FLOWADMM: PLUG-AND-PLAY ADMM WITH FLOW-BASED RENOISE-DENOISE PRIORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Plug-and-play (PnP) methods for solving inverse problems have recently achieved strong performance by leveraging denoising priors based on powerful generative diffusion and flow models. However, existing diffusion- and flow-based PnP methods typically rely on stochastic renoise-denoise operations, which complicate the analysis of their convergence behavior. In this work, we identify and formalize the deterministic renoise-denoise operator underlying flow-based plug-and-play methods. This perspective reveals that these methods implicitly define a deterministic operator given by the expectation of a denoiser over the latent noise distribution. Building on this insight, we propose *FlowADMM*, a PnP algorithm that integrates the renoise-denoise operator into the classical alternating direction method of multipliers (ADMM) framework. We establish convergence guarantees for FlowADMM under Lipschitz conditions on the underlying flow network, and extend the analysis to non-stationary time schedules. Empirically, FlowADMM matches or surpasses state-of-the-art performance among flow-based PnP methods on a range of inverse problems, including denoising, deblurring, super-resolution, and inpainting, while requiring fewer data consistency evaluations than prior approaches.

1 INTRODUCTION

Inverse problems such as denoising, deblurring, super-resolution, and inpainting are commonly formulated in the form

$$y = Ax + \nu \quad \nu \sim \mathcal{N}(0, \sigma_y^2 I_n), \tag{1}$$

where the goal is to recover the true image $x \in \mathbb{R}^m$ from degraded measurements $y \in \mathbb{R}^n$ and a known (linear) forward operator $A : \mathbb{R}^m \to \mathbb{R}^n$. These problems are often ill-posed. While in recent years deep learning methods were very successful in approximating a direct mapping $\mathcal{G}(y) \approx x$, this approach has two problems: (i) Direct end-to-end approaches often have limited interpretability and theoretical guarantees required for safety-critical applications and (ii) training is costly and each inverse problem requires a separate network to be trained.

On the other hand, classical variational approaches formulate this task as a minimization problem

$$x^* = \operatorname{argmin}_x F_y(x) + R(x) \tag{2}$$

where F_y is a data fidelity term, e.g. $F_y(x) = \frac{1}{2}\|Ax - y\|_2^2$, and R is a regularization term encoding prior information about the solution. Common approaches to solve (2) are first-order proximal algorithms (Parikh & Boyd, 2014; Boyd et al., 2010) such as forward-backward splitting (FBS) given by

$$x_{k+1} = \operatorname{prox}_{\tau R}(x_k - \tau \nabla F_y(x_k)), \tag{3}$$

where $\tau > 0$ denotes the step size.

While choosing a data term F_y is typically straightforward, hand-crafted regularizers, such as the popular total variation (TV) regularization $R(x) = \|\nabla x\|_1$, often cannot perfectly capture the complex structure of the space of natural images. Plug-and-Play (PnP) methods (Heide et al., 2014; Venkatakrishnan et al., 2013) combine the advantages of deep learning and variational approaches, and, for instance, replace the prox operator of hand-crafted regularizers with a (learned) denoiser D, yielding

$$x_{k+1} = D(x_k - \tau \nabla F_y(x_k)). \tag{4}$$

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This substitution allows powerful pretrained denoisers to act as implicit image priors, enabling the same model to be applied across different inverse problems by changing only the forward operator A.

Early PnP methods rely on denoisers trained for additive Gaussian noise (Romano et al., 2017; Meinhardt et al., 2017; Rick Chang et al., 2017; Zhang et al., 2017; Hurault et al., 2021). More recent approaches instead leverage priors derived from generative models, such as GANs (Duff et al., 2024) and VAEs (Gonzalez et al., 2019). Recently, generative diffusion models (Daras et al., 2024; Zhu et al., 2023; Graikos et al., 2022) and flow models (Martin et al., 2025; Pourya et al., 2026; Ben-Hamu et al., 2024) have received attention in the PnP literature. Diffusion and flow models provide a time-dependent family of denoisers $\{D_t\}_{t\in[0,1]}$ adapted to different noise levels, making them directly applicable to PnP.

A key challenge in integrating diffusion and flow denoisers into iterative reconstruction algorithms is that these models are trained to operate on samples distributed according to a specific noise level distribution p_t. During a reconstruction process, however, the iterates generally deviate from this distribution. Recent flow- and diffusion-based PnP methods therefore incorporate explicit renoising steps that perturb the current iterate toward the appropriate noise level before applying the denoiser, and potentially averaging this process over multiple noise samples (e.g. Martin et al. (2025)). Despite their strong empirical performance, existing renoise-denoise PnP methods are typically formulated as stochastic algorithms and are difficult to analyze theoretically. In particular, the deterministic operator implicitly induced by averaging the denoiser output over the latent noise distribution has not been characterized, and as a consequence the convergence behavior of these methods remains poorly understood.

In this work, we identify the deterministic renoise-denoise operator underlying flow-based PnP methods and propose *FlowADMM*, a PnP algorithm built on this operator. We derive convergence guarantees under weak Lipschitz conditions on the underlying flow network, including for non-stationary timestep schedules, and demonstrate strong performance on several inverse problems.

Our contributions are summarized as follows:

- We identify and analyze the mean renoise-denoise operator underlying stochastic flow-based PnP methods.
- We propose FlowADMM, a PnP algorithm integrating the mean renoise-denoise operator into classical ADMM.
- We derive convergence criteria for FlowADMM with non-constant time schedules under weak Lipschitz assumptions on the underlying flow network.
- We propose a timestep-dependent Monte Carlo sampling strategy motivated by the coarse-to-fine behavior of the mean renoise-denoise operator.
- We demonstrate state-of-the-art performance on several inverse problems while requiring fewer data term evaluations than previous competitive stochastic averaging approaches.

2 Preliminaries

2.1 Flow Matching

Continuous generative flow models (Chen et al., 2018) learn a transformation between a simple base distribution p_0 (e.g. standard Gaussian) and a target data distribution p_1. Given an initial sample $x_0 \sim p_0$, a sample $x_1 \sim p_1$ is obtained by following the flow $\phi_t(x_0)$ defined by the ordinary differential equation (ODE)

$$\frac{\partial}{\partial t}\phi_t(x) = v_t^\theta(\phi_t(x)), \qquad \phi_0(x) = x \tag{5}$$

where v_t^θ is a time-dependent velocity field parameterized by a neural network. We refer to $t \in [0,1]$ as "time" following the flow matching literature, although it does not represent physical time but rather parametrizes the interpolation between the noise distribution and the data distribution.

Flow matching (Lipman et al., 2023) is a training technique that trains v_t^θ using regression to a target conditional velocity field given by a probability path between p_0 and p_1. A common choice, that we

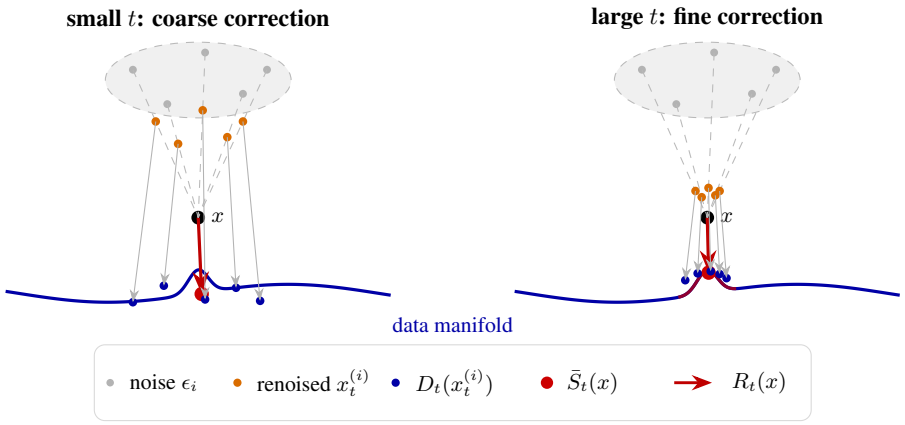

Figure 1: Effect of the time parameter t on the mean renoise-denoise operator. For small t, interpolated points remain closer to noise and naturally spread out farther, producing diverse trajectories that recover coarse structure but average fine details. For large t, interpolated points stay closer to x, producing trajectories that resolve the fine detail and place the mean operator on the local bump.

also assume throughout this paper, is linear interpolation, as e.g. used in rectified flow (Liu et al., 2023) and optimal transport (OT) flow (Tong et al., 2024), which defines a path conditioned on two endpoints (x_0 and x_1) by

$$x_t = tx_1 + (1-t)x_0 \tag{6}$$

which defines intermediate distributions p_t for $t \in [0,1]$, resulting in the flow matching objective

$$\min_\theta \mathbb{E}_{t\sim\mathcal{U}(0,1),x_0\sim p_0,x_1\sim p_1}(\|v_t^\theta(x_t) - (x_1 - x_0)\|_2^2). \tag{7}$$

Under this construction, the trained velocity network induces a family of MMSE denoisers

$$D_t(x_t) = \mathbb{E}[x_1 \mid x_t] = x_t + (1-t)v_t^\theta(x_t). \tag{8}$$

2.2 Flow-based plug-and-play reconstruction

Flow-based PnP methods apply the denoiser within an iterative reconstruction scheme while adapting the iterate to the corresponding noise level through an explicit renoising step. For example, PnP-Flow (Martin et al., 2025) performs the updates

$$\begin{aligned} z_{k+1} &= x_k - \tau_k \nabla F_y(x_k) \\ \tilde{z}_{k+1} &= t_k z_{k+1} + (1-t_k)\epsilon, \qquad \epsilon \sim p_0 = \mathcal{N}(0, I) \\ x_{k+1} &= D_{t_k}(\tilde{z}_{k+1}), \end{aligned} \tag{9}$$

where $(t_k)_k$ is a time-step schedule controlling the noise level. PnP-Flow additionally notes that the denoising step may be averaged over multiple noise realizations. However, this averaged mapping is introduced only as an algorithmic variant, and its intrinsic operator properties are not characterized. In particular, the existing convergence analysis is only carried out for the stochastic iteration and does not study the residual structure or Lipschitz behavior of the mean denoiser. In the next section, we make this averaged operator explicit and analyze it directly.

3 FlowADMM

3.1 Mean renoise-denoise operator

We now formalize the deterministic operator implicitly underlying sstochastic renoise-denoise plug-and-play methods. Assume that we have a pretrained flow velocity network $v^\theta : [0,1] \times \mathbb{R}^m \to \mathbb{R}^m$ defining a time-dependent MMSE denoiser

$$D_t = I + (1-t)v_t^\theta. \tag{10}$$

Existing flow-based PnP methods apply the denoiser after explicitly renoising the current iterate with a random noise sample. Averaging over the latent noise distribution naturally defines the following deterministic operator, which we call the *mean renoise-denoise operator*, as

$$\bar{S}_t(x) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[D_t(tx + (1-t)\epsilon)] \tag{11}$$

with timestep $t \in [0, 1]$. This operator represents the expected output of a renoise-denoise step and can be viewed as the deterministic limit of stochastic updates used by methods such as PnP-Flow (Martin et al., 2025) or Flower (Pourya et al., 2026). Figure 1 visualizes the action of the operator.

