From Generation to Restoration: Residual Diffusion for Neural Channel Decoding

Qinshan Zhang[1,2,*], **Shipeng Guo**[4,*], **Xuantai Wu**[1], **Bin Chen**[3,2,†], **Zhuochen Fan**[2,†], **Yong Jiang**[1,2], **Shu-Tao Xia**[1,2], **Qing Li**[2]

[1]Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[2]Pengcheng Laboratory, Shenzhen, China
[3]Harbin Institute of Technology (Shenzhen), Shenzhen, China
[4]Shandong University, Qingdao, China

{zhangqs24, xt-wu25}@mails.tsinghua.edu.cn, g_ship@mail.sdu.edu.cn,
chenbin2021@hit.edu.cn, fanzc@pku.edu.cn,
{jiangy, xiast}@sz.tsinghua.edu.cn, liq@pcl.ac.cn

Abstract

Neural decoders have shown strong potential for error correction in short- and moderate-length regimes, yet a fundamental tension remains between decoding accuracy and inference latency. Recent attempts to leverage diffusion probabilistic models for channel decoding typically adopt a fully generative paradigm, initializing the reverse process from an observation-agnostic prior, which overlooks a key structural property of channel decoding: the received signal already contains substantial information about the target codeword. In this work, we propose Channel Residual Diffusion Model (ChRes-DM), a principled diffusion-based decoding framework that reinterprets channel decoding as a directed restoration process anchored at the noisy observation. Instead of sampling from a generic prior, ChRes-DM constructs a geometric residual bridge between the received signal and the clean codeword, explicitly modeling the conditional transport induced by the channel and leading to a deterministic Probability Flow Ordinary Differential Equation (PF-ODE) that governs the decoding dynamics. By eliminating redundant stochastic sampling inherent in conventional diffusion models, ChRes-DM enables efficient iterative decoding with flexible step-skipping, offering fine-grained control over the accuracy–latency trade-off. Extensive experiments across representative channel coding benchmarks demonstrate that ChRes-DM achieves competitive or superior decoding performance compared to existing neural decoders while significantly reducing inference iterations, highlighting diffusion-based residual transport as a promising and scalable paradigm for neural channel decoding.

1 Introduction

Reliable communication is a fundamental requirement of modern digital communication systems. Balancing decoding performance and computational efficiency remains a challenging task, despite decades of progress in the design of soft-decision decoding algorithms for short and moderate-length codes. Recently, model-free neural decoding [1–6] approaches have been explored as complementary alternatives to classical methods. While promising, the computational efficiency of existing model-free decoding approaches remains a key bottleneck, motivating further research into more efficient algorithmic designs without sacrificing decoding performance.

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Unlike model-based neural decoders that are tightly coupled to specific conventional algorithmic frameworks [7–12], model-free approaches adopt generic neural network backbones, enabling greater capacity. Such decoders are commonly referred to as model-free neural decoders. Since the introduction of ECCT [2], this line of work—together with its Transformer-based architectural variants [3, 4, 6]—has demonstrated state-of-the-art empirical performance among model-free decoders under *single-pass* decoding settings. These models introduce a parity-check-based attention mechanism, which enhances the Transformer's ability to learn the underlying code structure, leading to notable empirical performance gains.

Iterative soft-input soft-output (SISO) decoding traditionally refers to message-passing–based algorithms, most notably belief propagation (BP) [13]. Broadly speaking, such methods refine estimates of the transmitted codeword by *iteratively* updating soft information based on the code parity-check constraints. This procedure can be interpreted as progressively generating updated codeword estimates in a continuous space through discretized iterations. Therefore, the overall decoding process admits a generative interpretation, whose objective is to trace a path from the channel observation (i.e., received signal) to a clean codeword estimate.

Following this line of thought, prior work has taken an initial step in this direction by adopting Diffusion Probabilistic Models (DDPMs) [5]. By modeling the transmission and decoding processes as the forward and reverse diffusion dynamics, respectively, this approach provides a new modeling perspective of the channel decoding problem. The resulting decoder operates iteratively, denoising the noisy observations, thereby improving decoding performance. Consequently, trading additional iterations for improved decoding accuracy emerges as a promising and well-motivated research direction.

However, when DDPMs are applied directly to channel decoding, certain efficiency limitations can arise. Classical diffusion models are primarily developed for generative tasks, where the forward process is designed to progressively transform structured data into a high-entropy distribution dominated by noise, effectively erasing the signal structure. In the context of error correction, this design choice may not be fully aligned with the nature of the problem. Channel decoding, generally speaking, is a restoration task: the received observation $\mathbf{y}$, although corrupted by noise, already retains substantial information about the transmitted codeword $\mathbf{x}_0$. Initiating the reverse process from an observation-agnostic prior implies a "blind" search that neglects the geometric proximity of the observation, potentially limiting the efficiency of the inference.

In this paper, motivated by this, we propose the Channel Residual Diffusion Model (ChRes-DM), a framework that reformulates the diffusion process as a directed transport from the noisy observation to the clean codeword, rather than a reconstruction from independent pure Gaussian noise. By defining the forward dynamics to asymptotically approach the channel observation, we construct a geometric "bridge" that allows the decoding dynamics to be described by the deterministic PF-ODE. This formulation effectively bypasses the inefficiency of stochastic Markovian sampling in DDPM, enabling a more efficient iterative decoding scheme. Consequently, the deterministic trajectory of the ODE allows for flexible step-skipping updates across time steps. This capability enables our method to reduce the required inference iterations by several folds compared to standard diffusion baselines, while achieving decoding performance that is comparable to the previous work.

Our main contributions are summarized as follows:

- We propose the ChRes-DM, a novel framework that anchors the diffusion process to the channel observation, formulating the forward and reverse dynamics as a directionally constrained state transport between the noisy observation and the clean codewords.

- We derive a tractable iterative decoding algorithm with deterministic update trajectories under the proposed framework. By leveraging the PF-ODE, we formulate a unified update rule that efficiently integrates the residual correction and noise removal.

- Empirical results demonstrate that our method improves inference efficiency while maintaining performance comparable to state-of-the-art diffusion baselines. These findings validate that through rational design, the decoder can effectively balance computational efficiency and error correction capability via iterative refinement.

2 Related Work

Model-based Neural Channel Decoder. Neural network–based channel decoders can be broadly categorized into model-based and model-free approaches. Model-based decoders are primarily represented by neural belief propagation (NBP) decoders and their variants [14, 7–10, 15, 11]. By unfolding the iterative BP updates into a parameterized trellis-like structure, NBP-like decoders form feedforward networks with learnable parameters, as a generalization of conventional BP.

Model-Free Neural Channel Decoder. Model-free decoders extend the network architecture to generic neural networks. Rather than explicitly mimicking the BP message-passing mechanism, these approaches typically focus on estimating an equivalent *multiplicative* noise model [13], thereby effectively mitigating overfitting to specific codewords and enabling codeword-independent training [1]. Among model-free decoders, Transformer-based architectures, most notably ECCT and its variants, currently represent the state of the art [2, 4, 6, 3]. By introducing masked attention mechanisms constrained by the code structure—specifically, the parity-check matrix—these models encourage the network to attend to symbol positions that are coupled through parity constraints. This mechanism substantially enhances the model's awareness of code structure and induces a strong inductive bias toward valid codeword configurations.

Diffusion Models. Denoising Diffusion Probabilistic Models (DDPMs) [16], Denoising Diffusion Implicit Models (DDIMs) [17], and score-based generative models [18] have demonstrated remarkable capabilities in modeling complex data distributions. A unified interpretation of these approaches was provided under the framework of Stochastic Differential Equations (SDEs), revealing that the reverse process can also be described by a deterministic PF-ODE [19]. In the realm of restoration, recent works such as InDI [20] and I2SB [21] bypass the pure Gaussian prior, constructing direct transport bridges by estimating the clean target or its linear transformation. Building on this, ResShift [22] and RDDM [23] explicitly formulate the diffusion process as a directed shift of the residual between the degraded observation and the ground truth. Beyond restoration, diffusion priors have also been explored in image compression and 3D reconstruction [24, 25]. In this work, we synthesize these observation-anchored paradigms by adapting the residual shifting philosophy to the PF-ODE, to design a geometric bridge specifically tailored for channel decoding.

3 Preliminaries

3.1 Channel Coding

We consider a standard channel coding scenario over the additive white Gaussian noise (AWGN) channel. Let $\mathbf{m} \in \{0,1\}^k$ be a length-k binary message. The message is encoded into a length-n binary codeword $\mathbf{c} = \mathbf{m}\mathbf{G}$, where $\mathbf{G} \in \{0,1\}^{k\times n}$ is the generator matrix of a linear block code, which defines a code $\mathcal{C}$. A parity-check matrix $\mathbf{H} \in \{0,1\}^{r\times n}$ can also define the code $\mathcal{C}$, satisfying $\mathbf{G}\mathbf{H}^\top = \mathbf{O}$ and $\mathbf{c}\mathbf{H}^\top = \mathbf{0}$ over the $\mathbb{F}_2$ finite field.

The codeword is then modulated using binary phase-shift keying (BPSK), yielding $\mathbf{x} = \mathcal{M}(\mathbf{c}) \in \{+1,-1\}^n$, where $\mathcal{M}(\mathbf{c}) = 1 - 2\mathbf{c}$, mapping 0 to $+1$ and 1 to -1. The modulated signal is transmitted over an AWGN channel, resulting in the received signal as $\mathbf{y} = \mathbf{x} + \mathbf{n}$, where $\mathbf{n} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$ denotes independent Gaussian noise. Throughout the paper, we also refer to $\mathbf{y}$ as the channel observation.

The decoding task is to recover the transmitted codeword $\mathbf{x}$ from the observation $\mathbf{y}$.

3.2 Data Transmission as a Diffusion Model

Recent work has proposed to interpret data transmission and decoding through the lens of diffusion probabilistic models (DDPMs), as illustrated in Fig. 1. Under this perspective, the physical transmission process and the decoding process are modeled as the forward and reverse diffusion processes, respectively.

Specifically, let $\mathbf{x}_0 \triangleq \mathbf{x}$ denote the clean transmitted codeword after modulation, as defined above. The DDPM forward process progressively corrupts $\mathbf{x}_0$ by injecting Gaussian noise, yielding a

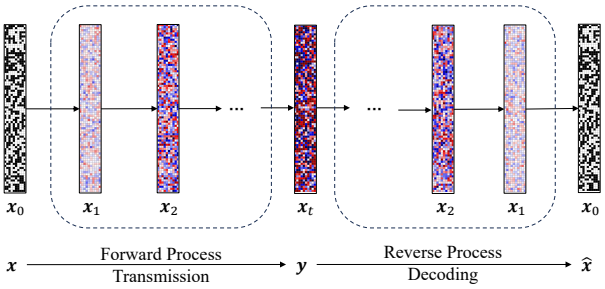

Figure 1: Data transmission viewed as a diffusion model, where the channel observation corresponds to a diffusion state at time t.

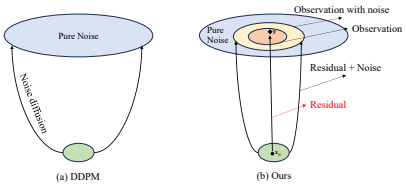

Figure 2: Different diffusion process for decoding. (a) DDPM method: The process disperses the signal into random directions, resulting in a curved and uncertain reverse path. (b) Our residual diffusion method: The process is constrained to a direct linear path connecting the observation $\mathbf{y}$ and the codeword $\mathbf{x}_0$.

sequence of latent variables $\{\mathbf{x}_t\}_{t=1}^T$ by:

$$\mathbf{x}_t = \mathbf{x}_0 + \sqrt{\bar{\beta}_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{1}$$

where $\{\beta_t\}_{t=1}^T$ is a predefined noise schedule and $\bar{\beta}_t \triangleq \sum_{i=1}^t \beta_i$ controls the overall degradation level at diffusion step t.

The forward transition admits the following conditional distribution: $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_{t-1}, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$, where $\tilde{\boldsymbol{\mu}}_t$ and $\tilde{\beta}_t$ are analytically determined coefficients that depend on the noise schedule. In DDPM-based decoding, the reverse process is parameterized by a neural network trained to estimate the injected noise ϵ. Starting from the noisy observation $\mathbf{y}$, which corresponds to a state in the sequence $\{\mathbf{x}_t\}$, the decoder iteratively applies a sequence of denoising updates to obtain progressively refined estimates of the clean codeword $\mathbf{x}_0$ based on the DDPM framework.

