KINETIC LANGEVIN DIFFUSION FOR CRYSTALLINE MATERIALS GENERATION

François Cornet[1,*] **Federico Bergamin**[1,*] **Arghya Bhowmik**[1]
Juan Maria García-Lastra[1] **Jes Frellsen**[1] **Mikkel N. Schmidt**[1]
[1] Technical University of Denmark

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 Models (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 consistent with 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 (DMs) 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*, which 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 a multi-modal data type, specifically a geometric graph (Joshi et al., 2023) that combines discrete and continuous features. 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

[*]These authors contributed equally to this work. Correspondence to {frjc,fedbe}@dtu.dk.

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., 2024a). In addition to this, the data distribution is governed by inherent symmetries, including permutation invariance (swapping atom indices or lattice bases), translation invariance (shifting atom coordinates), and rotation invariance (rotating atom positions and the unit cell together).

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 network. 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 an inconsistent training objective (Lin et al., 2024). While the issue has been acknowledged in the literature, the solution to this problem proposed in this paper is novel and elegant.

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 hypertorus directly as in previous work (Jiao et al., 2024a), we generalize Trivialized Diffusion Models (TDM) (Zhu et al., 2024) to estimate 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 $\boldsymbol{x}$,

$$\boldsymbol{x} = (\boldsymbol{f}, \boldsymbol{l}, \boldsymbol{a}), \tag{1}$$

where $\boldsymbol{f} = (\boldsymbol{f}_1, \dots, \boldsymbol{f}_K) \in [0,1)^{3\times K}$ denotes the fractional coordinates, $\boldsymbol{l} = (\boldsymbol{l}_1, \boldsymbol{l}_2, \boldsymbol{l}_3) \in \mathbb{R}^{3\times 3}$ refers to the lattice vectors of the unit cell, and $\boldsymbol{a} = (a_1, \dots, a_K) \in \mathbb{Z}^K$ encodes the chemical composition.

The infinite periodic structure can be represented as,

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

where $\mathbf{1}$ is a vector of ones with size K, and $\boldsymbol{k} = (k_1, k_2, k_3)$ translate the unit cell to tile the entire space. The lattice vectors $\boldsymbol{l}$ 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 $p_{\text{data}}(\boldsymbol{x})$ denote the true data distribution over the unit cells, and let $q(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\boldsymbol{x}_i}$ be the empirical data distribution defined by the N available samples. The goal of generative models is to learn an approximate model $p_\theta(\boldsymbol{x})$ of $p_{\text{data}}(\boldsymbol{x})$ using $q(\boldsymbol{x})$. In the realm of crystalline materials generation, there are two tasks of interest.

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

De-novo Generation (DNG) aims at discovering novel and stable materials. The goal is to generate samples from $p_{\text{data}}(\boldsymbol{f}, \boldsymbol{l}, \boldsymbol{a})$. We approximate the distribution by training a generative model $p_\theta(\boldsymbol{f}, \boldsymbol{l}, \boldsymbol{a})$ on samples $\{(\boldsymbol{f}, \boldsymbol{l}, \boldsymbol{a})\}_{i=1}^N$ 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 $g \in G$, $p(g \cdot \boldsymbol{x}) = p(\boldsymbol{x})$, with $\cdot$ denoting the group action. A conditional distribution is G-equivariant if for any $g \in G$, $p(g \cdot \boldsymbol{x} | g \cdot \boldsymbol{y}) = p(\boldsymbol{x}|\boldsymbol{y})$.

A number of transformations leave a material $\boldsymbol{x}$ unchanged, meaning the *true* data distribution has inherent symmetries that we want our model to inherit:

Permutation of atom indices

$$p(\boldsymbol{f}, \boldsymbol{l}, \boldsymbol{a}) = p(g \cdot \boldsymbol{f}, \boldsymbol{l}, g \cdot \boldsymbol{a}), \qquad \forall g \in S_K; \tag{3}$$

Periodic translation of fractional coordinates

$$p\big(\boldsymbol{f}, \boldsymbol{l}, \boldsymbol{a}\big) = p\big(g \cdot \boldsymbol{f}, \boldsymbol{l}, \boldsymbol{a}\big) \qquad \forall g \in \mathbb{T}^3 \cong \mathbb{R}^3/\mathbb{Z}^3; \tag{4}$$

Rotation of lattice vectors

$$p\big(\boldsymbol{f}, \boldsymbol{l}, \boldsymbol{a}\big) = p\big(\boldsymbol{f}, g \cdot \boldsymbol{l}, \boldsymbol{a}\big), \qquad \forall g \in \text{SO}(3); \tag{5}$$

Permutation of the lattice basis

$$p(\boldsymbol{f}, \boldsymbol{l}, \boldsymbol{a}) = p(g \cdot \boldsymbol{f}, g \cdot \boldsymbol{l}, \boldsymbol{a}), \qquad \forall g \in S_3. \tag{6}$$

Eq. (3) is naturally addressed by using graph neural networks, combined with a factorized prior distribution with no dependency on the index. Combining fractional coordinates and rotation invariant lattice representations addresses Eq. (5). Additional loss terms (Lin et al., 2024) or specific neural network architectures can address Eq. (6). Remaining is to handle Eq. (4).

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 $(\boldsymbol{x}_t)_{t\in[0,T]}$. Given an initial condition, $\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}) = p_{\text{data}}(\boldsymbol{x})$, the conditional distribution of $(\boldsymbol{x}_t)_{t\in[0,T]}$ can be described by a Stochastic Differential Equation (SDE),

$$\mathrm{d}\boldsymbol{x}_t = f(t)\boldsymbol{x}_t \mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}_t, \tag{7}$$

where $\boldsymbol{x}_t \in \mathbb{R}^d$ denotes the latent variable at time t, $f(t)$ and $g(t)$ are scalar function of t, and $\boldsymbol{w}_t$ 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 $p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\alpha_t\boldsymbol{x}_0, \sigma_t^2\mathbb{I})$ with $\alpha_t = \exp(-0.5\int_0^t \beta(s)\mathrm{d}s)$ and $\sigma_t^2 = 1 - \exp(\int_0^t \beta(s)\mathrm{d}s)$, and the process defined by Eq. (7) converges geometrically from a low-variance Gaussian distribution centered around the data to the standard Gaussian distribution $p_T(\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_t|\mathbf{0}, \mathbb{I})$, which can be therefore interpreted as an uninformative prior distribution.

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

$$\mathrm{d}\boldsymbol{x}_t = \left[f(t)\boldsymbol{x}_t - g^2(t)\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x})\right]\mathrm{d}t + g(t)\mathrm{d}\hat{\boldsymbol{w}}_t, \tag{8}$$

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

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{t\sim\mathcal{U}[0,1], \boldsymbol{x}_0\sim p_{\text{data}}(\boldsymbol{x}), \boldsymbol{x}_t\sim p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\lambda(t)\left\|s_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t}\log p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\|_2^2\right], \tag{9}$$

where $\lambda(t)$ is a time-dependent weighting function.

2.3 EXISTING DIFFUSION MODELS FOR FRACTIONAL COORDINATES

As previously mentioned, the fractional coordinates define a hypertorus $\boldsymbol{f} \in [0,1)^{3\times K} \cong \mathbb{T}^{3\times K}$. Most existing work has built upon DIFFCSP (Jiao et al., 2024a), that leverages the scored-based framework of Song et al. (2021) extended to Riemmanian manifolds (De Bortoli et al., 2022; Jing et al., 2022). In what follows, we present the key ingredients of DIFFCSP for modeling coordinates.

Transition kernel DIFFCSP implements a specific case of Eq. (7), with $f(t) = 0$ and $g(t) = \sqrt{\mathrm{d}\sigma^2(t)/\mathrm{d}t}$, 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 $\boldsymbol{f}_t$ given $\boldsymbol{f}_0$ using a normal distribution does not capture the bounded and cyclical nature of $p_{\text{data}}(\boldsymbol{f})$, the solution consists in first sampling from the normal distribution $\tilde{\boldsymbol{f}}_t \sim \mathcal{N}(\boldsymbol{f}_0, \sigma^2(t))$ and then wrapping the samples $\boldsymbol{f} = w(\tilde{\boldsymbol{f}}_t)$ – 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,

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

whose gradient can then be approximated by truncating the series. Due to the wrapping operation, Eq. (10) converges to a uniform distribution over the hypertorus as $t \to T$.

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

$$\mathcal{L}_{\boldsymbol{f}_t}(\theta) = \mathbb{E}_{\boldsymbol{f}_0\sim p_0, \boldsymbol{f}_t\sim p_{t|0}(\boldsymbol{f}_t|\boldsymbol{f}_0)}\Big[\lambda(t)\big|\big|\nabla_{\boldsymbol{f}_t}\log p_{t|0}(\boldsymbol{f}_t|\boldsymbol{f}_0) - s_\theta^{\boldsymbol{f}}(\boldsymbol{x}_t, t)\big|\big|_2^2\Big], \tag{11}$$

where $\lambda(t) = 1/\mathbb{E}_{\boldsymbol{f}_t\sim p_{t|0}(\boldsymbol{f}_t|\boldsymbol{f}_0)}\big[\|\nabla_{\boldsymbol{f}_t}\log p_{t|0}(\boldsymbol{f}_t|\boldsymbol{f}_0)\|_2^2\big]$ scales the loss magnitude to be constant in expectation for any time t (Jing et al., 2022; Jiao et al., 2024a).

3 KINETIC LANGEVIN DIFFUSION FOR MATERIALS GENERATION

We now introduce Kinetic Langevin Diffusion for Materials (KLDM), and particularize our exposition to the fractional coordinates $\boldsymbol{f} \in [0,1)^{3\times K} \cong \mathbb{T}^{3\times K}$, used to define the positions of atoms inside the unit cell. Given the isomorphism between the torus $\mathbb{T}$ and SO(2)[1], the fractional coordinates can also be represented as a collection of 2×2 rotation matrices. We denote this alternative representation $\hat{\boldsymbol{f}}$. Since $\hat{\boldsymbol{f}}$ is defined on a direct product of Lie groups[2], we propose to generalize Trivialized Diffusion Models (TDM) (Zhu et al., 2024) to operate on geometric graphs similar to those defined in Eq. (1).

While TDM (Zhu et al., 2024) in principle allows for a 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, as presented in Appendix E. In this section, we first present TDM and then detail the proposed modifications needed to reach faster convergence and better results.

3.1 TRIVIALIZED DIFFUSION MODELS (TDM) FOR FRACTIONAL COORDINATES

Building upon previous work on momentum-based optimization (Tao & Ohsawa, 2020) and sampling (Kong & Tao, 2023), TDM are specifically tailored to data defined on Lie groups, and exploit 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 a manifold with auxiliary variables representing velocities defined on the Lie algebra. More precisely, the fractional coordinates $\hat{\boldsymbol{f}}$, elements of the group G, are coupled with velocities $\hat{\boldsymbol{v}} \in \mathfrak{g}$ defined on the Lie algebra $\mathfrak{g}$. The latter corresponds to the tangent space T_eG of the identity element of the group $e \in G$, and crucially can be thought of as an Euclidean space, $\mathfrak{g} \cong \mathbb{R}^{3\times K}$. In the present setting, velocities $\hat{\boldsymbol{v}}$ are 2×2 skew-symmetric matrices.

