise levels, as it changes continuously.

E. Ablation study

In this section, we analyze the effect of the different design choices of our method. We focus on the initial velocity distributions, the score parameterization, and the translation-free velocity fields.

Initial-velocity distributions KLDM uses zero-initial velocities, which intuitively corresponds to considering the coordinates $\mathbf{f}$ at time $t = 0$ as being at rest, allowing the noise to gradually propagate from the velocity to these variables. This also allows us to use the simplified parameterization for the score formulated in Eq. (19) ablated above. In Fig. 1 (Left), we compare the choice of using zero initial velocities against having a zero-mean Gaussian distribution with three different variance values. We note there is a strong benefit in using zero initial velocities, potentially explained by the simplified parameterization.

Velocity field The last design choice we consider is the enforcement of a velocity field with a zero-net translation. We analyze the effect of these choices both for zero initial velocities and therefore simplified score parameterization, and in the case of non-zero initial velocities and direct parameterization. We report results in Fig. 1 (Right). By removing the degree of freedom of modeling overall translations of the system, we observe a gain, albeit marginal, in terms of validation set match rate during training in all cases, in particular with non-zero initial velocities, as displayed in Fig. 4.

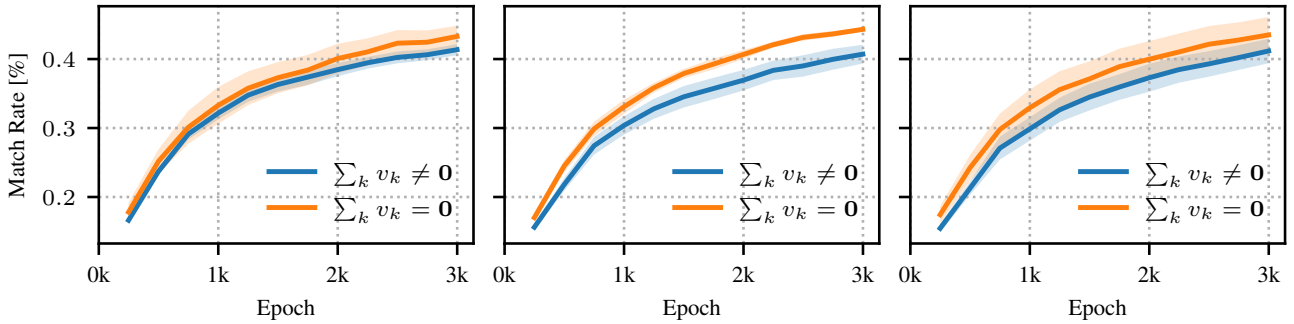


Figure 4: Ablation of velocity fields with zero net translation for non-zero initial velocities. **(Left)** Initial velocities are sampled from a $\mathcal{N}(\mathbf{0}, 0.1 \cdot \mathbf{I})$, i.e. $\sigma_{v_0}^2 = 0.1$. **(Center)** Initial velocities are sampled from a $\mathcal{N}(\mathbf{0}, 0.5 \cdot \mathbf{I})$, i.e. $\sigma_{v_0}^2 = 0.5$. **(Right)** Initial velocities are sampled from a $\mathcal{N}(\mathbf{0}, 1 \cdot \mathbf{I})$, i.e. $\sigma_{v_0}^2 = 1.0$. We report the mean and standard error from the mean of the validation Match Rate over 6 seeds. We note that regarding the initial distribution variance, there is a benefit given by enforcing a zero net translation velocity field.

Score parameterization We compare the simplified parameterization of the score formulated in Eq. (19) compared to the direct parameterization of Eq. (18). We present in Fig. 1, the validation match-rate during training on the MP-20 dataset. From Fig. 1 (Center), we note that the simplified parameterization leads to faster convergence and better performance. The direct parameterization slowly makes the gap smaller if trained for long enough. We also compare fully trained models on MP-20 and MPTS-52 in terms of MR and RMSE on the test set. We note in Table 3, that no matter the sampling scheme, the simplified parameterization always outperforms the direct one, in addition to being much faster to converge.