4 Method

In this section, we propose the Channel Residual Diffusion Model (ChRes-DM), by formulating the channel decoding problem as a constrained diffusion process anchored by the channel observation (i.e., received signal). We then derive its PF-ODE and the corresponding deterministic iterative solver. Unlike DDPM decoding, we construct a bridge process explicitly connecting the clean codeword $\mathbf{x}_0$ and the received signal $\mathbf{y}$.

4.1 Transmission as Forward Process

Definition 4.1 (ChRes-DM Forward Process)**.** Given a clean codeword $\mathbf{x}_0$ and received signal $\mathbf{y}$, we define the forward process state $\mathbf{x}_t$ at time $t \in [0, T]$ as:

$$\mathbf{x}_t = \mathbf{x}_0 + \eta_t(\mathbf{y} - \mathbf{x}_0) + \sigma_t\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2}$$

where σ_t is the diffusion noise schedule, and $\eta_t \in [0, 1]$ is the residual schedule controlling the injection of the observation residual $\mathbf{y} - \mathbf{x}_0$. This implies the marginal distribution $q_t(\mathbf{x}_t \mid \mathbf{x}_0, \mathbf{y}) = \mathcal{N}\left(\mathbf{x}_t; \boldsymbol{\mu}_t, \sigma_t^2\mathbf{I}\right)$ with mean:

$$\boldsymbol{\mu}_t(\mathbf{x}_0, \mathbf{y}) = \mathbf{x}_0 + \eta_t(\mathbf{y} - \mathbf{x}_0). \tag{3}$$

Boundary conditions are set such that $\eta_0 = 0, \sigma_0 = 0$ (clean codeword) and $\eta_T \approx 1, \sigma_T \approx 1$ (observation with noise).

Accordingly, (3) can be rewritten as:

$$\boldsymbol{\mu}_t(\mathbf{x}_0, \mathbf{y}) = (1 - \eta_t)\mathbf{x}_0 + \eta_t\mathbf{y}. \tag{4}$$

Therefore, the mean $\boldsymbol{\mu}_t(\mathbf{x}_0, \mathbf{y})$ can be interpreted as a convex combination of the clean codeword and the received signal. In simple terms, this implies that our forward process constructs a direct

straight line between the target $\mathbf{x}_0$ and the signal $\mathbf{y}$, as illustrated in Fig. 2. This construction is theoretically grounded in Maximum Likelihood (ML) decoding. For AWGN channels, the optimal ML decoding rule is equivalent to minimizing the Euclidean distance, which is equivalent to minimizing $\mathcal{E}(\mathbf{x}) = d_{\mathrm{E}}^2(\mathbf{y}, \mathbf{x}) = \|\mathbf{y}-\mathbf{x}\|_2^2$ [26]. The gradient of $\mathcal{E}(\mathbf{x})$ with respect to the state is $\nabla_{\mathbf{x}}\mathcal{E} \propto -(\mathbf{y}-\mathbf{x})$, which points directly along the residual vector. By defining our bridge mean $\boldsymbol{\mu}_t$ to interpolate linearly between $\mathbf{y}$ and $\mathbf{x}_0$, the drift term of our diffusion process effectively aligns with this negative gradient direction $(\mathbf{y} - \mathbf{x}_0)$. Consequently, our framework acts as a guided transport process: it naturally follows the Euclidean geodesic (shortest path) driven by the intrinsic geometry of the ML decoding metric, thereby maximizing the efficiency of error correction.

To empirically validate this conceptual comparison, we perform visualizations of the diffusion trajectories using Repetition Codes. We simulate the evolution of distributions for both the DDPM and our proposed framework. The detailed visualization results, including both 2D and 3D trajectory analyses, are provided in Appendix A.

Building on the above definition, we arrive at the following stochastic differential equation (SDE) for the forward process.

Proposition 4.2 (The ChRes-DM Differential Equation)**.** *The continuous-time dynamics of $\mathbf{x}_t$ are governed by the following SDE:*

$$d\mathbf{x}_t = (\mathbf{y} - \mathbf{x}_0)\, \dot{\eta}_t dt + \sqrt{2\sigma_t \dot{\sigma}_t} d\mathbf{w}_t, \tag{5}$$

where $\dot{\eta}_t$ and $\dot{\sigma}_t$ denote time derivatives, and $\mathbf{w}_t$ is the standard Wiener process.

The detailed proof of Proposition 4.2 is presented in Appendix B.

4.2 Decoding

To enable deterministic decoding, we solve the PF-ODE associated with the ChRes-DM SDE, obtaining the following iterative solver, which serves as the update rule for decoding at each iteration.

Theorem 4.3 (Iterative Decoding Solver)**.** *Adopting the first-order hold discretization, the deterministic reverse-time update from t to s ($s < t$) is given by:*

$$\mathbf{x}_s = \mathbf{x}_t - \left[(\eta_t - \eta_s) + (\sigma_t - \sigma_s)\right] \boldsymbol{\epsilon}_\theta^{\mathrm{add}}\left(\mathbf{x}_t\right), \tag{6}$$

where $\boldsymbol{\epsilon}_\theta^{add}\left(\mathbf{x}_t\right) \triangleq \mathbf{y} - \hat{\mathbf{x}}_0\left(\mathbf{x}_t\right)$ denotes the estimated additive residual by the decoder with parameter set θ.

The detailed proof of Theorem 4.3 is presented in Appendix C.

Based on the update rule derived in Theorem 4.3, we formulate the practical decoding procedure. The practical implementation requires a discretization of the timeline into T steps. In particular, as adopted in [5], the maximum number of iterations T is set to the number of parity-check equations. At each iteration, we first estimate the multiplicative noise, obtain the BPSK-modulated clean codeword, compute the corresponding additive residual, and finally update the current signal state. The complete inference workflow is summarized in Algorithm 1 in Appendix E.

4.3 Comparison with Prior Work

To further understand the theoretical advantages of the proposed method, we compare it with the previous work based on DDPM.

Standard DDPM derives the reverse process from a Markovian SDE, explicitly injecting random noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ at each step. While this stochasticity is theoretically essential for generating diverse samples in other tasks, such as image generation, it tends to degrade performance in channel decoding, which is fundamentally a maximum a posteriori (MAP) estimation task targeting a unique ground truth $\mathbf{x}_0$. Prior work applying diffusion to decoding adopts heuristic modifications by *discarding* the noise term $\mathbf{z}$ and performing a line search for an optimal step size factor. However, this modification reduces the rigorous Markovian sampling to a heuristic mean-value update at each step. From a probabilistic perspective, simply iterating the conditional means does not guarantee convergence to the MAP solution of the joint distribution, and lacks a formal justification within the standard variational framework. In contrast, we formalize the deterministic decoding trajectory via the PF-ODE. This yields a theoretically grounded deterministic solver derived from the proposed forward dynamics, rather than heuristically removing the stochastic term in DDPM-based decoding.

4.4 Model Architecture and Training

Backbone Architecture. To capture the parity-check–based dependencies inherent in linear block codes, we adopt the error correction code Transformer (ECCT) [2] within a model-free neural decoding framework. Consider a noisy signal $\mathbf{y}$. Let f_θ be the ECCT network parameterized by θ. The input to f_θ is formed by concatenating the magnitude $|\mathbf{y}|$ and the syndrome obtained from its hard decision. Specifically, the syndrome is defined as

$$s(\mathbf{y}) = \mathrm{bin}(\mathbf{y})\mathbf{H}^\top = 0.5(1 - \mathrm{sign}(\mathbf{y}))\mathbf{H}^\top, \tag{7}$$

where $\mathrm{bin}(\cdot)$ denotes element-wise binarization and the multiplication is over the finite field $\mathbb{F}_2$. To enhance the model's awareness of the noise level, we further incorporate a scalar noise indicator $e_t \triangleq \|s(\mathbf{y})\|_0$, i.e., the number of nonzero entries in the syndrome, which is positively correlated with the channel noise level and the expected number of time steps. It is embedded into a d-dimensional vector at each position, which is multiplied element-wise with the embedding of $[|\mathbf{y}|, s(\mathbf{y})]$.

Given the input $[|\mathbf{y}|, s(\mathbf{y})]$ and e_t, the ECCT outputs an estimate $\hat{\mathbf{z}}$ of the corresponding *multiplicative* noise. A binarized estimate of the clean codeword is then obtained as:

$$\begin{aligned} \hat{\mathbf{x}}_0 &= \mathrm{sign}(\mathbf{y} \odot \hat{\mathbf{z}}) \\ &= \mathrm{sign}(\mathbf{y} \odot f_\theta([|\mathbf{y}|, s(\mathbf{y})], e_t)), \end{aligned} \tag{8}$$

where $\odot$ denotes the element-wise multiplication. Finally, we compute the additive noise estimate $\boldsymbol{\epsilon}_\theta^{\mathrm{add}} = \mathbf{y} - \hat{\mathbf{x}}_0$.

Training. Intuitively, the primary goal of the diffusion solver is to estimate the additive residual vector $\boldsymbol{\epsilon}_\theta^{\mathrm{add}}$. However, directly regressing this in a model-free decoder setting presents significant challenges, often leading to overfitting [1]. Instead of predicting the additive residual directly, we estimate the equivalent *multiplicative* noise $\mathbf{z} \in \{-1, +1\}^n$, which characterizes the element-wise sign relation between the clean codeword $\mathbf{x}_0$ and the current noisy state. This multiplicative-noise representation follows the standard symmetry formulation for binary-input symmetric-output (BISO) channels [13].

We use the Binary Cross-Entropy (BCE) loss to estimate the bit-wise probabilities associated with the multiplicative noise. The optimization objective is formulated as: $\mathcal{L}(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\mathrm{BCE}\left(\sigma(\hat{\mathbf{z}}(\mathbf{x}_t, e_t)), \mathbf{z}_b\right)\right]$, where $\mathbf{z}_b \triangleq \mathrm{bin}(\mathbf{z}) \in \{0,1\}^n$ denotes the binary labels of the ground-truth multiplicative noise and $\sigma(\hat{\mathbf{z}}(\mathbf{x}_t, e_t))$ denotes the network's corresponding soft probability estimates obtained through the sigmoid function for the sampled time $t \sim \mathcal{U}(0, \ldots, T)$. The maximum number of time steps T is determined by the length of the syndrome.

Once $\hat{\mathbf{z}}$ is obtained, it can be converted back to the estimated codeword via $\hat{\mathbf{x}}_0$ (8) and the corresponding additive residual $\boldsymbol{\epsilon}_\theta^{\mathrm{add}}$.

We use the Adam optimizer with a mini-batch size of 128. Each epoch consists of 1000 mini-batches, and the model is trained for 2000 epochs. The learning rate follows a cosine decay schedule, annealing from 5×10^{-4} to 5×10^{-6}. For the schedule configuration, we adopt a constant variance setting ($\beta_t = 0.01$). To align with the previous DDPM decoding, we accumulate the noise levels as $\bar{\beta}_t = \sum_{i=1}^{t} \beta_i$. The model parameters are then instantiated based on this cumulative schedule, specifically setting η_t and σ_t as $\sqrt{\bar{\beta}_t}$.

5 Experiments

To evaluate the proposed ChRes-DM framework, we conducted experiments on various linear block codes, including Polar codes, BCH codes, and CCSDS LDPC codes from a public dataset [27]. Unless otherwise specified, experiments assume BPSK modulation over an AWGN channel. We benchmark our method against the ECCT decoder [2] and the diffusion-based decoder (baseline DDPM) [5]. For the main diffusion-based comparisons, ChRes-DM reuses the pretrained ECCT backbone parameters from the DDPM baseline, with modifications confined to the diffusion formulation and reverse-time decoding updates. We also present the BP-based decoding methods, including the traditional standard BP algorithm [28] and an autoregressive hyper-network BP method (ARBP) [29]. Our neural network configurations are denoted as (N, d), where N is the number of attention layers and d is the embedding dimension. We evaluate two configurations: a lightweight setting $(N = 2, d = 32)$

Table 1: Comparison of our method with literature baselines in terms of the negative natural logarithm of BER, i.e., $-\ln(\text{BER})$. BP-based results are obtained after 5 iterations in the first row and after 50 iterations in the second row (i.e., 10- and 100-layer NNs). For other decoders, the first and second rows are obtained by using the decoder with parameter $N = 2, d = 32$, and with parameter $N = 6, d = 128$, respectively. Higher values indicate better error-correction performance.

