# Deep Expectation Consistent Approximation Methods for Inverse Problems

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Saurav Kumaraswami Shastri, B.E.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2025

Dissertation Committee:

Prof. Philip Schniter, Advisor

Prof. Emre Ertin

Prof. Rizwan Ahmad

# Abstract

To solve linear inverse problems, plug-and-play (PnP) methods replace the proximal step in a convex optimization algorithm with a call to an application-specific denoiser, often implemented using a deep neural network (DNN). Although such methods yield accurate solutions, they can be improved. For example, denoisers are usually designed/trained to remove white Gaussian noise, but the denoiser input error in PnP algorithms is usually far from white or Gaussian. Approximate message passing (AMP) methods provide white and Gaussian denoiser input error, but only when the forward operator is sufficiently random. In the first work of the dissertation, for Fourier operator based linear inverse problem, we propose a PnP algorithm based on generalized expectation-consistent (GEC) approximation—a close cousin of AMP—that offers predictable error statistics at each iteration, as well as a new DNN denoiser that leverages those statistics. We apply our approach to magnetic resonance (MR) image recovery and demonstrate its advantages over existing PnP and AMP methods.

In the second work of the dissertation, we expand our focus to address both linear and non-linear inverse problems within the generalized linear model (GLM) framework. We propose a novel variant of the expectation-consistent (EC) approximation that iteratively leverages DNNs to solve GLM inverse problems. Unlike traditional EC implementations, our proposed framework does not require random forward operators. As a case study, we focus on a popular non-linear inverse problem of phase

retrieval, which involves accurately recovering images from noisy phaseless measurements. In addition to applying EC in a non-traditional manner, we also propose a novel stochastic damping scheme that is inspired by recent diffusion methods. Like existing phase-retrieval methods based on PnP priors, regularization by denoising, or diffusion, our approach iterates a denoising stage with a measurement-exploitation stage. But unlike existing methods, our approach requires far fewer denoiser calls. We demonstrate the efficacy and robustness of our approach for noisy phase-retrieval of colored images on oversampled-Fourier and coded-diffraction-pattern measurements and find improvements in both PSNR and SSIM with 5x fewer denoiser calls.

Given the ill-posed nature of noisy inverse problems, multiple plausible solutions can exist for a single set of measurements. In the Bayesian framework, these solutions are typically obtained by sampling from the posterior distribution. Unlike our initial works, which focused on obtaining point estimates—specifically, the posterior mean—the third and final part of this dissertation delves into the rapidly advancing field of diffusion models. To this end, we address the challenge of posterior sampling from first principles and introduce a novel framework called Recursive Annealed Posterior Sampling (RAPS). To enhance the computational efficiency and robustness of the algorithm, we introduce several improvements, including adaptive regularization and a stochastic extension. Additionally, by employing expectation-propagation, we expand the algorithm's applicability to a broader array of generalized-linear inverse problems. Our proposed method exhibits exceptional performance over a wide range of inverse problems and neural-function evaluation budgets.

To Amma, Appa, and Akka—my unwavering foundation and guiding lights.

# Acknowledgments

Completing a Ph.D. has been a deeply enriching and transformative journey—one that would not have been possible without the guidance and support of many wonderful people. I take this opportunity to express my heartfelt gratitude to everyone who played a role in this chapter of my life.

First, I am deeply grateful to my advisor, Prof. Philip Schniter, for his exceptional guidance, constant encouragement, and insightful discussions throughout my Ph.D. His depth of knowledge, clarity of thought, and intellectual rigor have left a lasting impact on the way I approach research. I feel very fortunate to have had the opportunity to learn from someone as brilliant and generous with his time and ideas.

I also thank my dissertation committee members—Prof. Rizwan Ahmad and Prof. Emre Ertin—for their valuable feedback and thought-provoking questions. I am also grateful to both Prof. Ahmad and Prof. Chris Metzler for their collaboration on parts of the work included in this dissertation. I also appreciate Prof. Lee Potter for serving on my candidacy committee, and all the other professors at The Ohio State University whose courses and teaching helped lay the foundation for my research.

This work was supported in part by the National Institutes of Health under Grant R01-HL135489 and Grant R01-EB029957, and in part by the National Science Foundation under Grant CCF-1955587. I am grateful for this support, which made much of this research possible.

A big thanks to Dr. Hassan Mansour and Dr. Yanting Ma for making my internship at Mitsubishi Electric Research Laboratories (MERL) both enjoyable and a great learning experience in industrial research. I am also grateful to the friends I made at MERL for making the summer of 2023 memorable.

I have had some amazing labmates during my time at Ohio State. Thank you to Jeffrey, Srijith, Matt, Subrata, Ted, Michael, and Xuan for all the interesting research discussions, coding, debugging help, and for just being there to chat and hang out. A special thanks to Jeffrey and Patricia for their kindness, the volleyball games that added much-needed balance to my life, and for including me in their get-togethers during football games and holidays.

Outside of academia, I was incredibly lucky to be surrounded by friends who brought warmth, laughter, and adventure into my life. Shrikanth, thank you for always being there—for the encouragement, the life advice, and the countless Zoom work sessions, hikes, runs, and city explorations that kept me going. Shashwat, your cheerful nature and love for food made the tough days easier. I also enjoyed our many conversations about life—you always helped me think things through and come up with a plan when I needed it.

Kirtan, Ananya, Ayesha, Pashmeen, and Surojit—thank you for being my first circle of friends in Columbus. From weekend outings and board game nights to navigating Ph.D. stress, you were a constant source of joy and support. Krutant, thanks for being a great roommate, classmate, and friend. Your street-smart guidance helped me adjust to life in the U.S., and I really enjoyed all the time we spent together. Ravleen, thank you for your kindness and amazing food—hanging out with you and Shashwat was always a pleasure. Ritvik, I will always admire your incredible knowledge

of world events and random facts, and I really enjoyed our conversations that ranged from serious topics to silly ones.

Thanks to my high school friends—Parikshit, Sooraj, and Krishna—for introducing me to fitness and running. Staying active really helped me during tough times, especially during COVID. A special thanks to Parikshit for helping me realize the importance of veganism and inspiring me to make more mindful lifestyle choices.

Thanks also to my other friends who have been part of this journey: Sunil, Vignesh, Prajwal, Rama, Sizhuo, Vishikh, Anurag, Meghana, Arjun, Thejas, Vijaylakshmi, Neel, Nihanth, Tejas, Kamya, Krishna, Purav, Shriya, and Ujash. Each of you made life a little lighter and a lot more fun.

Lastly, and most importantly, I want to thank my family. To my mother, father, and sister—thank you for your constant support, love, and encouragement. I wouldn't be here without you. To my nephew, thank you for bringing immense joy and laughter during the final stretch of my Ph.D. I am also thankful to my grandmothers, brother-in-law, and all my aunts, uncles, and cousins for their encouragement and blessings over the years.

Thank you all for being a part of this journey and for making it meaningful.

# Vita

| | |
|---|---|
| August 2014 - May 2018 ..... | B.E. Electrical and Electronics Engineering,<br>PES Institute of Technology, Bengaluru, India |
| June 2018 - July 2019 ..... | Project Assistant,<br>Dept. of Electrical Engineering,<br>Indian Institute of Science, Bengaluru, India |
| August 2019 - December 2024 ..... | Graduate Research Associate,<br>Dept. of Electrical and Computer Engineering,<br>The Ohio State University, Columbus, USA |
| May 2023 - August 2023 ..... | Research Intern,<br>Mitsubishi Electric Research Laboratories,<br>Cambridge, USA |
| August 2024 - present ..... | Graduate Teaching Associate,<br>Dept. of Electrical and Computer Engineering,<br>The Ohio State University, Columbus, USA |

# Publications

S. K. Shastri and P. Schniter, "Fast and Robust Phase Retrieval via Deep Expectation-Consistent Approximation," in *IEEE Trans. Comput. Imag.*, vol. 11, pp. 116-128, 2025.

S. K. Shastri, Y. Ma, P. Boufounos and H. Mansour, "Deep Calibration and Operator Learning for Ground Penetrating Radar Imaging," in *Proc. European Signal Process. Conf.*, pp. 1796-1800, 2024.

S. K. Shastri, R. Ahmad and P. Schniter, "Deep expectation-consistent approximation for phase retrieval," in *Proc. Asilomar Conf. Signals Syst. Comput.*, pp. 910-914, 2023.

S. K. Shastri, R. Ahmad, C. A. Metzler and P. Schniter, "Denoising generalized expectation-consistent approximation for MR image recovery," in *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 3, pp. 528–542, Sept. 2022.

S. K. Shastri, R. Ahmad, C. A. Metzler, and P. Schniter, "Expectation consistent plug-and-play for MRI," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 8667–8671, 2022.

S. K. Shastri, R. Ahmad and P. Schniter, "Autotuning plug-and-play algorithms for MRI," in *Proc. Asilomar Conf. Signals Syst. Comput.*, pp. 1400-1404, 2020.

## Fields of Study

Major Field: Electrical and Computer Engineering

# Table of Contents

**Page**

# List of Tables

**Table** **Page**

# List of Figures

# Chapter 1: Introduction

## 1.1 Linear Inverse Problem

In linear inverse problem, we aim to recover a signal $\boldsymbol{x}_{\mathsf{true}} \in \mathbb{C}^N$ from measurements $\boldsymbol{y} \in \mathbb{C}^P$ of the form

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_{\mathsf{true}} + \boldsymbol{w}, \tag{1.1}$$

where $\boldsymbol{A}$ is a known linear operator and $\boldsymbol{w}$ is unknown noise. Well-known examples of linear inverse problems include deblurring [1]; super-resolution [2, 3]; inpainting [4]; image recovery in magnetic resonance imaging (MRI) [5]; computed tomography [6]; holography [7]; and decoding in communications [8]. Importantly, when $\boldsymbol{A}$ is not full column rank (e.g., when $P < N$), the measurements $\boldsymbol{y}$ can be explained well by many different hypotheses of $\boldsymbol{x}_{\mathsf{true}}$. In such cases, it is essential to harness prior knowledge of $\boldsymbol{x}_{\mathsf{true}}$ when solving the inverse problem.

### 1.1.1 Prior Art

The traditional approach [9] to recovering $\boldsymbol{x}_{\mathsf{true}}$ from $\boldsymbol{y}$ in (1.1) is to solve an optimization problem like

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \left\{ g_1(\boldsymbol{x}) + g_2(\boldsymbol{x}) \right\}, \tag{1.2}$$

where $g_1(\boldsymbol{x})$ promotes measurement fidelity and the regularization $g_2(\boldsymbol{x})$ encourages consistency with the prior information about $\boldsymbol{x}_{\mathsf{true}}$. For example, if $\boldsymbol{w}$ is white Gaussian noise (WGN) with precision (i.e., inverse variance) $\gamma_w$, then $g_1(\boldsymbol{x}) = \frac{\gamma_w}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ is an appropriate choice. Choosing a good regularizer $g_2$ is much more difficult. A common choice is to construct $g_2$ so that $\boldsymbol{x}_{\mathsf{true}}$ is sparse in some transform domain, i.e., $g_2(\boldsymbol{x}) = \lambda\|\boldsymbol{\Psi}\boldsymbol{x}\|_1$ for $\lambda > 0$ and a suitable linear operator $\boldsymbol{\Psi}$. A famous example of this choice is total variation regularization [10] and in particular its anisotropic variant (e.g., [11]). However, the intricacies of many real-world signal classes (e.g., natural images) are not well captured by sparse models like these. Even so, these traditional methods provide useful building blocks for contemporary methods, as we describe below. We will discuss the algorithmic aspects of solving (1.2) in Section 2.1.

More recently, there has been a focus on training deep neural networks (DNNs) for image recovery given a sufficiently large set of examples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$ to train those networks. These DNN-based approaches come in many forms, including dealiasing approaches [12, 13], which use a convolutional DNN to recover $\boldsymbol{x}_{\mathsf{true}}$ from $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{y}$ or $\boldsymbol{A}^{+}\boldsymbol{y}$, where $(\cdot)^{+}$ denotes the pseudo-inverse; unrolled approaches [14,15], which unroll the iterations of an optimization algorithm into a neural network and then learn the network parameters that yield the best result after a fixed number of iterations; and inverse GAN approaches [16, 17], which first use a generative adversarial network (GAN) formulation to train a DNN to turn random code vectors $\boldsymbol{z}$ into realistic signal samples $\boldsymbol{x}$, and then search for the specific $\boldsymbol{z}$ that yields the $\widehat{\boldsymbol{x}}$ for which $\|\boldsymbol{A}\widehat{\boldsymbol{x}} - \boldsymbol{y}\|$ is minimal. Good overviews of these methods can be found in [18–20]. Although the aforementioned DNN-based methods have shown promise, they require large training datasets, which may be unavailable in some applications. Also, models trained under

particular assumptions about $\boldsymbol{A}$ and/or statistics of $\boldsymbol{w}$ may not generalize well to test scenarios with different $\boldsymbol{A}$ and/or $\boldsymbol{w}$.

So-called "plug-and-play" (PnP) approaches [21] give a middle-ground between traditional algorithmic approaches and the DNN-based approaches discussed above. In PnP, a DNN is first trained as a signal denoiser, and later that denoiser is used to replace the proximal step in an iterative optimization algorithm (see Section 2.1.2). One advantage of this approach is that the denoiser can be trained with relatively few examples of $\{\boldsymbol{x}_i\}$ (e.g., using only signal patches rather than the full signal) and no examples of $\{\boldsymbol{y}_i\}$. Also, because the denoiser is trained on signal examples alone, PnP methods have no trouble generalizing to an arbitrary $\boldsymbol{A}$ and/or $\boldsymbol{w}$ at test time. The regularization-by-denoising (RED) [22, 23] framework yields a related class of algorithms with similar properties. See [24] for a comprehensive overview of PnP and RED.

### 1.1.2 Summary of Contribution 1: Denoising GEC for MRI Recovery

With a well-designed DNN denoiser, PnP and RED significantly outperform sparsity-based approaches, as well as end-to-end DNNs in limited-data and mismatched-$\boldsymbol{A}$ scenarios (see, e.g., [24]). However, there is room for improvement. For example, while the denoisers used in PnP and RED are typically trained to remove the effects of additive WGN (AWGN), PnP and RED algorithms yield estimation errors that are not white nor Gaussian at each iteration. As a result, AWGN-trained denoisers will be mismatched at every iteration, thus requiring more iterations and compromising performance at the fixed point. Although recent work [25] has shown that deep equilibrium methods can be used to train the denoiser at the algorithm's fixed point,

the denoiser may still remain mismatched for the many iterations that it takes to reach that fixed point, and the final design will be dependent on the $\boldsymbol{A}$ and noise statistics used during training.

