

# A STATISTICAL BENCHMARK FOR DIFFUSION POSTERIOR SAMPLING ALGORITHMS

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

1 We propose a statistical benchmark for diffusion posterior sampling (DPS) al-  
 2 gorithms in linear inverse problems. Our test signals are discretized Lévy pro-  
 3 cesses whose posteriors admit efficient Gibbs methods. These Gibbs methods  
 4 provide gold-standard posterior samples for direct, distribution-level compar-  
 5 isons with DPS algorithms. They can also sample the denoising posteriors in  
 6 the reverse diffusion, which enables the arbitrary-precision Monte Carlo esti-  
 7 mation of various objects that may be needed in the DPS algorithms, such as  
 8 the expectation or the covariance of the denoising posteriors. In turn, this can  
 9 be used to isolate algorithmic errors from the errors due to learned compo-  
 10 nents. They also serve as oracle denoisers in the reverse diffusion, which enables  
 11 the isolation of the error that arises from the approximations to the likelihood  
 12 score. We instantiate the benchmark with the minimum-mean-squared-error op-  
 13 timality gap and posterior-coverage tests and evaluate popular algorithms on  
 14 the inverse problems of denoising, deconvolution, imputation, and reconstruction  
 15 from partial Fourier measurements. We release the benchmark code at <https://github.com/emblem-saying/dps-benchmark> and The repository  
 16 exposes simple plug-in interfaces, reference scripts, and config-driven runs so that  
 17 new algorithms can be added and evaluated with minimal effort. We invite the  
 18 community to contribute and report results.

## 20 1 INTRODUCTION

21 Diffusion models are among the leading generative models in imaging (Rombach et al., 2022),  
 22 visual computing (Po et al., 2024), finance and time-series analysis (Huang et al., 2024; Rasul et al.,  
 23 2021), de novo protein and drug design (Watson et al., 2023; Alakhdar et al., 2024), natural language  
 24 processing (Li et al., 2022), and other domains. Their ability to model complex distributions has  
 25 motivated their use as priors in the Bayesian resolution of inverse problems. In fact, reconstruction  
 26 methods that leverage diffusion models are competitive or state-of-the-art for problems such as,  
 27 e.g., deconvolution (Ren et al., 2023), phase retrieval (Xue et al., 2025), magnetic resonance imaging  
 28 and computed tomography reconstruction (Chung & Ye, 2022; Liu et al., 2023), weather-artifact  
 29 removal (Özdenizci & Legenstein, 2023), task-conditioned protein design (Bogensperger et al., 2025),  
 30 audio bandwidth extension and dereverberation (Lemercier et al., 2024), and denoising of financial  
 31 time-series (Wang & Ventre, 2024).

32 This empirical success has come in spite of a lack of~~despite~~ diffusion models lacking a natural  
 33 mechanism for the conditioning on measurements and active research explores how to incorporate  
 34 the likelihood (Yismaw et al., 2025; Erbach et al., 2025). Currently, conditioning strategies are  
 35 evaluated in one of two ways. (i) With respect to downstream applications: As an example, evaluations  
 36 with respect to perceptual metrics such as the structural similarity (Wang et al., 2004), the Fréchet  
 37 inception distance (Heusel et al., 2017), or the learned perceptual image-patch similarity (Zhang et al.,  
 38 2018) are common in the imaging sciences. A~~However~~, as pointed out by Pierret & Galerne (2025b)  
 39 and Cardoso et al. (2024), however, these metrics are ill-suited for the statistical evaluation of posterior  
 40 sampling algorithms. (ii) In overly simplistic settings: A common fallback is to evaluate conditioning  
 41 strategies in synthetic settings with (finite-component) Gaussian-mixture priors. Such mixtures  
 42 remain light-tailed with the tail decreasing exponentially like the widest component. Consequently  
 43 and, consequently, they cannot reproduce power-law-like extremes that are common in e.g., asset  
 44 returns (Blattberg & Gonedes, 1974; Cont, 2001) or and statistics of images (Wainwright & Simoncelli,

**[tkeZ]** Clarification of terminology and phrasing regarding our “oracle” objects.

**[tkeZ TmEt]** Rephrase to emphasize that isolating algorithmic errors from learned approximations is possible but not done systematically.

45 1999). We illustrate signals with such power-law-like extremes later in Figure 3. Benchmarks built  
 46 on such priors can therefore overstate posterior quality. A proper statistical evaluation in realistic  
 47 settings is critical in high-stakes applications such as medical imaging, remote sensing, and finance,  
 48 where decisions based on reconstructions and their associated uncertainties may have significant  
 49 consequences.

50 1.1 CONTRIBUTIONS

51 We propose such a statistical benchmark for diffusion posterior sampling (DPS) algorithms<sup>1</sup> for  
 52 linear inverse problems. OurWe consider a setting similar to Bohra et. al (2023) in which test signals  
 53 are obtained from discretized sparse Lévy- processes priors that admit efficient posterior-sampling  
 54 algorithms. Indeed, they admit efficient Gibbs methods with exact conditionals that provide gold-  
 55 standard posterior samples. OurThe framework supports general posterior-level comparisons (—e.g.,  
 56 (sliced) Wasserstein or energy distances or calibration via coverage and posterior predictive checks)  
 57 — by furnishing matched samples obtained from the DPS algorithms and the gold-standard Gibbs  
 58 methods.

[tkeZ] Clarification of our use of the DPS acronym.

59 The Gibbs methods are also suited to sample from the denoising posteriors in the reverse diffusion.  
 60 This motivates our contribution ofWe introduce a new template for DPS algorithms, in which update  
 61 steps utilize where each reverse-diffusion-step utilizes samples from the corresponding denoising  
 62 posterior. These samples can be used for arbitrary-precision Monte Carlo estimation of various ob-  
 63 jects that are needed in the update steps of the algorithms, such as the minimum-mean-squared-error  
 64 (MMSE) denoiser or its Jacobian, which enables the isolation of algorithmic errors from approxima-  
 65 tion errors due to learned components—as opposed to only the MMSE point estimate. Wand we  
 66 show how several popular DPS algorithms can be re-expressed within ourthis template. This template  
 67 arises naturally in our framework because the Gibbs methods can provide these denoising-posterior  
 68 samples. Our construction enables the isolation of algorithmic errors from errors due to learned  
 69 components by replacing those learned components with arbitrary-precision Monte Carlo estimates.

[tkeZ] Clarification of terminology and phrasing regarding the “oracle” objects.

70 Finally, we instantiate the framework with the MMSE optimality gap and highest-posterior-density  
 71 coverage checks across the inverse problems of denoising, deconvolution, imputation, and reconstruc-  
 72 tion from partial Fourier measurements. We target the realistic scenario where a learned denoiser is  
 73 used and check hyperparameter sensitivity by substituting the arbitrary-precision Monte Carlo counter-  
 74 parts for the learned components. The benchmark code—which is another substantial contribution—is  
 75 available in an online. Itrepository that contains efficient implementations of sampling routines and a  
 76 containerized runtime that allows novel algorithms to be easily benchmarkedbenhmarked easily.

[tkeZ TmEt] Clearly delineating isolating algorithmic error from robustness (for fixed hyperparameters) w.r.t. denoiser quality.

77 1.2 RELATED WORK

78 For unconditional sampling, many works derive theoretical bounds on various distances between  
 79 a target distribution and the distribution obtained by (approximations of) the reverse stochastic  
 80 differential equation (SDE) (see Section 2). For example, Gao et al. (2025) bound the Wasserstein-2  
 81 distance with respect to the discretization error of the SDE under the assumption that the target  
 82 distribution is smooth and log-concave. This directly bounds the number of reverse-diffusion steps  
 83 that are needed to obtain a desired accuracy. Under absolute continuity of the target with respect to a  
 84 Gaussian, Strasman et al. (2025) bound the Kullback–Leibler divergence with respect to properties of  
 85 the noise schedule. Additional results in other distances can be found in the references cited therein.

86 A common assumption that simplifies the analysis and facilitates the computation of various errors  
 87 and bounds is that of a Gaussian target. In that case, many objects in the forward and reverse SDE  
 88 admit closed forms, which facilitates the computation of various bounds. For example, Hurault et al.  
 89 (2025) analyze the error incurred when using due to the finite number of prior samples used for  
 90 in the estimation of the prior score (which is affine in this case) and track its propagation through  
 91 the iterations of the reverse-SDE solver. Pierret & Galerne (2025b) derive explicit solutions to the  
 92 SDE and use themthose to derive bounds on the Wasserstein-2 distance to the distributions that are  
 93 obtained via Euler–Maruyama discretizations.

<sup>1</sup>We use “DPS algorithms” as an umbrella term for posterior-sampling methods with diffusion priors. The method due to Chung et al. (2023), often called DPS in the literature, will be referred to later as C-DPS.

94 For Works that consider conditional sampling generation, and are closest to the present paper are  
 95 Pierret & Galerne (2025a) and Crafts & Villa (2025). Pierret & Galerne (2025a) derive expressions  
 96 for the Wasserstein-2 distances between the conditional forward marginals and the distributions  
 97 induced by specific likelihood approximations in the reverse SDE under the assumption of a Gaussian  
 98 prior. In contrast, our framework handles a broader set of priors (discretized Lévy processes) and  
 99 accommodates a broader set of algorithms than those that rely on specific likelihood approximations.  
 100 Moreover, deriving explicit expressions for new algorithms often requires a substantial amount of  
 101 nontrivial mathematics. In contrast, our benchmark is deliberately designed for a plug-and-play  
 102 evaluation of novel algorithms. Crafts & Villa (2025) systematically evaluate DPS algorithms numer-  
 103 ically under the assumption of a (finite-component) Gaussian-mixture prior numerically and. Similar  
 104 to the present work, they provide reference objects to the DPS algorithms to ensure a fair evaluation.  
 105 Cardoso et al. (2024) and Boys et al. (2024) also evaluate their algorithms on Gaussian-mixture pri-  
 106 ors. These However, they only consider Gaussian-mixture priors, however, which cannot reproduce  
 107 power-law-like extremes and can overstate posterior quality.

108 Beyond diffusion-specific theory, Thong et al. (2024) evaluate posterior calibration by checking the  
 109 coverage of credible regions produced by different Bayesian recovery strategies and. They find that  
 110 those recovery strategies that utilize diffusion models often under-report uncertainty. A shortcoming  
 111 of their approach is that they use an empirical distribution of images as a surrogate for the prior  
 112 distribution. Our framework, by contrast, relies on known priors from which infinitely many signals  
 113 and corresponding measurements can be generated. It isolates algorithmic errors without resorting  
 114 to surrogate priors and supports fair, repeatable comparisons across tasks and algorithms. Finally,  
 115 Bohra et al. (2023) also used efficient Gibbs methods to obtain gold-standard posterior samples.  
 116 Their main focus was to quantify the quality of neural MMSE estimators with different number of  
 117 parameters. Our work extends this to posterior-level comparisons.

## 118 2 BACKGROUND

119 **Bayesian Linear Inverse Problems** We seek to estimate a signal  $\mathbf{x} \in \mathbb{R}^d$  from the measurements

$$120 \mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}, \quad (1)$$

121 where the *forward operator*  $\mathbf{A} \in \mathbb{R}^{m \times d}$  models the noiseless linear measurement acquisition and  
 122  $\mathbf{n} \in \mathbb{R}^m$  is additive noise. In the Bayesian resolution of this problem (Stuart, 2010) (see, e.g., Stuart  
 123 (2010)), the signals are modeled as a random variable, denoted  $\mathbf{X}$ , with values in  $\mathbb{R}^d$  and distribution  
 124  $p_{\mathbf{X}}$ , referred to as the *prior*. Given any measurement  $\mathbf{y}$ , the ultimate goal is to analyze the *posterior*  
 125  $p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}$ . It which is related to the *likelihood*  $p_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}$  and the prior  $p_{\mathbf{X}}$  via Bayes' rule, which states  
 126 that

$$126 p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) \propto p_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y}) p_{\mathbf{X}}(\mathbf{x}). \quad (2)$$

127 In contrast to classical variational methods (Scherzer et al., 2008) (see, e.g., Scherzer et al (2008)),  
 128 the posterior distribution provides natural means to quantify uncertainty and can be summarized by  
 129 various point estimators. We provide a precise description of point estimators that are relevant in this  
 work in [Appendix A](#).

130 For a given signal  $\mathbf{x}$ , the likelihood  $p_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}$  is fully specified determined by the distribution of the  
 131 noise. A common assumption on the noise is that it is a vector of independent and identically  
 132 distributed (i.i.d.) Gaussian random variables with mean zero and variance  $\sigma_n^2$ .<sup>2</sup> In this case, the  
 133 likelihood is given by

$$133 p_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y}) \propto \exp\left(-\frac{1}{2\sigma_n^2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2\right). \quad (3)$$

134 Thus, once the forward model and the noise distribution are specified, the remaining modeling choice  
 135 is the prior. Diffusion models are good candidates due to their ability to encode complex distributions.

136 **Diffusion Models** Diffusion models were introduced by Song et al. (2021) by unifying the discrete  
 137 approaches proposed by from Song & Ermon (2019) and Ho et al. (2020) in a continuous theory based  
 138 on SDEs (Klenke, 2020, Chapters 25 and 26). We denote the (diffusion) SDE with *drift coefficient*  
 139  $\mathbf{f} : \mathbb{R}^d \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^d$  and *diffusion coefficient*  $g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$  as

$$139 d\mathbf{X}_t = \mathbf{f}(\mathbf{X}_t, t) dt + g(t) d\mathbf{W}_t, \quad (4)$$

<sup>2</sup>Our framework supports more general (possibly non-Gaussian) likelihoods, see [Section 3](#).

[tkeZ TmEt] Removed passages that may indicate that we systematically remove approximation errors.

[bQ8j] Added relevant references.

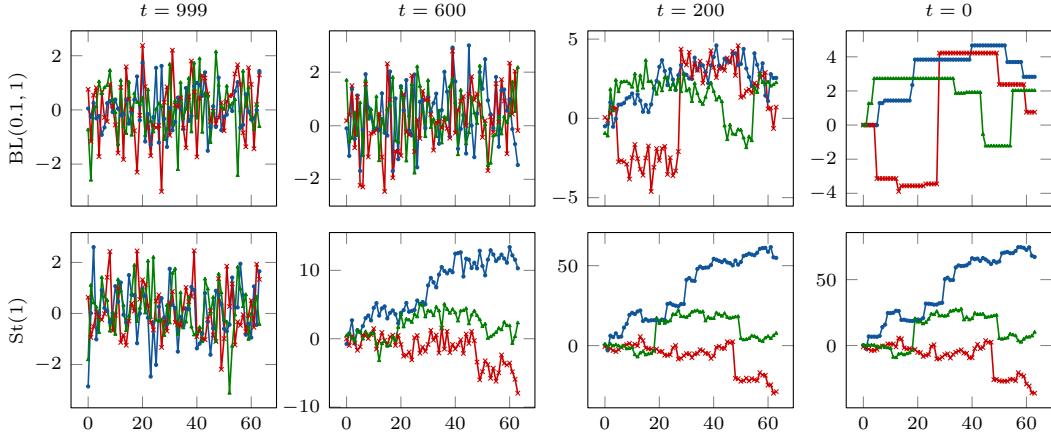


Figure 1: Unconditional reverse-diffusion trajectories obtained by DDPM using the arbitrary-precision Monte Carlo oracle denoiser. Rows: Increment distributions. Columns: Diffusion times. Line styles: Different random states.

140 with some suitable initial condition  $\mathbf{X}_0$ , where  $\mathbf{W}_t$  is the standard Wiener process. In our setup, the  
 141 initial condition  $\mathbf{X}_0$  is the random variable that describes the signal, thus,  $\mathbf{X}_0 = \mathbf{X}$ . Under suitable  
 142 choices for  $f$  and  $g$ , the forward process admits a limiting marginal  $\mathbf{X}_\infty$  as  $t \rightarrow \infty$ . Sampling from  
 143  $p_{\mathbf{X}_0}$  can then proceed by simulating the SDE (4) in reverse with initial condition  $\mathbf{X}_\infty$ . By Anderson’s  
 144 theorem (Anderson, 1982), the reverse SDE that reproduces the forward marginals satisfies

$$d\mathbf{X}_t = (f(\mathbf{X}_t, t) - g^2(t) \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t)) dt + g(t) d\mathbf{W}_t, \quad (5)$$

145 where  $p_{\mathbf{X}_t}$  denotes the density of  $\mathbf{X}_t$  defined by the forward process, and  $dt$  is negative.

146 The primary challenge in this approach lies in the computation of the scores  $\nabla \log p_{\mathbf{X}_t}$  for all  $t > 0$ .  
 147 A fundamental relation known as Tweedie’s formula connects the score with the MMSE denoiser: As  
 148 we derive rigorously in Appendix B, for  $f(\mathbf{x}, t) = (-\frac{\beta(t)}{2})\mathbf{x}$  and  $g(t) = \sqrt{\beta(t)}$ , we have that<sup>3</sup>

$$\nabla \log p_{\mathbf{X}_t}(\mathbf{x}) = -\sigma(t)^{-2}(\mathbf{x} - \alpha(t)\mathbb{E}[\mathbf{X}_0 \mid \mathbf{X}_t = \mathbf{x}]), \quad (6)$$

149 where  $\alpha(t) = \exp(-\frac{1}{2} \int_0^t \beta(s) ds)$  and  $\sigma^2(t) = (1 - \alpha^2(t))$ . This yields a practical way to compute  
 150 of computing  $\nabla \log p_{\mathbf{X}_t}(\mathbf{x})$  through the resolution of the MMSE denoising problem of finding  
 151  $\mathbb{E}[\mathbf{X}_0 \mid \mathbf{X}_t = \mathbf{x}]$ . In standard applications where the goal is the generation of new signals, one this is  
 152 typically tackled this by approximating the map  $(\mathbf{x}, t) \mapsto \mathbb{E}[\mathbf{X}_0 \mid \mathbf{X}_t = \mathbf{x}]$  with a neural network that  
 153 is learned in an off-line step. In our framework benchmark, we can instead obtain arbitrary-precision  
 154 oracle MMSE denoisers via Gibbs methods and thereby eliminate approximation errors from a  
 155 learned surrogate and isolate errors in DPS algorithms themselves.

156 The implementation of implementing the reverse SDE for generation requires its own time discretization,  
 157 for instance with Euler–Maruyama techniques (Higham, 2001). In this work, we will base  
 158 our backward processes on In practice, researchers typically use the alternative denoising diffusion  
 159 probabilistic model (DDPM) backward process (starting from  $\text{Gauss}(\mathbf{0}, \mathbf{I})$ )

$$\mathbf{X}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} (\mathbf{X}_t + \beta_t \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t)) + \sqrt{\beta_t} \mathbf{Z}_t, \quad (7)$$

160 that originates from the discrete-time Markov chain that was initially proposed by Sohl-Dickstein  
 161 et al. (2015) and revisited and popularized by Ho et al. (2020). We relate it to the Euler–Maruyama  
 162 discretization of the reverse SDE through Taylor expansions in Appendix B.1.

163 Though we defer details on our signals and the Gibbs methods that we use to obtain the arbitrary-  
 164 precision MMSE denoiser to Section 3, we demonstrate in Figure 1 that our signals can be generated by  
 165 coupling the unconditional backward process in (7) with this denoiser. We show trajectories of signals  
 166 generated by this backward process using the oracle MMSE denoiser in Figure 1. We further and

[tkeZ] Rephrasing of “researchers typically use”.

[tkeZ bQ8j] Improvement of the structure (bQ8j) and clarification of the purpose of Figure 1 (tkeZ).

<sup>3</sup>This is the variance-preserving (VP) formulation (Song et al., 2021, Section 3.4) with standard normal limiting marginal, where  $\beta : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  controls the speed of the contraction to zero and how much noise is injected.

167 motivate this arbitrary-precision the oracle denoiser in Figure 2 by comparing showing the histograms  
 168 of signal increments produced obtained by the learned denoiser and versus the arbitrary-precision  
 169 oracle denoiser for a  $St(1)$  increment target in figure 2 (notations summarized in Appendix C.2). The  
 170 construction of the signals and the oracle denoiser are described in Section 3 and our notations of  
 171 various distributions are summarized in Appendix C.2. The increments of signals generated with by  
 172 using the arbitrary-precision oracle denoiser follow the target increment almost perfectly. Residual  
 173 errors are due to the discretization error of the reverse diffusion and Monte Carlo error of the  
 174 arbitrary-precision oracle denoiser.

175 **Diffusion Posterior Sampling** Our The  
 176 reverse-diffusion sampler from the previous  
 177 section can be adapted to sample from a  
 178 posterior by replacing the prior score  $\nabla \log p_{\mathbf{X}_t}$   
 179 with the posterior score

$$\nabla \log p_{\mathbf{X}_t | \mathbf{Y} = \mathbf{y}} = \nabla \log p_{\mathbf{X}_t} + \nabla (\mathbf{x} \mapsto \log p_{\mathbf{Y} | \mathbf{X}_t = \mathbf{x}}(\mathbf{y})) \quad (8)$$

180 for some given measurement  $\mathbf{y}$ , obtained by  
 181 Bayes' theorem. Although the dependence be-  
 182 tween  $\mathbf{Y}$  and  $\mathbf{X}_0$  is known through (1) and the  
 183 likelihood is explicitly modeled via (3), it is gen-  
 184 erally challenging to relate  $\mathbf{Y}$  and  $\mathbf{X}_t$  for any  $t > 0$ . To overcome this, the conditioning on the  
 185 measurements is usually done in one of two ways. (i) A learned component models the conditional  
 186 posterior score and also gets the measurements as input. This strategy is (pursued by, e.g., Liu et al.  
 187 (2023); Özdenizci & Legenstein (2023); Bogensperger et al. (2025); Saharia et al. (2023)), and is  
 188 advantageous when the measurement process is unknown, difficult to model, or prohibitively expen-  
 189 sive to evaluate. However, its reconstructions obtained by this strategy typically degrade under shifts  
 190 in measurement conditions, since the learned components cannot adapt to the new measurement  
 191 conditions. (ii) The Bayesian separation that is described in (8) is pursued and the likelihood score is  
 192 approximated. This strategy (pursued by is done by, e.g., Chung & Ye (2022); Xue et al. (2025) and the  
 193 methods reviewed by Lemercier et al. (2024)), and is advantageous when the measurement process  
 194 is known, relatively inexpensive to evaluate, and subject to change, but prior knowledge should be  
 195 reused, which is frequently the case in, e.g., imaging or remote-sensing applications. However, this  
 196 requires approximations to the likelihood score  $\nabla (\mathbf{x} \mapsto \log p_{\mathbf{Y} | \mathbf{X}_t = \mathbf{x}}(\mathbf{y}))$  for all  $t > 0$ .

197 Our benchmark can evaluate either strategy, as well as (and) any other method that would claims to  
 198 sample from a posterior distribution like in (2)). Approach (i) The first approach, however, relies  
 199 on black-box learning of the conditional posterior score and its performance heavily depends on  
 200 various implementation details. Thus, we primarily focus on approach (ii) the second approach, which  
 201 that necessitates approximations of the likelihood score (and more general DPS algorithms with  
 202 explicit conditioning, see our proposed generalization in Section 3). For those, our framework can  
 203 supply arbitrary-precision Monte Carlo estimates of various reference objects — posterior samples  
 204 and arbitrary-precision oracle denoisers via Gibbs methods — to isolate and quantify the impact of  
 205 these approximations.

### 206 3 PROPOSED FRAMEWORK

207 The prior distributions in our framework will be that of signals of length  $d$  obtained by regularly  
 208 spaced samples of processes with independent, stationary increments (Lévy processes, described in  
 209 Appendix C). Let  $s$  be such a process and let the unit-step increments be  $[\mathbf{u}]_k = (s(k) - s(k-1))$   
 210 for  $k = 1, 2, \dots, d$ . Independence and stationarity imply that the distribution of  $[\mathbf{u}]_k$ , the increment  
 211 distribution  $p_U$ , does not depend on  $k$ . The increment vector is related to the signal  $\mathbf{x}$  via  $\mathbf{u} = \mathbf{D}\mathbf{x}$ ,  
 212 where  $\mathbf{D}$  is a finite-difference matrix with an initial condition that allows us to write  $\mathbf{x} = \mathbf{D}^{-1}\mathbf{u}$ ,  
 213 where  $\mathbf{D}^{-1}$  is a lower-triangular matrix of ones. This gives a convenient way to synthesize signals  
 214 once  $\mathbf{u}$  is drawn. The independence of the increments implies that the density of the discrete signal is

$$p_{\mathbf{x}}(\mathbf{x}) = \prod_{k=1}^d p_U([\mathbf{D}\mathbf{x}]_k) \quad (9)$$

[bQ8j] Improvement of the structure.

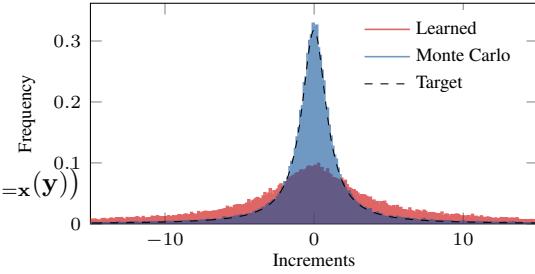


Figure 2: Histogram of increments of signals obtained by DDPM with different denoisers.

[tkeZ] Clarification on the type of algorithms that can be evaluated.

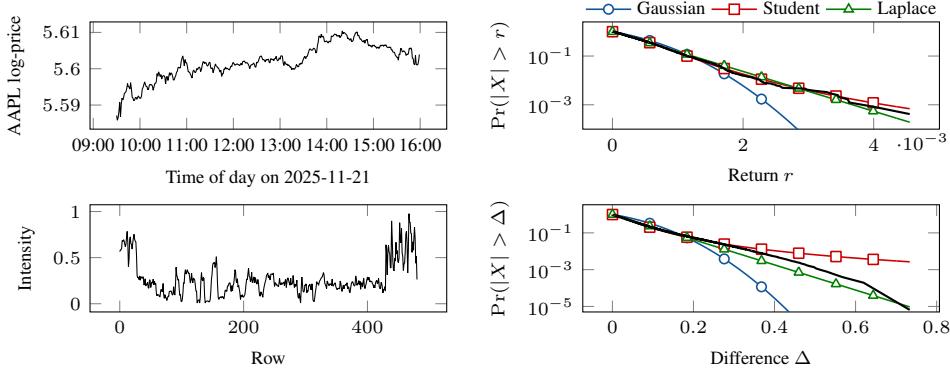


Figure 3: Examples of signals with heavy-tailed increment distributions. Top: Asset returns. Bottom: Columns in natural images. Left: Signals. Right: Survival function of absolute increments (no marker: empirical; markers: best fit to empirical within distribution).

215 We consider four increment distributions that are commonly used in sparse-process models: Gaussian;  
 216 Laplace;  
 217 Student-t;  
 218 and Bernoulli–Laplace (spike-and-slab). Such increment distributions are *sparse* or *heavy-tailed* according to the taxonomy due to Unser & Tafti (2014) in (Unser & Tafti, 2014) and are relevant in signal and image processing, finance, and other fields (Schoutens, 2003). We show instances signals with such heavy-tailed increment distributions in Figure 3. A precise definition of Lévy processes, the matrix  $\mathbf{D}$ , the increment distributions and their notation along with a discussion about extensions to higher-dimensional signals or signals with more complicated graph structure are given in Appendix C.

223 **Efficient Posterior Sampling** With the prior distribution specified in (9) and the assumption of 224 Gaussian noise, the posterior associated to the inverse problem intrinsic in (1) is

$$p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) \propto \exp\left(-\frac{1}{2\sigma_n^2}\|\mathbf{Ax} - \mathbf{y}\|^2\right) p_{\mathbf{X}}(\mathbf{x}) = \exp\left(-\frac{1}{2\sigma_n^2}\|\mathbf{Ax} - \mathbf{y}\|^2\right) \prod_{k=1}^d p_U([\mathbf{Dx}]_k [\mathbf{Dx}]_k). \quad (10)$$

225 Unless  $p_U$  is a Gaussian (the simplified setting in Pierret & Galerne (2025b)), this posterior is not 226 conjugate, so neither closed-form sampling nor direct evaluation of moments is available. Nevertheless, 227 for the increment distributions used in this paper, the posterior distributions admit efficient Gibbs 228 methods via standard latent-variable augmentations. Motivation and more details about the Gibbs 229 methods, such as the burn-in period  $B$  and the number of samples  $S$ , are provided in Appendix D.

230 The Gaussian, Laplace, and Student-t distributions admit latent representations 231 as infinite-component Gaussian mixtures, 232 which makes them suitable for the Gaussian latent machine (GLM) framework 233 that was recently introduced by Kuric et al. 234 (2025). If The GLM framework is generally 235 applicable to distributions of the form 236

$$p(\mathbf{x}) \propto \prod_{k=1}^n \phi_k([\mathbf{Kx}]_k [\mathbf{Kx}]_k), \quad (11)$$

238 where  $\mathbf{K} \in \mathbb{R}^{n \times d}$  and all distributions 239  $\phi_1, \phi_2, \dots, \phi_n : \mathbb{R} \rightarrow \mathbb{R}$  have a latent 240 representation

$$\phi_k(t) = \int_{\mathbb{R}} g_{\mu_k(z), \sigma_k^2(z)}(t) f_k(z) dz, \quad (12)$$

241 where the *latent distribution*  $f_{\mathbf{kz}}$  and the *latent maps*  $\mu_{\mathbf{kz}}, \sigma_{\mathbf{kz}}^2 : \mathbb{R} \rightarrow \mathbb{R}$  depend on the distribution 242  $\phi_{\mathbf{kz}}$ , and  $g_{\mu, \sigma^2}$  is the density of a one-dimensional Gaussian distribution with mean  $\mu$  and variance

[TmEt pM9c] Addition of examples of real signals with heavy-tailed increment distributions.

[TmEt pM9c] Addition of examples of real signals with heavy-tailed increment distribution.

---

**Algorithm 1** GLM Gibbs method.

---

**Require:**  $\mathbf{x}_0 \in \mathbb{R}^d$ ,  $\mathbf{K} \in \mathbb{R}^{n \times d}$ , conditional latent distributions  $\{p_{[\mathbf{Z}]_k [\mathbf{Z}]_k | X} \}_{k=1}^n$  and maps  $\{\mu_{\mathbf{kz}}, \sigma_{\mathbf{kz}}^2\}_{k=1}^n$

1: **for**  $s = 1, \dots, B + S$  **do**  
 2:   Draw  $[\mathbf{Z}]_k [\mathbf{Z}]_k \sim p_{[\mathbf{Z}]_k [\mathbf{Z}]_k | X = [\mathbf{Kx}_{s-1}]_k [\mathbf{Kx}_{s-1}]_k} \triangleright \text{par.}$   
    **allel over**  $\mathbf{kz}$   
 3:   Draw  $\mathbf{x}_s \sim \text{Gauss}(\mu(\mathbf{z}), \Sigma(\mathbf{z}))$   
 4: **return**  $\{\mathbf{x}_{B+s}\}_{s=1}^S$

---

243  $\sigma^2$ . We can cast the posterior distribution in (10) ~~can be cast~~ into this framework by rewriting it as

$$p_{\mathbf{x}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) \propto \prod_{k=1}^m g_{[\mathbf{y}]_k \mathbf{y}_k, \sigma_n^2}([\mathbf{Ax}]_k (\mathbf{Ax})_k) \prod_{k=1}^d p_U([\mathbf{Dx}]_k (\mathbf{Dx})_k) = \prod_{k=1}^{m+d} \phi_k([\mathbf{Kx}]_k (\mathbf{Kx})_k). \quad (13)$$

244 There, by setting  $\mathbf{K} = [\mathbf{A}; \mathbf{D}]$ ,  $\phi_k = g_{[\mathbf{y}]_k \mathbf{y}_k, \sigma_n^2}$  for  $k = 1, 2, \dots, m$ , and  $\phi_k = p_U$  for  $k =$   
 245  $m+1, m+2, \dots, m+d$ . We summarize the GLM sampling in algorithm 1. Importantly, non-  
 246 Gaussian likelihoods can be handled by some appropriate definition of adapting the first  $m$  distributions ~~appropriately~~.

247 The introduction of an appropriate  $n$ -dimensional random variable  $\mathbf{Z}$  with non-trivial distribution  
 248 (see the details in Kuric et al. (2025)) enables the efficient sampling from the conditionals: Sampling  
 249  $\mathbf{X} | \mathbf{Z} = \mathbf{z}$  amounts to sampling a Gaussian with covariance and mean

$$\Sigma(\mathbf{z}) = (\mathbf{K}^T \Sigma_0(\mathbf{z})^{-1} \mathbf{K})^{-1} \text{ and } \mu(\mathbf{z}) = \Sigma(\mathbf{z}) \mathbf{K}^T \Sigma_0(\mathbf{z})^{-1} \mu_0(\mathbf{z}), \quad (14)$$

250 respectively, where  $\Sigma_0(\mathbf{z}) = \text{diag}(\sigma_1^2([\mathbf{z}]_1 \mathbf{z}_1), \dots, \sigma_n^2([\mathbf{z}]_n \mathbf{z}_n))$  and  $\mu_0(\mathbf{z}) =$   
 251  $(\mu_1([\mathbf{z}]_1 \mathbf{z}_1), \dots, \mu_n([\mathbf{z}]_n \mathbf{z}_n))$ . Sampling  $\mathbf{Z} | \mathbf{X} = \mathbf{x}$  amounts to sampling  $n$  independent  
 252 one-dimensional *conditional latent distributions*  $p_{[\mathbf{z}]_1 \mathbf{z}_1 | \mathbf{X} = [\mathbf{Kx}]_1 (\mathbf{Kx})_1}, \dots, p_{[\mathbf{z}]_n \mathbf{z}_n | \mathbf{X} = [\mathbf{Kx}]_n (\mathbf{Kx})_n}$   
 253 that depend on the distributions  $\phi_1, \dots, \phi_n$  and are given in Table 3 in the appendix along with the  
 254 corresponding latent distributions and latent maps. We summarize the GLM sampling in Algorithm 1.

255 For the Bernoulli–Laplace *increment* distribution, we adapt the algorithm proposed by Bohra et al.  
 256 (2023) that introduces two  $d$ -dimensional latent variables: a Bernoulli indicator (“on”/“off”); and  
 257 a Laplace-distributed increment height. For a self-contained exposition, we rigorously derive the  
 258 resulting Gibbs method in Appendix D.1.

260 The Gibbs methods that we just described are  
 261 suitable for the generation of the gold-standard  
 262 samples from the posterior that corresponds to  
 263 the initial inverse problem *intrinsic* in (1) as  
 264 well as the generation of samples from the de-  
 265 noising posteriors in the DPS algorithms. In the  
 266 latter case, the forward operator  $\mathbf{A}$  is the iden-  
 267 tity, the measurements are the noisy interme-  
 268 diate reconstructions  $\mathbf{x}_t$ , and the noise variance  
 269  $\sigma_n^2 = \sigma_t^2$  follows the schedule at timestep  $t$ .

270 When these Gibbs methods are embedded  
 271 within the reverse-diffusion loop, an efficient  
 272 implementation is paramount to achieve accept-  
 273 able runtimes. This is ~~most crucial especially~~  
 274 ~~true for the Bernoulli–Laplace increment distributions~~, where the sequential drawing of the binary  
 275 support vector is ~~nested within the outer Gibbs loop~~, which in turn may be nested  
 276 within the reverse-diffusion loop. Accordingly, we ~~deliberately~~ tailored our implementation—~~which~~  
 277 ~~we regard as a substantial contribution~~—to modern, highly parallel compute units and optimized sev-  
 278 eral components, including custom CUDA- and Triton-compiled sampling routines and incremental  
 279 updates based on the Woodbury–Sherman–Morrison identities (see Appendix D.2). We achieved a  
 280 ~~cumulative speedup of  $74.61 \times$  over the baseline implementation~~ (illustrated in Figure 4 with details  
 281 in Appendix D.2).

282 **A Generalized DPS Template** Widely used methods, such as diffusion plug-and-play (DPnP) (Xu  
 283 & Chi, 2024), fall outside the pattern described in Section 2, where one approximates—~~approximating~~  
 284 ~~the~~ likelihood score inside the reverse diffusion. We therefore introduce a simple template that is  
 285 natural in our setting and accommodates a broader set of DPS algorithms. More precisely, we  
 286 characterize the iteration rule of DPS algorithms as ~~an iteration rule that can be summarized into~~  
 287 a two-stage process: Given an iterate  $\mathbf{x}_t$  with associated noise variance  $\sigma_t^2$ , the ~~computation of~~  
 288 ~~the~~ next iterate  $\mathbf{x}_{t-1}$  is ~~computed~~ ~~done~~ by (i) drawing  $S$  samples denoted  $\{\bar{\mathbf{x}}_{ks}\}_{ks=1}^S$  from the

**[tkeZ]** Clarification of the derivation of (13) through the introduction of new “indexing” notation.

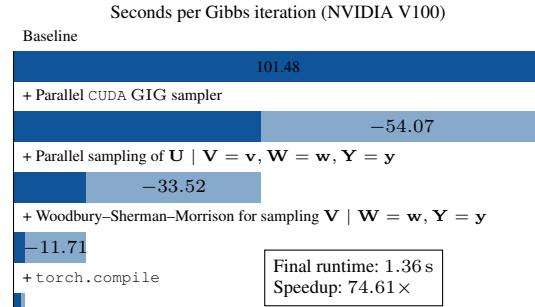


Figure 4: Runtime improvements of the Bernoulli–Laplace sampler.

**[pM9c]** Quantification of the speedup that we achieved with our implementation.

**[pM9c]** Quantification of the speedup that we achieved with our implementation.

289 denoising posterior  $p_{\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}_t} \propto \exp\left(-\frac{1}{2\sigma_t^2}\|\cdot - \mathbf{x}_t\|^2\right)p_{\mathbf{X}_0}(\cdot)$ ; and (ii) the subsequent computation of computing the next iterate  $\mathbf{x}_{t-1}$  through an update step  $\mathcal{S}$  that may utilize the current iterate  $\mathbf{x}_t$ , the samples  $\{\bar{\mathbf{x}}_{k_S}\}_{k_S=1}^S$ , the measurements  $\mathbf{y}$ , the forward operator  $\mathbf{A}$ , and, possibly, other algorithm-internal parameters such as a scalar that weights likelihood and prior terms or parameters that define the noise schedule. This template is summarized in [Algorithm 2](#) and specialized instances for the update step  $\mathcal{S}$  for a variety of popular algorithms that correspond to the three popular algorithms Chung diffusion posterior sampling (C-DPS) ([Chung et al., 2023](#)), diffusion models for plug-and-play image restoration (DiffPIR) ([Zhu et al., 2023](#)), and DPnP ([Zhu et al., 2023](#)) are given in [Appendix E.2](#). We have absorbed the (variance-preserving) scaling into the step  $\mathcal{S}$  since this template is not fundamentally limited to diffusion processes but supports any (also not monotonically decreasing) noise schedules. In addition, noise variances  $\{\sigma_t\}_{t=1}^T$  are usually derived from the algorithm-internal parameters  $\lambda$  that may include a noise schedule.

301 Through this construction, DPS algorithms can  
 302 use any statistic  $R$  of the samples  $\{\bar{\mathbf{x}}_{k_S}\}_{k_S=1}^S$   
 303 in their update steps. Most methods use the  
 304 mean  $R(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_S) = \frac{1}{S} \sum_{k_S=1}^S \bar{\mathbf{x}}_{k_S} := \bar{\mu}$ ,  
 305 which is the Monte Carlo estimate of  $\mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t]$ . An example of a DPS algorithm  
 306 that utilizes additional statistics is C-DPS,  
 307 which requires the Jacobian of  $\mathbf{x}_t \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t]$ . As we show in [Appendix E.1](#), this  
 308 Jacobian equals (up to the known variance-  
 309 preserving scaling) the conditional covariance  
 310 of  $\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t$ , an unbiased estimator of which can be obtained through the statistic  
 311 of  $\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t$ , an unbiased estimator of which can be obtained through the statistic  
 312 of  $\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t$ , an unbiased estimator of which can be obtained through the statistic  
 313  $R(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_S) = \frac{1}{S-1} \sum_{k_S=1}^S (\bar{\mathbf{x}}_{k_S} - \bar{\mu})(\bar{\mathbf{x}}_{k_S} - \bar{\mu})^T$ . An example of a DPS algorithm that util-  
 314 izes an alternative statistic is the DPnP algorithm that alternately samples from  $p_{\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}_t}$  and a  
 315 data-proximal problem. There,  $R(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_S) = \bar{\mathbf{x}}_1$  is used to obtain one sample from  $p_{\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}_t}$ .  
 316 This statistic is frequently used in the asymptotically exact and the CSGM-type algorithms (using the  
 317 taxonomy due to [Daras et al. \(2024\)](#)). When only a learned MMSE denoiser is available, obtaining this  
 318 one sample requires a full reverse diffusion. In contrast, it requires only one iteration (and the burn-in  
 319 period) with the Gibbs methods. Thus, these algorithms are typically faster when they are endowed  
 320 with the Gibbs methods (see the runtimes in [Tables 5 and 6](#)), which enables easy benchmarking.  
 321 However, CSGM-type algorithms typically do not aim at posterior sampling and we do not benchmark  
 322 them here.

323 Since the denoising posteriors are always sub-Gaussian, the Monte Carlo estimation of any object  
 324 enjoys favorable convergence. For instance, the computational complexity of estimating the covariance  
 325 up to a desired precision in the operator norm scales linearly with the dimensionality of the signal  
 326 ([Vershynin, 2018](#), Theorem 4.7.1).

327 **Extensions** A prerequisite for a quantitative evaluation of posterior-sampling algorithms is the  
 328 availability of reasonably efficient samplers that can provide gold-standard samples. The development  
 329 of such samplers for posteriors arising from nonlinear measurement models and non-Gaussian noise  
 330 is challenging, and existing methods currently address only specific cases (e.g., [Wang et al. \(2017\)](#)  
 331 study a nonlinear-Gaussian measurement model with a Laplace prior). Importantly, our framework  
 332 is modular: as more general-purpose samplers for these posteriors become available, they can be  
 333 plugged into our benchmark directly. The denoising posteriors in the reverse diffusion do not change  
 334 with the likelihood and can, therefore, always be efficiently sampled.

335 When going to higher dimensions, the primary challenge lies in the sampling of the high-dimensional  
 336 Gaussian distributions required in the Gibbs methods. Luckily, the structure of the involved operators  
 337 in our case is such that the Gaussians can be efficiently sampled with perturb-and-MAP approaches  
 338 with matrix-free conjugate gradient implementations; we discuss this in more detail and show how  
 339 the runtime of different samplers change with the dimensions in [Appendix D.2](#). Sampling high-  
 340 dimensional Gaussians is a well-studied problem and advances in that field can directly be used in  
 341 our framework.

**[tkeZ]** Showcase the broad applicability and generality of our template.

**Algorithm 2** Template for DPS algorithms.

**Require:** Initial point  $\mathbf{x}_T, \mathbf{y}, \mathbf{A}, \lambda$

```

1: for  $t = T, \dots, 1$  do ▷ Diffusion process
2:   Sample  $\{\bar{\mathbf{x}}_{k_S}\}_{k_S=1}^S \sim p_{\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}_t}$ 
3:   Update  $\mathbf{x}_{t-1} = \mathcal{S}(\mathbf{x}_t, \{\bar{\mathbf{x}}_{k_S}\}_{k_S=1}^S, \mathbf{y}, \mathbf{A}, \lambda, t)$ 
4: return  $\hat{\mathbf{x}}^{\text{alg}} = \mathbf{x}_0$  ▷ Posterior sample

```

**[tkeZ]** Significance of  $S = 1$  in DPnP and relation to optimization-based and CSGM-type methods.

**[tkeZ]** Computational complexity of covariance estimation.

**[tkeZ TmEt pM9c]** Extensions to nonlinear problems, higher dimensions, and other posterior solvers.

342 Our gold-standard posterior samples can be compared to samples obtained by *any* posterior-sampling  
 343 algorithm. This includes classical Markov-chain Monte Carlo algorithms, algorithms that utilize  
 344 flow-matching priors, and others. In this work, we primarily focus on DPS algorithms because  
 345 our framework can supply arbitrary-precision Monte Carlo objects to them. We believe that this  
 346 fundamental principle can be extended to other algorithms, in particular those that utilize flow-  
 347 matching priors. Such algorithms are frequently evaluated on toy examples based on Gaussian  
 348 mixtures (e.g. by Pourya et al. (2025)), that are overly simplistic.

## 349 4 NUMERICAL EXPERIMENTS

350 We consider signals of dimension  $d = 64$  and four inverse problems that are frequently encountered in  
 351 various estimation tasks throughout the natural sciences: denoising, deconvolution, imputation, and  
 352 reconstruction from partial Fourier measurements. The dimension of the signal is large enough such  
 353 that the corresponding operators can be sensibly defined, yet small enough such that the benchmark has  
 354 acceptable runtimes. We provide experiments about the runtime with larger signals in Appendix D.2,  
 355 details about the operators are provided in Appendix F.1, and precise descriptions of the  
 356 benchmarking pipeline (e.g., the number of training, validation, and test signals, and the number of  
 357 iterations in the Gibbs methods) in Appendix F.2.

[pM9c] Information and motivation about the dimensionality of our setup.

### 358 4.1 RECONSTRUCTION ALGORITHMS

359 The model-based methods and the DPS algorithms require the tuning of some hyperparameters.  
 360 These were found by grid search on validation data independently for each algorithm, increment  
 361 distribution, and forward operator. The precise setup for this grid search is given in Appendix F.5.  
 362 Importantly, the hyperparameters for the DPS algorithms were tuned to the learned denoiser due to  
 363 resource constraints and we view full oracle-tuning as a community task. Parameters obtained with  
 364 this procedure are later denoted with a star in the superscript.

365 **Model-Based Methods** As baseline reconstruction algorithms we consider the model-based  
 366 methods

$$\hat{\mathbf{x}}^{\ell_2}(\mathbf{y}, \lambda) = \arg \min_{\mathbf{x} \in \mathbb{R}^d} \left( \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|^2 + \lambda \|\mathbf{Dx}\|^2 \right), \quad (15)$$

367 and

$$\hat{\mathbf{x}}^{\ell_1}(\mathbf{y}, \lambda) = \arg \min_{\mathbf{x} \in \mathbb{R}^d} \left( \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|^2 + \lambda \|\mathbf{Dx}\|_1 \right), \quad (16)$$

368 as baseline reconstruction algorithms. They which coincide with the maximum-a-posteriori (MAP)  
 369 estimators of Lévy processes associated with Gaussian and Laplace increment distributions, respectively.  
 370

371 **Diffusion Posterior Sampling Algorithms** We consider three DPS algorithms that are popular  
 372 in the literature. First, the C-DPS algorithm due to (Chung et al., 2023), which was one of the first  
 373 algorithms that was proposed for the resolution of general noisy inverse problems with diffusion  
 374 priors. Second, the DiffPIR, algorithm due to (Zhu et al., 2023) that can be regarded as an extension of  
 375 the C-DPS algorithm and typically reports superior results in standard perception-based evaluations.  
 376 and Third, the DPnP algorithm due to (Xu & Chi, 2024) that alternates between sampling the  
 377 denoising subproblem and a data proximal subproblem. This selection demonstrates the applica-  
 378 bility of the framework to algorithms that require denoising-posterior samples (DPnP), the MMSE  
 379 denoiser (DiffPIR), and its Jacobian (C-DPS), which covers most of the existing DPS algorithms. We  
 380 include the DPnP algorithm to showcase the broad applicability of our framework to nonstandard  
 381 setups that utilize various statistics of the denoising posterior. For each DPS algorithm, we benchmark  
 382 a variant that uses learned components (learning details are provided in Appendix F.3) and a variant  
 383 that uses Gibbs samples of the denoising posterior. For DPnP, this fully removes approximation  
 384 errors. For the others, the learned components and the Monte Carlo estimates of those components  
 385 have varying quality for different distributions and noise variances that we systematically investigate  
 386 in Appendix F.4. We provide our main results, the MMSE optimality gap, for the learned variant  
 387 and then investigate changes when we substitute the Gibbs samples for the learned components.  
 388 two variants: One where the denoising posterior is sampled with the gold-standard Gibbs methods  
 389 (oracle denoiser) and statistics are computed from those samples, and one where the sampling (or the

[tkeZ] Removed ambiguous description of the relation between C-DPS and DiffPIR.

[tkeZ TmEt] Explanation on the isolation of algorithmic errors; clarification of our experimental setup; MCMC terminology.