Decoder	BP			ARBP			ECCT			DDPM			Ours ($\Delta t = 1$)			Ours ($\Delta t = 4$)		
E_b/N_0 (dB)	4	5	6	4	5	6	4	5	6	4	5	6	4	5	6	4	5	6
Polar(64,32)	3.52	4.04	4.48	4.77	6.30	8.19	4.27	5.44	6.95	6.14	8.28	10.86	6.20	8.32	11.12	6.19	8.29	11.21
	4.26	5.38	6.50	5.57	7.43	9.82	6.99	9.44	12.32	7.14	9.52	12.27	7.20	9.62	12.75	7.19	9.65	12.90
Polar(64,48)	4.15	4.68	5.31	5.25	6.96	9.00	4.92	6.46	8.41	5.59	7.54	9.87	5.68	7.68	10.07	5.62	7.65	10.15
	4.74	5.94	7.42	5.41	7.19	9.30	6.36	8.46	11.09	6.04	8.04	10.28	6.11	8.21	10.79	6.08	8.21	10.83
Polar(128,64)	3.38	3.80	4.15	4.02	5.48	7.55	3.51	4.52	5.93	5.64	8.11	10.94	5.65	8.10	10.94	5.61	8.08	10.99
	4.10	5.11	6.15	4.84	6.78	9.30	5.92	8.64	12.18	8.37	12.09	15.92	8.36	12.06	15.99	8.37	12.10	15.90
Polar(128,86)	3.80	4.19	4.62	4.81	6.57	9.04	4.30	5.58	7.34	5.75	7.96	10.76	5.74	7.94	10.85	5.71	7.91	10.73
	4.49	5.65	6.97	5.39	7.37	10.13	6.31	9.01	12.45	7.86	10.99	15.10	7.85	11.03	15.17	7.82	10.93	15.35
Polar(128,96)	3.99	4.41	4.78	4.92	6.73	9.30	4.56	5.98	7.93	5.71	8.05	10.93	5.74	8.06	10.91	5.69	8.01	10.90
	4.61	5.79	7.08	5.27	7.44	10.20	6.31	9.12	12.47	7.31	10.22	13.13	7.31	10.26	13.14	7.27	10.32	13.19
BCH(63,36)	3.72	4.65	5.66	4.33	5.94	8.21	3.79	4.87	6.35	4.71	6.38	8.69	4.74	6.43	8.80	4.68	6.34	8.61
	4.03	5.42	7.26	4.57	6.39	8.92	4.86	6.65	9.10	5.22	7.09	9.60	5.26	7.16	9.68	5.24	7.13	9.76
BCH(63,45)	4.08	4.96	6.07	4.80	6.43	8.69	4.47	5.88	7.81	5.20	7.17	9.79	5.27	7.32	10.18	5.21	7.28	10.09
	4.36	5.55	7.26	4.97	6.90	9.41	5.60	7.79	10.93	5.75	7.99	10.60	5.82	8.18	11.44	5.80	8.19	11.61
CCSDS(128,64)	6.55	9.65	13.78	7.25	10.99	16.36	4.35	6.01	8.30	5.98	8.93	12.40	5.98	8.95	12.42	5.91	8.80	12.33
	-	-	-	-	-	-	6.88	10.90	15.90	8.01	12.60	17.28	8.00	12.61	17.42	7.94	12.55	17.42

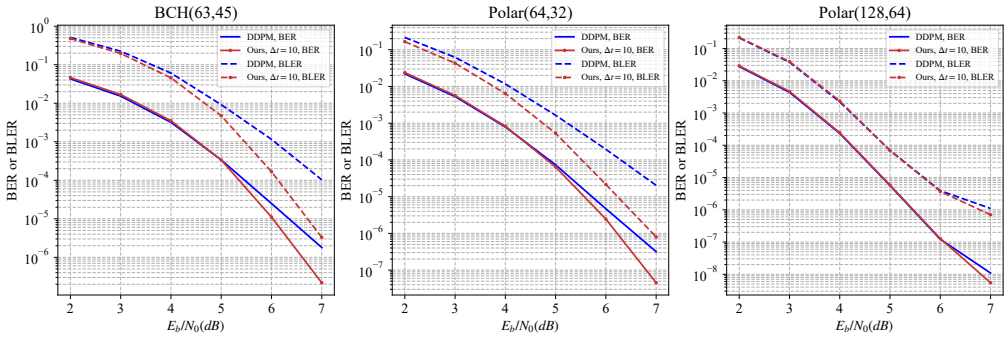

Figure 3: BER and BLER results for (a) BCH(63,45) code, (b) Polar(64,32) code and (c) Polar(128, 64) code. The model parameters are all set as $N = 6$, $d = 128$.

and a larger setting $(N = 6, d = 128)$. The decoding performance is measured by the bit error rate (BER) and block error rate (BLER). Furthermore, we conducted an ablation study, analyzing the computational cost to demonstrate the efficiency gains of our method.

5.1 Decoding Performance

We first examine the decoding performance at specific SNR levels, chosen to align with the experimental settings in prior studies. Table 1 presents the negative natural logarithm of BER, i.e., $-\ln(\text{BER})$, at specific SNRs ($E_b/N_0 = 4, 5, 6$ dB). A higher value indicates better performance (lower BER). We compare our method using different step sizes ($t - s = \Delta t \in \{1, 4\}$) against the baselines. The results demonstrate that our method consistently achieves the best or second-best performance across most code configurations. Even with step-skipping (e.g., jumping 4 steps), our method maintains a competitive BER, showcasing the robustness of the deterministic ODE trajectory.

To visually evaluate the performance, Fig. 3 plots the BER and BLER curves for BCH(63,45), Polar(64,32), and Polar(128,64) codes, comparing our accelerated ChRes-DM (with a skipping

Table 2: Results of different inference strategies for our method and comparison with literature baselines in terms of the negative natural logarithm of block error rate (BLER) and bit error rate (BER), i.e., $-\ln(\text{BLER})$ and $-\ln(\text{BER})$, respectively. The "LS" denotes the line search. The model parameters are set as $N = 6$, $d = 128$. We use $\Delta t = 1$ for the ChRes-DM. For each method, the first line shows $-\ln(\text{BLER})$, and the second line shows $-\ln(\text{BER})$. Higher is better.

Method	DDPM w/o LS					DDPM w/ LS					Ours w/o LS					Ours w/ LS				
E_b/N_0 (dB)	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6
Polar(64,32)	1.53	2.78	4.45	6.41	8.54	1.84	3.18	5.11	7.58	10.77	1.70	3.00	4.78	6.81	9.30	1.83	3.17	5.09	7.58	10.75
	3.82	5.24	7.14	9.52	12.27	3.69	5.13	7.10	9.64	12.94	3.82	5.26	7.20	9.62	12.75	3.68	5.12	7.09	9.64	12.90
Polar(64,48)	0.74	1.60	2.98	4.71	6.63	0.96	1.95	3.48	5.50	8.09	0.85	1.80	3.27	5.20	7.55	0.96	1.95	3.48	5.51	8.08
	3.35	4.44	6.04	8.04	10.28	3.23	4.36	6.03	8.17	10.85	3.36	4.47	6.11	8.21	10.79	3.22	4.36	6.03	8.19	10.86
BCH(63,36)	0.68	1.40	2.58	4.21	6.38	0.84	1.64	2.90	4.70	7.21	0.74	1.48	2.70	4.45	6.71	0.85	1.63	2.90	4.70	7.17
	2.94	3.85	5.22	7.09	9.60	2.90	3.81	5.19	7.06	9.69	2.96	3.87	5.26	7.16	9.68	2.89	3.80	5.18	7.08	9.66
BCH(63,45)	0.67	1.50	2.80	4.69	6.76	0.86	1.82	3.35	5.67	9.11	0.75	1.64	3.07	5.20	7.87	0.86	1.82	3.36	5.70	9.11
	3.15	4.18	5.75	7.99	10.60	3.02	4.06	5.65	8.04	11.49	3.16	4.21	5.82	8.18	11.44	3.02	4.05	5.66	8.09	11.46

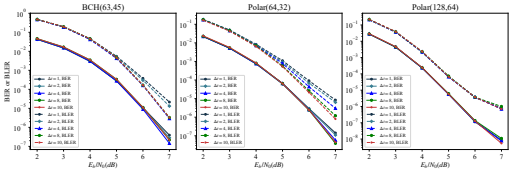

Figure 4: BER and BLER results of different skipping-step Δts for (a) BCH(63,45) code, (b) Polar(64,32) code, and (c) Polar(128,64) code. The model parameters are set as $N = 6$, $d = 128$.

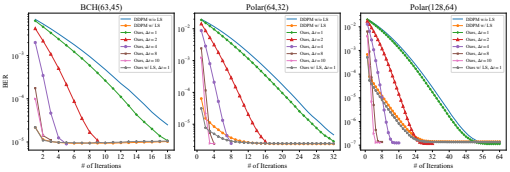

Figure 5: BER versus number of iterations at $E_b/N_0 = 6$ dB for (a) BCH(63,45) code, (b) Polar(64,32) code and (c) Polar(128,64) code. The model parameters are set as $N = 6$, $d = 128$. The BER is shown in logarithmic scale for clarity.

step of $\Delta t = 10$) against the standard step-by-step DDPM baseline (equivalent to $\Delta t = 1$). In the low-to-medium SNR regime ($2 \sim 5$ dB), the performance of the two methods nearly overlaps. However, in the higher SNR region ($5 \sim 7$ dB), our method demonstrates visible coding gains in both BER and BLER.

These results demonstrate improved error rates in the high-SNR regime [30] while substantially reducing the required diffusion iterations. The statistical consistency of the high-SNR gains is further evaluated in Appendix F.

5.2 Ablation Study and Analysis

5.2.1 Impact of Δt

To evaluate the sensitivity of ChRes-DM to the step-skipping interval, we compare the BER and BLER performance across five configurations: $\Delta t \in \{1, 2, 4, 8, 10\}$. While the curves overlap in the low SNR regime, significant differentiation emerges at high SNRs (> 5 dB). Interestingly, setting $\Delta t > 1$ not only accelerates inference but also actively reduces the error probability, with a particularly visible gain in BLER. This suggests that a larger skipping interval effectively prevents the network from over-refining (and potentially corrupting) the relatively clean high-SNR signals. We observe that the optimal Δt is code-dependent: for instance, $\Delta t = 4$ yields the best results for BCH(63,45), while $\Delta t = 10$ is optimal for the longer Polar(128,64) code. Consequently, an appropriately selected Δt, tailored to the specific code structure, can simultaneously unlock better high-SNR performance and lower latency.

5.2.2 Impact of Line Search

A heuristic method is the Line Search (LS) strategy employed in the baseline DDPM, which selects the best scaling factor from a candidate set (e.g., $\{1, \ldots, 20\}$ in the literature) at each step based on the number of non-zero elements in the syndrome (lower is better). We evaluate the compatibility of

Table 3: Comparison of computational efficiency and decoding performance on Polar(64,32) and BCH(63,45) codes. We report the average number of iterations (T_{avg}), the relative speedup compared to the baseline DDPM without Line Search (w/o LS). "Cand." denotes the number of candidate evaluations per diffusion step. Cand.=1 indicates decoding without Line Search (w/o LS), while Cand.$>$1 indicates the use of Line Search. **Bold** and underlined values represent the best and second-best results in terms of efficiency and accuracy, respectively.