Table 3: Impact of the score parameterization on the CSP task performance, on the realistic MP-20 and MPTS-52. The simplified parameterization introduced in Eq. (19) improves significantly upon the direct parameterization from Eq. (18).

sampler	MP-20		MPTS-52	
	MR [%] $\uparrow$	RMSE $\downarrow$	MR [%] $\uparrow$	RMSE $\downarrow$
DIRECT PARAMETERIZATION				
EM	56.25 $\pm .4$	0.1071 $\pm .001$	11.55 $\pm .2$	0.2535 $\pm .004$
PC	63.29 $\pm .2$	0.0591 $\pm .002$	15.85 $\pm .2$	0.1666 $\pm .004$
SIMPLIFIED PARAMETERIZATION				
EM	61.72 $\pm .2$	0.0686 $\pm .001$	17.71 $\pm .3$	0.2023 $\pm .005$
PC	65.37 $\pm .1$	0.0455 $\pm .001$	21.46 $\pm .2$	0.1339 $\pm .002$

F. Additional CSP results

Previous work also evaluated their approach on the CARBON-24 (Pickard, 2020) dataset, which consists of carbon-only materials with 6 to 24 atoms per unit cell. Since all structures are composed solely of carbon atoms, computing the MR@1 on the test set is inherently ill-defined—*i.e.* multiple distinct structures may satisfy a specific formula.

For completeness, we report results in Table 4, where KLDM performs comparably to existing methods on the @1 task experiment, but outperforms them on @20. Similar to observations on the PEROV-5 dataset, we find that the PC sampler does not improve sample quality over the standard EM sampler in this setting.

Table 4: Crystal Structure Prediction (CSP) task results on the CARBON-24 dataset. Baseline results are extracted from the respective papers. @ indicates the number of samples considered to evaluate the metrics, *e.g.* @20 indicates the best of 20. Error bars for @1 represent the standard deviation over the mean at sampling time across 20 different seeds. For this dataset, the CSP@1 task is ill-defined due to its one-to-many nature, and we only report the obtained values for completeness.

MODEL	CARBON-24	
	MR [%] $\uparrow$	RMSE $\downarrow$
	METRICS @ 1	
CDVAE	17.09	0.2969
DIFFCSP (PC)	17.54	0.2759
EQUICSP (PC)	—	—
FLOWMM	23.47	0.4122
KLDM- ϵ (EM)	18.04 $\pm .8$	0.3188 $\pm .008$
KLDM- ϵ (PC)	17.26 $\pm .7$	0.2827 $\pm .006$
KLDM- $\mathbf{x}_0$ (EM)	18.58 $\pm .9$	0.3278 ± 0.008
KLDM- $\mathbf{x}_0$ (PC)	17.69 $\pm .8$	0.2915 $\pm .007$
	METRICS @ 20	
CDVAE	88.37	0.2286
DIFFCSP (PC)	88.47	0.2192
FLOWMM	84.15	0.3301
KLDM- ϵ (EM)	90.19	0.2154
KLDM- ϵ (PC)	85.86	0.1988
KLDM- $\mathbf{x}_0$ (EM)	89.66	0.2198
KLDM- $\mathbf{x}_0$ (PC)	84.58	0.2011

G. Additional DNG results

In this section, we present additional results regarding the DNG task. These results were computed using the KLDM- ϵ model trained on the non-standardized values of the lattice parameters and implemented using the ϵ -parameterization for their score. We evaluate samples in terms of validity, coverage, and property statistics. A sample is deemed structurally valid if all pairwise distances are above 0.5Å, while SMACT (Davies et al., 2019) is used to determine compositional validity, by checking the overall charge neutrality. The coverage metrics are computed using fingerprints: CrystalNN structural fingerprints (Zimmermann & Jain, 2020) and Magpie compositional fingerprints (Ward et al., 2016). COV-R (recall) and COV-P (precision) are obtained by comparing the distances between generated and test fingerprints. Finally, the property statistics are obtained by comparing distributions of properties, computed on the generated samples and test set respectively: atomic densities d_ρ and number of unique elements d_{elem} .

We provide results in Table 5, where it can be observed that, unlike for the CSP task, the difference in performance between the compared methods is less pronounced. Nevertheless, KLDM remains competitive with the other models.

Table 5: Results for the De-novo Generation (DNG) task. Baseline results are extracted from the respective papers.