390 direct estimation of any point estimate) is done with learned components. For the former, we ensured  
 391 that the Monte Carlo error (in the estimation of the denoising-posterior expectation) is significantly  
 392 below the error of the learned denoisers with a rigorous protocol that is described in Appendix F.4.  
 393 For the latter, learning details are provided in Appendix F.3.

394 The model-based methods and the DPS algorithms require the tuning of some hyperparameters.  
 395 These were found by grid search on validation data independently for each algorithm, increment  
 396 distribution, and forward operator. The precise setup for this grid search is given in Appendix F.5.  
 397 The hyperparameters for the DPS algorithms were tuned to the learned denoiser. Parameters obtained  
 398 with this procedure are later denoted with a star in the superscript.

399 **Gold-Standard Gibbs Methods** The Gibbs methods are used to obtain gold-standard samples  
 400 from the posterior. As described in Section 3, the Gibbs methods are parameter- and bias-free and  
 401 efficient. Consequently, they are well-suited for our this purpose. Chain lengths, diagnostics,  
 402 and implementation details are given in Appendix F.2; we reuse the same settings across operators  
 403 and increment families.

404 4.2 RESULTS

405 **Before advancing, we introduce some notation.** For any given measurement  $\mathbf{y}$ , some any DPS  
 406 algorithm  $\text{alg}$  that depends on the any parameters  $\boldsymbol{\lambda}$  will produce samples that we denote  
 407  $\{\hat{\mathbf{x}}_k^{\text{alg}}(\mathbf{y}, \boldsymbol{\lambda})\}_{k=1}^{N_{\text{samples}}}$ . We moreover denote  $\hat{\mathbf{x}}_{\text{MMSE}}^{\text{alg}}(\mathbf{y}, \boldsymbol{\lambda}) := \frac{1}{N_{\text{samples}}} \sum_{k=1}^{N_{\text{samples}}} \hat{\mathbf{x}}_k^{\text{alg}}(\mathbf{y}, \boldsymbol{\lambda})$ . For an  
 408 estimation method  $\hat{\mathbf{x}}^{\text{est}}(\cdot)$  and data  $\mathbf{y}$  with corresponding data-generating signal  $\mathbf{x}$  we measure the  
 409 MMSE optimality gap (in decibel) defined by

$$10 \log_{10} \left( \frac{\|\hat{\mathbf{x}}^{\text{est}}(\mathbf{y}) - \mathbf{x}\|^2}{\|\hat{\mathbf{x}}_{\text{MMSE}}^{\text{alg}}(\mathbf{y}) - \mathbf{x}\|^2} \right), \quad (17)$$

410 where  $\hat{\mathbf{x}}^{\text{est}}(\mathbf{y}) = \hat{\mathbf{x}}^{\ell_1 \vee \ell_2}(\mathbf{y}, \boldsymbol{\lambda}^*)$  for model-based methods and  $\hat{\mathbf{x}}^{\text{est}}(\mathbf{y}) = \hat{\mathbf{x}}_{\text{MMSE}}^{\text{alg}}(\mathbf{y}, \boldsymbol{\lambda}^*)$  for DPS  
 411 algorithms. A gap of 0 indicates a perfect recovery of the gold-standard MMSE estimate and any the  
 412 positive nonzero values show the orders of magnitude of the error relative to the reference error. We  
 413 found that  $N_{\text{samples}} = 50$  provided a good tradeoff between runtime and accuracy by benchmarking  
 414 the gold-standard Gibbs method with that number of samples.

415 We report in Table 1 the mean and standard deviation of the MMSE optimality gap over all signal-  
 416 measurement pairs  $(\mathbf{x}, \mathbf{y})$  in the test set obtained by the model-based methods and the DPS algorithms  
 417 endowed with the learned denoiser in Table 1. The Gaussian increment distribution validates the  
 418 implementation: Since the MMSE and the MAP point estimates coincide, the model-based  $\ell_2$  esti-  
 419 mator matches the Gibbs reference up to the error due to the finite parameter-grid resolution. When  
 420 the posterior mean is smooth (e.g., imputation and some deconvolution cases),  $\ell_2$  is the best model-  
 421 based choice and frequently outperforms the DPS algorithms. When the posterior mean is close  
 422 to piecewise-constant (typical in denoising of signals with sparse increments), the  $\ell_1$  estimator is  
 423 preferred. Among DPS algorithms, DiffPIR is typically the top performer and often exceeds  $\ell_2$   
 424 and  $\ell_1$  baselines in deconvolution, imputation, and reconstruction from partial Fourier measure-  
 425 ments. For spike-and-slab settings (Bernoulli–Laplace), DPS algorithms substantially outperform the  
 426 model-based baselines across operators. In deconvolution and reconstruction from partial Fourier  
 427 measurements, DPS algorithms frequently match or surpass the best model-based estimator.

428 We now inspect the change in performance after we substitute higher-quality Monte Carlo com-  
 429 ponents for the learned components. We do this without retuning of the hyperparameters, which  
 430 allows us to see if the performance of the algorithms increases automatically with the quality of  
 431 the denoiser. In addition to the reconstruction performance obtained with the learned denoisers—for  
 432 which the parameters of the algorithms were tuned—we inspect the robustness of the algorithms  
 433 when replacing the learned denoiser with the oracle denoiser. Here, we discuss general trends; an  
 434 exhaustive quantitative evaluation and a precise quantification of the quality of the learned and Monte  
 435 Carlo objects is given in Appendix G. For the same hyperparameters, the performance of DPnP  
 436 increases significantly with the quality of the denoising-posterior samples. For example, the optimality  
 437 gap decreases by 10.46 dB for imputation of signals with  $\text{St}(1)$  increments, and significantly for  
 438 other measurement operators for signals with  $\text{St}(1)$  increments and  $\text{BL}(0.1, 1)$  increments. DPnP  
 439 is the most robust to swapping the learned denoiser with the arbitrary precision oracle denoiser and

[tkeZ TmEt] Clarification of our experimental setup; MCMC terminology.

Table 1: MMSE optimality gap in decibel (mean  $\pm$  standard deviation; lower is better; 0 is a perfect reconstruction) of various estimation methods over the test set. **Bold**: best among DPS algorithms.

		Gauss(0, 0.25)	Laplace(1)	BL(0.1, 1)	St(1)	St(2)	St(3)
Denoising	C-DPS	<b>0.12 <math>\pm</math> 0.18</b>	0.12 $\pm$ 0.20	2.22 $\pm$ 2.26	3.26 $\pm$ 1.01	0.28 $\pm$ 0.30	0.10 $\pm$ 0.18
	DiffPIR	0.16 $\pm$ 0.21	<b>0.09 <math>\pm</math> 0.16</b>	<b>0.72 <math>\pm</math> 1.10</b>	<b>0.93 <math>\pm</math> 1.06</b>	<b>0.07 <math>\pm</math> 0.14</b>	0.15 $\pm$ 0.21
	DPnP	0.24 $\pm$ 0.25	0.11 $\pm$ 0.17	1.33 $\pm$ 2.12	1.19 $\pm$ 1.38	0.10 $\pm$ 0.17	<b>0.10 <math>\pm</math> 0.17</b>
	$\ell_1$	0.15 $\pm$ 0.21	0.06 $\pm$ 0.12	3.44 $\pm$ 2.38	0.38 $\pm$ 0.43	0.14 $\pm$ 0.19	0.11 $\pm$ 0.18
	$\ell_2$	0.00 $\pm$ 0.01	0.16 $\pm$ 0.21	8.61 $\pm$ 3.10	3.25 $\pm$ 0.99	0.74 $\pm$ 0.83	0.25 $\pm$ 0.33
Deconvolution	C-DPS	0.12 $\pm$ 0.20	0.12 $\pm$ 0.23	4.30 $\pm$ 3.87	18.30 $\pm$ 5.28	0.46 $\pm$ 1.40	0.17 $\pm$ 0.53
	DiffPIR	<b>0.07 <math>\pm</math> 0.17</b>	<b>0.07 <math>\pm</math> 0.19</b>	<b>1.09 <math>\pm</math> 2.22</b>	10.45 $\pm$ 6.10	<b>0.09 <math>\pm</math> 0.57</b>	<b>0.08 <math>\pm</math> 0.26</b>
	DPnP	0.10 $\pm$ 0.18	0.13 $\pm$ 0.22	1.71 $\pm$ 2.49	<b>7.84 <math>\pm</math> 5.66</b>	0.35 $\pm$ 1.39	0.14 $\pm$ 0.41
	$\ell_1$	1.65 $\pm$ 0.84	1.38 $\pm$ 0.86	1.86 $\pm$ 3.14	1.87 $\pm$ 4.01	1.10 $\pm$ 1.19	1.28 $\pm$ 0.94
	$\ell_2$	0.00 $\pm$ 0.01	0.07 $\pm$ 0.23	6.11 $\pm$ 4.49	21.50 $\pm$ 4.46	1.44 $\pm$ 2.85	0.36 $\pm$ 1.09
Imputation	C-DPS	0.15 $\pm$ 0.29	0.18 $\pm$ 0.39	2.99 $\pm$ 2.82	23.33 $\pm$ 8.69	0.50 $\pm$ 1.09	0.14 $\pm$ 0.57
	DiffPIR	<b>0.09 <math>\pm</math> 0.23</b>	<b>0.08 <math>\pm</math> 0.24</b>	<b>0.24 <math>\pm</math> 1.14</b>	<b>0.88 <math>\pm</math> 3.50</b>	<b>0.11 <math>\pm</math> 0.62</b>	<b>0.08 <math>\pm</math> 0.42</b>
	DPnP	0.14 $\pm$ 0.32	0.17 $\pm$ 0.36	0.50 $\pm$ 1.28	10.89 $\pm$ 5.92	0.25 $\pm$ 0.82	0.27 $\pm$ 0.58
	$\ell_1$	1.74 $\pm$ 1.12	1.77 $\pm$ 1.35	1.25 $\pm$ 2.78	13.32 $\pm$ 5.32	1.37 $\pm$ 2.56	1.55 $\pm$ 1.58
	$\ell_2$	0.00 $\pm$ 0.01	0.01 $\pm$ 0.05	1.10 $\pm$ 1.88	0.42 $\pm$ 0.95	0.06 $\pm$ 0.34	0.02 $\pm$ 0.28
Fourier	C-DPS	0.15 $\pm$ 0.36	0.26 $\pm$ 0.65	5.90 $\pm$ 4.41	4.29 $\pm$ 5.78	0.53 $\pm$ 0.83	0.35 $\pm$ 0.77
	DiffPIR	<b>0.11 <math>\pm</math> 0.29</b>	<b>0.08 <math>\pm</math> 0.31</b>	<b>0.83 <math>\pm</math> 1.44</b>	3.19 $\pm$ 4.37	<b>0.11 <math>\pm</math> 0.39</b>	<b>0.12 <math>\pm</math> 0.37</b>
	DPnP	0.11 $\pm$ 0.35	0.20 $\pm$ 0.51	1.88 $\pm$ 2.47	<b>2.45 <math>\pm</math> 4.83</b>	0.39 $\pm$ 0.89	0.24 $\pm$ 0.64
	$\ell_1$	1.50 $\pm$ 1.59	0.73 $\pm$ 0.94	3.57 $\pm$ 2.82	1.07 $\pm$ 2.98	0.71 $\pm$ 0.99	0.78 $\pm$ 0.97
	$\ell_2$	0.00 $\pm$ 0.02	0.36 $\pm$ 0.73	12.22 $\pm$ 4.53	9.47 $\pm$ 8.34	2.66 $\pm$ 3.57	1.03 $\pm$ 1.79

440 significantly benefits from the arbitrary-precision oracle denoiser in the most challenging cases of the  
441 spike-and-slab and the extremely heavy-tailed St(1) increment distributions. By contrast, C-DPS  
442 and DiffPIR can require a retuning when the denoiser changes: Scores can deteriorate after one has  
443 substituted a higher-quality Monte Carlo denoiser for the learned one replacing the learned denoiser  
444 with the oracle denoiser, but whereas a brief hand-tuning of the hyperparameters on the validation set  
445 improves them way beyond the learned denoiser. For instance, for DiffPIR and imputation of signals  
446 with St(1) increments, reusing the hyperparameters deteriorates the gap by 13.56 dB, whereas a  
447 brief hand-tuning decreased the optimality gap by almost 10 dB over what is reported in Table 1  
448 (e.g., for DiffPIR and St(1) increments, the hand-tuning decreased the optimality gap by almost 10  
449 dB). The differences between the algorithms are generally greater than the differences between the  
450 learned and oracle variants except for the heavy-tailed cases, which confirms the findings in (Bohra  
451 et al., 2023) and indicates that the research of efficient and robust DPS algorithms is still crucial.  
452 Qualitative examples of the MMSE estimates and the marginal variances obtained by the DPS  
453 algorithms and the gold-standard Gibbs methods are shown in Figures 11 to 18 in the appendix.

454 Prototypical samples and the corresponding  
455 MMSE estimate obtained from a DPS algorithm  
456 (here DiffPIR for deconvolution of a signal with  
457 BL(0.1, 1) increments) are shown in Figure 5.  
458 (The full conditional reverse-diffusion trajectory,  
459 the data-generating signal, the measurements,  
460 and the MMSE estimated obtained with the gold-  
461 standard Gibbs methods are shown in Figure 19  
462 in the appendix.) The figure highlights a key dis-  
463 tinction: Posterior samples often preserve high-  
464 frequency structure and reflect prior variability,  
465 whereas the MMSE point estimate—obtained by  
466 averaging all samples—is much smoother. This  
467 explains why DPS methods tend to score higher on perception-oriented metrics, while regressors that  
468 target the MMSE point estimate (through training with the mean squared error) excel on distortion  
469 metrics like the peak signal-to-noise-ratio (PSNR). Consistent with this distinction, Saharia et al.  
470 (2023) fairly compare a sampling-based method to an MMSE regressor and find the expected trade-off:  
471 higher PSNR and structural similarity for the regressor; and better perceptual scores for the sampler.  
472 We therefore recommend to makemaking the Bayesian target explicit—point estimate versus sample  
473 quality—and to useusing evaluation protocols that are aligned to that target. Our framework supports  
474 this by offering gold-standard posterior samples and arbitrary-precision Monte Carlo estimatesoracle  
475 denoisers.

476 In addition to the evaluation of the MMSE optimality gap, which is on the point-estimator level, we  
477 analyze the highest-posterior-density coverage of the algorithms. Specifically, for any measurement

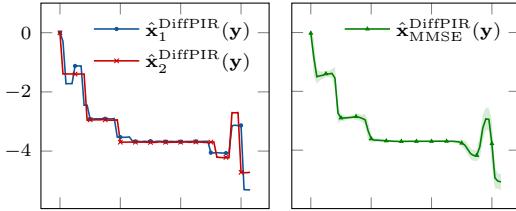


Figure 5: Conditional generation for deconvolution of a signal with BL(0.1, 1) increments with DiffPIR. The shaded area indicates the variance.

478  $\mathbf{y}$  and any  $k = 1, 2, \dots, N_{\text{samples}}$ , we defined denote<sup>4</sup>  $l_k(\mathbf{y}) := \log p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\hat{\mathbf{x}}_{P(k)}^{\text{alg}}(\mathbf{y}, \lambda^{\text{alg},*}))$  where  
 479  $P$  is the permutation that ensures that  $l_1(\mathbf{y}) \geq l_2(\mathbf{y}) \geq \dots \geq l_{N_{\text{samples}}}(\mathbf{y})$  and define the empirical  
 480 highest-posterior-density threshold at  $\alpha \in [0, 1]$  as  $l_{\lceil \alpha N_{\text{samples}} \rceil}(\mathbf{y})$ . We declare the data-generating  
 481 signal  $\mathbf{x}$  covered if  $\log p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) \geq l_{\lceil \alpha N_{\text{samples}} \rceil}(\mathbf{y})$  and define the coverage of a method as the  
 482 fraction of signal-measurement pairs  $(\mathbf{x}, \mathbf{y})$  in the test set for which  $\mathbf{x}$  is covered by the threshold  
 483  $l_{\lceil \alpha N_{\text{samples}} \rceil}(\mathbf{y})$ . The coverage of a calibrated posterior-sampling method will be  $\alpha$ , up to Monte Carlo  
 484 error. A coverage result that is significantly less than  $\alpha$  indicates that the samples obtained by the  
 485 method concentrate too heavily around the mode; a coverage result that is greater than  $\alpha$  indicates  
 486 that the samples are too spread out. We again discuss general trends here and present an exhaustive  
 487 quantitative evaluation in Appendix G. The coverages obtained by the DPS algorithms are generally  
 488 much smaller than  $\alpha$ , which indicates that they are uncalibrated and is in line with what is reported  
 489 by Thong et al. (2024). For C-DPS and DiffPIR, the reported coverage values are almost always 0  
 490 except for  $\text{BL}(0.1, 1)$  and  $\text{St}(1)$  increments, where the coverages are usually (close to) 1 for C-DPS  
 491 and inconsistent for DiffPIR. For almost all increment distributions and forward operators, DPnP  
 492 reports coverage values that are closest to but typically smaller than  $\alpha$ .

## 493 5 CONCLUSION

494 We have introduced a statistical benchmark for diffusion posterior sampling algorithms for linear  
 495 inverse problems. The framework proceeds by constructsing signals with a known distribution,  
 496 simulatesing the measurement process, and subsequently generatesing samples from the posterior  
 497 distribution that arises through the combination of the known prior and the known likelihood. Gold-  
 498 standard samples from this distribution are obtained via efficient Gibbs methods. T, and these  
 499 samples are then compared to those obtained by the diffusion posterior sampling algorithms. In  
 500 addition, the Gibbs methods can be used to obtain arbitrary-precision Monte Carlo estimates of  
 501 objects that are needed in the reverse stochastic differential equation, such as the minimum-mean-  
 502 squared-error denoiser or its Jacobian serve as oracle MMSE denoisers within the denoising posteriors  
 503 encountered in each iteration of the reverse SDE. Consequently, the framework also enables the  
 504 isolation and quantification of the error attributable to the likelihood approximations in the conditional  
 505 reverse diffusion. We have provided numerical results for three common diffusion posterior sampling  
 506 algorithms applied to four common inverse problems. A consistent theme across all tested algorithms  
 507 is that they are not calibrated, which demonstrates that research into algorithms that perform better in  
 508 this respect remains crucial. We invite other researchers to benchmark their algorithms on our open  
 509 implementation, which is deliberately designed so that novel DPS algorithms can be evaluated in a  
 510 plug-and-play manner.

[tkeZ] Clarification of terminology and phrasing regarding our “oracle” objects.

511 **Reproducibility Statement** We release an online repository with complete algorithm implemen-  
 512 tations and step-by-step instructions to reproduce all results. A containerized runtime enables one-  
 513 command setup and fully automated execution via the provided scripts. Each algorithm is specified  
 514 at a level that supports independent re-implementation: Tthe main text precisely details Gaussian  
 515 latent-machineGLM sampling,; and the appendix presents the Bernoulli–Laplace Gibbs method in  
 516 using implementation-aligned notation, together with practical optimizations required for accept-  
 517 able runtimes. The appendix also enumerates all experimental settings, including the numbers of  
 518 training/validation/test signals, the samples-per-datum for each sampler, and the exact grid-search  
 519 procedure used to select hyperparameters.

520 **Usage of Large Language Models** We used large language models to adapt passages of already-  
 521 written text for readability and conciseness.

## 522 REFERENCES

523 Amira Alakhdar, Barnabas Poczos, and Newell Washburn. Diffusion Models in De Novo Drug  
 524 Design. *Journal of Chemical Information and Modeling*, 64(19):7238–7256, September 2024.

<sup>4</sup>With some slight abuse of notation,  $\log p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}$  is the unnormalized ground-truth log-posterior (10). Since the additive constant is the same across all methods, this ranking is valid.

525 Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and Their*  
 526 *Applications*, 12(3):313–326, 1982.

527 Robert Bassett and Julio Deride. Maximum a posteriori estimators as a limit of Bayes estimators.  
 528 *Mathematical Programming*, 174(1–2):129–144, January 2018.

529 Robert C. Blattberg and Nicholas J. Gonedes. A Comparison of the Stable and Student Distributions  
 530 as Statistical Models for Stock Prices. *The Journal of Business*, 47(2):244–280, 1974.

531 Lea Bogensperger, Dominik Narnhofer, Ahmed Allam, Konrad Schindler, and Michael Krauthammer.  
 532 A variational perspective on generative protein fitness optimization. In *Forty-Second International*  
 533 *Conference on Machine Learning*, 2025.

534 Pakshal Bohra, Pol del Aguila Pla, Jean-François Giovannelli, and Michael Unser. A statistical  
 535 framework to investigate the optimality of signal-reconstruction methods. *IEEE Transactions on*  
 536 *Signal Processing*, 71:2043–2055, 2023.

537 Benjamin Boys, Mark Girolami, Jakiw Pidstrigach, Sebastian Reich, Alan Mosca, and Omer Deniz  
 538 Akyildiz. Tweedie moment projected diffusions for inverse problems. *Transactions on Machine*  
 539 *Learning Research*, 2024.

540 Gabriel Cardoso, Yazid Janati el idrissi, Sylvain Le Corff, and Eric Moulines. Monte carlo guided  
 541 denoising diffusion models for bayesian linear inverse problems. In *The Twelfth International*  
 542 *Conference on Learning Representations*, 2024.

543 George Casella and Edward I. George. Explaining the Gibbs sampler. *The American Statistician*, 46  
 544 (3):167–174, August 1992.

545 Hyungjin Chung and Jong Chul Ye. Score-based diffusion models for accelerated MRI. *Medical*  
 546 *Image Analysis*, 80:102479, 2022.

547 Hyungjin Chung, Jeongsol Kim, Michael Thompson McCann, Marc Louis Klasky, and Jong Chul Ye.  
 548 Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International*  
 549 *Conference on Learning Representations*, 2023.

550 Christian Clason, Tapio Helin, Remo Kretschmann, and Petteri Piiroinen. Generalized modes in  
 551 bayesian inverse problems. *SIAM/ASA Journal on Uncertainty Quantification*, 7(2):652–684, 2019.

552 Rama Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative*  
 553 *Finance*, 1(2):223–236, 2001.

554 Evan Scope Crafts and Umberto Villa. Benchmarking diffusion annealing-based Bayesian inverse  
 555 problem solvers. *IEEE Open Journal of Signal Processing*, 6:975–991, 2025.

556 Giannis Daras, Alexandros G. Dimakis, and Constantinos Daskalakis. Consistent diffusion meets  
 557 Tweedie: Training exact ambient diffusion models with noisy data. In *Proceedings of the Forty-First*  
 558 *International Conference on Machine Learning*, ICML’24. JMLR.org, 2024.

559 Luc Devroye. Random variate generation for the generalized inverse Gaussian distribution. *Statistics*  
 560 *and Computing*, 24(2):239–246, December 2012.

561 Julius Erbach, Dominik Narnhofer, Andreas Dombos, Bernt Schiele, Jan Eric Lenssen, and Konrad  
 562 Schindler. Solving inverse problems with FLAIR, 2025. arXiv.

563 Xuefeng Gao, Hoang M. Nguyen, and Lingjiong Zhu. Wasserstein convergence guarantees for a  
 564 general class of score-based generative models. *Journal of Machine Learning Research*, 26(43):  
 565 1–54, 2025.

566 Crispin W. Gardiner. *Handbook of Stochastic Methods: For Physics, Chemistry and the Natural*  
 567 *Sciences*. Springer, Berlin, Germany, 1990.

568 Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin.  
 569 *Bayesian Data Analysis*. Chapman and Hall/CRC, November 2013. ISBN 9780429113079.

570 Clément Gilavert, Saïd Moussaoui, and Jérôme Idier. Efficient gaussian sampling for solving large-  
 571 scale inverse problems using mcmc. *IEEE Transactions on Signal Processing*, 63(1):70–80,  
 572 2015.

573 Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs  
 574 trained by a two time-scale update rule converge to a local Nash equilibrium. In *Proceedings of*  
 575 *the Thirty-First International Conference on Neural Information Processing Systems*, NIPS’17, pp.  
 576 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

577 Desmond J. Higham. An algorithmic introduction to numerical simulation of stochastic differential  
 578 equations. *SIAM Review*, 43(3):525–546, 2001.

579 Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Proceedings*  
 580 *of the Thirty-Fourth International Conference on Neural Information Processing Systems*, NIPS  
 581 ’20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

582 Hongbin Huang, Minghua Chen, and Xiao Qiao. Generative learning for financial time series  
 583 with irregular and scale-invariant patterns. In *The Twelfth International Conference on Learning*  
 584 *Representations*, 2024.

585 Samuel Hurault, Matthieu Terris, Thomas Moreau, and Gabriel Peyré. From score matching to  
 586 diffusion: A fine-grained error analysis in the Gaussian setting, 2025. arXiv.

587 Ajil Jalal, Marius Arvinte, Giannis Daras, Eric Price, Alexandros G Dimakis, and Jon Tamir. Robust  
 588 compressed sensing MRI with deep generative priors. In *Advances in Neural Information*  
 589 *Processing Systems*, volume 34, pp. 14938–14954. Curran Associates, Inc., 2021.

590 Achim Klenke. *Probability Theory: A Comprehensive Course*. Springer International Publishing,  
 591 2020. ISBN 9783030564025.

592 Muhamed Kuric, Martin Zach, Andreas Habring, Michael Unser, and Thomas Pock. The Gaussian  
 593 latent machine: Efficient prior and posterior sampling for inverse problems, 2025. arXiv.

594 Hoang Trieu Vy Le, Marion Foare, Audrey Repetti, and Nelly Pustelnik. Embedding Blake–Zisserman  
 595 regularization in unfolded proximal neural networks for enhanced edge detection. *IEEE Signal*  
 596 *Processing Letters*, 32:1271–1275, 2025.

597 Jean-Marie Lemercier, Julius Richter, Simon Welker, Eloi Moliner, Vesa Välimäki, and Timo Gerk-  
 598 mann. Diffusion models for audio restoration: A review. *IEEE Signal Processing Magazine*, 41(6):  
 599 72–84, 2024.

600 Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-  
 601 LM improves controllable text generation. In *Advances in Neural Information Processing Systems*,  
 602 2022.

603 Jiaming Liu, Rushil Anirudh, Jayaraman J. Thiagarajan, Stewart He, K. Aditya Mohan, Ulugbek S.  
 604 Kamilov, and Hyojin Kim. DOLCE: A model-based probabilistic diffusion framework for limited-  
 605 angle CT reconstruction. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*,  
 606 pp. 10464–10474, 2023.

607 Ozan Özdenizci and Robert Legenstein. Restoring vision in adverse weather conditions with patch-  
 608 based denoising diffusion models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,  
 609 pp. 1–12, 2023.

610 Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming  
 611 Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic Differentiation in PyTorch. In  
 612 *NIPS-W*, 2017.

613 Emile Pierret and Bruno Galerne. Exact evaluation of the accuracy of diffusion models for inverse  
 614 problems with Gaussian data distributions, 2025a. arXiv.

615 Emile Pierret and Bruno Galerne. Diffusion models for Gaussian distributions: Exact solutions and  
 616 Wasserstein errors. In *Proceedings of the Forty-Second International Conference on Machine*  
 617 *Learning*, 2025b.

618 Rian Po, Wang Yifan, Vladislav Golyanik, Kfir Aberman, Jonathan T. Barron, Amit H. Bermano,  
 619 Eric R. Chan, Tali Dekel, Aleksander Holynski, Angjoo Kanazawa, Karen Liu, Lingjie Liu, Ben  
 620 Mildenhall, Matthias Nießner, Björn Ommer, Christian Theobalt, Peter Wonka, and Gordon Wet-  
 621 zstein. State of the Art on Diffusion Models for Visual Computing. *Computer Graphics Forum*, 43  
 622 (2), April 2024.

623 Mehrsa Pourya, Bassam El Rawas, and Michael Unser. Flower: A flow-matching solver for inverse  
 624 problems, 2025.

625 Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising  
 626 diffusion models for multivariate probabilistic time series forecasting. In *International conference*  
 627 *on Machine Learning*, pp. 8857–8868. PMLR, 2021.

628 Mengwei Ren, Mauricio Delbracio, Hossein Talebi, Guido Gerig, and Peyman Milanfar. Multiscale  
 629 structure guided diffusion for image deblurring. In *2023 IEEE/CVF International Conference*  
 630 *on Computer Vision (ICCV)*, pp. 10687–10699, Los Alamitos, CA, USA, October 2023. IEEE  
 631 Computer Society.

632 Severi Rissanen, Markus Heinonen, and Arno Solin. Free hunch: Denoiser covariance estimation  
 633 for diffusion models without extra costs. In *The Thirteenth International Conference on Learning*  
 634 *Representations*, 2025.

635 Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-  
 636 resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF Conference on*  
 637 *Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685. IEEE, June 2022.

638 Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi.  
 639 Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and*  
 640 *Machine Intelligence*, 45(4):4713–4726, 2023.

641 Ken-Iti Sato. *Lévy Processes and Infinitely Divisible Distributions*. Cambridge studies in advanced  
 642 mathematics. Cambridge University Press, Cambridge, England, 1999.

643 Otmar Scherzer, Markus Grasmair, Harald Grossauer, Markus Haltmeier, and Frank Lenzen. *Varia-  
 644 tional Methods in Imaging*. Applied mathematical sciences. Springer, New York, NY, 2009 edition,  
 645 October 2008.

646 Wim Schoutens. *Lévy Processes in Finance*. Wiley Series in Probability and Statistics. John Wiley &  
 647 Sons, Chichester, England, 2003.

648 Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised  
 649 learning using nonequilibrium thermodynamics. In *Proceedings of the Thirty-Second International*  
 650 *Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp.  
 651 2256–2265, Lille, France, 7 2015. PMLR.

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

654 Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion  
 655 models for inverse problems. In *International Conference on Learning Representations*, 2023.

656 Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution.  
 657 In *Advances in Neural Information Processing Systems*, pp. 11895–11907, 2019.

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

661 Stanislas Strasman, Antonio Ocello, Claire Boyer, Sylvain Le Corff, and Vincent Lemaire. An  
 662 analysis of the noise schedule for score-based generative models. *Transactions on Machine*  
 663 *Learning Research*, 2025.

664 Andrew M. Stuart. Inverse Problems: A Bayesian Perspective. *Acta Numerica*, 19:451–559, 2010.

665 Yu Sun, Zihui Wu, Yifan Chen, Berthy T. Feng, and Katherine L. Bouman. Provable probabilistic  
 666 imaging using score-based generative priors. *IEEE Transactions on Computational Imaging*, 10:  
 667 1290–1305, 2024.

668 David Y. W. Thong, Charlesquin Kemajou Mbakam, and Marcelo Pereyra. Do Bayesian imaging  
 669 methods report trustworthy probabilities?, 2024. arXiv.

670 Michael Unser and Pouya D. Tafti. *An Introduction to Sparse Stochastic Processes*. Cambridge  
 671 University Press, 2014.

672 David A. van Dyk and Taeyoung Park. Partially Collapsed Gibbs Samplers. *Journal of the American  
 673 Statistical Association*, 103(482):790–796, 2008.

674 Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*.  
 675 Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018.

676 Martin J. Wainwright and Eero P. Simoncelli. Scale Mixtures of Gaussians and the Statistics of  
 677 Natural Images. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press,  
 678 1999.

679 Zheng Wang, Johnathan M. Bardsley, Antti Solonen, Tiangang Cui, and Youssef M. Marzouk.  
 680 Bayesian inverse problems with  $\ell_1$  priors: A randomize-then-optimize approach. *SIAM Journal on Scientific Computing*, 39(5):S140–S166, 2017.

682 Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment:  
 683 from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):  
 684 600–612, 2004.

685 Zhuohan Wang and Carmine Ventre. A financial time series denoiser based on diffusion models. In  
 686 *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF ’24, pp. 72–80,  
 687 New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400710810.

688 Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eise-  
 689 nach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita  
 690 Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac  
 691 Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey  
 692 Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David  
 693 Baker. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):  
 694 1089–1100, July 2023.

695 Xingyu Xu and Yuejie Chi. Provably robust score-based diffusion posterior sampling for plug-  
 696 and-play image reconstruction. In *The Thirty-Eighth Annual Conference on Neural Information  
 697 Processing Systems*, 2024.

698 Duoduo Xue, Wenrui Dai, Ziyang Zheng, Xinyu Peng, Junni Zou, and Hongkai Xiong. Fourier phase  
 699 retrieval with diffusion priors guided by failure detection. In *2025 IEEE International Symposium  
 700 on Circuits and Systems (ISCAS)*, pp. 1–5, 2025.

701 Nebiyou Yismaw, Ulugbek S. Kamilov, and M. Salman Asif. Covariance-corrected diffusion models  
 702 for solving inverse problems. In *2025 IEEE Statistical Signal Processing Workshop (SSP)*, pp.  
 703 26–30, 2025.

704 Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable  
 705 effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer  
 706 Vision and Pattern Recognition*, pp. 586–595, 2018.

707 Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van  
 708 Gool. Denoising diffusion models for plug-and-play image restoration. In *IEEE Conference on  
 709 Computer Vision and Pattern Recognition Workshops (NTIRE)*, 2023.

## 710 A BAYES ESTIMATORS

711 A benefit of the Bayesian approach over classical variational methods (see, e.g., (Scherzer et al.,  
712 2008)) is that different point estimates arise from a fixed prior. For a given measurement  $\mathbf{y}$ , these point  
713 estimates summarize the posterior distribution  $p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}$  with respect to a given loss  $\ell : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$   
714 via the optimization problem of finding the point  $\hat{\mathbf{x}}_\ell(\mathbf{y})$  that minimizes the posterior risk:

$$\hat{\mathbf{x}}_\ell(\mathbf{y}) = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^d} \left( \int_{\mathbb{R}^d} \ell(\hat{\mathbf{x}}, \mathbf{x}) p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) d\mathbf{x} \right). \quad (18)$$

715 In this paper, the Bayes estimator with respect to the mean-squared error (MSE)  $\ell = \frac{1}{d} \|\cdot - \cdot\|^2$   
716 plays a key role due to its close relation to the prior *score* in the reverse diffusion (see Section 2) and  
717 because we quantify the performance of DPS algorithms via the MMSE optimality gap in Section 4.  
718 With this choice of  $\ell$ , (18) can be written as

$$\hat{\mathbf{x}}_{\text{MMSE}}(\mathbf{y}) = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^d} \left( \int_{\mathbb{R}^d} \frac{1}{d} \|\hat{\mathbf{x}} - \mathbf{x}\|^2 p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) d\mathbf{x} \right) = \int_{\mathbb{R}^d} \mathbf{x} p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) d\mathbf{x} = \mathbb{E}[\mathbf{X} \mid \mathbf{Y} = \mathbf{y}], \quad (19)$$

719 which is the expectation of the posterior  $p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}$ .

720 Another widely-used estimator arises through the choice

$$\ell(\hat{\mathbf{x}}, \mathbf{x}) = -\chi_{\{\hat{\mathbf{x}}\}}(\mathbf{x}) \quad (20)$$

721 where

$$\chi_A(\mathbf{x}) := \begin{cases} 1 & \text{if } \mathbf{x} \in A, \\ 0 & \text{else,} \end{cases} \quad (21)$$

722 which leads to the MAP estimator that seeks the mode of the posterior:<sup>5</sup>

$$\hat{\mathbf{x}}_{\text{MAP}}(\mathbf{y}) = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^d} \left( \int_{\mathbb{R}^d} -\chi_{\{\hat{\mathbf{x}}\}}(\mathbf{x}) p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) d\mathbf{x} \right) = \arg \max_{\hat{\mathbf{x}} \in \mathbb{R}^d} p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\hat{\mathbf{x}}). \quad (22)$$

723 Rewriting (22) as

$$\hat{\mathbf{x}}_{\text{MAP}}(\mathbf{y}) = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^d} \left( -\frac{1}{2\sigma_n^2} \|\mathbf{A}\hat{\mathbf{x}} - \mathbf{y}\|^2 - \log p_{\mathbf{X}}(\hat{\mathbf{x}}) \right), \quad (23)$$

724 reveals a close relation to classical variational approaches after identifying the regularizer with  
725  $-\log p_{\mathbf{X}}$ .

## 726 B TWEEDIE'S FORMULA

727 In the setting of Section 2, we now derive an equality that relates  $\nabla \log p_{\mathbf{X}_t}$  to  $\mathbb{E}[\mathbf{X}_0 \mid \mathbf{X}_t = \cdot]$ , i.e.,  
728 the MMSE estimate of  $\mathbf{X}_0$  given that  $\mathbf{X}_t$  takes on a certain value. Similar derivations can be found  
729 in, e.g., (Song et al., 2021; Chung et al., 2023; Daras et al., 2024), but we include it to underscore  
730 the relevance of the MMSE estimate in this paper and to facilitate the understanding of its relation  
731 to various objects. Under the variance-preserving choice for  $\mathbf{f}(\mathbf{x}, t) = -\frac{\beta(t)}{2}\mathbf{x}$  and  $g(t) = \sqrt{\beta(t)}$   
732 of the drift and diffusion coefficient, the diffusion SDE (4) simplifies to a time-inhomogeneous  
733 Ornstein–Uhlenbeck SDE (see Klenke (2020, Example 26.5))

$$d\mathbf{X}_t = -\frac{\beta(t)}{2} \mathbf{X}_t dt + \sqrt{\beta(t)} d\mathbf{W}_t, \quad (24)$$

734 whose pathwise solution

$$\mathbf{X}_t = \alpha(t) \mathbf{X}_0 + \int_0^t \frac{\alpha(t)}{\alpha(s)} \sqrt{\beta(t)} d\mathbf{W}_s, \quad (25)$$

<sup>5</sup>This definition is informal but sufficient for the purposes of this paper. For continuous posteriors, the strict 0–1 loss Bayes' rule is ill-posed. A common formalization defines MAP as the limit of Bayes estimators under shrinking small-ball 0–1 losses; under additional regularity, this limit agrees with the posterior mode (Bassett & Deride, 2018; Clason et al., 2019). The MAP estimator may also not be unique.

735 where  $\mathbf{X}_0$  is an appropriate initial condition and  $\alpha(t) = \exp(-\frac{1}{2} \int_0^t \beta(s) ds)$ , can be computed with  
 736 standard techniques, see, e.g., (Gardiner, 1990, Section 4.4.4). In addition, since

$$\int_0^t \left( \frac{\alpha(t)}{\alpha(s)} \right)^2 \beta(s) ds = \int_0^t \beta(s) \exp\left(-\int_s^t \beta(u) du\right) ds = 1 - \alpha^2(t), \quad (26)$$

737 we can write that

$$\mathbf{X}_t = \alpha(t)\mathbf{X}_0 + \sigma(t)\mathbf{N} \quad (27)$$

738 in distribution, where  $\sigma^2(t) = (1 - \alpha^2(t))$ . Consequently, the density of  $\mathbf{X}_t$  is given by the convolution  
 739 of  $p_{\mathbf{X}_0}$  with a Gaussian with variance  $\sigma^2(t)$  and appropriate scaling by  $\alpha(t)$ , which we write as:

$$p_{\mathbf{X}_t}(\mathbf{x}) = \int_{\mathbb{R}^d} g_{\mathbf{0}, \sigma(t)^2 \mathbf{I}}(\mathbf{x} - \alpha(t)\hat{\mathbf{x}}) p_{\mathbf{X}_0}(\hat{\mathbf{x}}) d\hat{\mathbf{x}}, \quad (28)$$

740 where  $g_{\mu, \Sigma}(\mathbf{x}) = (2\pi)^{-\frac{d}{2}} |\Sigma|^{-\frac{1}{2}} \exp(-\frac{1}{2} \|\mathbf{x} - \mu\|_{\Sigma^{-1}}^2)$ . Finally, after taking the gradient, we see  
 741 that

$$\begin{aligned} \nabla p_{\mathbf{X}_t}(\mathbf{x}) &= \int_{\mathbb{R}^d} \nabla g_{\mathbf{0}, \sigma(t)^2 \mathbf{I}}(\mathbf{x} - \alpha(t)\hat{\mathbf{x}}) p_{\mathbf{X}_0}(\hat{\mathbf{x}}) d\hat{\mathbf{x}} \\ &= \int_{\mathbb{R}^d} \left( -\frac{1}{\sigma^2(t)} (\mathbf{x} - \alpha(t)\hat{\mathbf{x}}) g_{\mathbf{0}, \sigma^2(t) \mathbf{I}}(\mathbf{x} - \alpha(t)\hat{\mathbf{x}}) p_{\mathbf{X}_0}(\hat{\mathbf{x}}) \right) d\hat{\mathbf{x}} \\ &= -\frac{1}{\sigma^2(t)} \left( \mathbf{x} p_{\mathbf{X}_t}(\mathbf{x}) - \alpha(t) \int_{\mathbb{R}^d} \hat{\mathbf{x}} g_{\mathbf{0}, \sigma^2(t) \mathbf{I}}(\mathbf{x} - \alpha(t)\hat{\mathbf{x}}) p_{\mathbf{X}_0}(\hat{\mathbf{x}}) d\hat{\mathbf{x}} \right) \\ &= -\frac{1}{\sigma(t)^2} (\mathbf{x} p_{\mathbf{X}_t}(\mathbf{x}) - \alpha(t) p_{\mathbf{X}_t}(\mathbf{x}) \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]). \end{aligned} \quad (29)$$

742 Finally such that, after dividing by  $p_{\mathbf{X}_t}(\mathbf{x})$  and since  $\frac{\nabla p_{\mathbf{X}_t}(\mathbf{x})}{p_{\mathbf{X}_t}(\mathbf{x})} = \nabla \log p_{\mathbf{X}_t}(\mathbf{x})$ , we find the celebrated  
 743 Tweedie identity

$$\nabla \log p_{\mathbf{X}_t}(\mathbf{x}) = -\sigma(t)^{-2} (\mathbf{x} - \alpha(t) \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]). \quad (30)$$

## 744 B.1 A CONNECTION BETWEEN THE DISCRETIZED REVERSE SDE AND DDPM

745 To show the connection between the Euler–Maruyama discretization of the reverse-diffusion SDE  
 746 and the DDPM backward process, we start by deriving the latter from the respective forward process.  
 747 DDPM has been introduced by Sohl-Dickstein et al. (2015) as a discrete-time Markov chain of  
 748 length  $T$  with Gaussian transitions:

$$p_{\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{x}_{t-1}} = \text{Gauss}(\sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}), \quad (31)$$

749 such that the transitions from  $\mathbf{X}_0$  to  $\mathbf{X}_t$  are also tractable as

$$\mathbf{X}_t = \sqrt{\bar{\alpha}_t} \mathbf{X}_0 + \sqrt{1 - \bar{\alpha}_t} \mathbf{Z}_t, \quad (32)$$

750 where  $\alpha_t = (1 - \beta_t)$ ,  $\bar{\alpha}_t = \prod_{s=0}^t \alpha_s$ , and  $\mathbf{Z}_t \sim \text{Gauss}(\mathbf{0}, \mathbf{I})$ . By definition,

$$\mathbf{X}_t = \sqrt{1 - \beta_t} \mathbf{X}_{t-1} + \sqrt{\beta_t} \mathbf{Z}_{t-1} \quad (33)$$

751 and a straightforward application of Tweedie’s formula (6) gives that

$$\mathbb{E}[\mathbf{X}_{t-1} | \mathbf{X}_t] = \frac{1}{\sqrt{\alpha_t}} (\mathbf{X}_t + (1 - \alpha_t) \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t)), \quad (34)$$

752 which leads to the DDPM backward transitions

$$\mathbf{X}_{t-1} = \frac{1}{\sqrt{1 - \beta_t}} (\mathbf{X}_t + \beta_t \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t)) + \sqrt{\beta_t} \mathbf{Z}_t \quad (35)$$

753 like they appear in (7) in the main text.

754 Now, we recall the reverse-diffusion SDE, which, under our choice of the drift and diffusion coefficient,  
 755 is given by

$$d\mathbf{X}_t = \left( -\frac{\beta(t)}{2} \mathbf{X}_t - \beta(t) \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t) \right) dt + \sqrt{\beta(t)} d\mathbf{W}_t. \quad (36)$$

756 A first-order step from  $t$  to  $(t-1)$  ( $dt = -1$ ) gives the Euler–Maruyama update

$$\mathbf{X}_{t-1} = \left( 1 + \frac{\beta_t}{2} \right) \mathbf{X}_t + \beta_t \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t) + \sqrt{\beta_t} \mathbf{Z}_t, \quad (37)$$

757 where  $\beta_t := \beta(t)$  and  $\mathbf{Z}_t \sim \text{Gauss}(\mathbf{0}, \mathbf{I})$ .

758 The DDPM reverse process (35) can be related to the the Euler–Maruyama discretization of the  
759 reverse SDE (37) via Taylor expansions, since

$$\frac{1}{\sqrt{1 - \beta_t}} = 1 + \frac{\beta_t}{2} + \mathcal{O}(\beta_t^2) \quad (38)$$

760 and

$$\frac{\beta_t}{\sqrt{1 - \beta_t}} = \beta_t + \mathcal{O}(\beta_t^2) \quad (39)$$

761 as  $\beta_t \rightarrow 0$ .

## 762 C LÉVY PROCESSES AND INCREMENT DISTRIBUTIONS

763 The prior distributions in our framework are those of signals obtained by regularly spaced samples of  
764 processes with independent, stationary increments (Lévy processes and their discrete-time counter-  
765 parts). We briefly recall the definition; see [Unser & Tafti \(2014\)](#); [Sato \(1999\)](#) for background and the  
766 link to infinitely divisible laws.

767 **Definition C.1** (Lévy process). A stochastic process  $s = \{s(t) : t \geq 0\}$  is a Lévy process if

- 768 1. (anchor at the origin) It holds that  $s(0) = 0$  almost surely;
- 769 2. (independent increments) for any  $N \in \mathbb{N} \setminus \{0, 1\}$  and  $0 \leq t_1 < t_2 < \dots < t_N < \infty$ , the  
770 increments  $(s(t_2) - s(t_1)), (s(t_3) - s(t_2)), \dots, (s(t_N) - s(t_{N-1}))$  are mutually independent;
- 771 3. (stationary increments) for any given step  $h$ , the increment process  $u_h = \{s(t) - s(t - h) :  
772 t > h\}$  is stationary;
- 773 4. (stochastic continuity) for any  $\varepsilon > 0$  and  $t \geq 0$ ,

$$\lim_{h \rightarrow 0} \Pr(|s(t + h) - s(t)| > \varepsilon) = 0.$$

774 We form discrete [and finite-length](#) signals by sampling  $s$  at integer times and stacking the values  
775 into  $\mathbf{x} = (s(1), s(2), \dots, s(d))$ . Let the unit-step increments be  $[\mathbf{u}]_k \underline{\mathbf{u}_k} = (s(k) - s(k - 1))$  for  
776  $k = 1, 2, \dots, d$ . By independence and stationarity, the law<sup>6</sup> of  $[\mathbf{u}]_k \underline{\mathbf{u}_k}$  does not depend on  $k$  and we  
777 denote it  $p_U$ . We define the finite-difference matrix

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \quad (40)$$

778 such that the increment vector satisfies

$$\mathbf{u} = \mathbf{D}\mathbf{x}. \quad (41)$$

779 Because  $s(0) = 0$ , the finite-difference matrix  $\mathbf{D}$  has an initial condition [that which](#) makes it invertible  
780 and  $\mathbf{D}^{-1}$  is a lower-triangular matrix of ones. [This, which](#) also implies that for all  $k = 1, 2, \dots, d$ ,

$$[\mathbf{x}]_k \underline{\mathbf{x}_k} = \sum_{n=1}^k [\mathbf{u}]_n \underline{\mathbf{u}_n}, \quad (42)$$

781 which is a convenient way to synthesize signals once  $\mathbf{u}$  is drawn. The combination of (41) with the  
782 independence of the increments implies that the density of the discrete signal is

$$p_{\mathbf{x}}(\mathbf{x}) = \prod_{k=1}^d p_U([\mathbf{D}\mathbf{x}]_k \underline{(\mathbf{D}\mathbf{x})}_k). \quad (43)$$

<sup>6</sup>For our choices, it always has a density w.r.t. a suitable reference measure.

Table 2: **USummary of univariate** distributions used throughout this work. Parameters appear in the order they are specified in this table, *e.g.*  $\text{Gauss}(\mu, \sigma^2)$ .

