Generative Modeling Project 4
Plug-and-play split Gibbs sampler

Rayane Kimouche
ENS Paris Saclay
rayane.kimouche@ens-paris-saclay.fr

Hugo Queniat
Télécom Paris
hugo.queniat@telecom-paris.fr

Abstract

This work explores the novel Plug-and-Play Split Gibbs Sampler (**PnP-SGS**), an advanced approach to address complex imaging inverse problems. The **PnP-SGS** method leverages the principles of variable splitting and Gibbs Sampling for effective posterior distribution sampling. The proposed technique uniquely integrates denoising diffusion probabilistic models (DDPM) within the Gibbs Sampling framework to enhance the regularization process. This allows for a comprehensive interpretation of the posterior distribution beyond the point estimates typically yielded by PnP optimization algorithms. We elucidate the theoretical underpinnings of **PnP-SGS** and demonstrate its application to image inpainting through our experiments while evaluating its efficiency, advantages and flaws.

1 Introduction

1.1 Article Overview

We examined the paper titled "Plug-and-play split Gibbs sampler: integrating deep generative priors into Bayesian inference" authored by Florentin Coeurdoux, Nicolas Dobigeon, and Pierre Chainais [1]. This study presents a cutting-edge plug-and-play (PnP) approach, employing split Gibbs sampling (SGS) to adeptly draw samples from a posterior distribution.

1.2 Theoretical Framework

The newly proposed sampler is crafted for posterior distribution sampling tasks, which involve deducing a variable $\mathbf{x} \in \mathbb{R}^n$ from an observation $\mathbf{y} \in \mathbb{R}^M$ that may be noisy or incomplete. This inverse problem is typically posed as the following optimization challenge:

$$\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) + g(\mathbf{x}) \tag{1}$$

Here, $f(\mathbf{x}, \mathbf{y})$ denotes the data-fidelity term, while $g(\mathbf{x})$ serves as the regularization term as we have seen during the course over PnP algorithms. The PnP paradigm applied in this sampler leverages a variable splitting technique, akin to the alternating direction method of multipliers (ADMM) [2] or to the half-quadratic splitting (HQS) [3], methods we saw while dealing with proximal splitting [4] during the classes. The essence of this technique involves introducing an auxiliary variable $\mathbf{z}$, transforming the problem (1) into:

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}, \mathbf{y}) + g(\mathbf{z}) \quad \text{subject to} \quad \mathbf{x} = \mathbf{z} \tag{2}$$

The equality constraint ensures that solving (2) is equivalent to solving the initial problem (1). Utilizing an alternating minimization approach, this strategy allows for independent handling of

the data-fidelity term and the regularization, culminating in Gibbs Sampling. Furthermore, The sub-problem concerning $\mathbf{z}$ is addressed via the regularization term's proximal operator as we will see in (5), interpreted as a denoising operation like we discussed in our lectures, permitting the use of a plug-and-play framework.

Nonetheless, PnP optimization algorithms typically yield singular estimations. In contrast, this document, applying the methodology for Bayesian inference, allows to have a complete comprehension of the ensuing posterior distribution :

$$\pi(\mathbf{x}) = p(\mathbf{x} \mid \mathbf{y}) \propto \exp[-f(\mathbf{x}, \mathbf{y}) - g(\mathbf{x})] \tag{3}$$

1.3 Split Gibbs Sampling Framework

The posterior distribution spurs the utilization of the splitting approach by the introduction of a supplemental variable, $\mathbf{z}$, resulting in an augmented distribution:

$$\pi_\rho(\mathbf{x}, \mathbf{z}) = p\left(\mathbf{x}, \mathbf{z} \mid \mathbf{y}; \rho^2\right) \tag{4}$$

$$\propto \exp\left[-f(\mathbf{x}, \mathbf{y}) - g(\mathbf{z}) - \frac{1}{2\rho^2}\|\mathbf{x} - \mathbf{z}\|_2^2\right] \tag{5}$$

where ρ is a parameter modulating the interdependence between $\mathbf{x}$ and $\mathbf{z}$. As established in the literature [5], we have

$$\|\pi - \pi_{\rho,\mathbf{X}}\|_{\mathrm{TV}} \underset{\rho^2 \to 0}{\to} 0 \tag{6}$$