[1] We provide an intuitive explanation of this correspondence in Appendix C.1.

[2] A 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 B.

For this reason, a standard diffusion process can be defined for $\hat{\boldsymbol{v}}$. Given its coupling with $\hat{\boldsymbol{f}}$ determined by left-trivialization (Zhu et al., 2024), the resulting forward process is defined as,

$$\begin{cases} \mathrm{d}\hat{\boldsymbol{f}}_t &= \hat{\boldsymbol{f}}_t \hat{\boldsymbol{v}}_t \mathrm{d}t, \\ \mathrm{d}\hat{\boldsymbol{v}}_t &= -\gamma \hat{\boldsymbol{v}}_t \mathrm{d}t + \sqrt{2\gamma}\mathrm{d}\boldsymbol{w}_t^{\mathfrak{g}}, \end{cases} \tag{12}$$

where the Ordinary Differential Equation (ODE) describes the time evolution of the fractional coordinates $\hat{\boldsymbol{f}}_t$ (that lie on a manifold) through a coupling with the velocity variables $\boldsymbol{v}_t$. The latter evolve according to an SDE similar to that of Eq. (7), with constant drift $f(t) = -\gamma$ and constant volatility $g(t) = \sqrt{2\gamma}$. We note that, in principle, f and g could be time-dependent functions. In summary, Eq. (12) corrupts $\hat{\boldsymbol{f}}_t$ living on a hypertorus via a standard Euclidean diffusion process on the auxiliary velocities, while guaranteeing that the trajectory $(\hat{\boldsymbol{f}}_t)_{t\in[0,T]}$ remains on the manifold.

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

$$\begin{cases} \mathrm{d}\hat{\boldsymbol{f}}_t &= \hat{\boldsymbol{f}}_t \hat{\boldsymbol{v}}_t \mathrm{d}t, \\ \mathrm{d}\hat{\boldsymbol{v}}_t &= \big[-\gamma \hat{\boldsymbol{v}}_t + 2\gamma \nabla_{\hat{\boldsymbol{v}}_t} \log p_t(\hat{\boldsymbol{f}}_t, \hat{\boldsymbol{v}}_t) \big] \mathrm{d}t + \sqrt{2\gamma}\mathrm{d}\hat{\boldsymbol{w}}_t^{\mathfrak{g}}, \end{cases} \tag{13}$$

where t flows backwards and $\nabla_{\hat{\boldsymbol{v}}_t} \log p_t(\hat{\boldsymbol{f}}_t, \hat{\boldsymbol{v}}_t)$ denotes the true score. The latter is unavailable but can be approximated with a neural network $s_\theta(\hat{\boldsymbol{f}}_t, \hat{\boldsymbol{v}}_t, t)$ trained using DSM as in Eq. (11). However, we stress that the gradient is now with respect to $\hat{\boldsymbol{v}}_t$, in contrast to Eq. (9) where it is with respect to $\boldsymbol{f}_t$. As Eq. (12) there is no direct noise in the forward dynamics of $\hat{\boldsymbol{f}}_t$, its time-reversal in Eq. (13) needs no score-based correction, and it corresponds to the reverse ODE.

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 $t + \mathrm{d}t$ involves solving an exponential map. Intuitively, this operation generalizes the concept of moving on a straight line to the manifold case. By considering an initial velocity $\hat{\boldsymbol{v}}_t$, an initial position $\hat{\boldsymbol{f}}_t$ and the step-size $\mathrm{d}t$, the update of the positions can be written as

$$\hat{\boldsymbol{f}}_{t+\mathrm{d}t} = \exp_{\hat{\boldsymbol{f}}_t}(\hat{\boldsymbol{f}}_t \hat{\boldsymbol{v}}_t) = \hat{\boldsymbol{f}}_t \operatorname{expm}(\hat{\boldsymbol{v}}_t \mathrm{d}t), \tag{14}$$

since the matrix exponential is the exact solution of the exponential map (Zhu et al., 2024).

3.2 Practicalities

The formalism introduced in the previous section considered $\hat{\boldsymbol{f}}$ as a collection of rotation matrices. However, the usual architectures used to parametrize score networks have been designed to process fractional coordinates directly (Jiao et al., 2024a; 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 $\boldsymbol{f}$. We simply need to express the matrix multiplication appearing in the deterministic part of the dynamics in Eqs. (12) and (13) as operations on the torus. Concretely, if we consider the exponential map defined in Eq. (14) and that $\hat{\boldsymbol{f}}$ is a single rotation matrix by an angle $\theta \in [-\pi, \pi)$, the matrix exponential corresponds to a periodic translation as follows,

$$\hat{\boldsymbol{f}} \operatorname{expm}(\hat{\boldsymbol{v}} dt) \to w(\theta + v dt), \tag{15}$$

where $\hat{\boldsymbol{v}}$ is a skew-symmetric matrix and v is its anti-diagonal element, while $w(\cdot) = \operatorname{atan2}\big(\sin(\cdot), \cos(\cdot)\big)$ is the wrapping function with atan2 denoting the signed atan function. We provide more details in Appendix C.2. From now on, all operations are presented in terms of $\boldsymbol{f}$.

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

$$p_{t|0}(\boldsymbol{f}_t, \boldsymbol{v}_t | \boldsymbol{f}_0, \boldsymbol{v}_0) = \mathrm{WN}_{\boldsymbol{r}}\big(\operatorname{logm}(\boldsymbol{f}_0^{-1}\boldsymbol{f}_t) | \boldsymbol{\mu}_{\boldsymbol{r}_t}, \boldsymbol{\sigma}^2_{\boldsymbol{r}_t}\big) \cdot \mathcal{N}_{\boldsymbol{v}}\big(\boldsymbol{v}_t | \boldsymbol{\mu}_{v_t}, \boldsymbol{\sigma}^2_{v_t}\big), \tag{16}$$

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

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

Denoising score-matching objective 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, $\boldsymbol{v}_t$, compared to the objective presented in Eq. (11) where the gradient was instead computed with respect to the coordinates $\boldsymbol{f}_t$. It writes,

$$\nabla_{\boldsymbol{v}_t} \log p_{t|0} = \nabla_{\boldsymbol{r}_t} \log \mathrm{WN}_{\boldsymbol{r}}\big(\boldsymbol{r}_t|\boldsymbol{\mu}_{\boldsymbol{r}_t}, \boldsymbol{\sigma}^2_{\boldsymbol{r}_t}\big) \frac{\partial \boldsymbol{\mu}_{\boldsymbol{r}_t}}{\partial \boldsymbol{v}_t} + \nabla_{\boldsymbol{v}_t} \log \mathcal{N}_{\boldsymbol{v}}\big(\boldsymbol{v}_t|\boldsymbol{\mu}_{v_t}, \boldsymbol{\sigma}^2_{v_t}\big), \tag{17}$$

where $p_{t|0}$ stands for $p_{t|0}(\boldsymbol{f}_t, \boldsymbol{v}_t|\boldsymbol{f}_0, \boldsymbol{v}_0)$, and where we renamed $\boldsymbol{r}_t = \mathrm{logm}(\boldsymbol{f}_0^{-1}\boldsymbol{f}_t)$.

3.3 Parametrization of the score function

By carefully inspecting the target expression, we note that the target writes as a sum of two terms,

$$\nabla_{\boldsymbol{v}_t} \mathrm{log}p_{t|0}(\boldsymbol{f}_t, \boldsymbol{v}_t|\boldsymbol{f}_0, \boldsymbol{v}_0) = \frac{1-e^{-t}}{1+e^{-t}} \nabla_{\boldsymbol{r}_t} \mathrm{WN}\big(\boldsymbol{r}_t|\boldsymbol{\mu}_f, \boldsymbol{\sigma}^2_f\big) - \frac{\varepsilon_{\boldsymbol{v}}}{\sigma_{\boldsymbol{v}_t}}, \tag{18}$$

where $\varepsilon_{\boldsymbol{v}}$ is the reparameterization noise sampled to obtain $\boldsymbol{v}_t$.

Initial velocities We consider zero initial velocities, *i.e.* $p(\boldsymbol{v}_0) = \delta(\boldsymbol{v}_0)$, which intuitively corresponds to considering the coordinates $\boldsymbol{f}$ to be at rest at time $t = 0$. This allows the noise to gradually propagate from the velocities to the coordinates over the course of the diffusion process. The benefit of gradually adding noise, starting from with velocities close to zero, was also observed by Dockhorn et al. (2022).

Simplified score parametrization While predicting the full score in Eq. (18) directly is possible, alternative parameterizations exist, in particular when considering zero initial velocities, *i.e.* $\boldsymbol{v}_0 = \mathbf{0}$. Using the relationship $\boldsymbol{v}_t = \boldsymbol{\mu}_{\boldsymbol{v}_t} + \boldsymbol{\sigma}_{\boldsymbol{v}_t}\varepsilon_{\boldsymbol{v}}$ and that $\boldsymbol{\mu}_{\boldsymbol{v}_t} = \mathbf{0}$, the second term in Eq. (18) can be computed in closed-form, $\varepsilon_{\boldsymbol{v}} = \boldsymbol{v}_t/\sigma_{\boldsymbol{v}_t}$, and does not need be learned. We are then left with one term only, and the simplified parametrization writes,

$$s^{\boldsymbol{v}}_\theta(\boldsymbol{x}_t, t) = \frac{1-e^{-t}}{1+e^{-t}} s^{\boldsymbol{f}}_\theta(\boldsymbol{x}_t, t) - \frac{\boldsymbol{v}_t}{\sigma^2_{\boldsymbol{v}_t}}, \tag{19}$$

where superscript $\boldsymbol{f}$ in $s^{\boldsymbol{f}}_\theta(\boldsymbol{x}_t, t)$ refers to the score contribution coming from the coordinates. More details are provided in Appendix D. 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 parametrize our score network, $s^{\boldsymbol{f}}_\theta(\boldsymbol{x}_t, t)$, using a graph neural network architecture. Its backbone is similar to that of previous work (Jiao et al., 2024a), *i.e.* periodic translation invariant by featurizing pairwise fractional coordinate differences with periodic functions of different frequencies. The sole difference is that the network now takes velocity variables $\boldsymbol{v}_t$ as additional inputs. Details about the architecture are provided in Appendix H.