These shortcomings of PnP algorithms motivate the following two questions:

1. Is it possible to construct a PnP-style algorithm that presents the denoiser with predictable error statistics at every iteration?

2. Is it possible to construct a DNN denoiser that can efficiently leverage those error statistics?

When $\boldsymbol{A}$ is a large unitarily invariant random matrix, the answers are well-known to be "yes": approximate message passing (AMP) algorithms [26] yield AWGN errors at each iteration with a known variance, which facilitates the use of WGN-trained DNN denoisers like DnCNN [27] (see Section 2.1.2 for more on AMP algorithms). In many inverse problems, however, $\boldsymbol{A}$ is either non-random or drawn from a distribution under which AMP algorithms do not behave as intended. So, the above two questions still stand.

In Chapter 2, we answer both of the above questions in the affirmative for Fourier-based $\boldsymbol{A}$. Using the framework of generalized expectation-consistent (GEC) approximation [28] in the wavelet domain [29], we propose a PnP algorithm that yields an AWGN error in each wavelet subband, with a predictable variance, at each iteration. We then propose a new DNN denoiser design that can exploit knowledge of the wavelet-domain error spectrum. For recovery of MR images from the fastMRI [30] and Stanford 2D FSE [31] datasets, we present experimental results that show the advantages of our proposed approach over existing PnP and AMP-based approaches.

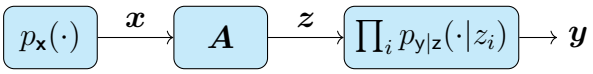


Figure 1.1: Generalized linear model relating signal $\boldsymbol{x}$ to measurements $\boldsymbol{y}$.

## 1.2 Phase Retrieval as an instance of Generalized Linear Model

We next expand our focus to Generalized Linear Model (GLM) framework [32]. A wide variety of linear and nonlinear inverse problems fall under the GLM framework, which models the relationship between the measurements $\boldsymbol{y} \in \mathcal{Y}^m$ and the signal/image hypothesis $\boldsymbol{x} \in \mathbb{R}^d$ or $\mathbb{C}^d$ using

$$p_{\mathsf{y}|\mathsf{x}}(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{m} p_{\mathsf{y}|\mathsf{z}}(y_i|z_i) \text{ for } \boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}, \tag{1.3}$$

for scalar "measurement channel" $p_{\mathsf{y}|\mathsf{z}}$ and forward transform $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ or $\mathbb{C}^{m\times d}$ (see Fig. 1.1). By choosing appropriate $p_{\mathsf{y}|\mathsf{z}}$, one can model, e.g., additive noise of an arbitrary distribution, logistic regression [33], Poisson regression [34], dequantization [35], and phase retrieval [36, 37].

A simple instance of the GLM is the Standard Linear Model (SLM) (1.1), where $p_{\mathsf{y}|\mathsf{z}}(y_i|z_i) = \mathcal{N}(y_i; z_i, v)$ and so

$$\boldsymbol{y} = \boldsymbol{z}_{\mathsf{true}} + \boldsymbol{w} = \boldsymbol{A}\boldsymbol{x}_{\mathsf{true}} + \boldsymbol{w}, \quad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, v\boldsymbol{I}). \tag{1.4}$$

Here, $v$ is the variance of the additive white Gaussian noise (AWGN) vector $\boldsymbol{w}$. As previously discussed, through appropriate choice of $\boldsymbol{A}$, the SLM covers, e.g., denoising, deblurring, compressed sensing, super-resolution, inpainting, magnetic resonance imaging, and computed tomography.

In Chapter 3, we tackle the phase retrieval (PR) problem. In PR one aims to recover an unknown signal or image $\boldsymbol{x}_{\mathsf{true}} \in \mathbb{R}^d$ from phaseless measurements

$$\boldsymbol{y} = |\boldsymbol{A}\boldsymbol{x}_{\mathsf{true}}| + \boldsymbol{w}, \tag{1.5}$$

where $\boldsymbol{A} \in \mathbb{C}^{m \times d}$ is a known linear operator, $|\cdot|$ is an element-wise operation that loses phase information, and $\boldsymbol{w} \in \mathbb{R}^m$ is noise. PR is needed when it is infeasible to measure phase information, as often occurs in optics [38], computational biology [39], astronomy [40], X-ray crystallography [41, 42], coherent diffraction imaging [43], speech recognition [44], electron microscopy [45], holography [46], non-line-of-sight imaging [47], and other fields. PR is challenging because, even with full-rank $\boldsymbol{A}$ and no noise, there exist many hypotheses of $\boldsymbol{x}$ that explain the phaseless measurements $\boldsymbol{y}$ [36].

### 1.2.1 Prior Art

Various approaches to PR have been proposed. Classical methods, like the Gerchberg-Saxton (GS) [48] and Hybrid Input-Output (HIO) [49] algorithms, are based on iterative projection. Although these algorithms and their variants [50–52] are simple to implement and fast to execute, their output qualities don't compete with those of contemporary methods, as we show in the sequel.

A more modern approach is to formulate PR as negative log-likelihood (NLL) minimization and solve it using gradient-based iterative methods. The PR NLL is non-convex, however, and so spectral initialization strategies have been proposed in an attempt to avoid bad local minima [53–55]. Spectral initialization tends to work well with randomized $\boldsymbol{A}$ [53–55] but often fails for the deterministic Fourier $\boldsymbol{A}$ (see, e.g., [56]). As a more direct attack on non-convexity, convex relaxations such

as PhaseLift [57] and PhaseMax [58] have been proposed (see [59] for an overview). However, these relaxations manifest as semidefinite programs in $d^2$-dimensional space, which are computationally impractical for imaging applications with $d$-pixel images. Approximate Message Passing (AMP) algorithms have also been proposed for PR under sparsity-based signal models [60, 61]. Although near-optimal for high-dimensional i.i.d. or rotationally invariant random $\boldsymbol{A}$ [62, 63], they tend to diverge for deterministic Fourier $\boldsymbol{A}$.

For image PR, various approaches have been proposed to exploit the prior knowledge that $\boldsymbol{x}$ is an image. For example, the plug-and-play (PnP) [21] and RED [22, 23] frameworks have been adapted to PR in [56,64,65]. As mentioned before, these methods iterate between NLL reduction and denoising, allowing them to harness the power of deep-network-based image denoisers such as DnCNN [27], and they tend to work well with both random and deterministic $\boldsymbol{A}$. AMP-based PnP PR approaches have also been proposed [66, 67], but they tend to struggle with deterministic Fourier $\boldsymbol{A}$ [56]. The Compressed-Sensing Generative Model (GSGM) framework [16] was applied to PR in [68], where—given an image generator $g_{\boldsymbol{\theta}}(\cdot)$ with fixed parameters $\boldsymbol{\theta}$—one searches for a code vector $\boldsymbol{z}$ for which $|\boldsymbol{A}g_{\boldsymbol{\theta}}(\boldsymbol{z})|$ matches the phaseless measurements $\boldsymbol{y}$. A variation [69, 70] inspired by Deep Image Prior (DIP) [71] fixes $\boldsymbol{z}$ and instead optimizes the generator parameters $\boldsymbol{\theta}$. A further evolution [72] applies dropout to $\boldsymbol{\theta}$ in an effort to approximate MMSE estimation. One weakness of these DIP-based approaches is that their inference speed is about $10\times$ slower than PnP methods like prDeep [72, Table 4]. End-to-end deep networks have also been proposed to learn the inverse mapping from $\boldsymbol{y}$ to $\boldsymbol{x}$ in PR [73, 74], but they often fail with deterministic $\boldsymbol{A}$ [65]. Recently, diffusion methods have been proposed to sample from the posterior

distribution [75–79] in PR. Some of them [78, 79] integrate Monte-Carlo sampling to enable tractable theoretical analysis at the expense of slower inference time. In Section 3.3, we test [76, 77], which have similar inference times to PnP methods, but find that they don't perform as well.

### 1.2.2 Summary of Contribution 2: Deep EC for Phase Retrieval

In Chapter 3, we propose a novel approach to PR based on the expectation-consistent (EC) approximation framework [28, 80], which we refer to as "deepECpr." Different from existing EC-based methods like VAMP [81] and GVAMP [61], deepECpr avoids the need for large random $\boldsymbol{A}$ by employing EC in a different way. In addition, deepECpr includes a novel "stochastic damping" scheme to further mitigate potential deviations from EC modeling assumptions. Like PnP, RED, and diffusion-based approaches to PR, deepECpr iteratively calls a deep-net denoiser, but it converges in significantly fewer iterations to more accurate solutions. We attribute deepECpr's excellent performance to its ability to track error-variances, thereby allowing efficient use of the denoiser.

## 1.3 Leveraging Diffusion Models for Inverse Problems

Generating samples from complicated real-world distributions $p_0$ is a longstanding problem in machine learning. Recently, diffusion modeling [82–87] has emerged as a powerful approach to this problem. In diffusion, the forward process gradually corrupts samples $\boldsymbol{x}_0 \sim p_0$ with ever-increasing amounts of noise to obtain $\boldsymbol{x}_T$ for some time $T$. The reverse process starts with pure noise $\boldsymbol{x}_T$ and gradually denoises intermediate samples $\boldsymbol{x}_t$ for $t \in (0, T)$, to ultimately generate a sample $\boldsymbol{x}_0 \sim p_0$. More details will be provided in the sequel.

In inverse problems like inpainting, deblurring, super-resolution, and phase retrieval, we wish to recover $\boldsymbol{x}_0 \sim p_0$ from possibly ill-posed and noisy measurements $\boldsymbol{y}$ in an unsupervised manner. Furthermore, we may wish to draw samples from the posterior distribution $p(\boldsymbol{x}_0|\boldsymbol{y})$. Recently, diffusion-based methods have emerged as an effective approach to addressing these challenges [88]. There, a diffusion model is trained to generate samples from $p_0$ and, at test time, the reverse process is modified to incorporate knowledge of the measurements $\boldsymbol{y}$, with the goal of sampling from the posterior distribution $p(\boldsymbol{x}_0|\boldsymbol{y})$. When implementing the reverse process, the main challenge is approximating the conditional score $\nabla_{\boldsymbol{x}} \ln p_t(\boldsymbol{x}_t|\boldsymbol{y})$ at each step $t$, where $\boldsymbol{x}_t$ is a additive-white-Gaussian-noise (AWGN) corrupted and possibly scaled version of $\boldsymbol{x}_0 \in \mathbb{R}^d$, and $\boldsymbol{y} \in \mathbb{R}^m$ is treated as a draw from a likelihood function $p(\boldsymbol{y}|\boldsymbol{x}_0)$.

### 1.3.1 Prior Art

For linear inverse problems of the form

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \sigma_{\mathsf{w}}\boldsymbol{w}, \quad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{1.6}$$

with known $\boldsymbol{A}$ and $\sigma_{\mathsf{w}}$, several approximations $\widehat{\boldsymbol{x}}_{0|t,\boldsymbol{y}} \approx \mathrm{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}\}$ have been proposed that avoid the need to train a $\boldsymbol{y}$-dependent model. For the noiseless case (i.e., $\sigma_{\mathsf{w}} = 0$), DDNM [89] uses the pre-trained unconditional denoiser to compute $\widehat{\boldsymbol{x}}_{0|t} \triangleq D_\theta(\boldsymbol{x}_t, \sigma_t)$ and then uses a hard data-consistency step to obtain

$$\widehat{\boldsymbol{x}}_{0|t,\boldsymbol{y}} = \boldsymbol{A}^+\boldsymbol{y} + (\boldsymbol{I} - \boldsymbol{A}^+\boldsymbol{A})\widehat{\boldsymbol{x}}_{0|t}, \tag{1.7}$$

where $(\cdot)^+$ is the pseudo-inverse. To handle $\sigma_{\mathsf{w}} > 0$, DiffPIR [90] uses the soft data-consistency step

$$\widehat{\boldsymbol{x}}_{0|t,\boldsymbol{y}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2 + \frac{\lambda\sigma_{\mathsf{w}}^2}{\sigma_t^2}\|\boldsymbol{x} - \widehat{\boldsymbol{x}}_{0|t}\|^2, \tag{1.8}$$

where $\lambda$ is a tunable parameter. If a singular value decomposition (SVD) is not available to compute (1.8), one step of gradient descent is used as an approximation. DDRM [91] is a related scheme that requires an SVD, which is prohibitive in many applications. DDS [92] proposes to approximately solve

$$\arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2 + \lambda\|\boldsymbol{x} - \widehat{\boldsymbol{x}}_{0|t}\|^2 \tag{1.9}$$

using a few ($M$) steps of conjugate gradients (CG) initialized with $\widehat{\boldsymbol{x}}_{0|t}$. In this case, both $M$ and $\lambda$ are tuning parameters.

For possibly non-linear inverse problems, which can be characterized by their likelihood function $p(\boldsymbol{y}|\boldsymbol{x}_0)$, other methods are needed. Perhaps the best-known approach, used by methods like DPS [76] and ΠGDM [93], is to combine Tweedie's formula [94]

$$\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{y}) = \frac{\mathrm{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}\} - \boldsymbol{x}_t}{\sigma_t^2}, \tag{1.10}$$

where $\boldsymbol{x}_t = \boldsymbol{x}_0 + \sigma_t \boldsymbol{n}$ for $\boldsymbol{x}_0 \sim p_0$ and $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, with Bayes rule $p(\boldsymbol{x}_t|\boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x}_t)p(\boldsymbol{x}_t)/p(\boldsymbol{y})$ to write

$$\begin{aligned}
\mathrm{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}\} &= \boldsymbol{x}_t + \sigma_t^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{y}) && (1.11)\\
&= \boldsymbol{x}_t + \sigma_t^2 [\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)] && (1.12)\\
&= \boldsymbol{x}_t + \sigma_t^2 \left[\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t) + \frac{\mathrm{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t\} - \boldsymbol{x}_t}{\sigma_t^2}\right] && (1.13)\\
&= \sigma_t^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t) + \mathrm{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t\}, && (1.14)
\end{aligned}$$

after which $\widehat{\boldsymbol{x}}_{0|t} = D_\theta(\boldsymbol{x}_t, \sigma_t)$ is used in place of $\mathrm{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t\}$. But since $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is unknown, $p(\boldsymbol{y}|\boldsymbol{x}_t) = \int p(\boldsymbol{y}|\boldsymbol{x}_0)p(\boldsymbol{x}_0|\boldsymbol{x}_t)\,\mathrm{d}\boldsymbol{x}$ in (1.14) is intractable. Consequently, DPS [76] approximates $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ as $\delta(\boldsymbol{x}_0 - \widehat{\boldsymbol{x}}_{0|t})$, while ΠGDM [93] approximates $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ as $\mathcal{N}(\boldsymbol{x}_0; \widehat{\boldsymbol{x}}_{0|t}, \sigma_{0|t}^2 \boldsymbol{I})$ for some $\sigma_{0|t}$. These approaches require backpropagation

through $D_\theta(\boldsymbol{x}_t, \sigma_t)$, however, which increases computational complexity and prevents batch generation of multiple samples.