	VALIDITY [%] $\uparrow$		COVERAGE [%] $\uparrow$		PROPERTY $\downarrow$	
	STRUC.	COMP.	COV-R	COV-P	d_p	d_{elem}
PEROV-5						
CDVAE	100.0	98.59	99.45	98.46	0.1258	0.0628
DIFFCSP	100.0	98.85	99.74	98.27	0.1110	0.0128
EQUICSP	100.0	98.60	99.60	98.76	0.1110	0.0503
KLDM-ϵ	99.97	98.83	99.60	98.61	0.2970	0.0478
CARBON-24						
CDVAE	100.0	–	99.80	83.08	0.1407	–
DIFFCSP	100.0	–	99.90	97.27	0.0805	–
EQUICSP	100.0	–	99.75	97.12	0.0734	–
KLDM-ϵ	100.0	–	99.90	98.86	0.0658	–
MP-20						
CDVAE	100.0	86.70	99.15	99.49	0.6875	1.432
DIFFCSP	100.0	83.25	99.71	99.76	0.3502	0.3398
EQUICSP	99.97	82.20	99.65	99.68	0.1300	0.3978
FLOWMM	96.85	83.19	99.49	99.58	0.2390	0.0830
KLDM-ϵ	99.88	84.86	98.94	99.50	0.4658	0.1280

H. Algorithms

This section presents all the algorithms at the core of our method. We show how one can sample the noisy inputs and the training targets in Algorithm 1, we then present the training loop used to train the parameters of the score network $s_\theta(\mathbf{f}_t, \mathbf{v}_t, \mathbf{l}_t, \mathbf{a}_t)$ in Algorithm 2. We then focus on sampling, presenting both a sampling scheme that uses an exponential integrator for simulating the reverse SDE of the dynamics of the velocities $\mathbf{v}_t$ and fractional coordinates $\mathbf{f}_t$ as proposed by Zhu et al. (2024) while using a classic Euler–Maruyama step for lattice parameters $\mathbf{l}_t$ and atom type $\mathbf{a}_t$ in Algorithm 3. In algorithm Algorithm 4, instead, we present how we sample from our model using the predictor-corrector steps as proposed in (Song et al., 2021).

Algorithm 1 `training-targets( $\mathbf{f}, \mathbf{v}, \mathbf{l}, \mathbf{a}, t$ )`: Routine for sampling $\mathbf{f}_t, \mathbf{v}_t, \mathbf{l}_t, \mathbf{a}_t$ from the transition kernels and the corresponding target scores

Require: task (either CSP or DNG), timestep t , scheduler $\alpha(t)$ and $\sigma(t)$ for $\mathbf{l}$ and $\mathbf{a}$, scheduler $\alpha_v(t)$ and $\sigma_v(t)$ for $\mathbf{v}$, scheduler $\mu_{\mathbf{r}_t}(t, \mathbf{v}_0, \mathbf{v}_t)$ and $\sigma_{\mathbf{r}_t}(t, \mathbf{v}_0, \mathbf{v}_t)$ for $\mathbf{r}$. Initial sample $\mathbf{x}_0 = (\mathbf{f}_0, \mathbf{l}_0, \mathbf{a}_0)$ and initial velocities $\mathbf{v}_0$. In our experiments we considered $\mathbf{v}_0 = \mathbf{0}$ (i.e. initial velocities are 0).

Sampling $\mathbf{v}_t$ and $\mathbf{f}_t$

Sample $\epsilon_v \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\epsilon_{\mathbf{r}_t} \sim \mathcal{N}_v(\mathbf{0}, \mathbf{I})$

▷ $\mathcal{N}_v$ is a normal distribution such that $\sum_i \mathbf{v}_i = \mathbf{0}$.

$\mathbf{v}_t = \alpha_v(t)\mathbf{v}_0 + \sigma_v(t) \cdot \epsilon_v$

if initial velocities are zero **then**

$\text{target}_v = -\mathbf{v}_t / \sigma_v^2(t)$

else

$\text{target}_v = -\epsilon_v / \sigma_v(t)$

▷ See Eq. (26).

end if

$\mathbf{r}_t = w(\mu_{\mathbf{r}_t}(t, \mathbf{v}_0, \mathbf{v}_t) + \sigma_{\mathbf{r}_t}(t, \mathbf{v}_0, \mathbf{v}_t) \cdot \epsilon_{\mathbf{r}_t})$

▷ w indicates the wrap function.

$\mathbf{f}_t = w(\mathbf{f}_0 + \mathbf{r}_t)$

$\mathbf{f}_t = \text{center}(\mathbf{f}_t)$

▷ $\text{center}(\cdot)$ keeps the center of gravity fixed.