indicates that the marginal of π_ρ asymptotically concurs with the target π, rendering this technique an exact data augmentation method. Consequently, in the limiting scenario where ρ approaches zero, sampling from π_ρ becomes feasible, and traditional Gibbs procedures can be applied, each variable sampled sequentially from its respective conditional distribution:

$$p\left(\mathbf{x} \mid \mathbf{z}, \mathbf{y}; \rho^2\right) \propto \exp\left[-f(\mathbf{x}, \mathbf{y}) - \frac{1}{2\rho^2}\|\mathbf{x} - \mathbf{z}\|_2^2\right] \tag{7}$$

$$p\left(\mathbf{z} \mid \mathbf{x}; \rho^2\right) \propto \exp\left[-g(\mathbf{z}) - \frac{1}{2\rho^2}\|\mathbf{z} - \mathbf{x}\|_2^2\right] \tag{8}$$

This method of sampling allows us to separate the potential functions associated to $\mathbf{x}$ and $\mathbf{z}$: $f(\cdot, \mathbf{y})$ and $g(\cdot)$. This useful property is also seen in most deterministic alternatives to SGS, like HQS and ADDM we discussed earlier.

When analyzing the distributions individually, the process of sampling from Equation (7) can be interpreted as addressing the initial inverse problem (1) equipped with a Gaussian prior. This prior has a mean of $\mathbf{z}$ and a covariance matrix defined as $\mathbf{\Sigma} = \rho^2\mathbf{I}$. Given its reliance on a standard Gaussian prior, sampling from this modified posterior distribution is more straightforward than from the original posterior described in Equation (3). This adjustment to the sampling process, being problem-specific, will be examined in greater detail in the forthcoming section, emphasizing the application of **PnP-SGS** to inverse problems.

Regarding Equation (8), this particular distribution mirrors the posterior distribution seen in Bayesian denoising tasks, where the aim is to recover $\mathbf{z}$ from $\mathbf{x}$ observations contaminated by additive Gaussian noise with a variance of ρ^2. The authors propose to use stochastic denoisers instead of sampling directly from (8). In particular, they chose denoising diffusion probabilistic models (DDPM) [6, 7, 8] to tackle this particular task. We were first introduced to DDPM during the class over diffusion models and were able to use them to tackle inverse problems as a homework. However, one can note that the flexibility of the PnP framework permits the incorporation of various denoising generative models into this phase of the sampling process.

1.4 DDPM as Stochastic Denoisers

As we have seen during the course on Diffusion models, DDPM [6] distinguishes itself from other types of latent variables models by the fact that the forward process of the model consists in constructing a Markov Chain which gradually adds Gaussian noise to the data :

$$p\left(\mathbf{u}_t \mid \mathbf{u}_{t-1}\right)=\mathcal{N}\left(\mathbf{u}_t ; \sqrt{1-\beta(t)} \mathbf{u}_{t-1}, \beta(t) \mathbf{I}\right) \tag{9}$$

where $\beta(t) \in (0,1)$ is a predefined function.

Following this factorization of the Markov Chain, one can express in closed form [1] any point within in the chain with respect to the initial data, using the notation $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, we have :

$$p\left(\mathbf{u}_t \mid \mathbf{u}_0\right)=\mathcal{N}\left(\mathbf{u}_t ; \sqrt{\bar{\alpha}_t} \mathbf{u}_0,\left(1-\bar{\alpha}_t\right) \mathbf{I}\right) \tag{10}$$

This formulation reveals that any image at the t^* step in the forward diffusion process is essentially the original image perturbed by Gaussian white noise of variance $(1 - \bar{\alpha}_t)$. Therefore, DDPM can be utilized as stochastic Gaussian denoisers by reversing the Markov Chain's direction using the reverse transition kernel:

$$q_{\boldsymbol{\theta}}\left(\mathbf{u}_{t-1} \mid \mathbf{u}_t\right)=\mathcal{N}\left(\mathbf{u}_{t-1} ; \boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{u}_t, t\right), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\left(\mathbf{u}_t, t\right)\right) \tag{11}$$

The ensuing strategy involves employing this reverse transition kernel (11) as a denoising mechanism to effectively sample from (8). However, given that DDPMs are inherently generative models designed to synthesize images from noise, the objective here narrows down to conducting a limited number of reverse operations. Starting the reverse diffusion process from a designated point $t^* < T$ within the Markov Chain ensures that the initial image for the reverse process is merely a noise-affected image rather than a pure noise realization. Therefore, determining t^*, which corresponds to the equivalent number steps of the forward diffusion process coming from sampling from (7), and then produce t^* steps of the reverse process.