Our upcoming analysis involves the residual form of the mean renoise-denoise operator given by

$$R_t(x) = \bar{S}_t(x) - x \tag{12}$$

where R_t represents the average correction induced by the denoiser at noise level t. The residual thus defines a vector field, which can be interpreted as a local correction toward the data manifold.

3.2 Integration into ADMM

We integrate the mean renoise-denoise operator into ADMM. Compared to forward-backward splitting as used in (9), ADMM allows multiple prior operators through auxiliary variables, can be extended to non-differentiable data terms, and does not contain an operator-dependent step size restriction (Boyd et al., 2010). The classical ADMM algorithm aims to solve (2) by restating it as a constrained problem

$$\min_{x,z} F_y(x) + R(z) \quad \text{s.t.} \quad x = z, \tag{13}$$

which can be solved by alternating steps on x and z. Introducing the (scaled) dual variable u, the updates are defined as

$$x_{k+1} = \mathrm{prox}_{\tau F_y}(z_k - u_k) \tag{14}$$
$$z_{k+1} = \mathrm{prox}_{\tau R}(x_{k+1} + u_k) \tag{15}$$
$$u_{k+1} = u_k + x_{k+1} - z_{k+1}. \tag{16}$$

where τ is a penalty parameter and the proximal operator is defined as

$$\mathrm{prox}_f(v) = \mathrm{argmin}_x \frac{1}{2}\|x - v\|_2^2 + f(x). \tag{17}$$

In the spirit of previous plug-and-play methods (Ryu et al., 2019; Meinhardt et al., 2017), we propose to replace the proximal operator of the regularization function with the implicit prior given by the mean renoise-denoise operator, defining the algorithmic scheme, which we call FlowADMM

$$x_{k+1} = \mathrm{prox}_{\tau F_y}(z_k - u_k) \tag{18}$$
$$z_{k+1} = \bar{S}_{t_k}(x_{k+1} + u_k) \tag{19}$$
$$u_{k+1} = u_k + x_{k+1} - z_{k+1}, \tag{20}$$

where $(t_k)_k$ defines a time-schedule controlling the strength of the denoiser.

3.3 Convergence analysis

We now analyze the stability of the mean renoise-denoise operator, which will be used to develop convergence criteria for FlowADMM. Proofs in this section are given in Appendix A and are based on averaged operator theory.

Lemma 1. *Let the flow network v_t^θ be Lipschitz with constant $L_v(t)$. Then the residual operator $R_t = (\bar{S}_t - I)$ is Lipschitz continuous with constant $(1-t)(1+tL_v(t))$.*

This result shows that the strength of the operator decreases as $t \to 1$, where $\bar{S}_t$ approaches the identity. A convergence analysis of FlowADMM is delicate, since the mean renoise-denoise operators changes between iterations through the time schedule $(t_k)_k$. We start the convergence analysis of FlowADMM with a proposition that gives convergence guarantees for a constant time schedule $t_k = t$, that is based on results of Ryu et al. (2019).

Proposition 1. *Let $t \in (0,1)$, let the flow network v_t^θ be Lipschitz continuous with constant $L_v(t) < \frac{1}{1-t}$ and assume F_y is differentiable and μ-strongly convex. Let $\xi = (1-t)(1+tL_v(t))$ denote the Lipschitz constant of the residual operator R_t. Then for a data term penalty τ satisfying*

$$\tau > \frac{\xi}{(1+\xi-2\xi^2)\mu}, \tag{21}$$

fixed-t FlowADMM converges to a fixed point, if it exists.

Proposition 1 establishes conditions on the underlying flow network and the step size of the data term in the simplified setting of a fixed timestep schedule. The flow parameterization of the denoiser yields a timestep-dependent sufficient condition $L_v(t) < \frac{1}{1-t}$ on the learned velocity field that becomes less restrictive at late timesteps.

In practice, PnP methods based on flow and diffusion models achieve better results when using increasing time schedules (or, equivalently, decreasing noise schedules in the diffusion setting). For large t, the renoising distribution remains concentrated near the current iterate x. Consequently, the operator acts as a local refinement step only when x is already close to the data distribution. If x is far from the data manifold, large-t denoising remains off-distribution and does not necessarily move the iterate toward a plausible image. This motivates using smaller values of t in early iterations to obtain coarse global corrections, followed by larger values of t for refinements once the iterate is closer to the learned prior. Note that Prop. 1 already covers the convergence analysis for time schedules that become constant after finitely many iterations, i.e., $t_k = t_{\max}$ for all $k \geq K$. The following proposition is a generalization for non-constant time schedules.

Proposition 2. *Let $(t_k)_k$ be a sequence in $[0, t_{max}]$ with $\lim_{k\to\infty} t_k = t_{max} < 1$ and*

$$\sum_{k=1}^{\infty} |t_k - t_{max}| < \infty. \tag{22}$$

Assume that there exists a $\delta > 0$ such that the assumptions of Prop. 1 hold for every $\hat{t} \in [t_{max} - \delta, t_{max}]$, with the same penalty parameter τ. Let the underlying flow network additionally be Lipschitz in t, i.e. for some positive constant L_t

$$\|v_s^\theta(x) - v_t^\theta(x)\| \leq L_t|s-t| \qquad \forall x. \tag{23}$$

Then FlowADMM converges to a fixed point of the fixed-t_{max} iteration, if such a fixed point exists.

Prop. 2 extends the fixed-t convergence result to schedules whose deviations from $t_{\max}$ are summable. Notably, the Lipschitz restriction with respect to x on the flow network only applies in the neighborhood of $t_{\max}$, which is close to 1 in practice. There is no contraction bound restriction on the Lipschitz constant with respect to time L_t.

Although in this paper we do not explicitly enforce the Lipschitz condition on the flow networks, we give experimental evidence for the stability of FlowADMM by analyzing the Lipschitz behavior of pretrained networks during late-stage FlowADMM iterations in Appendix B.

The strong convexity assumption excludes rank-deficient data terms such as inpainting. Appendix 1 gives a fixed-t convergence condition for general convex data terms under an additional firm nonexpansiveness assumption on $\bar{S}_t$, which we do not verify for the pretrained networks.

3.4 Practical implementation

In practice, the expectation in the definition of the mean renoise-denoise operator $\bar{S}_t$ is intractable and must be approximated using Monte Carlo sampling. Given N i.i.d. samples $\{\epsilon_i\}_{i=1}^N$, $\epsilon_i \sim \mathcal{N}(0, I)$, we define the empirical operator as the unbiased estimator

$$\hat{S}_t^{(N)}(x) = \frac{1}{N}\sum_{i=1}^{N} D_t(tx + (1-t)\epsilon_i). \tag{24}$$

In FlowADMM, we replace $\bar{S}_{t_k}$ with its Monte Carlo counterpart $\hat{S}_{t_k}^{(N_k)}$, yielding a stochastic iteration. Our experiments show that increasing N improves reconstruction PSNR monotonically

(see Appendix D). This provides empirical motivation for studying the underlying limiting mean operator $\bar{S}_t$, whose finite-sample approximation needs to be used in practice.

We control the number of samples and thus the variance of the operator *per iteration* through the schedule N_k. Motivated by the coarse-to-fine behavior of $\bar{S}_t$ (see Figure 1), we employ a non-constant sampling strategy in which the number of samples increases in later iterations.

For small t_k, and when the current iterate is still far away from the manifold, the operator makes large, coarse corrections and is relatively insensitive to sampling noise, so a small number of samples suffices. In contrast, during late iteration refinements, an accurate estimation becomes more critical to refine small local structure and not kick the iterate off the manifold. Increasing N_k in later iterations therefore reduces variance when precision matters most. In our implementation, we therefore fix the total budget of flow evaluations and distribute it across iterations by increasing N_k in late iterations.

4 Related Work

Stochastic regularization. SNORE (Renaud et al., 2024) is closely related to our work in that it also addresses the distribution mismatch of applying denoisers to PnP iterates outside of their training noise levels. SNORE introduces an explicit stochastic regularization function based on Gaussian perturbations of the current iterate, leading to a stochastic gradient descent algorithm. It also introduces an annealed variant, in which the noise levels are gradually reduced.

Our work differs in three main aspects. First, SNORE is formulated as optimization of an explicit scalar regularizer, whereas our prior is defined directly through a deterministic mean renoise-denoise operator and does not, in general, correspond to the gradient of a scalar energy. Second, SNORE relies on additive Gaussian corruption with Gaussian denoisers, while our construction is tailored to the linear interpolation path used by straight-line flow matching. Third, SNORE analyzes the convergence of their stochastic gradient descent method to critical points under boundedness assumptions, whereas our analysis studies fixed-point convergence of a non-stationary ADMM operator under Lipschitz conditions on the underlying flow network.

Flow-based PnP methods. Recent plug-and-play methods increasingly leverage denoisers derived from generative flow matching models (Martin et al., 2025; Pourya et al., 2026; Ben-Hamu et al., 2024). PnP-Flow (Martin et al., 2025) introduces a renoise-denoise iteration within a forward-backward splitting framework and additionally observes that averaging over multiple noise realizations can improve stability. Their convergence analysis, however, relies on strong assumptions on the boundedness of iterates and vanishing data-term step-sizes. Flower (Pourya et al., 2026) further improves reconstruction quality by using proximal steps on the data term with adaptive step size. Their best-performing variant averages multiple complete stochastic reconstruction trajectories.

In contrast to these approaches, our work explicitly formalizes and analyzes the deterministic operator induced by averaging over the latent noise distribution and integrates it directly into an ADMM framework. Moreover, we derive convergence guarantees for non-constant timestep schedules with Lipschitz conditions directly on the underlying flow network.

Beyond plug-and-play approaches, D-Flow (Ben-Hamu et al., 2024) formulates inverse problems through optimization in the latent source space of the flow model by differentiating through the generation dynamics.

Convergence analysis of PnP methods. The convergence analysis of PnP algorithms in prior work commonly relies on contraction or averagedness properties of the denoising operator residual (Ryu et al., 2019; Hurault et al., 2022), or boundedness of the denoising operator itself Chan et al. (2016). While these provide strong guarantees, directly extending them to flow-based methods is challenging due to the stochastic renoise-denoise setting and the time-dependence of the denoiser. Our analysis builds on classical PnP theory while accounting for flow-based renoise-denoise operators and non-stationary timestep schedules. Moreover, we do not require the denoiser to be non-expansive.

Posterior sampling methods for inverse problems. A related line of work uses diffusion or flow models for approximate posterior sampling in inverse problems (Kawar et al., 2022; Chung et al., 2023; Kim et al., 2025; Erbach et al., 2026). These approaches aim to sample from the posterior

distribution conditioned on the measurements and are typically motivated from a Bayesian perspective. In contrast, plug-and-play methods are generally optimization-oriented and aim to compute a single high-quality reconstruction rather than sampling from the full posterior distribution.