There are, of course, many other ways to design diffusion posterior samplers, as detailed in the recent overview [88]. For example, RED-diff [95] uses a stochastic version of the RED algorithm [22], whose regularization is based on the score function [23]. Another approach is to use Markov-chain Monte Carlo (MCMC) methods as inner iterations of the reverse process [78, 79, 96, 97].

### 1.3.2 Summary of Contribution 3: Recursive Annealed Posterior Sampling Framework for Inverse Problems

As seen above, pre-trained diffusion models can be leveraged to solve imaging inverse problems in an unsupervised manner. It is achieved by combining the pre-trained score function with the measurement likelihood function to approximate the measurement-conditional score function. While several approaches have been proposed for this approximation, we take a first-principles approach inspired by the idea of a decoupled forward process during reverse diffusion. Building on this foundation, Chapter 4 introduces Recursive Annealed Posterior Sampling (RAPS), a novel diffusion-based posterior sampling framework. To further enhance its efficiency and robustness, we propose StRAPS, an extension of RAPS that incorporates key improvements such as adaptive weighting, which eliminates the need for manual tuning, and a stochasticity-enhancing technique in the update steps to mitigate discretization and approximation errors. Additionally, we introduce an expectation-propagation method that extends our approach to nonlinear inverse problems, such as phase retrieval. The proposed framework is a practical and scalable solution for a wide

range of inverse problems, achieving state-of-the-art performance without requiring extensive hyperparameter tuning.

# Chapter 2: Denoising Generalized Expectation-Consistent Approximation for MR Image Recovery

## 2.1 Background

### 2.1.1 Magnetic resonance imaging

We now detail the version of the linear system model

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_{\mathsf{true}} + \boldsymbol{w}, \tag{2.1}$$

that manifests in $C$-coil MRI. There, $\boldsymbol{x}_{\mathsf{true}} \in \mathbb{C}^N$ is a vectorized version of the $N$-pixel image that we wish to recover, $\boldsymbol{y} \in \mathbb{C}^{CM}$ are the so-called "k-space" measurements, and

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{M}\boldsymbol{F}\,\mathrm{Diag}(\boldsymbol{s}_1) \\ \vdots \\ \boldsymbol{M}\boldsymbol{F}\,\mathrm{Diag}(\boldsymbol{s}_C) \end{bmatrix}. \tag{2.2}$$

In (2.2), $\boldsymbol{F} \in \mathbb{C}^{N\times N}$ is a unitary 2D discrete Fourier transform (DFT), $\boldsymbol{M} \in \mathbb{R}^{M\times N}$ is a sampling mask formed from $M$ rows of the identity matrix $\boldsymbol{I} \in \mathbb{R}^{N\times N}$, and $\boldsymbol{s}_c \in \mathbb{C}^N$ is the $c$th coil-sensitivity map. In the special case of single-coil MRI, we have $C = 1$ and $\boldsymbol{s}_1 = \boldsymbol{1}$, where $\boldsymbol{1}$ denotes the all-ones vector. In MRI, the ratio $R \triangleq N/M$ is known as the "acceleration rate." When $R > 1$, the matrix $\boldsymbol{A}$ can be column-rank deficient and/or poorly conditioned even when $C \geq R$, and so prior knowledge of $\boldsymbol{x}_{\mathsf{true}}$ must be exploited for accurate recovery.

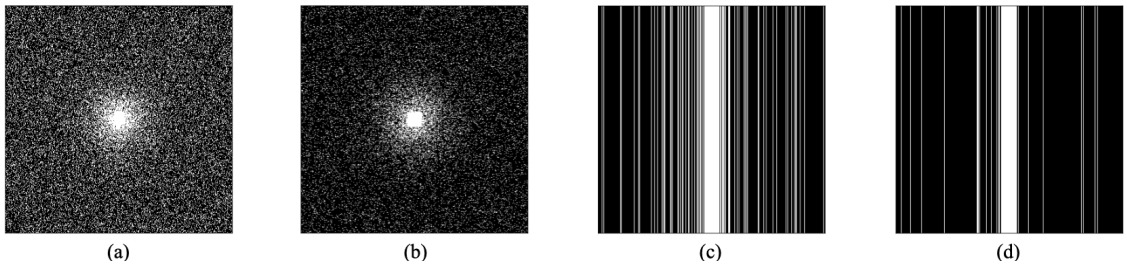


Figure 2.1: Examples of sampling masks $\boldsymbol{M}$: (a) 2D point sampling at $R = 4$, (b) 2D point sampling at $R = 8$ with a $24 \times 24$ fully sampled central autocalibration region, (c) 2D line sampling at $R = 4$ with a 24-wide fully sampled central autocalibration region, and (d) 2D line sampling at $R = 8$ with a 24-wide fully sampled central autocalibration region.

In practical MRI, physical constraints govern the construction of the sampling mask $\boldsymbol{M}$. For example, samples are always collected along lines or curves in k-space. In clinical practice, it is most common to sample along lines parallel to one dimension of k-space, as illustrated in Figs. 2.1(c)-(d) for 2D sampling. We will refer to this approach as "2D line sampling." In this case, one dimension of k-space is fully sampled and the other dimension is subsampled. For the subsampled dimension, it is common to sample pseudorandomly or randomly, but with a higher density near the k-space origin, as shown in Figs. 2.1(c)-(d). Also, when using ESPIRiT to estimate the coil-sensitivity maps $\{\boldsymbol{s}_c\}$, one must include a fully-sampled "autocalibration" region centered at the origin, as shown in Figs. 2.1(b)-(d).

2D line sampling, while attractive from an implementation standpoint, poses challenges for signal reconstruction due to high levels of coherence [98] in the resulting $\boldsymbol{A}$ matrix. This has led some algorithm designers to consider "2D point sampling" masks such as those shown in Fig. 2.1(a)-(b), since they yield $\boldsymbol{A}$ with much lower coherence [99]. But such masks are rarely encountered in practical 2D MR imaging. It is, however, possible to encounter a 2D point mask as a byproduct of the following

3D acquisition process: i) acquire a 3D k-space volume using 3D line sampling, ii) perform an inverse DFT along the fully sampled dimension, and iii) slice along that dimension to obtain a stack of 2D k-space acquisitions. The location of each line in 3D k-space determines the location of the respective point sample in 2D k-space, and these locations can be freely chosen. But 3D acquisition is uncommon because it is susceptible to motion; in 2D acquisition, the patient must lie still for the acquisition of a single slice, whereas in 3D acquisition the patient must lie still for the acquisition of an entire volume. We include experiments with 2D point masks only to compare with the VDAMP family of algorithms [100–103] discussed in the sequel, since these algorithms are all designed around the use of 2D point masks.

Although this work focuses on MRI, the methods we propose apply to any application where the goal is to recover a signal from undersampled Fourier measurements.

### 2.1.2 Plug-and-play recovery

Many algorithms have been proposed to solve the optimization problem (1.2) (see, e.g., [9]). The typical assumptions are that $g_1$ is convex and differentiable, $\nabla g_1$ is Lipschitz with constant $L > 0$, and $g_2$ is convex but possibly not differentiable, which allows sparsity-inducing regularizations like $g_2(\boldsymbol{x}) = \lambda\|\boldsymbol{\Psi}\boldsymbol{x}\|_1$. One of the most popular approaches is ADMM [104], summarized by the iterations

$$\boldsymbol{x}_1 \leftarrow \operatorname{prox}_{\gamma^{-1}g_1}(\boldsymbol{x}_2 - \boldsymbol{u}) \tag{2.3a}$$

$$\boldsymbol{x}_2 \leftarrow \operatorname{prox}_{\gamma^{-1}g_2}(\boldsymbol{x}_1 + \boldsymbol{u}) \tag{2.3b}$$

$$\boldsymbol{u} \leftarrow \boldsymbol{u} + (\boldsymbol{x}_1 - \boldsymbol{x}_2)\,, \tag{2.3c}$$

where $\gamma$ is a tunable parameter[1] that affects convergence speed but not the fixed point, and

$$\text{prox}_{\rho}(\boldsymbol{r}) \triangleq \arg\min_{\boldsymbol{x}} \left\{ \rho(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{r}\|^2 \right\}. \tag{2.4}$$

For example, when $g_1(\boldsymbol{x}) = \frac{\gamma_w}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2$, we get

$$\text{prox}_{\gamma^{-1}g_1}(\boldsymbol{r}) = \left(\gamma_w \boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} + \gamma \boldsymbol{I}\right)^{-1}\left(\gamma_w \boldsymbol{A}^{\mathsf{H}}\boldsymbol{y} + \gamma \boldsymbol{r}\right). \tag{2.5}$$

Based on the prox definition in (2.4), ADMM step (2.3b) can be interpreted as MAP estimation [105] of $\boldsymbol{x}_{\mathsf{true}}$ with prior $p(\boldsymbol{x}_{\mathsf{true}}) \propto e^{-g_2(\boldsymbol{x}_{\mathsf{true}})}$ from an observation $\boldsymbol{r} = \boldsymbol{x}_{\mathsf{true}} + \boldsymbol{e}$ of the true signal corrupted by $\gamma$-precision AWGN $\boldsymbol{e}$, i.e., MAP denoising. This observation led Venkatakrishnan et al. [21] to propose that the prox in (2.3b) be replaced by a high-performance image denoiser $\boldsymbol{f}_2 : \mathbb{R}^N \to \mathbb{R}^N$ like BM3D [106], giving rise to PnP-ADMM. It was later proposed to use a DNN-based denoiser in PnP [107], such as DnCNN [27]. Note that when (2.3b) is replaced with a denoising step of the form "$\boldsymbol{x}_2 \leftarrow \boldsymbol{f}_2(\boldsymbol{x}_1 + \boldsymbol{u})$," the parameter $\gamma$ does affect the fixed-point and thus must be tuned to obtain the best recovery accuracy.

The PnP framework was later extended to other algorithms, such as primal-dual splitting (PDS) in [107, 108] and proximal gradient descent (PGD) in [107, 109]. For use in the sequel, we write the PGD algorithm as

$$\boldsymbol{x}_1 \leftarrow \boldsymbol{x}_2 - \mu \nabla g_1(\boldsymbol{x}_2) \tag{2.6a}$$

$$\boldsymbol{x}_2 \leftarrow \text{prox}_{\mu g_2}(\boldsymbol{x}_1), \tag{2.6b}$$

where $\mu \in (0, 1/L)$ and $L$ is the Lipschitz constant of $\nabla g_1$. For example, when $g_1(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2$, we get $\nabla g_1(\boldsymbol{x}) = \boldsymbol{A}^{\mathsf{H}}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y})$. For all of these PnP incarnations,

[1]The parameter $\gamma$ arises from the augmented Lagrangian used by ADMM: $g_1(\boldsymbol{x}_1) + g_2(\boldsymbol{x}_2) + \text{Re}\{\boldsymbol{u}^{\mathsf{H}}(\boldsymbol{x}_1-\boldsymbol{x}_2)\} + \frac{\gamma}{2}\|\boldsymbol{x}_1-\boldsymbol{x}_2\|^2$.

the prox step in the original optimization algorithm is replaced by a high-performance denoiser $\boldsymbol{f}_2$. As shown in the recent overview [24], PnP methods have been shown to significantly outperform sparsity-based approaches in MRI, as well as end-to-end DNNs in limited-data and mismatched-$\boldsymbol{A}$ scenarios.

Although PnP algorithms work well for MRI [24,110], there is room for improvement. For example, while image denoisers are typically designed/trained to remove the effects of AWGN, PnP algorithms do not provide the denoiser with an AWGN-corrupted input at each iteration. Rather, the denoiser's input error has iteration-dependent statistics that are difficult to analyze or predict.

### 2.1.3 Approximate message passing

For the model (2.1) with $\boldsymbol{w} \sim \mathcal{N}(0, \tau_w \boldsymbol{I})$, the AMP algorithm[2] [26, 112] manifests as the following iteration over $t = 0, 1, 2, \ldots$:

$$\boldsymbol{v}^{t+1} = \left(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^t\right)\beta + \frac{\boldsymbol{v}^t}{M}\operatorname{tr}\left\{\nabla \boldsymbol{f}_2^t\left(\boldsymbol{x}^{t-1} + \beta \boldsymbol{A}^{\mathsf{H}}\boldsymbol{v}^t\right)\right\} \tag{2.7a}$$

$$\tau^{t+1} = \frac{1}{M}\left\|\boldsymbol{v}^{t+1}\right\|^2 \tag{2.7b}$$

$$\boldsymbol{x}^{t+1} = \boldsymbol{f}_2^{t+1}\left(\boldsymbol{x}^t + \beta \boldsymbol{A}^{\mathsf{H}}\boldsymbol{v}^{t+1}\right) \tag{2.7c}$$

initialized as $\boldsymbol{v}^0 = \boldsymbol{0} = \boldsymbol{x}^0$, where $\boldsymbol{f}_2^t(\cdot)$ is the iteration-$t$ denoising function (which may depend on $\tau^t$), $\operatorname{tr}\{\nabla \boldsymbol{f}_2^t(\boldsymbol{r})\}$ is the trace of the Jacobian of $\boldsymbol{f}_2^t$ at $\boldsymbol{r}$, and $\beta = \sqrt{N}/\|\boldsymbol{A}\|_F$. The last term in (2.7a), known as the "Onsager correction," is a key component of the AMP algorithm. Without it, (2.7) would reduce to the PnP version of the PGD algorithm (2.6) with $\mu = \beta^2$.