1.5 Determining t^*

We have seen previously that after t^* steps of the forward diffusion process, the image $\mathbf{u}_{t^*}$ corresponds to the original image corrupted by a Gaussian white noise of variance $(1 - \bar{\alpha}_t)$. Furthermore, given the strict monocity of $\beta(.)$, and consequently $\bar{\alpha}(t)$, it is evident that an increase in t^* correlates with an increase in the noise level of the image u_{t^*}. Crucially, this also means that any level of noise $(1 - \bar{\alpha}(t^*))$ within the forward Markov Chain is uniquely associated to a single timestamp t^*.

Given this observation, the method for sampling $\mathbf{z}^{(n)}$ relies on this one-to-one correlation to evaluate the number t^* of steps required for the backward diffusion procedure, i.e. the number of denoising steps in (11). Consequently, we would like to estimate the level of noise corrupting the current sample $\mathbf{x}^{(n)}$. The paper opted for the strategy proposed in [9], implemented in the library `skimage.restoration` as the function `estimate_sigma()`.

Post computing $\hat{\sigma} = \Phi(\mathbf{x}^{(n)})$, indicative of the current sample's noise level, deducing $\widehat{t^*}$ reduces to the computation of the inverse of $(1 - \bar{\alpha}(t^*))$. Going back to its very definition, this evaluation relies on the diffusion scheduling function, $\beta(\cdot)$. The paper offers various choices for this function as there are in the literature. However, we chose to keep only one, the scheduling function we used in class with the DDPM, a linearly increasing function such that :

$$\beta(t) = \beta(0) + rt \tag{12}$$

where $\beta(0) = 10^{-4}$ and the slope r is taken such that $\beta(T) = 2.0 \times 10^{-2}$ [6]. This simplistic scheduling mechanism facilitates straightforward inverse computation. Nonetheless, since we can

[1] Here, our paper [1] is mistaken as it indicates a covariance matrix of $\bar{\alpha}_t \mathbf{I}$. We thus stick to the initial notations from [6].

compute and store easily $(\bar{\alpha}(t))_{0\leq t\leq T}$ with t distributed evenly across the space of our diffusion process, $T = 1000$ samples, we chose an array strategy was preferred for computing $\hat{t^*}^2$ as per the following :

$$\widehat{t^*} = \underset{t\in\{0,\dots,T\}}{\operatorname{argmin}} \left|1 - \bar{\alpha}(t) - \hat{\sigma}^2\right| \tag{13}$$

2 Solving Inverse Problems with PnP-SGS

The PnP-SGS method has been instantiated by the authors for three distinct imaging inverse problem: deblurring, inpainting, and super-resolution, following the protocols set by [10]. In essence, one observes a deteriorated image $\mathbf{y} \in \mathbb{R}^M$ with the ambition of deriving the restored image $\mathbf{x} \in \mathbb{R}^N$, all encaplsulated in this linear framework :

$$\mathbf{y} = \mathbf{H}^\top \mathbf{x} + \mathbf{n} \tag{14}$$

where $\mathbf{H} \in \mathbb{R}^{N\times M}$ denotes the linear operator.[3] specific to the inverse problem and $\mathbf{n}$ a Gaussian noise with covariance matrix $\mathbf{\Omega}^{-1} \in \mathbb{R}^{M\times M}$ which accounts for both noise and errors of modeling and measurement.