5 NUMERICAL EXPERIMENTS

5.1 EXPERIMENTAL SETUP AND BASELINES

We evaluate the performance of FlowADMM on a range of inverse problems and compare it to recent state-of-the-art plug-and-play methods based on flow and diffusion models. We consider tasks including denoising, deblurring, super-resolution, random inpainting and box inpainting. Details on each task are provided in Appendix E.

Unless stated otherwise, we follow the experimental setup of Flower (Pourya et al., 2026) to ensure a fair comparison. Specifically, we use 100 images of size 128×128 from the CelebA dataset (Liu et al., 2015) and 100 images from the AFHQ-Cat dataset (Choi et al., 2020). For all experiments, we use the same pretrained flow models provided by (Martin et al., 2025), which are also used in the experiments in (Pourya et al., 2026).

In addition to PnP-Flow (Martin et al., 2025) and Flower (Pourya et al., 2026) we also compare against several other recent methods, including the flow-based approaches OT-ODE (Pokle et al., 2023), D-Flow (Ben-Hamu et al., 2024) and Flow-Priors (Zhang et al., 2024), the diffusion-based method DiffPIR (Zhu et al., 2023) and PnP-GS with a Gaussian gradient-step denoiser (Hurault et al., 2021). For these methods, we report the quantitative results provided by Pourya et al. (2026), which were obtained under the same experimental setup as our work.

5.2 COMPUTATIONAL BUDGET

To maintain a fair comparison, we match the total number of iterations K and flow evaluations to those used by Flower5-OT (Pourya et al., 2026), which is our main competitor. Flower5-OT averages 5 independent runs of the stochastic Flower algorithm, which leads to $5K$ flow evaluations. We want to stress that although Flower5-OT uses the same number of flow evaluations as FlowADMM, it actually requires $5\times$ more data-term proximal operator evaluations, because it averages full reconstructions, while FlowADMM only averages the prior step within each iteration. While flow evaluations often dominate the computational cost, the data consistency step can be non-negligible depending on the forward operator. Since the number of iterations is finite, we employ a power law time schedule between t_{min} and t_{max} defined as

$$t_k = t_{\text{min}} + \left(\frac{k+1}{K}\right)^{\gamma} (t_{\text{max}} - t_{\text{min}}), \tag{25}$$

where $\gamma < 1$ uses more late time steps and $\gamma > 1$ keeps the iterates near t_{min} for longer. For the Monte Carlo approximation of the mean renoise-denoise operator, we considered both a constant and a piecewise-constant three-stage sampling schedule, which allocates an increasing amount of samples to later iterations. We treat the positions and height of the steps as hyperparameters while keeping the total number of samples across all iterations fixed. This allows us to concentrate more computational resources on steps where higher accuracy is required.

5.3 HYPERPARAMETER TUNING

The hyperparameters of FlowADMM, including the data term weight τ and the step-based sampling schedule $(N_k)_k$, are tuned separately for each task on a validation set of 32 images per dataset, following the tuning protocol of Martin et al. (2025); Pourya et al. (2026). We observed that mainly the data term strength τ varies greatly between tasks, while the other parameters show a more consistent behavior. We also found that a late-heavy three-phase N_k schedule consistently outperforms the constant schedule overall, validating our sampling hypothesis. An ablation (see Appendix H.2) further shows that transferring the same late-heavy Monte Carlo sampling schedule idea to PnP-Flow improves all five CelebA tasks significantly, but still remains below FlowADMM, indicating that both the schedule and the ADMM formulation contribute to the final gains. Ablations and further details on the tuning and the final per-task parameter choices can be found in Appendix F.

5.4 RESULTS

Table 1: Results for different inverse problems on the CelebA test set.

Method	Denoising			Deblurring			Super-resolution			Random inpainting			Box inpainting		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Degraded	20.00	0.348	0.372	27.83	0.740	0.126	10.26	0.183	0.827	11.95	0.196	1.041	22.27	0.742	0.214
PnP-GS	32.64	0.910	0.035	34.03	0.924	0.041	31.31	0.892	0.064	29.22	0.875	0.070	–	–	–
DiffPIR	31.20	0.885	0.060	32.77	0.912	0.060	31.52	0.895	0.033	31.74	0.917	0.025	–	–	–
OT-ODE	30.54	0.859	0.032	33.01	0.921	0.029	31.46	0.907	**0.025**	28.68	0.871	0.051	29.40	0.920	0.038
D-Flow	26.04	0.607	0.092	31.25	0.854	0.038	30.47	0.843	0.026	33.67	0.943	**0.015**	30.70	0.899	0.026
Flow-Priors	29.34	0.768	0.134	31.54	0.858	0.056	28.35	0.713	0.102	32.88	0.871	0.019	30.07	0.858	0.048
PnP-Flow1	31.80	0.905	0.044	34.48	0.936	0.040	31.09	0.902	0.045	33.05	0.944	0.018	30.47	0.933	0.037
Flower1-OT	32.28	0.914	0.034	34.98	0.947	**0.026**	32.36	0.923	0.034	33.08	0.944	0.018	31.19	0.945	**0.022**
PnP-Flow5	32.30	0.911	0.056	34.80	0.940	0.047	31.49	0.906	0.056	33.98	0.953	0.022	31.09	0.940	0.043
Flower5-OT	33.14	**0.926**	0.038	**35.67**	**0.954**	0.032	33.09	0.932	0.040	33.95	0.953	0.020	**31.87**	**0.952**	0.023
FlowADMM (ours)	**33.19**	0.918	**0.027**	35.53	0.950	0.035	**33.94**	**0.940**	0.025	**35.23**	**0.965**	0.020	31.72	0.949	0.034

Table 2: Results for different inverse problems on the AFHQ-Cat test set.

Method	Denoising			Deblurring			Super-resolution			Random inpainting			Box inpainting		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Degraded	20.00	0.314	0.509	23.94	0.517	0.444	11.70	0.208	0.873	13.36	0.223	1.081	21.80	0.740	0.198
PnP-GS	32.58	**0.894**	**0.072**	27.91	0.753	0.349	24.15	0.632	0.362	29.42	0.836	0.126	–	–	–
DiffPIR	30.58	0.835	0.189	27.56	0.728	0.342	23.65	0.624	0.402	31.70	0.881	0.062	–	–	–
OT-ODE	30.03	0.815	0.076	27.06	0.713	**0.123**	25.91	0.716	**0.108**	29.40	0.839	0.090	24.62	0.875	0.085
D-Flow	26.13	0.574	0.175	27.82	0.721	0.164	24.64	0.601	0.190	32.20	0.894	0.040	26.26	0.842	0.077
Flow-Priors	29.41	0.763	0.153	26.47	0.700	0.181	23.51	0.570	0.272	32.37	0.906	0.047	26.20	0.818	0.118
PnP-Flow1	31.18	0.863	0.135	27.87	0.760	0.304	26.94	0.763	0.171	33.00	0.918	**0.037**	26.00	0.897	0.103
Flower1-OT	31.69	0.879	0.102	28.64	0.775	0.255	26.23	0.741	0.272	32.97	0.918	0.040	26.19	0.915	**0.063**
PnP-Flow5	31.43	0.864	0.168	28.19	0.766	0.332	27.37	0.774	0.183	33.75	0.929	0.048	26.68	0.901	0.120
Flower5-OT	32.35	0.891	0.116	**28.97**	0.784	0.283	26.57	0.750	0.282	33.70	0.927	0.045	26.88	**0.922**	0.066
FlowADMM (ours)	**32.90**	0.890	0.082	28.96	**0.792**	0.284	**28.12**	**0.808**	0.199	**34.82**	**0.938**	0.048	**27.39**	0.903	0.068

The quantitative evaluation in Table 1 and Table 2 shows that FlowADMM matches or improves the best competing methods in distortion metrics. In particular, FlowADMM obtains best PSNR in 7 out of 10 tasks and the best SSIM on 5 out of 10 tasks, with especially large gains for super-resolution and inpainting. FlowADMM performs worse on the LPIPS metric, which is consistent with the perception-distortion tradeoff (Blau & Michaeli, 2018), since we optimized for PSNR. Qualitative examples in Figure 2 further show that FlowADMM preserves fine structures, such as the cat whiskers in the super-resolution example. We give additional results in the Appendix.

Effectiveness of the mean-operator approximation On CelebA denoising and super-resolution, reconstructions improve monotonically as the constant Monte Carlo sample count increases from $N = 1$ to $N = 32$, with small variability across sampling seeds (see Appendix D). This supports the study of the underlying limiting mean renoise-denoise operator.

To assess the quality attainable with more accurate averaging, we additionally run FlowADMM for 100 iterations with a constant $N = 128$ samples per iteration. Our late-heavy schedule achieves a PSNR within 0.026 dB for denoising and 0.033 dB for super-resolution of this reference, while using only 500 rather than 12,800 flow evaluations. All other algorithm settings remained unchanged.

Timing and memory. We also compare wall-clock time and GPU memory on CelebA Gaussian deblurring, using the same protocol as in the PnP-Flow paper (100 test images, evaluated as 25 batches of 4 images each). The experiments were conducted on a machine with an i7-8700K CPU and a single RTX 4090 GPU. Table 3 reports the results for PnP-Flow5, Flower5-OT, and our 100-step FlowADMM configuration. We also evaluate FlowADMM with batched instead of sequential averaging, which substantially increases performance at the cost of higher peak memory. The speed-up compared to Flower5-OT is due to two factors: FlowADMM performs fewer data term evaluations, and in our implementation these proximal steps are computed in closed form using the structure of the forward operators, whereas Flower uses conjugate gradient iterations (see Appendix C). Additional comparisons using the same CG solver and a broader set of baselines are reported in Appendix G.