3.4 Ensuring consistent training targets

Invariant approximate distribution As introduced in Eq. (4), the true data distribution $p_{\mathrm{data}}(\boldsymbol{x})$, with $\boldsymbol{x} = (\boldsymbol{f}, \boldsymbol{l}, \boldsymbol{a})$ as defined in Section 2.1, is periodic translation invariant. Intuitively, this means that unit cells equivalent up to periodic translation are equally likely. In DiffCSP (Jiao et al., 2024a), the learned distribution $p_\theta(\boldsymbol{x})$ is made invariant by combining an invariant prior (*i.e.* the uniform distribution on the hypertorus, in this case) with an approximate reverse process that is equivariant to periodic translation, *i.e.* $p_\theta(\boldsymbol{f}_{t-1}, |\boldsymbol{x}_t) = p_\theta(g \cdot \boldsymbol{f}_{t-1}, |g \cdot \boldsymbol{x}_t), \forall g \in \mathbb{R}^3/\mathbb{Z}^3$. This is achieved in practice by having a score parametrization that is invariant to periodic translations, *i.e.* $s^{\boldsymbol{f}}_\theta(\boldsymbol{x}_t, t) = s^{\boldsymbol{f}}_\theta(g \cdot \boldsymbol{x}_t, t), \forall g \in \mathbb{R}^3/\mathbb{Z}^3$. If the learned score $s^{\boldsymbol{f}}_\theta(\boldsymbol{x}_t, t)$ correctly approximates the score of the true target distribution, then we are guaranteed that samples from $p_\theta(\boldsymbol{x}) \approx p_{\mathrm{data}}(\boldsymbol{x})$.

Score parametrization and targets Although the score parametrization guarantees the desired invariance of the learned distribution, we note that there exists a potential mismatch between training targets and the invariance property of the score network. As a concrete example, we assume two latents $\boldsymbol{f}_t$ and $g \cdot \boldsymbol{f}_t$ obtained by noising $\boldsymbol{f}_0$, *i.e.* $\boldsymbol{f}_t$ and $g \cdot \boldsymbol{f}_t$ are equivalent up to a periodic translation

$\boldsymbol{f}_t \sim g \cdot \boldsymbol{f}_t$. On the one hand, we have that $s^{\boldsymbol{f}}_\theta(\boldsymbol{x}_t, t) = s^{\boldsymbol{f}}_\theta(g \cdot \boldsymbol{x}_t, t)$ by construction, in other words, that the output of the score network is identical for both noisy samples. Whilst, on the other hand, the corresponding target scores, $\nabla_{\boldsymbol{f}_t} \log p_{t|0}(\boldsymbol{f}_t|\boldsymbol{f}_0)$ and $\nabla_{g\cdot\boldsymbol{f}_t} \log p_{t|0}(g \cdot \boldsymbol{f}_t|\boldsymbol{f}_0)$, can be different. We note that a similar observation has been previously pointed out in the literature (Lin et al., 2024) – we defer the discussion about Lin et al. (2024) to Section 4.

While the mismatch can be seen from the perspective of the score network, the invariant architecture is not the root of the problem, that instead originates from the employed corruption mechanism. The corresponding transition kernel, described in Eq. (10), is equivariant, as for each pair $(\boldsymbol{f}_0, \boldsymbol{f}_t)$, or any periodically translated version thereof $(g \cdot \boldsymbol{f}_0, g \cdot \boldsymbol{f}_t)$, we have that the target value is the same. However, the noising process mapping $\boldsymbol{f}_0$ to $\boldsymbol{f}_t$ does not ensure a *unique correspondence* between $\boldsymbol{f}_t$ and $\boldsymbol{f}_0$ – in other words, that among all possible $\boldsymbol{f}_0^{'} \sim \boldsymbol{f}_0$, the noised $\boldsymbol{f}_0$ always remains the *closest* to $\boldsymbol{f}_t$. The conditional score function used as training target in Eq. (9) always points towards $\boldsymbol{f}_0$, even if an $\boldsymbol{f}_0^{'} \sim \boldsymbol{f}_0$ lies *closer* to $\boldsymbol{f}_t$. With this in mind, we present now how this is solved in KLDM.

Velocity fields with zero net translation To enforce that for a given initial condition $\boldsymbol{f}_0$, all $(\boldsymbol{f}_t)_{t\in[0,T]}$ on trajectories generated by the forward process in Eq. (12) shares the same center of gravity as the one of $\boldsymbol{f}_0$, we prevent velocities $(\boldsymbol{v}_t)_{t\in[0,T]}$ from inducing a net overall translation at every time step. This constraint ensures that all the $(\boldsymbol{f}_t)_{t\in[0,T]}$ along the trajectory share the same group element $g \in \mathbb{R}^3/\mathbb{Z}^3$ as $\boldsymbol{f}_0$ by avoiding that $\boldsymbol{f}_t$ "jumps" to a different group element g at any t during the corruption process. As velocities, $\boldsymbol{v} \in \mathbb{R}^{3\times K}$, are simple Euclidean variables, we leverage a trick similar to that used for molecules, and only consider velocity fields living in the linear subspace where the center of gravity is always zero (Xu et al., 2022; Hoogeboom et al., 2022). In the transition kernel in Eq. (17), $\mathcal{N}_{\boldsymbol{v}}$ consequently corresponds to a normal distribution where all samples are such that $\sum_i \boldsymbol{v}_i = \boldsymbol{0}$. The corresponding velocity score output $s^{\boldsymbol{v}}_\theta(\boldsymbol{x}_t, t)$ in Eq. (19) is therefore also constructed to be mean-free, *i.e.* as to not induce a net overall translation.

With this in place, we can now check that, given velocities $\boldsymbol{v}_t$, the target score in Eq. (17) is identical for $\boldsymbol{f}_t$ and $g \cdot \boldsymbol{f}_t, \forall g \in \mathbb{R}^3/\mathbb{Z}^3$, ensuring consistent training targets for the invariant score network. We need only consider the first term in Eq. (17), as the second term only depends on velocities. By inspecting the first term, we note that $\boldsymbol{\mu}_{\boldsymbol{r}_t}$ and $\boldsymbol{\sigma}^2_{\boldsymbol{r}_t}$ also only depends on the velocities. Therefore we need only to check that $\boldsymbol{r}_t = \mathrm{logm}(\boldsymbol{f}_0^{-1}\boldsymbol{f}_t)$ is the same if we consider $g \cdot \boldsymbol{f}_t$ instead of $\boldsymbol{f}_t$. Using the fact that $\boldsymbol{f}_0$ and $\boldsymbol{f}_t$ are always represented with the same group element by construction of the forward process in Eq. (12) due to the zero net translation, we observe that $\boldsymbol{r}_t$ is identical for any pair $(\boldsymbol{f}_0, \boldsymbol{f}_t)$ and $(g \cdot \boldsymbol{f}_0, g \cdot \boldsymbol{f}_t)$, thereby ensuring consistency between training targets and the invariant score network.

3.5 Lattice vectors and atom types

Following previous work (Jiao et al., 2024a; Lin et al., 2024), we rely on standard Euclidean diffusion as defined in Eqs. (7) and (8) for the lattice parameters $\boldsymbol{l}$, where the drift and diffusion functions are defined by a linear schedule. We represent $\boldsymbol{l}$ as a 6-dimensional vector, collecting side lengths and angles. For the atomic compositions $\boldsymbol{a}$, we consider three representations used in previous work: one-hot encoding, as used in DiffCSP (Jiao et al., 2024a); analog-bits (Chen et al., 2023), following FlowMM (Miller et al., 2024); and discrete diffusion, as done in MatterGen (Zeni et al., 2025).

4 Related Work

Deep generative models for crystals 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 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., 2024a; 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 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 compositions, sometimes outperforming domain-specific methods. Finally, a recent line of work (Jiao et al., 2024b; 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 disordered materials (Petersen et al., 2025).

Addressing training inconsistency To ensure a consistent training procedure, *i.e.* solving the mismatch between the score parametrization and the target we highlighted in Section 3.4, Lin et al. (2024) proposed a so-called *Periodic CoM-free Noising* scheme, where the sampled noise is carefully constructed not to induce a translation of the CoM of $\boldsymbol{f}_0$. While this ensures the desired correspondence between $\boldsymbol{f}_0$ and $\boldsymbol{f}_t$, the drawback is that the resulting kernel is no longer a Wrapped Normal distribution as Eq. (10), and therefore Lin et al. (2024) propose to estimate the transition kernel numerically, via a Von-Mises distribution where the concentration parameter is estimated via Monte Carlo.

5 Experimental results

5.1 Settings

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

Datasets We follow previous work (Jiao et al., 2024a) 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; CARBON-24 (Pickard, 2020) including materials made of carbon only and containing $6-24$ atoms in the unit cell; 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., 2024a) 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

CSP 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 reconstruction from $q_\theta(\boldsymbol{f}, \boldsymbol{l}|\boldsymbol{a})$ 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., 2024a), EquiCSP (Lin et al., 2024), and FlowMM (Miller et al., 2024).

CSP 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 KLDM. On the simpler PEROV-5 and CARBON-24, KLDM performs on par with the compared models for the @1 experiment, and yields improved results for @20. We note that on these datasets the PC sampler did not improve the quality of the samples. On the realistic MP-20 and MPTS-52, KLDM already yields a competitive MR for the EM sampler, while the PC sampler improves the MR and lower the RMSE even further.

Table 1: Crystal Structure Prediction (CSP) task 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. For the CARBON-24 dataset, the CSP@1 task is ill-defined due to its one-to-many nature, and we only report the obtained values for completeness. Most notably, KLDM compares favorably to the competing models, achieving performance comparable to or better than state-of-the-art methods.

	PEROV-5		CARBON-24		MP-20		MPTS-52	
MODEL	MR [%] ↑	RMSE ↓	MR [%] ↑	RMSE ↓	MR [%] ↑	RMSE ↓	MR [%] ↑	RMSE ↓
	METRICS @ 1							
CDVAE	45.31	0.1138	17.09	0.2969	33.90	0.1045	5.34	0.2106
DIFFCSP (PC)	52.02	0.0760	17.54	**0.2759**	51.49	0.0631	12.19	0.1786
EQUICSP (PC)	52.02	0.0707	–	–	57.59	0.0510	14.85	**0.1169**
FLOWMM	**53.15**	0.0992	**23.47**	0.4122	61.39	0.0566	17.54	0.1726
KLDM (EM)	53.14 $\pm.6$	0.0758 $\pm.002$	18.04 $\pm.8$	0.3188 $\pm.008$	61.72 $\pm.2$	0.0686 $\pm.001$	17.71 $\pm.3$	0.2023 $\pm.005$
KLDM (PC)	52.72 $\pm.8$	**0.0678** $\pm.002$	17.26 $\pm.7$	0.2827 $\pm.006$	**65.37** $\pm.1$	**0.0455** $\pm.001$	**21.46** $\pm.2$	0.1339 $\pm.002$
	METRICS @ 20							
CDVAE	88.51	0.0464	88.37	0.2286	66.95	0.1026	20.79	0.2085
DIFFCSP (PC)	98.60	**0.0128**	88.47	0.2192	77.93	0.0492	34.02	0.1749
FLOWMM	98.60	0.0328	84.15	0.3301	75.81	0.0479	34.05	0.1813
KLDM (EM)	**99.97**	0.0152	**90.19**	0.2154	**83.68**	0.0532	39.04	0.1865
KLDM (PC)	99.94	0.0226	85.86	**0.1988**	81.08	**0.0440**	**39.81**	**0.1462**

5.3 DE-NOVO GENERATION (DNG) TASK

DNG metrics We evaluate samples using a machine-learning interatomic potential, based on the open-source pipeline from MATTERGEN (Zeni et al., 2025). Sample quality is measured in terms of: 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 (meta-)stability; stability, as measured by the proportion of samples with an energy above the hull below 0.1eV/Å; and S.U.N (stable, unique, novel) that measure the percentage of promising generated samples. We also provide additional results with the usual proxy metrics in Appendix F.