Name	Distribution	Parameter(s)	Supp.	Notation
Gaussian	$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right)$	$\mu \in \mathbb{R}, \sigma^2 \in \mathbb{R}_{>0}$	$\mathbb{R}$	Gauss
Exponential	$\lambda \exp(-\lambda x)$	$\lambda \in \mathbb{R}_{>0}$	$\mathbb{R}_{\geq 0}$	Exp
Laplace	$\frac{1}{2b} \exp\left(-\frac{ x }{b}\right)$	$b \in \mathbb{R}_{>0}$	$\mathbb{R}$	Laplace
Student-t	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi\nu}\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$	$\nu \in \mathbb{R}_{>0}$	$\mathbb{R}$	St
Gamma	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x)$	$\alpha, \beta \in \mathbb{R}_{>0}$	$\mathbb{R}_{>0}$	Gamma
Gen. inv. Gaussian	$\frac{(\frac{a}{b})^{\frac{p}{2}}}{2K_p(\sqrt{ab})} x^{p-1} \exp\left(-\frac{ax+b/x}{2}\right)$	$a, b \in \mathbb{R}_{>0}, p \in \mathbb{R}$	$\mathbb{R}_{>0}$	GIG
Bernoulli–Laplace	$\lambda\delta(x) + (1-\lambda)\frac{1}{2b} \exp\left(-\frac{ x }{b}\right)$	$\lambda \in [0, 1], b \in \mathbb{R}_{>0}$	$\mathbb{R}$	BL

Moreover,  $\Gamma$  denotes the gamma function is defined as  $\Gamma(x) = \int_0^\infty t^{x-1} \exp(-t) dt$  for any  $x \in \mathbb{R}_{>0}$ .

$K_\nu$  denotes the modified Bessel function of the second kind with parameter  $\nu$  is denoted by  $K_\nu$ .

Table 3: Latent variable representations and conditional distributions for common distributions.

Dist. $\phi_{k_i}$	Latent dist. $f_{k_i}$	Latent maps	Cond. latent dist. $p_{[\mathbf{Z}]_k [\mathbf{Z}_i]   \mathbf{X} = [\mathbf{Kx}]_k [\mathbf{Kx}]_i}$
Gauss( $\mu, \sigma^2$ )	$\delta(0)$	$\mu_{k_i}(z_i) = \mu, \sigma_{k_i}^2(z_i) = \sigma^2$	$\delta(0)$
Laplace( $b$ )	$\text{Exp}\left(\frac{1}{2b^2}\right)$	$\mu_{k_i}(z_i) = 0, \sigma_{k_i}^2(z_i) = z_i$	$\text{GIG}\left(\frac{1}{b^2}, [\mathbf{Kx}]_k [\mathbf{Kx}]_i^2, \frac{1}{2}\right)$
St( $\nu$ )	$\text{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2}\right)$	$\mu_{k_i}(z_i) = 0, \sigma_{k_i}^2(z_i) = \frac{1}{z_i}$	$\text{Gamma}\left(\frac{\nu+1}{2}, \frac{\nu+[\mathbf{Kx}]_k [\mathbf{Kx}]_i^2}{2}\right)$

### 783 C.1 EXTENSIONS

784 The approach in this paper can be extended to two- or higher-dimensional signals on grids, such as  
785 images or videos, and even to more specialized structures like signals defined over trees or **more**  
786 **general** graphs. The structure of the signal is effectively encoded through the choice of the matrix  $\mathbf{D}$ .  
787 For instance, a two-dimensional finite-difference matrix would result in a signal vector that can be  
788 interpreted as a two-dimensional image. The main additional (computational) challenge is sampling  
789 during signal generation: Whenever  $\mathbf{D}$  is not trivially reducible to a one-dimensional operator, the  
790 **resulting** model (43) will be overcomplete and, in general, no whitening transformation exists to  
791 decouple increments for independent sampling. The extension to higher-dimensional signals and the  
792 complications that arise in that context are rigorously treated in Kuric et al. (2025).

### 793 C.2 LATENT DISTRIBUTIONS AND NOTATION

794 Some of the distributions that we rely on in this work have multiple competing parametrizations. To  
795 avoid ambiguities, we provide precise definitions of the four increment distributions that we consider  
796 in this work:— Gaussian;— Laplace;— Student-t;— and Bernoulli–Laplace (spike-and-slab).— We give  
797 in Table 2 and our notations of these and other distributions that we use in this work in Table 2. We list  
798 in Table 3 lists the latent maps and conditional latent distributions that are needed for the GLM for  
799 the distributions in this work.

### 800 D GIBBS METHODS AND SAMPLING EFFICIENCY

801 Gibbs methods are Markov chain Monte Carlo (MCMC) methods to sample from a joint distribution  
802  $p_{\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n}$  of  $(n+1)$  blocks of variables that are advantageous when the direct sampling is com-  
803 putationally expensive.

**Algorithm 3** Latent-variable Gibbs sampling of  $p_{\mathbf{X}, \mathbf{Z}_1, \dots, \mathbf{Z}_n}$ .

**Require:** Burn-in period  $B \in \mathbb{N}$ , number of samples  $S \in \mathbb{N}$ , initial point  $(\mathbf{x}_0, \mathbf{z}_1, \dots, \mathbf{z}_n)$ .

```

1: for  $k \in 1, 2, \dots, B + S$  do
2:    $\mathbf{x}_{k \in} \sim p_{\mathbf{X} | \mathbf{Z}_1 = \mathbf{z}_1, \dots, \mathbf{Z}_n = \mathbf{z}_n}$ 
3:    $\mathbf{z}_1 \sim p_{\mathbf{Z}_1 | \mathbf{X} = \mathbf{x}_k, \dots, \mathbf{Z}_n = \mathbf{z}_n}$  ▷ Latent blocks do not need to be stored
4:    $\vdots$ 
5: return  $\{\mathbf{x}_{B+k}\}_{k=1}^S$ 

```

803 tationally difficult but sampling from the conditional distributions  $p_{\mathbf{X} | \mathbf{Z}_1, \mathbf{Z}_2, \dots, \mathbf{Z}_n}, p_{\mathbf{Z}_1 | \mathbf{X}, \mathbf{Z}_2, \dots, \mathbf{Z}_n}, \dots$   
804 is easy. Gibbs methods cycle through the conditional distributions with repeated draws, which  
805 maintains the joint distribution invariant (Casella & George, 1992). The naming of the variables  
806  $\mathbf{X}, \mathbf{Z}_1, \mathbf{Z}_2, \dots, \mathbf{Z}_n$  is deliberately chosen to emphasize that we use *latent-variable* Gibbs methods  
807 that rely on auxiliary variables that are introduced solely to make the conditionals simple. The steps  
808 of a general latent variable Gibbs sampler are shown in **Algorithm 3**, where the iteration counter in  
809 the sampling of the latent variables is omitted since they need not be stored and previous iterations  
810 can immediately be overwritten.

811 Kuric et al. (2025) recently showed that such methods are significantly faster than other standard  
812 sampling routines that are commonly used in settings similar to the one in this paper. They report  
813 sampling efficiencies of close to 1, while alternatives, such as the Metropolis-adjusted Langevin  
814 algorithm, achieve sampling efficiencies<sup>7</sup> of around  $1 \times 10^{-3}$ . In addition, Gibbs methods require no  
815 step-size or acceptance-rate tuning and introduce no discretization bias. These properties motivate  
816 our use of Gibbs methods for the fast and robust posterior sampling throughout this work.

817 Like all MCMC methods, in practice Gibbs methods benefit from the discarding of some number of  
818 initial samples (the *burn-in period*), when the initial point is located in low-density regions. After  
819 the burn-in period, the quality of the Monte Carlo estimate of any object depends on the number of  
820 samples one uses in their estimation. it is crucial to tune the number of samples in such a way that  
821 the Monte Carlo estimates of various quantities, such as the MMSE estimate in (19), are sufficiently  
822 accurate. We discuss our choice of the burn-in period and the number of samples for the various  
823 problems in Appendix F.2.

[tkeZ TmEt] Clarification of our experimental setup.

824 D.1 A GIBBS METHOD FOR BERNOUlli-LAPLACE INCREMENTS

825 Let  $\delta$  be the Dirac distribution. Then, letting  $\lambda$  be the Bernoulli parameter and  $b$  the scale parameter,  
826 we note We start by noting that the Bernoulli-Laplace density

$$p_U(u) = \lambda\delta(u) + (1 - \lambda)\frac{b}{2}\exp(-b|u|) \quad (44)$$

827 with Bernoulli parameter  $\lambda$  and scale parameter  $b$ , where  $\delta$  is the Dirac distribution, admits the  
828 representation

$$p_U(u) = \int_{\mathbb{R}} \left( \sum_{v=0}^1 p_{U|V=v, W=w}(u) p_V(v) \right) p_W(w) dw, \quad (45)$$

829 where

$$p_V(v) = \lambda^{1-v}(1 - \lambda)^v \quad (46)$$

830 for  $v \in \{0, 1\}$  is a Bernoulli distribution,

$$p_W(w) = \frac{b^2}{2} \exp\left(-\frac{b^2 w}{2}\right) \chi_{\mathbb{R}_{\geq 0}}(w) \quad (47)$$

831 is an exponential distribution, and

$$p_{U|V=v, W=w}(u) = \begin{cases} \delta(u) & \text{if } v = 0, \\ \text{Gauss}(0, w) & \text{if } v = 1. \end{cases} \quad (48)$$

<sup>7</sup>Sampling efficiency refers to effective samples per iteration; an efficiency of  $\rho$  means roughly  $1/\rho$  iterations per “effective sample” (Gelman et al., 2013, Section 11.5).

832 The algorithm relies on the introduction of two latent vectors  $\mathbf{v}, \mathbf{w} \in \mathbb{R}^d$  that satisfy

$$p_{\mathbf{U}|\mathbf{V}=\mathbf{v}, \mathbf{W}=\mathbf{w}}(\mathbf{u}) = \prod_{k=1}^d p_{U|V=[\mathbf{v}]_k \mathbf{v}_k, W=[\mathbf{w}]_k \mathbf{w}_k}([\mathbf{u}]_k \mathbf{u}_k) \quad (49)$$

833 such that, as a result, the distribution conditioned on the measurements can be written as

$$\begin{aligned} p_{\mathbf{U}, \mathbf{V}, \mathbf{W}|\mathbf{Y}=\mathbf{y}}(\mathbf{u}, \mathbf{v}, \mathbf{w}) &\propto \exp\left(-\frac{1}{2\sigma_n^2} \|\mathbf{H}\mathbf{u} - \mathbf{y}\|^2\right) \prod_{k=1}^d p_{U|V=[\mathbf{v}]_k \mathbf{v}_k, W=[\mathbf{w}]_k \mathbf{w}_k}([\mathbf{u}]_k \mathbf{u}_k) \\ &\times \prod_{k=1}^d \lambda^{1-[\mathbf{v}]_k \mathbf{v}_k} (1-\lambda)^{[\mathbf{v}]_k \mathbf{v}_k} \prod_{k=1}^d \frac{b^2}{2} \exp\left(-\frac{b^2 [\mathbf{w}]_k \mathbf{w}_k}{2}\right), \end{aligned} \quad (50)$$

834 where  $\mathbf{H} = \mathbf{A}\mathbf{D}^{-1}$ . Equations (48) and (50) imply that any sample from  $p_{\mathbf{U}|\mathbf{V}=\mathbf{v}, \mathbf{W}=\mathbf{w}, \mathbf{Y}=\mathbf{y}}$  takes  
835 the value zero at those indices where  $\mathbf{v}$  is zero, and values from a multivariate Gaussian distribution  
836 with covariance  $\mathbf{C} = (\sigma_n^2 \mathbf{H}\mathbf{H}^T + \text{diag}\text{diag}(\mathbf{w}))^{-1}$  and mean  $\sigma_n^{-2} \mathbf{C}\mathbf{H}^T \mathbf{y}$  otherwise. Sampling  
837  $\mathbf{W} | \mathbf{U} = \mathbf{u}, \mathbf{V} = \mathbf{v}, \mathbf{Y} = \mathbf{y}$  amounts to the independent sampling of  $d$  one-dimensional distributions,  
838 which are  $\text{Exp}(2/b^2)$  at those indices where  $\mathbf{v}$  is zero and  $\text{GIG}(b^2, [\mathbf{u}]_k \mathbf{u}_k^2, 0.5)$  those indices  $k$   
839 where  $\mathbf{v}$  is one. The conditional distribution of the binary support vector is

$$p_{\mathbf{v}|\mathbf{W}=\mathbf{w}, \mathbf{Y}=\mathbf{y}}(\mathbf{v}) \propto |\mathbf{B}(\mathbf{v}, \mathbf{w})|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \mathbf{y}^T \mathbf{B}(\mathbf{v}, \mathbf{w})^{-1} \mathbf{y}\right) \prod_{k=1}^d \lambda^{1-[\mathbf{v}]_k \mathbf{v}_k} (1-\lambda)^{[\mathbf{v}]_k \mathbf{v}_k}, \quad (51)$$

840 where<sup>8</sup>  $\mathbf{B}(\mathbf{v}, \mathbf{w}) = \sigma_n^2 \mathbf{I} + \mathbf{H} \text{diag}\text{diag}(\mathbf{v} \odot \mathbf{w}) \mathbf{H}^T$ . The standard way to sample from this distribution  
841 is to use a coordinate-wise Gibbs sampler that updates  $[\mathbf{v}]_k \mathbf{v}_k \sim \text{Bernoulli}(p_k(\mathbf{v}))$  with

$$p_k(\mathbf{v}) = (1 + \exp(-\Delta_k(\mathbf{v})))^{-1} \quad (52)$$

842 where the log-odds increment

$$\begin{aligned} \Delta_k(\mathbf{v}) &= \log \frac{1-\lambda}{\lambda} - \frac{1}{2} (\log |\mathbf{B}(\mathbf{v}_{k=1}, \mathbf{w})| - \log |\mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})|) \\ &\quad - \frac{1}{2} (\mathbf{y}^T \mathbf{B}(\mathbf{v}_{k=1}, \mathbf{w})^{-1} \mathbf{y} - \mathbf{y}^T \mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})^{-1} \mathbf{y}), \end{aligned} \quad (53)$$

843 where  $\mathbf{v}_{k= \cdot} := (\mathbf{v}_1, \dots, \mathbf{v}_{k-1}, \cdot, \mathbf{v}_{k+1}, \dots, \mathbf{v}_d)$  is the difference between the log-posterior when  
844 the bit is on and when it is off. The resulting algorithm that is summarized in Algorithm 4 and can  
845 be interpreted<sup>9</sup> as  $(d+2)$ -blockvariable (i.e., dimension-dependent) Gibbs method and an efficient  
846 implementation is crucial.

## 847 D.2 PRACTICAL GIBBS IMPLEMENTATIONS

848 **Sampling the Gaussians** The sampling of  $\mathbf{X} | \mathbf{Z}$  in the GLM and of  $\mathbf{U} | \mathbf{V}, \mathbf{W}, \mathbf{Y}$  for the  
849 Bernoulli–Laplace case reduces to drawing from a high-dimensional Gaussian, which is a well-  
850 studied problem. For settings that necessitate a matrix-free implementation such as those that are  
851 commonly encountered in imaging applications, Kuric et al. (2025) advocate a Perturb-and-MAP  
852 sampler with preconditioned conjugate-gradient solvers. We report the runtime of the Gibbs method  
853 as a function of signal dimension for a Laplace(1) increment distribution in Figure 6. A standard  
854 implementation based on a Cholesky factorization of the covariance matrix—which requires explicitly

<sup>8</sup>This is a different but equivalent formulation is equivalent to the one that is presented by Bohra et al. (2023), who explicitly “slice” the matrices  $\mathbf{H}$  and  $\text{diag}\text{diag}(\mathbf{w})$  with the indices where  $\mathbf{v}$  is one. We stick to this formulation since it requires less notation and emphasizes that implementations need not build variable-sized matrices, which is crucial for an efficient implementation on modern compute units that utilize highly parallelized computations.

<sup>9</sup>This is only an interpretation because not strictly correct since the density violates the classical positivity conditions that are needed for Gibbs methods. It is a partially collapsed Gibbs method, see (Bohra et al., 2023; van Dyk & Park, 2008).

**Algorithm 4** Bernoulli–Laplace Gibbs sampler.

---

**Require:** Initial increments  $\mathbf{u}_0 \in \mathbb{R}^{\frac{dn}{2}}$ , initial support vector  $\mathbf{v} \in \mathbb{R}^d$

- 1: **for**  $s = 1, 2, \dots, B + S$  **do**
- 2:     Draw  $[\mathbf{w}]_k \underline{\mathbf{w}_i} \sim p_{W|U=[\mathbf{u}_{s-1}]_k, V=[\mathbf{v}]_k \underline{\mathbf{v}_i}}$   $\triangleright$  parallel over  $k$
- 3:     **for**  $k = 1, 2, \dots, d$  **do**
- 4:         Draw  $[\mathbf{v}]_k \underline{\mathbf{v}_k} \sim \text{Bernoulli}(p_k(\mathbf{v}))$
- 5:     Draw  $\mathbf{u}_s \sim p_{U|V=\mathbf{v}, W=\mathbf{w}, Y=\mathbf{y}}$
- 6: **return**  $\{\mathbf{D}^{-1} \mathbf{u}_{B+k}\}_{k=1}^S$

---

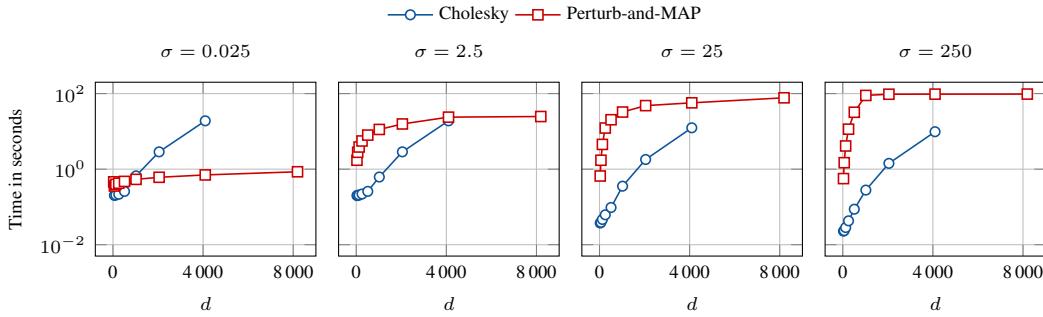


Figure 6: Runtimes needed to perform 20 Gibbs iterations on a denoising posterior (Laplace(1) increment distribution, 10 parallel chains) depending on the dimensionality of the signal. Missing entries are due to excessive memory requirements.

[TmEt] Scalability to higher dimensions.

855 instantiating the matrices  $\mathbf{A}$  and  $\mathbf{D}$  in memory—is faster than the Perturb-and-MAP sampler with a  
856 conjugate-gradient solver across a broad range of noise variances and dimensions. For our moderate-  
857 dimensional setting with  $d = 64$ , the Cholesky-based implementation is approximately an order of  
858 magnitude faster.

859 For our moderate-dimensional problems with  $d = 64$ , a standard implementation based on the  
860 Cholesky factorization of the covariance matrix offered a significantly faster (approximately one  
861 order of magnitude) sampling.

862 However, explicitly storing these matrices becomes infeasible at larger dimensions (in our setup, we  
863 ran out of memory at  $d = 8096$ ), and the expected cubic scaling is apparent in the figure. In contrast,  
864 the Perturb-and-MAP sampler (convergence criterion: squared residual norm below  $1 \times 10^{-6}$ ), while  
865 slower than Cholesky at small dimensions, exhibits substantially better scaling with signal dimension.  
866 In particular, it does not require materializing the operators: both the measurement operator  $\mathbf{A}$  and  
867 the finite-difference operator  $\mathbf{D}$  can be implemented efficiently in a matrix-free manner. Moreover,  
868 the sublinear runtime observed in this experiment suggests that the corresponding linear systems are  
869 well conditioned.

870 The sampling accuracy of Perturb-and-MAP depends on the termination criterion used by the  
871 optimization solver, and any finite stopping rule yields approximate samples. A principled refinement  
872 is to incorporate a Metropolis–Hastings correction step to remove bias, and to tune the solver accuracy  
873 to optimize overall runtime; this strategy was proposed by Gilavert et al. (2015), to which we refer for  
874 details. Overall, these results indicate that the Gibbs method scales favorably to higher dimensions.  
875 Combined with the fact that the denoising posteriors are sub-Gaussian, and with the relatively mild  
876 sample-complexity requirements for estimator accuracy in this setting, this suggests that the proposed  
877 framework scales well as the dimension increases.

878 **Sampling the Latent Variables** The sampling of the different latent variables necessitates the sam-  
879 pling of the one-dimensional conditional latent distributions. All the conditional latent distributions  
880 that are relevant in this paper admit efficient samplers that are readily available in standard scientific  
881 computing packages or can be implemented with little effort. We reuse the CUDA implementation

[TmEt] Scalability to higher dimensions.

882 of the generalized inverse Gaussian sampler from [Kuric et al. \(2025\)](#) that implements the method  
 883 proposed by [Devroye \(2012\)](#) in [\(Devroye, 2024\)](#) and rely on [PyTorch](#) ([Paszke et al., 2017](#)) for all  
 884 others. Wherever possible, latent updates are parallelized.

885 In the Gibbs methods for the Bernoulli–Laplace increments, the sequential drawing of the binary  
 886 support vector  $\mathbf{V}$  is embedded in the outer Gibbs loop, which, in turn, may be embedded in the  
 887 reverse-diffusion loop. This makes it crucial to minimize the use of heavy linear-algebra operations  
 888 to achieve acceptable runtimes. Writing  $\mathbf{B}(\mathbf{v}, \mathbf{w}) = \sigma_n^2 \mathbf{I} + \mathbf{H} \mathbf{diag}(\mathbf{v} \odot \mathbf{w}) \mathbf{H}^\top$ , we recognize  
 889 that the flipping of the  $k$ th bit of  $\mathbf{v}$  adds or removes a rank-one term  $[\mathbf{w}]_k \mathbf{w}_k^\top \mathbf{H}_k \mathbf{H}_k^\top$ , where  $\mathbf{H}_k$  is  
 890 the  $k$ th column of  $\mathbf{H}$ . Using the matrix-determinant lemma and Woodbury–Sherman–Morrison, we  
 891 update

$$\log |\mathbf{B}(\mathbf{v}_{k=1}, \mathbf{w})| = \log |\mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})| + \log(1 + [\mathbf{w}]_k \mathbf{w}_k^\top \tau_k) \quad (54)$$

892 and

$$\mathbf{y}^\top \mathbf{B}(\mathbf{v}_{k=1}, \mathbf{w})^{-1} \mathbf{y} = \mathbf{y}^\top \mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})^{-1} \mathbf{y} - \frac{[\mathbf{w}]_k \mathbf{w}_k^\top (\mathbf{H}_k^\top \mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})^{-1} \mathbf{y})^2}{1 + [\mathbf{w}]_k \mathbf{w}_k^\top \tau_k}, \quad (55)$$

893 where  $\tau_k = \mathbf{H}_k^\top \mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})^{-1} \mathbf{H}_k$ . Thus, an efficient implementation factors  $\mathbf{B}(\mathbf{v}, \mathbf{w})$  once per  
 894 latent state, obtains the needed scalars via triangular solves, and performs rank-one updates as bits  
 895 flip. [We report our cumulative runtime improvement over a naive implementation in Figure 4.](#)

## 896 E DPS UPDATE STEPS

### 897 E.1 COVARIANCE IN C-DPS

898 C-DPS ([Chung et al., 2023](#)) uses the approximation of the likelihood

$$p_{\mathbf{Y}|\mathbf{X}_t=\mathbf{x}}(\mathbf{y}) \approx p_{\mathbf{Y}|\mathbf{X}_0=\mathbb{E}[\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}]}(\mathbf{y}). \quad (56)$$

899 When the noise in the inverse problem is Gaussian, the likelihood score  
 900  $\nabla(\mathbf{x} \mapsto \log p_{\mathbf{Y}|\mathbf{X}_0=\mathbb{E}[\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}]}(\mathbf{y}))$  necessitates the computation of

$$\nabla(\mathbf{x} \mapsto \frac{1}{2} \|\mathbf{A} \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}] - \mathbf{y}\|^2), \quad (57)$$

901 which is

$$\mathbf{J}(\mathbf{x} \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]) (\cdot) \mathbf{A}^\top (\mathbf{A} \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \cdot] - \mathbf{y}) \quad (58)$$

902 after an application of the chain rule. The Jacobian  $\mathbf{J}(\mathbf{x} \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}])$  is typically computed  
 903 with automatic differentiation when  $(\mathbf{x}, t) \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]$  is approximated with a neural network.  
 904 In our framework, we use the connection with the covariance matrix  $\text{Cov}[\mathbf{X}_0 | \mathbf{X}_t = \cdot]$ . Indeed, as  
 905 also shown by [Rissanen et al. \(2025\)](#) in, e.g., [Rissanen et al., 2025](#), if  $\mathbf{X}_0$  and  $\mathbf{X}_t$  verify (32), then

$$\frac{1}{1-\bar{\alpha}_t} \text{Cov}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}] = \frac{1}{\bar{\alpha}_t} (\mathbf{I} + (1-\bar{\alpha}_t)^2 \nabla^2 \log p_{\mathbf{X}_t}(\mathbf{x})). \quad (59)$$

906 This identity, combined with the derivative of (6), yields

$$\mathbf{J}(\mathbf{x} \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]) (\mathbf{x}_t) = \frac{\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t} \text{Cov}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t]. \quad (60)$$

### 907 E.2 EXPLICIT UPDATE STEPS

908 We give the [define](#) some instantiations of the update steps  $\mathcal{S}(\mathbf{x}_t, \{\bar{\mathbf{x}}_s\}_{s=1}^S, \mathbf{y}, \mathbf{A}, \boldsymbol{\lambda}, t)$  a variety of DPS  
 909 algorithms below of a non-exhaustive list of DPS algorithms: C-DPS, DiffPIR, DPnP. Each  $\mathbf{z}_t$  is a  
 910  $d$ -dimensional random vector with i.i.d. standard Gaussian entries.

**[tkeZ]** Additional instantiations of the update step to showcase the versatility of our template.

911 **Score-ALD (Jalal et al., 2021)** The input parameters of this algorithm are composed of the following: A noise schedule  $\{\beta_t\}_{t=0}^{T-1}$ , the noise level of the inverse problem  $\sigma_n$ , and annealing parameters  $\{\eta_t\}_{t=0}^{T-1}$  and  $\{\gamma_t\}_{t=0}^{T-1}$ . The update step goes

$$\begin{aligned} \bar{\mu} &= \frac{1}{S} \sum_{s=1}^S \bar{\mathbf{x}}_s, \\ \mathbf{s}_t &= (\bar{\mu} - \mathbf{x}_t) / \beta_t^2, \\ \mathbf{x}_{t-1} &= \mathbf{x}_t + \eta_t (\mathbf{s}_t + \frac{1}{\gamma_t^2 + \sigma_n^2} \mathbf{A}^{\top T} (\mathbf{y} - \mathbf{A} \mathbf{x}_t)) + \sqrt{2\eta_t} \mathbf{z}_t. \end{aligned} \quad (61)$$

914

915 **C-DPS (Chung et al., 2023)** The input parameters are composed of the following.  $\bar{\alpha}_t$  is the  
 916 variance-preserving scaling weight  $\bar{\alpha}_t$  as in (32),  $\beta_t$  is the variance of the diffusion transitions  $\beta_t$  as  
 917 in (35), and a scalar  $\zeta$  that governs the likelihood-guidance strength. The diffusion noise  
 918 level that corresponds to the denoising posterior is denoted  $\sigma_t = (1 - \bar{\alpha}_t) / \sqrt{\bar{\alpha}_t}$ , which is used to  
 919 compute the samples  $\{\bar{\mathbf{x}}_{k\underline{s}}\}_{k\underline{s}=1}^S$ . The update step goes

$$\begin{aligned} \bar{\mu} &= \frac{1}{S} \sum_{k\underline{s}=1}^S \bar{\mathbf{x}}_{k\underline{s}}, \\ \mathbf{C} &= \frac{1}{S} \sum_{k\underline{s}=1}^S (\bar{\mathbf{x}}_{k\underline{s}} - \bar{\mu})(\bar{\mathbf{x}}_{k\underline{s}} - \bar{\mu})^{\top T}, \\ \mathbf{x}'_{t-1} &= \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \bar{\mu} + \sigma_t \mathbf{z}_t, \\ \tilde{\mathbf{x}}_{t-1} &= \mathbf{x}'_{t-1} - \frac{\zeta}{\|\mathbf{A}\bar{\mu} - \mathbf{y}\|} \frac{\sqrt{\bar{\alpha}_t}}{1 - \bar{\alpha}_t} \mathbf{C}^{\top T} \mathbf{A}^{\top T} (\mathbf{A} \bar{\mu} - \mathbf{y}), \\ \mathbf{x}_{t-1} &= \tilde{\mathbf{x}}_{t-1} / \sqrt{\bar{\alpha}_{t-1}}. \end{aligned} \quad (62)$$

920 **DiffPIR (Zhu et al., 2023)** The input parameters are similar to those of C-DPS.  $\bar{\alpha}_t$  and  $\sigma_t$  are  
 921 defined in the same way as in C-DPS, and  $\zeta$  parameterizes the likelihood-guidance strength. It also  
 922 uses the noise level of the inverse problem  $\sigma_n$  and an additional balance hyperparameter  $\gamma$ . The  
 923 update step goes

$$\begin{aligned} \bar{\mu} &= \frac{1}{S} \sum_{k\underline{s}=1}^S \bar{\mathbf{x}}_{k\underline{s}}, \\ \rho_t &= \zeta \frac{\sigma_n^2}{\sigma_t^2}, \\ \bar{\mathbf{x}}_0 &= \arg \min_{\mathbf{x} \in \mathbb{R}^d} \left( \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 + \frac{\rho_t}{2} \|\mathbf{x} - \bar{\mu}\|^2 \right), \\ \hat{\epsilon} &= \frac{1}{\sqrt{1 - \bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \bar{\mathbf{x}}_0), \\ \tilde{\mathbf{x}}_{t-1} &= \sqrt{\bar{\alpha}_{t-1}} \bar{\mathbf{x}}_0 + \sqrt{1 - \bar{\alpha}_{t-1}} (\sqrt{1 - \gamma} \hat{\epsilon} + \sqrt{\gamma} \mathbf{z}_t), \\ \mathbf{x}_{t-1} &= \tilde{\mathbf{x}}_{t-1} / \sqrt{\bar{\alpha}_{t-1}}. \end{aligned} \quad (63)$$

924 **PIGDM (Song et al., 2023)** The input parameters are the noise schedule  $\{\sigma_t\}_{t=0}^{T-1}$ , the data-  
 925 dependent noise schedule  $\{r_t\}_{t=0}^{T-1}$ , and the DDIM (Song et al., 2020) time-dependent coefficients

926  $\{c_t^{(1)}\}_{t=0}^{T-1}$  and  $\{c_t^{(2)}\}_{t=0}^{T-1}$ . The update step goes

$$\begin{aligned}
 \bar{\mu} &= \frac{1}{S} \sum_{k\underline{s}=1}^S \bar{\mathbf{x}}_{k\underline{s}}, \\
 \mathbf{C} &= \frac{1}{S} \sum_{s=1}^S (\bar{\mathbf{x}}_{k\underline{s}} - \bar{\mu})(\bar{\mathbf{x}}_{k\underline{s}} - \bar{\mu})^{\top T}, \\
 \mathbf{J} &= \frac{\sqrt{\bar{\alpha}_t}}{1 - \bar{\alpha}_t} \mathbf{C}, \\
 \mathbf{g} &= \left( (\mathbf{y} - \mathbf{A}\bar{\mu})^{\top T} \left( \mathbf{A}\mathbf{A}^{\top T} + \frac{\sigma_t^2}{r_t^2} \mathbf{I} \right)^{-1} \mathbf{A}\mathbf{J} \right)^{\top T}, \\
 \hat{\epsilon} &= \frac{1}{\sqrt{1 - \bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \bar{\mu}), \\
 \tilde{\mathbf{x}}_{t-1} &= \sqrt{\alpha_s} \bar{\mu} + c_1 \mathbf{z}_t + c_2 \hat{\epsilon} + \sqrt{\alpha_t} \mathbf{g}, \\
 \mathbf{x}_{t-1} &= \tilde{\mathbf{x}}_{t-1} / \sqrt{\bar{\alpha}_{t-1}}.
 \end{aligned} \tag{64}$$

927

928 **DPnP (Xu & Chi, 2024)** The diffusion noise level that corresponds to the denoising posterior is  
 929 denoted  $\eta_t$ , which is used to compute the sample  $\bar{\mathbf{x}}_{\underline{s}=1}$ . This same  $\eta_t$  defines the likelihood-guidance  
 930 strength. The update step goes

$$\begin{aligned}
 \mathbf{x}_0 &= \bar{\mathbf{x}}_1, \\
 \mathbf{x}_{t-1} &\sim \exp \left( -\frac{1}{2} \|\mathbf{A} \cdot -\mathbf{y}\|^2 - \frac{1}{2\eta_t^2} \|\cdot - \mathbf{x}_0\|^2 \right).
 \end{aligned} \tag{65}$$

931 **Annealed Plug-and-Play Monte Carlo (PnP and RED variants) (Sun et al., 2024)** The diffusion  
 932 noise level that corresponds to the denoising posterior is denoted  $\sigma_t$ , which are used to compute the  
 933 samples  $\{\bar{\mathbf{x}}_k\}_{k=1}^S$ . The parameter  $\eta$  denotes the likelihood guidance strength, and  $\kappa_t$  is an annealing  
 934 parameter.

935 The update step for the PnP variant goes

$$\begin{aligned}
 \bar{\mu} &= \frac{1}{S} \sum_{k\underline{s}=1}^S \bar{\mathbf{x}}_{k\underline{s}}, \\
 \mathbf{s} &= (\bar{\mu} - \mathbf{x}_t) / \sigma_t^2, \\
 \mathbf{x}_{t-1} &= \mathbf{x}_t + \gamma \kappa_t \mathbf{s} + \sqrt{2\gamma} \mathbf{z}_t, \\
 \mathbf{x}_t &\leftarrow \mathbf{x}_t - \gamma \mathbf{A}^{\top T} (\mathbf{A}\mathbf{x} - \mathbf{y}).
 \end{aligned} \tag{66}$$

936 The update step for the RED variant goes

$$\begin{aligned}
 \bar{\mu} &= \frac{1}{S} \sum_{k\underline{s}=1}^S \bar{\mathbf{x}}_{k\underline{s}}, \\
 \mathbf{s} &= (\bar{\mu} - \mathbf{x}_t) / \sigma_t^2, \\
 \mathbf{x}_{t-1} &= \mathbf{x}_t - \gamma (\mathbf{A}^{\top T} (\mathbf{A}\mathbf{x} - \mathbf{y}) - \kappa_t \mathbf{s}) + \sqrt{2\gamma} \mathbf{z}_t.
 \end{aligned} \tag{67}$$

937

938 The DPS template that is summarized in [Algorithm 2](#) is illustrated with a one-dimensional toy-example  
 939 in [Figure 7](#).

## 940 F NUMERICAL EXPERIMENTS

### 941 F.1 FORWARD OPERATORS

942 We consider four forward operators  $\mathbf{A}$  in our experiments. The first operator `isFirst`, the identity  
 943  $\mathbf{A} = \mathbf{I} \in \mathbb{R}^{d \times d}$ . This choice is motivated by the fundamental role that denoising algorithms currently

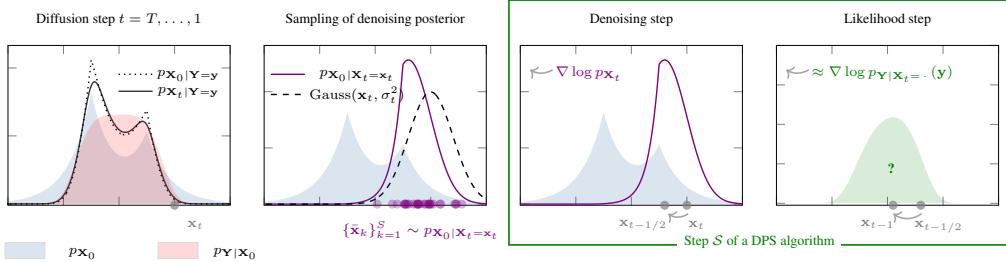


Figure 7: Illustration of the proposed template for DPS algorithms. The benchmarked posterior sampler targets  $\mathbf{x}_0 \sim p_{\mathbf{X}_0 | \mathbf{Y}=\mathbf{y}}$  via a diffusion process. At each diffusion time  $t$ , first the samples  $\{\mathbf{x}_k\}_{k=1}^S \sim p_{\mathbf{X}_0 | \mathbf{X}_t=\mathbf{x}_t}$  are drawn from the denoising posterior. Then, the step  $S$  updates the iterate typically through a prior-guided update from the samples and a likelihood-guided update from the data. The likelihood guidance term is intractable and must be approximated, which constitutes the primary source of sampling error.

play in many restoration algorithms and even in labeling problems such as edge detection (Le et al., 2025). The second `Second`, a convolution operator  $\mathbf{A} \in \mathbb{R}^{d \times d}$  that implements the convolution with a kernel that consists of the 13 central samples of a truncated Gaussian with variance 2 that are and is normalized to unit sum. We adopt circular boundary conditions to enable a fast computation of the proximal map that arises in the update step of DiffPIR (see Appendix E.2) via the fast Fourier transform. Deconvolution is a relevant problem with applications like in, e.g., microscopy and astronomy. The third operator is `Third`, a sampling operator  $\mathbf{A} \in \mathbb{R}^{m \times d}$  that returns  $m < d$  entries of its argument unchanged. This operator is also relevant in many fields such as image reconstruction and time-series forecasting. In particular, in a forecasting or prediction problem the operator would return can be modeled by returning the first  $m$  known entries, and the resolution of the inverse problem estimates recovering the remaining  $(d - m)$  entries through the resolution of the inverse problem. In our experiments, each entry has an independent chance of 40 % of being kept. The fourth and last `Fourth`, an operator is  $\mathbf{A} = \mathbf{MF} \in \mathbb{R}^{m \times d}$ , where  $\mathbf{F} \in \mathbb{R}^{2(\lfloor d/2 \rfloor + 1) \times d}$  is the matrix representation of the “real” one-dimensional discrete Fourier transform with separated real and imaginary components, and  $\mathbf{M} \in \mathbb{R}^{m \times 2(\lfloor d/2 \rfloor + 1)}$  is a sampling operator. Such operators are relevant in, e.g., medical imaging and astronomy. The sampling operator is constructed such that the 5 lowest frequencies (including the DC term included) are acquired, while the remaining frequencies independently have a 40 % chance of being kept.

For all operators, the noise variance  $\sigma_n^2$  is chosen such that the median measurement signal-to-noise ratio (SNR) is around 25 dB. We set  $N_{\text{train}} = 1 \times 10^6$ ,  $N_{\text{val}} = 1 \times 10^3$ , and  $N_{\text{test}} = 1 \times 10^3$ .

## 964 F.2 BENCHMARK IMPLEMENTATION-DETAILS

The benchmarking pipeline starts with the generation of  $N_{\text{test}}$  test signals denoted  $\{\mathbf{x}_k^{\text{test}}\}_{k=1}^{N_{\text{test}}}$  per increment distribution, each of which is independently synthesized by first drawing i.i.d. increments from the respective increment distribution and forming the signals via (42). It then proceeds to synthesize the  $N_{\text{test}}$  measurements (i.e. we use one noise instance per signal) denoted  $\{\mathbf{y}_k^{\text{test}}\}_{k=1}^{N_{\text{test}}}$  according to (1) and, for each of the measurements, computes the gold-standard posterior samples of the various inverse problems via the Gibbs methods described in Section 3. This stage is off-line (no reverse-diffusion loop) and trivially parallel across the measurements, which allows us to run long chains with burn-in periods of  $1 \times 10^5$  iterations and obtain  $2 \times 10^5$  draws from the posterior distribution. This far exceeds any values reported by Kuric et al. (2025) in (Kuric et al., 2025) or Bohra et al. (2023) (Bohra et al., 2023) and results in precise MMSE estimates.

The dataset-generation stage also involves the generation of  $N_{\text{train}}$  training signals  $\{\mathbf{x}_k^{\text{train}}\}_{k=1}^{N_{\text{train}}}$  and  $N_{\text{val}}$  validation signals (mutually disjoint from the test signals)  $\{\mathbf{x}_k^{\text{val}}\}_{k=1}^{N_{\text{val}}}$ , along with and the corresponding validation measurements  $\{\mathbf{y}_k^{\text{val}}\}_{k=1}^{N_{\text{val}}}$ . The training signals are used for the learning of a neural score function like those that are used for the resolution of inverse problems when the prior is unknown or too expensive to evaluate. Training details are provided in Appendix F.3. The validation signals are used to monitor the performance of the neural score function on unseen signals

981 during the training stage and to tune the regularization parameters for the model-based approaches as  
 982 well as the parameters of the DPS algorithms, see [Section 4.1](#) and [Section 4.1](#).

983 Unlike for the computation of the gold-standard MMSE estimate of the initial inverse problem, the  
 984 denoising posteriors are sampled  $T$  times per trajectory (we use  $T = 1000$ ). To ensure acceptable  
 985 runtimes in this setting, we therefore pick the smallest burn-in period and sample count that still yield  
 986 accurate estimates of the required statistics. We determine these settings with a rigorous protocol that  
 987 is detailed in [Appendix F.4](#). Ultimately, this protocol resulted in the choice of a burn-in period of 100  
 988 iterations and a sample count of 300.

989 **F.3 LEARNING DETAILS**

990 For learned-based denoisers, a noise-conditional neural network with UNet architecture (305 761  
 991 learnable parameters) is trained in an off-line step on the  $N_{\text{train}}$  training signals in a standard setup  
 992 (Adam optimizer with learning rate  $1 \times 10^{-4}$  with exponential decay with factor 0.9999, 100 000  
 993 parameter updates, batch size 10 000). The noise schedule in C-DPS and DiffPIR is defined by the  
 994 two endpoints  $\beta_0 = 1 \times 10^{-4}$  and  $\beta_T = 2 \times 10^{-2}$  with linear equidistant samples in-between. The  
 995 learned variant of DPnP is the “DDS-DDPM” variant ([Xu & Chi, 2024](#), Algorithms 1 and 3) that  
 996 contains an inner denoising-sampling loop. The [arbitrary-precision](#) variant does not require an  
 997 inner loop at all (except for the burn-in period), which makes the [arbitrary-precision](#) variant the  
 998 faster one for this case.

999 **F.4 A PROTOCOL TO DETERMINE THE BURN-IN PERIOD AND DENOISER QUALITY THE  
 1000 NUMBER OF SAMPLES**

1001 As discussed in [Appendix F.2](#), the burn-in period and the number of samples of the Gibbs samplers  
 1002 needs to be chosen appropriately to ensure an acceptable runtimes and a sufficiently small Monte Carlo  
 1003 error [when they serve as the gold-standard samplers of the denoising posteriors that are encountered](#)  
 1004 [in the DPS algorithms](#). We determine the burn-in period and the number of samples through the  
 1005 following protocol that is run in an off-line stage prior to running the benchmark. We synthesize  
 1006  $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \mathbf{n}$  where  $\sigma_t$  is in the range defined by the noise schedule  $\beta$ ,  $\mathbf{x}_0$  is constructed via  
 1007 [\(42\)](#) for all four considered increment distributions, and  $\mathbf{n}$  is some unknown but fixed vector of  
 1008 standard Gaussian noise. For each of the synthesized signals, we then launch  $C = 1000$  parallel  
 1009 Gibbs chains on the corresponding denoising posterior and run those chains for  $N_{\text{sufficient}}$  iterations,  
 1010 where  $N_{\text{sufficient}}$  is a sufficiently large natural number that guarantees that the chains are stationary  
 1011 for at least  $N_{\text{avg}}$  (which is also relatively large) iterations and that, consequently, we can compute  
 1012 precise estimates of various statistics of the posterior distribution from the iterates from the last  $N_{\text{avg}}$   
 1013 iterations across all  $C$  chains.

1014 To determine the burn-in period, we then proceed to calculate a statistic that we can monitor  
 1015 throughout the iterations and that we can compare against the reference statistic. Specifically,  
 1016 denoting with  $\mathbf{X}$  the random variable of the Gibbs sampler, we compute the empirical  
 1017 distribution of the increments at index 32 [like](#)  
 1018 [, that is,  \$\(\mathbf{X}\_{33} - \mathbf{X}\_{32}\)\$](#) . The distribution of differences that is obtained by taking the last  $N_{\text{avg}}$   
 1019 iterations across all  $C$  chains is considered the reference distribution. Then, we compute the  
 1020 Wasserstein-1 distance of that distribution to the one obtained by taking the average across  $N_{\text{avg}}$   
 1021 iterations and all  $C$  in a sliding-window starting from the first Gibbs iterations. This allows  
 1022 us to gauge the burn-in period through a visual  
 1023 inspection of the Wasserstein-1 distance through the Gibbs iterations. In particular, we expect the  
 1024 Wasserstein-1 distance to be large for a number of initial samples where the Gibbs sampler is not  
 1025 stationary and then to oscillate around a small but nonzero value. The value will be nonzero due to the  
 1026 finite sample size. The Wasserstein-1 distance between the reference distribution and the one obtained

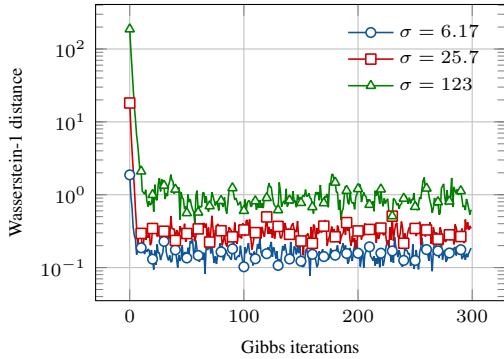


Figure 8: Wasserstein-1 distance of intermediate marginal distributions to that of the final sample.

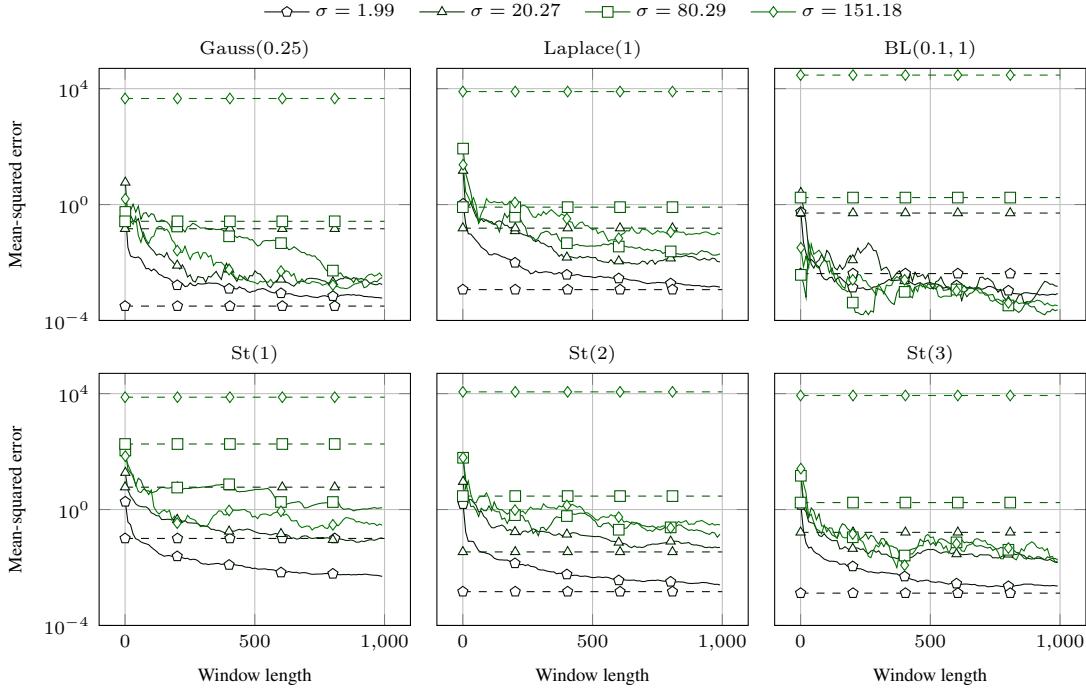


Figure 9: Mean squared error between MMSE estimates and the reference MMSE. Dashed lines: Learned neural MMSE estimate. Solid lines: Monte Carlo MMSE estimate in terms of the window length.

1034 through the Gibbs iterations is shown in Figure 8 (for the exemplary case of a  $St(1)$  distribution and  
1035 a selection of noise variances). We observe that the empirical distribution of increments converges  
1036 rapidly to the reference one. The Wasserstein-1 distance reaches the noise level after a single-digit  
1037 number of iterations, which is in line with the analysis provided by Kuric et al. (2025) in (Kuric et  
1038 al., 2025). Based on these findings, we chose the burn-in period as  $B = 100$  iterations for all our  
1039 experiments, which is more than sufficient to reach stationarity and has acceptable runtime.