Through linear operations on the Gaussian random vector $\mathbf{n}$, we easily obtain the likelihood of the observation $\mathbf{y}$

$$p(\mathbf{y} \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{y}; \mathbf{H}^\top \mathbf{x}, \mathbf{\Omega}^{-1}\right)$$

Now, we can plug-in this new information on our computation of the posterior augmented distribution (7).

$$\begin{aligned}
p\left(\mathbf{x} \mid \mathbf{z}, \mathbf{y}; \rho^2\right) &\propto p(\mathbf{x}, \mathbf{y} \mid \mathbf{z}) \\
&\propto p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{y} \mid \mathbf{x}, \mathbf{z}) \\
&\propto \exp\left(-\frac{1}{2\rho^2}(\mathbf{x}-\mathbf{z})^\top(\mathbf{x}-\mathbf{z}) - \frac{1}{2}\left(\mathbf{H}^\top\mathbf{x}-\mathbf{y}\right)^\top \mathbf{\Omega}(\mathbf{H}^\top\mathbf{x}-\mathbf{y})\right) \\
&\propto \exp\left(-\frac{1}{2}\mathbf{x}^\top\left(\mathbf{H}\mathbf{\Omega}\mathbf{H}^\top + \frac{1}{\rho^2}\mathbf{I}_N\right)\mathbf{x} - \frac{1}{2}\left(\frac{-2}{\rho^2}\mathbf{x}^\top - 2\mathbf{x}^\top\mathbf{H}\mathbf{\Omega}\mathbf{y}\right)\right)
\end{aligned}$$

with $\mathbf{Q}_\mathbf{x} = \mathbf{H}\mathbf{\Omega}\mathbf{H}^\top + \frac{1}{\rho^2}\mathbf{I}_N$, we get

$$\begin{aligned}
p\left(\mathbf{x} \mid \mathbf{z}, \mathbf{y}; \rho^2\right) &\propto \exp\left(-\frac{1}{2}\mathbf{x}^\top\mathbf{Q}_\mathbf{x}\mathbf{x} + \mathbf{x}^\top\left(\frac{\mathbf{z}}{\rho^2} + \mathbf{H}\mathbf{\Omega}\mathbf{y}\right)\right) \\
&\propto \exp\left(-\frac{1}{2}\mathbf{x}^\top\mathbf{Q}_\mathbf{x}\mathbf{x} + \mathbf{x}^\top\mathbf{Q}_\mathbf{x}\mathbf{Q}_\mathbf{x}^{-1}\left(\frac{\mathbf{z}}{\rho^2} + \mathbf{H}\mathbf{\Omega}\mathbf{y}\right)\right) \\
&\propto \exp\left[-\frac{1}{2}\left(\mathbf{x} - \mathbf{Q}_\mathbf{x}^{-1}\left(\mathbf{H}\mathbf{\Omega}\mathbf{y} + \frac{1}{\rho^2}\mathbf{z}\right)\right)^\top \mathbf{Q}_\mathbf{x}\left(\mathbf{x} - \mathbf{Q}_\mathbf{x}^{-1}\left(\mathbf{H}\mathbf{\Omega}\mathbf{y} + \frac{1}{\rho^2}\mathbf{z}\right)\right)\right]
\end{aligned}$$

Therefore, we have shown that in the context of a classical inverse problem, sampling from (7) reduces to sampling from a high dimensional multivariate Gaussian distribution :

$$p\left(\mathbf{x} \mid \mathbf{z}, \mathbf{y}; \rho^2\right) = \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_\mathbf{x}, \mathbf{Q}_\mathbf{x}^{-1}\right) \tag{15}$$

with

$$\begin{cases} \mathbf{Q}_\mathbf{x} = \mathbf{H}\mathbf{\Omega}\mathbf{H}^T + \frac{1}{\rho^2}\mathbf{I}_N \\ \boldsymbol{\mu}_\mathbf{x} = \mathbf{Q}_\mathbf{x}^{-1}\left(\mathbf{H}\mathbf{\Omega}\mathbf{y} + \frac{1}{\rho^2}\mathbf{z}\right) \end{cases} \tag{16}$$

[2]The previous mistake in [1] leads to another in this estimation of the denoising steps necessary, rendering the ensuing algorithm non convergent.

[3]The initial paper contains an error regarding the dimensions of the linear operator in the section on inpainting. To maintain consistency with the paper and our code, which was initially derived from the paper, we have preserved the introduction of the linear operator as presented but have transposed it to correct the dimensions.