DNG baselines We compare KLDM to DIFFCSP and MATTERGEN-MP on MP-20. For the baselines, we report numbers borrowed from MATTERGEN's own benchmark. We note that the compared models were trained on a re-optimized version of MP-20 where some chemical elements have been removed, specifically noble gases, radioactive elements and elements with atomic number greater than 84. Samples with energy above the hull bigger than 0.1 eV / atom were also filtered out. Our model was trained on the original MP-20.

DNG results As described in Section 3.5, the CSP model formulation can be readily extended to the DNG task, by having an additional diffusion process operating on the atom types, $\boldsymbol{a}$. To sample from $p_\theta(\boldsymbol{x})$, we first the number of atoms contained in the unit cell K from the empirical distribution over the training set, *i.e.* $p_\theta(\boldsymbol{x}|K)p(K)$. For completeness, we compare three ways of performing diffusion on the discrete atom types: continuous diffusion on one-hot encoded atom types (**C**), continuous diffusion on analog bits (**C-AB**), and discrete diffusion with absorbing state (**D**). The results are presented in Table 2. Notably, when relying on analog-bits or discrete diffusion to model the atom types, KLDM performs better than DIFFCSP in terms of RMSD, energy above the hull and stability, while being slightly subpar on S.U.N.. Compared to MATTERGEN-MP, we believe that the remaining gap can be explained by different elements: (1) a more expressive denoiser architecture operating in real space, (2) a PC sampler on the lattice parameters, and (3) the effect of the pre-processing of MP-20.

5.4 ABLATION STUDY

In this section, we analyze the impact of the different design choices: the simplified parametrization of the score formulated in Eq. (19) compared to the direct parametrization of Eq. (18), the effect of different distributions on the initial velocities, and the effect of enforcing a zero-net translation velocity field. Regarding the score parametrization, Table 3 and Fig. 1 *(Right)* show that the simplified parametrization leads to a better final performance across sampling schemes and datasets. Using

Table 2: Stability metrics on MP-20 as evaluated per MATTERGEN's pipeline (Zeni et al., 2025) (using `MatterSim-v1-1M` (Yang et al., 2024a)). The numbers reported for baselines are extracted from MATTERGEN's own benchmark. We note that KLDM was trained on the original MP-20 dataset, whereas MATTERGEN-MP* and DIFFCSP* were trained on a modified version thereof – i.e. structures were re-optimized, some chemical elements removed and samples with energy above the hull greater than 0.1 eV/atom removed. For completeness, we compare 3 ways of performing diffusion on the discrete atom types: continuous diffusion on one-hot encoded atom types (C), continuous diffusion on analog bits (C-AB), and discrete diffusion with absorbing state (D). For each variant of KLDM, we generated 10000 samples, from which we discarded samples that contain elements not supported by the validation pipeline. We report average and standard deviation across 3 seeds at sampling time. Notably, when relying on analog-bits or discrete diffusion to model the atom types, KLDM performs better than DiffCSP in terms of RMSD, energy above the hull and stability, while being slightly subpar on S.U.N..

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

zero-initial velocities significantly improves both performance and convergence speed as seen in Fig. 1 *(Center)*, which can potentially also be a consequence of the simplified parameterization. With these two design choices in place, enforcing a zero-net translation velocity field provides further improvements in terms of the validation set match rate, though to a lesser extent, as reported in Fig. 1 *(Left)*. We provide a more detailed discussion in Appendix E.

6 CONCLUSION

In this work, we presented KLDM, a novel diffusion model for periodic crystal structure generation. The key innovation of the model resided in the modeling of the fractional coordinates. By coupling coordinates with auxiliary Euclidean variables representing velocities, the diffusion process was offset to a flat space, allowing us to effectively perform diffusion on the hypertorus. It additionally provided a training objective consistent with the periodic translation symmetry of the true data distribution. We presented a simplified parameterization that allowed faster convergence and improved performance.

Empirically, we demonstrated the competitive performance of KLDM compared to current state-of-the-art baselines. On the CSP task, KLDM showed improved performance on the two larger datasets, MP-20 and MPTS-52. On the DNG task, KLDM improves upon DIFFCSP (Jiao et al., 2024a) on most metrics on the MP-20 dataset while being slightly subpar compared to MATTERGEN (Zeni et al., 2025).

Further validation with proper DFT simulations is a natural next step, to measure that the empirical performance improvement measured by proxy metrics and machine learning interatomic potential also translates to a practical advantage. We expect further improvements by optimizing the architecture and by investigating the combination of modality-specific noise schedules. Another interesting direction involves, instead, considering an informed prior for the lattice parameters, as it has been shown to provide a useful inductive bias (Miller et al., 2024). We also envision that including space-group information (Jiao et al., 2024b) or Wyckoff positions (Levy et al., 2024) with our approach could yield further performance improvements.

REFERENCES

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Dylan M Anstine and Olexandr Isayev. Generative models as an emerging paradigm in the chemical sciences. *Journal of the American Chemical Society*, 145(16):8736–8750, 2023.

Luis M Antunes, Keith T Butler, and Ricardo Grau-Crespo. Crystal structure generation with autoregressive large language modeling. *Nature Communications*, 15(1):1–16, 2024.

Andreas Arvanitogeōrgos. *An introduction to Lie groups and the geometry of homogeneous spaces*, volume 22. American Mathematical Soc., 2003.

Camille Bilodeau, Wengong Jin, Tommi Jaakkola, Regina Barzilay, and Klavs F Jensen. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 12(5):e1608, 2022.

Ivano E Castelli, David D Landis, Kristian S Thygesen, Søren Dahl, Ib Chorkendorff, Thomas F Jaramillo, and Karsten W Jacobsen. New cubic perovskites for one-and two-photon water splitting using the computational materials repository. *Energy & Environmental Science*, 5(10):9034–9043, 2012.

Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *The Twelfth International Conference on Learning Representations*, 2024.

Ting Chen, Ruixiang ZHANG, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations*, 2023.

François RJ Cornet, Grigory Bartosh, Mikkel N Schmidt, and Christian A Naesseth. Equivariant neural diffusion for molecule generation. *Advances in Neural Information Processing Systems*, 37, December 2024.

Callum J Court, Batuhan Yildirim, Apoorv Jain, and Jacqueline M Cole. 3-d inorganic crystal structure generation and property prediction via representation learning. *Journal of Chemical Information and Modeling*, 60(10):4518–4535, 2020.

Daniel W Davies, Keith T Butler, Adam J Jackson, Jonathan M Skelton, Kazuki Morita, and Aron Walsh. Smact: Semiconducting materials by analogy and chemical theory. *Journal of Open Source Software*, 4(38):1361, 2019.

Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. *Advances in Neural Information Processing Systems*, 35:2406–2422, 2022.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

Daniel Flam-Shepherd and Alán Aspuru-Guzik. 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.

Colin W Glass, Artem R Oganov, and Nikolaus Hansen. Uspex—evolutionary crystal structure prediction. *Computer physics communications*, 175(11-12):713–720, 2006.

Nate Gruver, Anuroop Sriram, Andrea Madotto, Andrew Gordon Wilson, C. Lawrence Zitnick, and Zachary Ward Ulissi. Fine-tuned language models generate stable inorganic materials as text. In *The Twelfth International Conference on Learning Representations*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jordan Hoffmann, Louis Maestrati, Yoshihide Sawada, Jian Tang, Jean Michel Sellier, and Yoshua Bengio. Data-driven approach to encoding and decoding 3-d crystal structures. *arXiv preprint arXiv:1909.00949*, 2019.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022.

Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1), 2013.

Rui Jiao, Wenbing Huang, Peijia Lin, Jiaqi Han, Pin Chen, Yutong Lu, and Yang Liu. Crystal structure prediction by joint equivariant diffusion. *Advances in Neural Information Processing Systems*, 36, 2024a.

Rui Jiao, Wenbing Huang, Yu Liu, Deli Zhao, and Yang Liu. Space group constrained crystal generation. In *The Twelfth International Conference on Learning Representations*, 2024b.

Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi S. Jaakkola. Torsional diffusion for molecular conformer generation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Chaitanya K Joshi, Cristian Bodnar, Simon V Mathis, Taco Cohen, and Pietro Lio. On the expressive power of geometric graph neural networks. In *International conference on machine learning*, pp. 15330–15355. PMLR, 2023.

Sungwon Kim, Juhwan Noh, Geun Ho Gu, Alan Aspuru-Guzik, and Yousung Jung. Generative adversarial networks for crystal structure prediction. *ACS central science*, 6(8):1412–1420, 2020.

Walter Kohn and Lu Jeu Sham. Self-consistent equations including exchange and correlation effects. *Physical review*, 140(4A):A1133, 1965.

Lingkai Kong and Molei Tao. Convergence of kinetic langevin monte carlo on lie groups. In Shipra Agrawal and Aaron Roth (eds.), *The Thirty Seventh Annual Conference on Learning Theory (COLT)*, volume 247, pp. 3011–3063. PMLR, 2023.

Daniel Levy, Siba Smarak Panigrahi, Sékou-Oumar Kaba, Qiang Zhu, Mikhail Galkin, Santiago Miret, and Siamak Ravanbakhsh. SymmCD: Symmetry-preserving crystal generation with diffusion models. In *AI for Accelerated Materials Design - NeurIPS 2024*, 2024.

Peijia Lin, Pin Chen, Rui Jiao, Qing Mo, Cen Jianhuan, Wenbing Huang, Yang Liu, Dan Huang, and Yutong Lu. Equivariant diffusion for crystal structure prediction. In *Forty-first International Conference on Machine Learning*, 2024.

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

Phillip Lippe, Bastiaan S. Veeling, Paris Perdikaris, Richard E Turner, and Johannes Brandstetter. PDE-refiner: Achieving accurate long rollouts with neural PDE solvers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Aaron Lou, Minkai Xu, Adam Farris, and Stefano Ermon. Scaling riemannian diffusion models. *Advances in Neural Information Processing Systems*, 36:80291–80305, 2023.