Code	Method	E_b/N_0 / Cand.	Avg. Iterations ($T_{\text{avg}} \downarrow$)					Speedup (vs. Baseline)					Performance ($-\ln(\text{BER}) \uparrow$)				
			2	3	4	5	6	2	3	4	5	6	2	3	4	5	6
Polar(64,32)	DDPM	1	22.89	18.20	13.74	9.78	6.44	1.00x	1.00x	1.00x	1.00x	1.00x	3.82	5.24	7.14	9.52	12.27
	DDPM	20	3.15	2.42	2.10	1.94	1.78	7.27x	7.51x	6.56x	5.04x	3.62x	3.69	5.13	7.10	9.64	**12.94**
	Ours ($\Delta t = 1$)	1	20.55	15.97	11.86	8.35	5.48	1.11x	1.14x	1.16x	1.17x	1.18x	3.82	5.26	7.20	9.62	12.75
	Ours ($\Delta t = 1$)	20	**2.98**	**2.33**	**2.06**	**1.93**	**1.78**	**7.67x**	**7.81x**	**6.68x**	**5.07x**	**3.62x**	3.68	5.12	7.09	9.64	12.90
	Ours ($\Delta t = 2$)	1	10.60	8.40	6.41	4.69	3.29	2.16x	2.17x	2.14x	2.08x	1.96x	**3.82**	5.26	**7.20**	**9.71**	12.75
	Ours ($\Delta t = 4$)	1	5.84	4.82	3.88	3.06	2.36	3.92x	3.77x	3.54x	3.20x	2.73x	3.81	**5.26**	7.19	9.65	12.90
	Ours ($\Delta t = 8$)	1	3.40	3.00	2.60	2.24	1.91	6.73x	6.07x	5.29x	4.36x	3.37x	3.78	5.21	7.12	9.65	12.93
BCH(63,45)	DDPM	1	14.53	11.27	7.81	4.96	2.95	1.00x	1.00x	1.00x	1.00x	1.00x	3.15	4.18	5.75	7.99	10.60
	DDPM	20	7.63	4.24	2.37	1.71	1.42	1.91x	2.66x	3.30x	2.90x	2.08x	3.02	4.06	5.65	8.04	11.49
	Ours ($\Delta t = 1$)	1	13.68	10.21	6.89	4.35	2.63	1.06x	1.10x	1.13x	1.14x	1.12x	**3.16**	**4.21**	**5.82**	8.18	11.44
	Ours ($\Delta t = 1$)	20	7.54	4.20	2.36	1.71	**1.42**	1.93x	2.68x	3.32x	2.90x	**2.08x**	3.02	4.05	5.66	8.09	11.46
	Ours ($\Delta t = 2$)	1	7.11	5.54	3.98	2.74	1.87	2.04x	2.04x	1.97x	1.81x	1.57x	3.15	4.20	5.82	8.17	11.42
	Ours ($\Delta t = 4$)	1	4.18	3.43	2.66	2.03	1.55	3.48x	3.28x	2.94x	2.45x	1.90x	3.14	4.18	5.80	**8.19**	**11.61**
	Ours ($\Delta t = 8$)	1	**2.65**	**2.33**	**1.99**	**1.70**	1.43	**5.48x**	**4.85x**	**3.93x**	**2.92x**	2.07x	3.07	4.08	5.65	8.00	11.41

this heuristic with our framework and compare it against our native ODE-based acceleration (step skipping).

As shown in Table 2, the final BERs of both methods with LS are comparable, confirming that our framework supports heuristic acceleration. Furthermore, we present the BER versus the diffusion iteration curves of decoding with LS in Fig. 5. When equipped with LS, our method achieves rapid convergence similar to the baseline with LS. Although LS reduces the number of iterations, it incurs a significant computational overhead. We next discuss the computational cost in the following part.

5.2.3 Computational Cost

We conduct an analysis focusing on the average number of iterations (T_{avg}) as the primary metric for latency, since each iteration corresponds to a single neural network forward pass. Table 3 presents the comparison between the baseline DDPM and our proposed ChRes-DM under various configurations. The following observations can be made:

Acceleration via Step-Skipping. As shown in Table 3, even without step-skipping ($\Delta t = 1$) and without relying on the LS heuristic (Cand. = 1), ChRes-DM consistently requires fewer iterations than the baseline DDPM across all SNRs (e.g., achieving a $1.18\times$ speedup on Polar(64,32) and $1.12\times$ on BCH(63,45) at 6 dB), while delivering better decoding accuracy. Furthermore, the deterministic and guided nature of our framework allows for aggressive acceleration through principled ODE step-skipping, effectively bypassing the need for heuristic search strategies. By increasing the skipping step Δt, T_{avg} decreases significantly. For instance, using step-skipping without heuristic candidates ($\Delta t = 8$, Cand. = 1) achieves a speedup of up to $\mathbf{6.73}\times$ on Polar(64,32) and $\mathbf{5.48}\times$ on BCH(63,45) relative to the standard baseline.

Accuracy-Efficiency Trade-off. Our step-skipping strategy achieves a competitive latency profile compared to the heuristically-driven DDPM w/ LS (Cand. = 20), while yielding better performance. For example, on BCH(63,45) at 6 dB, our method ($\Delta t = 4$, Cand. = 1) requires merely 1.55 iterations on average while achieving a best-in-class $-\ln(\text{BER})$ of $\mathbf{11.61}$, outperforming the baseline w/ LS's 11.49. Furthermore, our framework remains highly flexible; when equipped with LS (Cand. = 20), ChRes-DM achieves speedups of up to $\mathbf{7.81}\times$. This demonstrates that our framework provides a robust and highly efficient alternative to standard diffusion decoding.

Additional discussions and analyses on other aspects can be found in Appendices G–L.

6 Conclusion

In this paper, we propose ChRes-DM, which achieves a natural alignment between diffusion inference and the restorative objective of decoding by explicitly incorporating channel observations. By constructing a geometric bridge, we derive a deterministic PF-ODE algorithm that justifies the decoding trajectory and enables practical acceleration via flexible step-skipping. Extensive experiments demonstrate significant iteration reduction, although the trade-offs in decoding performance and inference efficiency relative to specialized classical benchmarks in certain regimes remain current limitations. Our results suggest that leveraging geometric inductive biases can be effective for adapting generative models to communication systems. Future research will explore higher-order modulations, neural architectures and training strategies tailored to diffusion-based decoding for more accurate noise estimation, adaptive step-skipping conditioned on the code structure and decoding state, and extensions to more complex channel distributions.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China under grant 62301189, 62576122, 62571298, and in part by the Guangdong Basic and Applied Basic Research Foundation under grant 2026A1515011139. This work was also supported in part by the Major Key Project of Pengcheng Laboratory under grant No. PCL2026A01 and grant No. PCL2025A05, and in part by the Basic and Frontier Research Project of Pengcheng Laboratory under grant No. 2025QYB004.

References

[1] Amir Bennatan, Yoni Choukroun, and Pavel Kisilev. Deep learning for decoding of linear codes-a syndrome-based approach. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1595–1599. IEEE, 2018.

[2] Yoni Choukroun and Lior Wolf. Error correction code transformer. *Advances in Neural Information Processing Systems*, 35:38695–38705, 2022.

[3] Seong-Joon Park, Hee-Youl Kwak, Sang-Hyo Kim, Sunghwan Kim, Yongjune Kim, and Jong-Seon No. Multiple-masks error correction code transformer for short block codes. *IEEE Journal on Selected Areas in Communications*, 2025.

[4] Seong-Joon Park, Hee-Youl Kwak, Sang-Hyo Kim, Yongjune Kim, and Jong-Seon No. CrossMPT: Cross-attention message-passing transformer for error correcting codes. In *The Thirteenth International Conference on Learning Representations*, 2025.

[5] Yoni Choukroun and Lior Wolf. Denoising diffusion error correction codes. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda*, 2023.

[6] Chin Wa Ken Lau, Xiang Shi, Ziyan Zheng, Haiwen Cao, and Nian Guo. Interplay between belief propagation and transformer: Differential-attention message passing transformer. In *2025 IEEE International Symposium on Information Theory (ISIT)*, pages 1–6. IEEE, 2025.

[7] Eliya Nachmani, Elad Marciano, Loren Lugosch, Warren J Gross, David Burshtein, and Yair Be'ery. Deep learning methods for improved decoding of linear codes. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):119–131, 2018.

[8] Loren Lugosch and Warren J Gross. Neural offset min-sum decoding. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1361–1365. IEEE, 2017.

[9] Andreas Buchberger, Christian Häger, Henry D Pfister, Laurent Schmalen, and Alexandre Graell i Amat. Pruning and quantizing neural belief propagation decoders. *IEEE Journal on Selected Areas in Communications*, 39(7):1957–1966, 2020.

[10] Jincheng Dai, Kailin Tan, Zhongwei Si, Kai Niu, Mingzhe Chen, H Vincent Poor, and Shuguang Cui. Learning to decode protograph LDPC codes. *IEEE Journal on Selected Areas in Communications*, 39(7):1983–1999, 2021.

[11] Qinshan Zhang, Bin Chen, Tianqu Zhuang, Yong Jiang, and Shu-Tao Xia. Revolving NBP-like decoders for cyclic and quasi-cyclic LDPC codes. *IEEE Transactions on Communications*, 74: 598–611, 2026.

[12] Haoyu Lei, Chin Wa Lau, Kaiwen Zhou, Nian Guo, and Farzan Farnia. Consistency flow model achieves one-step denoising error correction codes. *arXiv preprint arXiv:2512.01389*, 2025.

[13] Thomas J Richardson and Rüdiger L Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, 2001.

[14] Eliya Nachmani, Yair Be'ery, and David Burshtein. Learning to decode linear codes using deep learning. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 341–346. IEEE, 2016.

[15] Xiangyu Chen and Min Ye. Cyclically equivariant neural decoders for cyclic codes. In *International Conference on Machine Learning (ICML)*, pages 1771–1780. PMLR, 2021.

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

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

[18] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[19] Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon. The principles of diffusion models. *arXiv preprint arXiv:2510.21890*, 2025.

[20] Mauricio Delbracio and Peyman Milanfar. Inversion by direct iteration: An alternative to denoising diffusion for image restoration. *Transactions on Machine Learning Research*, 2023.

[21] Guan-Horng Liu, Arash Vahdat, De-An Huang, Evangelos Theodorou, Weili Nie, and Anima Anandkumar. I^2sb: Image-to-image schrödinger bridge. In *International Conference on Machine Learning*, pages 22042–22062. PMLR, 2023.

[22] Zongsheng Yue, Jianyi Wang, and Chen Change Loy. ResShift: Efficient diffusion model for image super-resolution by residual shifting. *Advances in Neural Information Processing Systems*, 36:13294–13307, 2023.

[23] Jiawei Liu, Qiang Wang, Huijie Fan, Yinong Wang, Yandong Tang, and Liangqiong Qu. Residual denoising diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2773–2783, 2024.

[24] Yichong Xia, Yimin Zhou, Jinpeng Wang, and Bin Chen. Towards efficient low-rate image compression with frequency-aware diffusion prior refinement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 10942–10950, 2026.

[25] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P Srinivasan, Dor Verbin, Jonathan T Barron, Ben Poole, et al. ReconFusion: 3D reconstruction with diffusion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21551–21561, 2024.

[26] William Ryan and Shu Lin. *Channel codes: classical and modern*. Cambridge University Press, 2009.

[27] Michael Helmling, Stefan Scholl, Florian Gensheimer, Tobias Dietz, Kira Kraft, Oliver Griebel, Stefan Ruzika, and Norbert Wehn. Database of Channel Codes and ML Simulation Results. `www.rptu.de/channel-codes`, 2025.

[28] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[29] Eliya Nachmani and Lior Wolf. Autoregressive belief propagation for decoding block codes. *arXiv preprint arXiv:2103.11780*, 2021.

[30] Tom Richardson. Error floors of LDPC codes. In *Proceedings of the annual Allerton conference on communication control and computing*, volume 41, pages 1426–1435. The University; 1998, 2003.

[31] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[32] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[33] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35: 26565–26577, 2022.

A Visualization of Diffusion Dynamics

To provide an intuitive understanding of the proposed ChRes-DM, we visualize the forward diffusion process using Repetition Codes with BPSK modulation:

- Generator matrix $\mathbf{G} = [1, 1]$, mapping a bit 0 to the codeword $\mathbf{x}_0 = (+1, +1)$.
- Generator matrix $\mathbf{G} = [1, 1, 1]$, mapping a bit 0 to the codeword $\mathbf{x}_0 = (+1, +1, +1)$.
- In both cases, we simulate the evolution of 500 particles. The "Mean Path" plotted in the figures represents the empirical expectation of the particle states $\mathbb{E}[\mathbf{x}_t]$ averaged over all realized trajectories at each time step t.