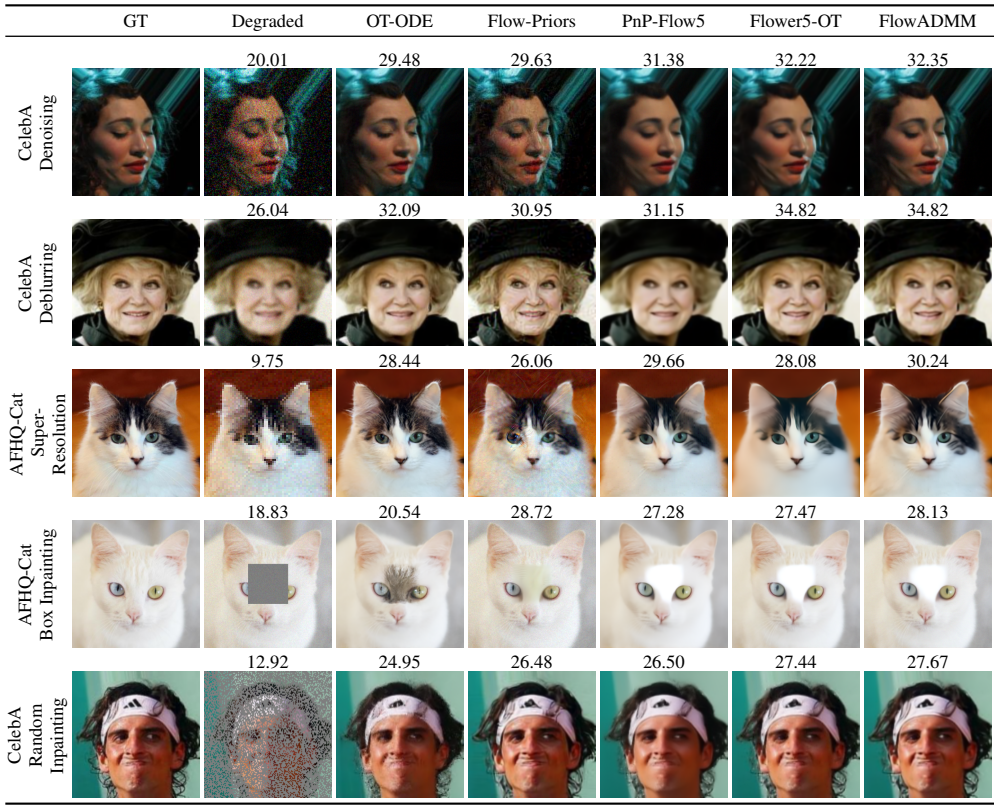

Figure 2: Example reconstructions for each task. We report the PSNR above each image.

Table 3: Timing and memory per image for deblurring on CelebA (128×128) for 100 iterations.

Method	Time / image (s)	Peak memory (GiB)
PnP-Flow5	2.84	0.30
Flower5-OT	5.10	0.30
FlowADMM (sequential averaging)	2.82	0.43
FlowADMM (batched averaging)	**1.51**	1.51

6 Conclusion and Limitations

We introduced FlowADMM, a PnP ADMM framework based on the mean renoise-denoise operator induced by flow matching models. By explicitly formalizing this operator, we derived convergence guarantees for both constant and non-constant timestep schedules under Lipschitz conditions on the underlying flow network. Empirically, FlowADMM matches or surpasses state-of-the-art reconstruction quality across a range of inverse problems. Our work analyzes flow-based PnP methods through an operator-theoretic lens and suggests several directions for future work, including stronger convergence guarantees and the explicit Lipschitz regularization of flow networks to improve stability.

Limitations. Our convergence analysis relies on the Lipschitz constant of the underlying flow network that is not explicitly penalized during training. Moreover, the convergence guarantees are limited to strongly convex data terms. In addition, our theoretical analysis focuses on the deterministic mean renoise-denoise operator, whereas the practical Monte Carlo approximation introduces additional stochasticity that is not fully captured by the current theory. Finally, although heuristic experiments show favorable Lipschitz conditions, the pretrained networks do not hard-enforce the required Lipschitz bounds.

AI USE STATEMENT

In this work, we used AI tools for assistance in coding, editing of the manuscript, online literature search, and assistance with the writing of mathematical proofs. We verified the references proposed by the LLM literature search manually, and all LLM-generated code was checked and tested. Some LaTeX tables in our work were formatted by LLMs. We verified the content of these tables against the raw data. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide code to reproduce the main experimental results and will make it publicly available upon acceptance. We provide details on the forward operators (Appendix E), hyperparameters and schedules, including the tuning process (Appendix F), and proofs of all theoretical results (Appendix A).

REFERENCES

Heinz H. Bauschke and Patrick L. Combettes. *Fejér Monotonicity and Fixed Point Iterations*, pp. 91–109. Springer International Publishing, Cham, 2017. ISBN 978-3-319-48311-5. doi: 10.1007/978-3-319-48311-5_5. URL https://doi.org/10.1007/978-3-319-48311-5_5.

Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. *arXiv preprint arXiv:2402.14017*, 2024.

Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6228–6237, 2018.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Machine Learning*, 3(1): 1–122, 2010.

Stanley H Chan, Xiran Wang, and Omar A Elgendy. Plug-and-play admm for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 3(1): 84–98, 2016.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

Giannis Daras, Hyungjin Chung, Chieh-Hsin Lai, Yuki Mitsufuji, Jong Chul Ye, Peyman Milanfar, Alexandros G Dimakis, and Mauricio Delbracio. A survey on diffusion models for inverse problems. *arXiv preprint arXiv:2410.00083*, 2024.

Margaret AG Duff, Neill DF Campbell, and Matthias J Ehrhardt. Regularising inverse problems with generative machine learning models. *Journal of Mathematical Imaging and Vision*, 66(1):37–56, 2024.

Julius Erbach, Dominik Narnhofer, Andreas Robert Dombos, Bernt Schiele, Jan Eric Lenssen, and Konrad Schindler. Solving inverse problems with FLAIR. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=w9xETx7HT1.

Mario Gonzalez, Andrés Almansa, Mauricio Delbracio, Pablo Musé, and Pauline Tan. Solving inverse problems by joint posterior maximization with a vae prior. *arXiv preprint arXiv:1911.06379*, 2019.

Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. *Advances in Neural Information Processing Systems*, 35:14715–14728, 2022.

Felix Heide, Markus Steinberger, Yun-Ta Tsai, Mushfiqur Rouf, Dawid Pająk, Dikpal Reddy, Orazio Gallo, Jing Liu, Wolfgang Heidrich, Karen Egiazarian, et al. Flexisp: A flexible camera image processing framework. *ACM Transactions on Graphics (ToG)*, 33(6):1–13, 2014.

Samuel Hurault, Arthur Leclaire, and Nicolas Papadakis. Gradient step denoiser for convergent plug-and-play. *arXiv preprint arXiv:2110.03220*, 2021.

Samuel Hurault, Arthur Leclaire, and Nicolas Papadakis. Proximal denoiser for convergent plug-and-play optimization with nonconvex regularization. In *International Conference on Machine Learning*, pp. 9483–9505. PMLR, 2022.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in neural information processing systems*, 35:23593–23606, 2022.

Jeongsol Kim, Bryan Sangwoo Kim, and Jong Chul Ye. Flowdps: Flow-driven posterior sampling for inverse problems. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12328–12337, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Ségolène Martin, Anne Gagneux, Paul Hagemann, and Gabriele Steidl. Pnp-flow: Plug-and-play image restoration with flow matching. In *International Conference on Learning Representations*, 2025.

Tim Meinhardt, Michael Moller, Caner Hazirbas, and Daniel Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In *Proceedings of the IEEE international conference on computer vision*, pp. 1781–1790, 2017.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3): 127–239, 2014.

Ashwini Pokle, Matthew J Muckley, Ricky TQ Chen, and Brian Karrer. Training-free linear image inverses via flows. *arXiv preprint arXiv:2310.04432*, 2023.

Mehrsa Pourya, Bassam El Rawas, and Michael Unser. Flower: A flow-matching solver for inverse problems. In *International Conference on Learning Representations*, 2026.

Marien Renaud, Jean Prost, Arthur Leclaire, and Nicolas Papadakis. Plug-and-play image restoration with stochastic denoising regularization. In *Forty-first International Conference on Machine Learning*, 2024.

JH Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them all–solving linear inverse problems using deep projection models. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5888–5897, 2017.

Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (red). *SIAM journal on imaging sciences*, 10(4):1804–1844, 2017.

Ernest Ryu, Jialin Liu, Sicheng Wang, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Plug-and-play methods provably converge with properly trained denoisers. In *International Conference on Machine Learning*, pp. 5546–5557. PMLR, 2019.

Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW.

Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE global conference on signal and information processing*, pp. 945–948. IEEE, 2013.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in neural information processing systems*, 31, 2018.

Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3929–3938, 2017.

Yasi Zhang, Peiyu Yu, Yaxuan Zhu, Yingshan Chang, Feng Gao, Ying N Wu, and Oscar Leong. Flow priors for linear inverse problems via iterative corrupted trajectory matching. *Advances in Neural Information Processing Systems*, 37:57389–57417, 2024.

Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van Gool. Denoising diffusion models for plug-and-play image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1219–1229, 2023.

A PROOFS

Lemma 2. *Let D_t be Lipschitz continuous with constant L. Then $\bar{S}_t$ is Lipschitz continuous with constant tL.*

Proof. Let $x, y \in \mathbb{R}^d$.

$$\bar{S}_t(x) - \bar{S}_t(y) = \mathbb{E}_\epsilon[D_t(tx - (1-t)\epsilon) - D_t(ty - (1-t)\epsilon)] \tag{26}$$

Taking the norm and applying Jensen's inequality yields

$$\begin{aligned}\|\bar{S}_t(x) - \bar{S}_t(y)\| &\leq \mathbb{E}_\epsilon[\|D_t(tx - (1-t)\epsilon) - D_t(ty - (1-t)\epsilon)\|] \\ &\leq \mathbb{E}_\epsilon[L\|tx - (1-t)\epsilon - (ty - (1-t)\epsilon)\|] \\ &= \mathbb{E}_\epsilon[L\|tx - ty\|] \\ &= tL\|x - y\|.\end{aligned}$$

□

Proof of Lemma 1. First, assume $(D_t - I)$ is Lipschitz with some constant L. Let $x, y \in \mathbb{R}^d$ and for brevity define $x_\epsilon = tx + (1-t)\epsilon$, $y_\epsilon = ty + (1-t)\epsilon$ which implies $x_\epsilon - y_\epsilon = t(x - y)$.

$$\begin{aligned}R_t(x) - R_t(y) &= (\bar{S}_t - I)(x) - (\bar{S}_t - I)(y) \\ &= \mathbb{E}_\epsilon[D_t(x_\epsilon) - D_t(y_\epsilon)] - (x - y) \\ &= \mathbb{E}_\epsilon[D_t(x_\epsilon) - D_t(y_\epsilon)] - (t(x - y) + (1-t)(x - y)) \\ &= \mathbb{E}_\epsilon[D_t(x_\epsilon) - D_t(y_\epsilon) - t(x - y)] - (1-t)(x - y) \\ &= \mathbb{E}_\epsilon[D_t(x_\epsilon) - D_t(y_\epsilon) - (x_\epsilon - y_\epsilon)] - (1-t)(x - y) \\ &= \mathbb{E}_\epsilon[(D_t - I)(x_\epsilon) - (D_t - I)(y_\epsilon)] - (1-t)(x - y).\end{aligned}$$

Taking the norm and applying Jensen's inequality and the triangle inequality yields

$$\begin{aligned}\|(\bar{S}_t - I)(x) - (\bar{S}_t - I)(y)\| &\leq \mathbb{E}_\epsilon[\|(D_t - I)(x_\epsilon) - (D_t - I)(y_\epsilon)\|] + (1-t)\|x - y\| \\ &\leq \mathbb{E}_\epsilon[L\|t(x - y)\|] + (1-t)\|x - y\| \\ &= tL\|x - y\| + (1-t)\|x - y\|.\end{aligned}$$

This means R_t is Lipschitz with constant $tL + (1-t)$. If the underlying flow network is L_v Lipschitz then $(D_t - I) = (1-t)v_t^\theta$ is $((1-t)L_v)$-Lipschitz. Setting $L = (1-t)L_v$ in the proof above proves the lemma. □

In the following, we analyze FlowADMM through its equivalent plug-and-play Douglas-Rachford splitting (DRS) iteration (Ryu et al., 2019). Let $P = \text{prox}_{\tau F_y}$ and $w_k = x_{k+1} + u_k$. The FlowADMM updates imply

$$w_{k+1} = T_{t_k}(w_k) \tag{27}$$

with

$$T_t = \frac{1}{2}I + \frac{1}{2}(2P - I)(2\bar{S}_t - I). \tag{28}$$

Throughout the appendix, T_t denotes this DRS operator. The original FlowADMM states are recovered using the identities above.