The goal of Onsager correction is to make the denoiser input error

$$\boldsymbol{e}^{t+1} \triangleq \boldsymbol{x}^t + \beta \boldsymbol{A}^{\mathsf{H}}\boldsymbol{v}^{t+1} - \boldsymbol{x}_{\mathsf{true}} \tag{2.8}$$

[2]For generalized linear models, one would instead use the Generalized AMP algorithm from [111].

behave like a realization of WGN with variance $\tau^{t+1}$, where $\tau^{t+1}$ is given in (2.7b). Note that if

$$\boldsymbol{e}^{t+1} \sim \mathcal{N}(\boldsymbol{0}, \tau^{t+1}\boldsymbol{I}) \tag{2.9}$$

did hold, it would be straightforward to design the denoiser $\boldsymbol{f}_2^{t+1}$ for MAP or MMSE optimality. For example, in (1.2), if we interpret $g_1(\boldsymbol{x})$ as the log-likelihood and $g_2(\boldsymbol{x})$ as the log-prior, then $g_1(\boldsymbol{x}) + g_2(\boldsymbol{x})$ becomes the log-posterior (up to a constant) and so $\widehat{\boldsymbol{x}}$ in (1.2) becomes the MAP estimate [113]. Thus, for the case of MAP estimation, we would use the MAP denoiser $\boldsymbol{f}_2^t(\boldsymbol{r}) = \text{prox}_{\tau^t g_2}(\boldsymbol{r})$, and $\boldsymbol{x}^t$ would approach the MAP estimate as $t \to \infty$ [26]. On the other hand, for the case of MMSE estimation, where we would like to compute the conditional mean $\widehat{\boldsymbol{x}}_{\mathsf{mmse}} \triangleq \text{E}\{\boldsymbol{x}|\boldsymbol{y}\}$, we would use the MMSE denoiser $\boldsymbol{f}_2^t(\boldsymbol{r}) = \text{E}\{\boldsymbol{x} \,|\, \boldsymbol{r}\}$ for $\boldsymbol{r} = \boldsymbol{x}_{\mathsf{true}} + \boldsymbol{e}$ with $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \tau^t\boldsymbol{I})$ [112].

Importantly, when the forward operator $\boldsymbol{A} \in \mathbb{R}^{P\times N}$ is i.i.d. sub-Gaussian, the dimensions $P, N \to \infty$ with a fixed ratio $P/N$, and $\boldsymbol{f}_2^t$ is Lipschitz, [114,115] established that the WGN property (2.9) does indeed hold. Furthermore, defining the MSE $\mathcal{E}^t \triangleq \frac{1}{N}\|\boldsymbol{x}^t - \boldsymbol{x}_{\mathsf{true}}\|^2$, [114, 115] established that AMP obeys the following scalar state-evolution over $t = 0, 1, 2, \ldots$:

$$\tau^t = \tau_w + \frac{N}{P}\mathcal{E}^t \tag{2.10a}$$

$$\mathcal{E}^{t+1} = \frac{1}{N}\,\text{E}\left\{\left\|\boldsymbol{f}_2^t\left(\boldsymbol{x}_{\mathsf{true}} + \mathcal{N}\left(\boldsymbol{0}, \tau^t\boldsymbol{I}\right)\right) - \boldsymbol{x}_{\mathsf{true}}\right\|^2\right\}. \tag{2.10b}$$

Remarkably, the AMP state evolution shows that, in the large-system limit, the trajectory of the mean-squared recovery error can be predicted in advance knowing only the dimensions of i.i.d. sub-Gaussian $\boldsymbol{A}$ (not the values in $\boldsymbol{A}$) and the MSE behavior of the denoiser $\boldsymbol{f}_2^t(\cdot)$ when faced with the task of removing white Gaussian noise. Moreover, when $\boldsymbol{f}_2^t$ is the MMSE denoiser and the state-evolution has a unique

fixed point, [114, 115] established that AMP provably converges to the MMSE-optimal estimate $\widehat{\boldsymbol{x}}_{\mathsf{mmse}}$. These theoretical results were first established for separable denoisers $\boldsymbol{f}_2$ in [114] and later extended to non-separable denoisers in [115]. By "separable" we mean that $\boldsymbol{f}_2$ takes the form $\boldsymbol{f}_2(\boldsymbol{x}) = [f_2(x_1), \ldots, f_2(x_N)]^\top$ for some scalar denoiser $f_2 : \mathbb{R} \to \mathbb{R}$.

For practical image recovery problems, [116] proposed to approximate the MMSE denoiser by a high-performance image denoiser like BM3D or a DNN, and called it "denoising-AMP" (D-AMP). Since these image denoisers are non-separable and high-dimensional, the trace-Jacobian term in (2.7a) (known as the "divergence") is difficult to compute, and so D-AMP uses the Monte-Carlo approximation [117]

$$\operatorname{tr}\left\{\nabla \boldsymbol{f}_2^t(\boldsymbol{r})\right\} \approx \delta^{-1}\boldsymbol{q}^{\mathsf{H}}\left[\boldsymbol{f}_2^t(\boldsymbol{r}+\delta\boldsymbol{q}) - \boldsymbol{f}_2^t(\boldsymbol{r})\right], \tag{2.11}$$

where $\boldsymbol{q}$ is a fixed realization of $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\delta$ is a small positive number. D-AMP performs very well with large i.i.d. sub-Gaussian $\boldsymbol{A}$, but can diverge with non-random $\boldsymbol{A}$, such as those encountered in MRI (recall (2.2)).

### 2.1.4 Expectation-consistent approximation and VAMP for SLM

Expectation-consistent (EC) approximation [118] is an inference framework with close connections to both PnP-ADMM and AMP. In EC, one is assumed to have access to the prior density $p_{\mathrm{x}}(\boldsymbol{x})$ on $\boldsymbol{x}_{\mathsf{true}}$ and the likelihood function $\ell(\boldsymbol{x}; \boldsymbol{y})$, and the goal is to approximate the mean of the posterior $p_{\mathrm{x}|\mathrm{y}}(\boldsymbol{x}|\boldsymbol{y})$, i.e., the MMSE estimate $\widehat{\boldsymbol{x}}_{\mathsf{mmse}} = \mathrm{E}\{\boldsymbol{x}|\boldsymbol{y}\}$. Although Bayes rule says that $p_{\mathrm{x}|\mathrm{y}}(\boldsymbol{x}|\boldsymbol{y}) = Z^{-1}(\boldsymbol{y}) p_{\mathrm{x}}(\boldsymbol{x})\ell(\boldsymbol{x}; \boldsymbol{y})$ for $Z(\boldsymbol{y}) \triangleq \int p_{\mathrm{x}}(\boldsymbol{x})\ell(\boldsymbol{x}; \boldsymbol{y})\,\mathrm{d}\boldsymbol{x}$, this integral is usually too difficult to compute in the

high-dimensional case. But note that we can write

$$
\begin{aligned}
& p_{\mathrm{x}|\mathrm{y}}(\boldsymbol{x}|\boldsymbol{y}) \\
& = \arg\min_{q} D\big(q(\boldsymbol{x})\big\|p_{\mathrm{x}|\mathrm{y}}(\boldsymbol{x}|\boldsymbol{y})\big) && (2.12) \\
& = \arg\min_{q} D\big(q(\boldsymbol{x})\big\|\ell(\boldsymbol{x};\boldsymbol{y})\big) + D\big(q(\boldsymbol{x})\big\|p_{\mathrm{x}}(\boldsymbol{x})\big) + H\big(q(\boldsymbol{x})\big) \\
& = \arg\min_{q_1,q_2,q_3} J_{\mathsf{Gibbs}}(q_1,q_2,q_3) \ \text{ such that } q_1=q_2=q_3, && (2.13)
\end{aligned}
$$

where the minimizations are conducted over sets of probability densities, $D(q_1\|p_{\mathrm{x}}) \triangleq \int q_1(\boldsymbol{x}) \log \frac{q_1(\boldsymbol{x})}{p_{\mathrm{x}}(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x}$ is the Kullback-Liebler (KL) divergence from $p_{\mathrm{x}}$ to $q_1$, $H(q_3) \triangleq -\int q_3(\boldsymbol{x}) \log q_3(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$ is the differential entropy of $q_3$, and

$$
\begin{aligned}
& J_{\mathsf{Gibbs}}(q_1,q_2,q_3) \\
& \triangleq D\big(q_1(\boldsymbol{x})\big\|\ell(\boldsymbol{x};\boldsymbol{y})\big) + D\big(q_2(\boldsymbol{x})\big\|p_{\mathrm{x}}(\boldsymbol{x})\big) + H\big(q_3(\boldsymbol{x})\big) && (2.14)
\end{aligned}
$$

where $J_{\mathsf{Gibbs}}(q,q,q)$ is known as the Gibbs free energy of $q$. So, if (2.13) could be solved, it would give a way to compute the posterior that avoids computing $Z(\boldsymbol{y})$. However, (2.13) is generally too difficult to solve, and so it was proposed in [118] to relax the equality constraints in (3.5) to moment-matching constraints, i.e.,

$$
\begin{aligned}
& \arg\min_{q_1,q_2,q_3} J_{\mathsf{Gibbs}}(q_1,q_2,q_3) \text{ such that} && (2.15) \\
& \qquad \begin{cases} \mathrm{E}\{\boldsymbol{x}|q_1\} = \mathrm{E}\{\boldsymbol{x}|q_2\} = \mathrm{E}\{\boldsymbol{x}|q_3\} \\ \mathrm{tr}(\mathrm{Cov}\{\boldsymbol{x}|q_1\}) = \mathrm{tr}(\mathrm{Cov}\{\boldsymbol{x}|q_2\}) = \mathrm{tr}(\mathrm{Cov}\{\boldsymbol{x}|q_3\}), \end{cases}
\end{aligned}
$$

where $\mathrm{E}\{\boldsymbol{x}|q_i\}$ and $\mathrm{Cov}\{\boldsymbol{x}|q_i\}$ denote the mean and covariance of $\boldsymbol{x}$ under $\boldsymbol{x} \sim q_i$ for $i = 1, 2, 3$, respectively. The authors of [118] then showed that the optimization

problem (2.15) is solved by the densities

$$q_1(\boldsymbol{x};\boldsymbol{r}_1,\gamma_1) \propto \ell(\boldsymbol{x};\boldsymbol{y})\mathcal{N}(\boldsymbol{x};\boldsymbol{r}_1,\boldsymbol{I}/\gamma_1) \tag{2.16}$$

$$q_2(\boldsymbol{x};\boldsymbol{r}_2,\gamma_2) \propto p_{\mathrm{x}}(\boldsymbol{x})\mathcal{N}(\boldsymbol{r}_2;\boldsymbol{x},\boldsymbol{I}/\gamma_2) \tag{2.17}$$

$$q_3(\boldsymbol{x};\widehat{\boldsymbol{x}},\eta) = \mathcal{N}(\boldsymbol{x};\widehat{\boldsymbol{x}},\boldsymbol{I}/\eta) \tag{2.18}$$

for the values of $(\boldsymbol{r}_1,\gamma_1,\boldsymbol{r}_2,\gamma_2,\widehat{\boldsymbol{x}},\eta)$ that lead to the satisfaction of the constraints in (2.15). The resulting $\widehat{\boldsymbol{x}}$ approximates the MMSE estimate $\widehat{\boldsymbol{x}}_{\mathsf{mmse}}$ and $\eta^{-1}$ approximates the resulting MMSE $\frac{1}{N}\operatorname{tr}(\operatorname{Cov}\{\boldsymbol{x}|\boldsymbol{y}\})$.

Although there is generally no closed-form expression for the moment-matching values of $(\boldsymbol{r}_1,\gamma_1,\boldsymbol{r}_2,\gamma_2,\widehat{\boldsymbol{x}},\eta)$, one can iteratively solve for them using the EC algorithm shown in Alg. 1 (a form of expectation propagation (EP) [119]) using the estimation functions

$$\boldsymbol{f}_1(\boldsymbol{r}_1;\gamma_1) = \mathrm{E}\{\boldsymbol{x}|q_1\} = \int \boldsymbol{x}\, q_1(\boldsymbol{x};\boldsymbol{r}_1;\gamma_1)\,\mathrm{d}\boldsymbol{x} \tag{2.19}$$

$$\boldsymbol{f}_2(\boldsymbol{r}_2;\gamma_2) = \mathrm{E}\{\boldsymbol{x}|q_2\} = \int \boldsymbol{x}\, q_2(\boldsymbol{x};\boldsymbol{r}_2;\gamma_2)\,\mathrm{d}\boldsymbol{x}. \tag{2.20}$$

It is straightforward to show (see, e.g., [28]) that, at a fixed point of Alg. 1, one obtains $\widehat{\boldsymbol{x}}_1 = \widehat{\boldsymbol{x}}_2 = \widehat{\boldsymbol{x}}$ and $\eta_1 = \eta_2 = \eta = \gamma_1 + \gamma_2$.

For WGN-corrupted linear measurements $\boldsymbol{y}$ as in (2.1), the likelihood becomes $\ell(\boldsymbol{x};\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y};\boldsymbol{A}\boldsymbol{x},\boldsymbol{I}/\gamma_w)$ and so $\boldsymbol{f}_1$ in (2.19) manifests as

$$\boldsymbol{f}_1(\boldsymbol{r}_1;\gamma_1) = \big(\gamma_w\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} + \gamma_1\boldsymbol{I}\big)^{-1}\big(\gamma_w\boldsymbol{A}^{\mathsf{H}}\boldsymbol{y} + \gamma_1\boldsymbol{r}_1\big). \tag{2.21}$$

This $\boldsymbol{f}_1$ can be interpreted as the MMSE estimator of $\boldsymbol{x}_{\mathsf{true}}$ from the measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_{\mathsf{true}} + \mathcal{N}(\boldsymbol{0},\boldsymbol{I}/\gamma_w)$ under the pseudo-prior $\boldsymbol{x}_{\mathsf{true}} \sim \mathcal{N}(\boldsymbol{r}_1,\boldsymbol{I}/\gamma_1)$. Meanwhile $\boldsymbol{f}_2$ in (2.20) can be interpreted as the MMSE estimator of $\boldsymbol{x}_{\mathsf{true}}$ from the pseudo-measurements $\boldsymbol{r}_2 = \boldsymbol{x}_{\mathsf{true}} + \mathcal{N}(\boldsymbol{0},\boldsymbol{I}/\gamma_2)$ under the prior $\boldsymbol{x}_{\mathsf{true}} \sim p_{\mathrm{x}}(\boldsymbol{x})$. In other words,

**Algorithm 1** EC / VAMP

**Require:** $\boldsymbol{f}_1(\cdot;\cdot)$ and $\boldsymbol{f}_2(\cdot;\cdot)$.

Select initial $\boldsymbol{r}_1 \in \mathbb{R}^N, \gamma_1 > 0$
**repeat**
  // Measurement fidelity
  $\widehat{\boldsymbol{x}}_1 \leftarrow \boldsymbol{f}_1(\boldsymbol{r}_1;\gamma_1)$
  $\eta_1 \leftarrow \gamma_1 N / \operatorname{tr}(\nabla \boldsymbol{f}_1(\boldsymbol{r}_1;\gamma_1))$
  $\gamma_2 \leftarrow \eta_1 - \gamma_1$
  $\boldsymbol{r}_2 \leftarrow (\eta_1 \widehat{\boldsymbol{x}}_1 - \gamma_1 \boldsymbol{r}_1)/\gamma_2$
  // Denoising
  $\widehat{\boldsymbol{x}}_2 \leftarrow \boldsymbol{f}_2(\boldsymbol{r}_2;\gamma_2)$
  $\eta_2 \leftarrow \gamma_2 N / \operatorname{tr}(\nabla \boldsymbol{f}_2(\boldsymbol{r}_2;\gamma_2))$
  $\gamma_1 \leftarrow \eta_2 - \gamma_2$
  $\boldsymbol{r}_1 \leftarrow (\eta_2 \widehat{\boldsymbol{x}}_2 - \gamma_2 \boldsymbol{r}_2)/\gamma_1$
**until** Terminated
**return** $\widehat{\boldsymbol{x}}_2$

$\boldsymbol{f}_2$ can be interpreted as the MMSE denoiser of $\boldsymbol{r}_2$. This pseudo-measurement model is exactly the same one that arises in AMP (recall (2.9)).