$\text{target}_s = (1 - \exp(-t)) / (1 + \exp(-t)) \cdot \nabla_{\mathbf{r}(\mathbf{v})} \mathcal{N}_w$

▷ Equivalent computation of Eq. (26).

$\text{target}_v = \text{target}_v + \text{target}_s$

Sampling $\mathbf{l}_t$

Sample $\epsilon_l \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{l}_t = \alpha(t)\mathbf{l}_0 + \sigma(t) \cdot \epsilon_l$

$\text{target}_l = \epsilon_l$

if task is DNG **then**

 ## Sampling $\mathbf{a}_t$

 Sample $\epsilon_a \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{a}_t = \alpha(t)\mathbf{a}_0 + \sigma(t) \cdot \epsilon_a$

$\text{target}_a = \epsilon_a$

return $(\mathbf{v}_t, \mathbf{f}_t, \mathbf{l}_t, \mathbf{a}_t), (\text{target}_v, \text{target}_l, \text{target}_a)$

▷ Return both noisy samples and training targets.

else

return $(\mathbf{v}_t, \mathbf{f}_t, \mathbf{l}_t), (\text{target}_v, \text{target}_l)$

end if

Algorithm 2 Training algorithm

Require: score network $s_\theta(t, \mathbf{f}_t, \mathbf{v}_t, \mathbf{l}_t, \mathbf{a}_t)$, fractional coordinates transition kernel $p_{t|0}(\mathbf{f}_t, \mathbf{v}_t | \mathbf{f}_0, \mathbf{v}_0)$, lattice parameters transition kernel $p_{t|0}(\mathbf{l}_t | \mathbf{l}_0)$, empirical distribution $q(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \delta_{\mathbf{x}_i}$ and distribution over initial velocities $p_0(\mathbf{v})$. In our experiments we considered $p(\mathbf{v}_0) = \delta(\mathbf{v}_0)$ (i.e. initial velocities are 0). Losses weights $\lambda_v, \lambda_l, \lambda_a$. We always used $\lambda_v = 1$ and $\lambda_l = 1$. For DNG task we require also an atom type transition kernel $p_{t|0}(\mathbf{a}_t | \mathbf{a}_0)$.

for training iterations **do**

$\mathbf{x}_0 = \{(\mathbf{f}_0, \mathbf{l}_0, \mathbf{a}_0)\}_{i=1}^B \sim q(\mathbf{x}), t \sim \mathcal{U}(t), \mathbf{v}_0 \sim p_0(\mathbf{v})$ $\triangleright B$ indicates the batch size.

if task is DNG **then**

$(\mathbf{v}_t, \mathbf{f}_t, \mathbf{l}_t, \mathbf{a}_t), (\text{target}_v, \text{target}_l, \text{target}_a) = \text{training_targets}(\mathbf{f}, \mathbf{v}, \mathbf{l}, \mathbf{a}, t)$

$\text{out}_v, \text{out}_l, \text{out}_a = s_\theta(t, \mathbf{f}_t, \mathbf{v}_t, \mathbf{l}_t, \mathbf{a}_t)$ $\triangleright$ The network takes $t, \mathbf{f}_t, \mathbf{v}_t, \mathbf{l}_t, \mathbf{a}_t$ as input and output all the scores.

$\mathcal{L}_a = \|\text{out}_a - \text{target}_a\|_2^2$

else

$(\mathbf{v}_t, \mathbf{f}_t, \mathbf{l}_t), (\text{target}_v, \text{target}_l) = \text{training_targets}(\mathbf{f}, \mathbf{v}, \mathbf{l}, \text{None}, t)$

$\text{out}_v, \text{out}_l = s_\theta(t, \mathbf{f}_t, \mathbf{v}_t, \mathbf{l}_t, \mathbf{a}_t)$

end if

$\mathcal{L}_l = \|\text{out}_l - \text{target}_l\|_2^2$

$\text{out}_v = (1 - \exp(-t)) / (1 + \exp(-t)) \cdot \text{out}_v - \mathbf{v}_t / \sigma_{\mathbf{v}_t}^2$

$\triangleright$ Construct the score, see Eq. (19)