Peder Lyngby and Kristian Sommer Thygesen. Data-driven discovery of 2d materials by deep generative models. *npj Computational Materials*, 8(1):232, 2022.

Amil Merchant, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023.

Benjamin Kurt Miller, Ricky TQ Chen, Anuroop Sriram, and Brandon M Wood. Flowmm: Generating materials with riemannian flow matching. In *Forty-first International Conference on Machine Learning*, 2024.

Asma Nouira, Nataliya Sokolovska, and Jean-Claude Crivello. Crystalgan: learning to discover crystallographic structures with generative adversarial networks. *arXiv preprint arXiv:1810.11203*, 2018.

Artem R Oganov, Chris J Pickard, Qiang Zhu, and Richard J Needs. Structure prediction drives materials discovery. *Nature Reviews Materials*, 4(5):331–348, 2019.

Shyue Ping Ong, William Davidson Richards, Anubhav Jain, Geoffroy Hautier, Michael Kocher, Shreyas Cholia, Dan Gunter, Vincent L Chevrier, Kristin A Persson, and Gerbrand Ceder. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319, 2013.

Martin H. Petersen, Ruiming Zhu, Haiwen Dai, Savyasanchi Aggarwal, Nong Wei, Andy P. Chen, Arghya Bhowmik, Juan Maria G. Lastra, and Kedar Hippalgaonkar. Dis-CSP: Disordered crystal structure predictions. In *AI for Accelerated Materials Design - ICLR 2025*, 2025.

Chris J Pickard. 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.

Chris J Pickard and RJ Needs. Ab initio random structure searching. *Journal of Physics: Condensed Matter*, 23(5):053201, 2011.

Zekun Ren, Siyu Isaac Parker Tian, Juhwan Noh, Felipe Oviedo, Guangzong Xing, Jiali Li, Qiaohao Liang, Ruiming Zhu, Armin G Aberle, Shijing Sun, et al. An invertible crystallographic representation for general inverse design of inorganic crystals with targeted properties. *Matter*, 5(1): 314–335, 2022.

François Rozet and Gilles Louppe. Score-based data assimilation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José Miguel Hernández-Lobato, Richard E. Turner, and Emile Mathieu. On conditional diffusion models for PDE simulations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Anuroop Sriram, Benjamin Kurt Miller, Ricky T. Q. Chen, and Brandon M Wood. 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.

Molei Tao and Tomoki Ohsawa. 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.

Yanchao Wang, Jian Lv, Li Zhu, and Yanming Ma. Crystal structure prediction via particle-swarm optimization. *Physical Review B—Condensed Matter and Materials Physics*, 82(9):094116, 2010.

Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2(1):1–7, 2016.

Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018.

Tian Xie, Xiang Fu, Octavian-Eugen Ganea, Regina Barzilay, and Tommi S Jaakkola. Crystal diffusion variational autoencoder for periodic material generation. In *International Conference on Learning Representations*, 2022.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022.

Minkai Xu, Alexander S Powers, Ron O Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pp. 38592–38610. PMLR, 2023.

Han Yang, Chenxi Hu, Yichi Zhou, Xixian Liu, Yu Shi, Jielan Li, Guanzhi Li, Zekun Chen, Shuizhou Chen, Claudio Zeni, et al. Mattersim: A deep learning atomistic model across elements, temperatures and pressures. *arXiv preprint arXiv:2405.04967*, 2024a.

Sherry Yang, KwangHwan Cho, Amil Merchant, Pieter Abbeel, Dale Schuurmans, Igor Mordatch, and Ekin Dogus Cubuk. Scalable diffusion for materials generation. In *The Twelfth International Conference on Learning Representations*, 2024b.

Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Zilong Wang, Aliaksandra Shysheya, Jonathan Crabbé, Shoko Ueda, Roberto Sordillo, Lixin Sun, Jake Smith, Bichlien Nguyen, Hannes Schulz, Sarah Lewis, Chin-Wei Huang, Ziheng Lu, Yichi Zhou, Han Yang, Hongxia Hao, Jielan Li, Chunlei Yang, Wenjie Li, Ryota Tomioka, and Tian Xie. A generative model for inorganic materials design. *Nature*, Jan 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-08628-5.

Yuchen Zhu, Tianrong Chen, Lingkai Kong, Evangelos A Theodorou, and Molei Tao. Trivialized momentum facilitates diffusion generative modeling on lie groups. *arXiv preprint arXiv:2405.16381*, 2024.

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

A Organization of the Supplementary Material

In this supplementary, we start by first providing a short introduction to Lie groups and manifolds in Appendix B, aiming at explaining the intuition behind Eq. (7), Eq. (13), and the TDM and KLDM model in general. In Appendix C, we then provide some intuitions as to why data on a torus can be represented either as a rotation matrix or an angle. In addition to that, we show that in the case of the torus, the matrix exponential corresponds to a simple rotation matrix that allows us to express the update on the coordinates as a translation. In Appendix D we instead derive step-by-step the target used to train the score network in the case of KLDM. Ablations of all the different design choices that led to the final KLDM model are presented in Appendix E, while we report additional DNG results in Appendix F. We then present pseudocode for all the core algorithms needed to implement our approach in Appendix G. We conclude with Appendix H, where we present the experimental details that can be used to reproduce our results, and we present a brief discussion on the main differences between KLDM and the other baselines.

B A primer on Lie groups and manifolds

In this section, we provide a simple and intuitive explanation of several of the mathematical concepts used to present our Kinetic Langevin diffusion dynamics. The fractional coordinates $\boldsymbol{f}$ used to define the positions of atoms inside the unit cell define a hypertorus, i.e. $\boldsymbol{f} \in [0, 1)^{3\times K} \cong \mathbb{T}^{3\times K}$. The hypertorus is a special case of a Lie Group, which is a smooth manifold equipped with a group structure G and smooth group operations.

Lie group In mathematics, a *group* is denoted by $(G, \cdot)$, with G being the non-empty set of elements that belongs 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 an unique element $e \in G$, such that $e \cdot a = a$ and $a \cdot e = a$ (existence of an 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 $\mathbb{T}^{3\times K}$ defined by the fractional coordinates $\boldsymbol{f}$ and this fact allows us to define the transition kernel Eq. (16) in closed form. 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 shed some intuition about the reason why we can model data on a manifold by just modeling quantities on 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 $L_a : G \to G$ 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 $R_a : G \to G, g \mapsto g \cdot a$.

Tangent space Before describing other key elements of a Lie group we have to introduce the concept of *tangent space*, which comes 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, and it is usually denoted as $\mathcal{T}_g\mathcal{M}$ for manifolds and $\mathcal{T}_g G$ for Lie Group. The tangent space offers us 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 usual definition of the Lie algebra, which we denote by $\mathfrak{g}$, consists of identifying it as the tangent space $\mathcal{T}_e G$ at the identity

element of the group $e \in G$. Since, as we have seen before, the tangent space is approximately Euclidean, then we have that we can treat the Lie algebra $\mathfrak{g}$ as Euclidean too. A different approach to introduce the Lie algebra $\mathfrak{g}$ is, instead, by considering the vector fields that are *invariant to left-translation*. This is still related to the previous definition and, in a certain way, tries to explain why we can think of the Lie algebra $\mathfrak{g}$ that way. We start by defining a vector field X defined on the group G as $X : G \to \mathcal{T}G$, which takes an element of the group $g \in G$ and returns a vector $X_g \in \mathcal{T}_g G$ in the tangent space of the point g. Let us recall again the left-translation operation L_a defined above, that given two group elements $a, g \in G$ returns the new element $a \cdot g$. We can now introduce the differential operation of L_a, defined as $(dL_a)_g : \mathcal{T}_g G \to \mathcal{T}_{a \cdot g} G$, which can be interpreted as operating on tangent vectors instead of directly on the group elements. Then, a vector field is said to be a left-translation invariant vector field if the following relation is satisfied for any two group elements $a, b \in G$

$$(dL_a)_g(X_g) = X_{a \cdot g} \quad . \tag{20}$$

Intuitively, a vector field is said to be left-invariant if we take a vector $X_g \in \mathcal{T}_g G$ in the tangent space of $g \in G$ and by applying the differential $(dL_a)_g$ of the left-operation L_a we get a vector $X_{a \cdot g}$ that lies on the tangent space of the element $a \cdot g \in G$ of the group that results from left-translating g by using element a. Therefore, the Lie algebra can be seen as the set of all the vector fields defined on G that are left-translation invariant, thus the set of vector fields that satisfy Eq. (20). In particular, it can be shown (Arvanitogeōrgos, 2003) that there exists a linear isomorphism between the Lie algebra $\mathfrak{g}$ and the tangent space of G at the identity element e, i.e. $\mathfrak{g} \cong \mathcal{T}_e G$. This allows us to rewrite the differential operation in terms of the identity element as follows:

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

where we use the property of the identity element that states that $a \cdot e = a$. The Eq. (21) tells us that we can consider any vector $X_e \in \mathcal{T}_e G$ in the tangent space of the identity element, which is an Euclidean space and by using the operation $(dL_a)_e(X_e)$ translate that into a tangent vector $X_a \in \mathcal{T}_a G$. At this point, we have both the element $a \in G$ and a tangent vector $X_a \in \mathcal{T}_a G$ in its tangent space. An additional operation that can be computed is the *exponential map* $\exp_a : \mathcal{T}_a G \to G$, which computes the point $a' \in G$ that we reach by starting from a with velocity X_a in one unit time. It can be interpreted as extending the notion of moving in an Euclidean space, which in that case corresponds to moving along a straight line, to the manifold.

How does this relate to modeling data that lives on a torus? The fractional coordinates of crystalline materials lie on the hypertorus $\mathbb{T}^{3 \times K}$, a particular case of an Abelian Lie group. As already mentioned in the main paper, diffusion-based models can either take an intrinsic or extrinsic parametrization of the manifold. The first approach requires defining the noising SDE directly on the manifold. In this case, the Brownian motion corresponds to the heat kernel on a manifold and the computation of it usually requires multiple approximations which leads to learning a suboptimal score function. The extrinsic approach, instead, defines the forward SDE on the ambient space, which is Euclidean by assumption, but then requires a projection operation that "puts back" the data in the manifold. In our case, we can leverage the fact that the torus is an Abelian Lie group and exploits some of the operations we have presented in the previous section, for example, the fact that we have a vector space that is Euclidean and that we have an operation to translate vectors defined in this space to the tangent space at any point of the manifold. Therefore, our approach can be interpreted as follows: we start by coupling the fractional coordinates, which are the elements of our Lie group, with a velocity, which corresponds to a vector in tangent space at those positions. The coupling follows Eq. (12), which can be interpreted as similar to Newton's second law. Using the fact that the torus is an Abelian Lie group, we have seen that these velocity vectors can be expressed in terms of other vectors that live in the Lie algebra $\mathfrak{g}$, which is Euclidean. This allows us to define a standard Euclidean diffusion model over these velocities in $\mathfrak{g}$ and use Eq. (20) to get the corresponding velocities tangential to the position we are interested. We can then translate the velocities defined on $\mathfrak{g}$ to tangential velocities at the specific data we are interested in modeling by using Eq. (21) and then use the exponential map to implicitly noise them. This is another simple explanation of the coupled process defined by Eq. (12). In the case we represent both the vectors on the Lie algebra $\mathfrak{g}$ and the elements of the group as $d \times d$ matrices, then the differential operation in Eq. (21) can be explicitly calculated as matrix multiplication. Therefore, if we have a vector $X_e \in \mathfrak{g}$ and an element $a \in G$,

we can just get the tangent vector in a by simply performing the following matrix multiplication $X_a = aX_e \in T_aG$. Moreover, the exponential map operation, as it involves a constant velocity, can be obtained in closed form with the exact solution given by the matrix exponential. If we are interested in computing the group element a' we reach by starting in a with velocity X_a, we can just compute $a' = \exp_a(X_a) = a\,\mathrm{expm}(X_a)$ which involves the computation of a matrix exponential.