2.1 Image inpainting

In the specific context of image inpainting addressed in our following experiments, the analytic process is considerably easier due to the linear operator's structure. In this scenario, the objective is to recover the original image $\mathbf{x}$ from the noisy and partial observation $\mathbf{y}$. Therefore, the linear operator $\mathbf{H} \in \{0,1\}^{N \times M}$ now represents a binary matrix corresponding to an irregular subsampling with $M \ll N$, implying a significant reduction in the number of observed pixels relative to the total. The noise, in this scenario, is presumed to be white and Gaussian such that the covariance matrix is simpler : $\mathbf{\Omega}^{-1} = \sigma^2 \mathbf{I}_M$. This makes (16) computations simpler :

$$\begin{cases} \mathbf{Q}_{\mathbf{x}} = \frac{1}{\sigma^2}\mathbf{H}\mathbf{H}^T + \frac{1}{\rho^2}\mathbf{I}_N \\ \boldsymbol{\mu}_{\mathbf{x}} = \mathbf{Q}_{\mathbf{x}}^{-1}\left(\frac{1}{\sigma^2}\mathbf{H}\mathbf{y} + \frac{1}{\rho^2}\mathbf{z}\right) \end{cases} \tag{17}$$

Going further, knowing that the operator $\mathbf{H}^\top$ consists, in this case, of a subset of rows of the identity matrix $\mathbf{I}_N$ [10], we observe $\mathbf{H}^\top\mathbf{H} = \mathbf{I}_M$ and the Sherman-Morrison-Woodbury formula [11] yields

$$\mathbf{Q}_{\mathbf{x}}^{-1} = \rho^2\left(\mathbf{I}_N - \frac{\rho^2}{\sigma^2 + \rho^2}\mathbf{H}\mathbf{H}^\top\right) \tag{18}$$

This observation is crucial considering the size of the matrices we deal with in our experiments. Even though the matrices are sparse, dealing with 256×256 RGB images leads $\mathbf{Q}_{\mathbf{x}}$ to be squared of size $196,608$. Knowing that in the most general case, the complexity of the computation of the inverse of a matrix is $O(p^3)$ with p its size, this formula is essential for computing times.

3 Comparison between PNP-SGS and DPS

3.1 Spliting Sampling-Based Method

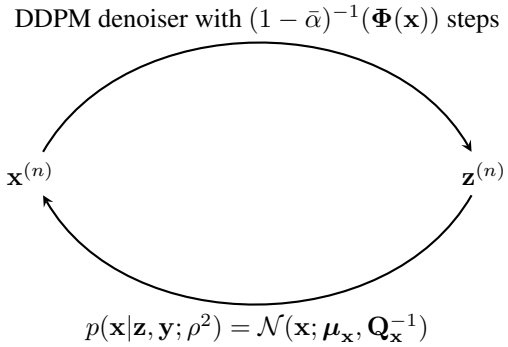

Figure 1: Reverse Inpainting Process

As previously outlined, this method alternatively samples from the conditional posterior distributions of the augmented posterior distribution (5) using Split Gibbs Sampling. In the first step of SGS, sampling according to the conditional posterior distribution which is Gaussian, $p(\mathbf{x}|\mathbf{z},\mathbf{y};\rho^2) = \mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{\mathbf{x}},\mathbf{Q}_{\mathbf{x}}^{-1})$. The second step for regularization will involve stochastic denoising of the current sample using a pre-trained DDPM (Unet in our case). The number of denoising steps in the backward diffusion process, denoted as t^*, plays a crucial role in adjusting the amount of regularization applied during sampling. It is determined based on the noise level estimated from the current sample, allowing for adaptive regularization.

3.2 Gradient Correction-Based Method

This method is based on Diffusion Posterior Sampling (DPS). It aims to approximate the likelihood function $p(y|\mathbf{x}_t)$ by factorizing it into tractable components. It leverages the relationship between the posterior mean $\mathbf{x}_0$ and the observed data $\mathbf{y}$ to approximate the likelihood function. The method

provides a tractable approximation for $p(y|\mathbf{x}_t)$, allowing for posterior sampling based on maximizing the likelihood. So, in order to reach the distribution of the desired image, it only uses the usual sampling (or unconditionnal) difussion backward process, but this time by adding a correction term $\nabla_{x_t}\|A\hat{x}_0(x_t,t)-y\|^2$. This term is added in order to perform a look ahead in the estimated x_0 then trying to estimate back the degraded version of it, intending to approach the the original distibution of y.