$\mathcal{L}_v = \lambda(t) \|\text{out}_v - \text{target}_v\|_2^2$

$\triangleright \lambda(t)$ computed as [Jiao et al. \(2023\)](#)

if task is DNG **then**

$\mathcal{L}_{\text{tot}} = \lambda_v \mathcal{L}_v + \lambda_l \mathcal{L}_l + \lambda_a \mathcal{L}_a$

$\triangleright$ Depending on how we model the atom types, λ_a has different values

else

$\mathcal{L}_{\text{tot}} = \lambda_v \mathcal{L}_v + \lambda_l \mathcal{L}_l$

end if

Compute gradients of $\mathcal{L}_{\text{tot}}$ with respect to θ and perform a gradient step.

end for

Algorithm 3 Sampling algorithm

Require: trained score network $s_\theta(t, \mathbf{f}_t, \mathbf{v}_t, \mathbf{l}_t, \mathbf{a}_t)$, N discretization steps, $dt = 1/N$ step-size, prior distributions over velocities $p_T(\mathbf{v}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, over fractional coordinates $p_T(\mathbf{f}) = \mathcal{U}(\mathbf{0}, \mathbf{1})$, over lattice parameters $p_T(\mathbf{l}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, and over atom types $p_T(\mathbf{a}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. We require also the knowledge of the forward drift $f(t)$ and the diffusion coefficient $g(t)$ of the SDEs describing the evolution of $\mathbf{l}$ and $\mathbf{a}$.

Note: in the paper we use 0 as index for samples at $t = 0$. However, here it will be a slightly different notation.

Sample from the prior $\mathbf{v}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{f}_0 \sim \mathcal{U}(\mathbf{0}, \mathbf{1})$, $\mathbf{l}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Set $\mathbf{f}_0 = w(\mathbf{f}_0)$

▷ w indicates the wrap function.

if task is DNG **then**

 Sample $\mathbf{a}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

end if

for $n = 1, \dots, N$ **do**

if task is DNG **then**

$\text{out}_v^{(n-1)}, \text{out}_l^{(n-1)}, \text{out}_a^{(n-1)} = s_\theta((1 - (n-1) * dt), \mathbf{f}_{n-1}, \mathbf{v}_{n-1}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

else

$\text{out}_v^{(n-1)}, \text{out}_l^{(n-1)} = s_\theta((1 - (n-1) * dt), \mathbf{f}_{n-1}, \mathbf{v}_{n-1}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

end if

 ## Update step for $\mathbf{v}$ and $\mathbf{f}$

$\text{out}_v = (1 - \exp(-(1 - (n-1)dt))) / (1 + \exp(-(1 - (n-1)dt))) \cdot \text{out}_v - \mathbf{v}_t / \sigma_{v_t}^2$ ▷ Follow Eq. (19)

 Sample $\epsilon_v \sim \mathcal{N}_v(\mathbf{0}, \mathbf{I})$ ▷ $\mathcal{N}_v$ is a normal distribution such that $\sum_i \mathbf{v}_i = \mathbf{0}$.

$\mathbf{v}_n = \exp(dt)\mathbf{v}_{n-1} + 2(\exp(2dt) - 1)\text{out}_v^{(n-1)} + \sqrt{\exp(2dt) - 1}\epsilon_v$ ▷ Update on $\mathbf{v}$

$\mathbf{f}_n = w(\mathbf{f}_{n-1} - \mathbf{v}_n dt)$ ▷ Update on $\mathbf{f}$

 ## Update step for $\mathbf{l}$

 Sample $\epsilon_l \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{l}_n = \mathbf{l}_{n-1} - (f(t) - g^2(t)s(\text{out}_l^{(n-1)}))dt + \sqrt{dt}\epsilon_l$ ▷ EM step for $\mathbf{l}$

if task is DNG **then**

 ## Update step for $\mathbf{a}$

 Sample $\epsilon_a \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{a}_n = \mathbf{a}_{n-1} - (f(t) - g^2(t)s(\text{out}_a^{(n-1)}))dt + \sqrt{dt}\epsilon_a$ ▷ EM step for $\mathbf{a}$

end if

end for

if task is DNG **then**

return A crystalline material sample $(\mathbf{f}_N, \mathbf{l}_N, \mathbf{a}_N)$

else

return A crystalline material sample $(\mathbf{f}_N, \mathbf{l}_N)$

end if

Algorithm 4 Sampling with a single Predictor-Corrector step (PC) similar to [Rozet & Louppe \(2023, Algorithm 4\)](#)