1040 To determine the number of samples that are needed for a sufficiently accurate computation of various  
1041 statistics that any DPS algorithm may utilize in their update steps, we compute a precise estimation  
1042 of the MMSE estimate of the denoising posterior by averaging the last  $N_{avg}$  iterations across all  $C$   
1043 chains. Then, we pick one arbitrary chain and grow a window from iteration  $(N_{avg} - 1)$  to the left,  
1044 average the samples in that window, and compute the MSE from the MMSE estimates obtained in  
1045 the one-chain window to the precise estimate obtained by averaging the  $C$  chains and the last  $N_{avg}$   
1046 iterations. We show this error in terms of the window length and the noise variance for all increment  
1047 distributions in Figure 9. The quality of the learned denoiser and the Monte Carlo denoiser differ over  
1048 the noise variances and the learned denoiser improves relative to the Monte Carlo denoiser as the  
1049 noise variance vanishes. Our final choice of  $S = 300$  samples is motivated by the fact that the quality  
1050 of the Monte Carlo denoiser, *when averaged across all noise variances that appear in the reverse*  
1051 *diffusion*, is always strictly better than the learned denoiser. Since it is relevant for the discussion  
1052 in Section 4.2, we highlight that for this choice the quality of the Monte Carlo denoiser is superior  
1053 to the learned one *across all noise variances* for the  $St(1)$  and  $BL(0.1, 1)$  increment distributions.  
1054 Motivated by the training loss of the neural denoisers, we pick a tolerance of  $1 \times 10^{-2}$  and monitor  
1055 at which window length the MSE falls below that tolerance. The results in Figure 8 show that this  
1056 tolerance is consistently reached when the averaging window is 300 samples long, which motivates  
1057 our choice of using  $S = 300$  samples for all our experiments.

[tkeZ TmEt] Nu-  
 anced inspection  
 of the Monte Carlo  
 denoiser.

## 1058 F.5 ALGORITHM PARAMETERS IDENTIFICATION

1059 The adjustable regularization parameter for the method est  $\in \{\ell_2, \ell_1\}$  satisfies was found by

$$\lambda^{\text{est},*} = \arg \min_{\lambda \in \Lambda} \frac{1}{N_{\text{val}}} \sum_{k=1}^{N_{\text{val}}} \frac{1}{d} \|\hat{\mathbf{x}}^{\text{est}}(\mathbf{y}_k^{\text{val}}, \lambda) - \mathbf{x}_k^{\text{val}}\|^2, \quad (68)$$

1060 There, where  $\Lambda$  is the loglinear grid  $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_{N_{\text{mb}}}\}$  with where

$$\lambda_{ni} = 10^{a + (ni-1) \frac{(b-a)}{N_{\text{mb}}-1}} \quad (69)$$

1061 with  $a = (-5)$  and  $b = 5$ . Since the model-based methods are very fast, we can use the relatively  
1062 high  $N_{\text{mb}} = 1000$ .

1063 The adjustable hyperparameters of the DPS methods were found by

$$\lambda^{\text{alg},*} = \arg \min_{\lambda \in \Theta^{\text{alg}}} \frac{1}{N_{\text{val}}} \sum_{k=1}^{N_{\text{val}}} \frac{1}{d} \|\hat{\mathbf{x}}^{\text{alg}}_{\text{MMSE}}(\mathbf{y}_k^{\text{val}}, \lambda) - \mathbf{x}_k^{\text{val}}\|^2 \quad (70)$$

1064 where the grid  $\Theta^{\text{alg}}$  is method-dependent. This Note that this tuning is specifically tailored towards  
1065 the evaluation with respect to the MMSE optimality gap. Due to resource constraints, the parameters  
1066 are tuned for the learned denoiser. We use  $N_{\text{samples}} = 10$  for the grid search on the validation set.  
1067 We define a modest number of  $N_{\text{dps}} = 40$  grid-points and found the extreme points of the grid (i.e.,  
1068 the values of the parameters that clearly lead to worse results) by hand. For C-DPS and DiffPIR,  
1069 we fix the diffusion schedule to standard choices ( $\beta_0 = 1 \times 10^{-4}$ ,  $\beta_T = 0.02$ ). In addition to the  
1070 diffusion schedule, C-DPS has one tunable parameter  $\gamma$  that we tune on 40 loglinear grid points  
1071 ( $ni = 1, \dots, N_{\text{dps}}$ )

$$10^{a + (ni-1) \frac{(b-a)}{N_{\text{dps}}-1}} \quad (71)$$

1072 Twhere,  $a = (-3)$  and  $b = 1$ . DiffPIR has two tunable parameters  $\gamma$  and  $\zeta$ , with although  $\gamma$   
1073 beingis typically considered uncriticalnot so critical. Thus, we split the 40 grid points into a two-  
1074 dimensional grid  $\Theta^{\text{DiffPIR}} = \{0.3, 0.7\} \times \Theta^\zeta$ , withi.e., 2 points for  $\gamma$  and 20 points for  $\zeta$  given by  
1075  $\Theta^\zeta = \{\Theta_1^\zeta, \dots, \Theta_{N_{\text{dps}}/2}^\zeta\}$ , where

$$\Theta_{ni}^\zeta = 10^{a + (ni-1) \frac{(b-a)}{(N_{\text{dps}}/2)-1}} \quad (72)$$

1076 with  $a = (-4)$  and  $b = 1$ . The DPnP algorithm only has the schedule  $\{\eta_t\}_{t=1}^T$  to tune. In this case,  
1077 since DPnP is asymptotically correct, the schedule is a practical vehicle that enables to trade off  
1078 between speed and accuracy. Therefore, thewe use a schedule of this paperthat is similar to the one  
1079 that was proposed by (Xu & Chi, 2024)in the original publication (Xu & Chi, 2024): We fix a small  
1080  $\eta_{\text{final}} = 0.15$ , and linearly decrease  $\eta_{\text{eta}}$  from some  $\eta_{\text{initial}}$  to  $\eta_{\text{final}}$  after  $K/5$  initial iterations with  
1081  $\eta_{\text{initial}}$ , like:

$$\eta_{ni} = \begin{cases} \eta_{\text{initial}} & \text{if } ni = 1, \dots, K/5 \\ \frac{\eta_{\text{final}} - \eta_{\text{initial}}}{K - K/5} ni + \eta_{\text{initial}} & \text{if } ni = K/5 + 1, \dots, K \end{cases} \quad (73)$$

1082 We treat  $\eta_{\text{initial}}$  as a tunable parameter and search over  $\Theta^{\text{DPnP}} = \{\eta_1, \eta_2, \dots, \eta_{40}\}$ , where for  
1083  $i = 1, \dots, 40$ ,

$$\eta_{ni} = 10^{a + (ni-1) \frac{(b-a)}{40-1}}. \quad (74)$$

1084 There, with  $a = (-1)$  and  $b = 4$ . Like in the original publication, we use the comparatively small  
1085  $K = 40$ .1086 The MSE over the validation data depending on the value of the adjustable regularization parameter  
1087 of the  $\ell_2$  and  $\ell_1$  estimators and the adjustable hyperparameters of C-DPS, DiffPIR, and DPnP is  
1088 shown in Figure 10. Since the  $\gamma$  parameter of DiffPIR is assumed to be uncriticalconsidered not so  
1089 critical, we only show the values of the MSE for various choices of  $\zeta$ , where  $\gamma$  is set to the value of  
1090 the optimal  $(\gamma, \zeta)$  pair.

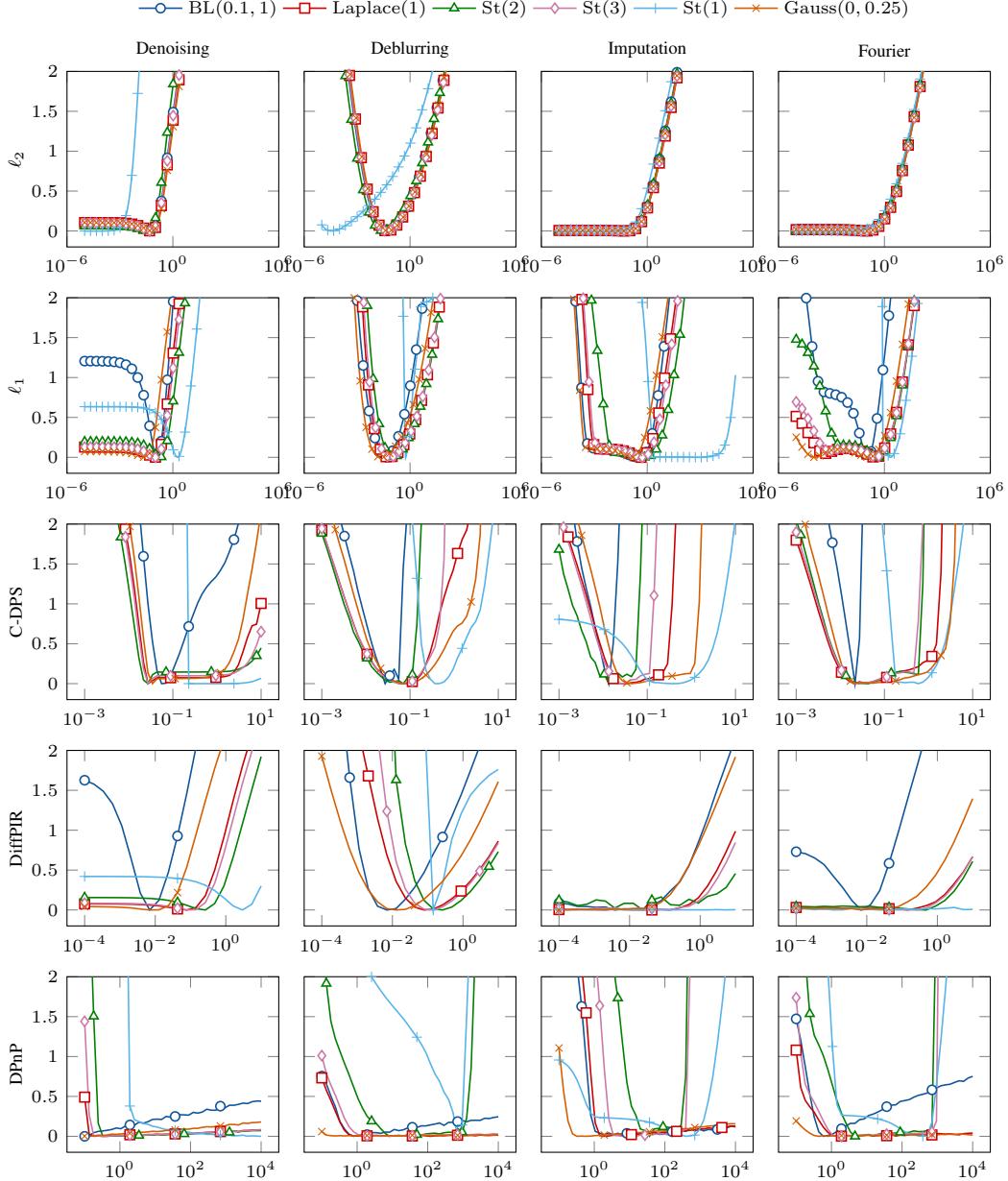


Figure 10: Grid-search diagnostics (logarithm of the MSE over the validation data set) for the model-based methods and the DPS algorithms. Rows:  $\ell_2$ ;  $\ell_1$ ; C-DPS; DiffPIR; DPnP. Columns: Denoising; deconvolution; imputation; reconstruction from partial Fourier measurements. For better visualization, each curve has had its minimum subtracted. To limit clutter, marks are spaced ten apart placed only at every 10th grid point.

Table 4: Change in MMSE optimality gap (mean  $\pm$  standard deviation) after substituting the learned denoiser with the **arbitrary-precisionoraele** denoiser. An asterisk indicates a significant changes according to a Wilcoxon signed-rank test ( $p = 0.05$ ). Negative number with asterisk: MMSE estimates obtained with the **arbitrary-precisionoraele** denoiser are significantly better. Positive number with asterisk: MMSE estimates obtained with the learned denoiser are significantly better.

		Gauss(0.25)	Laplace(1)	BL(0.1, 1)	St(1)	St(2)	St(3)
Denoising	C-DPS	0.00 $\pm$ 0.11	0.00 $\pm$ 0.16	-0.46 $\pm$ 1.16*	0.00 $\pm$ 0.01	0.02 $\pm$ 0.79*	-0.01 $\pm$ 0.14
	DiffPIR	0.00 $\pm$ 0.13	0.00 $\pm$ 0.17	-0.05 $\pm$ 0.78*	-0.41 $\pm$ 0.80*	0.00 $\pm$ 0.20	0.00 $\pm$ 0.15
	DPnP	0.04 $\pm$ 0.27*	-0.01 $\pm$ 0.22	-0.55 $\pm$ 1.31*	-0.77 $\pm$ 1.31*	0.00 $\pm$ 0.24	0.00 $\pm$ 0.23
Deconvolution	C-DPS	-0.01 $\pm$ 0.24	0.00 $\pm$ 0.26	0.09 $\pm$ 0.97*	6.64 $\pm$ 3.21*	-0.12 $\pm$ 1.11*	-0.03 $\pm$ 0.43
	DiffPIR	-0.01 $\pm$ 0.23	0.00 $\pm$ 0.23	0.04 $\pm$ 1.12	13.56 $\pm$ 9.90*	-0.01 $\pm$ 0.47	0.00 $\pm$ 0.31
	DPnP	0.00 $\pm$ 0.25	-0.01 $\pm$ 0.27*	-0.02 $\pm$ 1.20	-4.98 $\pm$ 3.86*	0.06 $\pm$ 0.77	-0.02 $\pm$ 0.34
Imputation	C-DPS	0.00 $\pm$ 0.30	0.01 $\pm$ 0.35	0.41 $\pm$ 1.51*	3.41 $\pm$ 4.99*	-0.12 $\pm$ 1.01*	-0.01 $\pm$ 0.57
	DiffPIR	0.00 $\pm$ 0.29	0.00 $\pm$ 0.33	0.03 $\pm$ 1.05	-0.20 $\pm$ 3.05*	0.03 $\pm$ 0.71	0.00 $\pm$ 0.47
	DPnP	0.00 $\pm$ 0.35	-0.02 $\pm$ 0.38	-0.02 $\pm$ 1.02	-10.46 $\pm$ 5.70*	0.02 $\pm$ 0.67	-0.01 $\pm$ 0.48
Fourier	C-DPS	-0.02 $\pm$ 0.43	-0.01 $\pm$ 0.49	0.80 $\pm$ 1.43*	0.09 $\pm$ 5.63*	-0.03 $\pm$ 0.79*	0.01 $\pm$ 0.49
	DiffPIR	-0.01 $\pm$ 0.39	0.00 $\pm$ 0.40	0.12 $\pm$ 0.83*	-0.64 $\pm$ 1.70*	-0.03 $\pm$ 0.42*	-0.02 $\pm$ 0.38
	DPnP	-0.01 $\pm$ 0.43	0.00 $\pm$ 0.45	-0.33 $\pm$ 1.13*	-1.32 $\pm$ 3.18*	0.00 $\pm$ 0.54	0.01 $\pm$ 0.46

Table 5: Runtime of the benchmark with learned objects.

		Gauss(0.25)	Laplace(1)	BL(0.1, 1)	St(1)	St(2)	St(3)
Denoising	C-DPS	00:04:52	00:04:52	00:02:56	00:04:52	00:04:52	00:04:52
	DiffPIR	00:01:59	00:01:58	00:01:12	00:01:58	00:01:59	00:01:59
	DPnP	00:02:33	00:04:58	00:01:15	00:59:33	00:06:13	00:04:58
Deconvolution	C-DPS	00:04:52	00:04:53	00:02:57	00:04:53	00:04:53	00:04:52
	DiffPIR	00:01:59	00:01:59	00:01:12	00:01:59	00:01:59	00:01:59
	DPnP	00:13:54	00:46:39	00:05:48	00:53:30	00:28:24	00:28:24
Imputation	C-DPS	00:04:53	00:04:53	00:02:59	00:04:53	00:04:53	00:04:53
	DiffPIR	00:01:59	00:01:59	00:01:13	00:01:59	00:01:59	00:01:59
	DPnP	00:04:58	00:16:18	00:18:56	00:51:41	00:39:04	00:32:50
Fourier	C-DPS	00:04:54	00:04:54	00:02:59	00:04:55	00:04:55	00:04:54
	DiffPIR	00:01:59	00:01:59	00:01:13	00:01:59	00:01:59	00:01:59
	DPnP	00:06:13	00:13:53	00:04:42	00:51:41	00:23:39	00:16:18

[TmEt] Runtimes of the experiments.

## 1091 G ADDITIONAL RESULTS

1092 We provide in Table 4 anAn exhaustive quantitative evaluation of the change in the optimality gap  
1093 afterwhen we substituteing the **arbitrary-precision Monte Carlo** denoiser for the learned denoiserfor  
1094 theoracle denoiser is provided in Table 4. WeThe table also reports for which cases the **arbitrary-  
1095 precisionoraele** denoiser enjoysreports significantly better results than the learned denoiser according  
1096 to a Wilcoxon signed-rank test ( $p = 0.05$ ,  $N_{\text{test}}$  pairs, two-sided test with the winner determined  
1097 by the median of differences). We attribute a better performance of the learned denoiser to the fact  
1098 that the algorithms are fine-tuned using the learned component or to the cases where the likelihood  
1099 score approximation is compensated by the one of the learned component. Note that this table must  
1100 be interpreted with the quality of the denoisers in mind. As we show in Figure 9, for our particular  
1101 choice of  $S = 300$  samples, the Monte Carlo denoiser is strictly better than the learned denoiser over  
1102 all noise variances only for signals with  $\text{BL}(0.1, 1)$  and  $\text{St}(1)$  increment distributions.

[tkeZ TmEt] Nu-  
1103 anced inspection  
1104 of the Monte Carlo  
1105 denoiser.

1106 We show uncurated qualitative results of the MMSE estimate obtained by the DPS algorithms and  
1107 the gold-standard Gibbs methods in Figures 11 to 18. The figures alternate between the **arbitrary-  
1108 precisionoraele** denoiser and the learned denoiser and show the results for deconvolution, denoising,  
1109 imputation, and reconstruction from partial Fourier samples, in that order. Each figure contains results  
1110 for  $\text{BL}(0.1, 1)$ ,  $\text{St}(1)$ ,  $\text{St}(2)$ , and  $\text{Laplace}(1)$  increment distributions.

1111 The coverage results for  $\alpha = 0.9$  are presented in Table 7. The Gibbs row again validates the  
1112 implementation; for all forward operators, they achieve coverages that are very close to 0.9. In  
1113 contrast, the coverage values obtained by the DPS algorithms are generally much smaller than 0.9.  
1114 For C-DPS and DiffPIR, the reported coverage values are almost always 0 except for  $\text{BL}(0.1, 1)$   
1115 and  $\text{St}(1)$  increments, where the coverages are usually (close to) 1 for C-DPS and inconsistent for  
1116 DiffPIR. For almost all increment distributions and forward operators, DPnP reports coverage values  
1117 that are closest to, but typically smaller than, 0.9. Note that a coverage of 1 can be considered the  
1118 worst case even at a target of 0.9. For instance, it would beis achieved by setting all samples to a  
1119 constant vector with extremely large (*i.e.*, “unlikely”) entries.

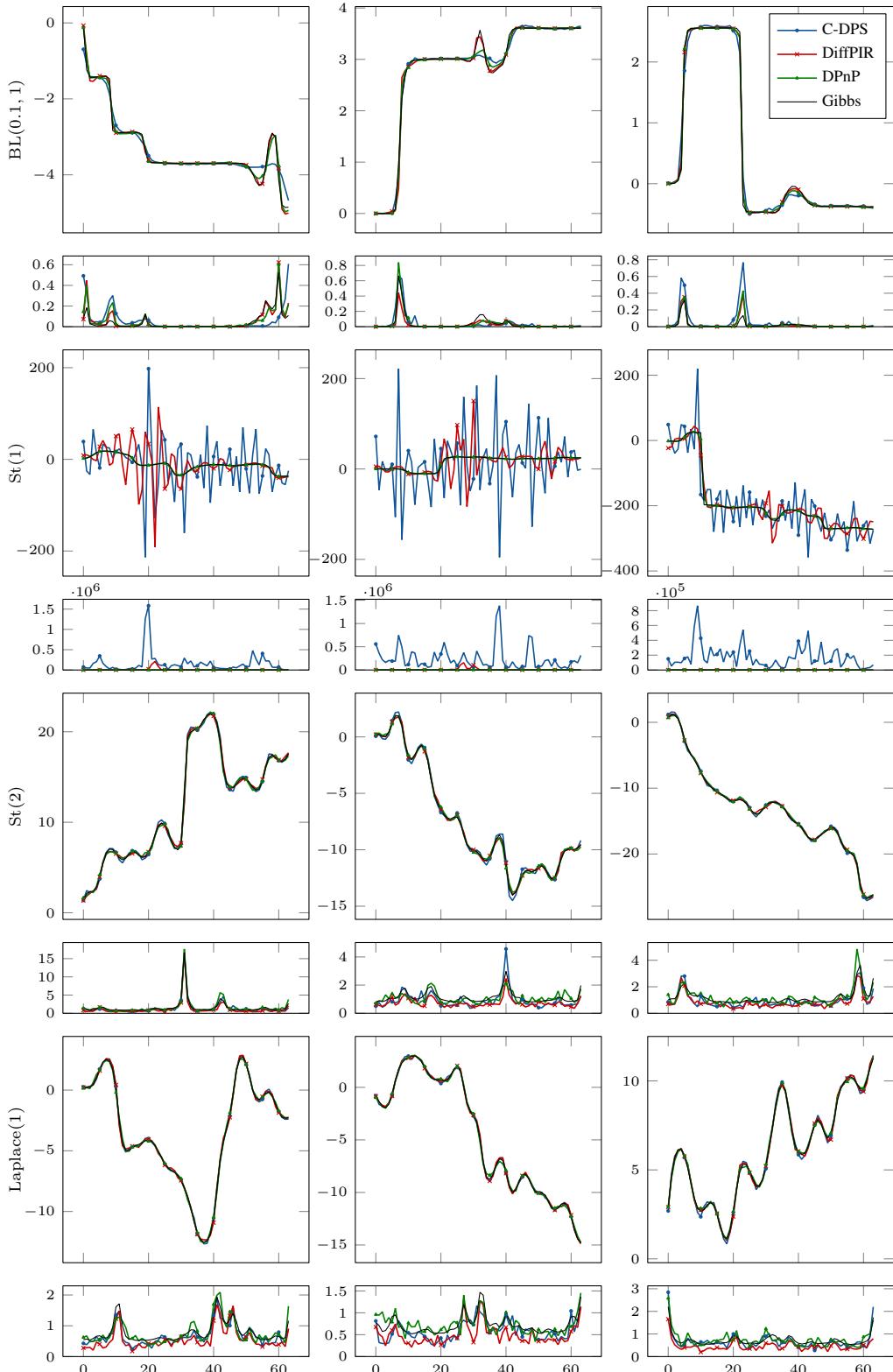


Figure 11: Qualitative results for deconvolution using the [Monte Carlo and denoising sampler](#). Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

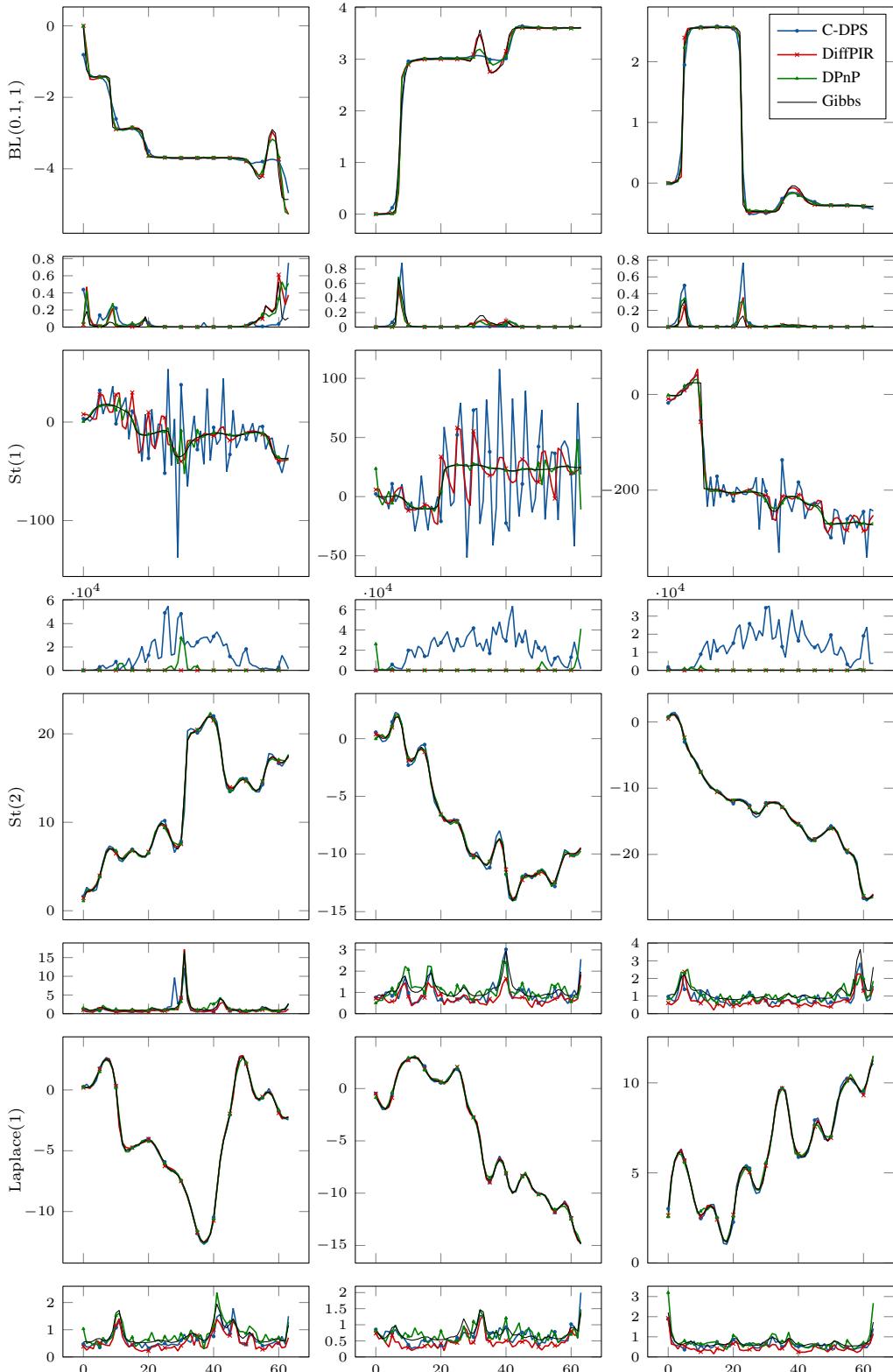


Figure 12: Qualitative results for deconvolution using the learned denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

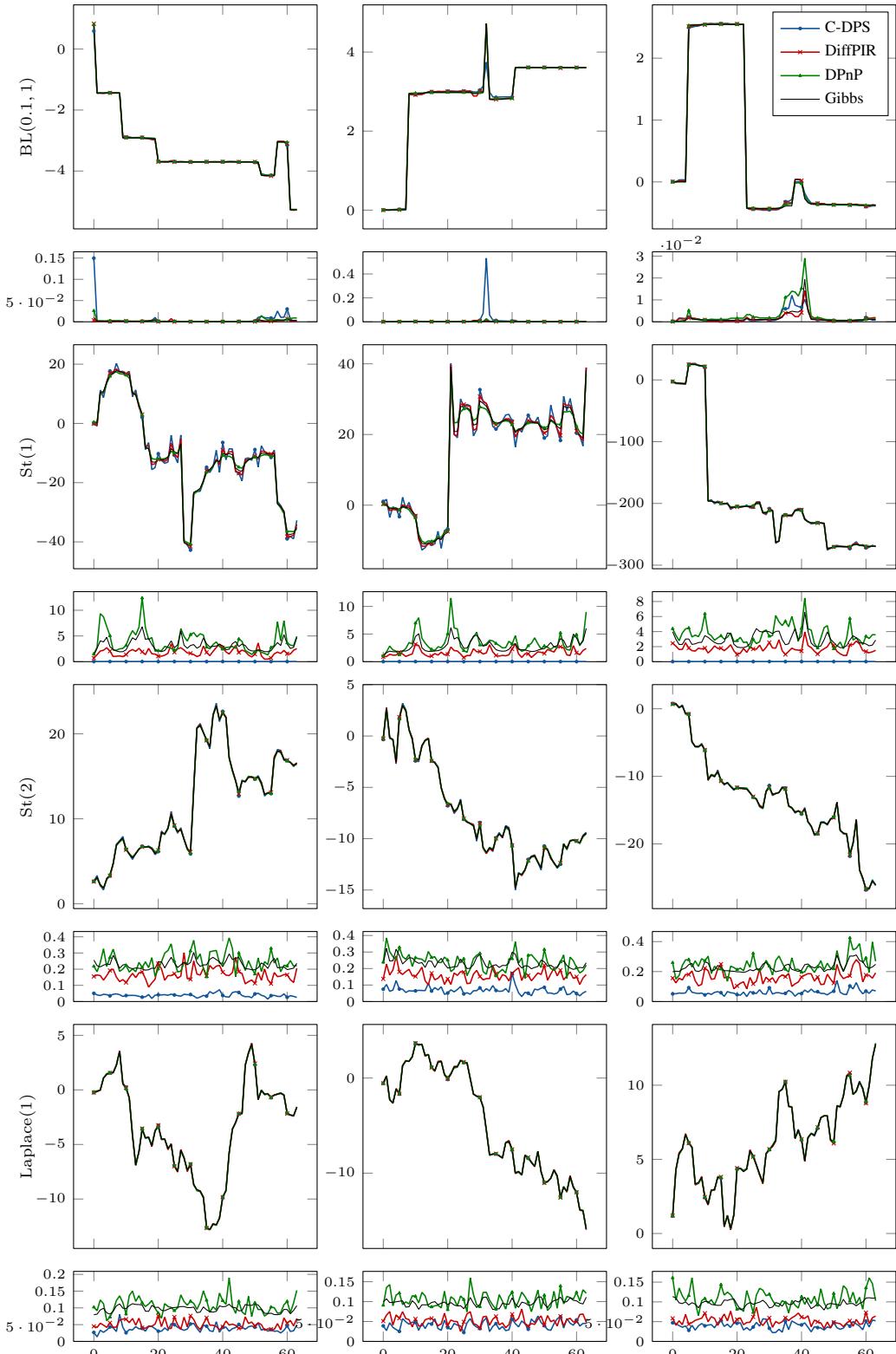


Figure 13: Qualitative results for denoising using the **Monte Carlo and MMSE** denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

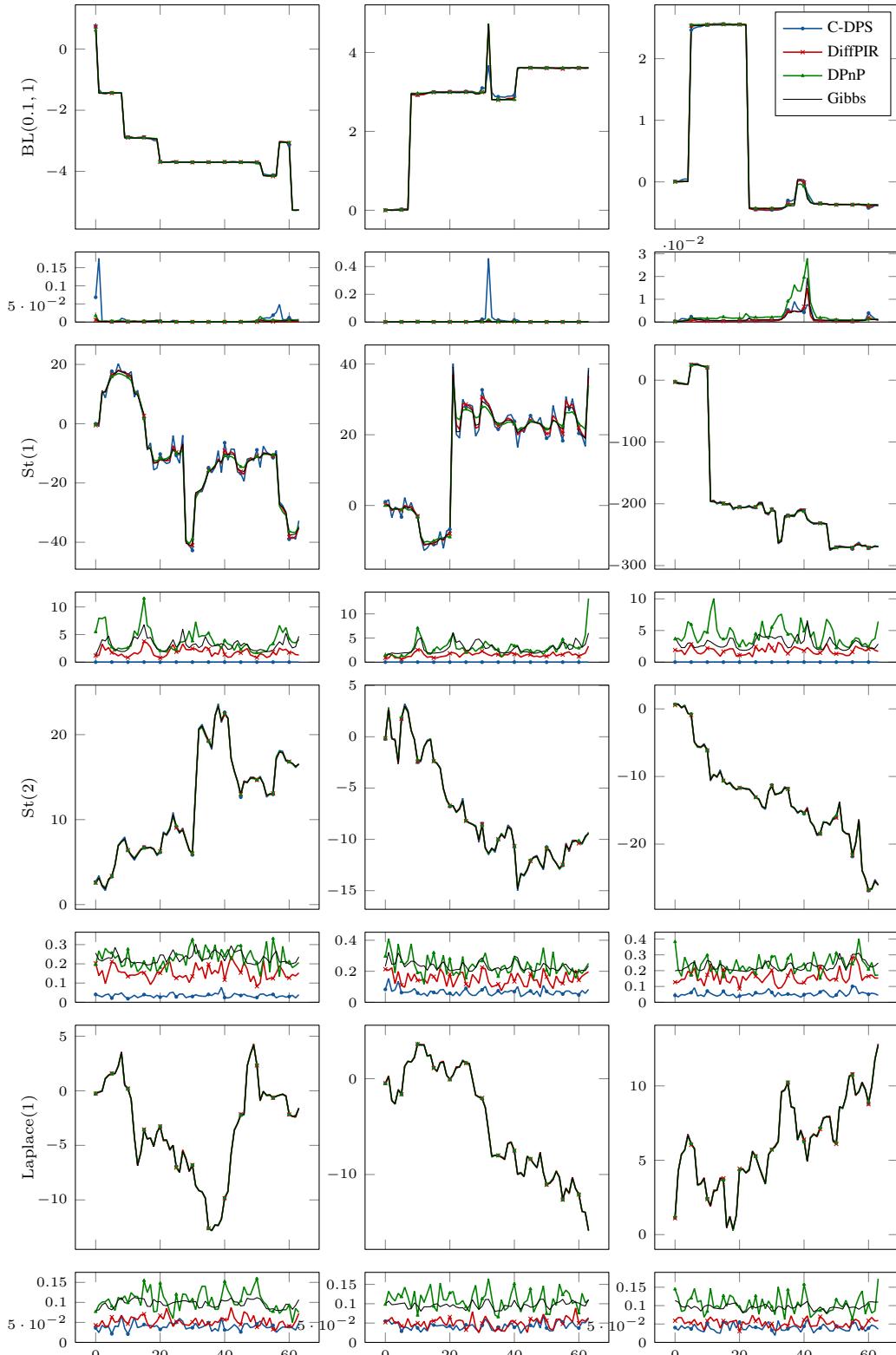


Figure 14: Qualitative results for denoising using the learned denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

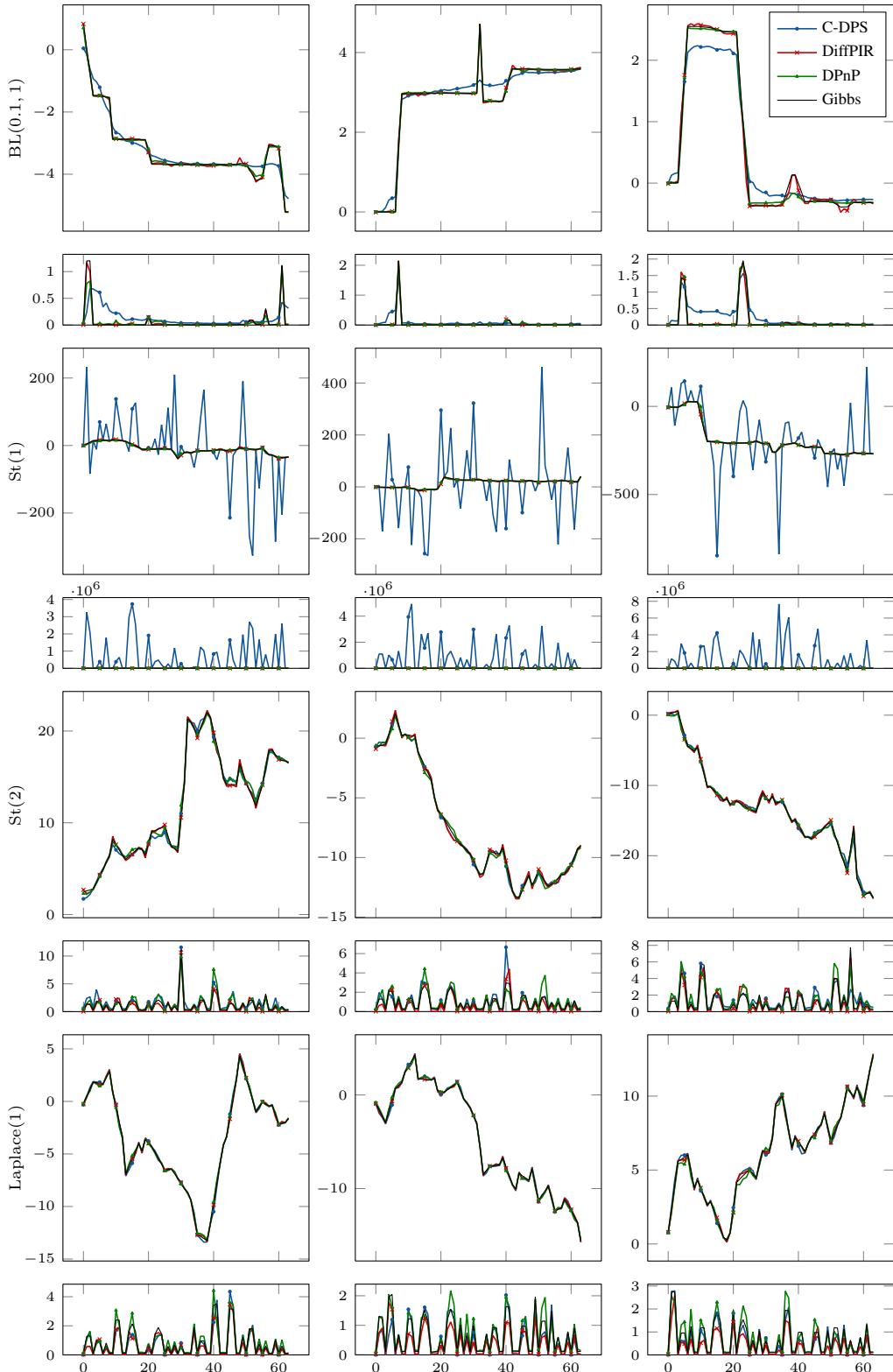


Figure 15: Qualitative results for imputation using the [Monte Carlo denoiseræle sampler](#). Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

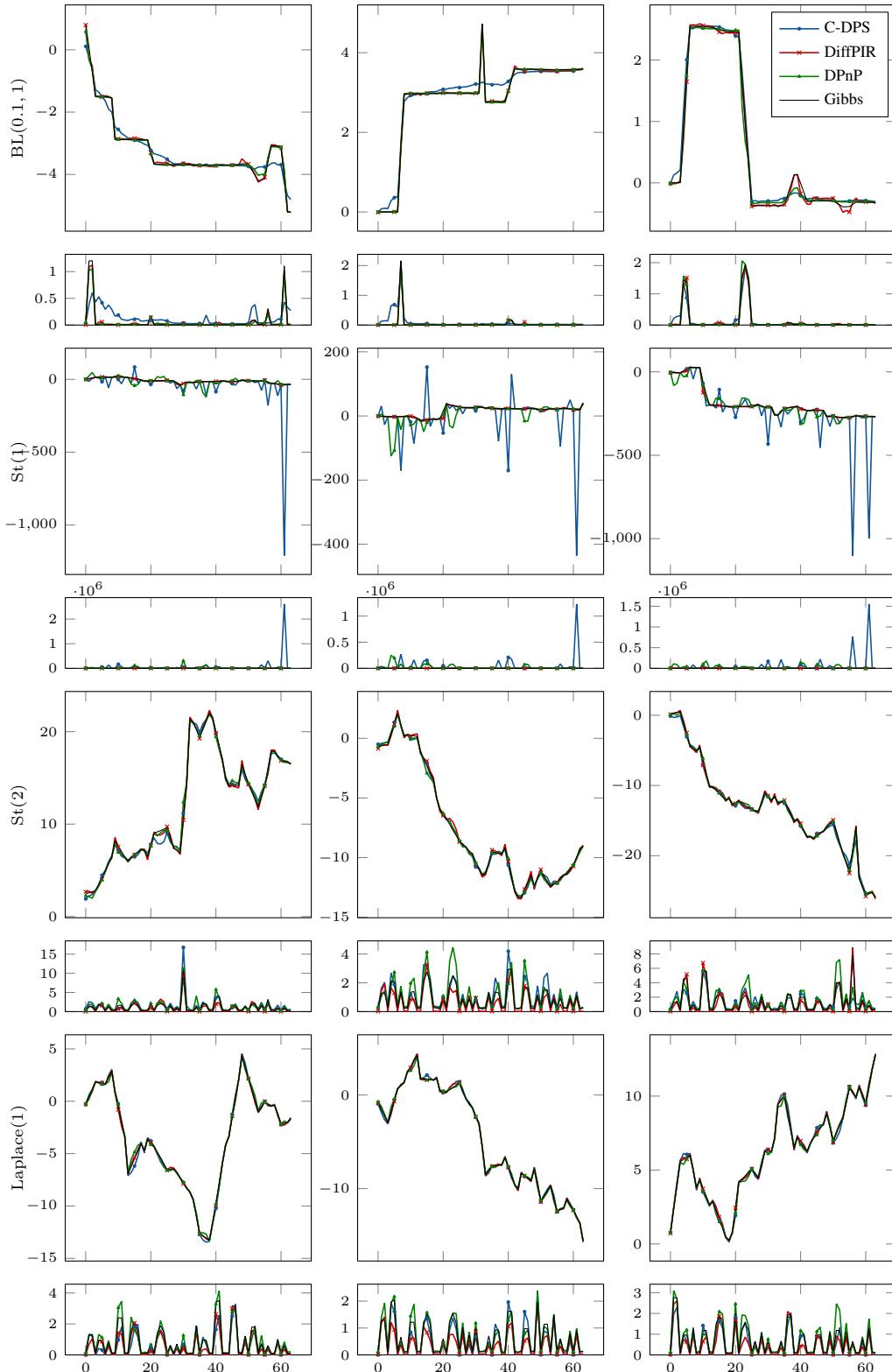


Figure 16: Qualitative results for imputation using the learned denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

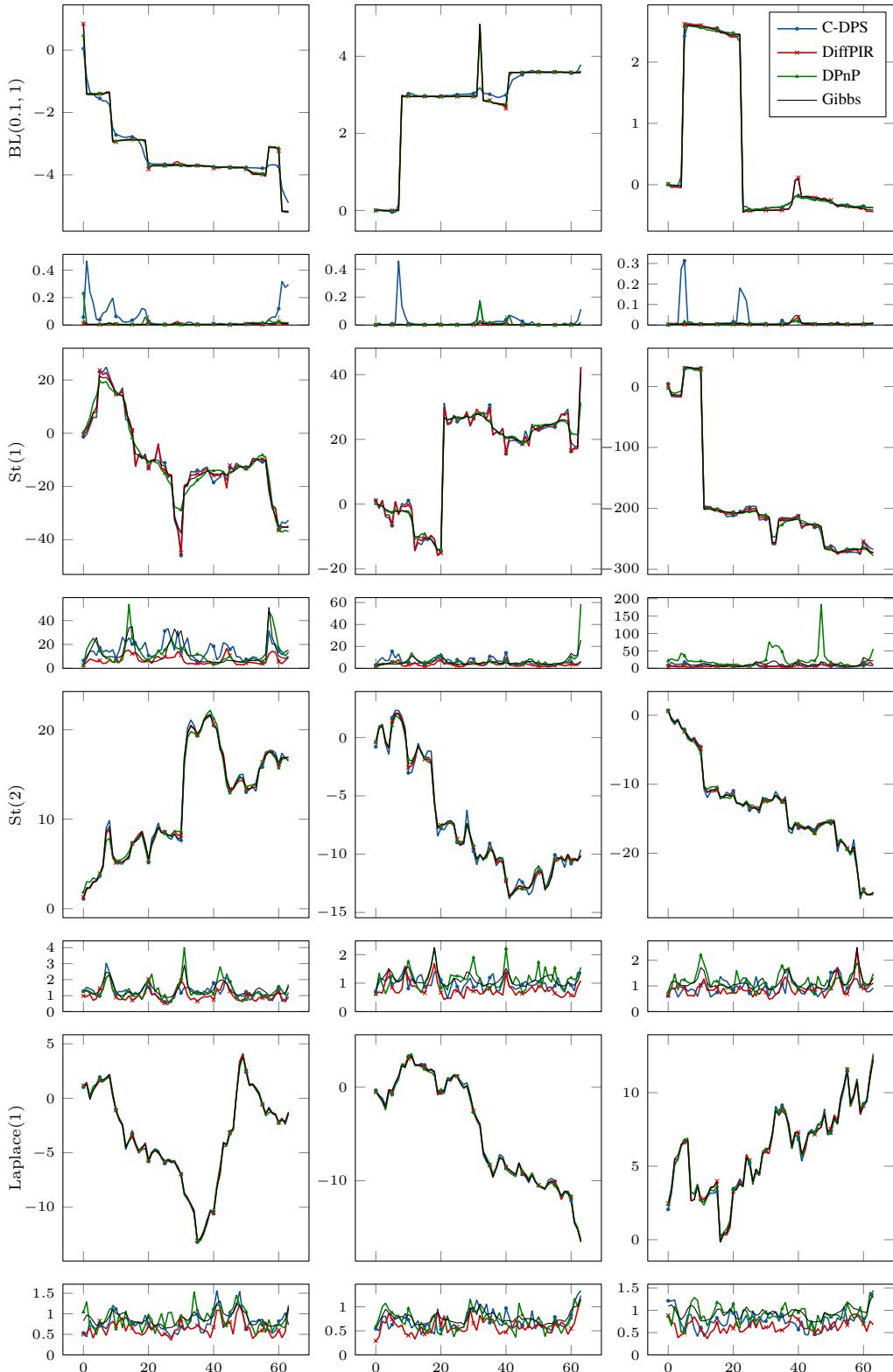


Figure 17: Qualitative results for reconstruction from partial Fourier measurements using the [Monte Carlo oracle](#) denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

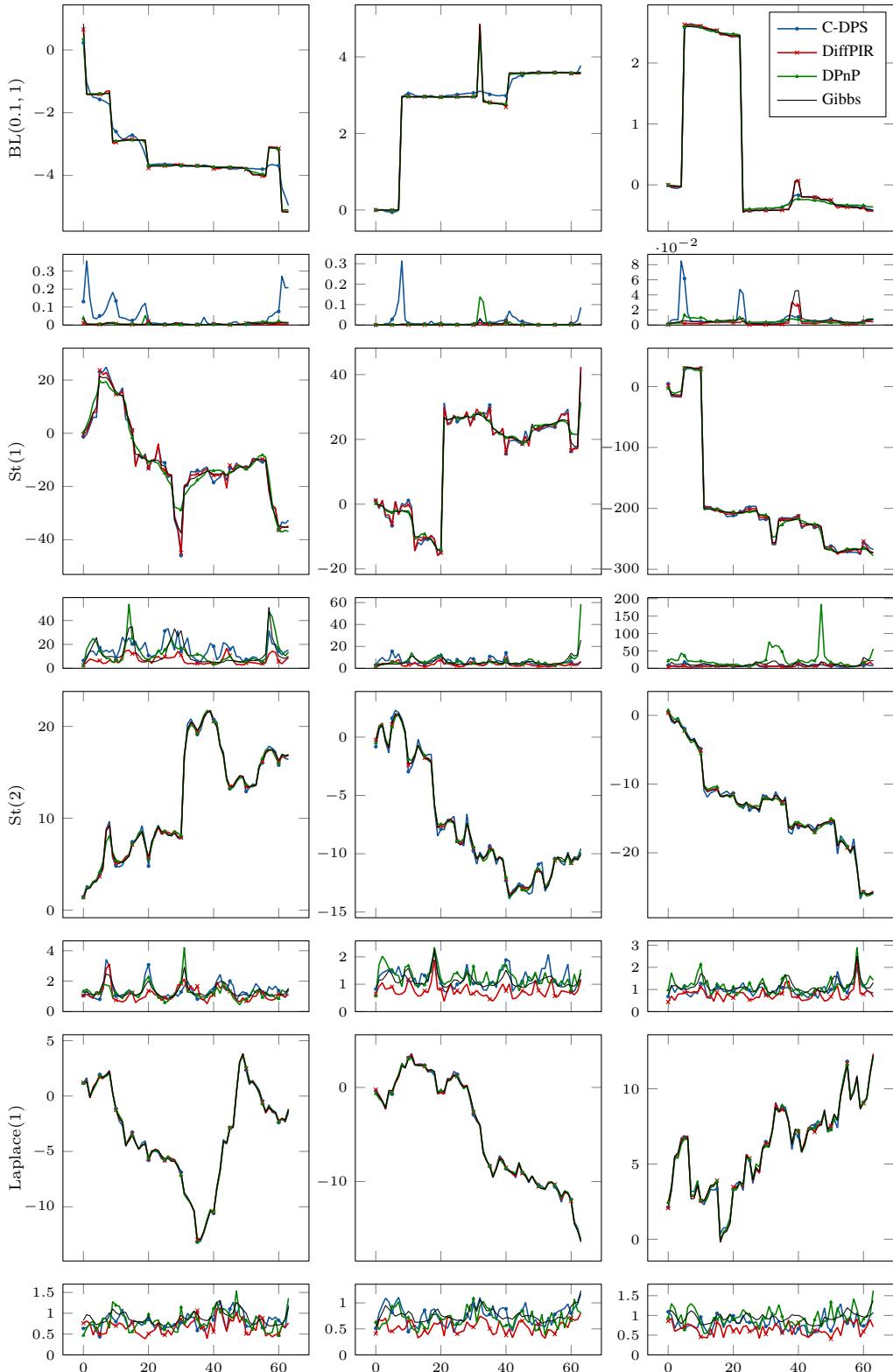


Figure 18: Qualitative results for reconstruction from partial Fourier measurements using the learned denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

Table 6: Runtime of the benchmark with Monte Carlo objects.