For generic $\boldsymbol{A}$, there are no guarantees on the quality of the EC estimate $\widehat{\boldsymbol{x}}$ or even the convergence of Alg. 1. But when $\boldsymbol{A}$ is a right orthogonally invariant (ROI) random matrix, EC has a rigorous high-dimensional analysis. ROI matrices can be understood as those with singular value decompositions of the form $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$, for orthogonal $\boldsymbol{U}$, diagonal $\boldsymbol{S}$, and random $\boldsymbol{V}$ uniformly distributed over the set of orthogonal matrices; the ROI class includes the i.i.d. Gaussian class but is more general. In particular, [81, 120] showed that, for asymptotically large ROI matrices $\boldsymbol{A}$, EC's denoiser input error $\boldsymbol{e}_2 = \boldsymbol{r}_2 - \boldsymbol{x}_{\mathsf{true}}$ obeys

$$\boldsymbol{e}_2 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}/\gamma_2) \tag{2.22}$$

at every iteration, similar to AMP (recall (2.9)). Likewise, macroscopic statistical quantities like MSE $\mathcal{E} = \frac{1}{N}\|\widehat{\boldsymbol{x}} - \boldsymbol{x}_{\mathsf{true}}\|^2$ obey a scalar state evolution. Importantly,

these results hold not only for the MMSE denoising functions $\boldsymbol{f}_2$ specified by EC, but also for general Lipschitz $\boldsymbol{f}_2$ [81, 121]. Due to the tight connections with AMP, the EC algorithm with general Lipschitz $\boldsymbol{f}_2$ was referred to as Vector AMP (VAMP) in [81,121]. A similar rigorous analysis of EC with asymptotically large, right unitarily invariant (RUI) matrices $\boldsymbol{A}$ was given in [122]. For those matrices, the SVD of $\boldsymbol{A}$ takes the form $\boldsymbol{USV}^{\mathsf{H}}$ with random $\boldsymbol{V}$ uniformly distributed over the set of unitary matrices.

Given that the EC/VAMP algorithm can be used with estimation functions other than the MMSE choices in (2.19)-(2.20), one might wonder whether it can be applied to solve optimization problems of the form (1.2), i.e., MAP estimation. This was answered affirmatively in [28]. In particular, it suffices to choose

$$\boldsymbol{f}_1(\boldsymbol{r}_1, \gamma_1) = \text{prox}_{\gamma_1^{-1} g_1}(\boldsymbol{r}_1) \tag{2.23}$$

$$\boldsymbol{f}_2(\boldsymbol{r}_2, \gamma_2) = \text{prox}_{\gamma_2^{-1} g_2}(\boldsymbol{r}_2). \tag{2.24}$$

Furthermore, the resulting EC/VAMP algorithm can be recognized as a form of ADMM. If we fix the values of $\gamma_1$ and $\gamma_2$ over the iterations (which forces $\eta_1 = \eta_2 = \gamma_1 + \gamma_2$) and define $\boldsymbol{u}_1 \triangleq \gamma_1(\widehat{\boldsymbol{x}}_2 - \boldsymbol{r}_1)$ and $\boldsymbol{u}_2 \triangleq \gamma_2(\boldsymbol{r}_2 - \widehat{\boldsymbol{x}}_1)$, we can rewrite EC/VAMP from Alg. 1 as the recursion

$$\widehat{\boldsymbol{x}}_1 \leftarrow \text{prox}_{\gamma_1^{-1} g_1}(\widehat{\boldsymbol{x}}_2 - \boldsymbol{u}_1/\gamma_1) \tag{2.25a}$$

$$\boldsymbol{u}_2 \leftarrow \boldsymbol{u}_1 + \gamma_1(\widehat{\boldsymbol{x}}_1 - \widehat{\boldsymbol{x}}_2) \tag{2.25b}$$

$$\widehat{\boldsymbol{x}}_2 \leftarrow \text{prox}_{\gamma_2^{-1} g_2}(\widehat{\boldsymbol{x}}_1 + \boldsymbol{u}_2/\gamma_2) \tag{2.25c}$$

$$\boldsymbol{u}_1 \leftarrow \boldsymbol{u}_2 + \gamma_2(\widehat{\boldsymbol{x}}_1 - \widehat{\boldsymbol{x}}_2) \tag{2.25d}$$

which is a generalization of ADMM in (2.3) to two dual updates and two penalty parameters. If we additionally constrain $\gamma_1 = \gamma_2 \triangleq \gamma$ then (2.25) reduces to

$$\widehat{\boldsymbol{x}}_1 \leftarrow \text{prox}_{\gamma^{-1}g_1}(\widehat{\boldsymbol{x}}_2 - \boldsymbol{u}) \tag{2.26a}$$

$$\boldsymbol{u} \leftarrow \boldsymbol{u} + (\widehat{\boldsymbol{x}}_1 - \widehat{\boldsymbol{x}}_2) \tag{2.26b}$$

$$\widehat{\boldsymbol{x}}_2 \leftarrow \text{prox}_{\gamma^{-1}g_2}(\widehat{\boldsymbol{x}}_1 + \boldsymbol{u}) \tag{2.26c}$$

$$\boldsymbol{u} \leftarrow \boldsymbol{u} + (\widehat{\boldsymbol{x}}_1 - \widehat{\boldsymbol{x}}_2), \tag{2.26d}$$

which is known as the Peaceman-Rachford or symmetric variant of ADMM, and which is said to converge faster than standard ADMM [123, 124]. The important point is that EC/VAMP can be understood as a generalization of ADMM that i) uses two penalty parameters and ii) adapts those penalty parameters with the iterations.

Inspired by D-AMP [116], a "Denoising VAMP" (D-VAMP) was proposed in [125], which used VAMP with high-performance image denoisers and the Monte-Carlo approximation (2.11). Although D-VAMP was shown to work well with large ROI $\boldsymbol{A}$, it can diverge with non-random $\boldsymbol{A}$, such as those encountered in MRI. Some intuition behind the failure of VAMP with non-ROI $\boldsymbol{A}$ will be given in Section 2.2.1

### 2.1.5 AMP/VAMP for MRI

The versions of $\boldsymbol{A}$ that manifest in linear inverse problems often do not have sufficient randomness for the AMP and EC/VAMP algorithms to work as intended. If used without modification, AMP and EC/VAMP algorithms may simply diverge. This is definitely the case for MRI, where $\boldsymbol{A}$ is the Fourier-based matrix shown in (2.2). Consequently, modified AMP and VAMP algorithms have been proposed specifically for MRI image recovery.

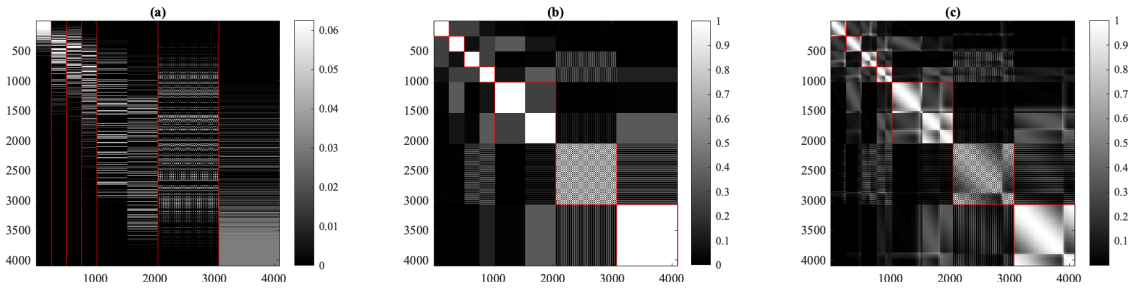


Figure 2.2: Approximate block-diagonality of 2D Fourier-wavelet matrices. Using abs$(\cdot)$ to denote the entry-wise magnitude operation, (a) shows abs$(\boldsymbol{F}\boldsymbol{\Psi}^\top)$ with rows sorted according to distance from the k-space origin, columns sorted according to wavelet subbands, and subband boundaries denoted by red lines. Meanwhile, (b) shows the matrix product abs$(\boldsymbol{F}\boldsymbol{\Psi}^\top)^\top$ abs$(\boldsymbol{F}\boldsymbol{\Psi}^\top)$ and (c) shows abs$(\boldsymbol{G})^\top$ abs$(\boldsymbol{G})$ for the multi-coil Fourier-wavelet matrix $\boldsymbol{G}$ defined in (2.29). The approximate block-diagonality of (b) and (c) suggests that the columns of the 2D Fourier-wavelet matrices are well decoupled in the single- and multi-coil cases.

For example, [126] proposed to use D-AMP (2.7) with $\beta \ll \sqrt{N}/\|\boldsymbol{A}\|_F$, which helps to slow down the algorithm and help it converge, but at the cost of degrading its fixed points, as we show in Section 2.3.5. The authors of [127] instead used damping to help D-VAMP converge without disturbing its fixed points. In conjunction with a novel initialization based on Peaceman-Rachford ADMM, the latter scheme was competitive with PnP-ADMM for single-coil MRI.

For the special case of 2D point-sampled MRI, the principle of density compensation [128] has also been exploited for the design of AMP-based algorithms. For applications where k-space is non-uniformly sampled, density compensation applies a gain to each k-space sample that is proportional to the inverse sampling density at that sample, changing $\boldsymbol{y}$ to $\boldsymbol{G}\boldsymbol{y}$ in (2.1) with diagonal gain matrix $\boldsymbol{G}$. When $\boldsymbol{A}$ uses a 2D point mask, the error in the density-compensated linear estimate $\widehat{\boldsymbol{x}} = \boldsymbol{A}^{\mathsf{H}}\boldsymbol{G}\boldsymbol{y}$ behaves much more like white Gaussian noise than does the error in the standard linear estimate

$\widehat{\boldsymbol{x}} = \boldsymbol{A}^{\mathsf{H}}\boldsymbol{y}$ (see, e.g., [129]). After observing the error to behave even more like white noise within wavelet subbands, Millard et al. [100] proposed a VAMP modification that employs density compensation in the linear stage and wavelet thresholding in the denoising stage. The resulting "Variable-Density AMP" (VDAMP) algorithm was empirically observed to successfully track the error variance in each subband over the algorithm iterations. The authors then extended their work from single- to multicoil MRI in [102], calling their approach Parallel VDAMP (P-VDAMP).

To improve on VDAMP, Metzler and Wetzstein [101] proposed a PnP extension of the algorithm, where the wavelet-thresholding denoiser was replaced by a novel DNN that accepts a vector of subband error variances at each iteration. The resulting Denoising VDAMP (D-VDAMP) showed a significant boost in recovery accuracy over VDAMP for single-coil 2D point-sampled MRI [101]. Although D-VDAMP works relatively well, it requires early stopping for good performance (as we demonstrate in Section 2.3.5), which suggests that D-VDAMP has suboptimal fixed points and hence can be improved. Most recently, a "Denoising P-VDAMP" (DP-VDAMP) was proposed [103, 130] that replaces the wavelet thresholding step in P-VDAMP with a DNN denoiser. A major shortcoming of VDAMP, P-VDAMP, D-VDAMP, and DP-VDAMP is that they are designed around the use of 2D point sampling masks, which are impractical and uncommon in clinical MRI. These shortcomings motivate our proposed approach, which is described in the next section.

## 2.2 Proposed Approach

We now propose a new approach to MRI recovery that, like the VDAMP-based algorithms [100–103], formulates signal recovery in the wavelet domain, but, unlike the

VDAMP-based algorithms, does not use density compensation and does not require the use of 2D point masks. Our approach is based on a PnP version of the generalized EC algorithm, which is described in Section 2.2.1, in conjunction with a DNN denoiser that can handle parameterized colored noise, which is described in Section 2.2.2.

### 2.2.1 Wavelet-domain denoising GEC algorithm

To motivate wavelet-domain signal recovery, we first present an intuitive explanation of the problems faced by EC/VAMP with non-ROI $\boldsymbol{A}$. To start, one can show (see Appendix A) that EC/VAMP's denoiser input error $\boldsymbol{e}_2 \triangleq \boldsymbol{r}_2 - \boldsymbol{x}_{\mathsf{true}}$ can be written as

$$\boldsymbol{e}_2 = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{\mathsf{H}}\boldsymbol{e}_1 + \boldsymbol{u}, \tag{2.27}$$

where $\boldsymbol{V}$ is the right singular vector matrix of $\boldsymbol{A}$, the matrix $\boldsymbol{D}$ is diagonal with $\mathrm{tr}(\boldsymbol{D}) = 0$, $\boldsymbol{e}_1 \triangleq \boldsymbol{r}_1 - \boldsymbol{x}_{\mathsf{true}}$ is the error on the input to $\boldsymbol{f}_1$, and $\boldsymbol{u}$ is a linear transformation of the measurement noise vector $\boldsymbol{w}$ from (2.1). When $\boldsymbol{A}$ is ROI or RUI, $\boldsymbol{V}$ is drawn uniformly from the group of orthogonal or unitary matrices, respectively. Appendix B shows for the orthogonal case that, if $\boldsymbol{V}$ and $\boldsymbol{e}_1$ are treated as independent up to the fourth moment and $\boldsymbol{w}$ and $\boldsymbol{e}_1$ are uncorrelated, then, conditioned on $\boldsymbol{e}_1$, both $\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{\mathsf{H}}\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are asymptotically white and zero-mean Gaussian. Importantly, this behavior occurs despite the tendency for $\boldsymbol{e}_1$ to be highly structured and non-Gaussian.