Require: trained score network $s_\theta(t, \mathbf{f}_t, \mathbf{v}_t, \mathbf{l}_t, \mathbf{a}_t)$, N discretization steps, $dt = 1/N$ step-size, prior distributions over velocities $p_T(\mathbf{v}) = \mathcal{N}_v(\mathbf{0}, \mathbf{I})$, over fractional coordinates $p_T(\mathbf{f}) = \mathcal{U}(\mathbf{0}, \mathbf{1})$, over lattice parameters $p_T(\mathbf{l}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, and over atom types $p_T(\mathbf{a}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. We require the scheduler $\alpha_v(t)$ and $\sigma_v(t)$ for $\mathbf{v}$ and also the knowledge of the forward drift $f(t)$ and the diffusion coefficient $g(t)$ of the SDEs describing the evolution of $\mathbf{l}$ and $\mathbf{a}$. We also require a hyperparameter $\tau > 0$.

Note: in the paper we use 0 as index for samples at $t = 0$. However, here it will be a slightly different notation.

Sample from the prior $\mathbf{v}_0 \sim \mathcal{N}_v(\mathbf{0}, \mathbf{I})$, $\mathbf{f}_0 \sim \mathcal{U}(\mathbf{0}, \mathbf{1})$, $\mathbf{l}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

▷ First steps are similar to Algorithm 3

Set $\mathbf{f}_0 = w(\mathbf{f}_0)$

▷ w indicates the wrap function.

if task is DNG **then**

 Sample $\mathbf{a}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

end if

for $n = 1, \dots, N$ **do**

if task is DNG **then**

$\text{out}_v^{(n-1)}, \text{out}_l^{(n-1)}, \text{out}_a^{(n-1)} = s_\theta((1 - (n-1) * dt), \mathbf{f}_{n-1}, \mathbf{v}_{n-1}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

else

$\text{out}_v^{(n-1)}, \text{out}_l^{(n-1)} = s_\theta((1 - (n-1) * dt), \mathbf{f}_{n-1}, \mathbf{v}_{n-1}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

end if

 ## Update step for $\mathbf{v}$ and $\mathbf{f}$

 ## Prediction step on velocities $\mathbf{v}$ and coordinates $\mathbf{f}$

 Compute $r = \alpha_v(n)/\alpha_v(n-1)$

 Compute $c = (r\sigma_v(n-1) - \sigma_v(n))\sigma_v(n-1)$

$\mathbf{v}_n^{\text{pred}} = r\mathbf{v}_{n-1} + c\text{out}_v^{(n-1)}$

$\mathbf{f}_n^{\text{pred}} = w(\mathbf{f}_{n-1} + \mathbf{v}_{n-1}^{\text{pred}} dt)$

 ## Correction step on velocities $\mathbf{v}$ and coordinates $\mathbf{f}$

if task is DNG **then**

$\text{out}_v, \text{out}_l, \text{out}_a = s_\theta((1 - n * dt), \mathbf{f}_n^{\text{pred}}, \mathbf{v}_n^{\text{pred}}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

else

$\text{out}_v, \text{out}_l = s_\theta((1 - n * dt), \mathbf{f}_n^{\text{pred}}, \mathbf{v}_n^{\text{pred}}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

end if

$\text{out}_v = (1 - \exp(-(1 - n * dt)))/(1 + \exp(-(1 - n * dt))) \cdot \text{out}_v - \mathbf{v}_t/\sigma_{v_t}^2$

▷ Follow Eq. (19)

 Compute $\delta = \tau \frac{\dim(\text{out}_v)}{\|\text{out}_v\|_2^2}$

 Sample $\epsilon_v \sim \mathcal{N}_v(\mathbf{0}, \mathbf{I})$

▷ $\mathcal{N}_v$ is a normal distribution such that $\sum_i \mathbf{v}_i = \mathbf{0}$.