		Gauss(0.25)	Laplace(1)	BL(0.1, 1)	St(1)	St(2)	St(3)
Denoising	C-DPS	05:52:28	07:23:23	34:07:44	05:52:40	05:34:51	05:31:10
	DiffPIR	05:04:40	06:36:25	33:46:29	05:12:30	05:39:06	05:38:24
	DPnP	00:03:04	00:03:57	00:20:36	00:03:23	00:03:08	00:03:10
Deconvolution	C-DPS	05:53:40	07:25:17	34:17:12	05:28:38	05:24:12	05:24:00
	DiffPIR	05:28:09	06:55:34	34:16:17	05:31:29	05:32:32	05:22:39
	DPnP	00:03:05	00:03:59	00:21:01	00:03:13	00:03:21	00:03:21
Imputation	C-DPS	05:49:07	07:15:41	34:29:37	05:53:44	05:27:44	05:26:05
	DiffPIR	05:50:15	07:00:13	33:52:26	05:34:00	05:24:16	05:09:56
	DPnP	00:03:23	00:04:18	00:20:58	00:03:09	00:03:05	00:03:22
Fourier	C-DPS	05:49:49	07:09:51	34:30:13	05:49:44	05:49:26	05:49:07
	DiffPIR	05:13:06	06:38:32	34:31:38	05:17:58	06:14:52	05:15:14
	DPnP	00:03:04	00:04:12	00:20:59	00:03:05	00:03:19	00:03:32

[TmEt] Runtimes of the experiments.

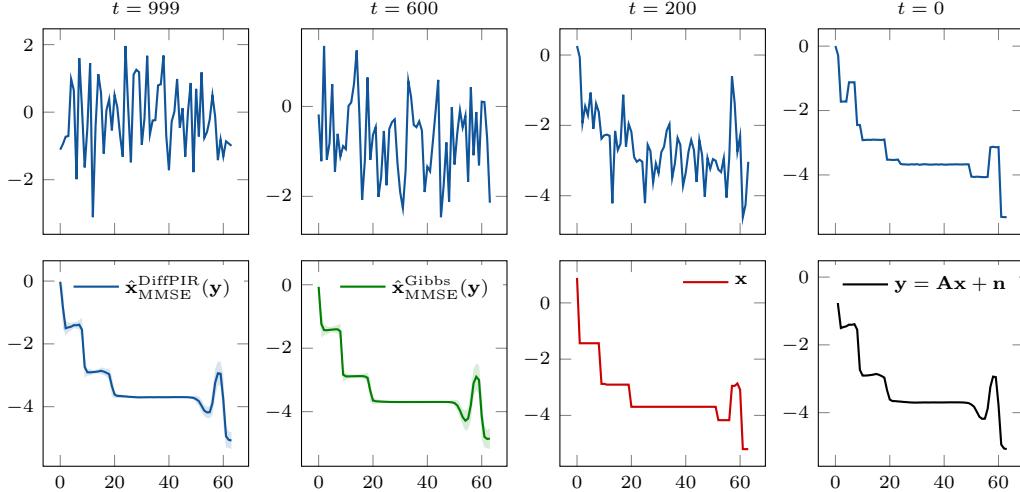


Figure 19: Conditional generation for deconvolution of a signal with  $BL(0.1, 1)$  increments with DiffPIR. Top: Prototypical sampling trajectory at times  $t = 999, 600, 200, 0$ . Bottom: From left to right: MMSE estimate obtained by averaging all DiffPIR samples; gold-standard MMSE estimate obtained by the Gibbs method; the data-generating signal; the data.

Table 7: Posterior coverage of various estimation methods at  $\alpha = 0.9$ . MC: Monte Carlo.

	Gauss(0, 0.25)		Laplace(1)		BL(0.1, 1)		St(1)		St(2)		St(3)		
	Learned	MC <sub>oreale</sub>	Learned	MC <sub>oreale</sub>	Learned	MC <sub>oreale</sub>	Learned	MC <sub>oreale</sub>	Learned	MC <sub>oreale</sub>	Learned	MC <sub>oreale</sub>	
Denoising	Gibbs	—	0.90	—	0.91	—	0.91	—	0.89	—	0.91	—	0.89
	C-DPS	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
	DiffPIR	0.00	0.00	0.00	0.00	1.00	1.00	0.28	0.02	0.00	0.00	0.00	0.00
	DPnP	0.58	0.67	0.11	0.11	1.00	0.41	0.53	0.08	0.09	0.09	0.09	0.10
Deconvolution	Gibbs	—	0.89	—	0.90	—	0.90	—	0.91	—	0.91	—	0.91
	C-DPS	0.00	0.00	0.01	0.00	1.00	1.00	1.00	0.83	0.01	0.00	0.00	0.00
	DiffPIR	0.00	0.00	0.00	0.00	1.00	1.00	0.97	0.92	0.00	0.00	0.00	0.00
	DPnP	0.12	0.12	0.06	0.07	1.00	0.31	0.50	0.06	0.06	0.06	0.07	0.06
Imputation	Gibbs	—	0.89	—	0.90	—	0.86	—	0.91	—	0.91	—	0.91
	C-DPS	0.00	0.00	0.00	0.00	1.00	1.00	0.94	0.78	0.15	0.15	0.00	0.00
	DiffPIR	0.00	0.00	0.00	0.00	1.00	1.00	0.72	0.32	0.00	0.00	0.00	0.00
	DPnP	0.28	0.31	0.09	0.08	1.00	0.41	0.56	0.07	0.14	0.13	0.12	0.13
Fourier	Gibbs	—	0.91	—	0.90	—	0.90	—	0.91	—	0.92	—	0.91
	C-DPS	0.00	0.00	0.00	0.00	1.00	1.00	0.96	0.74	0.01	0.01	0.00	0.00
	DiffPIR	0.00	0.00	0.00	0.00	1.00	1.00	0.92	0.65	0.00	0.01	0.00	0.00
	DPnP	0.19	0.19	0.08	0.06	1.00	0.32	0.50	0.06	0.07	0.07	0.07	0.06

# A STATISTICAL BENCHMARK FOR DIFFUSION POSTERIOR SAMPLING ALGORITHMS

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

1 We propose a statistical benchmark for diffusion posterior sampling (DPS) algo-  
 2 rithms in linear inverse problems. Our test signals are discretized Lévy processes  
 3 whose posteriors admit efficient Gibbs methods. These Gibbs methods provide gold-  
 4 standard posterior samples for direct, distribution-level comparisons with DPS algo-  
 5 rithms. They can also sample the denoising posteriors in the reverse diffusion, which  
 6 enables the arbitrary-precision Monte Carlo estimation of various objects that may  
 7 be needed in the DPS algorithms, such as the expectation or the covariance of the  
 8 denoising posteriors. In turn, this can be used to isolate algorithmic errors from the  
 9 errors due to learned components. We instantiate the benchmark with the minimum-  
 10 mean-squared-error optimality gap and posterior-coverage tests and evaluate pop-  
 11 ular algorithms on the inverse problems of denoising, deconvolution, imputation,  
 12 and reconstruction from partial Fourier measurements. We release the benchmark  
 13 code at <https://github.com/ emblem-saying/dps-benchmark> and  
 14 invite the community to contribute and report results.

## 15 1 INTRODUCTION

16 Diffusion models are among the leading generative models in imaging (Rombach et al., 2022),  
 17 visual computing (Po et al., 2024), finance and time-series analysis (Huang et al., 2024; Rasul et al.,  
 18 2021), de novo protein and drug design (Watson et al., 2023; Alakhdar et al., 2024), natural language  
 19 processing (Li et al., 2022), and other domains. Their ability to model complex distributions has  
 20 motivated their use as priors in the Bayesian resolution of inverse problems. In fact, reconstruction  
 21 methods that leverage diffusion models are competitive or state-of-the-art for problems such as  
 22 deconvolution (Ren et al., 2023), phase retrieval (Xue et al., 2025), magnetic resonance imaging  
 23 and computed tomography reconstruction (Chung & Ye, 2022; Liu et al., 2023), weather-artifact  
 24 removal (Özdenizci & Legenstein, 2023), task-conditioned protein design (Bogensperger et al., 2025),  
 25 audio bandwidth extension and dereverberation (Lemercier et al., 2024), and denoising of financial  
 26 time-series (Wang & Ventre, 2024).

27 This empirical success has come in spite of a lack of a natural mechanism for the conditioning on  
 28 measurements and active research explores how to incorporate the likelihood (Yismaw et al., 2025;  
 29 Erbach et al., 2025). Currently, conditioning strategies are evaluated in one of two ways. (i) With  
 30 respect to downstream applications: As an example, evaluations with respect to perceptual metrics  
 31 such as the structural similarity (Wang et al., 2004), the Fréchet inception distance (Heusel et al.,  
 32 2017), or the learned perceptual image-patch similarity (Zhang et al., 2018) are common in the  
 33 imaging sciences. As pointed out by Pierret & Galerne (2025b) and Cardoso et al. (2024), however,  
 34 these metrics are ill-suited for the statistical evaluation of posterior-sampling algorithms. (ii) In  
 35 overly simplistic settings: A common fallback is to evaluate conditioning strategies in synthetic  
 36 settings with (finite-component) Gaussian-mixture priors. Such mixtures remain light-tailed with  
 37 the tail decreasing exponentially like the widest component. Consequently, they cannot reproduce  
 38 power-law-like extremes that are common in asset returns (Blattberg & Gonedes, 1974; Cont, 2001) or  
 39 statistics of images (Wainwright & Simoncelli, 1999). We illustrate signals with such power-law-like  
 40 extremes later in Figure 3. Benchmarks built on such priors can therefore overstate posterior quality. A  
 41 proper statistical evaluation in realistic settings is critical in high-stakes applications such as medical  
 42 imaging, remote sensing, and finance, where decisions based on reconstructions and their associated  
 43 uncertainties may have significant consequences.

## 44 1.1 CONTRIBUTIONS

45 We propose such a statistical benchmark for diffusion posterior sampling (DPS) algorithms<sup>1</sup> for  
 46 linear inverse problems. Our test signals are discretized Lévy processes that admit efficient posterior-  
 47 sampling algorithms. Indeed, they admit efficient Gibbs methods with exact conditionals that provide  
 48 gold-standard posterior samples. Our framework supports general posterior-level comparisons (*e.g.*,  
 49 (sliced) Wasserstein or energy distances or calibration via coverage and posterior predictive checks)  
 50 by furnishing matched samples obtained from the DPS algorithms and the Gibbs methods.

51 The Gibbs methods are also suited to sample from the denoising posteriors in the reverse diffusion.  
 52 This motivates our contribution of a new template for DPS algorithms, in which update steps utilize  
 53 *samples* from the corresponding denoising posterior. These samples can be used for arbitrary-precision  
 54 Monte Carlo estimation of various objects that are needed in the update steps of the algorithms, such  
 55 as the minimum-mean-squared-error (MMSE) denoiser or its Jacobian, which enables the isolation  
 56 of algorithmic errors from approximation errors due to learned components. We show how several  
 57 popular DPS algorithms can be re-expressed within our template.

58 Finally, we instantiate the framework with the MMSE optimality gap and highest-posterior-density  
 59 coverage checks across the inverse problems of denoising, deconvolution, imputation, and reconstruc-  
 60 tion from partial Fourier measurements. We target the realistic scenario where a learned denoiser is  
 61 used and check hyperparameter sensitivity by substituting the arbitrary-precision Monte Carlo counter-  
 62 parts for the learned components. The benchmark code—which is another substantial contribution—is  
 63 available online. It contains efficient implementations of sampling routines and a containerized run-  
 64 time that allows novel algorithms to be easily benchmarked.

## 65 1.2 RELATED WORK

66 For unconditional sampling, many works derive theoretical bounds on distances between a target  
 67 distribution and the distribution obtained by (approximations of) the reverse stochastic differential  
 68 equation (SDE) (see [Section 2](#)). For example, [Gao et al. \(2025\)](#) bound the Wasserstein-2 distance  
 69 with respect to the discretization error of the SDE under the assumption that the target distribution is  
 70 smooth and log-concave. This directly bounds the number of reverse-diffusion steps needed to obtain  
 71 a desired accuracy. Under absolute continuity of the target with respect to a Gaussian, [Strasman et al.](#)  
 72 ([2025](#)) bound the Kullback–Leibler divergence with respect to properties of the noise schedule.

73 A common assumption that simplifies the analysis and facilitates the computation of various errors and  
 74 bounds is that of a Gaussian target. For example, [Hurault et al. \(2025\)](#) analyze the error incurred when  
 75 using a finite number of prior samples for the estimation of the prior score and track its propagation  
 76 through the iterations of the reverse-SDE solver. [Pierret & Galerne \(2025b\)](#) derive explicit solutions  
 77 to the SDE and use them to derive bounds on the Wasserstein-2 distance to the distributions that are  
 78 obtained via Euler–Maruyama discretizations.

79 For conditional sampling, [Pierret & Galerne \(2025a\)](#) derive expressions for the Wasserstein-2 distances  
 80 between the conditional forward marginals and the distributions induced by specific likelihood  
 81 approximations in the reverse SDE under the assumption of a Gaussian prior. [Crafts & Villa \(2025\)](#)  
 82 systematically evaluate DPS algorithms numerically under the assumption of a (finite-component)  
 83 Gaussian-mixture prior and provide reference objects to the DPS algorithms to ensure a fair evaluation.  
 84 [Cardoso et al. \(2024\)](#) and [Boys et al. \(2024\)](#) also evaluate their algorithms on Gaussian-mixture  
 85 priors. These Gaussian-mixture priors, however, cannot reproduce power-law-like extremes and can  
 86 overstate posterior quality.

87 Beyond diffusion-specific theory, [Thong et al. \(2024\)](#) check the coverage of credible regions produced  
 88 by different Bayesian recovery strategies and find that those that utilize diffusion models often  
 89 under-report uncertainty. A shortcoming of their approach is that they use an empirical distribution  
 90 of images as a surrogate for the prior distribution. Finally, [Bohra et al. \(2023\)](#) also used efficient  
 91 Gibbs methods to obtain gold-standard posterior samples. Their main focus was to quantify the  
 92 quality of neural MMSE estimators with different number of parameters. Our work extends this to  
 93 posterior-level comparisons.

<sup>1</sup>We use “DPS algorithms” as an umbrella term for posterior-sampling methods with diffusion priors. The method due to [Chung et al. \(2023\)](#), often called DPS in the literature, will be referred to later as C-DPS.

94 

## 2 BACKGROUND

95 **Bayesian Linear Inverse Problems** We seek to estimate a signal  $\mathbf{x} \in \mathbb{R}^d$  from the measurements

96 
$$\mathbf{y} = \mathbf{Ax} + \mathbf{n}, \quad (1)$$

97 where the *forward operator*  $\mathbf{A} \in \mathbb{R}^{m \times d}$  models the noiseless linear-measurement acquisition and  
98  $\mathbf{n} \in \mathbb{R}^m$  is additive noise. In the Bayesian resolution of this problem (Stuart, 2010), the signals are  
99 modeled as a random variable, denoted  $\mathbf{X}$ , with values in  $\mathbb{R}^d$  and distribution  $p_{\mathbf{X}}$ , referred to as the  
100 *prior*. Given any measurement  $\mathbf{y}$ , the ultimate goal is to analyze the *posterior*  $p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}$ . It is related to  
the *likelihood*  $p_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}$  and the prior  $p_{\mathbf{X}}$  via Bayes' rule, which states that

101 
$$p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) \propto p_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y})p_{\mathbf{X}}(\mathbf{x}). \quad (2)$$

102 In contrast to classical variational methods (Scherzer et al., 2008), the posterior distribution provides  
103 natural means to quantify uncertainty and can be summarized by various point estimators. We provide  
104 a precise description of point estimators that are relevant in this work in [Appendix A](#).105 For a given signal  $\mathbf{x}$ , the likelihood  $p_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}$  is fully specified by the distribution of the noise. A  
106 common assumption on the noise is that it is a vector of independent and identically distributed (i.i.d.)  
107 Gaussian random variables with mean zero and variance  $\sigma_n^2$ .<sup>2</sup> In this case, the likelihood is given by

108 
$$p_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y}) \propto \exp\left(-\frac{1}{2\sigma_n^2}\|\mathbf{Ax} - \mathbf{y}\|^2\right). \quad (3)$$

109 Thus, once the forward model and the noise distribution are specified, the remaining modeling choice  
110 is the prior. Diffusion models are good candidates due to their ability to encode complex distributions.111 **Diffusion Models** Diffusion models were introduced by Song et al. (2021) by unifying the discrete  
112 approaches proposed by Song & Ermon (2019) and Ho et al. (2020) in a continuous theory based  
113 on SDEs (Klenke, 2020, Chapters 25 and 26). We denote the (diffusion) SDE with *drift coefficient*  
114  $\mathbf{f} : \mathbb{R}^d \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^d$  and *diffusion coefficient*  $g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$  as

115 
$$d\mathbf{X}_t = \mathbf{f}(\mathbf{X}_t, t) dt + g(t) d\mathbf{W}_t, \quad (4)$$

116 where  $\mathbf{W}_t$  is the standard Wiener process. In our setup, the initial condition  $\mathbf{X}_0$  is the random variable  
117 that describes the signal, thus,  $\mathbf{X}_0 = \mathbf{X}$ . Under suitable choices for  $\mathbf{f}$  and  $g$ , the forward process  
118 admits a limiting marginal  $\mathbf{X}_\infty$  as  $t \rightarrow \infty$ . Sampling from  $p_{\mathbf{X}_0}$  can then proceed by simulating the  
119 SDE (4) in reverse with initial condition  $\mathbf{X}_\infty$ . By Anderson's theorem (Anderson, 1982), the reverse  
120 SDE that reproduces the forward marginals satisfies

121 
$$d\mathbf{X}_t = (\mathbf{f}(\mathbf{X}_t, t) - g^2(t)\nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t)) dt + g(t) d\mathbf{W}_t, \quad (5)$$

122 where  $p_{\mathbf{X}_t}$  denotes the density of  $\mathbf{X}_t$  defined by the forward process, and  $dt$  is negative.123 The primary challenge in this approach lies in the computation of the *scores*  $\nabla \log p_{\mathbf{X}_t}$  for all  $t > 0$ .  
124 A fundamental relation known as Tweedie's formula connects the score with the MMSE denoiser: As  
125 we derive rigorously in [Appendix B](#), for  $\mathbf{f}(\mathbf{x}, t) = (-\frac{\beta(t)}{2})\mathbf{x}$  and  $g(t) = \sqrt{\beta(t)}$ , we have that<sup>3</sup>

126 
$$\nabla \log p_{\mathbf{X}_t}(\mathbf{x}) = -\sigma(t)^{-2}(\mathbf{x} - \alpha(t)\mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]), \quad (6)$$

127 where  $\alpha(t) = \exp(-\frac{1}{2} \int_0^t \beta(s) ds)$  and  $\sigma^2(t) = (1 - \alpha^2(t))$ . This yields a practical way to compute  
128  $\nabla \log p_{\mathbf{X}_t}(\mathbf{x})$  through the resolution of the MMSE denoising problem of finding  $\mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]$ .  
129 In standard applications where the goal is the generation of new signals, one typically tackles this  
130 by approximating the map  $(\mathbf{x}, t) \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]$  with a neural network that is learned in an  
131 offline step. In our framework, we can instead obtain arbitrary-precision MMSE denoisers via Gibbs  
132 methods and thereby eliminate approximation errors from a learned surrogate and isolate errors in  
DPS algorithms themselves.133 The implementation of the reverse SDE for generation requires its own time discretization, for instance  
134 with Euler–Maruyama techniques (Higham, 2001). In this work, we will base our backward processes  
135 on the alternative denoising diffusion probabilistic model (DDPM) backward process (starting from  
136  $\mathbf{Gauss}(\mathbf{0}, \mathbf{I})$ )

137 
$$\mathbf{X}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}(\mathbf{X}_t + \beta_t \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t)) + \sqrt{\beta_t} \mathbf{Z}_t, \quad (7)$$

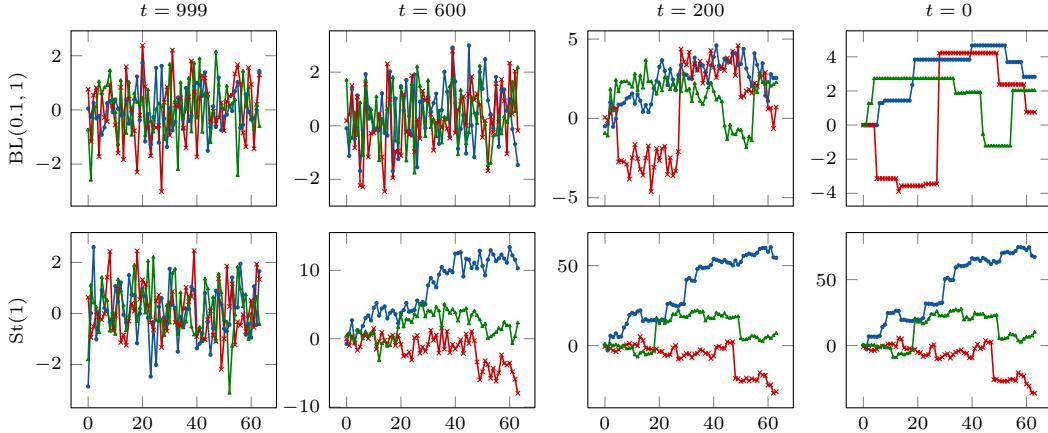


Figure 1: Unconditional reverse-diffusion trajectories obtained by DDPM using the arbitrary-precision Monte Carlo denoiser. Rows: Increment distributions. Columns: Diffusion times. Line styles: Different random states.

133 that originates from the discrete-time Markov chain that was initially proposed by [Sohl-Dickstein](#)  
 134 [et al. \(2015\)](#) and revisited and popularized by [Ho et al. \(2020\)](#). We relate it to the Euler–Maruyama  
 135 discretization of the reverse SDE through Taylor expansions in [Appendix B.1](#).

136 Though we defer details on our signals and  
 137 the Gibbs methods that we use to obtain the  
 138 arbitrary-precision MMSE denoiser to [Section 3](#),  
 139 we demonstrate in [Figure 1](#) that our signals can  
 140 be generated by coupling the unconditional back-  
 141 ward process in [\(7\)](#) with this denoiser. We fur-  
 142 ther motivate this arbitrary-precision denoiser  
 143 in [Figure 2](#) by comparing histograms of signal  
 144 increments produced by the learned denoiser  
 145 and the arbitrary-precision denoiser for a St(1)  
 146 increment target (notations summarized in [Ap-  
 147 pendix C.2](#)). The increments of signals gener-  
 148 ated with the arbitrary-precision denoiser follow the target almost perfectly. Residual errors are due  
 149 to the discretization of the reverse diffusion and Monte Carlo error of the arbitrary-precision denoiser.

150 **Diffusion Posterior Sampling** Our reverse-diffusion sampler can be adapted to sample from a  
 151 posterior by replacing the prior score  $\nabla \log p_{\mathbf{X}_t}$  with the posterior score

$$\nabla \log p_{\mathbf{X}_t | \mathbf{Y}=\mathbf{y}} = \nabla \log p_{\mathbf{X}_t} + \nabla(\mathbf{x} \mapsto \log p_{\mathbf{Y} | \mathbf{X}_t=\mathbf{x}}(\mathbf{y})) \quad (8)$$

152 for some given measurement  $\mathbf{y}$ , obtained by Bayes’ theorem. Although the dependence between  $\mathbf{Y}$  and  
 153  $\mathbf{X}_0$  is known through [\(1\)](#) and the likelihood is explicitly modeled via [\(3\)](#), it is generally challenging to  
 154 relate  $\mathbf{Y}$  and  $\mathbf{X}_t$  for any  $t > 0$ . To overcome this, the conditioning on the measurements is usually done  
 155 in one of two ways. (i) A learned component models the conditional posterior score and also gets the  
 156 measurements as input. This strategy (pursued by [Liu et al. \(2023\)](#); [Özdenize & Legenstein \(2023\)](#);  
 157 [Bogensperger et al. \(2025\)](#); [Saharia et al. \(2023\)](#)) is advantageous when the measurement process  
 158 is unknown, difficult to model, or prohibitively expensive to evaluate. However, its reconstructions  
 159 typically degrade under shifts in measurement conditions since the learned components cannot adapt  
 160 to the new measurement conditions. (ii) The Bayesian separation that is described in [\(8\)](#) is pursued  
 161 and the likelihood score is approximated. This strategy (pursued by [Chung & Ye \(2022\)](#); [Xue et al.](#)  
 162 [\(2025\)](#) and reviewed by [Lemercier et al. \(2024\)](#)) is advantageous when the measurement process  
 163 is known, relatively inexpensive to evaluate, and subject to change, but prior knowledge should be  
 164 reused, which is frequently the case in, e.g., imaging or remote-sensing applications. However, this  
 165 requires approximations to the likelihood score  $\nabla(\mathbf{x} \mapsto \log p_{\mathbf{Y} | \mathbf{X}_t=\mathbf{x}}(\mathbf{y}))$  for all  $t > 0$ .

<sup>2</sup>Our framework supports more general (possibly non-Gaussian) likelihoods, see [Section 3](#).

<sup>3</sup>This is the *variance-preserving* formulation ([Song et al., 2021](#), Section 3.4) with standard normal limiting marginal, where  $\beta : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  controls the speed of the contraction to zero and how much noise is injected.

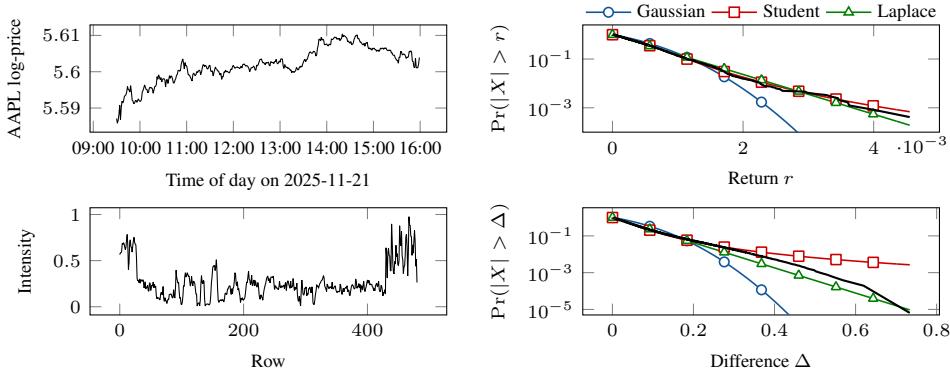


Figure 3: Examples of signals with heavy-tailed increment distributions. Top: Asset returns. Bottom: Columns in natural images. Left: Signals. Right: Survival function of absolute increments (no marker: empirical; markers: best fit to empirical within distribution).

166 Our benchmark can evaluate either strategy, as well as any other method that would claim to sample  
 167 from a posterior distribution like in (2). Approach (i), however, relies on black-box learning of the  
 168 conditional posterior score and its performance heavily depends on various implementation details.  
 169 Thus, we primarily focus on approach (ii), which necessitates approximations of the likelihood  
 170 score (and more general DPS algorithms with explicit conditioning, see our proposed generalization  
 171 in Section 3). For those, our framework can supply arbitrary-precision Monte Carlo estimates of  
 172 various objects to isolate and quantify the impact of these approximations.

### 173 3 PROPOSED FRAMEWORK

174 The prior distributions in our framework will be that of signals of length  $d$  obtained by regularly  
 175 spaced samples of processes with independent, stationary increments (Lévy processes, described in  
 176 Appendix C). Let  $s$  be such a process and let the unit-step increments be  $[\mathbf{u}]_k = (s(k) - s(k-1))$   
 177 for  $k = 1, 2, \dots, d$ . Independence and stationarity imply that the distribution of  $[\mathbf{u}]_k$ , the *increment*  
 178 *distribution*  $p_U$ , does not depend on  $k$ . The increment vector is related to the signal  $\mathbf{x}$  via  $\mathbf{u} = \mathbf{D}\mathbf{x}$ ,  
 179 where  $\mathbf{D}$  is a finite-difference matrix with an initial condition that allows us to write  $\mathbf{x} = \mathbf{D}^{-1}\mathbf{u}$   
 180 where  $\mathbf{D}^{-1}$  is a lower-triangular matrix of ones. This gives a convenient way to synthesize signals  
 181 once  $\mathbf{u}$  is drawn. The independence of the increments implies that the density of the discrete signal is

$$p_{\mathbf{x}}(\mathbf{x}) = \prod_{k=1}^d p_U([\mathbf{D}\mathbf{x}]_k). \quad (9)$$

182 We consider four increment distributions that are commonly used in sparse-process models: Gaussian;  
 183 Laplace; Student-t; and Bernoulli-Laplace (spike-and-slab). Such increment distributions are *sparse*  
 184 or *heavy-tailed* according to the taxonomy due to Unser & Tafti (2014) and are relevant in signal  
 185 and image processing, finance, and other fields (Schoutens, 2003). We show instances signals with  
 186 such heavy-tailed increment distributions in Figure 3. A precise definition of Lévy processes, the  
 187 matrix  $\mathbf{D}$ , the increment distributions and their notation along with a discussion about extensions to  
 188 higher-dimensional signals or signals with more complicated graph structure are given in Appendix C.

189 **Efficient Posterior Sampling** With the prior distribution specified in (9) and the assumption of  
 190 Gaussian noise, the posterior associated to the inverse problem intrinsic in (1) is

$$p_{\mathbf{x}|\mathbf{y}=\mathbf{y}}(\mathbf{x}) \propto \exp\left(-\frac{1}{2\sigma_n^2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2\right) p_{\mathbf{x}}(\mathbf{x}) = \exp\left(-\frac{1}{2\sigma_n^2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2\right) \prod_{k=1}^d p_U([\mathbf{D}\mathbf{x}]_k). \quad (10)$$

191 Unless  $p_U$  is a Gaussian (the simplified setting in Pierret & Galerne (2025b)), this posterior is not  
 192 conjugate, so neither closed-form sampling nor direct evaluation of moments is available. Nevertheless,  
 193 for the increment distributions used in this paper, the posterior distributions admit efficient Gibbs  
 194 methods via standard latent-variable augmentations. Motivation and more details about the Gibbs  
 195 methods, such as the burn-in period  $B$  and the number of samples  $S$ , are provided in Appendix D.

196 The Gaussian, Laplace, and Student-t distributions admit latent representations  
 197 as infinite-component Gaussian mixtures,  
 198 which makes them suitable for the Gaussian  
 199 latent machine (GLM) that was recently  
 200 introduced by [Kuric et al. \(2025\)](#).  
 201 It is generally applicable to distributions

$$p(\mathbf{x}) \propto \prod_{k=1}^n \phi_k([\mathbf{Kx}]_k), \quad (11)$$

203 where  $\mathbf{K} \in \mathbb{R}^{n \times d}$  and all distributions  $\phi_1, \phi_2, \dots, \phi_n : \mathbb{R} \rightarrow \mathbb{R}$  have a latent representation

$$\phi_k(t) = \int_{\mathbb{R}} g_{\mu_k(z), \sigma_k^2(z)}(t) f_k(z) dz, \quad (12)$$

204 where the *latent distribution*  $f_k$  and the *latent maps*  $\mu_k, \sigma_k^2 : \mathbb{R} \rightarrow \mathbb{R}$  depend on the distribution  $\phi_k$ ,  
 205 and  $g_{\mu, \sigma^2}$  is the density of a one-dimensional Gaussian distribution with mean  $\mu$  and variance  $\sigma^2$ . We  
 206 can cast the posterior distribution in (10) into this framework by rewriting it as

$$p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) \propto \prod_{k=1}^m g_{[\mathbf{y}]_k, \sigma_n^2}([\mathbf{Ax}]_k) \prod_{k=1}^d p_U([\mathbf{Dx}]_k) = \prod_{k=1}^{m+d} \phi_k([\mathbf{Kx}]_k). \quad (13)$$

207 There,  $\mathbf{K} = [\mathbf{A}; \mathbf{D}]$ ,  $\phi_k = g_{[\mathbf{y}]_k, \sigma_n^2}$  for  $k = 1, 2, \dots, m$ , and  $\phi_k = p_U$  for  $k = m+1, m+2, \dots, m+d$ .  
 208 Importantly, non-Gaussian likelihoods can be handled by some appropriate definition of the first  $m$   
 209 distributions.

210 The introduction of an appropriate  $n$ -dimensional random variable  $\mathbf{Z}$  with nontrivial distribution (see  
 211 the details in [Kuric et al. \(2025\)](#)) enables the efficient sampling from the conditionals: Sampling  
 212  $\mathbf{X} | \mathbf{Z} = \mathbf{z}$  amounts to sampling a Gaussian with covariance and mean

$$\Sigma(\mathbf{z}) = (\mathbf{K}^\top \Sigma_0(\mathbf{z})^{-1} \mathbf{K})^{-1} \text{ and } \mu(\mathbf{z}) = \Sigma(\mathbf{z}) \mathbf{K}^\top \Sigma_0(\mathbf{z})^{-1} \mu_0(\mathbf{z}), \quad (14)$$

213 respectively, where  $\Sigma_0(\mathbf{z}) = \text{diag}(\sigma_1^2([\mathbf{z}]_1), \dots, \sigma_n^2([\mathbf{z}]_n))$  and  $\mu_0(\mathbf{z}) = (\mu_1([\mathbf{z}]_1), \dots, \mu_n([\mathbf{z}]_n))$ .  
 214 Sampling  $\mathbf{Z} | \mathbf{X} = \mathbf{x}$  amounts to sampling  $n$  independent one-dimensional *conditional latent*  
 215 *distributions*  $p_{[\mathbf{Z}]_1 | \mathbf{X} = [\mathbf{Kx}]_1}, \dots, p_{[\mathbf{Z}]_n | \mathbf{X} = [\mathbf{Kx}]_n}$  that depend on the distributions  $\phi_1, \dots, \phi_n$  and are  
 216 given in [Table 3](#) in the appendix along with the corresponding latent distributions and latent maps.  
 217 We summarize the GLM sampling in [Algorithm 1](#).

218 For the Bernoulli–Laplace increment distribution, we adapt the algorithm proposed by [Bohra et al. \(2023\)](#)  
 219 that introduces two  $d$ -dimensional latent variables: a Bernoulli indicator (“on”/“off”); and  
 220 a Laplace-distributed increment height. For a self-contained exposition, we rigorously derive the  
 221 resulting Gibbs method in [Appendix D.1](#).

222 The Gibbs methods that we just described are  
 223 suitable for the generation of the gold-standard  
 224 samples from the posterior that corresponds to  
 225 the initial inverse problem intrinsic in (1) as  
 226 well as the generation of samples from the de-  
 227 noising posteriors in the DPS algorithms. In the  
 228 latter case, the forward operator  $\mathbf{A}$  is the identity,  
 229 the measurements are the noisy intermediate  
 230 reconstructions  $\mathbf{x}_t$ , and the noise variance  
 231  $\sigma_n^2 = \sigma_t^2$  follows the schedule at timestep  $t$ .

232 When these Gibbs methods are embedded  
 233 within the reverse-diffusion loop, an efficient  
 234 implementation is paramount to achieve ac-  
 235 ceptable runtimes. This is most crucial for the  
 236 Bernoulli–Laplace increment distribution, where the sequential drawing of the binary support vector is  
 237 nested within the Gibbs loop, which in turn may be nested within the reverse-diffusion loop. Accord-  
 238 ingly, we tailored our implementation to modern, highly parallel compute units and optimized several  
 239 components, including custom CUDA- and Triton-compiled sampling routines and incremental  
 240 updates based on the Woodbury–Sherman–Morrison identities. We achieved a cumulative speedup  
 241 of  $74.61 \times$  over the baseline implementation (illustrated in [Figure 4](#) with details in [Appendix D.2](#)).

---

**Algorithm 1** GLM Gibbs method.

---

**Require:**  $\mathbf{x}_0 \in \mathbb{R}^d$ ,  $\mathbf{K} \in \mathbb{R}^{n \times d}$ , conditional la-  
 tent distributions  $\{p_{[\mathbf{Z}]_k | \mathbf{X}}\}_{k=1}^n$  and maps  
 $\{\mu_k, \sigma_k^2\}_{k=1}^n$   
 1: **for**  $s = 1, \dots, B + S$  **do**  
 2:   Draw  $[\mathbf{z}]_k \sim p_{[\mathbf{Z}]_k | \mathbf{X} = [\mathbf{Kx}_{s-1}]_k}$   $\triangleright$  *par. over k*  
 3:   Draw  $\mathbf{x}_s \sim \text{Gauss}(\mu(\mathbf{z}), \Sigma(\mathbf{z}))$   
 4: **return**  $\{\mathbf{x}_{B+s}\}_{s=1}^S$

---

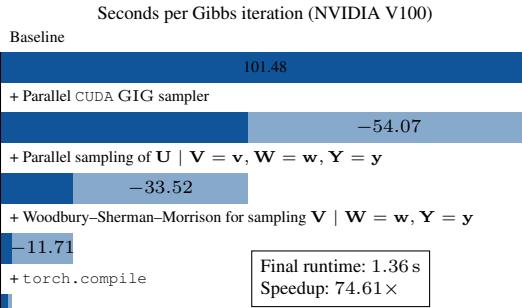


Figure 4: Runtime improvements of the Bernoulli–Laplace sampler.

242 **A Generalized DPS Template** Widely used  
 243 methods, such as diffusion plug-and-play  
 244 (DPnP) (Xu & Chi, 2024), fall outside the pat-  
 245 tern described in Section 2, where one approx-  
 246 imates likelihood score inside the reverse dif-  
 247 fusion. We therefore introduce a simple tem-  
 248 plate that is natural in our setting and accom-  
 249 modates a broader set of DPS algorithms. More  
 250 precisely, we characterize the iteration rule of

251 DPS algorithms as a two-stage process: Given an iterate  $\mathbf{x}_t$  with associated noise variance  $\sigma_t^2$ , the next  
 252 iterate  $\mathbf{x}_{t-1}$  is computed by (i) drawing  $S$  samples denoted  $\{\bar{\mathbf{x}}_k\}_{k=1}^S$  from the denoising posterior  
 253  $p_{\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}_t} \propto \exp\left(-\frac{1}{2\sigma_t^2} \|\cdot - \mathbf{x}_t\|^2\right) p_{\mathbf{X}_0}(\cdot)$ ; and (ii) the subsequent computation of an update step  
 254  $\mathcal{S}$  that may utilize the current iterate  $\mathbf{x}_t$ , the samples  $\{\bar{\mathbf{x}}_k\}_{k=1}^S$ , the measurements  $\mathbf{y}$ , the forward  
 255 operator  $\mathbf{A}$ , and, possibly, other algorithm-internal parameters such as a scalar that weights likeli-  
 256 hood and prior terms or parameters that define the noise schedule. This template is summarized in  
 257 Algorithm 2 and specialized instances for the update step  $\mathcal{S}$  for a variety of popular algorithms are  
 258 given in Appendix E.2. We have absorbed the (variance-preserving) scaling into the step  $\mathcal{S}$  since this  
 259 template is not fundamentally limited to diffusion processes but supports any (also not monotonically  
 260 decreasing) noise schedules. In addition, noise variances  $\{\sigma_t\}_{t=1}^T$  are usually derived from the internal  
 261 parameters  $\lambda$  that may include a noise schedule.

262 Through this construction, DPS algorithms can use any statistic  $R$  of the samples  $\{\bar{\mathbf{x}}_k\}_{k=1}^S$  in their  
 263 update steps. Most methods use the mean  $R(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_S) = \frac{1}{S} \sum_{k=1}^S \bar{\mathbf{x}}_k := \bar{\mu}$ , which is the Monte  
 264 Carlo estimate of  $\mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t]$ . An example of a DPS algorithm that utilizes additional statistics is  
 265 Chung diffusion posterior sampling (C-DPS), which requires the Jacobian of  $\mathbf{x}_t \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t]$ .  
 266 As we show in Appendix E.1, this Jacobian equals (up to the known variance-preserving scaling) the  
 267 conditional covariance of  $\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t$ , an unbiased estimator of which can be obtained through  
 268 the statistic  $R(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_S) = \frac{1}{S-1} \sum_{k=1}^S (\bar{\mathbf{x}}_k - \bar{\mu})(\bar{\mathbf{x}}_k - \bar{\mu})^\top$ . An example of a DPS algorithm that  
 269 utilizes an alternative statistic is the DPnP algorithm that alternately samples from  $p_{\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}_t}$  and a  
 270 data-proximal problem. There,  $R(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_S) = \bar{\mathbf{x}}_1$  is used to obtain one sample from  $p_{\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}_t}$ .  
 271 This statistic is frequently used in the asymptotically exact and the CSGM-type algorithms (using the  
 272 taxonomy due to Daras et al. (2024)). When only a learned MMSE denoiser is available, obtaining this  
 273 one sample requires a full reverse diffusion. In contrast, it requires only one iteration (and the burn-in  
 274 period) with the Gibbs methods. Thus, these algorithms are typically faster when they are endowed  
 275 with the Gibbs methods (see the runtimes in Tables 5 and 6), which enables easy benchmarking.  
 276 However, CSGM-type algorithms typically do not aim at posterior sampling and we do not benchmark  
 277 them here.

278 Since the denoising posteriors are always sub-Gaussian, the Monte Carlo estimation of any object  
 279 enjoys favorable convergence. For instance, the computational complexity of estimating the covariance  
 280 up to a desired precision in the operator norm scales linearly with the dimensionality of the signal  
 281 (Vershynin, 2018, Theorem 4.7.1).

282 **Extensions** A prerequisite for a quantitative evaluation of posterior-sampling algorithms is the  
 283 availability of reasonably efficient samplers that can provide gold-standard samples. The development  
 284 of such samplers for posteriors arising from nonlinear measurement models and non-Gaussian noise  
 285 is challenging, and existing methods currently address only specific cases (e.g., Wang et al. (2017)  
 286 study a nonlinear-Gaussian measurement model with a Laplace prior). Importantly, our framework  
 287 is modular: as more general-purpose samplers for these posteriors become available, they can be  
 288 plugged into our benchmark directly. The denoising posteriors in the reverse diffusion do not change  
 289 with the likelihood and can, therefore, always be efficiently sampled.

290 When going to higher dimensions, the primary challenge lies in the sampling of the high-dimensional  
 291 Gaussian distributions required in the Gibbs methods. Luckily, the structure of the involved operators  
 292 in our case is such that the Gaussians can be efficiently sampled with perturb-and-MAP approaches  
 293 with matrix-free conjugate gradient implementations; we discuss this in more detail and show how  
 294 the runtime of different samplers change with the dimensions in Appendix D.2. Sampling high-  
 295 dimensional Gaussians is a well-studied problem and advances in that field can directly be used in  
 296 our framework.

297 Our gold-standard posterior samples can be compared to samples obtained by *any* posterior-sampling  
 298 algorithm. This includes classical Markov-chain Monte Carlo algorithms, algorithms that utilize  
 299 flow-matching priors, and others. In this work, we primarily focus on DPS algorithms because  
 300 our framework can supply arbitrary-precision Monte Carlo objects to them. We believe that this  
 301 fundamental principle can be extended to other algorithms, in particular those that utilize flow-  
 302 matching priors. Such algorithms are frequently evaluated on toy examples based on Gaussian  
 303 mixtures (*e.g.* by [Pourya et al. \(2025\)](#)), that are overly simplistic.

304 **4 NUMERICAL EXPERIMENTS**

305 We consider signals of dimension  $d = 64$  and four inverse problems that are frequently encountered  
 306 in various estimation tasks throughout the natural sciences: denoising; deconvolution; imputation; and  
 307 reconstruction from partial Fourier measurements. The dimension of the signal is large enough such  
 308 that the corresponding operators can be sensibly defined, yet small enough such that the benchmark has  
 309 acceptable runtimes. We provide experiments about the runtime with larger signals in [Appendix D.2](#),  
 310 details about the operators in [Appendix F.1](#), and precise descriptions of the benchmarking pipeline  
 311 (*e.g.*, the number of training, validation, and test signals, and the number of iterations in the Gibbs  
 312 methods) in [Appendix F.2](#).

313 **4.1 RECONSTRUCTION ALGORITHMS**

314 **Model-Based Methods** We consider the model-based methods

$$\hat{\mathbf{x}}^{\ell_2}(\mathbf{y}, \lambda) = \arg \min_{\mathbf{x} \in \mathbb{R}^d} \left( \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|^2 + \lambda \|\mathbf{Dx}\|^2 \right), \quad (15)$$

315 and

$$\hat{\mathbf{x}}^{\ell_1}(\mathbf{y}, \lambda) = \arg \min_{\mathbf{x} \in \mathbb{R}^d} \left( \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|^2 + \lambda \|\mathbf{Dx}\|_1 \right) \quad (16)$$

316 as baseline reconstruction algorithms. They coincide with the maximum-a-posteriori (MAP) estima-  
 317 tors of Lévy processes associated with Gaussian and Laplace increment distributions, respectively.

318 **Diffusion Posterior Sampling Algorithms** We consider C-DPS ([Chung et al., 2023](#)), diffusion  
 319 models for plug-and-play image restoration ([DiffPIR](#)), ([Zhu et al., 2023](#)) and DPnP ([Xu & Chi,  
 320 2024](#)). This selection demonstrates the applicability of the framework to algorithms that require  
 321 denoising-posterior samples (DPnP), the MMSE denoiser ([DiffPIR](#)), and its Jacobian (C-DPS), which  
 322 covers most of the existing DPS algorithms. For each DPS algorithm, we benchmark a variant that  
 323 uses learned components (learning details are provided in [Appendix F.3](#)) and a variant that uses Gibbs  
 324 samples of the denoising posterior. For DPnP, this fully removes approximation errors. For the others,  
 325 the learned components and the Monte Carlo estimates of those components have varying quality  
 326 for different distributions and noise variances that we systematically investigate in [Appendix F.4](#).  
 327 We provide our main results, the MMSE optimality gap, for the learned variant and then investigate  
 328 changes when we substitute the Gibbs samples for the learned components.

329 The model-based methods and the DPS algorithms require the tuning of some hyperparameters.  
 330 These were found by grid search on validation data independently for each algorithm, increment  
 331 distribution, and forward operator. The precise setup for this grid search is given in [Appendix F.5](#).  
 332 The hyperparameters for the DPS algorithms were tuned to the learned denoiser. Parameters obtained  
 333 with this procedure are later denoted with a star in the superscript.

334 **Gold-Standard Gibbs Methods** The Gibbs methods are used to obtain gold-standard samples from  
 335 the posterior. As described in [Section 3](#), the Gibbs methods are parameter- and bias-free and efficient.  
 336 Consequently, they are well-suited for our purpose. Chain lengths, diagnostics, and implementation  
 337 details are given in [Appendix F.2](#); we reuse the same settings across operators and increment families.