C SAMPLING FROM THE NOISING AND DENOISING PROCESSES

C.1 INTUITION ABOUT THE CORRESPONDENCE BETWEEN $\mathbb{T}$ AND $\mathrm{SO}(2)$

In the following, we provide an intuitive explanation of the isomorphism between $\mathbb{T}$ and $\mathrm{SO}(2)$. 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 SO(2).

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,

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

and in turn x,

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

C.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 $\boldsymbol{f}_t = \boldsymbol{f}_0 \exp(\boldsymbol{r}_t)$ where then $\boldsymbol{r}_t \sim \mathrm{WN}\big(\boldsymbol{r}_t | \mu_{\boldsymbol{r}_t}, \sigma^2_{\boldsymbol{r}_t}\big)$. Therefore, the update involves a matrix exponential operation. In the case of the torus, we have that the structure of $\boldsymbol{r}_t$ can be written as a 2×2 skew-symmetric matrix of the form:

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

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

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

This corresponds to the following infinite sum

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

which can be rewritten as follows:

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

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

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

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 $\mathrm{d}t$, we are interested in computing the following update:

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

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:

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

D 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):

$$p_{t|0}(\boldsymbol{f}_t, \boldsymbol{v}_t|\boldsymbol{f}_0, \boldsymbol{v}_0) = \mathrm{WN}\big(\operatorname{logm}(\boldsymbol{f}_0^{-1}\boldsymbol{f}_t)|\mu_{\boldsymbol{r}_t}, \sigma^2_{\boldsymbol{r}_t}\big) \cdot \mathcal{N}_{\boldsymbol{v}}\big(\boldsymbol{v}_t|\mu_{\boldsymbol{v}_t}, \sigma^2_{\boldsymbol{v}_t}\big),$$

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

$$\mu_{\boldsymbol{r}_t} = \frac{1 - e^{-t}}{1 + e^{-t}}(\boldsymbol{v}_t + \boldsymbol{v}_0) \qquad \mu_{\boldsymbol{v}_t} = e^{-t}\boldsymbol{v}_0 \tag{22}$$

$$\sigma^2_{\boldsymbol{r}_t} = 2t + \frac{8}{e^t + 1} - 4 \qquad \sigma^2_{\boldsymbol{v}_t} = 1 - e^{-2t} \tag{23}$$

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

The target score used in the DSM loss (Eq. (9)) for KLDM is $\nabla_{\boldsymbol{v}_t} \log p_{t|0}(\boldsymbol{f}_t, \boldsymbol{v}_t|\boldsymbol{f}_0, \boldsymbol{v}_0)$. 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 be therefore computed as follows:

$$\begin{aligned}\nabla_{\boldsymbol{v}_t} \log p_{t|0}(\boldsymbol{f}_t, \boldsymbol{v}_t|\boldsymbol{f}_0, \boldsymbol{v}_0) &= \nabla_{\boldsymbol{v}_t} \log \big[\mathrm{WN}\big(\operatorname{logm}(\boldsymbol{f}_0^{-1}\boldsymbol{f}_t)|\mu_{\boldsymbol{r}_t}, \sigma^2_{\boldsymbol{r}_t}\big) \cdot \mathcal{N}_{\boldsymbol{v}}\big(\boldsymbol{v}_t|\mu_{\boldsymbol{v}}, \sigma^2_{\boldsymbol{v}}\big)\big] \\ &= \underbrace{\nabla_{\boldsymbol{v}_t} \log \mathrm{WN}\big(\operatorname{logm}(\boldsymbol{f}_0^{-1}\boldsymbol{f}_t)|\mu_{\boldsymbol{r}_t}, \sigma^2_{\boldsymbol{r}_t}\big)}_{\boldsymbol{s_c}\text{ coordinates term}} + \underbrace{\nabla_{\boldsymbol{v}_t} \log \mathcal{N}_{\boldsymbol{v}}\big(\boldsymbol{v}_t|\mu_{\boldsymbol{v}}, \sigma^2_{\boldsymbol{v}}\big)}_{\boldsymbol{s_v}\text{ velocity term}}\end{aligned} \tag{24}$$

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

$$\boldsymbol{s_v} = \frac{-\boldsymbol{v}_t + \mu_{\boldsymbol{v}}}{\sigma^2_{\boldsymbol{v}}} = -\frac{\epsilon}{\sigma_{\boldsymbol{v}}} \tag{25}$$

We can now focus on the $\boldsymbol{s_c}$ term, where for simplicity we define $\boldsymbol{r}_t = \operatorname{logm}(\boldsymbol{f}_0^{-1}\boldsymbol{f}_t)$ as above:

$$\begin{aligned}\boldsymbol{s_c} &= \nabla_{\boldsymbol{v}_t} \log \mathrm{WN}\big(\operatorname{logm}(\boldsymbol{f}_0^{-1}\boldsymbol{f}_t)|\mu_{\boldsymbol{r}_t}, \sigma^2_{\boldsymbol{r}_t}\big) = \nabla_{\boldsymbol{v}_t} \log \mathrm{WN}\big(\boldsymbol{r}_t|\mu_{\boldsymbol{r}_t}, \sigma^2_{\boldsymbol{r}_t}\big) \\
&= \nabla_{\boldsymbol{v}_t} \log \left(\frac{1}{\sqrt{2\pi}\sigma_{\boldsymbol{r}_t}} \sum_{k=-\infty}^{+\infty} \exp\Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big) \right) \\
&= \nabla_{\boldsymbol{v}_t} \log \left(\sum_{k=-\infty}^{+\infty} \exp\Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big) \right) \\
&= \underbrace{\frac{1}{\sum_{k=-\infty}^{+\infty} \exp\Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big)}}_{C} \nabla_{\boldsymbol{v}_t} \left[\sum_{k=-\infty}^{+\infty} \exp\Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big) \right] \\
&= C \cdot \left[\sum_{k=-\infty}^{+\infty} \nabla_{\boldsymbol{v}_t} \exp\Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big) \right] \\
&= C \cdot \left[\sum_{k=-\infty}^{+\infty} \exp\Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big) \nabla_{\boldsymbol{v}_t} \Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big) \right] \\
&= C \cdot \left[\sum_{k=-\infty}^{+\infty} \exp\Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big) \nabla_{\boldsymbol{v}_t} \Big(-\frac{(\boldsymbol{r}_t - \frac{1-e^{-t}}{1+e^{-t}}(\boldsymbol{v}_t + \boldsymbol{v}_0) + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big) \right] \\
&= C \cdot \left[\sum_{k=-\infty}^{+\infty} \exp\Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big) \frac{1-e^{-t}}{1+e^{-t}} \frac{1}{\sigma^2_{\boldsymbol{r}_t}} \Big(\boldsymbol{r}_t - \frac{1-e^{-t}}{1+e^{-t}}(\boldsymbol{v}_t + \boldsymbol{v}_0) + 2\pi k \Big) \right] \\
&= \frac{1}{\sum_{k=-\infty}^{+\infty} \exp\Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big)} \cdot \\
&\qquad \cdot \left[\sum_{k=-\infty}^{+\infty} \frac{1-e^{-t}}{1+e^{-t}} \frac{1}{\sigma^2_{\boldsymbol{r}_t}} \Big(\boldsymbol{r}_t - \frac{1-e^{-t}}{1+e^{-t}}(\boldsymbol{v}_t + \boldsymbol{v}_0) + 2\pi k \Big) \exp\Big(-\frac{(\boldsymbol{r}_t - \mu_{\boldsymbol{r}_t} + 2\pi k)^2}{2\sigma^2_{\boldsymbol{r}_t}} \Big) \right]\end{aligned} \tag{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 $\boldsymbol{s_v}$ can be rewritten as follows:

$$\boldsymbol{s_v} = \frac{-\boldsymbol{v}_t + \mu_{\boldsymbol{v}}}{\sigma^2_{\boldsymbol{v}}} = \frac{-\boldsymbol{v}_t + e^{-t}\boldsymbol{v}_0}{\sigma^2_{\boldsymbol{v}}} = -\frac{\boldsymbol{v}_t}{\sigma^2_{\boldsymbol{v}}} \tag{27}$$

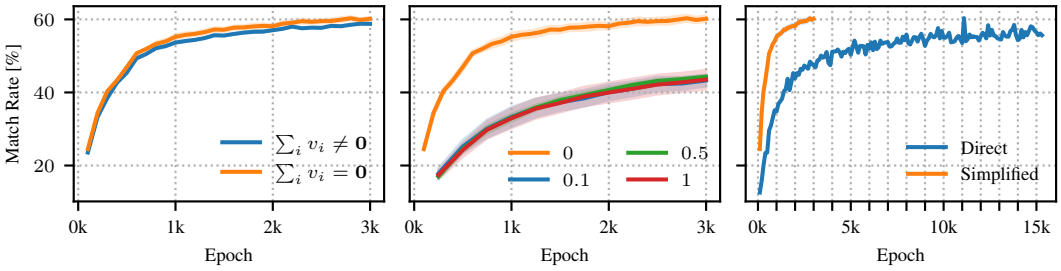

Figure 1: *(Left)* Ablation of velocity fields with zero net translation for zero initial velocities. *(Center)* Impact of the variance of the initial velocities distributions. When variance is 0, we refers to the zero-initial velocity case, i.e. it is a delta distribution. *(Right)* Impact of the score parameterization on the performance. We report mean and standard error from the mean of the validation match-rate over 6 seeds. We note that enforcing zero initial velocities, which allows us to use the simplified parameterization, is a key element for faster convergence and competitive results. With these two elements in place, a marginal gain can then be obtained by considering a velocity fields with zero net translation.