$\mathbf{v}_n = \mathbf{v}_n^{\text{pred}} + \delta \text{out}_v + \sqrt{2\delta} \epsilon_v$

▷ Update on $\mathbf{v}$

$\mathbf{f}_n = w(\mathbf{f}_n^{\text{pred}} - \mathbf{v}_n dt)$

▷ Update on $\mathbf{f}$

 ## Update step for $\mathbf{l}$

 Sample $\epsilon_l \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{l}_n = \mathbf{l}_{n-1} - (f(t) - g^2(t)s(\text{out}_l))dt + \sqrt{dt} \epsilon_l$

▷ EM step for $\mathbf{l}$

if task is DNG **then**

 ## Update step for $\mathbf{a}$

 Sample $\epsilon_a \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{a}_n = \mathbf{a}_{n-1} - (f(t) - g^2(t)s(\text{out}_a))dt + \sqrt{dt} \epsilon_a$

▷ EM step for $\mathbf{a}$

end if

end for

if task is DNG **then**

return A crystalline material sample $(\mathbf{f}_N, \mathbf{l}_N, \mathbf{a}_N)$

else

return A crystalline material sample $(\mathbf{f}_N, \mathbf{l}_N)$

end if

I. Experimental details

I.1. Hardware

All experiments presented in this paper can be performed on a single GPU. We relied on a GPU cluster with a mix of RTX 3090 and RTX A5000, with 24GB of memory.

I.2. Architecture

As mentioned in Section 3.2, our score network is parameterized using an architecture whose backbone is similar to that of previous work (Jiao et al., 2023, DIFFCSP) which enforces the periodic translation invariant by featurizing pairwise fractional coordinate differences with periodic functions of different frequencies. There are two main differences compared to the architecture of DIFFCSP: as we are coupling the fractional coordinates with an additional auxiliary variable that represents the velocity, we need the network to also take this velocity variable $\mathbf{v}_t$ as additional inputs. The network is then outputting the score for all the diffusion processes involved in the model: the score related to the velocities, the one for the lattice parameters, and the one for the atom types. The additional modification we do is to consider a two-layer network instead of a single layer to predict the score related to the velocities. Regarding the network parameters, we considered 4 message-passing layers for PEROV-5, while we increased them to 6 for the remaining three datasets. In all the experiments, we considered the hidden dimension to be 512, the time embedding to be a 256-dimensional vector and we used SiLU activation with layer norm. While the presentation in the paper is done in terms of continuous time diffusion models, the implementation is done in discrete time to guarantee an apples-to-apples comparison with baselines such as DIFFCSP and EQUICSP.

DIFFCSP employs a graph-neural network as a score network that adapts EGNN from Satorras et al. (2021) to fractional coordinates. In the following, we are going to present all the components that form this architecture.

Lattice parameters pre-processing We follow the same pre-processing steps for the lattice parameters (both lengths and angles) used by Lin et al. (2024, EQUICSP). Lengths are usually defined from $[0, +\infty)$ while angles are defined in the $(0, \pi)$ interval. However, the diffusion process defined in Eq. (7) operates in the $(-\infty, +\infty)$ domain, and therefore can result in unreasonable lattice parameters. Therefore, we use a logarithmic transformation for the lengths, mapping them from $(0, +\infty)$ to $(-\infty, +\infty)$. For angles, we map them using the following operation $\tan(\phi - \pi/2)$ from $(0, \pi)$ to $(-\infty, +\infty)$.

Components of EGNN Let consider $\mathbf{f}_t, \mathbf{v}_t, \mathbf{l}_t, \mathbf{a}$ being the input of our network. The input features are computed by

$$\mathbf{h}_i^{(0)} = \text{NN}(f_{\text{atom}}, f_{\text{pos}(t)}),$$

where $f_{\text{atom}}, f_{\text{pos}}$ are the atomic embedding and sinusoidal positional embedding and NN is an MLP.

Then the input features are processed by a series of s message-passing layers that compute

$$\begin{aligned} \mathbf{m}_{ij}^{(s)} &= \varphi_m(\mathbf{h}_i^{(s-1)}, \mathbf{h}_j^{(s-1)}, \mathbf{v}, \mathbf{l}, \text{SinusoidalEmbedding}(\mathbf{f}_j - \mathbf{f}_i)) \\ \mathbf{m}_i^{(s)} &= \sum_{j=1}^N \mathbf{m}_{ij}^{(s)} \\ \mathbf{h}_i^{(s)} &= \mathbf{h}_i^{(s-1)} + \varphi_h(\mathbf{h}_i^{(s-1)}, \mathbf{m}_i^{(s)}) \end{aligned}$$

where $\mathbf{m}_{ij}^{(s)}$ and $\mathbf{h}_j^{(s-1)}$ represent the messages at layer s between nodes i and j . φ_m and φ_h are two MLPs. Compared to the DIFFCSP implementation, we want to highlight that we are also passing the velocity $\mathbf{v}$ as input.