4 Experiments

Despite the absence of reproducible code supplied with the paper, we tried to experiment this approach with our own implementation of Algorithm 1, whose pseudo-code can be seen in appendix A. Specifically, we utilized it in one of the linear problems which is **"Inpainting"** and compared it to the gradient correction based approach DSP (Diffusion Posterior Sampling) which we saw in class. The reverse process of PNP-SGS is illustrated in the figure 1. Our experiment involves the following steps :

- We choose an image from the net, distinct to what had been used in the paper.
- We create an **Inpainting** mask that covers 75% of the RGB pixels and then derive the operator H.
- All matrices are computed as sparse because of the high dimensionality and its computing costs.
- To alleviate the potential memory constraints, there were some supplementary tricks added such as chunk sum in order to address this limitations.

The experiment was to perform inpainting over a noisy and partial image of a young girl face. This mask was generated randomly so as to keep 128×128 pixels across all 3 channels of the RGB image, thus accounting for the 75% of masked pixels.

The authors were rather not very talkative and precise about the choice of hyperparameters, whether about ρ, σ or the number of steps necessary both in terms of burn-in or for the complete Monte Carlo Markov Chain process. They did give the set of parameters they used -except for σ- but did not justify their choices. Upon testing on our own, we found that the following set performed satisfyingly :

- $\rho = 0.33$
- $\sigma = 0.1$
- $N_{MC} = 150$
- $N_{bi} = 40$

Hence, we chose this parameters for our baseline, and they were used in our experiment presented in figure 2.
To pursue our comparison with the method approached in class, we also performed the reconstruction with DPS, with the output in figure 3.

Figure 2: Reverse Inpainting Reconstruction for PNP-SGS (observed, groundtruth, reconstructed, predicted x0)

Figure 3: Reverse Inpainting Reconstruction for DPS (observed, groundtruth, reconstructed, predicted x0)

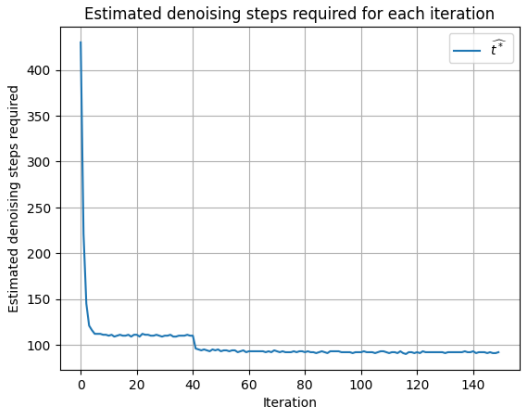

Figure 4: **Inpainting**: evolution of $\widehat{t^*}$ along the PnP-SGS iterations ($T = 1000$).
We observe the convergence of $\widehat{t^*}$ as the paper highlighted. We also see the shift around the 40th step which corresponds to the end of the burn-in phase.

Reconstruction Quality: Regarding the **PnP-SGS** method, we see that the algorithm is able to perform the reconstruction of the face accurately and successfully recovers facial details despite the lack of information about the original image. However, we see that the contrast is not perfect as the reconstruction features lighter colours. Furthermore, we see that the background, and most notably the sky, is not accurately recovered as we just see a really light upper background instead of the clouds in the original image.
Although the output of **PnP-SGS** was satisfying, the algorithm was outperformed by the method DPS we saw in class. This method allowed us to do a complete recovery of the original image without any of the defaults exhibited by the image produced by **PnP-SGS**.

Algorithmic Complexity and Efficiency: One could also discuss the differences in terms of computing times[4] between both methods. Indeed, while **PnP-SGS** took 6 minutes 53 seconds to perform the reconstruction, the DPS was able to recover the original image in less than a minute. Hence, one could argue that the Split Gibbs Sampling is inefficient. However, we should nuance the fact that first, as emphasized by the authors, the algorithm may take longer but it allows a more comprehensive study of the posterior distribution by, for instance, quantifying uncertainties. Furthermore, we highlight the

[4]Times computed on a personal machine equipped with a NVIDIA GeForce RTX 3080, with 10 GB of VRAM.

fact that we had to produce our own implementation of **PnP-SGS** and, certainly, there may be room for improvement in terms of efficiency. Notably, the implementation of the sampling from (7) was quite difficult and required regular shifts between various libraries such as `numpy`, `scipy.sparse` and `torch`. Moreover, we did not implement the exact perturbation-optimization algorithm (E-PO) [12] which, according to the designers of **PnP-SGS**, allows efficient sampling from (7).