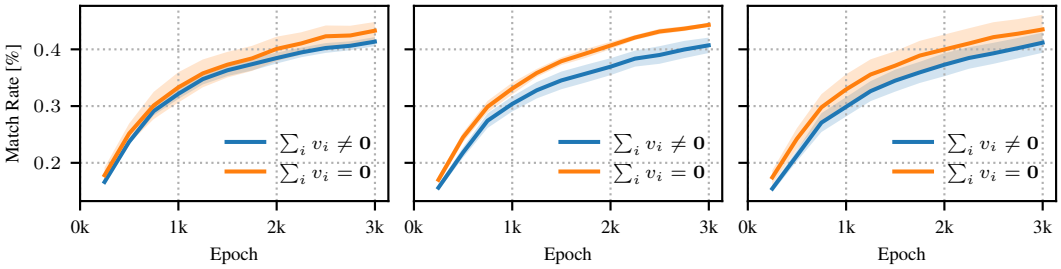

Figure 2: Ablation of velocity fields with zero net translation for non-zero initial velocities. *(Left)* Initial velocities are sampled from a $\mathcal{N}(\mathbf{0}, 0.1 \cdot \mathbf{I})$, i.e. $\sigma^2_{\boldsymbol{v}_0} = 0.1$. *(Center)* Initial velocities are sampled from a $\mathcal{N}(\mathbf{0}, 0.5 \cdot \mathbf{I})$, i.e. $\sigma^2_{\boldsymbol{v}_0} = 0.5$ *(Right)* Initial velocities are sampled from a $\mathcal{N}(\mathbf{0}, 1 \cdot \mathbf{I})$, i.e. $\sigma^2_{\boldsymbol{v}_0} = 1.0$. We report mean and standard error from the mean of the validation match-rate over 6 seeds. We note that regarding of the initial distribution variance, there is a benefit given by enforcing a zero net translation velocity field.

where we used the definition of $\mu_{\boldsymbol{v}} = e^{-t}\boldsymbol{v}_0$ and the fact that we assumed that $\boldsymbol{v}_0 = \mathbf{0}$, resulting in $\mu_{\boldsymbol{v}} = \mathbf{0}$.

E DESIGN CHOICES ABLATION

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

	MP-20		MPTS-52	
sampler	MR [%] ↑	RMSE ↓	MR [%] ↑	RMSE ↓
	DIRECT PARAMETRIZATION			
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 PARAMETRIZATION			
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 $_{\pm.002}$

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

Score parameterization We compare the simplified parametrization of the score formulated in Eq. (19) compared to the direct parametrization of Eq. (18). We present in Fig. 1, the validation match-rate during training on the MP-20 dataset. From Fig. 1 *(Right)*, 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 match rate (MR) and RMSE on the test set. We note in Table 3, that no matter the sampling scheme, the simplified parametrization always outperforms the direct one, in addition to be much faster to converge.

Initial-velocity distributions KLDM uses zero-initial velocities, which intuitively corresponds to considering the coordinates $\boldsymbol{f}$ at time $t = 0$ as being at rest, allowing the noise to gradually propagate from the velocity to these variables. This also allowis us to use the simplified parametrization for the score formulated in Eq. (19) ablated above. In Fig. 1 *(Center)*, 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 parametrization, and in the case of non-zero initial velocities and direct parametrization. We report results in Fig. 1 *(Left)*. 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. 2.

F Additional DNG results

In this section, we present additional results regarding the DNG task. 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 4, where it can be observed that, unlike for the CSP task, the difference in performance between the compared methods that KLDM is competitive with the compared models.

G Algorithms

This section presents all the algorithms at our method's core. 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 $s_\theta(\boldsymbol{f}_t, \boldsymbol{v}_t, \mathbf{l}_t, \mathbf{a}_t)$ 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 $\boldsymbol{v}_t$ and fractional coordinates $\boldsymbol{f}_t$ as proposed by Zhu et al. (2024) while using a classic Euler–Maruyama step for lattice parameters $\boldsymbol{l}_t$ and atom type $\boldsymbol{a}_t$ 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).

Table 4: Results for the De-novo Generation (DNG) task. Baseline results are extracted from the respective papers.

	VALIDITY [%] ↑		COVERAGE [%] ↑		PROPERTY ↓	
	STRUC.	COMP.	COV-R	COV-P	d_ρ	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	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	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	99.88	84.86	98.94	99.50	0.4658	0.1280

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

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

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

Sample $\epsilon_{\boldsymbol{v}} \sim \mathcal{N}_{\boldsymbol{v}}(\boldsymbol{0}, \boldsymbol{I})$, $\epsilon_{\boldsymbol{r}_t} \sim \mathcal{N}_{\boldsymbol{v}}(\boldsymbol{0}, \boldsymbol{I})$ ▷ $\mathcal{N}_{\boldsymbol{v}}$ is a normal distribution such that $\sum_i \boldsymbol{v}_i = \boldsymbol{0}$.

$\boldsymbol{v}_t = \alpha_{\boldsymbol{v}}(t)\boldsymbol{v}_0 + \sigma_{\boldsymbol{v}}(t) \cdot \epsilon_{\boldsymbol{v}}$

if initial velocities are zero **then**

  $\text{target}_{\boldsymbol{v}} = -\boldsymbol{v}_t / \sigma^2_{\boldsymbol{v}}(t)$

else

  $\text{target}_{\boldsymbol{v}} = -\epsilon_{\boldsymbol{v}} / \sigma_{\boldsymbol{v}}(t)$ ▷ See Eq. (26).

end if

$\boldsymbol{r}_t = w(\mu_{\boldsymbol{r}}(t, \boldsymbol{v}_0, \boldsymbol{v}_t) + \sigma_{\boldsymbol{r}_t}(t, \boldsymbol{v}_0, \boldsymbol{v}_t) \cdot \epsilon_{\boldsymbol{r}_t})$ ▷ w indicates the wrap function.

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

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

$\text{target}_{\boldsymbol{s}} = (1 - \exp(-t))/(1 + \exp(-t)) \cdot \nabla_{\boldsymbol{r}(\boldsymbol{v})} \mathcal{N}_w$ ▷ Equivalent computation of Eq. (26).

$\text{target}_{\boldsymbol{v}} = \text{target}_{\boldsymbol{v}} + \text{target}_{\boldsymbol{s}}$

Sampling $\boldsymbol{l}_t$

Sample $\epsilon_{\boldsymbol{l}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

$\boldsymbol{l}_t = \alpha(t)\boldsymbol{l}_0 + \sigma(t) \cdot \epsilon_{\boldsymbol{l}}$

$\text{target}_{\boldsymbol{l}} = \epsilon_{\boldsymbol{l}}$

if task is DNG **then**

  ## Sampling $\boldsymbol{a}_t$

  Sample $\epsilon_{\boldsymbol{a}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

  $\boldsymbol{a}_t = \alpha(t)\boldsymbol{a}_0 + \sigma(t) \cdot \epsilon_{\boldsymbol{l}}$

  $\text{target}_{\boldsymbol{a}} = \epsilon_{\boldsymbol{a}}$

  return $(\boldsymbol{v}_t, \boldsymbol{f}_t, \boldsymbol{l}_t, \boldsymbol{a}_t)$, $(\text{target}_{\boldsymbol{v}}, \text{target}_{\boldsymbol{l}}, \text{target}_{\boldsymbol{a}})$ ▷ Return both noisy samples and training targets.

else

  return $(\boldsymbol{v}_t, \boldsymbol{f}_t, \boldsymbol{l}_t)$, $(\text{target}_{\boldsymbol{v}}, \text{target}_{\boldsymbol{l}})$

end if

Algorithm 2 Training algorithm

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

for training iterations **do**
  $\boldsymbol{x}_0 = \{(\boldsymbol{f}_0, \boldsymbol{l}_0, \boldsymbol{a}_0)\}_{i=1}^{B} \sim q(\boldsymbol{x})$, $t \sim \mathcal{U}(t)$, $\boldsymbol{v}_0 \sim p_0(\boldsymbol{v})$ ▷ B indicates the batch size.
  if task is DNG **then**
    $(\boldsymbol{v}_t, \boldsymbol{f}_t, \boldsymbol{l}_t, \boldsymbol{a}_t)$, $(\text{target}_{\boldsymbol{v}}, \text{target}_{\boldsymbol{l}}, \text{target}_{\boldsymbol{a}})$ = `training_targets`$(\boldsymbol{f}, \boldsymbol{v}, \boldsymbol{l}, \boldsymbol{a}, t)$
    $\text{out}_{\boldsymbol{v}}, \text{out}_{\boldsymbol{l}}, \text{out}_{\boldsymbol{a}} = s_\theta(t, \boldsymbol{f}_t, \boldsymbol{v}_t, \mathbf{l}_t, \mathbf{a}_t)$ ▷ The network takes $t, \boldsymbol{f}_t, \boldsymbol{v}_t, \mathbf{l}_t, \mathbf{a}_t$ as input and output all the scores.
    $\mathcal{L}_{\boldsymbol{a}} = \|\text{out}_{\boldsymbol{a}} - \text{target}_{\boldsymbol{a}}\|_2^2$
  else
    $(\boldsymbol{v}_t, \boldsymbol{f}_t, \boldsymbol{l}_t)$, $(\text{target}_{\boldsymbol{v}}, \text{target}_{\boldsymbol{l}})$ = `training_targets`$(\boldsymbol{f}, \boldsymbol{v}, \boldsymbol{l}, \text{None}, t)$
    $\text{out}_{\boldsymbol{v}}, \text{out}_{\boldsymbol{l}} = s_\theta(t, \boldsymbol{f}_t, \boldsymbol{v}_t, \mathbf{l}_t, \mathbf{a}_t)$
  end if
  $\mathcal{L}_{\boldsymbol{l}} = \|\text{out}_{\boldsymbol{l}} - \text{target}_{\boldsymbol{l}}\|_2^2$
  $\text{out}_{\boldsymbol{v}} = (1 - \exp(-t))/(1 + \exp(-t)) \cdot \text{out}_{\boldsymbol{v}} - \boldsymbol{v}_t/\sigma^2_{\boldsymbol{v}_t}$ ▷ Construct the score, see Eq. (19)
  $\mathcal{L}_{\boldsymbol{v}} = \lambda(t)\|\text{out}_{\boldsymbol{v}} - \text{target}_{\boldsymbol{v}}\|_2^2$ ▷ $\lambda(t)$ computed as Jiao et al. (2024a)
  if task is DNG **then**
    $\mathcal{L}_{\text{tot}} = \mathcal{L}_{\boldsymbol{v}} + \mathcal{L}_{\boldsymbol{l}} + \mathcal{L}_{\boldsymbol{a}}$
  else
    $\mathcal{L}_{\text{tot}} = \mathcal{L}_{\boldsymbol{v}} + \mathcal{L}_{\boldsymbol{l}}$
  end if
  Compute gradients of $\mathcal{L}_{\text{tot}}$ with respect step θ and perform a gradient step.
end for