The SinusoidalEmbedding is a sinusoidal embedding layer defined as

$$\text{SinusoidalEmbedding}(x) := (\sin(2\pi kx), \cos(2\pi kx))_{k=0, \dots, n_{\text{freq}}}^T,$$

with being a n_{freq} being an hyper-parameter.

After S steps of message passing, we compute all the different scores by doing the following:

$$\begin{aligned} \mathbf{s}_v^{(i)} &= \varphi_v(\mathbf{h}_i^{(S)}) \\ \mathbf{s}_l &= \varphi_l\left(\frac{1}{N} \sum_{i=1}^N \mathbf{h}_i^{(S)}\right) \\ \mathbf{s}_a^{(i)} &= \varphi_a(\mathbf{h}_i^{(S)}) \end{aligned}$$

where we want to stress that φ_v is a 2-layer neural network while φ_l and φ_a are single-layer MLPs.

For training, we used the same loss weights that were used by DIFFSCP. We consider $\lambda_v = 1$ and $\lambda_l = 1$ for the CSP task. For the DNG task, instead, as we consider three different ways for modelling the discrete atom type features, we used different weights depending on the modelling choice. If we use one-hot encoding for the atom types, we still rely on the DIFFSCP weights given by $\lambda_v = 1$, $\lambda_l = 1$, and $\lambda_a = 20$. In the case of analog-bits, we used $\lambda_a = 1$, while when using discrete diffusion for the atom types, we scale the losses using $\lambda_a = 0.33$. The scaling is needed to make the different loss terms have similar magnitudes.

I.3. Parameters for the KLDM forward process

We kept the drift coefficient $\gamma(t)$ constant at 1 in all the experiments presented in the paper following [Zhu et al. \(2024\)](#). In their experiments, they also tuned the time horizon T of the process in Eq. (12) depending on the considered task. In our experiments for material generation, we kept the time horizon constant at $T = 2$. In terms of the implementation, as we implemented everything in discrete time, we discretize the time in the interval $[0, 2)$ for the diffusion process. For lattice parameters and atom types, we rely on the standard Euclidean diffusion model where the drift and diffusion coefficients are defined by a linear schedule on the interval $[0, 1)$. We trained all the networks using AdamW with the default PyTorch parameters, without gradient clipping and by performing early stopping based on metrics computed on a subset of the validation set: match-rate for the CSP task and valid structures for the DNG task.

Table 6: Dataset hyperparameters

INFO	PEROV-5	CARBON-24	MP-20	MPTS-52
Max Atoms	5	24	20	52
Total Number of Samples	18928	10153	45231	40476
Batch Size	1024	256	256	256

I.4. Major difference between KLDM and competitors

In this section, we compare the key differences between our proposed KLDM and the baseline methods discussed in Section 5, namely DIFFCSP, EQUICSP, and FLOWMM. Among these, we think that DIFFCSP is the closest to our approach. The main differences are that we model fractional coordinates differently and we use a linear noise schedule, as opposed to their cosine noise schedule. In addition to that, DIFFCSP employs a matrix representation for the lattice parameters, while we treat them as a vector of six scalars.

EQUICSP builds upon DIFFCSP with two main modifications: it introduces additional losses to ensure the lattice permutation invariance of the learned distributions, and it defines a different noising mechanism called *Periodic CoM-free Noising* scheme. This scheme ensures that the sampled noise does not induce a translation of the center of mass of $\mathbf{f}_0$, thereby preserving the periodic translation invariance in the target score. In contrast, we define the forward process on the coupling given by fractional coordinates and associated velocity variables, and we do not account for lattice permutation invariance, leaving that as a direction for future work.

FLOWMM generalizes Riemannian Flow matching for material generation, which, although closely related to diffusion, is a different modeling approach. In addition to that, it considers an informed prior distribution over the lattice parameters, and in the DNG task, it represents atom types using analog bits ([Chen et al., 2023](#)), in contrast to DIFFCSP and EQUICSP, which use a one-hot encoding. We present results by considering the same informed prior and also by using three different representations for the discrete atom type features. Additionally, for DNG, FLOWMM provides extra inputs to the score

network that neither KLDM nor the other baselines use. The additional input represents the cosine of the angles between the Cartesian edges between atoms and three lattice vectors