When $\boldsymbol{A}$ is not a high-dimensional ROI or RUI matrix, however, there is no guarantee that $\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{\mathsf{H}}\boldsymbol{e}_1$ will asymptotically be white and zero-mean Gaussian. For example, when $\boldsymbol{A} = \boldsymbol{M}\boldsymbol{F}$ as in single-coil MRI and $\boldsymbol{x}_{\mathsf{true}}$ is a natural image, this desired property does not manifest because the $\boldsymbol{x}_{\mathsf{true}}$ (and thus $\boldsymbol{e}_1$) has a high concentration of energy at low frequencies and $\boldsymbol{V}^{\mathsf{H}} = \boldsymbol{F}$ focuses that error into a few dimensions of $\boldsymbol{D}$.

We now explain why using an AMP/EC algorithm to recover the wavelet coefficients $\boldsymbol{c}_{\mathsf{true}} \triangleq \boldsymbol{\Psi}\boldsymbol{x}_{\mathsf{true}}$, rather than the image pixels $\boldsymbol{x}_{\mathsf{true}}$, offers a path to circumvent these issues. For an orthogonal discrete wavelet transform (DWT) $\boldsymbol{\Psi}$, we have $\boldsymbol{x}_{\mathsf{true}} = \boldsymbol{\Psi}^\top \boldsymbol{c}_{\mathsf{true}}$ and so (2.1) implies the measurement model

$$\boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}_{\mathsf{true}} + \boldsymbol{w} \quad \text{with} \quad \boldsymbol{B} \triangleq \boldsymbol{A}\boldsymbol{\Psi}^\top. \tag{2.28}$$

In the case where $\boldsymbol{A}$ is a subsampled version of the Fourier matrix $\boldsymbol{F}$, the matrix $\boldsymbol{B}$ is a subsampled Fourier-wavelet matrix $\boldsymbol{F}\boldsymbol{\Psi}^\top$. The Fourier-wavelet matrix is known to be approximately block diagonal after appropriate row-sorting [131], where the blocks correspond to the wavelet subbands. This means that $\boldsymbol{B}$ in (2.28) primarily mixes the wavelet coefficients $\boldsymbol{c}_{\mathsf{true}}$ *within* subbands rather than *across* subbands. Consequently, if that mixing has a sufficiently randomizing effect on each subband of $\boldsymbol{e}_1$, then—with an appropriate EC-style algorithm design—the subband error vectors $\boldsymbol{e}_2$ can be kept approximately i.i.d. Gaussian across the iterations, although with a possibly different variance in each subband. In Fig. 2.2(a), we plot $\mathrm{abs}(\boldsymbol{F}\boldsymbol{\Psi}^\top)$ for the 2D case with the rows sorted according to the distance of their corresponding k-space sample to the origin. Although this row-sorting does not yield an approximately block-diagonal matrix, it should be clear from the discussion above that row-sorting is unimportant; it only matters that the columns of $\boldsymbol{B}$ for each given subband have a sufficiently randomizing effect on that subband and are approximately decoupled from the columns of other subbands. To illustrate the degree of column-decoupling in $\boldsymbol{F}\boldsymbol{\Psi}^\top$, we plot $\mathrm{abs}(\boldsymbol{F}\boldsymbol{\Psi}^\top)^\top \mathrm{abs}(\boldsymbol{F}\boldsymbol{\Psi}^\top)$ in Fig. 2.2(b). We plot this particular quantity because, if $\boldsymbol{F}\boldsymbol{\Psi}^\top = \boldsymbol{J}\boldsymbol{D}$ where $\boldsymbol{J}$ is a permutation matrix and $\boldsymbol{D}$ is a perfectly block-diagonal matrix, then $\mathrm{abs}(\boldsymbol{F}\boldsymbol{\Psi}^\top)^\top \mathrm{abs}(\boldsymbol{F}\boldsymbol{\Psi}^\top)$ will be perfectly block-diagonal for *any* $\boldsymbol{J}$, i.e.,

for any row-sorting. The fact that Fig. 2.2(b) looks approximately block-diagonal suggests that the column-blocks of $\boldsymbol{F}\boldsymbol{\Psi}^\top$ are significantly decoupled.

The discussion in the previous paragraph pertains to single-coil MRI. In the multi-coil case, the matrix $\boldsymbol{A}$ takes the form in (2.2) and so $\boldsymbol{B}$ from (2.28) manifests as

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{M} & & \\ & \ddots & \\ & & \boldsymbol{M} \end{bmatrix} \boldsymbol{G} \quad \text{with} \quad \boldsymbol{G} \triangleq \begin{bmatrix} \boldsymbol{F}\,\mathrm{Diag}(\boldsymbol{s}_1)\boldsymbol{\Psi}^\top \\ \vdots \\ \boldsymbol{F}\,\mathrm{Diag}(\boldsymbol{s}_C)\boldsymbol{\Psi}^\top \end{bmatrix}. \tag{2.29}$$

We would like that the multi-coil Fourier-wavelet matrix $\boldsymbol{G}$ has a sufficiently randomizing effect on each given subband in $\boldsymbol{c}_0$ and that the columns corresponding to that subband are decoupled from the columns of other subbands. To investigate the decoupling behavior of $\boldsymbol{G}$, we plot $\mathrm{abs}(\boldsymbol{G})^\top \mathrm{abs}(\boldsymbol{G})$ in Fig. 2.2(c) for the case of $C = 8$ ESPIRiT-estimated coils and notice that, similar to the single-coil quantity $\mathrm{abs}(\boldsymbol{F}\boldsymbol{\Psi}^\top)^\top \mathrm{abs}(\boldsymbol{F}\boldsymbol{\Psi}^\top)$ in Fig. 2.2(b), the multi-coil quantity $\mathrm{abs}(\boldsymbol{G})^\top \mathrm{abs}(\boldsymbol{G})$ looks approximately block-diagonal.

The first AMP-based method that exploited the aforementioned Fourier-wavelet properties was the VAMPire algorithm from [132], where a normalization of the subband energies in $\boldsymbol{c}_{\mathsf{true}}$ was used to equalize the subband error variances in $\boldsymbol{e}_2$, with the goal of tracking a single variance across the iterations (thus facilitating the use of D-VAMP). In other words, (2.28) was written as $\boldsymbol{y} = \overline{\boldsymbol{B}}\overline{\boldsymbol{c}}_{\mathsf{true}} + \boldsymbol{w}$ with $\overline{\boldsymbol{B}} = \boldsymbol{B}\,\mathrm{Diag}(\boldsymbol{g})$ and $\overline{\boldsymbol{c}}_{\mathsf{true}} = \mathrm{Diag}(\boldsymbol{g})^{-1}\boldsymbol{c}_{\mathsf{true}}$, for $\boldsymbol{g}$ such that $\mathrm{diag}(\mathrm{Cov}(\overline{\boldsymbol{c}}_{\mathsf{true}})) \approx \boldsymbol{1}$. But, because the variances of the subbands in $\boldsymbol{e}_2$ do change with the iterations, the scheme in [132] was far from optimal.

In this work, we propose an EC-based PnP method that recovers the wavelet coefficients $\boldsymbol{c}_{\mathsf{true}}$ and tracks the variances of both $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ in each wavelet subband. Our approach leverages the Generalized EC (GEC) framework from [28], which is

**Algorithm 2** Generalized EC (GEC)

**Require:** $\boldsymbol{f}_1(\cdot;\cdot)$, $\boldsymbol{f}_2(\cdot;\cdot)$, and $\text{gdiag}(\cdot)$.

Select initial $\boldsymbol{r}_1, \boldsymbol{\gamma}_1$
**repeat**
  // Measurement fidelity
  $\widehat{\boldsymbol{x}}_1 \leftarrow \boldsymbol{f}_1(\boldsymbol{r}_1, \boldsymbol{\gamma}_1)$
  $\boldsymbol{\eta}_1 \leftarrow \text{Diag}(\text{gdiag}(\nabla \boldsymbol{f}_1(\boldsymbol{r}_1, \boldsymbol{\gamma}_1)))^{-1}\boldsymbol{\gamma}_1$
  $\boldsymbol{\gamma}_2 \leftarrow \boldsymbol{\eta}_1 - \boldsymbol{\gamma}_1$
  $\boldsymbol{r}_2 \leftarrow \text{Diag}(\boldsymbol{\gamma}_2)^{-1}(\text{Diag}(\boldsymbol{\eta}_1)\widehat{\boldsymbol{x}}_1 - \text{Diag}(\boldsymbol{\gamma}_1)\boldsymbol{r}_1)$
  // Denoising
  $\widehat{\boldsymbol{x}}_2 \leftarrow \boldsymbol{f}_2(\boldsymbol{r}_2, \boldsymbol{\gamma}_2)$
  $\boldsymbol{\eta}_2 \leftarrow \text{Diag}(\text{gdiag}(\nabla \boldsymbol{f}_2(\boldsymbol{r}_2, \boldsymbol{\gamma}_2)))^{-1}\boldsymbol{\gamma}_2$
  $\boldsymbol{\gamma}_1 \leftarrow \boldsymbol{\eta}_2 - \boldsymbol{\gamma}_2$
  $\boldsymbol{r}_1 \leftarrow \text{Diag}(\boldsymbol{\gamma}_1)^{-1}(\text{Diag}(\boldsymbol{\eta}_2)\widehat{\boldsymbol{x}}_2 - \text{Diag}(\boldsymbol{\gamma}_2)\boldsymbol{r}_2)$
**until** Terminated
**return** $\widehat{\boldsymbol{x}}_2$

summarized in Alg. 2 and (2.30). GEC is a generalization of EC from Alg. 1 that averages the diagonal of the Jacobian $\nabla \boldsymbol{f}_i$ separately over $L$ coefficient subsets using the $\text{gdiag}\colon \mathbb{R}^{N\times N} \to \mathbb{R}^N$ operator:

$$\text{gdiag}(\boldsymbol{Q}) \;\triangleq\; \left[d_1 \mathbf{1}_{N_1}^\top, \ldots, d_L \mathbf{1}_{N_L}^\top\right]^\top \tag{2.30a}$$

$$d_\ell \;=\; \frac{\text{tr}\{\boldsymbol{Q}_{\ell\ell}\}}{N_\ell}. \tag{2.30b}$$

In (2.30), $N_\ell$ denotes the size of the $\ell$th subset and $\boldsymbol{Q}_{\ell\ell} \in \mathbb{R}^{N_\ell \times N_\ell}$ denotes the $\ell$th diagonal subblock of the matrix input $\boldsymbol{Q}$. When GEC is used to solve a convex optimization problem of the form (1.2), the functions $\boldsymbol{f}_i$ take the form

$$\boldsymbol{f}_i(\boldsymbol{r}, \boldsymbol{\gamma}) = \text{gprox}_{g_i,\boldsymbol{\gamma}}(\boldsymbol{r}) \tag{2.31a}$$

$$\text{gprox}_{\rho,\boldsymbol{\gamma}}(\boldsymbol{r}) \triangleq \arg\min_{\boldsymbol{x}} \left\{ \rho(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{r}\|_{\boldsymbol{\gamma}}^2 \right\}, \tag{2.31b}$$

where $\|\boldsymbol{q}\|_{\boldsymbol{\gamma}} \triangleq \sqrt{\boldsymbol{q}^{\mathsf{H}} \,\text{Diag}(\boldsymbol{\gamma})\boldsymbol{q}}$. When $L=1$, GEC reduces to EC/VAMP. In that case, $\boldsymbol{\gamma} = \gamma \mathbf{1}$ and $\text{gprox}_{\rho,\boldsymbol{\gamma}} = \text{prox}_{\gamma^{-1}\rho}$.

**Algorithm 3** Denoising GEC operating in the wavelet domain

**Require:** $\boldsymbol{f}_1(\cdot,\cdot)$, $\boldsymbol{f}_2(\cdot,\cdot)$, gdiag$(\cdot)$, and $\boldsymbol{\Psi}$.

1: Select initial $\boldsymbol{r}_1, \boldsymbol{\gamma}_1$
2: **repeat**
3: // Measurement fidelity
4: $\widehat{\boldsymbol{c}}_1 \leftarrow \boldsymbol{f}_1(\boldsymbol{r}_1, \boldsymbol{\gamma}_1)$
5: $\boldsymbol{\eta}_1 \leftarrow \mathrm{Diag}(\mathrm{gdiag}(\nabla \boldsymbol{f}_1(\boldsymbol{r}_1, \boldsymbol{\gamma}_1)))^{-1}\boldsymbol{\gamma}_1$
6: $\boldsymbol{\gamma}_2 \leftarrow \boldsymbol{\eta}_1 - \boldsymbol{\gamma}_1$
7: $\boldsymbol{r}_2 \leftarrow \mathrm{Diag}(\boldsymbol{\gamma}_2)^{-1}(\mathrm{Diag}(\boldsymbol{\eta}_1)\widehat{\boldsymbol{c}}_1 - \mathrm{Diag}(\boldsymbol{\gamma}_1)\boldsymbol{r}_1)$
8: // Denoising
9: $\widehat{\boldsymbol{c}}_2 \leftarrow \boldsymbol{\Psi}\boldsymbol{f}_2(\boldsymbol{\Psi}^\top \boldsymbol{r}_2, \boldsymbol{\gamma}_2)$
10: $\boldsymbol{\eta}_2 \leftarrow \mathrm{Diag}(\mathrm{gdiag}(\nabla \boldsymbol{f}_2(\boldsymbol{r}_2, \boldsymbol{\gamma}_2)))^{-1}\boldsymbol{\gamma}_2$
11: $\boldsymbol{\gamma}_1 \leftarrow \boldsymbol{\eta}_2 - \boldsymbol{\gamma}_2$
12: $\boldsymbol{r}_1 \leftarrow \mathrm{Diag}(\boldsymbol{\gamma}_1)^{-1}(\mathrm{Diag}(\boldsymbol{\eta}_2)\widehat{\boldsymbol{c}}_2 - \mathrm{Diag}(\boldsymbol{\gamma}_2)\boldsymbol{r}_2)$
13: **until** Terminated
14: **return** $\widehat{\boldsymbol{x}}_2 = \boldsymbol{\Psi}^\top \widehat{\boldsymbol{c}}_2$

Our proposed wavelet-domain Denoising GEC (D-GEC) approach is outlined in Alg. 3. For the gdiag operator, we use (2.30) with the diagonalization subsets defined by the $L = 3D + 1$ subbands of a depth-$D$ dyadic 2D orthogonal DWT. Also, when computing gdiag$(\nabla \boldsymbol{f}_1)$ and gdiag$(\nabla \boldsymbol{f}_2)$ in lines 5 and 10, we approximate the tr$\{\boldsymbol{Q}_{\ell\ell}\}$ terms in (2.30b) using the Monte Carlo approach [117]

$$\mathrm{tr}\{\boldsymbol{Q}_{\ell\ell}\} \;\approx\; \delta_\ell^{-1}\boldsymbol{q}_\ell^{\mathsf{H}}\left[\boldsymbol{f}_i(\boldsymbol{r}+\delta_\ell\boldsymbol{q}_\ell,\boldsymbol{\gamma}) - \boldsymbol{f}_i(\boldsymbol{r},\boldsymbol{\gamma})\right], \tag{2.32}$$

where we use i.i.d. unit-variance Gaussian coefficients for the $\ell$th coefficient subset in $\boldsymbol{q}_\ell$ and set all other coefficients in $\boldsymbol{q}_\ell$ to zero. As a result of the chosen diagonalization, the $\boldsymbol{\gamma}_i$ vectors (for $i = 1, 2$) are structured as

$$\boldsymbol{\gamma}_i = \left[\gamma_{i,1}\mathbf{1}_{N_1}^\top, \ldots, \gamma_{i,L}\mathbf{1}_{N_L}^\top\right]^\top, \tag{2.33}$$

and the $\boldsymbol{\eta}_i$ vectors have a similar structure. In (2.32) we used $\delta_\ell = \min\{\sqrt{1/\gamma_\ell}, \|\boldsymbol{r}_\ell\|_1/N_\ell\}$ where $\boldsymbol{r}_\ell$ denotes the $\ell$th coefficient subset of $\boldsymbol{r}$.