An operator $T : \mathbb{R}^m \to \mathbb{R}^m$ is called α-averaged if

$$T = (1-\alpha)I + \alpha N \tag{29}$$

for some $\alpha \in (0, 1)$ and some nonexpansive operator N. An averaged operator is nonexpansive, and its fixed-point iteration converges if its fixed-point set is nonempty (Bauschke & Combettes, 2017).

We can now give a proof of FlowADMM convergence with fixed t:

Proof of Proposition 1. From Lemma 1 it follows that $(\bar{S}_t - I)$ is $\xi := ((1-t)(1+tL_v))$-Lipschitz. If $L_v < \frac{1}{1-t}$ then $\xi < 1$. Since L_v is a Lipschitz constant it is also greater or equal to 0 which means $\xi \geq (1-t)$. This means that Corollary 3 in Ryu et al. (2019) is applicable, which proves the convergence of the algorithm under the conditions stated in the proposition. Moreover, the proof in Ryu et al. (2019) shows that the underlying DRS opeator T_t is averaged. □

We just showed that under certain conditions T_t is averaged. Assume that T_t has a fixed point. Then the fixed point iteration

$$w_{k+1} = T_t(w_k) \tag{30}$$

converges. In the following, we will generalize this result to varying t-schedules. We start with a more general proposition about general sequences of operators that eventually become averaged.

Proposition 3. *Let $(t_k)_k$ be a sequence in $[0, t_{max}]$ with $\lim_{k\to\infty} t_k = t_{max} < 1$ and*

$$\sum_{k=1}^{\infty} |t_k - t_{max}| < \infty. \tag{31}$$

Let $T_{t_k} : \mathbb{R}^d \to \mathbb{R}^d$ be a family of operators indexed by t_k. Let $T_* = T_{t_{max}}$ be averaged and

$$\|T_s(w) - T_t(w)\| \leq C|s - t|. \tag{32}$$

Then the iteration

$$w_{k+1} = T_{t_k}(w_k) \tag{33}$$

converges to a fixed point of T_*.

Proof. Denote

$$e_k = T_{t_k}(w_k) - T_*(w_k). \tag{34}$$

We have

$$\|e_k\| = \|T_{t_k}(w_k) - T_*(w_k)\| \leq C|t_k - t_{\max}| \tag{35}$$

and thus

$$\sum_{k=1}^{\infty} \|e_k\| < \infty \tag{36}$$

from the assumptions. Since T_* is averaged, we can write

$$T_* = (1-\lambda)I + \lambda N \tag{37}$$

with some non-expansive operator N. Thus,

$$T_{t_k}(w_k) = T_*(w_k) + e_k \tag{38}$$
$$= w_k + \lambda(N(w_k) - w_k) + e_k \tag{39}$$
$$= w_k + \lambda(N(w_k) + \frac{e_k}{\lambda} - w_k). \tag{40}$$

This form allows us to directly apply Proposition 5.34 of Bauschke & Combettes (2017). With $(\lambda_k)_k = \lambda$ we get

$$\sum_{k=1}^{\infty} \lambda_k \|\frac{e_k}{\lambda}\| = \sum_{k=1}^{\infty} \|e_k\| < \infty. \tag{41}$$

The convergence to a fixed point now directly follows from Proposition 5.34 in Bauschke & Combettes (2017) and the fact that T_* has the same fixed points as N. □

These results can finally be used to show the convergence of FlowADMM with varying t.

Proof of Proposition 2. From the assumptions of Proposition 1 $v_t^\theta(x)$ is already Lipschitz in x. To apply Proposition 3 we need to show that $T_t(w)$ is Lipschitz in t. Together with the new condition on the Lipschitzness in t on the flow network we have

$$\|v_s^\theta(x) - v_t^\theta(y)\| \leq L_x\|x - y\| + L_t|s - t|. \tag{42}$$

Clearly the denoiser $D_t(x) = x + (1 - t)v_t^\theta(x)$ is also Lipschitz in x and t, with constants C_x and C_t. For the mean denoising operator with $x_\alpha = \alpha x + (1 - \alpha)\epsilon$ we have

$$\|\bar{S}_t(x) - \bar{S}_s(x)\| = \|\mathbb{E}[D_t(x_t)] - \mathbb{E}[D_s(x_s)]\| \tag{43}$$
$$= \|\mathbb{E}[D_t(x_t) - D_t(x_s)] + \mathbb{E}[D_t(x_s) - D_s(x_s)]\| \tag{44}$$
$$\leq \mathbb{E}[\|D_t(x_t) - D_t(x_s)\|] + \mathbb{E}[\|D_t(x_s) - D_s(x_s)\|] \tag{45}$$
$$\leq C_x\mathbb{E}[\|x_t - x_s\|] + C_t|s - t|. \tag{46}$$

Since $x_t - x_s = (t - s)(x - \epsilon)$ we get

$$\mathbb{E}[\|x_t - x_s\|] = |t - s|\,\mathbb{E}[\|(x - \epsilon)\|]. \tag{47}$$

From the assumption that T_{t_k} becomes averaged in a δ-neighborhood of $t_{\max}$, it follows that it becomes averaged after a finite number of steps and the sequence of iterates is therefore bounded. Therefore, $\mathbb{E}[\|(x - \epsilon)\|]$ is also bounded, so

$$\mathbb{E}[\|x_t - x_s\|] \leq C_\epsilon|t - s| \tag{48}$$

and in total we have

$$\|\bar{S}_t(x) - \bar{S}_s(x)\| \leq (C_xC_\epsilon + C_t)|s - t|. \tag{49}$$

Since the only ADMM step that depends on t is the application of $\bar{S}_t$, the T_* operator inherits the Lipschitzness from $\bar{S}_t$. The rest follows immediately from Proposition 3. □

The following standard result of averaged operator theory (Bauschke & Combettes, 2017) gives a sufficient condition for convergence without strong convexity.

Remark 1 (Non-strongly convex data terms)**.** *Let F_y be proper, closed, and convex, let t be fixed and $\tau > 0$. If $\bar{S}_t$ is firmly nonexpansive, i.e.*

$$\|\bar{S}_t(x) - \bar{S}_t(y)\|^2 \leq \langle \bar{S}_t(x) - \bar{S}_t(y), x - y\rangle \qquad \forall x, y, \tag{50}$$

then exact-mean FlowADMM converges whenever its DRS operator has a fixed point. This assumption on $\bar{S}_t$ is not implied by the residual Lipschitz condition in Lemma 1.

Proof. The proximal operator $P = \mathrm{prox}_{\tau F_y}$ is firmly non-expansive. This means $2P - I$ and $2\bar{S}_t - I$ are non-expansive and their composition is non-expansive too. Thus

$$T_t = \frac{1}{2}I + \frac{1}{2}(2P - I)(2\bar{S}_t - I) \tag{51}$$

is $\frac{1}{2}$-averaged, which means the underlying FlowADMM iteration converges to a fixed point (if it exists). □

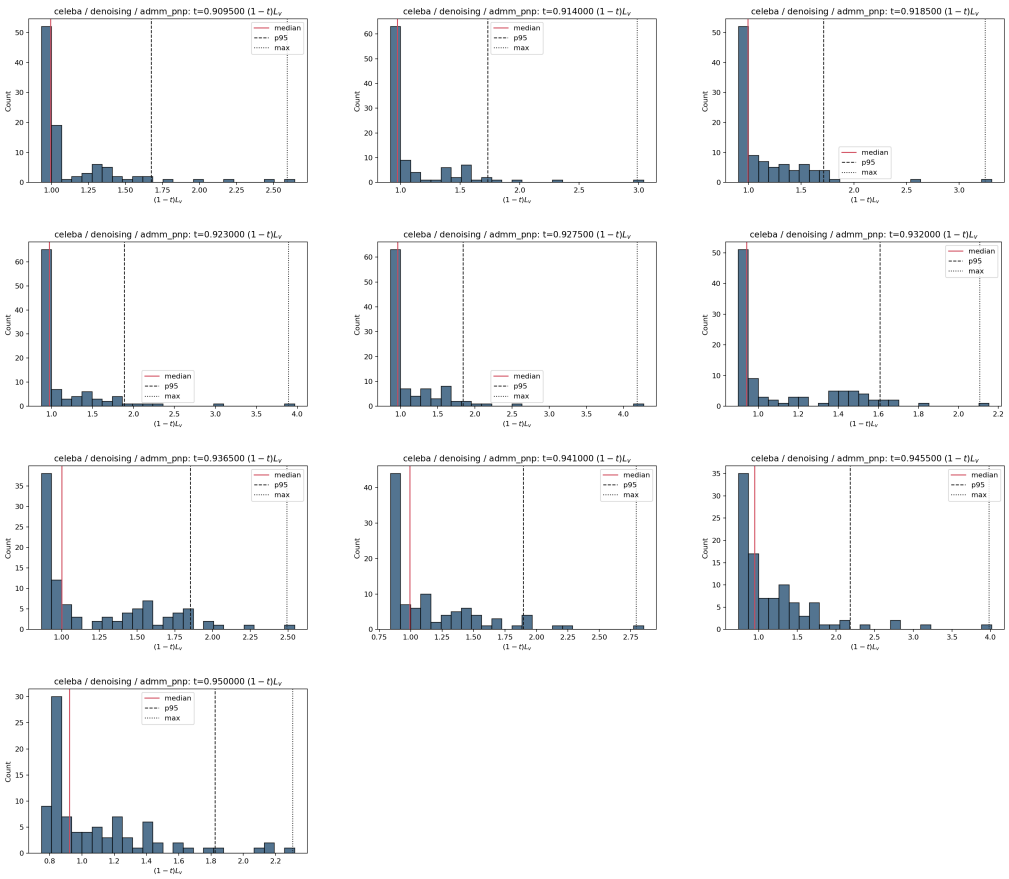

Figure 3: Distribution of the Jacobian norm Lipschitz estimates along the late-stage FlowADMM trajectory on the denoising task.