Algorithm 3 Sampling algorithm

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

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 $\boldsymbol{v}_0 \sim \mathcal{N}_{\boldsymbol{v}}(\mathbf{0}, \boldsymbol{I})$, $\boldsymbol{f}_0 \sim \mathcal{U}(\mathbf{0}, \mathbf{1})$, $\boldsymbol{l}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

Set $\boldsymbol{f}_0 = w(\boldsymbol{f}_0)$ ▷ w indicates the wrap function.

if task is DNG **then**

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

end if

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

if task is DNG **then**

$\text{out}_{\boldsymbol{v}}^{(n-1)}, \text{out}_{\boldsymbol{l}}^{(n-1)}, \text{out}_{\boldsymbol{a}}^{(n-1)} = s_\theta((1 - (n-1) * \mathrm{d}t), \boldsymbol{f}_{n-1}, \boldsymbol{v}_{n-1}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

else

$\text{out}_{\boldsymbol{v}}^{(n-1)}, \text{out}_{\boldsymbol{l}}^{(n-1)} = s_\theta((1 - (n-1) * \mathrm{d}t), \boldsymbol{f}_{n-1}, \boldsymbol{v}_{n-1}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

end if

Update step for $\boldsymbol{v}$ and $\boldsymbol{f}$

$\text{out}_{\boldsymbol{v}} = (1 - \exp(-(1 - (n-1)\mathrm{d}t)))/(1 + \exp(-(1 - (n-1)\mathrm{d}t))) \cdot \text{out}_{\boldsymbol{v}} - \boldsymbol{v}_t/\sigma^2_{\boldsymbol{v}_t}$ ▷ Follow Eq. (19)

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

$\boldsymbol{v}_n = \exp(\mathrm{d}t)\boldsymbol{v}_{n-1} + 2(\exp(2\mathrm{d}t) - 1)\text{out}_{\boldsymbol{v}}^{(n-1)} + \sqrt{\exp(2\mathrm{d}t) - 1}\epsilon_{\boldsymbol{v}}$ ▷ Update on $\boldsymbol{v}$

$\boldsymbol{f}_n = w(\boldsymbol{f}_{n-1} - \boldsymbol{v}_n \mathrm{d}t)$ ▷ Update on $\boldsymbol{f}$

Update step for $\boldsymbol{l}$

Sample $\epsilon_{\boldsymbol{l}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

$\boldsymbol{l}_n = \boldsymbol{l}_{n-1} - (f(t) - g^2(t)s(\text{out}_{\boldsymbol{l}}^{(n-1)}))\mathrm{d}t + \sqrt{dt}\epsilon_{\boldsymbol{l}}$ ▷ Euler–Maruyama step for $\boldsymbol{l}$, $s(\cdot)$ gets score from ϵ-param.

if task is DNG **then**

Update step for $\boldsymbol{a}$

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

$\boldsymbol{a}_n = \boldsymbol{a}_{n-1} - (f(t) - g^2(t)s(\text{out}_{\boldsymbol{a}}^{(n-1)}))\mathrm{d}t + \sqrt{dt}\epsilon_{\boldsymbol{a}}$ ▷ Euler–Maruyama step for $\boldsymbol{a}$, $s(\cdot)$ gets score from ϵ-param.

end if

end for

if task is DNG **then**

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

else

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

end if

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

Require: trained score network $s_\theta(t, \boldsymbol{f}_t, \boldsymbol{v}_t, \mathbf{l}_t, \mathbf{a}_t)$, N discretization steps, $dt = 1/N$ step-size, prior distributions over velocities $p_T(\boldsymbol{v}) = \mathcal{N}_{\boldsymbol{v}}(\mathbf{0}, \boldsymbol{I})$, over fractional coordinates $p_T(\boldsymbol{f}) = \mathcal{U}(\mathbf{0}, \mathbf{1})$, over lattice parameters $p_T(\boldsymbol{l}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and over atom types $p_T(\boldsymbol{a}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We require the scheduler $\alpha_{\boldsymbol{v}}(t)$ and $\sigma_{\boldsymbol{v}}(t)$ for $\boldsymbol{v}$ and also the knowledge of the forward drift $f(t)$ and the diffusion coefficient $g(t)$ of the SDEs describing the evolution of $\boldsymbol{l}$ and $\boldsymbol{a}$. 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 $\boldsymbol{v}_0 \sim \mathcal{N}_{\boldsymbol{v}}(\mathbf{0}, \boldsymbol{I})$, $\boldsymbol{f}_0 \sim \mathcal{U}(\mathbf{0}, \mathbf{1})$, $\boldsymbol{l}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ ▷ First steps are similar to Algorithm 3

Set $\boldsymbol{f}_0 = w(\boldsymbol{f}_0)$ ▷ w indicates the wrap function.

if task is DNG **then**

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

end if

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

  if task is DNG **then**

    $\text{out}_{\boldsymbol{v}}^{(n-1)}, \text{out}_{\boldsymbol{l}}^{(n-1)}, \text{out}_{\boldsymbol{a}}^{(n-1)} = s_\theta((1 - (n-1) * \mathrm{d}t), \boldsymbol{f}_{n-1}, \boldsymbol{v}_{n-1}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

  else

    $\text{out}_{\boldsymbol{v}}^{(n-1)}, \text{out}_{\boldsymbol{l}}^{(n-1)} = s_\theta((1 - (n-1) * \mathrm{d}t), \boldsymbol{f}_{n-1}, \boldsymbol{v}_{n-1}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

  end if

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

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

  Compute $r = \alpha_{\boldsymbol{v}}(n)/\alpha_{\boldsymbol{v}}(n-1)$

  Compute $c = (r\sigma_{\boldsymbol{v}}(n-1) - \sigma_{\boldsymbol{v}}(n))\sigma_{\boldsymbol{v}}(n-1)$

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

  $\boldsymbol{f}_n^{\text{pred}} = w(\boldsymbol{f}_{n-1} + \boldsymbol{v}_{n-1}^{\text{pred}}\mathrm{d}t)$

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

  if task is DNG **then**

    $\text{out}_{\boldsymbol{v}}, \text{out}_{\boldsymbol{l}}, \text{out}_{\boldsymbol{a}} = s_\theta((1 - n * \mathrm{d}t), \boldsymbol{f}_n^{\text{pred}}, \boldsymbol{v}_n^{\text{pred}}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

  else

    $\text{out}_{\boldsymbol{v}}, \text{out}_{\boldsymbol{l}} = s_\theta((1 - n * \mathrm{d}t), \boldsymbol{f}_n^{\text{pred}}, \boldsymbol{v}_n^{\text{pred}}, \mathbf{l}_{n-1}, \mathbf{a}_{n-1})$

  end if

  $\text{out}_{\boldsymbol{v}} = (1 - \exp(-(1 - n * \mathrm{d}t)))/(1 + \exp(-(1 - n * \mathrm{d}t))) \cdot \text{out}_{\boldsymbol{v}} - \boldsymbol{v}_t/\sigma_{\boldsymbol{v}_t}^2$ ▷ Follow Eq. (19)

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

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

  $\boldsymbol{v}_n = \boldsymbol{v}_n^{\text{pred}} + \delta\text{out}_{\boldsymbol{v}} + \sqrt{2\delta}\epsilon_{\boldsymbol{v}}$ ▷ Update on $\boldsymbol{v}$

  $\boldsymbol{f}_n = w(\boldsymbol{f}_n^{\text{pred}} - \boldsymbol{v}_n\mathrm{d}t)$ ▷ Update on $\boldsymbol{f}$

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

  Sample $\epsilon_{\boldsymbol{l}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

  $\boldsymbol{l}_n = \boldsymbol{l}_{n-1} - (f(t) - g^2(t)s(\text{out}_{\boldsymbol{l}}))\mathrm{d}t + \sqrt{dt}\epsilon_{\boldsymbol{l}}$ ▷ Euler–Maruyama step for $\boldsymbol{l}$, $s(\cdot)$ gets score from ϵ-param.

  if task is DNG **then**

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

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

    $\boldsymbol{a}_n = \boldsymbol{a}_{n-1} - (f(t) - g^2(t)s(\text{out}_{\boldsymbol{a}}))\mathrm{d}t + \sqrt{dt}\epsilon_{\boldsymbol{a}}$ ▷ Euler–Maruyama step for $\boldsymbol{a}$, $s(\cdot)$ gets score from ϵ-param.

  end if

end for

if task is DNG **then**

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

else

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

end if

H EXPERIMENTAL DETAILS

H.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.

H.2 ARCHITECTURE

As we have mentioned in Section 3.3, our score network is parametrized using an architecture whose backbone is similar to that of previous work (Jiao et al., 2024a, 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 take also this velocity variable $\boldsymbol{v}_t$ 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 apple-to-apple 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 $\boldsymbol{f}_t, \boldsymbol{v}_t, \boldsymbol{l}_t, \boldsymbol{a}$ being the input of our network. The input features are computed by

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

where $f_{\mathrm{atom}}, f_{\mathrm{pos}}$ 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

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

where $\boldsymbol{m}_{ij}^{(s)}$ and $\boldsymbol{h}_j^{(s-1)}$ 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 using the velocity $\boldsymbol{v}$ as input.

The $\mathrm{SinusoidalEmbedding}$ is a sinusoidal embedding layer defined as

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

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:

$$\begin{aligned} \boldsymbol{s}_{\boldsymbol{v}}^{(i)} &= \varphi_{\boldsymbol{v}}(\boldsymbol{h}_i^{(S)}) \\ \boldsymbol{s}_{\boldsymbol{l}} &= \varphi_{\boldsymbol{l}}\left(\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{h}_i^{(S)}\right) \\ \boldsymbol{s}_{\boldsymbol{a}}^{(i)} &= \varphi_{\boldsymbol{a}}(\boldsymbol{h}_i^{(S)}) \end{aligned}$$

where we want to stress that $\varphi_{\boldsymbol{v}}$ is a 2-layer neural network while $\varphi_{\boldsymbol{l}}$ and $\varphi_{\boldsymbol{a}}$ are single-layer MLPs.

For training, we used the same loss weights that were used by DIFFSCP. We consider $\lambda_{\boldsymbol{v}} = 1$ and $\lambda_{\boldsymbol{l}} = 1$ 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 $\lambda_{\boldsymbol{v}} = 1$, $\lambda_{\boldsymbol{l}} = 1$, and $\lambda_{\boldsymbol{a}} = 20$. In the case of analog-bits, we used $\lambda_{\boldsymbol{a}} = 1$, while when using discrete diffusion for the atom types, we scale the losses using $\lambda_{\boldsymbol{a}} = 0.33$. The scaling is needed to make the different loss terms have similar magnitudes.

H.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 5: Dataset hyperparameters

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

H.4 MAJOR DIFFERENCE BETWEEN KLDM AND THE COMPETING MODELS

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 $\boldsymbol{f}_0$, 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 generalize 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 using three different representations for the discrete atom type features. Additionally, for DNG, FlowMM uses additional 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