For the wavelet-measurement model (2.28) with WGN $\boldsymbol{w}$, (2.31) implies that the $\boldsymbol{f}_1$ estimation function in line 4 of Alg. 3 manifests as

$$\begin{aligned}&\boldsymbol{f}_1(\boldsymbol{r}_1, \boldsymbol{\gamma}_1)\\&= \big(\gamma_w \boldsymbol{B}^{\mathsf{H}}\boldsymbol{B} + \mathrm{Diag}(\boldsymbol{\gamma}_1)\big)^{-1}\big(\gamma_w \boldsymbol{B}^{\mathsf{H}}\boldsymbol{y} + \mathrm{Diag}(\boldsymbol{\gamma}_1)\boldsymbol{r}_1\big).\end{aligned} \tag{2.34}$$

When numerically solving (2.34), we exploit the fact that $\boldsymbol{B}$ is a fast operator by using the conjugate gradient (CG) method [133].

For $\boldsymbol{f}_2$ in line 9 of Alg. 3, we use a pixel-domain DNN denoiser. As shown in line 9, we convert from the wavelet domain to the pixel domain and back when calling this denoiser. Note that the denoiser $\boldsymbol{f}_2$ is provided with the vector $\boldsymbol{\gamma}_2$ of subband error precisions. The design of this denoiser will be discussed in Section 2.2.2. The experiments in Section 2.3.2 suggest that the denoiser input error $\boldsymbol{e}_2 = \boldsymbol{r}_2 - \boldsymbol{c}_{\mathsf{true}}$ does indeed obey

$$\boldsymbol{e}_2 \sim \mathcal{N}\big(\boldsymbol{0}, \mathrm{Diag}(\boldsymbol{\gamma}_2)^{-1}\big) \tag{2.35}$$

for the $\boldsymbol{\gamma}_2$ vector computed in line 6 of Alg. 3, similar to other AMP, VAMP, EC, and GEC algorithms. Further work is needed to understand if this behavior can be predicted by a rigorous analysis. The error model (2.35) facilitates a principled way to train the DNN denoiser, as we discuss in the next section.

We now discuss the initialization of D-GEC. For (2.35) to hold at all iterations, we need that the initial $\boldsymbol{\gamma}_1$ contains the precisions (i.e., inverse variances) of the subbands of the initial $\boldsymbol{e}_1 = \boldsymbol{r}_1 - \boldsymbol{c}_{\mathsf{true}}$. But initializing $\boldsymbol{\gamma}_1$ is complicated by the fact that $\boldsymbol{c}_{\mathsf{true}}$ is unknown. In response, we suggest initializing $\boldsymbol{\gamma}_1$ at an *average* value such as

$$\widehat{\boldsymbol{\gamma}}_1 = \mathrm{Diag}\big(\,\mathrm{gdiag}\big(\,\mathrm{E}\,\big\{(\boldsymbol{r}_1 - \boldsymbol{c}_{\mathsf{true}})(\boldsymbol{r}_1 - \boldsymbol{c}_{\mathsf{true}})^{\mathsf{H}}\big\}\big)\big)^{-1}\boldsymbol{1}, \tag{2.36}$$

where the expectation is approximated using a sample average over a training set (e.g., the dataset used to train the denoiser). But this approach could fail if the precision of the initial error falls far from $\widehat{\gamma}_1$, which can happen if $\boldsymbol{r}_1$ is strongly dependent on $\boldsymbol{y}$. Thus, we propose to initialize $\boldsymbol{r}_1 = \boldsymbol{B}^{\mathsf{H}}\boldsymbol{y} + \boldsymbol{n}$, where $\boldsymbol{n}$ is Gaussian and white in each subband. The per-subband variance of $\boldsymbol{n}$ should be large enough to dominate the behavior of $\boldsymbol{e}_1$, which makes the subband precisions easy to predict, but not so large that the algorithm is initialized at a terribly bad state. For the experiments in Section 2.3.2, we set the per-subband variance of $\boldsymbol{n}$ at 10 times the per-subband variance of $\boldsymbol{B}^{\mathsf{H}}\boldsymbol{y} - \boldsymbol{c}_{\mathsf{true}}$, and observed that (2.35) held at all iterations. Although a careful choice of initialization is important for (2.35) to hold at all iterations, we find that the initialization has little effect on the fixed points of D-GEC. So, for the experiments in Sections 2.3.3, 2.3.4, and 2.3.5, we set $\boldsymbol{n} = \boldsymbol{0}$ to improve the accuracy of the initial $\boldsymbol{r}_1$ and thus speed D-GEC convergence.

Computationally, the cost of D-GEC is driven by lines 4-5 and 9-10 of Alg. 3, which call $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$, respectively, $L+1$ times when implementing (2.32). The $L+1$ calls to $\boldsymbol{f}_1$ can be performed in parallel (e.g., in a single minibatch on a GPU), as can the calls to $\boldsymbol{f}_2$. As described above, each call to $\boldsymbol{f}_1$ involves running several iterations of CG. For accurate D-GEC fixed points, we find that 10 CG iterations suffice, and we use this setting in Sections 2.3.3, 2.3.4, and 2.3.5. For D-GEC error to match the state-evolution predictions at all iterations, we find that 150 CG iterations suffice, and we use this value in Section 2.3.2. Each call to $\boldsymbol{f}_2$ involves calling the DNN denoiser that is described in the next subsection.

### 2.2.2 A DNN denoiser for correlated noise

As suggested by (2.35), the denoiser $\boldsymbol{f}_2$ in Alg. 3 faces the task of denoising the pixel-domain signal $\boldsymbol{\Psi}^\top \boldsymbol{r}_2$, where $\boldsymbol{r}_2 = \boldsymbol{c}_{\mathsf{true}} + \boldsymbol{n}$ for $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \mathrm{Diag}(\boldsymbol{\gamma}_2)^{-1})$ and $\boldsymbol{c}_{\mathsf{true}}$ are the wavelet coefficients of the true image $\boldsymbol{x}_{\mathsf{true}}$. The denoiser input can thus be modeled as

$$\boldsymbol{\Psi}^\top \boldsymbol{r}_2 = \boldsymbol{x}_{\mathsf{true}} + \boldsymbol{n} \text{ for } \boldsymbol{n} \sim \mathcal{N}\big(\boldsymbol{0}, \boldsymbol{\Psi}^\top \mathrm{Diag}(\boldsymbol{\gamma}_2)^{-1} \boldsymbol{\Psi}\big), \tag{2.37}$$

i.e., the true image corrupted by colored Gaussian noise with (known) covariance matrix $\boldsymbol{\Psi}^\top \mathrm{Diag}(\boldsymbol{\gamma}_2)^{-1} \boldsymbol{\Psi}$. Here, the $\boldsymbol{\gamma}_2$ vector takes the form shown in (2.33).

Although several DNNs have been proposed to tackle denoising with correlated noise (e.g., [134–136]), to our knowledge, the only one compatible with our denoising task is the DNN proposed by Metzler and Wetzstein in [101]. There, they built on the DnCNN network by providing every layer with $L$ additional channels, where the $\ell$th channel contains the standard deviation (SD) of the noise in the $\ell$th wavelet subband (i.e., $\sqrt{1/\gamma_{2,\ell}}$). Their approach can be interpreted as an extension of FFDNet [137], which provides one additional channel containing the SD of the assumed white corrupting noise, to multiple additional channels containing subband SDs. In our numerical experiments in Section 2.3, we find that Metzler's denoising approach works well in some cases but poorly in others. We believe that the observed poor performance may be the result of the fact that their DNN operates in the pixel domain, while their SD side information is given in the wavelet domain and the network is given no information about the wavelet transform $\boldsymbol{\Psi}$.

We now propose a novel approach to DNN denoising that can handle colored Gaussian noise with an arbitrary known covariance matrix. Our approach starts with

an arbitrary DNN denoiser (e.g., DnCNN [27], UNet [138], RNN [139], etc.) that normally accepts $C$ input channels (e.g., 3 channels for color-image denoising or 2 channels for complex-image denoising). It then adds $K \geq 1$ sets of $C$ additional channels, where each set is fed an independently generated realization of noise with the same statistics as that corrupting the signal to be denoised. In other words, if $\boldsymbol{u} \in \mathbb{R}^{CN}$ denotes the (vectorized) noisy input signal, which obeys (recall (2.37))

$$\boldsymbol{u} = \boldsymbol{x}_{\mathsf{true}} + \boldsymbol{n} \text{ for } \boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) \tag{2.38}$$

with arbitrary known $\boldsymbol{\Sigma}$, then the (vectorized) input to the $k$th additional channel-set would be

$$\boldsymbol{n}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) \;\; \forall k = 1, \dots, K, \tag{2.39}$$

where $\{\boldsymbol{n}_k\}_{k=1}^K$ are mutually independent and independent of $\boldsymbol{u}$. The hope is that, during training, the denoiser learns how to i) extract the relevant statistics from $\{\boldsymbol{n}_k\}_{k=1}^K$ and ii) use them productively for the denoising of $\boldsymbol{u}$. Here, $K$ is a design parameter; for our D-GEC application we find that $K = 1$ suffices. Because the denoiser accepts a signal corrupted by correlated noise plus additional realizations of correlated noise, we call our approach "corr+corr."

To train our corr+corr denoiser, we use the following approach. Suppose that we have access to a training set of clean signals $\{\boldsymbol{x}_i\}$, and that we would like to train the denoiser to handle $\boldsymbol{\gamma}_2$ vectors from some distribution $p_\Gamma$. During training, we draw many $\boldsymbol{\gamma}_2 \sim p_\Gamma$ and, for each realization of $\boldsymbol{\gamma}_2$, we draw independent realizations of $\boldsymbol{v}$ and $\{\boldsymbol{n}_k\}_{k=1}^N$ from the distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Psi}^\top \operatorname{Diag}(\boldsymbol{\gamma}_2)^{-1} \boldsymbol{\Psi})$. The $\boldsymbol{v}$ vector is then used to form the noisy signal $\boldsymbol{u}_i = \boldsymbol{x}_i + \boldsymbol{v}$ and the denoiser is given access to $\boldsymbol{N} \triangleq [\boldsymbol{n}_1, \dots, \boldsymbol{n}_K]$ when denoising $\boldsymbol{u}_i$. Concretely, if we denote the corr+corr denoiser

as $\boldsymbol{f}_2(\boldsymbol{u}_i, \boldsymbol{N}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ contains the trainable denoiser parameters, then we train those parameters using

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_i E\big\{\mathcal{L}\big(\boldsymbol{x}_i, \boldsymbol{f}_2(\boldsymbol{x}_i + \boldsymbol{v}, \boldsymbol{N}; \boldsymbol{\theta})\big)\big\}, \tag{2.40}$$

where $\mathcal{L}(\cdot, \cdot)$ is a loss function that quantifies the error between its two vector-valued arguments. Popular losses include [140] $\boldsymbol{\ell}_2$, $\boldsymbol{\ell}_1$, SSIM [141], or combinations thereof, and in our experiments we used $\ell_2$ loss. The expectation in (2.40) is taken over both $\boldsymbol{v}$ and $\boldsymbol{N}$, which implicitly involves $p_\Gamma$.

In inference mode, we are given a noisy $\boldsymbol{u}$ and a single precision vector $\boldsymbol{\gamma}_2$. From the latter, we generate a single independent realization of $\boldsymbol{N} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Psi}^\top \operatorname{Diag}(\boldsymbol{\gamma}_2)^{-1} \boldsymbol{\Psi})$ and then compute the denoised pixel-domain image estimate via $\widehat{\boldsymbol{x}}_2 = \boldsymbol{f}_2(\boldsymbol{u}, \boldsymbol{N}; \widehat{\boldsymbol{\theta}})$.

In Section 2.3.1 we show that our corr+corr denoiser performs better than Metzler's DnCNN and nearly as well as a genie-aided denoiser that knows the distribution of the test noise $\boldsymbol{v} \sim \boldsymbol{\Psi}^\top \operatorname{Diag}(\boldsymbol{\gamma}_2)^{-1} \boldsymbol{\Psi}$, with fixed $\boldsymbol{\gamma}_2$, at training time.

## 2.3 Numerical Experiments

In this section, we present numerical experiments demonstrating the performance of the proposed corr+corr denoiser as well as the proposed D-GEC method applied to both single-coil and multicoil MRI recovery.