Figure 6 presents the comparison between the DDPM diffusion and our proposed ChRes-DM for channel coding. For DDPM, the diffusion process is centered at the clean codeword $\mathbf{x}_0$. The noise variance explodes blindly in all directions. Consequently, the Mean Path remains static at $\mathbf{x}_0$. This indicates that the diffusion process ignores the geometric location of the received observation $\mathbf{y}$, relying solely on random exploration to cover the signal space. In contrast, our method introduces a deterministic drift term. The Mean Path clearly traces a linear trajectory connecting the clean codeword $\mathbf{x}_0$ to the noisy observation $\mathbf{y}$. The particle distribution, while expanding in variance, is actively transported towards $\mathbf{y}$, ensuring that the terminal distribution is concentrated in the vicinity of the observation.

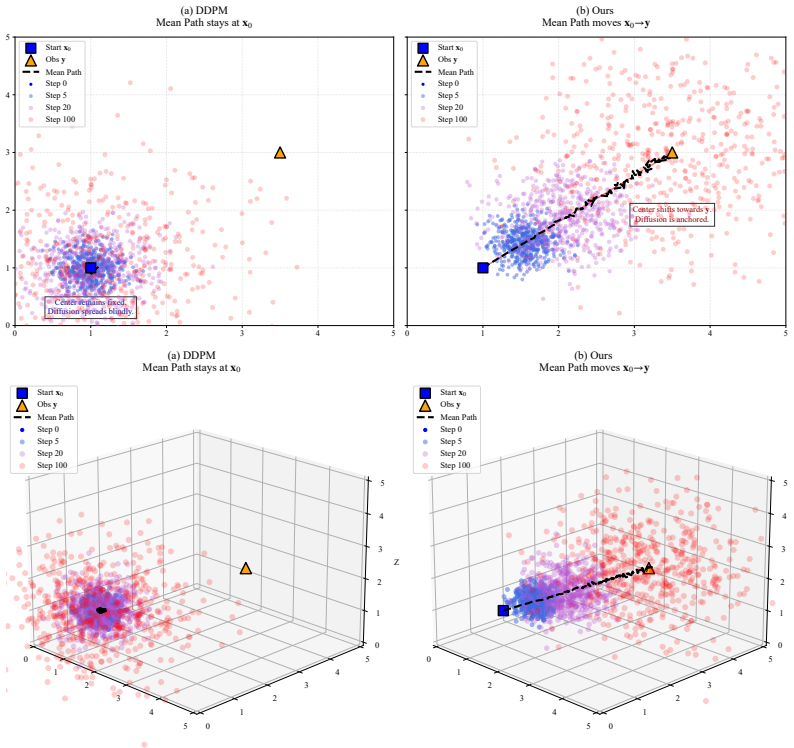

Figure 6: Visualization of forward diffusion trajectories for Repetition Codes with BPSK modulation. The blue square (■) denotes the clean codeword $\mathbf{x}_0 = (+1, \ldots, +1)$, and the orange triangle (▲) denotes the noisy observation $\mathbf{y}$. Scattered points illustrate the probabilistic evolution of particles over time steps. (a) DDPM: The mean path (black dashed line) stays fixed at $\mathbf{x}_0$, leading to blind variance expansion that ignores the observation. (b) Ours: The mean path exhibits a directed drift from $\mathbf{x}_0$ to $\mathbf{y}$. The diffusion is anchored to the observation, creating a geometric bridge that facilitates efficient reverse decoding. The top row shows the 2D case ($\mathbf{G} = [1, 1]$), and the bottom row shows the 3D case ($\mathbf{G} = [1, 1, 1]$).

B Proof of Proposition 4.2

Proof. The goal is to find a continuous-time Stochastic Differential Equation (SDE) of the form:

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + g(t)d\mathbf{w}_t, \tag{9}$$

such that its marginal distribution matches the state definition in Definition 4.1. From the definition, the marginal distribution of $\mathbf{x}_t$ conditioned on the initial codeword $\mathbf{x}_0$ and observation $\mathbf{y}$ is Gaussian: $q(\mathbf{x}_t|\mathbf{x}_0, \mathbf{y}) = \mathcal{N}\left(\mathbf{x}_t; \boldsymbol{\mu}_t, \sigma_t^2\mathbf{I}\right)$, where the mean vector is $\boldsymbol{\mu}_t = \mathbf{x}_0 + \eta_t(\mathbf{y} - \mathbf{x}_0)$.

According to the Fokker-Planck equation for linear SDEs, the time evolution of the mean $\boldsymbol{\mu}_t$ and the covariance $\boldsymbol{\Sigma}_t = \sigma_t^2\mathbf{I}$ are governed by the drift and diffusion coefficients, respectively.

1. $\mathbf{f}(\mathbf{x}_t, t)$: The rate of change of the mean is driven purely by the drift term. Differentiating $\boldsymbol{\mu}_t$ with respect to time t, we obtain:

$$\mathbf{f}(\mathbf{x}_t, t) = \dot{\boldsymbol{\mu}}_t = \frac{d}{dt}\left[\mathbf{x}_0 + \eta_t(\mathbf{y} - \mathbf{x}_0)\right] = (\mathbf{y} - \mathbf{x}_0)\dot{\eta}_t. \tag{10}$$

Since $\mathbf{x}_0$ and $\mathbf{y}$ are time-invariant constants for a specific transmission, the drift term is independent of the current state $\mathbf{x}_t$.

2. $g(t)$: Since the derived drift term depends solely on time and is independent of the state $\mathbf{x}_t$, it does not affect the evolution of the covariance. The rate of change of the covariance matrix is determined only by the diffusion coefficient, given by $\frac{d\boldsymbol{\Sigma}_t}{dt} = g(t)^2\mathbf{I}$. For our marginal distribution, the covariance is $\sigma_t^2\mathbf{I}$. Thus, matching the variance derivative yields:

$$g(t)^2 = \frac{d(\sigma_t^2)}{dt} = 2\sigma_t\dot{\sigma}_t. \tag{11}$$

Taking the square root, the diffusion coefficient must be: $g(t) = \sqrt{2\sigma_t\dot{\sigma}_t}$.

Substituting the derived drift and diffusion coefficients back into the SDE (9), we arrive at the continuous-time dynamics governing the forward process and complete the proof:

$$d\mathbf{x}_t = (\mathbf{y} - \mathbf{x}_0)\dot{\eta}_t dt + \sqrt{2\sigma_t\dot{\sigma}_t}d\mathbf{w}_t. \tag{12}$$

□

C Proof of Theorem 4.3

Proof. We start with the PF-ODE form [18] for the process defined in Proposition 4.2:

$$\begin{aligned} d\mathbf{x}_t &= \left[\mathbf{f}(\mathbf{x}_t, t) - \frac{1}{2}g(t)^2\nabla_{\mathbf{x}_t}\log q_t\left(\mathbf{x}_t\right)\right]dt \\ &= \left[\dot{\boldsymbol{\mu}}_t - \frac{1}{2}(2\sigma_t\dot{\sigma}_t)\nabla_{\mathbf{x}_t}\log q_t\left(\mathbf{x}_t\right)\right]dt. \end{aligned} \tag{13}$$

Using Tweedie's formula [31], the exact score can be expressed through the posterior statistics of the latent perturbation. As analyzed in Appendix D, we approximate this score using the observation-anchored residual:

$$\nabla_{\mathbf{x}_t}\log q_t\left(\mathbf{x}_t\right) \approx -\frac{\boldsymbol{\epsilon}_\theta^{\text{add}}\left(\mathbf{x}_t\right)}{\sigma_t}. \tag{14}$$

From (10), the mean derivative is obtained as $\dot{\boldsymbol{\mu}}_t = (\mathbf{y} - \mathbf{x}_0)\,\dot{\eta}_t$. Replacing the true residual with the network estimate $\boldsymbol{\epsilon}_\theta^{\text{add}}$, we have $\dot{\boldsymbol{\mu}}_t \approx \boldsymbol{\epsilon}_\theta^{\text{add}}\dot{\eta}_t$. Also, from (11) we have $g(t) = \sqrt{2\sigma_t\dot{\sigma}_t}$. Substituting these two terms and the score approximation (14), we have:

$$d\mathbf{x}_t = \left(\dot{\eta}_t\boldsymbol{\epsilon}_\theta^{\text{add}} + \dot{\sigma}_t\boldsymbol{\epsilon}_\theta^{\text{add}}\right)dt = \left(\dot{\eta}_t + \dot{\sigma}_t\right)\boldsymbol{\epsilon}_\theta^{\text{add}}dt. \tag{15}$$

Integrating from t to s and applying the first-order hold assumption, we have

$$\begin{aligned} \mathbf{x}_s - \mathbf{x}_t &= \int_t^s \left(\dot{\eta}_\tau + \dot{\sigma}_\tau\right)\boldsymbol{\epsilon}_\theta^{\text{add}}\left(\mathbf{x}_\tau\right)d\tau \\ &\approx \boldsymbol{\epsilon}_\theta^{\text{add}}\left(\mathbf{x}_t\right)\left[\int_t^s d\eta_\tau + \int_t^s d\sigma_\tau\right] \\ &= \left[\left(\eta_s - \eta_t\right) + \left(\sigma_s - \sigma_t\right)\right]\boldsymbol{\epsilon}_\theta^{\text{add}}\left(\mathbf{x}_t\right). \end{aligned} \tag{16}$$

By rearranging (16), the iterative update process as (6) can be obtained. This completes the proof. □

D Score Approximation and Error Analysis for Equation (14)

Throughout this section, the received observation $\mathbf{y}$ is fixed. Conditioning on $\mathbf{y}$ is therefore implicit and omitted from the conditional-expectation notation for brevity.

From Definition 4.1,

$$\mathbf{x}_t = \boldsymbol{\mu}_t + \sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\mu}_t = (1-\eta_t)\mathbf{x}_0 + \eta_t \mathbf{y}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{17}$$

Conditioned on $\mathbf{x}_0$ and $\mathbf{y}$, $\mathbf{x}_t$ is Gaussian, while the marginal $q_t(\mathbf{x}_t)$ is obtained by averaging over the codeword posterior. Define

$$\bar{\mathbf{x}}_{0,t} \triangleq \mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_t], \qquad \bar{\boldsymbol{\epsilon}}_t \triangleq \mathbb{E}[\boldsymbol{\epsilon} \mid \mathbf{x}_t] = \frac{\mathbf{x}_t - \mathbb{E}[\boldsymbol{\mu}_t \mid \mathbf{x}_t]}{\sigma_t}. \tag{18}$$

Using the standard Gaussian-mixture score identity (equivalently, Tweedie's formula [31]), the exact conditional score is

$$\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) = -\frac{\mathbf{x}_t - \mathbb{E}[\boldsymbol{\mu}_t \mid \mathbf{x}_t]}{\sigma_t^2} = -\frac{\bar{\boldsymbol{\epsilon}}_t}{\sigma_t}. \tag{19}$$

The practical decoder does not have access to $\bar{\boldsymbol{\epsilon}}_t$. Instead, it forms a clean-codeword estimate $\hat{\mathbf{x}}_{0,t}$ and uses

$$\boldsymbol{\epsilon}_\theta^{\mathrm{add}}(\mathbf{x}_t) = \mathbf{x}_t - \hat{\mathbf{x}}_{0,t} \tag{20}$$

as the residual direction. Note that Algorithm 1 reuses $\mathbf{y}$ as an iteratively updated working variable initialized by the channel observation. Within the reverse loop, this working variable corresponds to the current state $\mathbf{x}_t$. Equation (14) should therefore be interpreted as a score approximation rather than an exact equality.