Parameters Tuning: As we discussed earlier, the authors did not go in depth about which parameters they used and why. Thus, we delved ourselves in it and focused on the coupling parameter ρ. Indeed, we thought this parameter might be crucial for the convergence of the algorithm since the precision of the data augmentation scheme shown in (6) is only guaranteed asymptotically with respect to ρ. Furthermore, the choices of ρ made in the paper - either 0.7 or 1.625- caught our attention because, well, evidently a value greater than 1 when we have a result in the vicinity of 0 is alarming to us. Nonetheless, this choice did perform for them but it prompted us to test a few values of ρ on our own.

The results of our experiments on this parameter are presented in Figure 5, in appendix B. The figures are unequivocal about the importance of the tuning of this parameter. Indeed, we see that if in the vicinity of $\rho = 0.1$ we get proper reconstructions -not without any defaults though- and that $\rho = 0.33$ seems to perform the best, whenever we choose a ρ a little bit farther, the precision of the method drops dramatically.
Notably, we see that when ρ is too low, for $\rho = 0.01$, the model outputs a face that does not correspond with the one we target. This can be interpreted as a too strong dependence on the variable $\mathbf{z}$ and, hence, a dependence on the initialization. Indeed, the initialization was not discussed by the authors so we chose a simple $\mathbf{z}^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In the specific case of a very low ρ, the first sampled $\mathbf{z}^{(0)}$ will basically be some noise, and we fall in the generative structure of the DDPM. Hence, the output image which is one we would expect from a Generative model.
On the other side of the ladder, we see that the algorithm does not perform aswell, which was foreseen knowing our result on the data augmentation scheme.

5 Conclusion

For this study, we have presented an overview of the PNP-SGS algorithm and its application to the task of image inpainting. Through experiments and comparisons with the DPS method, we have demonstrated the effectiveness of PNP-SGS in achieving accurate reconstructions of the original image content, but there are still some issues to take into account like time complexity and the fine-tuning of its parameters as a faulty choice can lead to misconstructions. While PnP-SGS performed satisfyingly, the method DPS seen in class still appears to have an edge in terms of reconstruction quality and potentially computational efficiency.

References

[1] Florentin Coeurdoux, Nicolas Dobigeon, and Pierre Chainais. Plug-and-play split gibbs sampler: embedding deep generative priors in bayesian inference, 2023.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers, 2010.

[3] Donald Geman and Chengda Yang. Nonlinear image recovery with half-quadratic regularization. *IEEE Transactions on Image Processing*, 4, 1995.

[4] Patrick L. Combettes and Jean Christophe Pesquet. *Proximal splitting methods in signal processing*, volume 49. 2011.

[5] Maxime Vono, Nicolas Dobigeon, and Pierre Chainais. Asymptotically exact data augmentation: Models, properties, and algorithms. *Journal of Computational and Graphical Statistics*, 30, 2020.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. volume 2020-December, 2020.

[7] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. volume 11, 2021.

[8] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. volume 139, 2021.

[9] David L. Donoho and Jain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81, 1994.

[10] Maxime Vono, Nicolas Dobigeon, and Pierre Chainais. Split-and-augmented gibbs sampler - application to large-scale inference problems. *IEEE Transactions on Signal Processing*, 67, 2019.

[11] Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21, 1950.

[12] Yosra Marnissi, Emilie Chouzenoux, Amel Benazza-Benyahia, and Jean Christophe Pesquet. An auxiliary variable method for markov chain monte carlo algorithms in high dimension. *Entropy*, 20, 2018.