B Experiments on the Lipschitz condition

To better understand the stability of the mean denoiser in FlowADMM, we estimate the Lipschitz constants of the underlying flow network by analyzing the distribution of Jacobian spectral norms along the late-stage trajectory of FlowADMM for a denoising task

$$\hat{L}_v(x,t) \approx \|J_x v_t^\theta(x)\|_2. \tag{52}$$

We estimated this quantity at denoiser inputs $\tilde{z}_k = (t(x_{k+1} + u_k) + (1-t)\epsilon)$ on the last 10 of the 100 stages (corresponding to $t > 0.9$) of FlowADMM evaluated on CelebA. We estimate the Jacobian norms by matrix free power iterations on $(J_x v_t(\tilde{z}_k))^\top J_x v_t(\tilde{z}_k)$, implemented with forward and reverse mode autodiff in PyTorch, which efficiently compute jacobian-vector-products (jvp) and vector-jacobian-products (vjp). Specifically, we iterate

$$\tilde{w}_{k+1} = \frac{Jw_k}{\|Jw_k\|} \quad \text{(using jvp)}, \tag{53}$$

$$w_{k+1} = \frac{J^\top \tilde{w}_{k+1}}{\|J^\top \tilde{w}_{k+1}\|} \quad \text{(using vjp)}. \tag{54}$$

We plot the distributions of the $(1-t)\hat{L}_v(\tilde{z}_k, t)$ estimates in histograms for each evaluated timestep in Figure 3. This quantity corresponds to the Lipschitz condition in Proposition 1, where values smaller than 1 suggest stability. The histograms in Figure 3 shows that the median of $(1-t)\hat{L}_v$ is below 1. This suggests that FlowADMM trajectories largely stay in stable regions. Nevertheless, the distributions also show long tails with clear spikes above 1, which shows that convergence using

Table 4: Estimated velocity-network Jacobian spectral norms at 100 random inputs per timestep. The final column counts estimates below $1/(1-t)$.

t	$1/(1-t)$	Median	95th percentile	Maximum	Below threshold
0.900	10.00	5.641	6.878	10.207	99/100
0.925	13.33	4.946	6.065	6.727	100/100
0.950	20.00	4.252	5.944	10.609	100/100

Proposition 1 cannot be guaranteed without enforcing a hard Lipschitz constraint on the underlying flow network. Note, that the experiments in this section are only heuristics and we do not claim to compute the global Lipschitz constant of the flow network. Finding a tight upper bound for the global Lipschitz constant of a neural network is in general hard to do Virmaux & Scaman (2018).

B.1 Additional random-input Jacobian measurements

To complement the trajectory-based measurements, we evaluate the velocity network at 100 i.i.d. sampled inputs in $[-1,1]^{3\times128\times128}$ for each $t \in \{0.9, 0.925, 0.95\}$. We use the same power-iteration procedure as above to estimate $\hat{L}_v(x,t) \approx \|J_x v_t^\theta(x)\|_2$.

Table 4 compares these estimates with the threshold $1/(1-t)$ appearing in Proposition 1. The estimated norms fall below the threshold for 99 of 100 inputs at $t = 0.9$ and all sampled inputs for $t = 0.925$ and 0.95. Substituting the estimates into

$$\hat{\xi}(x,t) = (1-t)(1+t\hat{L}_v(x,t)) \tag{55}$$

gives median ξ-estimates of 0.608, 0.418, and 0.252, respectively. These empirical measurements indicate favorable local behavior of the sampled inputs and are evidence for the stability of FlowADMM. We want to stress that these local estimates do not prove a global Lipschitz bound and are not a proof for Proposition 1. Provably enforcing the condition during training remains open for future work.

C Computing proximal operators

In this work, we consider quadratic data fidelity terms of the form

$$F_y(x) = \frac{1}{2}\|Ax - y\|_2^2, \tag{56}$$

whose proximal operator is given in closed form by

$$\operatorname{prox}_{\tau F_y}(v) = (I + \tau A^\top A)^{-1}(v + \tau A^\top y). \tag{57}$$

For large-scale problems, explicitly forming or inverting the matrix $(I+\tau A^\top A)$ is typically infeasible, and matrix-free iterative solvers are often employed.

However, many linear operators in imaging applications have a decomposition

$$A = Q^\top \Lambda P, \tag{58}$$

where P and Q are orthogonal transforms and Λ is diagonal. This includes, for example, convolution operators (diagonalized by the discrete Fourier transform), masking operators (diagonal in the pixel domain), and subsampling operators.

Using this decomposition, we obtain

$$A^\top A = P^\top \Lambda^\top \Lambda P, \qquad A^\top y = P^\top \Lambda^\top Q y, \tag{59}$$

which allows rewriting the proximal operator in the transformed domain (left multiplying with P) as

$$P \operatorname{prox}_{\tau F_y}(v) = (I + \tau \Lambda^\top \Lambda)^{-1} \left(Pv + \tau \Lambda^\top Q y\right). \tag{60}$$

Defining $z = P \operatorname{prox}_{\tau F_y}(v)$, the problem reduces into element-wise equations that can be solved by

$$z_i = \frac{(Pv)_i + \tau \lambda_i (Qy)_i}{1 + \tau \lambda_i^2}, \tag{61}$$

Table 5: Effect of a constant Monte Carlo sample count on CelebA. PSNR is reported in dB as mean ± standard deviation over Monte Carlo seeds. Relative operator discrepancies use independent $N = 128$ reference estimates. SR denotes super-resolution.

	PSNR ↑		Relative discrepancy ↓	
N	Denoising	SR	Denoising	SR
1	31.954 ± 0.0048	32.586 ± 0.0100	0.0683	0.1381
2	32.582 ± 0.0075	33.296 ± 0.0120	0.0481	0.0955
4	32.904 ± 0.0044	33.648 ± 0.0079	0.0340	0.0681
8	33.067 ± 0.0024	33.822 ± 0.0077	0.0243	0.0491
16	33.151 ± 0.0017	33.918 ± 0.0040	0.0176	0.0354
32	33.192 ± 0.0012	33.963 ± 0.0024	0.0131	0.0265

Table 6: Comparison with a constant-$N = 128$ reference on 100 CelebA images. PSNR differences are reference minus proposed schedule.

	Constant $N = 128$	Proposed N_k	Difference
Denoising PSNR (dB)	33.224	33.198	0.026
Super-resolution PSNR (dB)	33.996	33.963	0.033
Flow evaluations	12,800	500	—

where λ_i denotes the i-th diagonal entry of Λ. The solution is then obtained via the inverse transform

$$\mathrm{prox}_{\tau F_y}(v) = P^\top z. \tag{62}$$

We use this formulation wherever possible and fall back to a CG solver where such a decomposition is not available.

D Effect of Monte Carlo sampling

Our convergence analysis concerns the exact mean operator $\bar{S}_t$, whereas the implementation uses a finite-sample estimator $\hat{S}_t^{(N)}$. We investigate the influence of this approximation empirically on CelebA denoising and super resolution using 100 images, five Monte Carlo seeds, and constant sample counts $N \in \{1, 2, 4, 8, 16, 32\}$.

At iterations 1, 10, 25, 50, and 100, we compare the operator estimate with an independently sampled reference using $N_{\text{ref}} = 128$ samples. For an evaluated operator input v, the relative discrepancy is

$$e_N(v, t) = \frac{\|\hat{S}_t^{(N)}(v) - \hat{S}_t^{(128)}(v)\|}{\|\hat{S}_t^{(128)}(v)\|}. \tag{63}$$

Table 5 reports final PSNR and relative operator discrepancies. Increasing N consistently improves PSNR and reduces the operator discrepancy, with diminishing reconstruction gains at larger sample counts. The reported variability across Monte Carlo seeds generally decreases as N increases. The monotonic quality increase suggests that the analysis of the limiting operator $\bar{S}_t$ is meaningful and useful, even if practical implementations need to approximate it.

We additionally compare the proposed varying-N_k schedule with the constant-$N = 128$ reference reconstruction on all 100 CelebA test images in Table 6. The constant schedule uses $12,800$ flow evaluations, whereas the paper schedule uses only 500 flow evaluations. The proposed schedule therefore reaches almost the same quality with only $1/25$ of the flow evaluations.

E Task Details and Forward Operators

In the main paper, we evaluated on the same task settings as Martin et al. (2025) and Pourya et al. (2026). In all cases, the measurement model is

$$y = Ax + \epsilon, \tag{64}$$

where A is the task-specific forward operator and ϵ is additive Gaussian noise with standard deviation σ_{noise}. The implementation of the forward operators and their transposes was taken without change from the official PnP-Flow code Martin et al. (2025), which will be detailed in the following paragraphs.

Denoising. For denoising, the forward operator is the identity, $A = I$. Thus the measurement is simply a noisy version of the clean image. We use Gaussian noise with standard deviation $\sigma_{\text{noise}} = 0.2$.

Gaussian deblurring. For deblurring, A is convolution with a Gaussian blur kernel implemented in the Fourier domain (with zero padding). The transposed operator $A^\top$ is implemented as convolution with the conjugate filter in the FFT representation. We use a kernel size of 61×61, with blur standard deviation $\sigma_{\text{blur}} = 1.0$ on CelebA and $\sigma_{\text{blur}} = 3.0$ on AFHQ-Cat, followed by additive Gaussian noise with $\sigma_{\text{noise}} = 0.05$.

Super-resolution. For super-resolution, A is a strided subsampling operator that keeps one pixel every s pixels in each spatial direction. Its adjoint $A^\top$ is the corresponding zero-filled upsampling operator. We use scale factor $s = 2$ on CelebA and $s = 4$ on AFHQ-Cat, with additive Gaussian noise $\sigma_{\text{noise}} = 0.05$.

Box inpainting. For box inpainting, A is a binary masking operator that removes a centered square region, and $A^\top = A$ since the mask is diagonal. The mask half-size is 20 pixels on CelebA and 40 pixels on AFHQ-Cat, corresponding to a missing square of side length 40 and 80, respectively. We add Gaussian noise with $\sigma_{\text{noise}} = 0.05$.

Random inpainting. For random inpainting, A is a Bernoulli masking operator with independent probability of missing pixels of $p = 0.7$, so that on average 70% of pixels are removed. Again, the operator is symmetric, i.e. $A^\top = A$. The random mask is fixed across runs using a fixed random seed. We add Gaussian noise with $\sigma_{\text{noise}} = 0.01$ on the observed pixels.