To characterize its error, define

$$\kappa_t \triangleq \left\| (\mathbf{x}_t - \bar{\mathbf{x}}_{0,t}) - \bar{\boldsymbol{\epsilon}}_t \right\|, \tag{21}$$

which measures the structural residual-to-score mismatch, and

$$\xi_t \triangleq \left\| \hat{\mathbf{x}}_{0,t} - \bar{\mathbf{x}}_{0,t} \right\|, \tag{22}$$

which measures the codeword-estimation error of the neural decoder. Combining (19) with the practical residual, we obtain

$$\begin{aligned} &\left\| \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) - \left(-\frac{\boldsymbol{\epsilon}_\theta^{\mathrm{add}}(\mathbf{x}_t)}{\sigma_t} \right) \right\| \\ &= \frac{1}{\sigma_t} \left\| (\mathbf{x}_t - \hat{\mathbf{x}}_{0,t}) - \bar{\boldsymbol{\epsilon}}_t \right\| \\ &= \frac{1}{\sigma_t} \left\| [(\mathbf{x}_t - \bar{\mathbf{x}}_{0,t}) - \bar{\boldsymbol{\epsilon}}_t] - [\hat{\mathbf{x}}_{0,t} - \bar{\mathbf{x}}_{0,t}] \right\| \\ &\le \frac{\kappa_t + \xi_t}{\sigma_t}. \end{aligned} \tag{23}$$

Hence, the approximation error is explicitly decomposed into a structural term κ_t determined by the residual parameterization and a learned-estimation term ξ_t determined by the codeword predictor. In particular, improving the neural predictor directly reduces ξ_t, whereas κ_t quantifies the remaining mismatch between the chosen residual direction and the exact diffusion score.

Relation to the BCE training objective. We next clarify the relation between the BCE training objective and the posterior-mean codeword estimate. Let $F_{t,i} \triangleq \mathbb{I}\{\mathrm{sign}(x_{t,i}) \neq x_{0,i}\}$ be the bit-flip indicator and $p_{t,i}^{\mathrm{flip}} = \Pr(F_{t,i} = 1 \mid \mathbf{x}_t)$. BCE is a proper loss for estimating this Bernoulli posterior. For BPSK,

$$x_{0,i} = \mathrm{sign}(x_{t,i})(1 - 2F_{t,i}), \tag{24}$$

and therefore

$$\bar{x}_{0,t,i} = \mathbb{E}[x_{0,i} \mid \mathbf{x}_t] = \mathrm{sign}(x_{t,i}) \left(1 - 2p_{t,i}^{\mathrm{flip}}\right). \tag{25}$$

Thus, the BCE objective estimates the flip posterior that determines the posterior-mean BPSK symbol. The practical hard codeword estimate $\hat{\mathbf{x}}_{0,t}$ is subsequently used to construct the residual, and its departure from the posterior mean is captured by ξ_t.

Algorithm 1 Iterative Decoding Procedure

Input: Channel observation $\mathbf{y}$, Total diffusion steps T, Schedules $\{\eta_t\}_{t=0}^{T}, \{\sigma_t\}_{t=0}^{T}$, skipping step Δt.
Output: Estimated clean codeword $\hat{\mathbf{x}}_0$.
for $t = T, T-1, \ldots, 1$ **do**
  $s \leftarrow \max\{t - \Delta t, 0\}$
  1. Estimate Multiplicative Noise:
  $e_t \leftarrow \|s(\mathbf{y})\|_0$ by (7)
  $\hat{\mathbf{z}} \leftarrow f_\theta([|\mathbf{y}|, s(\mathbf{y})], e_t)$
  2. Estimate Clean Codeword:
  $\hat{\mathbf{x}}_0 \leftarrow \text{sign}(\mathbf{y} \odot \hat{\mathbf{z}})$
  if $e_t = 0$ **then**
    return $\hat{\mathbf{x}}_0$
  end if
  3. Compute Residual:
  $\boldsymbol{\epsilon}_\theta^{\text{add}} \leftarrow \mathbf{y} - \hat{\mathbf{x}}_0$
  4. Compute Dynamic Step Size:
  $\Delta_{\text{step}} \leftarrow (\eta_t - \eta_s) + (\sigma_t - \sigma_s)$
  5. Update:
  $\mathbf{y} \leftarrow \mathbf{y} - \Delta_{\text{step}} \cdot \boldsymbol{\epsilon}_\theta^{\text{add}}$
end for
return $\hat{\mathbf{x}}_0$

Analysis using complete codebook enumeration. We further quantify the approximation in (23) using complete codebook enumeration for Hamming$(7, 4)$ and BCH$(31, 16)$, containing 16 and 65536 codewords, respectively. We use $N = 2, d = 32$ models trained with the same BCE objective and evaluate at 5 dB. Define the per-coordinate RMS score error and the corresponding RMS upper bound as

$$E_{\text{score}}(t) \triangleq \sqrt{\mathbb{E}\left[\frac{\|(\mathbf{x}_t - \hat{\mathbf{x}}_{0,t}) - \bar{\boldsymbol{\epsilon}}_t\|^2}{n\sigma_t^2}\right]}, \tag{26}$$

and

$$B_{\text{score}}(t) \triangleq \sqrt{\mathbb{E}\left[\frac{(\kappa_t + \xi_t)^2}{n\sigma_t^2}\right]}. \tag{27}$$

For Hamming$(7, 4)$, both the measured score error and its bound decrease from 9.14 at $t/T = 0.125$ to 2.62 at $t/T = 1$. For BCH$(31, 16)$, the measured error decreases from 3.52 at $t/T = 0.25$ to 1.40 at $t/T = 1$, while the bound decreases from 3.52 to 1.44. At the latter point, the separate RMS magnitudes of $\kappa_t/(\sigma_t\sqrt{n})$ and $\xi_t/(\sigma_t\sqrt{n})$ are 1.363 and 0.225, respectively. These values show that the upper bound decreases together with the score-approximation error in the examined cases. For BCH$(31, 16)$ at $t/T = 1$, the substantially smaller ξ_t further indicates that the learned-estimation error is not the dominant contributor to the overall approximation error.

E Decoding Algorithm

The complete iterative decoding procedure is presented in Algorithm 1.

F Statistical Evaluation at High SNR

To further assess the statistical reliability of the high-SNR results, where error events become increasingly rare, we repeat the evaluation using five independent Monte Carlo test seeds while keeping the trained model fixed. For each code, SNR, and decoding method, we report the mean BER and BLER together with the 95% confidence interval. The evaluation covers four codes over $E_b/N_0 \in \{5, 5.5, 6, 6.5\}$ dB. As shown in Fig. 7, ChRes-DM achieves a lower mean error rate than DDPM across the complete 5–6.5 dB range for all four codes. These results indicate that the observed high-SNR improvements are consistent across independent test realizations.

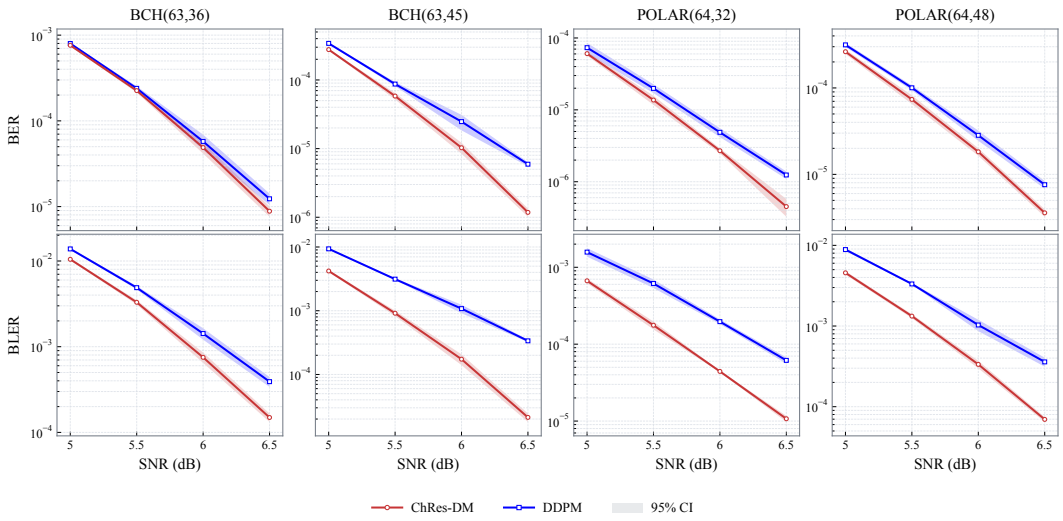

Figure 7: High-SNR decoding performance of ChRes-DM and the DDPM baseline over five independent Monte Carlo tests. The top row reports BER and the bottom row reports BLER; the four columns correspond to BCH$(63, 36)$, BCH$(63, 45)$, Polar$(64, 32)$, and Polar$(64, 48)$, respectively. Curves show the mean error rate across the five seeds, and the shaded regions denote the corresponding 95% confidence intervals of the mean based on Student's t distribution. Lower is better.

G Comparison with Other Decoders

In this section, we compare the proposed method with both classical decoders and representative Transformer-based decoders. All experiments are conducted on an AMD EPYC 7542 32-Core Processor CPU and an NVIDIA RTX 3090 GPU.

G.1 Performance Comparison with Classical Decoders

We present comprehensive comparisons with classical baselines, including SCL and CA-SCL for Polar codes, and OSD for BCH codes in Table 4. These methods are all implemented by the open-source and deeply engineered toolkit named Sionna developed by NVIDIA.

For the short Polar codes considered, CA-SCL with list size $L = 4$ is widely regarded as near-optimal. It is a highly specialized, serial heuristic tailored to the recursive Plotkin construction, and further benefits from CRC-aided path pruning, which significantly enhances its error-correction performance. For BCH codes, OSD at the considered parameters serves as a near-maximum-likelihood (ML) benchmark; it is well known that sufficiently high-order OSD approaches ML performance, and the orders considered here are relatively high.

G.2 Performance Comparison with a Recent Transformer-Based Decoder

For completeness, we include additional comparisons with a recent Transformer-based decoder [4] in Table 5.

G.3 Decoding Latency Comparison

Table 6 reports the decoding latency comparison between classical decoding algorithms and ECCT-based decoders. For the proposed method, the decoding latency is computed as the latency of a single ECCT forward pass multiplied by the average number of diffusion iterations, denoted by T_{avg} in Section 5.2.3.

It is worth noting that the computational complexity of conventional OSD grows combinatorially with both the decoding order t and the code dimension (i.e., number of message bits to be encoded) k, due to the enumeration of all test error patterns up to order t, resulting in a complexity on the

Table 4: Performance comparison ($-\ln(\text{BER})\uparrow$) between the proposed ChRes-DM algorithm and classical decoding algorithms (SCL, CA-SCL, and OSD) over the *AWGN channel*. The CA-SCL adopts the generator polynomial "CRC6" defined in the 5G NR standard. Best results are in **bold**, and second-best are underlined. N/A indicates the error rate is too low to be measured due to the time limit.

Code (Polar)	SCL $L=1$			SCL $L=4$			CA-SCL $L=1$			CA-SCL $L=4$			Ours ($\Delta t=1$)		
E_b/N_0 (dB)	4	5	6	4	5	6	4	5	6	4	5	6	4	5	6
(64,32)	6.58	9.15	12.33	7.35	10.14	13.28	6.15	8.62	12.34	**9.78**	**14.24**	**16.71**	7.20	9.62	12.75
(64,48)	5.08	7.29	9.70	5.52	7.57	10.00	4.10	5.96	8.59	5.92	**8.89**	**12.50**	**6.11**	8.21	10.79
(128,64)	6.76	10.28	15.10	10.20	13.68	16.52	6.81	10.49	14.74	**11.18**	**15.09**	**N/A**	8.36	12.06	15.99
(128,86)	6.40	9.53	13.78	8.06	11.62	16.93	5.94	9.10	12.58	**9.46**	**14.37**	**N/A**	7.85	11.03	15.17
(128,96)	5.43	8.73	11.70	6.96	9.68	12.08	4.81	7.48	10.72	**7.63**	**12.61**	**17.65**	7.31	10.26	13.14

Code (BCH)	OSD $t=1$			OSD $t=2$			OSD $t=4$			Ours ($\Delta t=1$)			Ours ($\Delta t=4$)		
E_b/N_0 (dB)	4	5	6	4	5	6	4	5	6	4	5	6	4	5	6
(63,36)	8.07	10.93	14.73	10.27	14.79	N/A	**10.49**	**15.09**	**N/A**	5.26	7.16	9.68	5.24	7.13	9.76
(63,45)	7.84	11.02	15.37	8.30	11.83	16.36	**8.35**	**11.73**	**N/A**	5.82	8.18	11.44	5.80	8.19	11.61

Table 5: Comparison of decoding performance ($-\ln(\text{BER})\uparrow$) across various error-correcting codes. We compare our proposed method with different step sizes (Δt) against the CrossMPT baseline [4]. For each code and E_b/N_0, the best performance is highlighted in **bold**, and the second-best is underlined.