A PnP-SGS Algorithm

Algorithm 1 PnP-SGS using DDPM

Input: Parameter ρ^2, total number of iterations N_{MC}, number of burn-in iterations N_{bi}, pre-trained DDPM $s_0(\cdot,\cdot)$, scheduling variance function $\alpha(\cdot)$, initialization $z^{(0)}$
for $n \leftarrow 1$ to N_{MC} **do**
    # Sampling the variable of interest $x^{(n)}$
    Draw $x^{(n)} \sim p(x \mid z, y; \rho^2)$ according to (7)
    # Estimating noise level in $x^{(n)}$
    Set $\hat{\sigma} = \Phi(x^{(n)})$ using [9]
    # Setting the number of diffusion steps to denoise $x^{(n)}$
    Set $t^* = \alpha^{-1}(\hat{\sigma}^2)$
    # Sampling the splitting variable $z^{(n)}$ according to (8)
    Set $u_{t^*} = x^{(n)}$
    for $j \leftarrow t^*$ downto 1 **do**
        Draw $u_{j-1} \sim q(u_{j-1} \mid u_j)$ according to (11)
    end for
    Set $z^{(n)} = u_0$
end for
Output: Collection of samples $\{x^{(n)}, z^{(n)}\}_{t=N_{bi}+1}^{N_{MC}}$ asymptotically distributed according to (3)

B ρ-parameter Turing

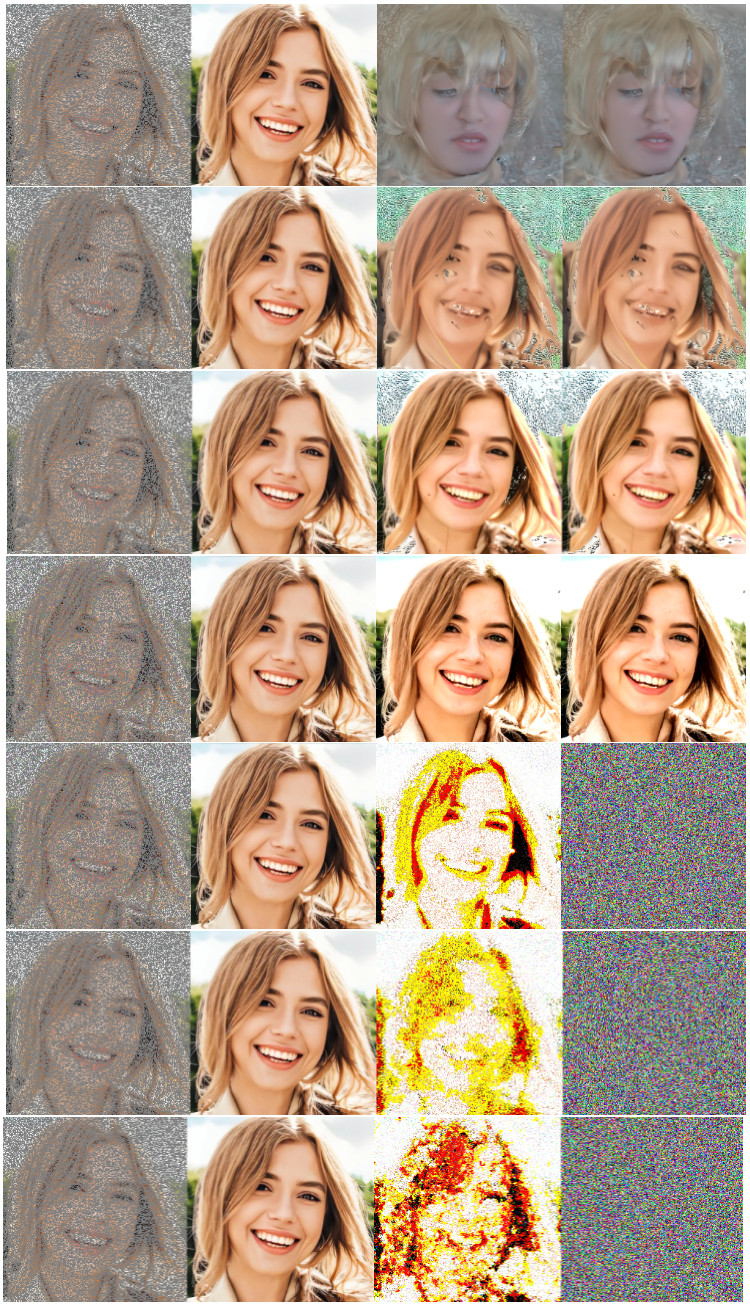

Figure 5: Hyperparameter tuning : the values of ρ are respectively from the top : [0.01, 0.05, 0.1,0.33, 0.5, 1.2, 1.6]