F HYPERPARAMETER TUNING DETAILS

Algorithm 1 FlowADMM with Monte Carlo Sampling

Require: Measurements y, forward operator A, pretrained flow denoiser D_t, iterations K, data term step size τ, time schedule $(t_k)_k$, sample schedule $(N_k)_k$

Initialize $z_0 = A^\top y$, $x_0 = A^\top y$, $u_0 = 0$
for $k = 0, \dots, K-1$ **do**
  $x_{k+1} \leftarrow \text{prox}_{\tau F_y}(z_k - u_k)$
  $z_{k+1} \leftarrow 0$
  for $i = 1, \dots, N_k$ **do**
    Sample $\epsilon_i \sim \mathcal{N}(0, I)$
    $z_{k+1} \leftarrow z_{k+1} + \frac{1}{N_k} D_{t_k}(t_k(x_{k+1} + u_k) + (1 - t_k)\epsilon_i)$
  end for
  $u_{k+1} \leftarrow u_k + x_{k+1} - z_{k+1}$
end for
return z_K

We explored a 3-phase schedule and a constant schedule for N_k. The three phase schedule is defined as

$$N_k = \begin{cases} N_e, & \frac{k}{K} < s_1, \\ N_m, & s_1 \le \frac{k}{K} < s_2, \\ N_l & \frac{k}{K} \ge s_2, \end{cases} \tag{65}$$

We denote a three phase schedule with the above parameters as $3\text{ph}(N_e, N_m, N_l; s_1, s_2)$ and "const" means $N_k = 5$ for all k. The sweep over the three-phase schedule parameters was done in such a way that the total number of N_k stays constant. This ensures that FlowADMM uses the same number

Table 7: Hyperparameters used for the CelebA experiments. For baselines, we report the tuned parameters computed in Pourya et al. (2026)

Method	Hyperparameter	Denoising	Deblurring	Super-resolution	Random inpainting	Box inpainting
OT-ODE	t_0	0.3	0.4	0.1	0.1	0.1
	γ	td	td	const	const	td
D-Flow	λ	0.001	0.001	0.001	0.001	0.001
	α	0.1	0.1	0.1	0.1	0.1
	n_{iter}	3	7	10	20	9
Flow-Priors	λ	100	1000	10000	10000	10000
	η	0.01	0.01	0.1	0.01	0.01
PnP-Flow5	K	100	100	100	100	100
	N_{avg}	5	5	5	5	5
	α	0.8	0.01	0.3	0.01	0.5
Flower5-OT	K	100	100	100	100	100
	N_{avg}	5	5	5	5	5
FlowADMM	K	100	100	100	100	100
	τ	5.0	0.5	0.5	0.25	1.0
	$t_{\min}$	0.5	0.5	0.3	0.3	0.1
	$t_{\max}$	0.95	0.95	0.95	0.95	0.95
	γ	1.0	0.5	1.0	0.5	2.0
	N_k	$\text{3ph}(1, 1, 41; 0.5, 0.9)$	$\text{3ph}(1, 1, 41; 0.5, 0.9)$	$\text{3ph}(1, 3, 35; 0.6, 0.9)$	$\text{3ph}(1, 4, 29; 0.5, 0.9)$	$\text{3ph}(1, 4, 35; 0.7, 0.9)$

Table 8: Hyperparameters used for the AFHQ-Cat experiments. For baselines, we report the tuned parameters computed in Pourya et al. (2026)

Method	Hyperparameter	Denoising	Deblurring	Super-resolution	Random inpainting	Box inpainting
OT-ODE	t_0	0.3	0.3	0.1	0.1	0.1
	γ	td	td	const	const	td
Flow-Priors	λ	100	1000	10000	10000	10000
	η	0.01	0.01	0.1	0.01	0.01
PnP-Flow5	K	100	500	500	200	100
	N_{avg}	5	5	5	5	5
	α	0.8	0.01	0.01	0.01	0.5
Flower5-OT	K	100	100	500	200	100
	N_{avg}	5	5	5	5	5
FlowADMM	K	100	100	500	200	100
	τ	5.0	0.25	0.25	0.125	0.5
	$t_{\min}$	0.5	0.5	0.3	0.3	0.1
	$t_{\max}$	0.95	0.95	0.95	0.95	0.9
	γ	1.0	0.5	1.0	0.5	2.0
	N_k	$\text{3ph}(1, 1, 41; 0.5, 0.9)$	const	$\text{3ph}(1, 4, 29; 0.5, 0.9)$	$\text{3ph}(1, 3, 33; 0.5, 0.9)$	$\text{3ph}(1, 3, 19; 0.6, 0.8)$

of flow evaluations as Flower5-OT and PnP-Flow5. We want to mention that we mainly explored 3-phase over 2-phase or other schedules to be more easily able to satisfy that constraint, and mainly explored schedules with low N_m, i.e. the mid-phase was used to "fill up" the remaining sample budget. In real world settings more complicated schedules are possible and might even lead to better performance. We did a grid search with the following values

- $\tau \in \{0.125, 0.25, 0.5, 1.0, 2.0, 5.0, 10.0\}$
- $t_{\text{min}} \in \{0.1, 0.2, 0.3, 0.5\}$
- $t_{\text{max}} \in \{0.9, 0.95\}$
- $\gamma \in \{0.5, 1.0, 2.0\}$
- $s_1 \in \{0.5, 0.6, 0.7\}$
- $s_2 \in \{0.8, 0.9\}$
- $N_e = 1$ (fixed early phase)
- $N_m \in \{1, 3, 4\}$,
- N_l was chosen to fix the flow evaluation budget constraint of matching Flower5-OT's and PnP-Flow5's flow evalutions, i.e. $K \cdot N_{\text{avg}}$ flow evaluations in total

Note, that while all CelebA experiments use $K = 100$ iterations, some experiments on AFHQ-Cat use more, namely super-resolution with 500 and random inpainting with 200 iterations.

Table 9: Runtime in seconds per image using the same CG solver and 500 flow evaluations.

Task	FlowADMM (sequential)	Flower	FlowADMM (batched)
Denoising	2.845	2.896	1.516
Deblurring	2.940	4.819	1.646
Super-resolution	2.885	3.045	1.436

Table 10: Data-fidelity workload in the matched-CG deblurring experiment.

Measurement	FlowADMM	Flower
Proximal steps	100	500
Total CG iterations	1,537	22,405
Data-step time (s/image)	0.1425	1.9680

G Additional experiments on computational cost

G.1 Comparison using the same CG solver

To compare computational cost with a common proximal implementation, we rerun FlowADMM and Flower using the same conjugate-gradient (CG) sovler for the data fidelity proximal step and a budget of 500 flow evaluations. Although the benchmark operators admit closed-form proximal operator evaluations, this experiment investigates the cost when both methods must use an iterative solver (which would be the case for more complex operators such as MRI, non-uniform blur, etc.).

Table 9 reports runtimes. Sequential FlowADMM and Flower have similar runtimes for denoising and super-resolution. For deblurring, sequential FlowADMM takes 2.940 s/image vs. 4.819 s/image for Flower, which corresponds to a $1.64\times$ speedup. The workload measurements in Table 10 show fewer proximal evaluations and CG iterations for FlowADMM. The iteration counts also depend on the linear systems encountered along the trajectory of each method.

Batching the Monte Carlo sammples further reduces runtime at the expense of peak memory (see Table 11).

G.2 Comparison with additional baselines

We additionally evaluate runtimes for PnP-GS, OT-ODE, Flow-Priors, and FlowADMM on a common CelebA subset, using a batch size of four. D-Flow exceeded tthe available GPU memory under the common batch size four setting and is therefore omitted from this timing comparison. Tables 12 and 13 show runtime, peak memory, and reconstruction quality.

FlowADMM does not have a universal runtime advantage. PnP-GS is faster on all three tasks, and OT-ODE is faster than batched FlowADMM for denoising and deblurring. However, FlowADMM achieves the highest PSNR among these methods on each task and uses substantially less peak memory. Batched evaluation improves its runtime while increasing memory slightly.

H Ablations

H.1 Influence of hyperparameters

We evaluate the sensitivity of our hyperparameters and whether they can be shared between tasks. Table 14 shows that the main gains come from late-heavy three-phase averaging rather than from fine-tuning the time schedule. Relative to the final per-task tuned method, replacing the task-specific schedule by the shared three-phase schedule $3\mathrm{ph}(1, 3, 18; 0.5, 0.8)$ costs only 0.0288 dB on average, whereas reverting to constant averaging loses 0.2751 dB, showing that the principled late-heavy schedule leads to significant performance gains. We also find that sharing $t_{\min} = 0.3$ or $\gamma = 1.0$ incurs modest average drops of 0.0419 dB and 0.0617 dB, respectively, with the largest degradation concentrated on box inpainting. These results suggest that most hyperparameters except τ can be unified between tasks, incuring only modest performance penalties.

Table 11: Peak-memory ranges across the matched-CG tasks.

Method	Peak memory
FlowADMM (sequential)	446–684 MiB
FlowADMM (batched)	1.56–1.80 GiB
Flower	320–576 MiB

Table 12: Runtime and peak memory on a common CelebA subset. Runtimes are in seconds per image; memory is in GiB.

Method	Denoising	Deblurring	Super-resolution	Peak memory
PnP-GS	0.018	0.569	0.345	6.35
OT-ODE	1.268	1.095	1.947	2.33
Flow-Priors	15.267	15.044	15.274	8.81
FlowADMM (sequential)	2.769	2.887	2.830	0.44
FlowADMM (batched)	1.499	1.522	1.391	1.52

H.2 Applying the late-heavy sampling schedule to PnP-Flow

We further investigate the influence of the late-heavy N_k schedule by also applying it to PnP-Flow. We tune the schedule together with other PnP-Flow hyperparameters for each task on CelebA. Table 15 shows that late-heavy sampling schedule improves PnP-Flow on all five CelebA tasks, with the largest gains on super-resolution and box inpainting. However, FlowADMM still remains better on every task, with the remaining gap ranging from 0.1609 dB on box inpainting to 1.1730 dB on random inpainting. This suggests that the N_k schedule is inherently advantageous for renoise-denoise methods, while also giving evidence that ADMM has an advantage over forward-backward splitting methods for flow-based PnP methods.

I Additional results

I.1 Trajectory visualizations

Figures 4, 5, 6 and 7 visualize the FlowADMM trajectory through the primal data-consistency iterate x, the prior iterate z, and the dual variable u. In the first iterations, x and z can differ a lot. x reacts directly to the measurement model through the proximal update, whereas z stays closer to the flow prior. As the iterations proceed, the gap between the two iterates shrinks and both converge to the same reconstruction, indicating that the consensus constraint is being satisfied. This highlights an advantage of the ADMM over proximal gradient descent: data consistency and prior regularization are handled by separate, interpretable states rather than being entangled in a single update.

This separation is especially useful in the early phase of the reconstruction, where the observation model and the learned prior may disagree strongly. FlowADMM allows each component to make large task-specific corrections before forcing consensus, which is not directly visible in single-state proximal gradient descent trajectories. Note that we initialize $x = z = A^\top y$.

Toward the end of the iteration the dual variable u becomes 0. This can be explained with the following. Near the end of the algorithm $\bar{S}_t(x) \approx x$. This happens because t gets closer to 1 (i.e. the prior becomes weaker). This means the z and u updates become

$$z_{k+1} = x_{k+1} + u_k \tag{66}$$

$$u_{k+1} = u_k + x_{k+1} - (x_{k+1} + u_k) = 0, \tag{67}$$

so u naturally vanishes in late iterations.

I.2 Additional tasks

We additionally consider the severely ill-posed problem of $8\times$ super-resolution on the CelebA dataset, which is not part of the standard benchmark used in prior work, which only evaluated $2\times$ super-resolution on that dataset. We tuned FlowADMM and PnP-Flow on the validation set with the same

Table 13: Mean PSNR in dB on the common CelebA subset used for the additional runtime comparison.