Decoder	CrossMPT			Ours ($\Delta t=1$)			Ours ($\Delta t=4$)			Ours ($\Delta t=8$)		
E_b/N_0 (dB)	4	5	6	4	5	6	4	5	6	4	5	6
Polar(64,32)	**7.50**	**9.97**	**13.31**	7.20	9.62	12.75	7.19	9.65	12.90	7.12	9.65	12.93
Polar(64,48)	**6.51**	**8.70**	**11.31**	6.11	8.21	10.79	6.08	8.21	10.83	5.95	8.09	10.78
Polar(128,64)	7.52	11.21	14.76	8.36	12.06	**15.99**	**8.37**	**12.10**	15.90	8.33	12.00	15.81
Polar(128,86)	7.51	10.83	15.24	**7.85**	**11.03**	15.17	7.82	10.93	**15.35**	7.72	10.88	15.21
Polar(128,96)	7.15	10.15	13.13	**7.31**	10.26	13.14	7.27	10.32	**13.19**	7.22	**10.28**	13.16
BCH(63,36)	5.03	6.91	9.37	**5.26**	**7.16**	9.68	5.24	7.13	**9.76**	5.17	7.05	9.61
BCH(63,45)	**5.90**	**8.20**	**11.62**	5.82	8.18	11.44	5.80	8.19	11.61	5.65	8.00	11.41
CCSDS(128,64)	7.68	11.88	**17.50**	**8.00**	**12.61**	17.42	7.94	12.55	17.42	7.86	12.42	17.47
MacKay(96,48)	7.97	11.77	15.52	**8.25**	**12.00**	**15.78**	8.16	11.92	15.76	8.12	11.82	15.75

order of $\sum_{i=0}^{t}\binom{k}{i}$. Consequently, the complexity rapidly becomes prohibitive as the code length or decoding order increases. Additionally, SCL and CA-SCL decoders are inherently sequential due to the recursive successive-cancellation structure, where each bit decision depends on previously decoded bits. Maintaining a decoding list of size L further introduces path expansion, pruning, and synchronization overhead, leading to additional memory consumption and decoding latency. As the code length or list size increases, these sequential dependencies increasingly limit parallelism and scalability on modern hardware accelerators.

Table 6: Comparison of decoding latency (ms) for a batch size of 2048 on the same machine. Latency for SCL, CA-SCL, OSD, and BP represents the total decoding time, and these methods are all implemented by the open-source toolkit named Sionna developed by NVIDIA, which is deeply engineered and supports GPU acceleration. For ECCT-based methods (*), the value denotes the time per iteration; total latency is the reported value multiplied by the average number of iterations (T_{avg}). Note that $T_{\text{avg}}=1$ for standard ECCT, while for variants, it depends on the framework, code type, and SNR.

Code	SCL		CA-SCL		OSD			BP			ECCT*	
	$L=1$	$L=4$	$L=1$	$L=4$	$t=1$	$t=2$	$t=4$	$T=50$	$T=100$	$T=150$	$N=2,d=32$	$N=6,d=128$
Polar(64,32)	31.56	31.75	39.99	40.73	—	—	—	28.51	57.50	86.72	$26.75\times T_{\text{avg}}$	$125.16\times T_{\text{avg}}$
Polar(128,64)	63.77	110.42	68.85	119.04	—	—	—	88.91	180.49	272.24	$86.49\times T_{\text{avg}}$	$354.29\times T_{\text{avg}}$
BCH(63,45)	—	—	—	—	14.84	30.20	2960.90	26.62	52.87	78.41	$21.80\times T_{\text{avg}}$	$102.94\times T_{\text{avg}}$
BCH(127,64)	—	—	—	—	56.31	117.02	24612.93	94.62	192.10	289.64	$85.54\times T_{\text{avg}}$	$350.20\times T_{\text{avg}}$

G.4 Comparison with DDIM

To isolate the effect of the proposed residual diffusion formulation from deterministic sampling itself, we construct a matched DDIM baseline [17] based on the conventional Gaussian forward diffusion used in DDPM-based channel decoding [5]. Specifically, the DDIM baseline retains the standard forward process

$$\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{28}$$

where $\sigma_t = \sqrt{\bar{\beta}_t}$. Unlike ChRes-DM, this process does not include the observation-anchored residual schedule η_t. Under the additive-residual parameterization used by the decoder, the corresponding deterministic update from t to $s = t - \Delta t$ is

$$\mathbf{x}_s = \mathbf{x}_t - (\sigma_t - \sigma_s)\, \boldsymbol{\epsilon}_\theta^{\mathrm{add}}(\mathbf{x}_t). \tag{29}$$

Compared with the ChRes-DM update in Eq. (6), the DDIM update contains only the diffusion-schedule term $\sigma_t - \sigma_s$, while ChRes-DM additionally incorporates the residual-bridge term $\eta_t - \eta_s$ derived from its observation-anchored forward process.

We compare the two methods using the same code, pretrained network parameters, SNR, and step size $\Delta t = 4$. Table 7 reports $-\ln(\mathrm{BER})$, where a larger value indicates better decoding performance. We include the five codes for which complete matched results are available under both model configurations. ChRes-DM achieves higher $-\ln(\mathrm{BER})$ than the matched DDIM baseline. Since the step size is fixed to $\Delta t = 4$ for both methods, these results show that the observed gains persist after controlling for deterministic step skipping, supporting the contribution of the observation-anchored residual diffusion formulation.

Table 7: Comparison between ChRes-DM and DDIM at $\Delta t = 4$. The code, model size, SNR, and step size are matched for each comparison. Best results in each setting are shown in **bold** ($-\ln(\mathrm{BER})\uparrow$).

Code	Method	$N=2,\ d=32$			$N=6,\ d=128$		
		4 dB	5 dB	6 dB	4 dB	5 dB	6 dB
Polar(64,32)	ChRes-DM	**6.19**	**8.29**	**11.21**	**7.19**	**9.65**	**12.90**
	DDIM	5.19	6.68	8.29	5.66	6.99	8.40
Polar(64,48)	ChRes-DM	**5.62**	**7.65**	**10.15**	**6.08**	**8.21**	**10.83**
	DDIM	5.24	6.86	8.59	5.58	7.14	8.73
Polar(128,96)	ChRes-DM	**5.69**	**8.01**	**10.90**	**7.27**	**10.32**	**13.19**
	DDIM	5.42	7.54	10.05	6.79	9.00	10.79
BCH(63,36)	ChRes-DM	**4.68**	**6.34**	**8.61**	**5.24**	**7.13**	**9.76**
	DDIM	4.34	5.69	7.32	4.74	6.12	7.61
BCH(63,45)	ChRes-DM	**5.21**	**7.28**	**10.09**	**5.80**	**8.19**	**11.61**
	DDIM	4.91	6.57	8.49	5.35	7.04	8.75

H Impact of Scheduling Strategies

We conduct an ablation study using five different noise scheduling strategies. Since our decoder conditions the diffusion process on the syndrome weight–a state-dependent metric–rather than a rigid time-step index, it allows for zero-shot evaluation on schedules different from the training setup. We compare the following schedules, all constrained to the same range, as illustrated in Fig. 8:

- Linear Beta (Ours): $\eta_t, \sigma_t \propto \sqrt{t}$, implying a concave trajectory that allocates smaller denoising steps in the high-noise regime (large t).
- Linear Sigma: $\eta_t, \sigma_t \propto t$, representing a uniform noise reduction rate.
- Cosine: $\eta_t, \sigma_t \propto \sin(\frac{\pi t}{2T})$, a standard schedule in generative modeling [32] that smooths the transition, exhibiting slow decay characteristics near the large noise level similar to the Linear Beta schedule.

- Quadratic: $\eta_t, \sigma_t \propto t^2$, inspired by EDM [33], this convex schedule aggressively reduces noise in the early stages and reserves the majority of iterations for fine-grained refinement at low noise levels.
- Geometric: $\eta_t, \sigma_t \propto c^t$ with a constant c defined as in [18], characterized by rapid initial noise reduction and extremely dense sampling near the zero-noise range.

The performance results in Table 8 indicate that while the optimal schedule can vary slightly depending on the specific code structure and SNR levels, the Linear Beta schedule offers robust performance across the tested configurations. Furthermore, our default Linear Beta schedule aligns with the training phase in the literature, ensuring that the residual drift term operates within its learned effective range. We therefore use the Linear Beta schedule as the default setting.

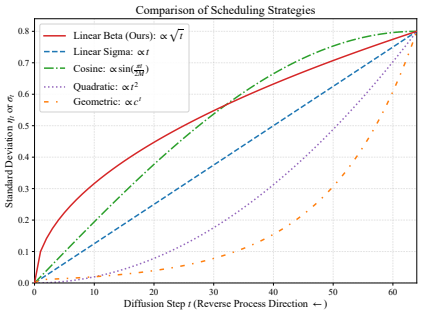

Figure 8: Comparison of different scheduling strategies

Table 8: Performance comparison of different scheduling strategies. For each strategy, we report $-\ln(\text{BER})$ (the first row) and $-\ln(\text{BLER})$ (the second row). Higher is better.

	Linear			Cosine			Quadratic			Geometric			Default (Ours)		
E_b/N_0 (dB)	4	5	6	4	5	6	4	5	6	4	5	6	4	5	6
BCH(63,45)	5.19	6.58	8.06	5.60	7.48	9.70	4.53	5.31	6.11	4.18	4.90	5.70	5.82	8.18	11.44
	1.61	2.65	3.95	2.37	3.86	5.68	0.77	1.29	2.00	0.50	0.95	1.64	3.07	5.20	7.87
Polar(64,32)	6.27	7.87	9.58	7.15	9.69	12.40	4.12	4.49	4.92	3.62	4.06	4.58	4.78	6.81	9.30
	2.50	3.81	5.42	4.36	6.34	8.31	0.38	0.60	0.92	0.16	0.34	0.68	7.20	9.62	12.75
LDPC(49,24)	5.23	6.74	8.29	5.86	8.20	11.64	3.84	4.32	4.84	3.39	3.83	4.36	5.90	8.25	11.61
	1.94	3.06	4.42	3.64	5.89	9.04	0.32	0.59	1.01	0.15	0.34	0.68	3.95	6.27	9.61

I Robustness on Non-Gaussian Channel

To evaluate the generalization capability of our proposed ChRes-DM, we extend our evaluation to non-Gaussian channels. Specifically, we apply the models trained only on the AWGN channel *directly* to a Rayleigh flat-fading channel without any fine-tuning. This "zero-shot" setting serves as a test of the decoder's robustness to mismatched channels.

The received signal under the flat-fading model is given by $\mathbf{y} = \mathbf{h} \odot \mathbf{x} + \mathbf{n}$, where $\odot$ denotes element-wise multiplication. The fading coefficients $\mathbf{h}$ are assumed to be i.i.d. following a Rayleigh distribution. To ensure a fair comparison with unit average power gain (i.e., $\mathbb{E}[h^2] = 1$), we generate the raw fading coefficients from a Rayleigh distribution with a scale parameter of $\alpha = 1$ and subsequently normalize them by dividing by $\sqrt{2}$ [26]. The noise $\mathbf{n}$ remains i.i.d. Gaussian. Due to the destructive nature of deep fades, the evaluation is conducted at higher E_b/N_0 levels compared to the AWGN experiments.

We compare the performance of our ChRes-DM against the baseline DDPM using a single-step update ($\Delta t = 1$) for both methods with network configuration $N = 6, d = 128$. Table 9 reports the decoding accuracy in terms of $-\ln(\text{BER})$, $-\ln(\text{BLER})$, and the average number of iterations required to reach a valid codeword (all-zero syndrome).