338 **4.2 RESULTS**

339 For any measurement  $\mathbf{y}$ , some DPS algorithm  $\text{alg}$  that depends on the parameters  $\lambda$  will  
 340 produce samples that we denote  $\{\hat{\mathbf{x}}_k^{\text{alg}}(\mathbf{y}, \lambda)\}_{k=1}^{N_{\text{samples}}}$ . We moreover denote  $\hat{\mathbf{x}}_{\text{MMSE}}^{\text{alg}}(\mathbf{y}, \lambda) :=$

Table 1: MMSE optimality gap in decibel (mean  $\pm$  standard deviation; lower is better; 0 is a perfect reconstruction) of various estimation methods over the test set. **Bold**: best among DPS algorithms.

		Gauss(0, 0.25)	Laplace(1)	BL(0.1, 1)	St(1)	St(2)	St(3)
Denoising	C-DPS	<b>0.12 <math>\pm</math> 0.18</b>	0.12 $\pm$ 0.20	2.22 $\pm$ 2.26	3.26 $\pm$ 1.01	0.28 $\pm$ 0.30	0.10 $\pm$ 0.18
	DiffPIR	0.16 $\pm$ 0.21	<b>0.09 <math>\pm</math> 0.16</b>	<b>0.72 <math>\pm</math> 1.10</b>	<b>0.93 <math>\pm</math> 1.06</b>	<b>0.07 <math>\pm</math> 0.14</b>	0.15 $\pm$ 0.21
	DPnP	0.24 $\pm$ 0.25	0.11 $\pm$ 0.17	1.33 $\pm$ 2.12	1.19 $\pm$ 1.38	0.10 $\pm$ 0.17	<b>0.10 <math>\pm</math> 0.17</b>
	$\ell_1$	0.15 $\pm$ 0.21	0.06 $\pm$ 0.12	3.44 $\pm$ 2.38	0.38 $\pm$ 0.43	0.14 $\pm$ 0.19	0.11 $\pm$ 0.18
	$\ell_2$	0.00 $\pm$ 0.01	0.16 $\pm$ 0.21	8.61 $\pm$ 3.10	3.25 $\pm$ 0.99	0.74 $\pm$ 0.83	0.25 $\pm$ 0.33
Deconvolution	C-DPS	0.12 $\pm$ 0.20	0.12 $\pm$ 0.23	4.30 $\pm$ 3.87	18.30 $\pm$ 5.28	0.46 $\pm$ 1.40	0.17 $\pm$ 0.53
	DiffPIR	<b>0.07 <math>\pm</math> 0.17</b>	<b>0.07 <math>\pm</math> 0.19</b>	<b>1.09 <math>\pm</math> 2.22</b>	10.45 $\pm$ 6.10	<b>0.09 <math>\pm</math> 0.57</b>	<b>0.08 <math>\pm</math> 0.26</b>
	DPnP	0.10 $\pm$ 0.18	0.13 $\pm$ 0.22	1.71 $\pm$ 2.49	<b>7.84 <math>\pm</math> 5.66</b>	0.35 $\pm$ 1.39	0.14 $\pm$ 0.41
	$\ell_1$	1.65 $\pm$ 0.84	1.38 $\pm$ 0.86	1.86 $\pm$ 3.14	1.87 $\pm$ 4.01	1.10 $\pm$ 1.19	1.28 $\pm$ 0.94
	$\ell_2$	0.00 $\pm$ 0.01	0.07 $\pm$ 0.23	6.11 $\pm$ 4.49	21.50 $\pm$ 4.46	1.44 $\pm$ 2.85	0.36 $\pm$ 1.09
Imputation	C-DPS	0.15 $\pm$ 0.29	0.18 $\pm$ 0.39	2.99 $\pm$ 2.82	23.33 $\pm$ 8.69	0.50 $\pm$ 1.09	0.14 $\pm$ 0.57
	DiffPIR	<b>0.09 <math>\pm</math> 0.23</b>	<b>0.08 <math>\pm</math> 0.24</b>	<b>0.24 <math>\pm</math> 1.14</b>	<b>0.88 <math>\pm</math> 3.50</b>	<b>0.11 <math>\pm</math> 0.62</b>	<b>0.08 <math>\pm</math> 0.42</b>
	DPnP	0.14 $\pm$ 0.32	0.17 $\pm$ 0.36	0.50 $\pm$ 1.28	10.89 $\pm$ 5.92	0.25 $\pm$ 0.82	0.27 $\pm$ 0.58
	$\ell_1$	1.74 $\pm$ 1.12	1.77 $\pm$ 1.35	1.25 $\pm$ 2.78	13.32 $\pm$ 5.32	1.37 $\pm$ 2.56	1.55 $\pm$ 1.58
	$\ell_2$	0.00 $\pm$ 0.01	0.01 $\pm$ 0.05	1.10 $\pm$ 1.88	0.42 $\pm$ 0.95	0.06 $\pm$ 0.34	0.02 $\pm$ 0.28
Fourier	C-DPS	0.15 $\pm$ 0.36	0.26 $\pm$ 0.65	5.90 $\pm$ 4.41	4.29 $\pm$ 5.78	0.53 $\pm$ 0.83	0.35 $\pm$ 0.77
	DiffPIR	<b>0.11 <math>\pm</math> 0.29</b>	<b>0.08 <math>\pm</math> 0.31</b>	<b>0.83 <math>\pm</math> 1.44</b>	3.19 $\pm$ 4.37	<b>0.11 <math>\pm</math> 0.39</b>	<b>0.12 <math>\pm</math> 0.37</b>
	DPnP	0.11 $\pm$ 0.35	0.20 $\pm$ 0.51	1.88 $\pm$ 2.47	<b>2.45 <math>\pm</math> 4.83</b>	0.39 $\pm$ 0.89	0.24 $\pm$ 0.64
	$\ell_1$	1.50 $\pm$ 1.59	0.73 $\pm$ 0.94	3.57 $\pm$ 2.82	1.07 $\pm$ 2.98	0.71 $\pm$ 0.99	0.78 $\pm$ 0.97
	$\ell_2$	0.00 $\pm$ 0.02	0.36 $\pm$ 0.73	12.22 $\pm$ 4.53	9.47 $\pm$ 8.34	2.66 $\pm$ 3.57	1.03 $\pm$ 1.79

341  $\frac{1}{N_{\text{samples}}} \sum_{k=1}^{N_{\text{samples}}} \hat{\mathbf{x}}_k^{\text{alg}}(\mathbf{y}, \boldsymbol{\lambda})$ . For an estimation method  $\hat{\mathbf{x}}^{\text{est}}(\cdot)$  and data  $\mathbf{y}$  with corresponding  
 342 data-generating signal  $\mathbf{x}$  we measure the MMSE optimality gap (in decibel) defined by

$$10 \log_{10} \left( \frac{\|\hat{\mathbf{x}}^{\text{est}}(\mathbf{y}) - \mathbf{x}\|^2}{\|\hat{\mathbf{x}}^{\text{Gibbs}}(\mathbf{y}) - \mathbf{x}\|^2} \right), \quad (17)$$

343 where  $\hat{\mathbf{x}}^{\text{est}}(\mathbf{y}) = \hat{\mathbf{x}}^{\ell_1 \vee 2}(\mathbf{y}, \lambda^*)$  for model-based methods and  $\hat{\mathbf{x}}^{\text{est}}(\mathbf{y}) = \hat{\mathbf{x}}^{\text{alg}}_{\text{MMSE}}(\mathbf{y}, \boldsymbol{\lambda}^*)$  for DPS  
 344 algorithms. A gap of 0 indicates a perfect recovery of the gold-standard MMSE estimate and any  
 345 positive values show the orders of magnitude of the error relative to the reference error. We found  
 346 that  $N_{\text{samples}} = 50$  provided a good tradeoff between runtime and accuracy by benchmarking the  
 347 gold-standard Gibbs method with that number of samples.

348 We report in [Table 1](#) the mean and standard deviation of the MMSE optimality gap over all signal-  
 349 measurement pairs  $(\mathbf{x}, \mathbf{y})$  in the test set obtained by the model-based methods and the DPS algorithms  
 350 endowed with the learned denoiser. The Gaussian increment distribution validates the implementation:  
 351 Since the MMSE and the MAP point estimates coincide, the model-based  $\ell_2$  estimator matches the  
 352 Gibbs reference up to the error due to the finite parameter-grid resolution. When the posterior  
 353 mean is smooth (*e.g.*, imputation and some deconvolution cases),  $\ell_2$  is the best model-based choice  
 354 and frequently outperforms the DPS algorithms. When the posterior mean is close to piecewise-  
 355 constant (typical in denoising of signals with sparse increments), the  $\ell_1$  estimator is preferred. Among  
 356 DPS algorithms, DiffPIR is typically the top performer and often exceeds  $\ell_2$  and  $\ell_1$  baselines in  
 357 deconvolution, imputation, and reconstruction from partial Fourier measurements. For spike-and-slab  
 358 settings (Bernoulli–Laplace), DPS algorithms substantially outperform the model-based baselines  
 359 across operators. In deconvolution and reconstruction from partial Fourier measurements, DPS  
 360 algorithms frequently match or surpass the best model-based estimator.

361 We now inspect the change in performance after we substitute higher-quality Monte Carlo com-  
 362 ponents for the learned components. We do this *without retuning of the hyperparameters*, which  
 363 allows us to see if the performance of the algorithms increases automatically with the quality of  
 364 the denoiser. Here, we discuss general trends; an exhaustive quantitative evaluation and a precise  
 365 quantification of the quality of the learned and Monte Carlo objects is given in [Appendix G](#). For  
 366 the same hyperparameters, the performance of DPnP increases significantly with the quality of the  
 367 denoising-posterior samples. For example, the optimality gap decreases by 10.46 dB for imputation  
 368 of signals with St(1) increments, and significantly for other measurement operators for signals with  
 369 St(1) increments and BL(0.1, 1) increments. By contrast, C-DPS and DiffPIR can require a retuning  
 370 when the denoiser changes: Scores can deteriorate after one has substituted a higher-quality Monte  
 371 Carlo denoiser for the learned one, but a brief hand-tuning of the hyperparameters on the validation set  
 372 improves them way beyond the learned denoiser. For instance, for DiffPIR and imputation of signals  
 373 with St(1) increments, reusing the hyperparameters deteriorates the gap by 13.56 dB, whereas a brief  
 374 hand-tuning decreased the optimality gap by almost 10 dB over what is reported in [Table 1](#). Qualita-  
 375 tive examples of the MMSE estimates and the marginal variances obtained by the DPS algorithms  
 376 and the gold-standard Gibbs methods are shown in [Figures 11 to 18](#) in the appendix.

377 Prototypical samples and the corresponding  
 378 MMSE estimate obtained from a DPS algorithm  
 379 (here DiffPIR for deconvolution of a signal with  
 380  $BL(0.1, 1)$  increments) are shown in Figure 5.  
 381 (The full conditional reverse-diffusion trajectory,  
 382 the data-generating signal, the measurements,  
 383 and the MMSE estimated obtained with the gold-  
 384 standard Gibbs methods are shown in Figure 19  
 385 in the appendix.) The figure highlights a key dis-  
 386 tinction: Posterior *samples* often preserve high-  
 387 frequency structure and reflect prior variability,  
 388 whereas the *MMSE point estimate*—obtained by  
 389 averaging all samples—is much smoother. This explains why DPS methods tend to score higher on  
 390 perception-oriented metrics, while regressors that target the MMSE point estimate (through training  
 391 with the mean squared error) excel on distortion metrics like the peak signal-to-noise-ratio (PSNR).  
 392 Consistent with this distinction, Saharia et al. (2023) fairly compare a sampling-based method  
 393 to an MMSE regressor and find the expected tradeoff: higher PSNR and structural similarity for  
 394 the regressor; and better perceptual scores for the sampler. We therefore recommend to make the  
 395 Bayesian target explicit—point estimate versus sample quality—and to use evaluation protocols that  
 396 are aligned to that target. Our framework supports this by offering gold-standard posterior samples  
 397 and arbitrary-precision Monte Carlo estimates.

398 In addition to the evaluation of the MMSE optimality gap we analyze the highest-posterior-density  
 399 coverage of the algorithms. Specifically, for any measurement  $y$  and any  $k = 1, 2, \dots, N_{\text{samples}}$ ,  
 400 we define<sup>4</sup>  $l_k(y) := \log p_{\mathbf{X}|\mathbf{Y}=y}(\hat{\mathbf{x}}_{P(k)}^{\text{alg}}(y, \lambda^{\text{alg},*}))$  where  $P$  is the permutation that ensures that  
 401  $l_1(y) \geq l_2(y) \geq \dots \geq l_{N_{\text{samples}}}(y)$  and define the empirical highest-posterior-density threshold at  
 402  $\alpha \in [0, 1]$  as  $l_{\lceil \alpha N_{\text{samples}} \rceil}(y)$ . We declare the data-generating signal  $\mathbf{x}$  covered if  $\log p_{\mathbf{X}|\mathbf{Y}=y}(\mathbf{x}) \geq$   
 403  $l_{\lceil \alpha N_{\text{samples}} \rceil}(y)$  and define the coverage of a method as the fraction of signal-measurement pairs  $(\mathbf{x}, y)$   
 404 in the test set for which  $\mathbf{x}$  is covered by the threshold  $l_{\lceil \alpha N_{\text{samples}} \rceil}(y)$ . The coverage of a calibrated  
 405 posterior-sampling method will be  $\alpha$ , up to Monte Carlo error. A coverage result that is less than  $\alpha$   
 406 indicates that the samples concentrate too heavily around the mode; a coverage result that is greater  
 407 than  $\alpha$  indicates that the samples are too spread out. We again discuss general trends here and present  
 408 an exhaustive quantitative evaluation in Appendix G. The coverages obtained by the DPS algorithms  
 409 are generally much smaller than  $\alpha$ , which indicates that they are uncalibrated and is in line with what  
 410 is reported by Thong et al. (2024). For C-DPS and DiffPIR, the reported coverage values are almost  
 411 always 0 except for  $BL(0.1, 1)$  and  $St(1)$  increments, where the coverages are usually (close to) 1  
 412 for C-DPS and inconsistent for DiffPIR. For almost all increment distributions and forward operators,  
 413 DPnP reports coverage values that are closest to but typically smaller than  $\alpha$ .

## 414 5 CONCLUSION

415 We have introduced a statistical benchmark for diffusion posterior sampling algorithms for linear  
 416 inverse problems. The framework constructs signals with a known distribution, simulates the mea-  
 417 surement process, and subsequently generates samples from the posterior distribution that arises  
 418 through the combination of the known prior and the known likelihood. Gold-standard samples from  
 419 this distribution are obtained via efficient Gibbs methods. These samples are then compared to those  
 420 obtained by the diffusion posterior sampling algorithms. In addition, the Gibbs methods can be  
 421 used to obtain arbitrary-precision Monte Carlo estimates of objects that are needed in the reverse  
 422 stochastic differential equation, such as the minimum-mean-squared-error denoiser or its Jacobian.  
 423 Consequently, the framework also enables the isolation and quantification of the error attributable to  
 424 the likelihood approximations in the conditional reverse diffusion. We have provided numerical results  
 425 for three common diffusion posterior sampling algorithms applied to four common inverse problems.  
 426 A consistent theme across all tested algorithms is that they are not calibrated, which demonstrates that  
 427 research into algorithms that perform better in this respect remains crucial. We invite other researchers  
 428 to benchmark their algorithms on our open implementation.

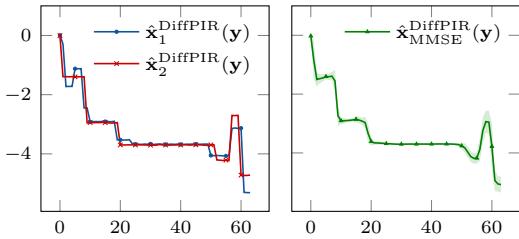


Figure 5: Conditional generation for deconvolution of a signal with  $BL(0.1, 1)$  increments with DiffPIR. The shaded area indicates the variance.

<sup>4</sup>With some slight abuse of notation,  $\log p_{\mathbf{X}|\mathbf{Y}=y}$  is the unnormalized ground-truth log-posterior (10). Since the additive constant is the same across all methods, this ranking is valid.

429 **Reproducibility Statement** We release an online repository with complete algorithm implemen-  
 430 tations and step-by-step instructions to reproduce all results. A containerized runtime enables one-  
 431 command setup and fully automated execution via the provided scripts. Each algorithm is specified at  
 432 a level that supports independent re-implementation: The main text precisely details Gaussian latent-  
 433 machine sampling; and the appendix presents the Bernoulli–Laplace Gibbs method in implementation-  
 434 aligned notation, together with practical optimizations required for acceptable runtimes. The appendix  
 435 also enumerates all experimental settings, including the numbers of training/validation/test signals,  
 436 the samples-per-datum for each sampler, and the exact grid-search procedure used to select hyperpa-  
 437 rameters.

438 **Usage of Large Language Models** We used large language models to adapt passages of already-  
 439 written text for readability and conciseness.

440 REFERENCES

441 Amira Alakhdar, Barnabas Poczos, and Newell Washburn. Diffusion Models in De Novo Drug  
 442 Design. *Journal of Chemical Information and Modeling*, 64(19):7238–7256, September 2024.

443 Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and Their  
 444 Applications*, 12(3):313–326, 1982.

445 Robert Bassett and Julio Deride. Maximum a posteriori estimators as a limit of Bayes estimators.  
 446 *Mathematical Programming*, 174(1–2):129–144, January 2018.

447 Robert C. Blattberg and Nicholas J. Gonedes. A Comparison of the Stable and Student Distributions  
 448 as Statistical Models for Stock Prices. *The Journal of Business*, 47(2):244–280, 1974.

449 Lea Bogensperger, Dominik Narnhofer, Ahmed Allam, Konrad Schindler, and Michael Krauthammer.  
 450 A variational perspective on generative protein fitness optimization. In *Forty-Second International  
 451 Conference on Machine Learning*, 2025.

452 Pakshal Bohra, Pol del Aguila Pla, Jean-François Giovannelli, and Michael Unser. A statistical  
 453 framework to investigate the optimality of signal-reconstruction methods. *IEEE Transactions on  
 454 Signal Processing*, 71:2043–2055, 2023.

455 Benjamin Boys, Mark Girolami, Jakub Pidstrigach, Sebastian Reich, Alan Mosca, and Omer Deniz  
 456 Akyildiz. Tweedie moment projected diffusions for inverse problems. *Transactions on Machine  
 457 Learning Research*, 2024.

458 Gabriel Cardoso, Yazid Janati el idrissi, Sylvain Le Corff, and Eric Moulines. Monte carlo guided  
 459 denoising diffusion models for bayesian linear inverse problems. In *The Twelfth International  
 460 Conference on Learning Representations*, 2024.

461 George Casella and Edward I. George. Explaining the Gibbs sampler. *The American Statistician*, 46  
 462 (3):167–174, August 1992.

463 Hyungjin Chung and Jong Chul Ye. Score-based diffusion models for accelerated MRI. *Medical  
 464 Image Analysis*, 80:102479, 2022.

465 Hyungjin Chung, Jeongsol Kim, Michael Thompson McCann, Marc Louis Klasky, and Jong Chul Ye.  
 466 Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International  
 467 Conference on Learning Representations*, 2023.

468 Christian Clason, Tapani Helin, Remo Kretschmann, and Petteri Piiroinen. Generalized modes in  
 469 bayesian inverse problems. *SIAM/ASA Journal on Uncertainty Quantification*, 7(2):652–684, 2019.

470 Rama Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative  
 471 Finance*, 1(2):223–236, 2001.

472 Evan Scope Crafts and Umberto Villa. Benchmarking diffusion annealing-based Bayesian inverse  
 473 problem solvers. *IEEE Open Journal of Signal Processing*, 6:975–991, 2025.

474 Giannis Daras, Alexandros G. Dimakis, and Constantinos Daskalakis. Consistent diffusion meets  
 475 Tweedie: Training exact ambient diffusion models with noisy data. In *Proceedings of the Forty-First*  
 476 *International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

477 Luc Devroye. Random variate generation for the generalized inverse Gaussian distribution. *Statistics*  
 478 and *Computing*, 24(2):239–246, December 2012.

479 Julius Erbach, Dominik Narnhofer, Andreas Dombos, Bernt Schiele, Jan Eric Lenssen, and Konrad  
 480 Schindler. Solving inverse problems with FLAIR, 2025. arXiv.

481 Xuefeng Gao, Hoang M. Nguyen, and Lingjiong Zhu. Wasserstein convergence guarantees for a  
 482 general class of score-based generative models. *Journal of Machine Learning Research*, 26(43):  
 483 1–54, 2025.

484 Crispin W. Gardiner. *Handbook of Stochastic Methods: For Physics, Chemistry and the Natural*  
 485 *Sciences*. Springer, Berlin, Germany, 1990.

486 Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin.  
 487 *Bayesian Data Analysis*. Chapman and Hall/CRC, November 2013. ISBN 9780429113079.

488 Clément Gilavert, Saïd Moussaoui, and Jérôme Idier. Efficient gaussian sampling for solving large-  
 489 scale inverse problems using mcmc. *IEEE Transactions on Signal Processing*, 63(1):70–80,  
 490 2015.

491 Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs  
 492 trained by a two time-scale update rule converge to a local Nash equilibrium. In *Proceedings of*  
 493 *the Thirty-First International Conference on Neural Information Processing Systems*, NIPS'17, pp.  
 494 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

495 Desmond J. Higham. An algorithmic introduction to numerical simulation of stochastic differential  
 496 equations. *SIAM Review*, 43(3):525–546, 2001.

497 Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Proceedings*  
 498 *of the Thirty-Fourth International Conference on Neural Information Processing Systems*, NIPS  
 499 '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

500 Hongbin Huang, Minghua Chen, and Xiao Qiao. Generative learning for financial time series  
 501 with irregular and scale-invariant patterns. In *The Twelfth International Conference on Learning*  
 502 *Representations*, 2024.

503 Samuel Hurault, Matthieu Terris, Thomas Moreau, and Gabriel Peyré. From score matching to  
 504 diffusion: A fine-grained error analysis in the Gaussian setting, 2025. arXiv.

505 Ajil Jalal, Marius Arvinte, Giannis Daras, Eric Price, Alexandros G Dimakis, and Jon Tamir. Robust  
 506 compressed sensing MRI with deep generative priors. In *Advances in Neural Information*  
 507 *Processing Systems*, volume 34, pp. 14938–14954. Curran Associates, Inc., 2021.

508 Achim Klenke. *Probability Theory: A Comprehensive Course*. Springer International Publishing,  
 509 2020. ISBN 9783030564025.

510 Muhamed Kuric, Martin Zach, Andreas Habring, Michael Unser, and Thomas Pock. The Gaussian  
 511 latent machine: Efficient prior and posterior sampling for inverse problems, 2025. arXiv.

512 Hoang Trieu Vy Le, Marion Foare, Audrey Repetti, and Nelly Pustelnik. Embedding Blake–Zisserman  
 513 regularization in unfolded proximal neural networks for enhanced edge detection. *IEEE Signal*  
 514 *Processing Letters*, 32:1271–1275, 2025.

515 Jean-Marie Lemercier, Julius Richter, Simon Welker, Eloi Moliner, Vesa Välimäki, and Timo Gerk-  
 516 mann. Diffusion models for audio restoration: A review. *IEEE Signal Processing Magazine*, 41(6):  
 517 72–84, 2024.

518 Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-  
 519 LM improves controllable text generation. In *Advances in Neural Information Processing Systems*,  
 520 2022.

521 Jiaming Liu, Rushil Anirudh, Jayaraman J. Thiagarajan, Stewart He, K. Aditya Mohan, Ulugbek S.  
 522 Kamilov, and Hyojin Kim. DOLCE: A model-based probabilistic diffusion framework for limited-  
 523 angle CT reconstruction. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*,  
 524 pp. 10464–10474, 2023.

525 Ozan Özdenizci and Robert Legenstein. Restoring vision in adverse weather conditions with patch-  
 526 based denoising diffusion models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,  
 527 pp. 1–12, 2023.

528 Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming  
 529 Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic Differentiation in PyTorch. In  
 530 *NIPS-W*, 2017.

531 Emile Pierret and Bruno Galerne. Exact evaluation of the accuracy of diffusion models for inverse  
 532 problems with Gaussian data distributions, 2025a. arXiv.

533 Emile Pierret and Bruno Galerne. Diffusion models for Gaussian distributions: Exact solutions and  
 534 Wasserstein errors. In *Proceedings of the Forty-Second International Conference on Machine  
 535 Learning*, 2025b.

536 Rian Po, Wang Yifan, Vladislav Golyanik, Kfir Aberman, Jonathan T. Barron, Amit H. Bermano,  
 537 Eric R. Chan, Tali Dekel, Aleksander Holynski, Angjoo Kanazawa, Karen Liu, Lingjie Liu, Ben  
 538 Mildenhall, Matthias Nießner, Björn Ommer, Christian Theobalt, Peter Wonka, and Gordon Wet-  
 539 zstein. State of the Art on Diffusion Models for Visual Computing. *Computer Graphics Forum*, 43  
 540 (2), April 2024.

541 Mehrsa Pourya, Bassam El Rawas, and Michael Unser. Flower: A flow-matching solver for inverse  
 542 problems, 2025.

543 Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising  
 544 diffusion models for multivariate probabilistic time series forecasting. In *International conference  
 545 on Machine Learning*, pp. 8857–8868. PMLR, 2021.

546 Mengwei Ren, Mauricio Delbracio, Hossein Talebi, Guido Gerig, and Peyman Milanfar. Multiscale  
 547 structure guided diffusion for image deblurring. In *2023 IEEE/CVF International Conference  
 548 on Computer Vision (ICCV)*, pp. 10687–10699, Los Alamitos, CA, USA, October 2023. IEEE  
 549 Computer Society.

550 Severi Rissanen, Markus Heinonen, and Arno Solin. Free hunch: Denoiser covariance estimation  
 551 for diffusion models without extra costs. In *The Thirteenth International Conference on Learning  
 552 Representations*, 2025.

553 Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-  
 554 resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF Conference on  
 555 Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685. IEEE, June 2022.

556 Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi.  
 557 Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and  
 558 Machine Intelligence*, 45(4):4713–4726, 2023.

559 Ken-Iti Sato. *Lévy Processes and Infinitely Divisible Distributions*. Cambridge studies in advanced  
 560 mathematics. Cambridge University Press, Cambridge, England, 1999.

561 Otmar Scherzer, Markus Grasmair, Harald Grossauer, Markus Haltmeier, and Frank Lenzen. *Varia-  
 562 tional Methods in Imaging*. Applied mathematical sciences. Springer, New York, NY, 2009 edition,  
 563 October 2008.

564 Wim Schoutens. *Lévy Processes in Finance*. Wiley Series in Probability and Statistics. John Wiley &  
 565 Sons, Chichester, England, 2003.

566 Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised  
 567 learning using nonequilibrium thermodynamics. In *Proceedings of the Thirty-Second International  
 568 Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp.  
 569 2256–2265, Lille, France, 7 2015. PMLR.

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

572 Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion  
 573 models for inverse problems. In *International Conference on Learning Representations*, 2023.

574 Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution.  
 575 In *Advances in Neural Information Processing Systems*, pp. 11895–11907, 2019.

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

579 Stanislas Strasman, Antonio Ocello, Claire Boyer, Sylvain Le Corff, and Vincent Lemaire. An  
 580 analysis of the noise schedule for score-based generative models. *Transactions on Machine  
 581 Learning Research*, 2025.

582 Andrew M. Stuart. Inverse Problems: A Bayesian Perspective. *Acta Numerica*, 19:451–559, 2010.

583 Yu Sun, Zihui Wu, Yifan Chen, Berthy T. Feng, and Katherine L. Bouman. Provable probabilistic  
 584 imaging using score-based generative priors. *IEEE Transactions on Computational Imaging*, 10:  
 585 1290–1305, 2024.

586 David Y. W. Thong, Charlesquin Kemajou Mbakam, and Marcelo Pereyra. Do Bayesian imaging  
 587 methods report trustworthy probabilities?, 2024. arXiv.

588 Michael Unser and Pouya D. Tafti. *An Introduction to Sparse Stochastic Processes*. Cambridge  
 589 University Press, 2014.

590 David A. van Dyk and Taeyoung Park. Partially Collapsed Gibbs Samplers. *Journal of the American  
 591 Statistical Association*, 103(482):790–796, 2008.

592 Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*.  
 593 Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018.

594 Martin J. Wainwright and Eero P. Simoncelli. Scale Mixtures of Gaussians and the Statistics of  
 595 Natural Images. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press,  
 596 1999.

597 Zheng Wang, Johnathan M. Bardsley, Antti Solonen, Tiangang Cui, and Youssef M. Marzouk.  
 598 Bayesian inverse problems with  $\ell_1$  priors: A randomize-then-optimize approach. *SIAM Journal  
 599 on Scientific Computing*, 39(5):S140–S166, 2017.

600 Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment:  
 601 from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):  
 602 600–612, 2004.

603 Zhuohan Wang and Carmine Ventre. A financial time series denoiser based on diffusion models. In  
 604 *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF ’24, pp. 72–80,  
 605 New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400710810.

606 Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eise-  
 607 nach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita  
 608 Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac  
 609 Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey  
 610 Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David  
 611 Baker. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):  
 612 1089–1100, July 2023.

613 Xingyu Xu and Yuejie Chi. Provably robust score-based diffusion posterior sampling for plug-  
 614 and-play image reconstruction. In *The Thirty-Eighth Annual Conference on Neural Information  
 615 Processing Systems*, 2024.

616 Duoduo Xue, Wenrui Dai, Ziyang Zheng, Xinyu Peng, Junni Zou, and Hongkai Xiong. Fourier phase  
 617 retrieval with diffusion priors guided by failure detection. In *2025 IEEE International Symposium*  
 618 *on Circuits and Systems (ISCAS)*, pp. 1–5, 2025.

619 Nebiyou Yismaw, Ulugbek S. Kamilov, and M. Salman Asif. Covariance-corrected diffusion models  
 620 for solving inverse problems. In *2025 IEEE Statistical Signal Processing Workshop (SSP)*, pp.  
 621 26–30, 2025.

622 Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable  
 623 effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer*  
 624 *Vision and Pattern Recognition*, pp. 586–595, 2018.

625 Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van  
 626 Gool. Denoising diffusion models for plug-and-play image restoration. In *IEEE Conference on*  
 627 *Computer Vision and Pattern Recognition Workshops (NTIRE)*, 2023.

## 628 A BAYES ESTIMATORS

629 A benefit of the Bayesian approach over classical variational methods (see, e.g., (Scherzer et al.,  
 630 2008)) is that different point estimates arise from a fixed prior. For a given measurement  $\mathbf{y}$ , these point  
 631 estimates summarize the posterior distribution  $p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}$  with respect to a given loss  $\ell : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$   
 632 via the optimization problem of finding the point  $\hat{\mathbf{x}}_\ell(\mathbf{y})$  that minimizes the posterior risk:

$$\hat{\mathbf{x}}_\ell(\mathbf{y}) = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^d} \left( \int_{\mathbb{R}^d} \ell(\hat{\mathbf{x}}, \mathbf{x}) p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) d\mathbf{x} \right). \quad (18)$$

633 In this paper, the Bayes estimator with respect to the mean-squared error (MSE)  $\ell = \frac{1}{d} \|\cdot - \cdot\|^2$   
 634 plays a key role due to its close relation to the prior *score* in the reverse diffusion (see Section 2) and  
 635 because we quantify the performance of DPS algorithms via the MMSE optimality gap in Section 4.  
 636 With this choice of  $\ell$ , (18) can be written as

$$\hat{\mathbf{x}}_{\text{MMSE}}(\mathbf{y}) = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^d} \left( \int_{\mathbb{R}^d} \frac{1}{d} \|\hat{\mathbf{x}} - \mathbf{x}\|^2 p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) d\mathbf{x} \right) = \int_{\mathbb{R}^d} \mathbf{x} p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) d\mathbf{x} = \mathbb{E}[\mathbf{X} \mid \mathbf{Y} = \mathbf{y}], \quad (19)$$

637 which is the expectation of the posterior  $p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}$ .

638 Another widely-used estimator arises through the choice

$$\ell(\hat{\mathbf{x}}, \mathbf{x}) = -\chi_{\{\hat{\mathbf{x}}\}}(\mathbf{x}) \quad (20)$$

639 where

$$\chi_A(\mathbf{x}) := \begin{cases} 1 & \text{if } \mathbf{x} \in A, \\ 0 & \text{else,} \end{cases} \quad (21)$$

640 which leads to the MAP estimator that seeks the mode of the posterior:<sup>5</sup>

$$\hat{\mathbf{x}}_{\text{MAP}}(\mathbf{y}) = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^d} \left( \int_{\mathbb{R}^d} -\chi_{\{\hat{\mathbf{x}}\}}(\mathbf{x}) p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) d\mathbf{x} \right) = \arg \max_{\hat{\mathbf{x}} \in \mathbb{R}^d} p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\hat{\mathbf{x}}). \quad (22)$$

641 Rewriting (22) as

$$\hat{\mathbf{x}}_{\text{MAP}}(\mathbf{y}) = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^d} \left( -\frac{1}{2\sigma_n^2} \|\mathbf{A}\hat{\mathbf{x}} - \mathbf{y}\|^2 - \log p_{\mathbf{X}}(\hat{\mathbf{x}}) \right), \quad (23)$$

642 reveals a close relation to classical variational approaches after identifying the regularizer with  
 643  $-\log p_{\mathbf{X}}$ .

<sup>5</sup>This definition is informal but sufficient for the purposes of this paper. For continuous posteriors, the strict 0–1 loss Bayes’ rule is ill-posed. A common formalization defines MAP as the limit of Bayes estimators under shrinking small-ball 0–1 losses; under additional regularity, this limit agrees with the posterior mode (Bassett & Deride, 2018; Clason et al., 2019). The MAP estimator may also not be unique.

## 644 B TWEEDIE'S FORMULA

645 In the setting of [Section 2](#), we now derive an equality that relates  $\nabla \log p_{\mathbf{X}_t}$  to  $\mathbb{E}[\mathbf{X}_0 \mid \mathbf{X}_t = \cdot]$ , *i.e.*,  
 646 the MMSE estimate of  $\mathbf{X}_0$  given that  $\mathbf{X}_t$  takes on a certain value. Similar derivations can be found  
 647 in, *e.g.*, ([Song et al., 2021](#); [Chung et al., 2023](#); [Daras et al., 2024](#)), but we include it to underscore  
 648 the relevance of the MMSE estimate in this paper and to facilitate the understanding of its relation  
 649 to various objects. Under the variance-preserving choice for  $\mathbf{f}(\mathbf{x}, t) = -\frac{\beta(t)}{2}\mathbf{x}$  and  $g(t) = \sqrt{\beta(t)}$   
 650 of the drift and diffusion coefficient, the diffusion SDE (4) simplifies to a time-inhomogeneous  
 651 Ornstein–Uhlenbeck SDE (see [Klenke \(2020, Example 26.5\)](#))

$$d\mathbf{X}_t = -\frac{\beta(t)}{2}\mathbf{X}_t dt + \sqrt{\beta(t)} d\mathbf{W}_t, \quad (24)$$

652 whose pathwise solution

$$\mathbf{X}_t = \alpha(t)\mathbf{X}_0 + \int_0^t \frac{\alpha(t)}{\alpha(s)} \sqrt{\beta(s)} d\mathbf{W}_s, \quad (25)$$

653 where  $\mathbf{X}_0$  is an appropriate initial condition and  $\alpha(t) = \exp\left(-\frac{1}{2} \int_0^t \beta(s) ds\right)$ , can be computed with  
 654 standard techniques, see, *e.g.*, ([Gardiner, 1990](#), Section 4.4.4). In addition, since

$$\int_0^t \left(\frac{\alpha(t)}{\alpha(s)}\right)^2 \beta(s) ds = \int_0^t \beta(s) \exp\left(-\int_s^t \beta(u) du\right) ds = 1 - \alpha^2(t), \quad (26)$$

655 we can write that

$$\mathbf{X}_t = \alpha(t)\mathbf{X}_0 + \sigma(t)\mathbf{N} \quad (27)$$

656 in distribution, where  $\sigma^2(t) = (1 - \alpha^2(t))$ . Consequently, the density of  $\mathbf{X}_t$  is given by the convolution  
 657 of  $p_{\mathbf{X}_0}$  with a Gaussian with variance  $\sigma^2(t)$  and appropriate scaling by  $\alpha(t)$ , which we write as

$$p_{\mathbf{X}_t}(\mathbf{x}) = \int_{\mathbb{R}^d} g_{\mathbf{0}, \sigma(t)^2 \mathbf{I}}(\mathbf{x} - \alpha(t)\hat{\mathbf{x}}) p_{\mathbf{X}_0}(\hat{\mathbf{x}}) d\hat{\mathbf{x}}, \quad (28)$$

658 where  $g_{\mu, \Sigma}(\mathbf{x}) = (2\pi)^{-\frac{d}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \|\mathbf{x} - \mu\|_{\Sigma^{-1}}^2\right)$ . Finally, after taking the gradient, we see  
 659 that

$$\begin{aligned} \nabla p_{\mathbf{X}_t}(\mathbf{x}) &= \int_{\mathbb{R}^d} \nabla g_{\mathbf{0}, \sigma(t)^2 \mathbf{I}}(\mathbf{x} - \alpha(t)\hat{\mathbf{x}}) p_{\mathbf{X}_0}(\hat{\mathbf{x}}) d\hat{\mathbf{x}} \\ &= \int_{\mathbb{R}^d} -\frac{1}{\sigma^2(t)}(\mathbf{x} - \alpha(t)\hat{\mathbf{x}}) g_{\mathbf{0}, \sigma^2(t) \mathbf{I}}(\mathbf{x} - \alpha(t)\hat{\mathbf{x}}) p_{\mathbf{X}_0}(\hat{\mathbf{x}}) d\hat{\mathbf{x}} \\ &= -\frac{1}{\sigma^2(t)} \left( \mathbf{x} p_{\mathbf{X}_t}(\mathbf{x}) - \alpha(t) \int_{\mathbb{R}^d} \hat{\mathbf{x}} g_{\mathbf{0}, \sigma^2(t) \mathbf{I}}(\mathbf{x} - \alpha(t)\hat{\mathbf{x}}) p_{\mathbf{X}_0}(\hat{\mathbf{x}}) d\hat{\mathbf{x}} \right) \\ &= -\frac{1}{\sigma(t)^2} (\mathbf{x} p_{\mathbf{X}_t}(\mathbf{x}) - \alpha(t) p_{\mathbf{X}_t}(\mathbf{x}) \mathbb{E}[\mathbf{X}_0 \mid \mathbf{X}_t = \mathbf{x}]). \end{aligned} \quad (29)$$

660 Finally, after dividing by  $p_{\mathbf{X}_t}(\mathbf{x})$  and since  $\frac{\nabla p_{\mathbf{X}_t}(\mathbf{x})}{p_{\mathbf{X}_t}(\mathbf{x})} = \nabla \log p_{\mathbf{X}_t}(\mathbf{x})$ , we find the celebrated Tweedie  
 661 identity

$$\nabla \log p_{\mathbf{X}_t}(\mathbf{x}) = -\sigma(t)^{-2} (\mathbf{x} - \alpha(t) \mathbb{E}[\mathbf{X}_0 \mid \mathbf{X}_t = \mathbf{x}]). \quad (30)$$

## 662 B.1 A CONNECTION BETWEEN THE DISCRETIZED REVERSE SDE AND DDPM

663 To show the connection between the Euler–Maruyama discretization of the reverse-diffusion SDE  
 664 and the DDPM backward process, we start by deriving the latter from the respective forward process.  
 665 DDPM has been introduced by [Sohl-Dickstein et al. \(2015\)](#) as a discrete-time Markov chain of length  
 666  $T$  with Gaussian transitions

$$p_{\mathbf{X}_t \mid \mathbf{X}_{t-1} = \mathbf{x}_{t-1}} = \text{Gauss}(\sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}), \quad (31)$$

667 such that the transitions from  $\mathbf{X}_0$  to  $\mathbf{X}_t$  are also tractable as

$$\mathbf{X}_t = \sqrt{\alpha_t} \mathbf{X}_0 + \sqrt{1 - \alpha_t} \mathbf{Z}_t, \quad (32)$$

668 where  $\alpha_t = (1 - \beta_t)$ ,  $\bar{\alpha}_t = \prod_{s=0}^t \alpha_s$ , and  $\mathbf{Z}_t \sim \text{Gauss}(\mathbf{0}, \mathbf{I})$ . By definition,

$$\mathbf{X}_t = \sqrt{1 - \beta_t} \mathbf{X}_{t-1} + \sqrt{\beta_t} \mathbf{Z}_{t-1} \quad (33)$$

669 and a straightforward application of Tweedie's formula (6) gives that

$$\mathbb{E}[\mathbf{X}_{t-1}|\mathbf{X}_t] = \frac{1}{\sqrt{\alpha_t}} (\mathbf{X}_t + (1 - \alpha_t) \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t)), \quad (34)$$

670 which leads to the DDPM backward transitions

$$\mathbf{X}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} (\mathbf{X}_t + \beta_t \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t)) + \sqrt{\beta_t} \mathbf{Z}_t \quad (35)$$

671 like they appear in (7).

672 Now, we recall the reverse-diffusion SDE which, under our choice of the drift and diffusion coefficient,  
673 is given by

$$d\mathbf{X}_t = \left( -\frac{\beta(t)}{2} \mathbf{X}_t - \beta(t) \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t) \right) dt + \sqrt{\beta(t)} d\mathbf{W}_t. \quad (36)$$

674 A first-order step from  $t$  to  $(t-1)$  ( $dt = -1$ ) of gives the Euler–Maruyama update

$$\mathbf{X}_{t-1} = \left( 1 + \frac{\beta_t}{2} \right) \mathbf{X}_t + \beta_t \nabla \log p_{\mathbf{X}_t}(\mathbf{X}_t) + \sqrt{\beta_t} \mathbf{Z}_t, \quad (37)$$

675 where  $\beta_t := \beta(t)$  and  $\mathbf{Z}_t \sim \text{Gauss}(\mathbf{0}, \mathbf{I})$ .

676 The DDPM reverse process (35) can be related to the the Euler–Maruyama discretization of the  
677 reverse SDE (37) via Taylor expansions, since

$$\frac{1}{\sqrt{1-\beta_t}} = 1 + \frac{\beta_t}{2} + \mathcal{O}(\beta_t^2) \quad (38)$$

678 and

$$\frac{\beta_t}{\sqrt{1-\beta_t}} = \beta_t + \mathcal{O}(\beta_t^2) \quad (39)$$

679 as  $\beta_t \rightarrow 0$ .

## 680 C LÉVY PROCESSES AND INCREMENT DISTRIBUTIONS

681 The prior distributions in our framework are those of signals obtained by regularly spaced samples of  
682 processes with independent, stationary increments (Lévy processes and their discrete-time counter-  
683 parts). We briefly recall the definition; see [Unser & Tafti \(2014\)](#); [Sato \(1999\)](#) for background and the  
684 link to infinitely divisible laws.

685 **Definition C.1** (Lévy process). A stochastic process  $s = \{s(t) : t \geq 0\}$  is a Lévy process if

- 686 1. (anchor at the origin) It holds that  $s(0) = 0$  almost surely;
- 687 2. (independent increments) for any  $N \in \mathbb{N} \setminus \{0, 1\}$  and  $0 \leq t_1 < t_2 < \dots < t_N < \infty$ , the  
688 increments  $(s(t_2) - s(t_1)), (s(t_3) - s(t_2)), \dots, (s(t_N) - s(t_{N-1}))$  are mutually independent;
- 689 3. (stationary increments) for any given step  $h$ , the increment process  $u_h = \{s(t) - s(t-h) :  
690 t > h\}$  is stationary;
- 691 4. (stochastic continuity) for any  $\varepsilon > 0$  and  $t \geq 0$ ,

$$\lim_{h \rightarrow 0} \Pr(|s(t+h) - s(t)| > \varepsilon) = 0.$$

692 We form discrete and finite-length signals by sampling  $s$  at integer times and stacking the values  
693 into  $\mathbf{x} = (s(1), s(2), \dots, s(d))$ . Let the unit-step increments be  $[\mathbf{u}]_k = (s(k) - s(k-1))$  for  
694  $k = 1, 2, \dots, d$ . By independence and stationarity, the law<sup>6</sup> of  $[\mathbf{u}]_k$  does not depend on  $k$  and we  
695 denote it  $p_U$ . We define the finite-difference matrix

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \quad (40)$$

<sup>6</sup>For our choices, it always has a density w.r.t. a suitable reference measure.

Table 2: Univariate distributions used throughout this work. Parameters appear in the order they are specified in this table, *e.g.*  $\text{Gauss}(\mu, \sigma^2)$ .

Name	Distribution	Parameter(s)	Supp.	Notation
Gaussian	$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right)$	$\mu \in \mathbb{R}, \sigma^2 \in \mathbb{R}_{>0}$	$\mathbb{R}$	Gauss
Exponential	$\lambda \exp(-\lambda x)$	$\lambda \in \mathbb{R}_{>0}$	$\mathbb{R}_{\geq 0}$	Exp
Laplace	$\frac{1}{2b} \exp\left(-\frac{ x }{b}\right)$	$b \in \mathbb{R}_{>0}$	$\mathbb{R}$	Laplace
Student-t	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi\nu}\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$	$\nu \in \mathbb{R}_{>0}$	$\mathbb{R}$	St
Gamma	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x)$	$\alpha, \beta \in \mathbb{R}_{>0}$	$\mathbb{R}_{>0}$	Gamma
Gen. inv. Gaussian	$\frac{(\frac{a}{b})^{\frac{p}{2}}}{2K_p(\sqrt{ab})} x^{p-1} \exp\left(-\frac{ax+b/x}{2}\right)$	$a, b \in \mathbb{R}_{>0}, p \in \mathbb{R}$	$\mathbb{R}_{>0}$	GIG
Bernoulli–Laplace	$\lambda\delta(x) + (1-\lambda)\frac{1}{2b} \exp\left(-\frac{ x }{b}\right)$	$\lambda \in [0, 1], b \in \mathbb{R}_{>0}$	$\mathbb{R}$	BL

Moreover, the gamma function is defined as  $\Gamma(x) = \int_0^\infty t^{x-1} \exp(-t) dt$  for any  $x \in \mathbb{R}_{>0}$ .  
The modified Bessel function of the second kind with parameter  $\nu$  is denoted by  $K_\nu$ .

696 such that the increment vector satisfies

$$\mathbf{u} = \mathbf{D}\mathbf{x}. \quad (41)$$

697 Because  $s(0) = 0$ , the finite-difference matrix  $\mathbf{D}$  has an initial condition that makes it invertible and  
698  $\mathbf{D}^{-1}$  is a lower-triangular matrix of ones. This also implies that for all  $k = 1, 2, \dots, d$ ,

$$[\mathbf{x}]_k = \sum_{n=1}^k [\mathbf{u}]_n, \quad (42)$$

699 which is a convenient way to synthesize signals once  $\mathbf{u}$  is drawn. The combination of (41) with the  
700 independence of the increments implies that the density of the discrete signal is

$$p_{\mathbf{x}}(\mathbf{x}) = \prod_{k=1}^d p_U([\mathbf{D}\mathbf{x}]_k). \quad (43)$$

### 701 C.1 EXTENSIONS

702 The approach in this paper can be extended to two- or higher-dimensional signals on grids, such as  
703 images or videos, and even to more specialized structures like signals defined over trees or graphs.  
704 The structure of the signal is effectively encoded through the choice of the matrix  $\mathbf{D}$ . For instance, a  
705 two-dimensional finite-difference matrix would result in a signal vector that can be interpreted as a  
706 two-dimensional image. The main additional (computational) challenge is sampling during signal  
707 generation: Whenever  $\mathbf{D}$  is not trivially reducible to a one-dimensional operator, the model (43)  
708 will be overcomplete and, in general, no whitening transformation exists to decouple increments for  
709 independent sampling. The extension to higher-dimensional signals and the complications that arise  
710 in that context are rigorously treated in [Kuric et al. \(2025\)](#).

### 711 C.2 LATENT DISTRIBUTIONS AND NOTATION

712 Some of the distributions that we rely on in this work have multiple competing parametrizations.  
713 To avoid ambiguities, we provide precise definitions of the four increment distributions that we  
714 consider in this work: Gaussian; Laplace; Student-t; and Bernoulli–Laplace (spike-and-slab). We give  
715 in [Table 2](#) our notations of these and other distributions that we use in this work. We list in [Table 3](#)  
716 the latent maps and conditional latent distributions that are needed for the GLM for the distributions  
717 in this work.

## 718 D GIBBS METHODS AND SAMPLING EFFICIENCY

719 Gibbs methods are Markov chain Monte Carlo (MCMC) methods to sample from a joint distribution  
720  $p_{\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n}$  of  $(n+1)$  blocks of variables that are advantageous when the direct sampling is com-  
721 putationally expensive.

Table 3: Latent variable representations and conditional distributions for common distributions.