Method	Denoising	Deblurring	Super-resolution
PnP-GS	32.159	33.290	29.258
OT-ODE	30.322	32.848	31.264
Flow-Priors	29.248	31.326	28.352
FlowADMM	**33.041**	**35.405**	**33.685**

Table 14: Ablation of FlowADMM on the CelebA validation set for the five inverse problems. We report mean PSNR over the five tasks. The top row is the final method used in the paper, and all subsequent rows are measured relative to it.

Setting	Mean PSNR ↑	Δ vs final method	Max single-task drop
Final method	**34.1815**	-	-
Shared $3\mathrm{ph}(1, 3, 18; 0.5, 0.8)$	34.1527	-0.0288	-0.0484
Constant $N_k = 5$	33.9064	-0.2751	-0.3499
Shared $t_{\min} = 0.3$	34.1396	-0.0419	-0.1508
Shared $\gamma = 1.0$	34.1198	-0.0617	-0.2594

procedure as usual. Flower does not expose hyperparameters for tuning, so we use it as is. As always, we keep the total number of flow iterations the same between methods. The tuned hyperparameters for these experiments are shown in Table 16

The results are shown in Table 17. In this challenging setting, FlowADMM drastically outperforms the two competitors.

We also tested the generality of our method on radial mask compressed sensing MRI (similar to the supplementary material of Flower) and motion deblurring. We did not tune hyperparameters for these tasks. The results can be found in Figures 8 and 9.

Table 15: CelebA test-set per-task PSNR comparison between the standard PnP-Flow5 baseline, using the late-heavy N_k schedule in PnP-Flow (PnP-Flow-KAvg), and FlowADMM. Positive deltas indicate improvements over the corresponding reference method.

Task	PnP-Flow5	PnP-Flow-KAvg	FlowADMM	Δ vs PnP-Flow5	Δ vs FlowADMM
Denoising	32.3119	32.3972	33.2049	+0.0853	-0.8077
Deblurring	34.8035	34.8616	35.6008	+0.0581	-0.7392
Super-resolution	31.5030	33.3271	34.0372	+1.8241	-0.7101
Box inpainting	31.0589	31.5672	31.7281	+0.5083	-0.1609
Random inpainting	34.0700	34.2278	35.4008	+0.1578	-1.1730

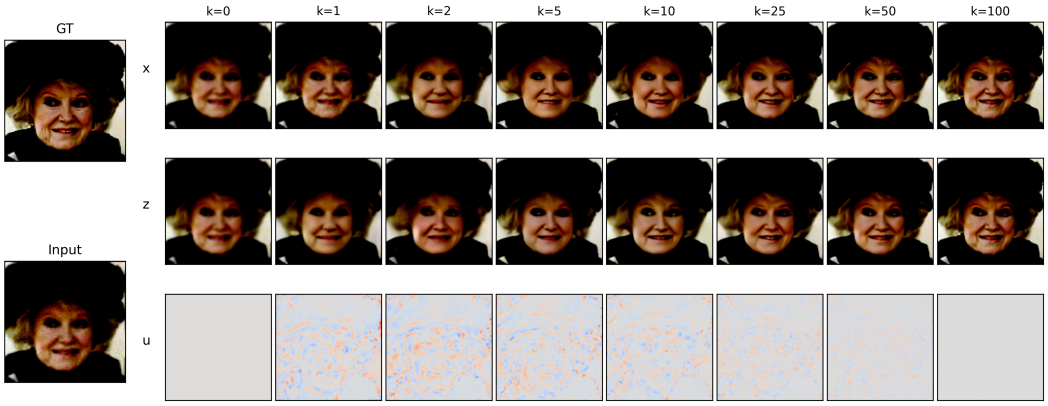

Figure 4: Trajectory visualization of a deblurring experiment

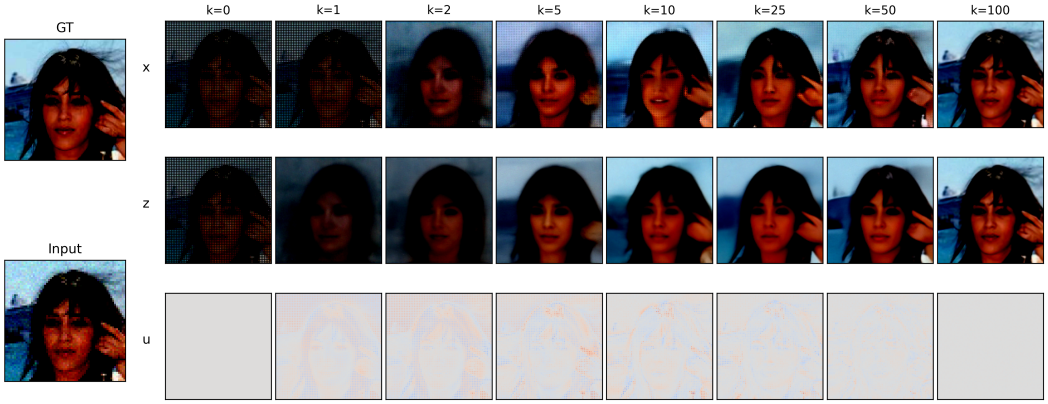

Figure 5: Trajectory visualization of a super-resolution experiment

Table 16: Hyperparameters for the CelebA $8\times$ super-resolution test.

Method	Hyperparameters
Flower5-OT	$(K, N_{\text{avg}}) = (100, 5)$
PnP-Flow5	$(K, N_{\text{avg}}, \text{lr}, \gamma\text{-style}, \alpha) = (100, 5, 2.0, \text{const}, 0.001)$
FlowADMM	$(K, \tau, t_{\min}, t_{\max}, \gamma, N_k)$ $= (100, 0.1, 0.2, 0.95, 1.0, 3\text{ph}(1, 3, 35; 0.6, 0.9))$

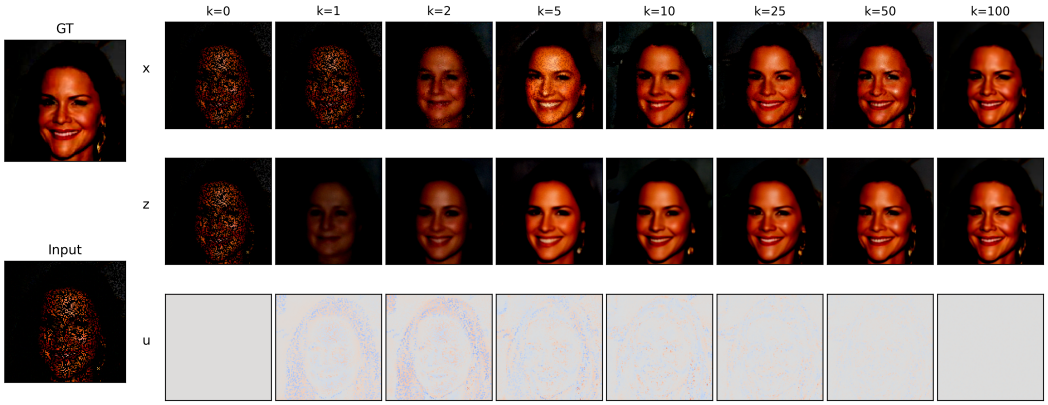

Figure 6: Trajectory visualization of a random inpainting experiment

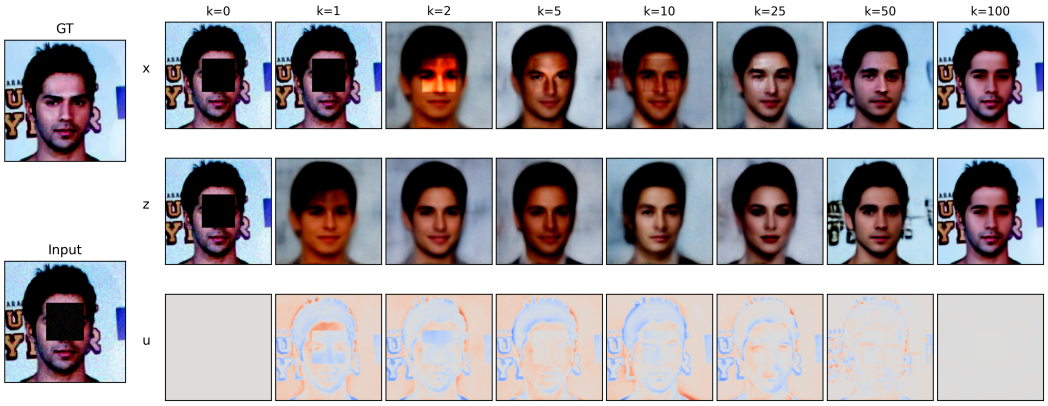

Figure 7: Trajectory visualization of a box inpainting experiment.

Table 17: CelebA 8× super-resolution on 100 test images. We compare against our main flow-based competitors

Method	PSNR ↑	SSIM ↑	LPIPS ↓
Flower5-OT	14.21	0.374	0.383
PnP-Flow5	16.32	0.418	0.297
FlowADMM	**21.28**	**0.667**	**0.184**

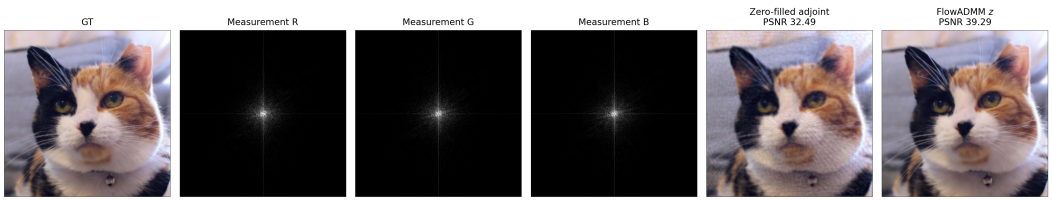

Figure 8: Results for CS MRI

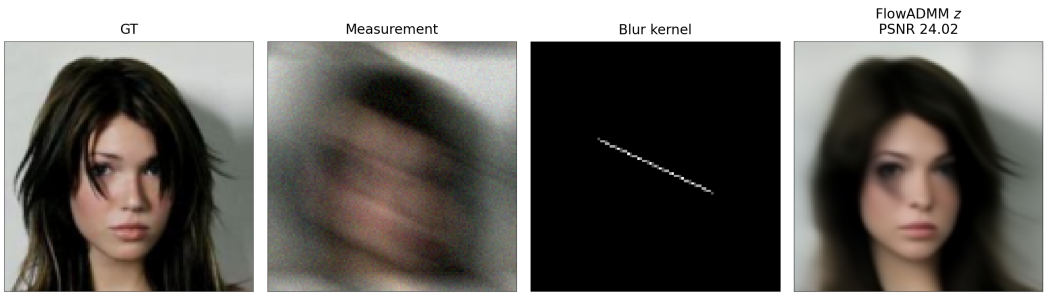

Figure 9: Results for motion deblurring