Table 9: Performance on Rayleigh fading channels

		DDPM				Ours			
E_b/N_0 (dB)	Metric	8	9	10	11	8	9	10	11
BCH(63,45)	$-\ln(\text{BER})$	4.21	4.92	5.79	6.70	4.21	4.92	5.80	6.70
	$-\ln(\text{BLER})$	1.68	2.34	3.17	4.08	1.75	2.42	3.25	4.15
	It. To 0	9.11	7.30	5.69	4.43	8.17	6.42	4.93	3.80
Polar(64,32)	$-\ln(\text{BER})$	6.75	8.00	9.30	10.86	6.75	7.97	9.33	10.87
	$-\ln(\text{BLER})$	4.57	5.84	7.18	8.68	4.67	5.89	7.26	8.80
	It. To 0	11.67	9.54	7.71	6.20	10.03	8.13	6.53	5.21

While the performance gap is modest, ChRes-DM consistently yields slightly higher accuracy and comparable convergence speed relative to the baseline DDPM. The results suggest that the proposed update remains effective under this tested channel mismatch without fine-tuning.

J Compatibility with Other Backbones

To further validate the compatibility of the proposed ChRes-DM framework with different neural decoders, we investigate its applicability to two recent Transformer-based model-free decoder backbones, MM-ECCT [3] and CrossMPT [4]. We first integrate MM-ECCT as the denoising backbone within our diffusion framework, and then further evaluate CrossMPT as an alternative neural estimator.

The original MM-ECCT employs a dual-branch architecture, where two parallel branches process the input using masks derived from two algebraically equivalent parity-check matrices of the same code. The outputs are subsequently fused to generate the final codeword estimate. To adapt this single-pass model to our iterative diffusion framework, we introduced the following modifications: Consistent with our main methodology, the time embedding for each branch is derived from the number of non-zero elements of the syndrome vector corresponding to its specific parity-check matrix. This allows each branch to sense the noise level relative to its own parity constraints. Apart from the injection of these time embeddings, the internal dual-branch structure remains identical to the original design. We train the MM-ECCT networks with ChRes-DM from scratch, maintaining the same training settings as in the main text.

Fig. 9 presents the decoding performance under the ChRes-DM framework compared to the original baseline. The experimental results demonstrate that applying our ChRes-DM framework yields consistent performance gains over the original MM-ECCT.

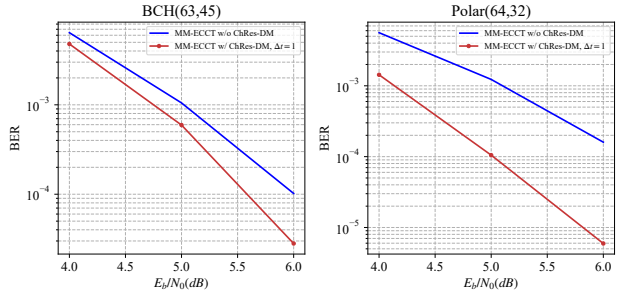

Figure 9: BER results for MM-ECCT with and without ChRes-DM. The model parameters are set as $N = 2, d = 128$. The BER results for the original MM-ECCT are obtained directly from the source literature.

We further evaluate ChRes-DM with CrossMPT [4], a recent Transformer-based decoder that improves the neural decoding backbone through cross-attention message passing. Table 10 compares the

resulting decoder with standalone CrossMPT and the original ECCT-based ChRes-DM with $\Delta t = 1$ under the same code–SNR settings.

Table 10: Compatibility of ChRes-DM with the CrossMPT backbone. Decoding performance is measured by $-\ln(\mathrm{BER}) \uparrow$. Standalone CrossMPT and the ECCT-based ChRes-DM ($\Delta t = 1$) are included as references. The best result for each code and E_b/N_0 is highlighted in **bold**.

Code	Method	E_b/N_0 (dB)		
		4	5	6
Polar(64,32)	CrossMPT	7.50	9.97	13.31
	ChRes-DM ($\Delta t = 1$)	7.20	9.62	12.75
	CrossMPT + ChRes-DM	**7.99**	**10.68**	**13.89**
Polar(64,48)	CrossMPT	6.51	8.70	**11.31**
	ChRes-DM ($\Delta t = 1$)	6.11	8.21	10.79
	CrossMPT + ChRes-DM	**6.67**	**8.71**	11.20
Polar(128,64)	CrossMPT	7.52	11.21	14.76
	ChRes-DM ($\Delta t = 1$)	8.36	12.06	**15.99**
	CrossMPT + ChRes-DM	**9.77**	**12.95**	14.34
BCH(63,36)	CrossMPT	5.03	6.91	9.37
	ChRes-DM ($\Delta t = 1$)	5.26	7.16	9.68
	CrossMPT + ChRes-DM	**6.05**	**8.28**	**11.30**
BCH(63,45)	CrossMPT	5.90	8.20	11.62
	ChRes-DM ($\Delta t = 1$)	5.82	8.18	11.44
	CrossMPT + ChRes-DM	**6.77**	**9.61**	**13.52**

CrossMPT+ChRes-DM improves over standalone CrossMPT and over the ECCT-based ChRes-DM. Although the combined decoder does not dominate at every SNR point, the improvements observed across both Polar and BCH codes are consistent with the MM-ECCT results above: advances in the neural backbone for noise estimation can be incorporated into the ChRes-DM inference framework and provide gains that are complementary to the iterative restoration process.

K Integration with Classical List Decoding

To examine whether ChRes-DM can benefit from classical candidate generation, we integrate order-1 ordered-statistics decoding (OSD-1) into the reverse trajectory and evaluate the resulting decoder on BCH$(63, 45)$. At each reverse step, the neural predictor provides the reliabilities used to construct an OSD ordering for every active trajectory. OSD-1 enumerates the weight-0 and weight-1 test-error patterns over the $k = 45$ information positions, producing $M = 1 + k = 46$ valid codeword candidates per active trajectory. After merging duplicate codewords, the best L candidates are retained according to the maximum-likelihood metric for BPSK-AWGN, $d(\mathbf{c}; \mathbf{y}) = \|\mathbf{y} - (1 - 2\mathbf{c})\|_2^2$, where $\mathbf{y}$ is the original channel observation. Each retained codeword is then used as the clean-codeword estimate for the corresponding deterministic ChRes-DM update. Here, L denotes the number of retained trajectories and is distinct from the $M = 46$ candidates generated by each OSD-1 invocation. We evaluate $L = 1$ and $L = 2$.

Because standalone OSD-1 already yields very few errors at higher SNRs, we evaluate this extension at 2 and 3 dB, where sufficient error events remain for a meaningful comparison. Table 11 reports the mean BLER over three independent tests. Both iterative OSD variants substantially improve the original ChRes-DM decoder and also outperform standalone OSD-1.

The additional computation is controlled by the retained width L. Let K denote the number of reverse steps and $M = 46$ the number of OSD-1 candidates generated per active trajectory. Standard ChRes-DM uses K predictor evaluations, whereas the list extension requires at most $1 + L(K - 1)$ active-trajectory evaluations using the neural backbone and $M\,[1 + L(K - 1)]$ OSD candidate evaluations. The active trajectories can be processed in parallel within each reverse step, while both candidate-processing cost and active-state memory scale approximately linearly with L for fixed K and M. In our $K = 18$ experiment, $L = 1$ corresponds to 18 predictor evaluations and 828 nominal candidate evaluations per frame, while $L = 2$ corresponds to 35 and 1610, respectively. These results show that OSD-generated codeword candidates can be incorporated throughout the ChRes-DM trajectory, providing an additional performance–complexity trade-off through the retained list width L.

Table 11: BLER comparison on BCH$(63, 45)$, averaged over three independent tests. The iterative variants use OSD order 1 and retain L trajectories throughout the ChRes-DM reverse process. Best results are in **bold**.

Method	2 dB	3 dB
ChRes-DM	4.709×10^{-1}	1.958×10^{-1}
Standalone OSD-1	1.650×10^{-1}	3.162×10^{-2}
Iterative OSD-list ChRes-DM ($L = 1$)	1.524×10^{-1}	2.747×10^{-2}
Iterative OSD-list ChRes-DM ($L = 2$)	$\mathbf{1.499 \times 10^{-1}}$	$\mathbf{2.653 \times 10^{-2}}$

L Scalability on Longer Codes

To explore the scalability of ChRes-DM beyond short block codes, we evaluate its performance on a longer LDPC code from the IEEE 802.16e (WiMAX) standard, with parameters $(576, 432)$. Since belief propagation (BP) is the standard SISO decoding algorithm for LDPC codes, we compare our method against the normalized min-sum algorithm (NMSA) with 50 iterations, which is a widely deployed BP-like decoding algorithm for LDPC codes. Following the settings in [26], we adopt the scaling factor of 0.5 in the NMSA, which provides robust performance across various conditions without requiring SNR-specific tuning.

It is important to acknowledge that the sparsity inherent in LDPC codes (especially very long codes) results in Tanner graphs with relatively few short cycles, thereby enabling message-passing algorithms (such as BP) to effectively exploit their well-known optimality on cycle-free graphs in the asymptotic limit. Our results serve as a proof of concept, validating that the proposed generative restoration framework is capable of handling longer codes.

Furthermore, this comparison reinforces the primary motivation of our work presented in the main text. Unlike the *sparse* graphs of LDPC codes, short block codes (e.g., BCH and Polar codes) are characterized by *dense* parity-check matrices with many short cycles, a regime where the performance of BP algorithms degrades significantly. Since achieving effective SISO decoding under these structural constraints remains a challenging problem, our work explores diffusion models not to displace BP in its optimal domain, but to provide a robust, high-performance SISO alternative specifically where traditional graphical assumptions break down.

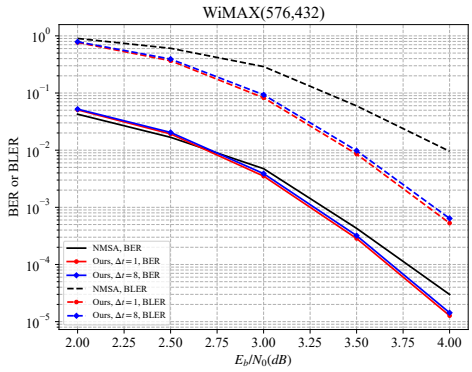

Figure 10: BER and BLER results for the WiMAX $(576, 432)$ LDPC code. The maximum number of iterations of NMSA is 50, and our model parameters are set as $N = 6, d = 32$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately describe the paper's main contributions and scope, namely improving diffusion-based channel decoding through the proposed framework.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section 6, we discuss the limitations of the proposed method in terms of performance and efficiency compared with classical decoding algorithms. These limitations are common to neural decoders, since the considered classical baselines are already well-established near-optimal solutions under the evaluated settings. Our work primarily aims to provide a novel framework for reformulating diffusion-based channel decoding and improving its efficiency.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Building upon established and widely accepted results, we provide complete proofs for all theoretical results presented in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All necessary information required to reproduce the main experimental results is included in the submitted paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The proposed algorithm is described in detail in the paper, and the employed pipeline is based on the existing open-source method (i.e., baseline DDPM). Therefore, the paper contains sufficient information for faithful reproduction of the main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup is largely based on established existing methods. All necessary training and testing details are specified in the paper, including detailed hyperparameter settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The experiments follow the widely adopted Monte Carlo simulation in channel coding, where a sufficiently large number of samples are collected (i.e., at least 100 erroneous codewords for each SNR point). Under this setting, the statistical error is negligible compared with the reported performance differences.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The required computational resources, including the specific hardware used for the experiments, are provided in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research presented in this paper complies with the NeurIPS Code of Ethics and does not involve any ethical concerns related to data usage, human subjects, or harmful applications.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This paper presents work whose goal is to advance the field of machine learning and information theory for reliable digital communications. Our proposed framework aims to enhance the computational efficiency and accuracy of error correction systems, potentially contributing to a more energy-efficient communication infrastructure, and we do not foresee any specific negative ethical consequences.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not involve the release of high-risk models, scraped datasets, or other resources with significant potential for misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper properly credits the sources of the employed codebase and datasets. The implementation is based on an existing repository released under the MIT License, whose terms are respected. The parity-check matrices are obtained from publicly available benchmark datasets commonly used in the literature.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This work does not involve human participants, personal data, or any procedures requiring IRB approval or equivalent ethical review.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used as important, original, or non-standard components of the core methodology in this work. Any use of LLMs was limited to writing or editing assistance and does not affect the scientific contributions of the paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.