Dist. $\phi_k$	Latent dist. $f_k$	Latent maps	Cond. latent dist. $p_{[\mathbf{Z}]_k   X = [\mathbf{Kx}]_k}$
Gauss( $\mu, \sigma^2$ )	$\delta(0)$	$\mu_k(z) = \mu, \sigma_k^2(z) = \sigma^2$	$\delta(0)$
Laplace( $b$ )	$\text{Exp}\left(\frac{1}{2b^2}\right)$	$\mu_k(z) = 0, \sigma_k^2(z) = z$	$\text{GIG}\left(\frac{1}{b^2}, [\mathbf{Kx}]_k^2, \frac{1}{2}\right)$
St( $\nu$ )	$\text{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2}\right)$	$\mu_k(z) = 0, \sigma_k^2(z) = \frac{1}{z}$	$\text{Gamma}\left(\frac{\nu+1}{2}, \frac{\nu+[\mathbf{Kx}]_k^2}{2}\right)$

---

**Algorithm 3** Latent-variable Gibbs sampling of  $p_{\mathbf{X}, \mathbf{Z}_1, \dots, \mathbf{Z}_n}$ .

---

**Require:** Burn-in period  $B \in \mathbb{N}$ , number of samples  $S \in \mathbb{N}$ , initial point  $(\mathbf{x}_0, \mathbf{z}_1, \dots, \mathbf{z}_n)$ .

```

1: for  $k = 1, 2, \dots, B + S$  do
2:    $\mathbf{x}_k \sim p_{\mathbf{X} | \mathbf{Z}_1 = \mathbf{z}_1, \dots, \mathbf{Z}_n = \mathbf{z}_n}$ 
3:    $\mathbf{z}_1 \sim p_{\mathbf{Z}_1 | \mathbf{X} = \mathbf{x}_k, \dots, \mathbf{Z}_n = \mathbf{z}_n}$  ▷ Latent blocks do not need to be stored
4:    $\vdots$ 
5: return  $\{\mathbf{x}_{B+k}\}_{k=1}^S$ 

```

---

721 tationally difficult but sampling from the conditional distributions  $p_{\mathbf{X} | \mathbf{Z}_1, \mathbf{Z}_2, \dots, \mathbf{Z}_n}, p_{\mathbf{Z}_1 | \mathbf{X}, \mathbf{Z}_2, \dots, \mathbf{Z}_n}, \dots$   
722 is easy. Gibbs methods cycle through the conditional distributions with repeated draws, which  
723 maintains the joint distribution invariant (Casella & George, 1992). The naming of the variables  
724  $\mathbf{X}, \mathbf{Z}_1, \mathbf{Z}_2, \dots, \mathbf{Z}_n$  is deliberately chosen to emphasize that we use *latent-variable* Gibbs methods  
725 that rely on auxiliary variables that are introduced solely to make the conditionals simple. The steps  
726 of a general latent variable Gibbs sampler are shown in Algorithm 3, where the iteration counter in  
727 the sampling of the latent variables is omitted since they need not be stored and previous iterations  
728 can immediately be overwritten.

729 Kuric et al. (2025) recently showed that such methods are significantly faster than other standard  
730 sampling routines that are commonly used in settings similar to the one in this paper. They report  
731 sampling efficiencies of close to 1, while alternatives, such as the Metropolis-adjusted Langevin  
732 algorithm, achieve sampling efficiencies<sup>7</sup> of around  $1 \times 10^{-3}$ . In addition, Gibbs methods require no  
733 stepsize or acceptance-rate tuning and introduce no discretization bias. These properties motivate our  
734 use of Gibbs methods for the fast and robust posterior sampling throughout this work.

735 Like all MCMC methods, in practice Gibbs methods benefit from the discarding of some number  
736 of initial samples (the *burn-in period*) when the initial point is located in low-density regions. After  
737 the burn-in period, the quality of the Monte Carlo estimate of any object depends on the number of  
738 samples one uses in their estimation. We discuss our choice of the burn-in period and the number of  
739 samples for the various problems in Appendix F.2.

740 D.1 A GIBBS METHOD FOR BERNOUlli–LAPLACE INCREMENTS

741 Let  $\delta$  be the Dirac distribution. Then, letting  $\lambda$  be the Bernoulli parameter and  $b$  the scale parameter,  
742 we note that the Bernoulli–Laplace density

$$p_U(u) = \lambda\delta(u) + (1 - \lambda)\frac{b}{2}\exp(-b|u|) \quad (44)$$

743 admits the representation

$$p_U(u) = \int_{\mathbb{R}} \left( \sum_{v=0}^1 p_{U|V=v, W=w}(u) p_V(v) \right) p_W(w) dw, \quad (45)$$

744 where

$$p_V(v) = \lambda^{1-v}(1 - \lambda)^v \quad (46)$$

745 for  $v \in \{0, 1\}$  is a Bernoulli distribution,

$$p_W(w) = \frac{b^2}{2} \exp\left(-\frac{b^2 w}{2}\right) \chi_{\mathbb{R}_{\geq 0}}(w) \quad (47)$$

<sup>7</sup>Sampling efficiency refers to effective samples per iteration; an efficiency of  $\rho$  means roughly  $1/\rho$  iterations per “effective sample” (Gelman et al., 2013, Section 11.5).

**Algorithm 4** Bernoulli–Laplace Gibbs sampler.

**Require:** Initial increments  $\mathbf{u}_0 \in \mathbb{R}^d$ , initial support vector  $\mathbf{v} \in \mathbb{R}^d$

```

1: for  $s = 1, 2, \dots, B + S$  do
2:   Draw  $[\mathbf{w}]_k \sim p_{\mathbf{W}|U=[\mathbf{u}_{s-1}]_k, V=[\mathbf{v}]_k}$  ▷ parallel over  $k$ 
3:   for  $k = 1, 2, \dots, d$  do
4:     Draw  $[\mathbf{v}]_k \sim \text{Bernoulli}(p_k(\mathbf{v}))$ 
5:     Draw  $\mathbf{u}_s \sim p_{\mathbf{U}|V=\mathbf{v}, \mathbf{W}=\mathbf{w}, \mathbf{Y}=\mathbf{y}}$ 
6:   return  $\{\mathbf{D}^{-1} \mathbf{u}_{B+k}\}_{k=1}^S$ 

```

746 is an exponential distribution, and

$$p_{U|V=v, W=w}(u) = \begin{cases} \delta(u) & \text{if } v = 0, \\ \text{Gauss}(0, w) & \text{if } v = 1. \end{cases} \quad (48)$$

747 The algorithm relies on the introduction of two latent vectors  $\mathbf{v}, \mathbf{w} \in \mathbb{R}^d$  that satisfy

$$p_{\mathbf{U}|\mathbf{V}=\mathbf{v}, \mathbf{W}=\mathbf{w}}(\mathbf{u}) = \prod_{k=1}^d p_{U|V=[\mathbf{v}]_k, W=[\mathbf{w}]_k}([\mathbf{u}]_k) \quad (49)$$

748 such that, as a result, the distribution conditioned on the measurements can be written as

$$\begin{aligned} p_{\mathbf{U}, \mathbf{V}, \mathbf{W}|\mathbf{Y}=\mathbf{y}}(\mathbf{u}, \mathbf{v}, \mathbf{w}) &\propto \exp\left(-\frac{1}{2\sigma_n^2} \|\mathbf{H}\mathbf{u} - \mathbf{y}\|^2\right) \prod_{k=1}^d p_{U|V=[\mathbf{v}]_k, W=[\mathbf{w}]_k}([\mathbf{u}]_k) \\ &\times \prod_{k=1}^d \lambda^{1-[\mathbf{v}]_k} (1-\lambda)^{[\mathbf{v}]_k} \prod_{k=1}^d \frac{b^2}{2} \exp\left(-\frac{b^2 [\mathbf{w}]_k}{2}\right), \end{aligned} \quad (50)$$

749 where  $\mathbf{H} = \mathbf{A}\mathbf{D}^{-1}$ . Equations (48) and (50) imply that any sample from  $p_{\mathbf{U}|\mathbf{V}=\mathbf{v}, \mathbf{W}=\mathbf{w}, \mathbf{Y}=\mathbf{y}}$  takes  
750 the value zero at those indices where  $\mathbf{v}$  is zero, and values from a multivariate Gaussian distribution  
751 with covariance  $\mathbf{C} = (\sigma_n^2 \mathbf{H}\mathbf{H}^\top + \text{diag}(\mathbf{w}))^{-1}$  and mean  $\sigma_n^{-2} \mathbf{C}\mathbf{H}^\top \mathbf{y}$  otherwise. Sampling  $\mathbf{W} |$   
752  $\mathbf{U} = \mathbf{u}, \mathbf{V} = \mathbf{v}, \mathbf{Y} = \mathbf{y}$  amounts to the independent sampling of  $d$  one-dimensional distributions,  
753 which are  $\text{Exp}(2/b^2)$  at indices where  $\mathbf{v}$  is zero and  $\text{GIG}(b^2, [\mathbf{u}]_k^2, 0.5)$  those indices  $k$  where  $\mathbf{v}$  is  
754 one. The conditional distribution of the binary support vector is

$$p_{\mathbf{V}|\mathbf{W}=\mathbf{w}, \mathbf{Y}=\mathbf{y}}(\mathbf{v}) \propto |\mathbf{B}(\mathbf{v}, \mathbf{w})|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \mathbf{y}^\top \mathbf{B}(\mathbf{v}, \mathbf{w})^{-1} \mathbf{y}\right) \prod_{k=1}^d \lambda^{1-[\mathbf{v}]_k} (1-\lambda)^{[\mathbf{v}]_k}, \quad (51)$$

755 where<sup>8</sup>  $\mathbf{B}(\mathbf{v}, \mathbf{w}) = \sigma_n^2 \mathbf{I} + \mathbf{H} \text{diag}(\mathbf{v} \odot \mathbf{w}) \mathbf{H}^\top$ . The standard way to sample from this distribution is  
756 to use a coordinate-wise Gibbs sampler that updates  $[\mathbf{v}]_k \sim \text{Bernoulli}(p_k(\mathbf{v}))$  with

$$p_k(\mathbf{v}) = (1 + \exp(-\Delta_k(\mathbf{v})))^{-1} \quad (52)$$

757 where the log-odds increment

$$\begin{aligned} \Delta_k(\mathbf{v}) &= \log \frac{1-\lambda}{\lambda} - \frac{1}{2} (\log |\mathbf{B}(\mathbf{v}_{k=1}, \mathbf{w})| - \log |\mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})|) \\ &\quad - \frac{1}{2} (\mathbf{y}^\top \mathbf{B}(\mathbf{v}_{k=1}, \mathbf{w})^{-1} \mathbf{y} - \mathbf{y}^\top \mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})^{-1} \mathbf{y}), \end{aligned} \quad (53)$$

758 where  $\mathbf{v}_{k= \cdot} := (\mathbf{v}_1, \dots, \mathbf{v}_{k-1}, \cdot, \mathbf{v}_{k+1}, \dots, \mathbf{v}_d)$  is the difference between the log-posterior when  
759 the bit is on and when it is off. The resulting algorithm is summarized in [Algorithm 4](#) and can be  
760 interpreted<sup>9</sup> as  $(d+2)$ -block (*i.e.*, dimension-dependent) Gibbs method.

<sup>8</sup>This formulation is equivalent to the one presented by [Bohra et al. \(2023\)](#), who explicitly “slice” the matrices  $\mathbf{H}$  and  $\text{diag}(\mathbf{w})$  with the indices where  $\mathbf{v}$  is one. We stick to this formulation since it requires less notation and emphasizes that implementations need not build variable-sized matrices, which is crucial for an efficient implementation on modern compute units that utilize highly parallelized computations.

<sup>9</sup>This is only an interpretation because the density violates the classical positivity conditions that are needed for Gibbs methods. It is a *partially collapsed* Gibbs method, see ([Bohra et al., 2023](#); [van Dyk & Park, 2008](#)).

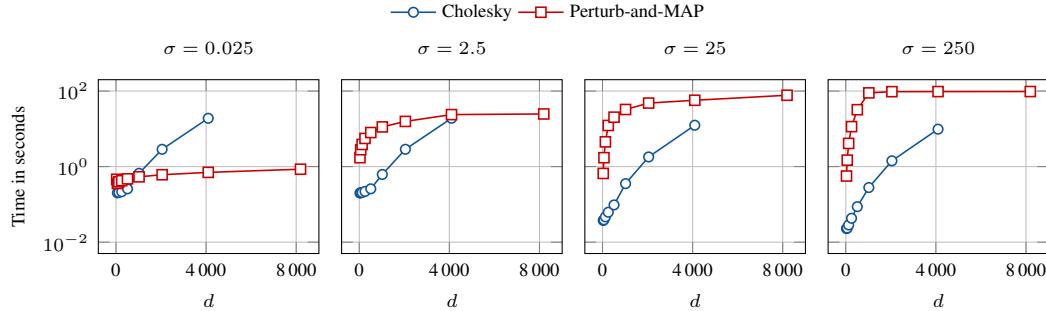


Figure 6: Runtimes needed to perform 20 Gibbs iterations on a denoising posterior (Laplace(1) increment distribution, 10 parallel chains) depending on the dimensionality of the signal. Missing entries are due to excessive memory requirements.

761 D.2 PRACTICAL GIBBS IMPLEMENTATIONS

762 **Sampling the Gaussians** The sampling of  $\mathbf{X} \mid \mathbf{Z}$  in the GLM and of  $\mathbf{U} \mid \mathbf{V}, \mathbf{W}, \mathbf{Y}$  for the  
 763 Bernoulli–Laplace case reduces to drawing from a high-dimensional Gaussian, which is a well-  
 764 studied problem. For settings that necessitate a matrix-free implementation such as those that are  
 765 commonly encountered in imaging applications, Kuric et al. (2025) advocate a Perturb-and-MAP  
 766 sampler with preconditioned conjugate-gradient solvers. We report the runtime of the Gibbs method  
 767 as a function of signal dimension for a Laplace(1) increment distribution in Figure 6. A standard  
 768 implementation based on a Cholesky factorization of the covariance matrix—which requires explicitly  
 769 instantiating the matrices  $\mathbf{A}$  and  $\mathbf{D}$  in memory—is faster than the Perturb-and-MAP sampler with a  
 770 conjugate-gradient solver across a broad range of noise variances and dimensions. For our moderate-  
 771 dimensional setting with  $d = 64$ , the Cholesky-based implementation is approximately an order of  
 772 magnitude faster.

773 However, explicitly storing these matrices becomes infeasible at larger dimensions (in our setup, we  
 774 ran out of memory at  $d = 8096$ ), and the expected cubic scaling is apparent in the figure. In contrast,  
 775 the Perturb-and-MAP sampler (convergence criterion: squared residual norm below  $1 \times 10^{-6}$ ), while  
 776 slower than Cholesky at small dimensions, exhibits substantially better scaling with signal dimension.  
 777 In particular, it does not require materializing the operators: both the measurement operator  $\mathbf{A}$  and  
 778 the finite-difference operator  $\mathbf{D}$  can be implemented efficiently in a matrix-free manner. Moreover,  
 779 the sublinear runtime observed in this experiment suggests that the corresponding linear systems are  
 780 well conditioned.

781 The sampling accuracy of Perturb-and-MAP depends on the termination criterion used by the  
 782 optimization solver, and any finite stopping rule yields approximate samples. A principled refinement  
 783 is to incorporate a Metropolis–Hastings correction step to remove bias, and to tune the solver accuracy  
 784 to optimize overall runtime; this strategy was proposed by Gilavert et al. (2015), to which we refer for  
 785 details. Overall, these results indicate that the Gibbs method scales favorably to higher dimensions.  
 786 Combined with the fact that the denoising posteriors are sub-Gaussian, and with the relatively mild  
 787 sample-complexity requirements for estimator accuracy in this setting, this suggests that the proposed  
 788 framework scales well as the dimension increases.

789 **Sampling the Latent Variables** The sampling of the latent variables necessitates the sampling  
 790 of the one-dimensional conditional latent distributions. All the conditional latent distributions that  
 791 are relevant in this paper admit efficient samplers that are readily available in standard scientific  
 792 computing packages or can be implemented with little effort. We reuse the CUDA implementation  
 793 of the generalized inverse Gaussian sampler from Kuric et al. (2025) that implements the method  
 794 proposed by Devroye (2012) and rely on PyTorch (Paszke et al., 2017) for all others. Wherever  
 795 possible, latent updates are parallelized.

796 In the Gibbs methods for the Bernoulli–Laplace increments, the sequential drawing of the binary  
 797 support vector  $\mathbf{V}$  is embedded in the outer Gibbs loop which, in turn, may be embedded in the  
 798 reverse-diffusion loop. This makes it crucial to minimize the use of heavy linear-algebra operations  
 799 to achieve acceptable runtimes. Writing  $\mathbf{B}(\mathbf{v}, \mathbf{w}) = \sigma_n^2 \mathbf{I} + \mathbf{H} \text{diag}(\mathbf{v} \odot \mathbf{w}) \mathbf{H}^\top$ , we recognize that

800 the flipping of the  $k$ th bit of  $\mathbf{v}$  adds or removes a rank-one term  $[\mathbf{w}]_k \mathbf{H}_k \mathbf{H}_k^\top$ , where  $\mathbf{H}_k$  is the  $k$ th  
 801 column of  $\mathbf{H}$ . Using the matrix-determinant lemma and Woodbury–Sherman–Morrison, we update

$$\log |\mathbf{B}(\mathbf{v}_{k=1}, \mathbf{w})| = \log |\mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})| + \log(1 + [\mathbf{w}]_k \tau_k) \quad (54)$$

802 and

$$\mathbf{y}^\top \mathbf{B}(\mathbf{v}_{k=1}, \mathbf{w})^{-1} \mathbf{y} = \mathbf{y}^\top \mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})^{-1} \mathbf{y} - \frac{[\mathbf{w}]_k (\mathbf{H}_k^\top \mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})^{-1} \mathbf{y})^2}{1 + [\mathbf{w}]_k \tau_k}, \quad (55)$$

803 where  $\tau_k = \mathbf{H}_k^\top \mathbf{B}(\mathbf{v}_{k=0}, \mathbf{w})^{-1} \mathbf{H}_k$ . Thus, an efficient implementation factors  $\mathbf{B}(\mathbf{v}, \mathbf{w})$  once per  
 804 latent state, obtains the needed scalars via triangular solves, and performs rank-one updates as bits  
 805 flip. We report our cumulative runtime improvement over a naive implementation in [Figure 4](#).

## 806 E DPS UPDATE STEPS

### 807 E.1 COVARIANCE IN C-DPS

808 C-DPS ([Chung et al., 2023](#)) uses the approximation of the likelihood

$$p_{\mathbf{Y}|\mathbf{X}_t=\mathbf{x}}(\mathbf{y}) \approx p_{\mathbf{Y}|\mathbf{X}_0=\mathbb{E}[\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}]}(\mathbf{y}). \quad (56)$$

809 When the noise in the inverse problem is Gaussian, the likelihood score  
 810  $\nabla(\mathbf{x} \mapsto \log p_{\mathbf{Y}|\mathbf{X}_0=\mathbb{E}[\mathbf{X}_0|\mathbf{X}_t=\mathbf{x}]}(\mathbf{y}))$  necessitates the computation of

$$\nabla(\mathbf{x} \mapsto \frac{1}{2} \|\mathbf{A} \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}] - \mathbf{y}\|^2), \quad (57)$$

811 which is

$$\mathbf{J}(\mathbf{x} \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]) (\cdot) \mathbf{A}^\top (\mathbf{A} \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \cdot] - \mathbf{y}) \quad (58)$$

812 after an application of the chain rule. The Jacobian  $\mathbf{J}(\mathbf{x} \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}])$  is typically computed  
 813 with automatic differentiation when  $(\mathbf{x}, t) \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]$  is approximated with a neural network.  
 814 In our framework, we use the connection with the covariance matrix  $\text{Cov}[\mathbf{X}_0 | \mathbf{X}_t = \cdot]$ . Indeed, as  
 815 also shown by [Rissanen et al. \(2025\)](#), if  $\mathbf{X}_0$  and  $\mathbf{X}_t$  verify (32), then

$$\frac{1}{1-\bar{\alpha}_t} \text{Cov}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}] = \frac{1}{\bar{\alpha}_t} (\mathbf{I} + (1 - \bar{\alpha}_t)^2 \nabla^2 \log p_{\mathbf{X}_t}(\mathbf{x})). \quad (59)$$

816 This identity, combined with the derivative of (6), yields

$$\mathbf{J}(\mathbf{x} \mapsto \mathbb{E}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}]) (\mathbf{x}_t) = \frac{\sqrt{\bar{\alpha}_t}}{1 - \bar{\alpha}_t} \text{Cov}[\mathbf{X}_0 | \mathbf{X}_t = \mathbf{x}_t]. \quad (60)$$

### 817 E.2 EXPLICIT UPDATE STEPS

818 We give the instantiations of the update step  $\mathcal{S}$  a variety of DPS algorithms below. Each  $\mathbf{z}_t$  is a  
 819  $d$ -dimensional random vector with i.i.d. standard Gaussian entries.

820 **Score-ALD** ([Jalal et al., 2021](#)) The input parameters of this algorithm are composed of the following:  
 821 A noise schedule  $\{\beta_t\}_{t=0}^{T-1}$ , the noise level of the inverse problem  $\sigma_n$ , and annealing parameters  
 822  $\{\eta_t\}_{t=0}^{T-1}$  and  $\{\gamma_t\}_{t=0}^{T-1}$ . The update step goes

$$\begin{aligned} \bar{\mu} &= \frac{1}{S} \sum_{s=1}^S \bar{\mathbf{x}}_s, \\ \mathbf{s}_t &= (\bar{\mu} - \mathbf{x}_t) / \beta_t^2, \\ \mathbf{x}_{t-1} &= \mathbf{x}_t + \eta_t (\mathbf{s}_t + \frac{1}{\gamma_t^2 + \sigma_n^2} \mathbf{A}^\top (\mathbf{y} - \mathbf{A} \mathbf{x}_t)) + \sqrt{2\eta_t} \mathbf{z}_t. \end{aligned} \quad (61)$$

823 **C-DPS** ([Chung et al., 2023](#)) The input parameters are the the variance-preserving scaling weight  
 824  $\bar{\alpha}_t$  as in (32), the variance of the diffusion transitions  $\beta_t$  as in (35), and a scalar  $\zeta$  that governs the

825 likelihood-guidance strength. The diffusion noise level that corresponds to the denoising posterior is  
 826 denoted  $\sigma_t = (1 - \bar{\alpha}_t)/\sqrt{\bar{\alpha}_t}$ , which is used to compute the samples  $\{\bar{\mathbf{x}}_k\}_{k=1}^S$ . The update step goes

$$\begin{aligned}\bar{\mu} &= \frac{1}{S} \sum_{k=1}^S \bar{\mathbf{x}}_k, \\ \mathbf{C} &= \frac{1}{S} \sum_{k=1}^S (\bar{\mathbf{x}}_k - \bar{\mu})(\bar{\mathbf{x}}_k - \bar{\mu})^\top, \\ \mathbf{x}'_{t-1} &= \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \bar{\mu} + \sigma_t \mathbf{z}_t, \\ \tilde{\mathbf{x}}_{t-1} &= \mathbf{x}'_{t-1} - \frac{\zeta}{\|\mathbf{A}\bar{\mu} - \mathbf{y}\|} \frac{\sqrt{\bar{\alpha}_t}}{1 - \bar{\alpha}_t} \mathbf{C}^\top \mathbf{A}^\top (\mathbf{A}\bar{\mu} - \mathbf{y}), \\ \mathbf{x}_{t-1} &= \tilde{\mathbf{x}}_{t-1} / \sqrt{\bar{\alpha}_{t-1}}.\end{aligned}\tag{62}$$

827 **DiffPIR (Zhu et al., 2023)** The input parameters are similar to those of C-DPS. It also uses the  
 828 noise level of the inverse problem  $\sigma_n$  and an additional balance hyperparameter  $\gamma$ . The update step  
 829 goes

$$\begin{aligned}\bar{\mu} &= \frac{1}{S} \sum_{k=1}^S \bar{\mathbf{x}}_k, \\ \rho_t &= \zeta \frac{\sigma_n^2}{\sigma_t^2}, \\ \bar{\mathbf{x}}_0 &= \arg \min_{\mathbf{x} \in \mathbb{R}^d} \left( \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 + \frac{\rho_t}{2} \|\mathbf{x} - \bar{\mu}\|^2 \right), \\ \hat{\epsilon} &= \frac{1}{\sqrt{1 - \bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \bar{\mathbf{x}}_0), \\ \tilde{\mathbf{x}}_{t-1} &= \sqrt{\bar{\alpha}_{t-1}} \bar{\mathbf{x}}_0 + \sqrt{1 - \bar{\alpha}_{t-1}} (\sqrt{1 - \gamma} \hat{\epsilon} + \sqrt{\gamma} \mathbf{z}_t), \\ \mathbf{x}_{t-1} &= \tilde{\mathbf{x}}_{t-1} / \sqrt{\bar{\alpha}_{t-1}}.\end{aligned}\tag{63}$$

830 **PIGDM (Song et al., 2023)** The input parameters are the noise schedule  $\{\sigma_t\}_{t=0}^{T-1}$ , the data-  
 831 dependent noise schedule  $\{r_t\}_{t=0}^{T-1}$ , and the DDIM (Song et al., 2020) time-dependent coefficients  
 832  $\{c_t^{(1)}\}_{t=0}^{T-1}$  and  $\{c_t^{(2)}\}_{t=0}^{T-1}$ . The update step goes

$$\begin{aligned}\bar{\mu} &= \frac{1}{S} \sum_{k=1}^S \bar{\mathbf{x}}_k, \\ \mathbf{C} &= \frac{1}{S} \sum_{s=1}^S (\bar{\mathbf{x}}_s - \bar{\mu})(\bar{\mathbf{x}}_s - \bar{\mu})^\top, \\ \mathbf{J} &= \frac{\sqrt{\bar{\alpha}_t}}{1 - \bar{\alpha}_t} \mathbf{C}, \\ \mathbf{g} &= \left( (\mathbf{y} - \mathbf{A}\bar{\mu})^\top \left( \mathbf{A}\mathbf{A}^\top + \frac{\sigma_n^2}{r_t^2} \mathbf{I} \right)^{-1} \mathbf{A}\mathbf{J} \right)^\top, \\ \hat{\epsilon} &= \frac{1}{\sqrt{1 - \bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \bar{\mu}), \\ \tilde{\mathbf{x}}_{t-1} &= \sqrt{\alpha_s} \bar{\mu} + c_1 \mathbf{z}_t + c_2 \hat{\epsilon} + \sqrt{\alpha_t} \mathbf{g}, \\ \mathbf{x}_{t-1} &= \tilde{\mathbf{x}}_{t-1} / \sqrt{\bar{\alpha}_{t-1}}.\end{aligned}\tag{64}$$

833 **DPnP (Xu & Chi, 2024)** The diffusion noise level that corresponds to the denoising posterior is  
 834 denoted  $\eta_t$ , which is used to compute the sample  $\bar{\mathbf{x}}_1$ . This same  $\eta_t$  defines the likelihood-guidance  
 835 strength. The update step goes

$$\begin{aligned}\mathbf{x}_0 &= \bar{\mathbf{x}}_1, \\ \mathbf{x}_{t-1} &\sim \exp \left( -\frac{1}{2} \|\mathbf{A} \cdot -\mathbf{y}\|^2 - \frac{1}{2\eta_t^2} \|\cdot - \mathbf{x}_0\|^2 \right).\end{aligned}\tag{65}$$

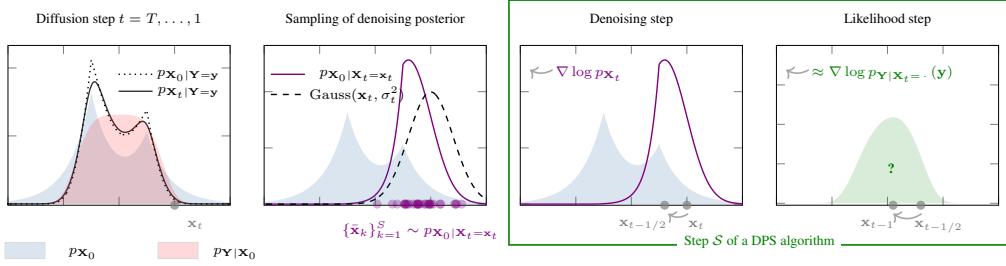


Figure 7: Illustration of the proposed template for DPS algorithms.

836 **Annealed Plug-and-Play Monte Carlo (PnP and RED variants) (Sun et al., 2024)** The diffusion  
 837 noise level that corresponds to the denoising posterior is denoted  $\sigma_t$ , which are used to compute the  
 838 samples  $\{\bar{x}_k\}_{k=1}^S$ . The parameter  $\eta$  denotes the likelihood guidance strength, and  $\kappa_t$  is an annealing  
 839 parameter.

840 The update step for the PnP variant goes

$$\begin{aligned} \bar{\mu} &= \frac{1}{S} \sum_{k=1}^S \bar{x}_k, \\ \mathbf{s} &= (\bar{\mu} - \mathbf{x}_t) / \sigma_t^2, \\ \mathbf{x}_{t-1} &= \mathbf{x}_t + \gamma \kappa_t \mathbf{s} + \sqrt{2\gamma} \mathbf{z}_t, \\ \mathbf{x}_t &\leftarrow \mathbf{x}_t - \gamma \mathbf{A}^\top (\mathbf{A}\mathbf{x} - \mathbf{y}). \end{aligned} \quad (66)$$

841 The update step for the RED variant goes

$$\begin{aligned} \bar{\mu} &= \frac{1}{S} \sum_{k=1}^S \bar{x}_k, \\ \mathbf{s} &= (\bar{\mu} - \mathbf{x}_t) / \sigma_t^2, \\ \mathbf{x}_{t-1} &= \mathbf{x}_t - \gamma (\mathbf{A}^\top (\mathbf{A}\mathbf{x} - \mathbf{y}) - \kappa_t \mathbf{s}) + \sqrt{2\gamma} \mathbf{z}_t. \end{aligned} \quad (67)$$

842 The DPS template that is summarized in [Algorithm 2](#) is illustrated with a one-dimensional toy-example  
 843 in [Figure 7](#).

## 844 F NUMERICAL EXPERIMENTS

### 845 F.1 FORWARD OPERATORS

846 We consider four forward operators  $\mathbf{A}$  in our experiments. The first operator is the identity  $\mathbf{A} = \mathbf{I} \in$   
 847  $\mathbb{R}^{d \times d}$ . This choice is motivated by the fundamental role that denoising algorithms currently play in  
 848 many restoration algorithms and even in labeling problems such as edge detection (Le et al., 2025).  
 849 The second operator  $\mathbf{A} \in \mathbb{R}^{d \times d}$  implements the convolution with a kernel that consists of the 13  
 850 central samples of a truncated Gaussian with variance 2 that are normalized to unit sum. We adopt  
 851 circular boundary conditions to enable a fast computation of the proximal map that arises in the update  
 852 step of DiffPIR (see [Appendix E.2](#)) via the fast Fourier transform. Deconvolution is a relevant problem  
 853 with applications like microscopy or astronomy. The third operator is a sampling operator  $\mathbf{A} \in \mathbb{R}^{m \times d}$   
 854 that returns  $m < d$  entries of its argument unchanged. This operator is relevant in many fields such  
 855 as image reconstruction and time-series forecasting. In particular, in a forecasting or prediction  
 856 problem the operator would return the first  $m$  known entries, and the resolution of the inverse problem  
 857 estimates the remaining  $(d - m)$  entries. In our experiments, each entry has an independent chance  
 858 of 40 % of being kept. The fourth and last operator is  $\mathbf{A} = \mathbf{MF} \in \mathbb{R}^{m \times d}$ , where  $\mathbf{F} \in \mathbb{R}^{2(\lfloor d/2 \rfloor + 1) \times d}$   
 859 is the matrix representation of the “real” one-dimensional discrete Fourier transform with separated  
 860 real and imaginary components, and  $\mathbf{M} \in \mathbb{R}^{m \times 2(\lfloor d/2 \rfloor + 1)}$  is a sampling operator. Such operators are  
 861 relevant in medical imaging or astronomy. The sampling operator is constructed such that the 5 lowest

862 frequencies (the DC term included) are acquired, while the remaining frequencies independently have  
 863 a 40 % chance of being kept.

864 For all operators, the noise variance  $\sigma_n^2$  is chosen such that the median measurement signal-to-noise  
 865 ratio (SNR) is around 25 dB. We set  $N_{\text{train}} = 1 \times 10^6$ ,  $N_{\text{val}} = 1 \times 10^3$ , and  $N_{\text{test}} = 1 \times 10^3$ .

## 866 F.2 BENCHMARK IMPLEMENTATION

867 The benchmarking pipeline starts with the generation of  $N_{\text{test}}$  test signals denoted  $\{\mathbf{x}_k^{\text{test}}\}_{k=1}^{N_{\text{test}}}$  per  
 868 increment distribution, each of which is independently synthesized by first drawing i.i.d. increments  
 869 from the respective increment distribution and forming the signals via (42). It then proceeds to  
 870 synthesize the  $N_{\text{test}}$  measurements (*i.e.* we use one noise instance per signal) denoted  $\{\mathbf{y}_k^{\text{test}}\}_{k=1}^{N_{\text{test}}}$   
 871 according to (1) and, for each of the measurements, computes the gold-standard posterior samples  
 872 of the various inverse problems via the Gibbs methods described in Section 3. This stage is off-line  
 873 (no reverse-diffusion loop) and trivially parallel across the measurements, which allows us to run  
 874 long chains with burn-in periods of  $1 \times 10^5$  iterations and obtain  $2 \times 10^5$  draws from the posterior  
 875 distribution. This far exceeds any values reported by Kuric et al. (2025) or Bohra et al. (2023) and  
 876 results in precise MMSE estimates.

877 The dataset-generation stage also involves the generation of  $N_{\text{train}}$  training signals  $\{\mathbf{x}_k^{\text{train}}\}_{k=1}^{N_{\text{train}}}$   
 878 and  $N_{\text{val}}$  validation signals (mutually disjoint from the test signals)  $\{\mathbf{x}_k^{\text{val}}\}_{k=1}^{N_{\text{val}}}$ , along with the  
 879 corresponding validation measurements  $\{\mathbf{y}_k^{\text{val}}\}_{k=1}^{N_{\text{val}}}$ . The training signals are used for the learning  
 880 of a neural score function like those that are used for the resolution of inverse problems when the  
 881 prior is unknown or too expensive to evaluate. Training details are provided in Appendix F.3. The  
 882 validation signals are used to monitor the performance of the neural score function on unseen signals  
 883 during the training stage and to tune the regularization parameters for the model-based approaches as  
 884 well as the parameters of the DPS algorithms, see Section 4.1.

885 Unlike for the computation of the gold-standard MMSE estimate of the initial inverse problem, the  
 886 denoising posteriors are sampled  $T$  times per trajectory (we use  $T = 1000$ ). To ensure acceptable  
 887 runtimes in this setting, we therefore pick the smallest burn-in period and sample count that still yield  
 888 accurate estimates of the required statistics. We determine these settings with a rigorous protocol that  
 889 is detailed in Appendix F.4. Ultimately, this protocol resulted in the choice of a burn-in period of 100  
 890 iterations and a sample count of 300.

## 891 F.3 LEARNING DETAILS

892 For learned-based denoisers, a noise-conditional neural network with UNet architecture (305 761  
 893 learnable parameters) is trained in an off-line step on the  $N_{\text{train}}$  training signals in a standard setup  
 894 (Adam optimizer with learning rate  $1 \times 10^{-4}$  with exponential decay with factor 0.9999, 100 000  
 895 parameter updates, batch size 10 000). The noise schedule in C-DPS and DiffPIR is defined by the  
 896 two endpoints  $\beta_0 = 1 \times 10^{-4}$  and  $\beta_T = 2 \times 10^{-2}$  with linear equidistant samples in-between. The  
 897 learned variant of DPnP is the “DDS-DDPM” variant (Xu & Chi, 2024, Algorithms 1 and 3) that  
 898 contains an inner denoising-sampling loop. The arbitrary-precision variant does not require an inner  
 899 loop at all (except for the burn-in period), which makes the arbitrary-precision variant the faster one  
 900 for this case.

## 901 F.4 BURN-IN PERIOD AND DENOISER QUALITY

902 As discussed in Appendix F.2, the burn-in period and the number of samples of the Gibbs samplers  
 903 needs to be chosen appropriately to ensure an acceptable runtimes and a sufficiently small Monte  
 904 Carlo error. We determine the burn-in period and the number of samples through the following  
 905 protocol that is run in an off-line stage prior to running the benchmark. We synthesize  $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \mathbf{n}$   
 906 where  $\sigma_t$  is in the range defined by the noise schedule  $\beta$ ,  $\mathbf{x}_0$  is constructed via (42) for all four  
 907 considered increment distributions, and  $\mathbf{n}$  is some unknown but fixed vector of standard Gaussian  
 908 noise. For each of the synthesized signals, we then launch  $C = 1000$  parallel Gibbs chains on the  
 909 corresponding denoising posterior and run those chains for  $N_{\text{sufficient}}$  iterations, where  $N_{\text{sufficient}}$   
 910 is a sufficiently large natural number that guarantees that the chains are stationary for at least  $N_{\text{avg}}$   
 911 (which is also relatively large) iterations and that, consequently, we can compute precise estimates of

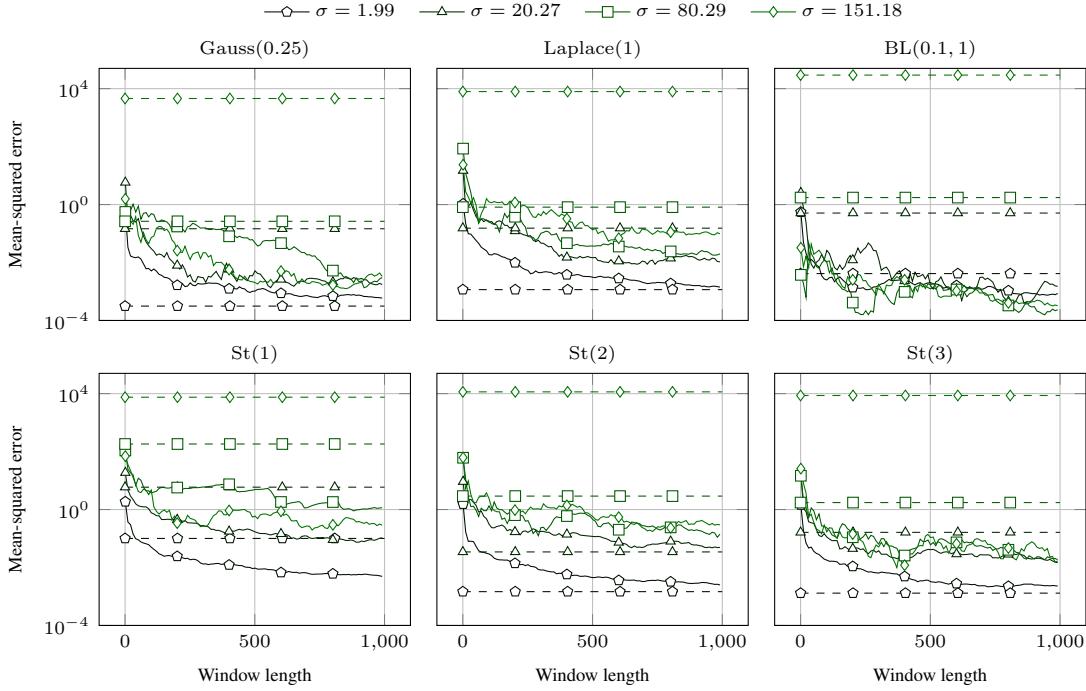


Figure 9: Mean squared error between MMSE estimates and the reference MMSE. Dashed lines: Learned neural MMSE estimate. Solid lines: Monte Carlo MMSE estimate in terms of the window length.

912 various statistics of the posterior distribution from the iterates from the last  $N_{\text{avg}}$  iterations across all  
913  $C$  chains.

914 To determine the burn-in period, we then proceed to calculate a statistic that we can monitor  
915 throughout the iterations and that we can compare against the reference statistic. Specifically,  
916 denoting with  $\mathbf{X}$  the random variable of the Gibbs sampler, we compute the empirical  
917 distribution of the increments at index 32 like  $(\mathbf{X}_{33} - \mathbf{X}_{32})$ . The distribution of differences  
918 that is obtained by taking the last  $N_{\text{avg}}$  iterations across all  $C$  chains is considered the reference  
919 distribution. Then, we compute the Wasserstein-1 distance of that distribution to the one obtained  
920 by taking the average across  $N_{\text{avg}}$  iterations and all  $C$  in a sliding-window starting from the first  
921 Gibbs iterations. This allows us to gauge the burn-in period through a visual inspection of the  
922 Wasserstein-1 distance through the Gibbs iterations. In particular, we expect the Wasserstein-1 dis-  
923 tance to be large for a number of initial samples where the Gibbs sampler is not stationary and then to  
924 oscillate around a small but nonzero value. The value will be nonzero due to the finite sample size.  
925 The Wasserstein-1 distance between the reference distribution and the one obtained through the Gibbs  
926 iterations is shown in Figure 8 (for the exemplary case of a  $\text{St}(1)$  distribution and a selection of noise  
927 variances). We observe that the empirical distribution of increments converges rapidly to the reference  
928 one. The Wasserstein-1 distance reaches the noise level after a single-digit number of iterations,  
929 which is in line with the analysis provided by Kuric et al. (2025). Based on these findings, we chose  
930 the burn-in period as  $B = 100$  iterations for all our experiments, which is more than sufficient to  
931 reach stationarity and has acceptable runtime.

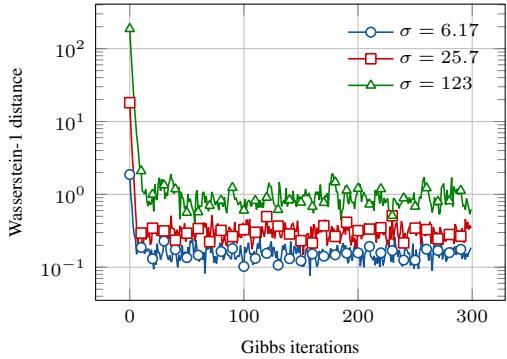


Figure 8: Wasserstein-1 distance of intermediate marginal distributions to that of the final sample.

940 To determine the number of samples that are needed for a sufficiently accurate computation of various  
 941 statistics that any DPS algorithm may utilize in their update steps, we compute a precise estimation  
 942 of the MMSE estimate of the denoising posterior by averaging the last  $N_{\text{avg}}$  iterations across all  $C$   
 943 chains. Then, we pick one arbitrary chain and grow a window from iteration  $(N_{\text{avg}} - 1)$  to the left,  
 944 average the samples in that window, and compute the MSE from the MMSE estimates obtained in  
 945 the one-chain window to the precise estimate obtained by averaging the  $C$  chains and the last  $N_{\text{avg}}$   
 946 iterations. We show this error in terms of the window length and the noise variance for all increment  
 947 distributions in [Figure 9](#). The quality of the learned denoiser and the Monte Carlo denoiser differ over  
 948 the noise variances and the learned denoiser improves relative to the Monte Carlo denoiser as the  
 949 noise variance vanishes. Our final choice of  $S = 300$  samples is motivated by the fact that the quality  
 950 of the Monte Carlo denoiser, *when averaged across all noise variances that appear in the reverse*  
 951 *diffusion*, is always strictly better than the learned denoiser. Since it is relevant for the discussion in  
 952 [Section 4.2](#), we highlight that for this choice the quality of the Monte Carlo denoiser is superior to  
 953 the learned one *across all noise variances* for the  $\text{St}(1)$  and  $\text{BL}(0.1, 1)$  increment distributions.

954 **F.5 ALGORITHM PARAMETERS**

955 The adjustable regularization parameter for  $\text{est} \in \{\ell_2, \ell_1\}$  satisfies

$$\lambda^{\text{est},*} = \arg \min_{\lambda \in \Lambda} \frac{1}{N_{\text{val}}} \sum_{k=1}^{N_{\text{val}}} \frac{1}{d} \|\hat{\mathbf{x}}^{\text{est}}(\mathbf{y}_k^{\text{val}}, \lambda) - \mathbf{x}_k^{\text{val}}\|^2. \quad (68)$$

956 There,  $\Lambda$  is the loglinear grid  $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_{N_{\text{mb}}}\}$  with

$$\lambda_n = 10^{a + (n-1) \frac{(b-a)}{N_{\text{mb}} - 1}} \quad (69)$$

957 with  $a = (-5)$  and  $b = 5$ . Since the model-based methods are very fast, we can use the relatively  
 958 high  $N_{\text{mb}} = 1000$ .

959 The adjustable hyperparameters of the DPS methods were found by

$$\boldsymbol{\lambda}^{\text{alg},*} = \arg \min_{\boldsymbol{\lambda} \in \Theta^{\text{alg}}} \frac{1}{N_{\text{val}}} \sum_{k=1}^{N_{\text{val}}} \frac{1}{d} \|\hat{\mathbf{x}}^{\text{alg}}_{\text{MMSE}}(\mathbf{y}_k^{\text{val}}, \boldsymbol{\lambda}) - \mathbf{x}_k^{\text{val}}\|^2 \quad (70)$$

960 where the grid  $\Theta^{\text{alg}}$  is method-dependent. This tuning is tailored to the evaluation with respect to  
 961 the MMSE optimality gap. Due to resource constraints, the parameters are tuned for the learned  
 962 denoiser. We use  $N_{\text{samples}} = 10$  for the grid search on the validation set. We define a modest number  
 963 of  $N_{\text{dps}} = 40$  grid-points and found the extreme points of the grid (*i.e.*, the values of the parameters  
 964 that clearly lead to worse results) by hand. For C-DPS and DiffPIR, we fix the diffusion schedule to  
 965 standard choices ( $\beta_0 = 1 \times 10^{-4}$ ,  $\beta_T = 0.02$ ). In addition to the diffusion schedule, C-DPS has one  
 966 tunable parameter  $\gamma$  that we tune on 40 loglinear grid points ( $n = 1, \dots, N_{\text{dps}}$ )

$$10^{a + (n-1) \frac{(b-a)}{N_{\text{dps}} - 1}}. \quad (71)$$

967 There,  $a = (-3)$  and  $b = 1$ . DiffPIR has two tunable parameters  $\gamma$  and  $\zeta$ , with  $\gamma$  being typically  
 968 considered uncritical. Thus, we split the 40 grid points into a two-dimensional grid  $\Theta^{\text{DiffPIR}} =$   
 969  $\{0.3, 0.7\} \times \Theta^{\zeta}$ , with 2 points for  $\gamma$  and 20 points for  $\zeta$  given by  $\Theta^{\zeta} = \{\Theta_1^{\zeta}, \dots, \Theta_{N_{\text{dps}}/2}^{\zeta}\}$ , where

$$\Theta_n^{\zeta} = 10^{a + (n-1) \frac{(b-a)}{(N_{\text{dps}}/2) - 1}} \quad (72)$$

970 with  $a = (-4)$  and  $b = 1$ . The DPnP algorithm only has the schedule  $\{\eta_t\}_{t=1}^T$  to tune. In this case,  
 971 since DPnP is asymptotically correct, the schedule is a practical vehicle that enables to trade off  
 972 between speed and accuracy. Therefore, the schedule of this paper is similar to the one that was  
 973 proposed by ([Xu & Chi, 2024](#)): We fix a small  $\eta_{\text{final}} = 0.15$  and linearly decrease  $\eta$  from some  
 974  $\eta_{\text{initial}}$  to  $\eta_{\text{final}}$  after  $K/5$  initial iterations with  $\eta_{\text{initial}}$ , like

$$\eta_n = \begin{cases} \eta_{\text{initial}} & \text{if } n = 1, \dots, K/5 \\ \frac{\eta_{\text{final}} - \eta_{\text{initial}}}{K - K/5} \eta_{\text{initial}} & \text{if } n = K/5 + 1, \dots, K \end{cases} \quad (73)$$

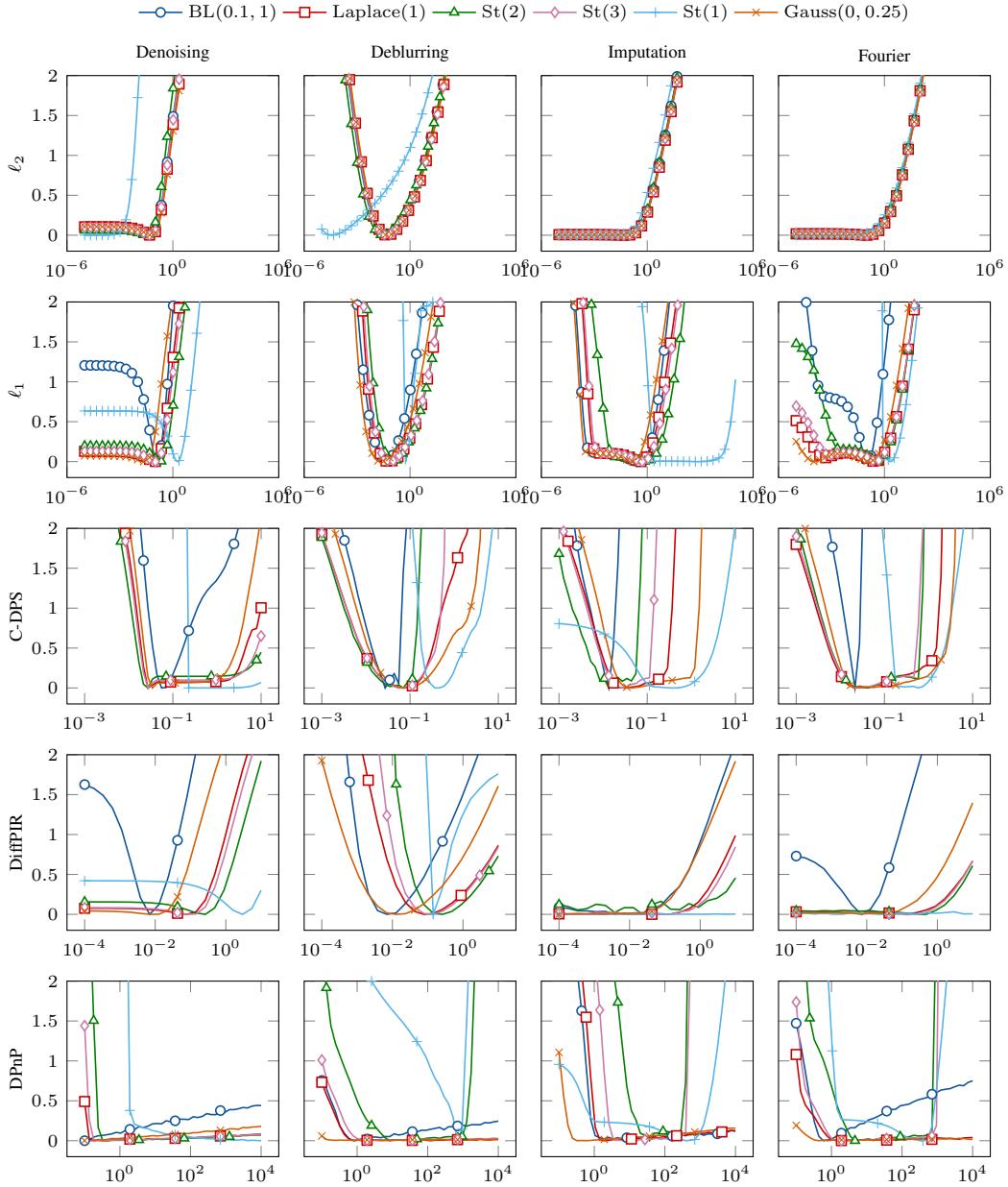


Figure 10: Grid-search diagnostics (logarithm of the MSE over the validation data set) for the model-based methods and the DPS algorithms. Rows:  $\ell_2$ ;  $\ell_1$ ; C-DPS; DiffPIR; DPnP. Columns: Denoising; deconvolution; imputation; reconstruction from partial Fourier measurements. For better visualization, each curve has had its minimum subtracted. To limit clutter, marks are spaced ten apart.

975 We treat  $\eta_{\text{initial}}$  as a tunable parameter and search over  $\Theta^{\text{DPnP}} = \{\eta_1, \eta_2, \dots, \eta_{40}\}$ , where

$$\eta_n = 10^{a + (n-1) \frac{(b-a)}{40-1}}. \quad (74)$$

976 There,  $a = (-1)$  and  $b = 4$ . Like in the original publication, we use the comparatively small  $K = 40$ .

977 The MSE over the validation data depending on the value of the adjustable regularization parameter  
978 of the  $\ell_2$  and  $\ell_1$  estimators and the adjustable hyperparameters of C-DPS, DiffPIR, and DPnP is  
979 shown in Figure 10. Since the  $\gamma$  parameter of DiffPIR is assumed to be uncritical, we only show the  
980 values of the MSE for various choices of  $\zeta$ , where  $\gamma$  is set to the value of the optimal  $(\gamma, \zeta)$  pair.

Table 4: Change in MMSE optimality gap (mean  $\pm$  standard deviation) after substituting the learned denoiser with the arbitrary-precision denoiser. An asterisk indicates a significant changes according to a Wilcoxon signed-rank test ( $p = 0.05$ ). Negative number with asterisk: MMSE estimates obtained with the arbitrary-precision denoiser are significantly better. Positive number with asterisk: MMSE estimates obtained with the learned denoiser are significantly better.

		Gauss(0.25)	Laplace(1)	BL(0.1, 1)	St(1)	St(2)	St(3)
Denoising	C-DPS	0.00 $\pm$ 0.11	0.00 $\pm$ 0.16	-0.46 $\pm$ 1.16*	0.00 $\pm$ 0.01	0.02 $\pm$ 0.79*	-0.01 $\pm$ 0.14
	DiffPIR	0.00 $\pm$ 0.13	0.00 $\pm$ 0.17	-0.05 $\pm$ 0.78*	-0.41 $\pm$ 0.80*	0.00 $\pm$ 0.20	0.00 $\pm$ 0.15
	DPnP	0.04 $\pm$ 0.27*	-0.01 $\pm$ 0.22	-0.55 $\pm$ 1.31*	-0.77 $\pm$ 1.31*	0.00 $\pm$ 0.24	0.00 $\pm$ 0.23
Deconvolution	C-DPS	-0.01 $\pm$ 0.24	0.00 $\pm$ 0.26	0.09 $\pm$ 0.97*	6.64 $\pm$ 3.21*	-0.12 $\pm$ 1.11*	-0.03 $\pm$ 0.43
	DiffPIR	-0.01 $\pm$ 0.23	0.00 $\pm$ 0.23	0.04 $\pm$ 1.12	13.56 $\pm$ 9.90*	-0.01 $\pm$ 0.47	0.00 $\pm$ 0.31
	DPnP	0.00 $\pm$ 0.25	-0.01 $\pm$ 0.27*	-0.02 $\pm$ 1.20	-4.98 $\pm$ 3.86*	0.06 $\pm$ 0.77	-0.02 $\pm$ 0.34
Imputation	C-DPS	0.00 $\pm$ 0.30	0.01 $\pm$ 0.35	0.41 $\pm$ 1.51*	3.41 $\pm$ 4.99*	-0.12 $\pm$ 1.01*	-0.01 $\pm$ 0.57
	DiffPIR	0.00 $\pm$ 0.29	0.00 $\pm$ 0.33	0.03 $\pm$ 1.05	-0.20 $\pm$ 3.05*	0.03 $\pm$ 0.71	0.00 $\pm$ 0.47
	DPnP	0.00 $\pm$ 0.35	-0.02 $\pm$ 0.38	-0.02 $\pm$ 1.02	-10.46 $\pm$ 5.70*	0.02 $\pm$ 0.67	-0.01 $\pm$ 0.48
Fourier	C-DPS	-0.02 $\pm$ 0.43	-0.01 $\pm$ 0.49	0.80 $\pm$ 1.43*	0.09 $\pm$ 5.63*	-0.03 $\pm$ 0.79*	0.01 $\pm$ 0.49
	DiffPIR	-0.01 $\pm$ 0.39	0.00 $\pm$ 0.40	0.12 $\pm$ 0.83*	-0.64 $\pm$ 1.70*	-0.03 $\pm$ 0.42*	-0.02 $\pm$ 0.38
	DPnP	-0.01 $\pm$ 0.43	0.00 $\pm$ 0.45	-0.33 $\pm$ 1.13*	-1.32 $\pm$ 3.18*	0.00 $\pm$ 0.54	0.01 $\pm$ 0.46

Table 5: Runtime of the benchmark with learned objects.

		Gauss(0.25)	Laplace(1)	BL(0.1, 1)	St(1)	St(2)	St(3)
Denoising	C-DPS	00:04:52	00:04:52	00:02:56	00:04:52	00:04:52	00:04:52
	DiffPIR	00:01:59	00:01:58	00:01:12	00:01:58	00:01:59	00:01:59
	DPnP	00:02:33	00:04:58	00:01:15	00:59:33	00:06:13	00:04:58
Deconvolution	C-DPS	00:04:52	00:04:53	00:02:57	00:04:53	00:04:53	00:04:52
	DiffPIR	00:01:59	00:01:59	00:01:12	00:01:59	00:01:59	00:01:59
	DPnP	00:13:54	00:46:39	00:05:48	00:53:30	00:28:24	00:28:24
Imputation	C-DPS	00:04:53	00:04:53	00:02:59	00:04:53	00:04:53	00:04:53
	DiffPIR	00:01:59	00:01:59	00:01:13	00:01:59	00:01:59	00:01:59
	DPnP	00:04:58	00:16:18	00:18:56	00:51:41	00:39:04	00:32:50
Fourier	C-DPS	00:04:54	00:04:54	00:02:59	00:04:55	00:04:55	00:04:54
	DiffPIR	00:01:59	00:01:59	00:01:13	00:01:59	00:01:59	00:01:59
	DPnP	00:06:13	00:13:53	00:04:42	00:51:41	00:23:39	00:16:18

## 981 G ADDITIONAL RESULTS

982 We provide in [Table 4](#) an exhaustive quantitative evaluation of the change in the optimality gap  
983 after we substitute the arbitrary-precision Monte Carlo denoiser for the learned denoiser. We also  
984 report for which cases the arbitrary-precision denoiser enjoys significantly better results than the  
985 learned denoiser according to a Wilcoxon signed-rank test ( $p = 0.05$ ,  $N_{\text{test}}$  pairs, two-sided test  
986 with the winner determined by the median of differences). We attribute a better performance of  
987 the learned denoiser to the fact that the algorithms are fine-tuned using the learned component or  
988 to the cases where the likelihood score approximation is compensated by the one of the learned  
989 component. Note that this table must be interpreted with the quality of the denoisers in mind. As we  
990 show in [Figure 9](#), for our particular choice of  $S = 300$  samples, the Monte Carlo denoiser is strictly  
991 better than the learned denoiser over all noise variances only for signals with  $\text{BL}(0.1, 1)$  and  $\text{St}(1)$   
992 increment distributions.

993 We show uncurated qualitative results of the MMSE estimate obtained by the DPS algorithms and the  
994 gold-standard Gibbs methods in [Figures 11 to 18](#). The figures alternate between the arbitrary-precision  
995 denoiser and the learned denoiser and show the results for deconvolution, denoising, imputation, and  
996 reconstruction from partial Fourier samples, in that order. Each figure contains results for  $\text{BL}(0.1, 1)$ ,  
997  $\text{St}(2)$ , and  $\text{Laplace}(1)$  increment distributions.

998 The coverage results for  $\alpha = 0.9$  are presented in [Table 7](#). The Gibbs row again validates the  
999 implementation; for all forward operators, they achieve coverages that are very close to 0.9. In  
1000 contrast, the coverage values obtained by the DPS algorithms are generally much smaller than 0.9.  
1001 For C-DPS and DiffPIR, the reported coverage values are almost always 0 except for  $\text{BL}(0.1, 1)$   
1002 and  $\text{St}(1)$  increments, where the coverages are usually (close to) 1 for C-DPS and inconsistent for  
1003 DiffPIR. For almost all increment distributions and forward operators, DPnP reports coverage values  
1004 that are closest to, but typically smaller than, 0.9. Note that a coverage of 1 can be considered the  
1005 worst case even at a target of 0.9. For instance, it would be achieved by setting all samples to a  
1006 constant vector with extremely large (*i.e.*, “unlikely”) entries.

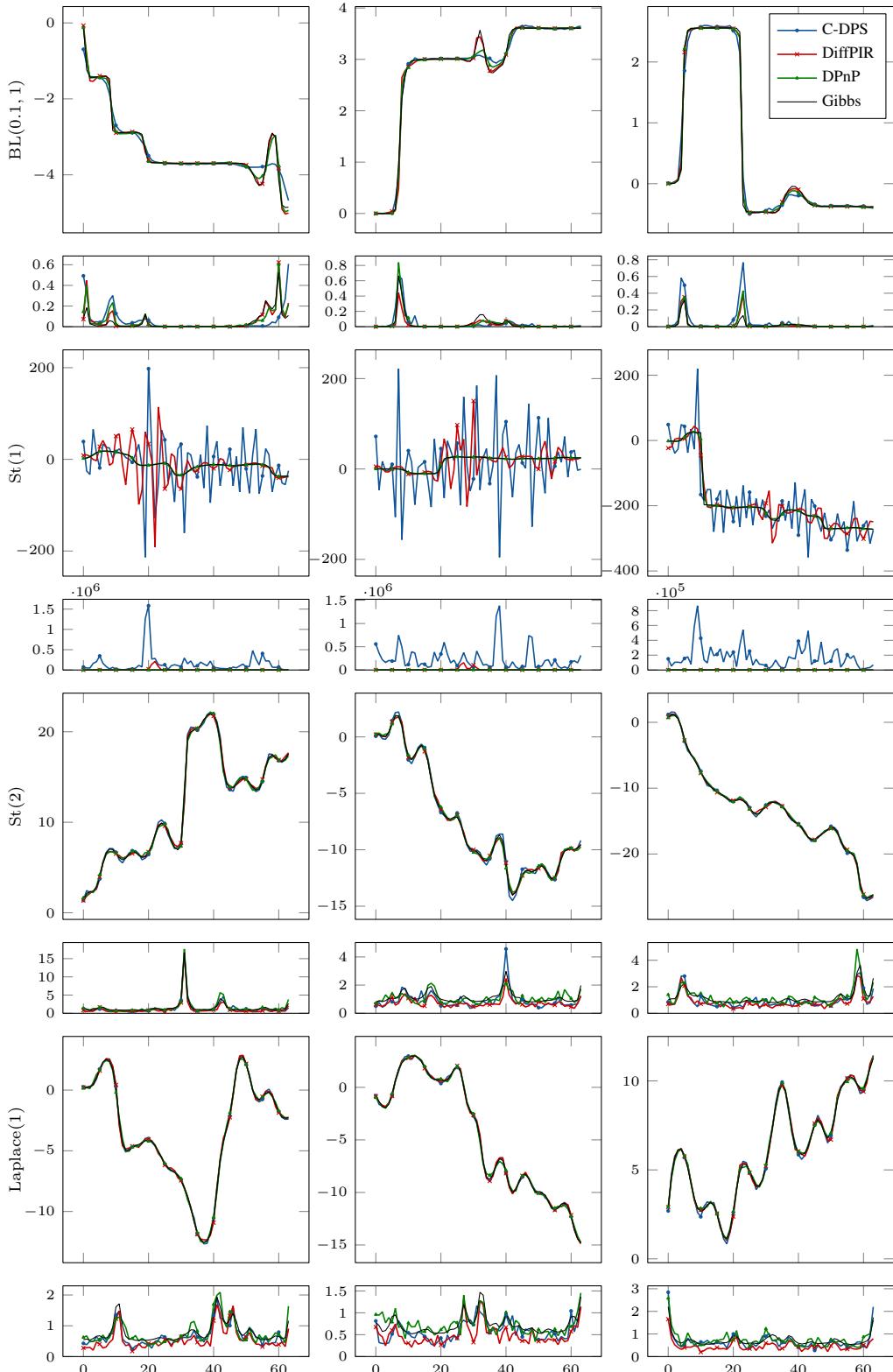


Figure 11: Qualitative results for deconvolution using the Monte Carlo. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

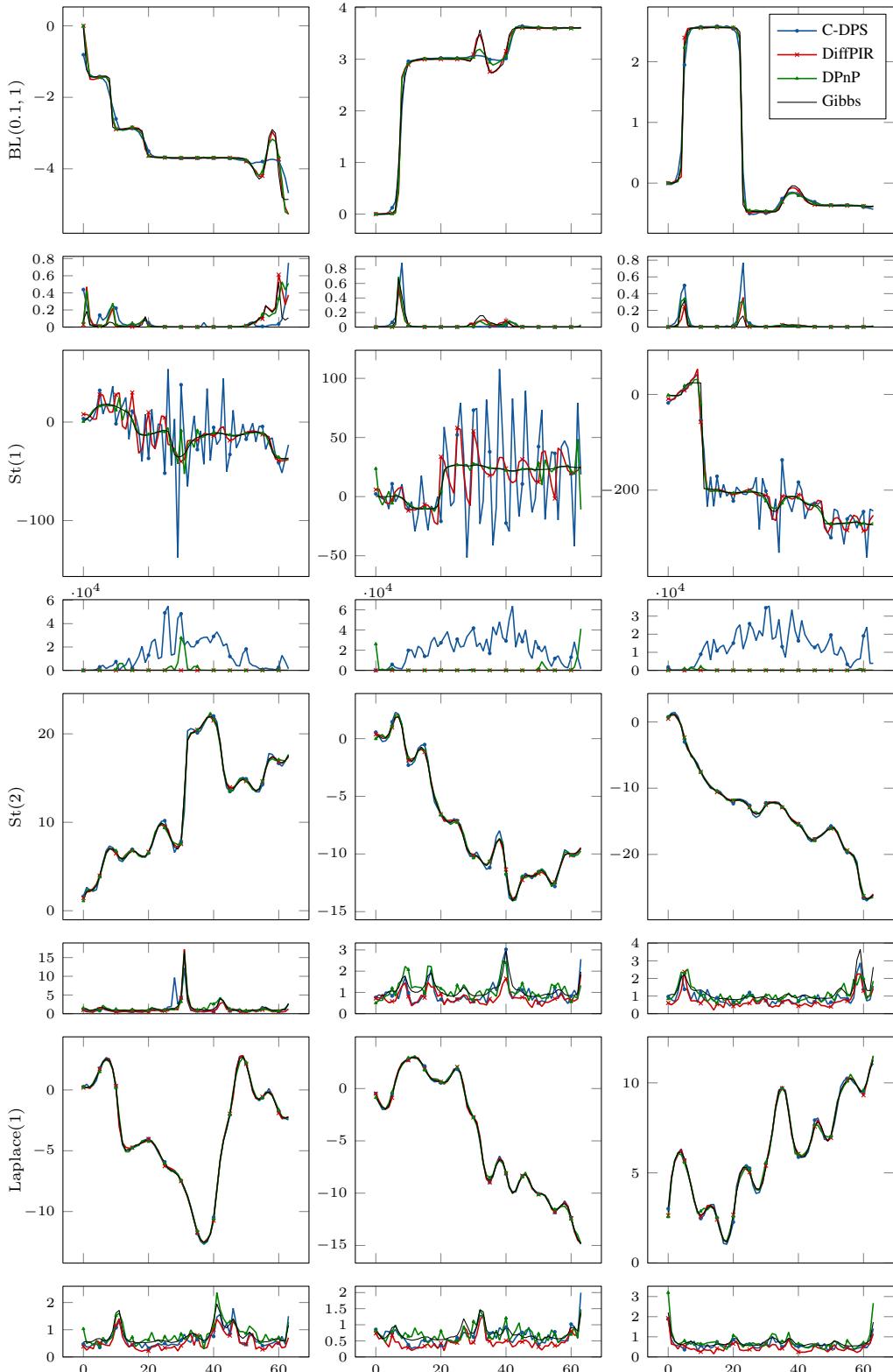


Figure 12: Qualitative results for deconvolution using the learned denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

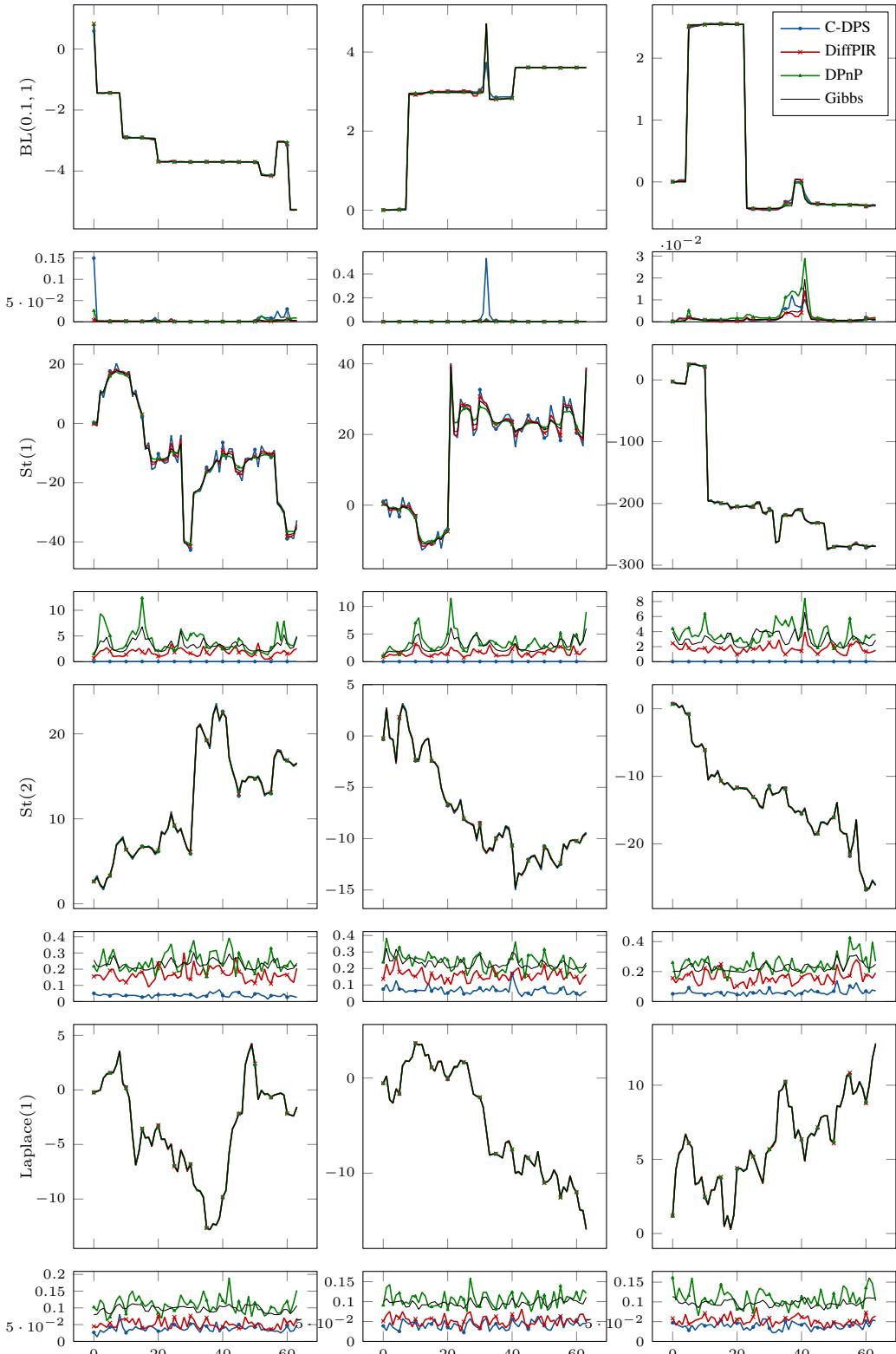


Figure 13: Qualitative results for denoising using the Monte Carlo denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

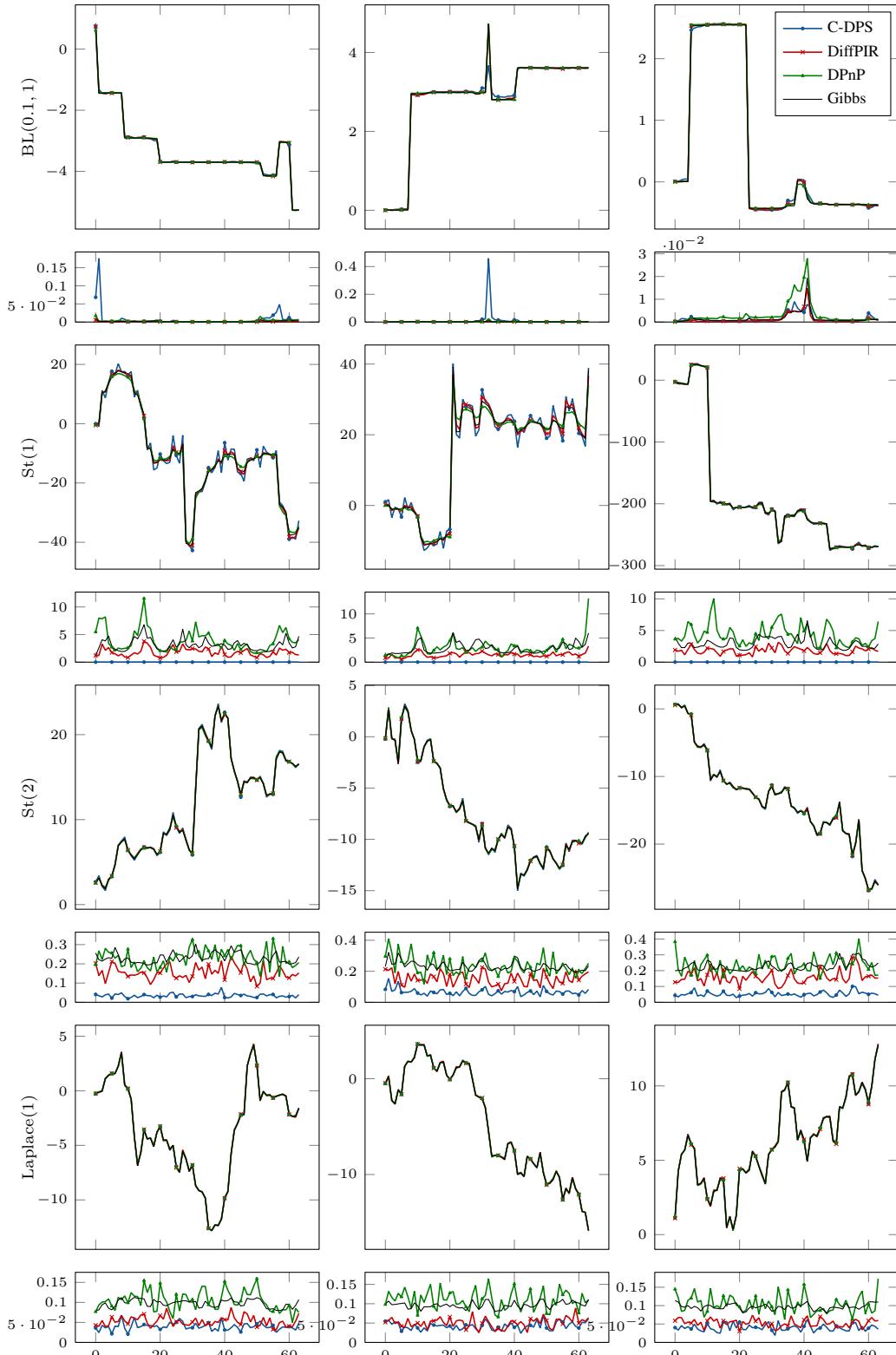


Figure 14: Qualitative results for denoising using the learned denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

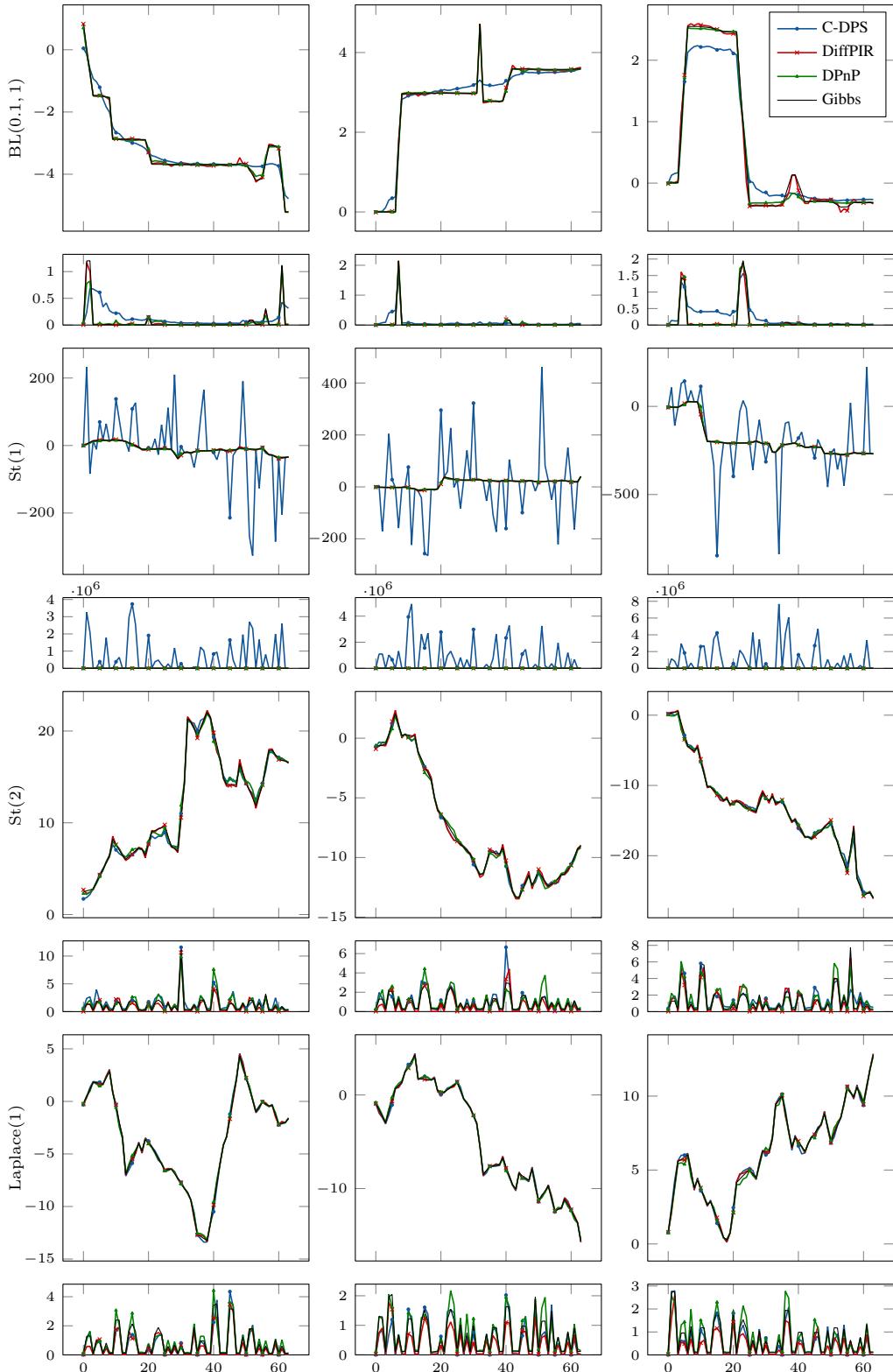


Figure 15: Qualitative results for imputation using the Monte Carlo denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

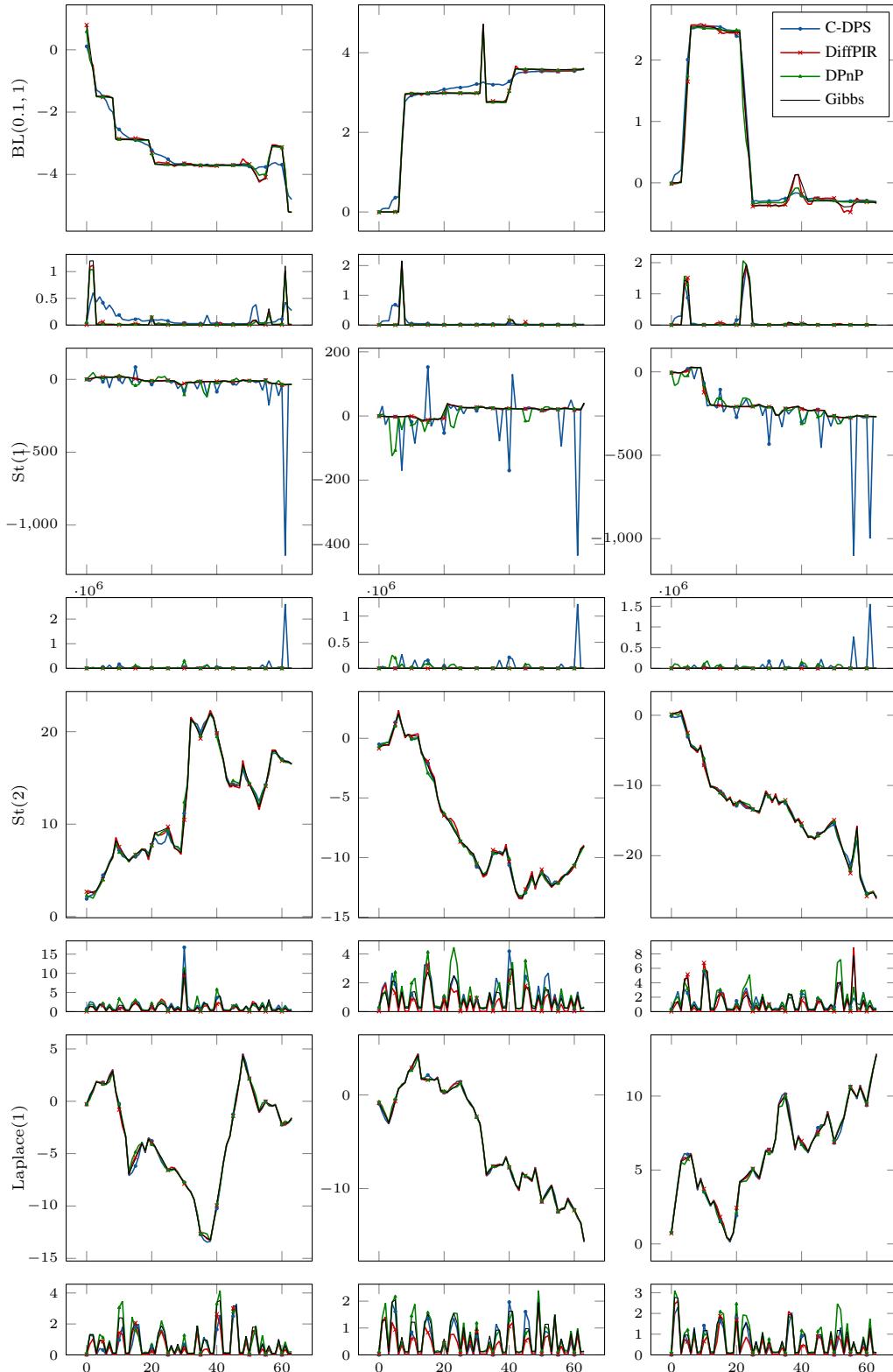


Figure 16: Qualitative results for imputation using the learned denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

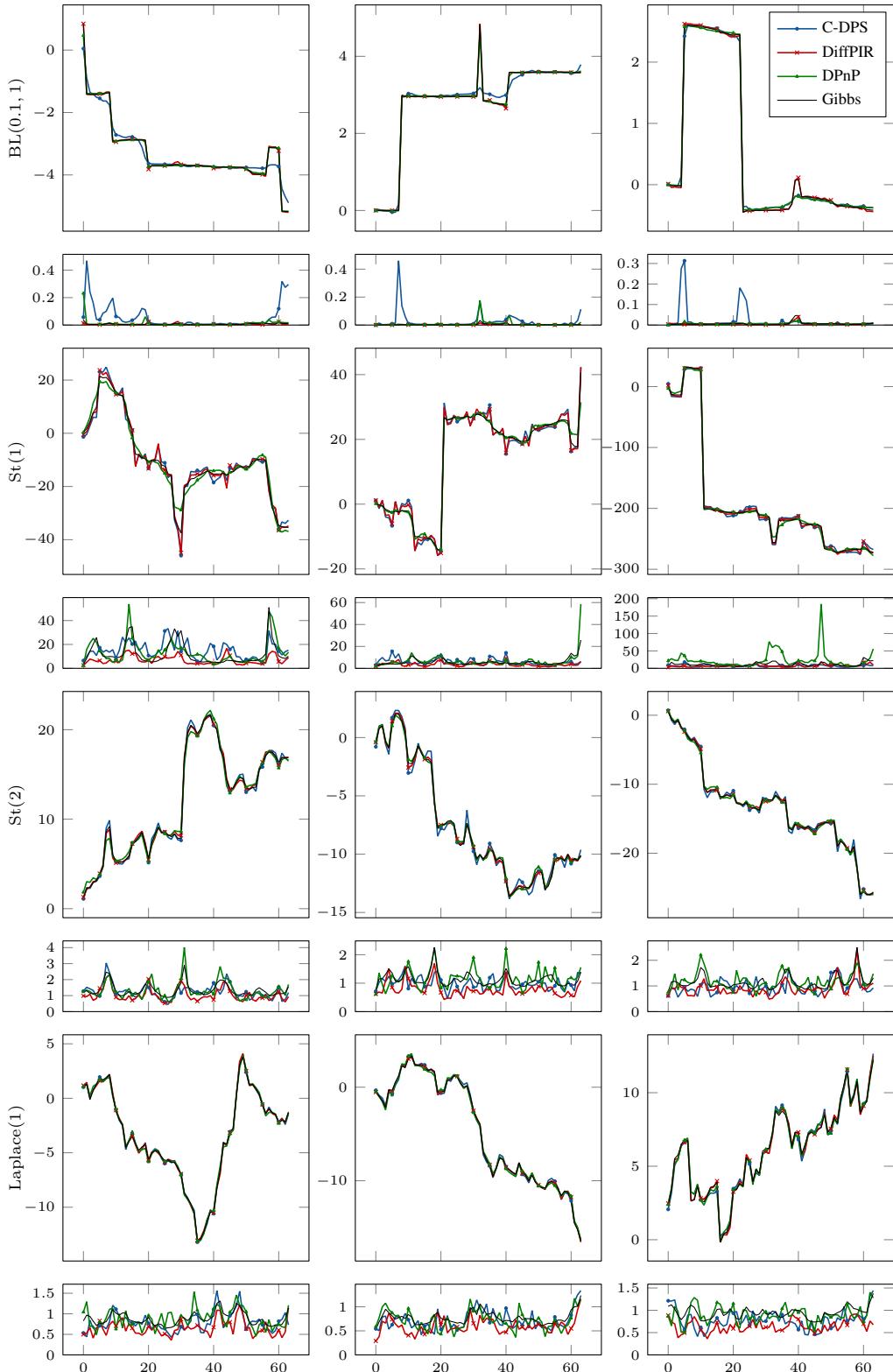


Figure 17: Qualitative results for reconstruction from partial Fourier measurements using the Monte Carlo denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

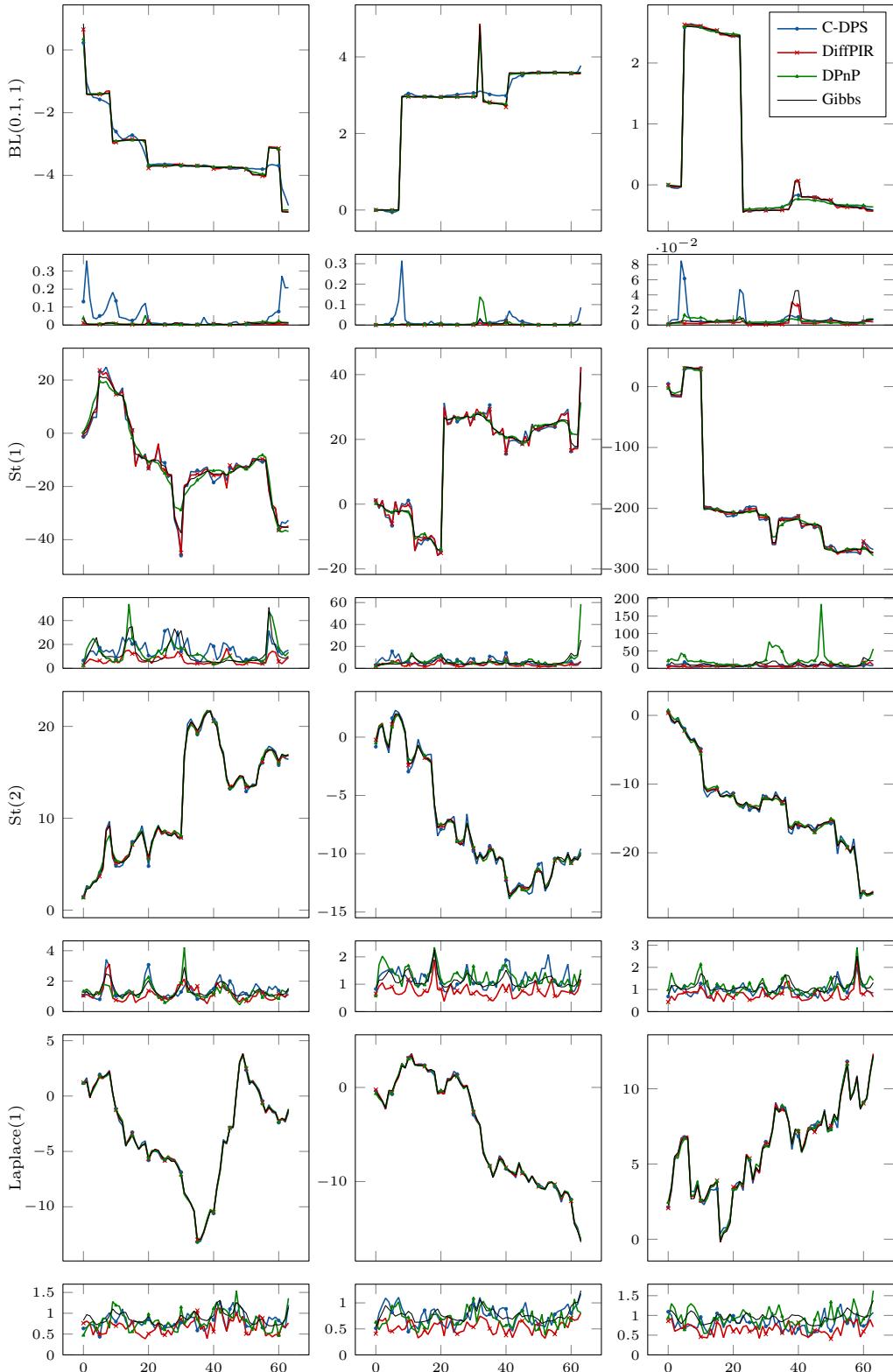


Figure 18: Qualitative results for reconstruction from partial Fourier measurements using the learned denoiser. Rows: increment distributions. For each increment distribution, the MMSE estimates obtained by the different DPS algorithms and the gold-standard Gibbs methods are shown on top of the corresponding index-wise marginal variances. Columns: Different measurements.

Table 6: Runtime of the benchmark with Monte Carlo objects.

		Gauss(0.25)	Laplace(1)	BL(0.1, 1)	St(1)	St(2)	St(3)
Denoising	C-DPS	05:52:28	07:23:23	34:07:44	05:52:40	05:34:51	05:31:10
	DiffPIR	05:04:40	06:36:25	33:46:29	05:12:30	05:39:06	05:38:24
	DPnP	00:03:04	00:03:57	00:20:36	00:03:23	00:03:08	00:03:10
Deconvolution	C-DPS	05:53:40	07:25:17	34:17:12	05:28:38	05:24:12	05:24:00
	DiffPIR	05:28:09	06:55:34	34:16:17	05:31:29	05:32:32	05:22:39
	DPnP	00:03:05	00:03:59	00:21:01	00:03:13	00:03:21	00:03:21
Imputation	C-DPS	05:49:07	07:15:41	34:29:37	05:53:44	05:27:44	05:26:05
	DiffPIR	05:50:15	07:00:13	33:52:26	05:34:00	05:24:16	05:09:56
	DPnP	00:03:23	00:04:18	00:20:58	00:03:09	00:03:05	00:03:22
Fourier	C-DPS	05:49:49	07:09:51	34:30:13	05:49:44	05:49:26	05:49:07
	DiffPIR	05:13:06	06:38:32	34:31:38	05:17:58	06:14:52	05:15:14
	DPnP	00:03:04	00:04:12	00:20:59	00:03:05	00:03:19	00:03:32

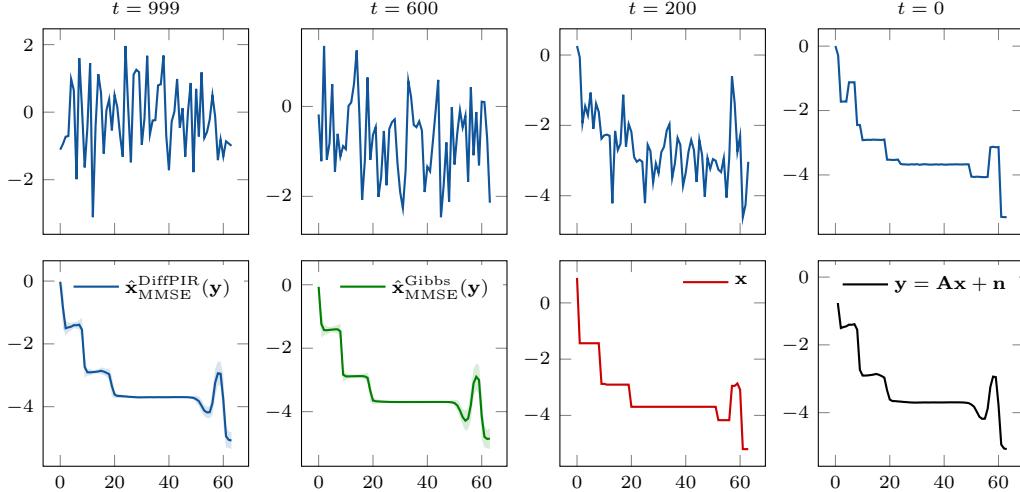


Figure 19: Conditional generation for deconvolution of a signal with  $BL(0.1, 1)$  increments with DiffPIR. Top: Prototypical sampling trajectory at times  $t = 999, 600, 200, 0$ . Bottom: From left to right: MMSE estimate obtained by averaging all DiffPIR samples; gold-standard MMSE estimate obtained by the Gibbs method; the data-generating signal; the data.

Table 7: Posterior coverage of various estimation methods at  $\alpha = 0.9$ . MC: Monte Carlo.

	Gauss(0, 0.25)		Laplace(1)		BL(0.1, 1)		St(1)		St(2)		St(3)		
	Learned	MC	Learned	MC	Learned	MC	Learned	MC	Learned	MC	Learned	MC	
Denoising	Gibbs	—	0.90	—	0.91	—	0.91	—	0.89	—	0.91	—	0.89
	C-DPS	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
	DiffPIR	0.00	0.00	0.00	0.00	1.00	1.00	0.28	0.02	0.00	0.00	0.00	0.00
	DPnP	0.58	0.67	0.11	0.11	1.00	0.41	0.53	0.08	0.09	0.09	0.09	0.10
Deconvolution	Gibbs	—	0.89	—	0.90	—	0.90	—	0.91	—	0.91	—	0.91
	C-DPS	0.00	0.00	0.01	0.00	1.00	1.00	1.00	0.83	0.01	0.00	0.00	0.00
	DiffPIR	0.00	0.00	0.00	0.00	1.00	1.00	0.97	0.92	0.00	0.00	0.00	0.00
	DPnP	0.12	0.12	0.06	0.07	1.00	0.31	0.50	0.06	0.06	0.07	0.06	0.06
Imputation	Gibbs	—	0.89	—	0.90	—	0.86	—	0.91	—	0.91	—	0.91
	C-DPS	0.00	0.00	0.00	0.00	1.00	1.00	0.94	0.78	0.15	0.15	0.00	0.00
	DiffPIR	0.00	0.00	0.00	0.00	1.00	1.00	0.72	0.32	0.00	0.00	0.00	0.00
	DPnP	0.28	0.31	0.09	0.08	1.00	0.41	0.56	0.07	0.14	0.13	0.12	0.13
Fourier	Gibbs	—	0.91	—	0.90	—	0.90	—	0.91	—	0.92	—	0.91
	C-DPS	0.00	0.00	0.00	0.00	1.00	1.00	0.96	0.74	0.01	0.01	0.00	0.00
	DiffPIR	0.00	0.00	0.00	0.00	1.00	1.00	0.92	0.65	0.00	0.01	0.00	0.00
	DPnP	0.19	0.19	0.08	0.06	1.00	0.32	0.50	0.06	0.07	0.07	0.07	0.06