### 2.3.1 Denoising experiments

In this subsection, we compare the corr+corr denoiser proposed in Section 2.2.2 to several existing denoisers. We test all denoisers on the 10 MRI images from the Stanford 2D FSE dataset [31] shown in Fig. 2.3, which ranged in size from $320 \times 320$ to $416 \times 416$. Noisy images were obtained by corrupting those test images by additive

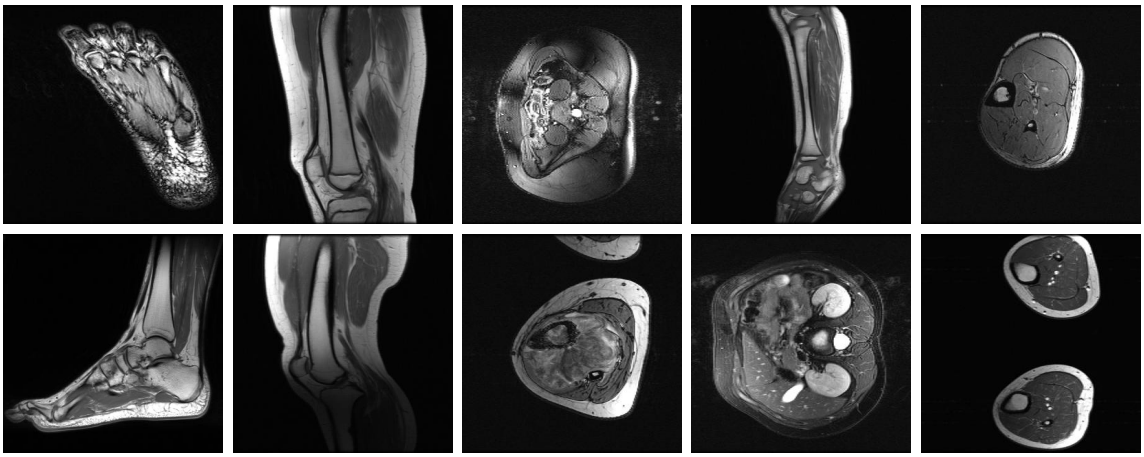

Figure 2.3: Test images from the Stanford 2D FSE MRI dataset.

zero-mean Gaussian noise of covariance

$$\boldsymbol{\Sigma} = \boldsymbol{\Psi}^\top \operatorname{Diag}(\boldsymbol{\gamma})^{-1} \boldsymbol{\Psi}, \tag{2.41}$$

with $\boldsymbol{\Psi}$ a 2D Haar wavelet transform of depth $D = 1$. This wavelet transform has $L = 4$ subbands, and so the precision vector $\boldsymbol{\gamma}$ in (2.41) is structured as $\boldsymbol{\gamma} = [\gamma_1 \mathbf{1}_{N/4}^\top, \ldots, \gamma_4 \mathbf{1}_{N/4}^\top]^\top$ and thus parameterized by the four precisions $[\gamma_1, \gamma_2, \gamma_3, \gamma_4]$, or equivalently the four SDs $\left[\frac{1}{\sqrt{\gamma_1}}, \frac{1}{\sqrt{\gamma_2}}, \frac{1}{\sqrt{\gamma_3}}, \frac{1}{\sqrt{\gamma_4}}\right]$. We test the denoisers under different assumptions on these SDs, as indicated by the rows in Table 2.1. For some tests, we use a fixed SD vector, while for other tests we average over a distribution of SD vectors.

When training the denoisers, we used the 70 training MRI images from the Stanford 2D FSE dataset. We trained to minimize $\ell_2$ loss on a total of 44 000 patches of size $40 \times 40$ taken with stride $10 \times 10$. All denoisers used the bias-free version of DnCNN from [142], with the exception of Metzler's DnCNN from [101], which used the publicly available code provided by the author. For both corr+corr and Metzler's DnCNN,

Table 2.1: Performance comparison of four different DnCNN denoisers for various cases of colored noise

| test standard deviations $\left[\frac{1}{\sqrt{\gamma_1}}, \frac{1}{\sqrt{\gamma_2}}, \frac{1}{\sqrt{\gamma_3}}, \frac{1}{\sqrt{\gamma_4}}\right]$ | white DnCNN | | Metzler's DnCNN | | corr+corr DnCNN | | genie DnCNN | |
|---|---|---|---|---|---|---|---|---|
| | PSNR ± SE | SSIM ± SE | PSNR ± SE | SSIM ± SE | PSNR ± SE | SSIM ± SE | PSNR ± SE | SSIM ± SE |
| $\left[\frac{48}{255}, \frac{47}{255}, \frac{9}{255}, \frac{19}{255}\right]$ | 25.36 ± 0.02 | 0.7328 ± 0.0013 | 31.23 ± 0.03 | 0.8783 ± 0.0006 | 31.69 ± 0.03 | 0.8899 ± 0.0005 | 32.12 ± 0.04 | 0.9012 ± 0.0005 |
| $\left[\frac{10}{255}, \frac{40}{255}, \frac{23}{255}, \frac{14}{255}\right]$ | 32.44 ± 0.03 | 0.9044 ± 0.0006 | 34.87 ± 0.04 | 0.9363 ± 0.0004 | 35.24 ± 0.04 | 0.9407 ± 0.0004 | 35.54 ± 0.04 | 0.9449 ± 0.0004 |
| $\left[\frac{13}{255}, \frac{7}{255}, \frac{8}{255}, \frac{10}{255}\right]$ | 36.50 ± 0.03 | 0.9421 ± 0.0003 | 31.03 ± 0.03 | 0.9359 ± 0.0003 | 37.02 ± 0.03 | 0.9535 ± 0.0003 | 37.41 ± 0.03 | 0.9569 ± 0.0003 |
| $\left[\frac{10}{255}, \frac{10}{255}, \frac{10}{255}, \frac{10}{255}\right]$ | 37.41 ± 0.03 | 0.9571 ± 0.0003 | 31.94 ± 0.02 | 0.9413 ± 0.0003 | 37.31 ± 0.03 | 0.9559 ± 0.0003 | 37.63 ± 0.03 | 0.9586 ± 0.0003 |
| $\left[0\text{-}\frac{50}{255}, 0\text{-}\frac{50}{255}, 0\text{-}\frac{50}{255}, 0\text{-}\frac{50}{255}\right]$ | 31.07 ± 0.05 | 0.8597 ± 0.0013 | 33.24 ± 0.05 | 0.9132 ± 0.0006 | 34.08 ± 0.05 | 0.9213 ± 0.0006 | n/a | n/a |

when training, we used random subband SDs $\{1/\sqrt{\gamma_\ell}\}_{\ell=1}^4$ drawn independently from a uniform distribution over the interval $[0, 50/255]$. When interpreting the value "50/255," note that the image pixel values were in $[0, 1]$ for this dataset. As a baseline method, we trained bias-free DnCNN using white noise with a standard deviation distributed uniformly over the interval $[0, 50/255]$. We expect this "white DnCNN" to perform poorly with colored testing noise. As an upper bound on performance, we trained bias-free DnCNN using the same fixed value of the SD vector $\left[\frac{1}{\sqrt{\gamma_1}}, \frac{1}{\sqrt{\gamma_2}}, \frac{1}{\sqrt{\gamma_3}}, \frac{1}{\sqrt{\gamma_4}}\right]$ that is used when testing. The resulting "genie DnCNN" is specialized to that particular SD vector, and thus not useful in practical situations where the test SD is unknown during training (e.g., in D-GEC).

The results of our denoiser comparison are presented in Table 2.1 using the metrics of PSNR and SSIM [141] along with the respective standard errors (SE). In the first four rows of the table, performance is evaluated for a fixed value of the SD vector $\left[\frac{1}{\sqrt{\gamma_1}}, \frac{1}{\sqrt{\gamma_2}}, \frac{1}{\sqrt{\gamma_3}}, \frac{1}{\sqrt{\gamma_4}}\right]$, while in the last row the results are averaged over subband SDs $\{1/\sqrt{\gamma_\ell}\}_{\ell=1}^4$ drawn independently from a uniform distribution over the interval $[0, 50/255]$. The fourth row corresponds to white Gaussian noise with a fixed standard deviation of 10, while all other rows correspond to colored noise. The fifth row corresponds to noise that is non-Gaussian in general, but Gaussian when conditioned

on $\boldsymbol{\gamma}$. All results in the table represent the average over 500 different noise realizations. The results in Table 2.1 are summarized as follows.

- As expected, white DnCNN performs relatively poorly for all test cases except that in the fourth row, where the testing noise was white, and that in the third row, where the testing noise was lightly colored. In the fourth row, white DnCNN performs slightly worse than genie DnCNN, which is expected because white DnCNN was trained using white noise with SDs in the range $[0, 50/255]$, while genie DnCNN was trained using a white noise with a fixed SD that exactly matches the test noise.

- As expected, genie DnCNN is the best method in the first four rows. In all of those cases, genie DnCNN is specialized to handle exactly the noise distribution used for the test, and thus is impractical. By definition, genie DnCNN is not applicable to the fifth row.

- Metzler's DnCNN performs relatively well in the first two rows, but relatively poorly in the second two rows. We believe that the inconsistency is the result of the fact that the DNN operates in the pixel domain, while the SD side information is given in the wavelet domain and the DNN is given no information about the wavelet transform itself.

- The proposed corr+corr outperforms Metzler's DnCNN in all cases and is only 0.3 to 0.5 dB away from the genie DnCNN. This is notable because genie DnCNN gives an (impractical) upper bound on the performance achievable with the chosen architecture and training method.

Code for our corr+corr experiments can be found at https://github.com/Saurav-K-Shastri/corr-plus-corr.

### 2.3.2 Example D-GEC behavior in multicoil MRI with a 2D line mask

In this section, we demonstrate the typical behavior of D-GEC when applied to multicoil MRI image recovery with a 2D line mask; experiments with a 2D point mask will be presented in Section 2.3.3. The full details of our multicoil experimental setup are given in Appendix C-A. One of our main goals is to demonstrate that D-GEC's denoiser input error behaves as in (2.35), i.e., that the error in each wavelet band is white and Gaussian with a predictable variance. For the experiments in this section, we used the corr+corr denoiser proposed in Section 2.2.2, a signal-to-noise ratio (SNR) of 40 dB, and an acceleration of $R = 4$. Code for our D-GEC experiments can be found at https://github.com/Saurav-K-Shastri/D-GEC.

Before discussing our results, there is one peculiarity to multicoil MRI that should be explained. In practice, both the coil-sensitivity maps $\{\boldsymbol{s}_c\}_{c=1}^C$ in $\boldsymbol{A}$ from (2.2) and the image $\boldsymbol{x}_{\mathsf{true}}$ in (2.1) are unknown. The standard recovery approach is to first use an algorithm like ESPIRiT [143] to estimate the coil maps $\{\boldsymbol{s}_c\}_{c=1}^C$, then plug the estimated maps into the $\boldsymbol{A}$ matrix, and finally solve the inverse problem with the estimated $\boldsymbol{A}$ to recover $\boldsymbol{x}_{\mathsf{true}}$. One complication with ESPIRiT is that, in pixel regions where the true image $\boldsymbol{x}_{\mathsf{true}}$ is zero or nearly zero (e.g., the outer regions of many MRI images), the ESPIRiT-estimated coil maps can be uniformly zero-valued, depending on how ESPIRiT is configured. In other words, there may exist pixels $n$ such that $[\boldsymbol{s}_c]_n = 0 \; \forall c = 1 \dots C$, which causes the corresponding columns of $\boldsymbol{A}$ to be zero. In our experiments, we use the default ESPIRiT parameters from the

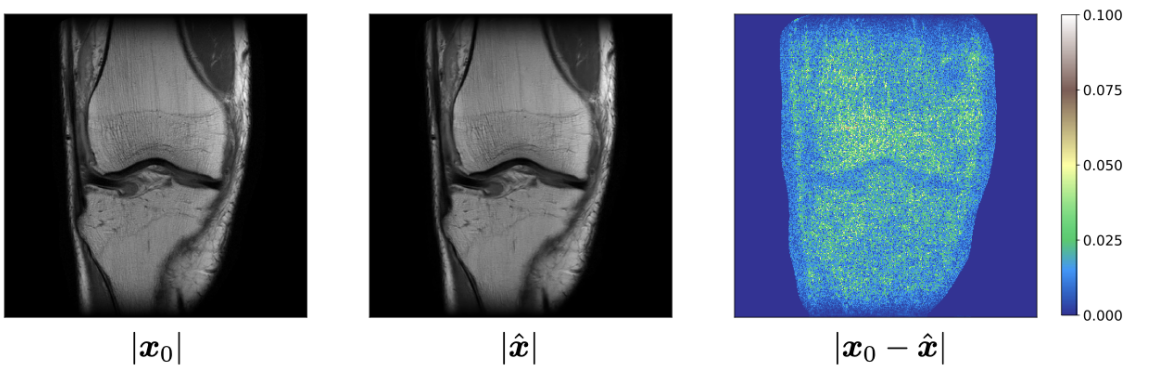


Figure 2.4: Example multicoil knee image recovery: True image magnitude $|\boldsymbol{x}_{\mathsf{true}}|$, D-GEC's recovered image magnitude $|\widehat{\boldsymbol{x}}|$ at iteration 20, and the error magnitude $|\boldsymbol{x}_{\mathsf{true}} - \widehat{\boldsymbol{x}}|$, for $R = 4$ and measurement SNR $= 40$ dB.

SigPy implementation[3] and find such zero-valued regions do occur. Although the presence of zero-valued columns in $\boldsymbol{A}$ might appear to make the inverse problem (2.1) more difficult, the (known) coil-map estimates can be exploited as side-information to tell the algorithm which pixels in $\boldsymbol{x}_{\mathsf{true}}$ are nearly zero-valued. *Consequently, in our multicoil experiments, for all algorithms, we set those pixels of the recovered image $\widehat{\boldsymbol{x}}$ to zero wherever the estimated coil maps are uniformly zero.* In the sequel, we will refer to the pixel region with zero-valued coil map estimates as the "zero-coil region."

For a typical MRI knee image, Fig. 2.4 shows the magnitude $|\boldsymbol{x}_{\mathsf{true}}|$ of the true image, D-GEC's recovery $|\widehat{\boldsymbol{x}}|$ after 20 iterations, and the error magnitude $|\widehat{\boldsymbol{x}} - \boldsymbol{x}_{\mathsf{true}}|$. The error is exactly zero in the previously defined zero-coil region because both $\boldsymbol{x}_{\mathsf{true}}$ and $\widehat{\boldsymbol{x}}$ are zero-valued there. The PSNR $\triangleq 10\log_{10}[(N \max_n |[\boldsymbol{x}_{\mathsf{true}}]_n|^2)/\|\widehat{\boldsymbol{x}} - \boldsymbol{x}_{\mathsf{true}}\|^2]$ and SSIM [141] values for this example reconstruction were 36.87 dB and 0.9397, respectively.

[3]https://sigpy.readthedocs.io/en/latest/generated/sigpy.mri.app.EspiritCalib.html.

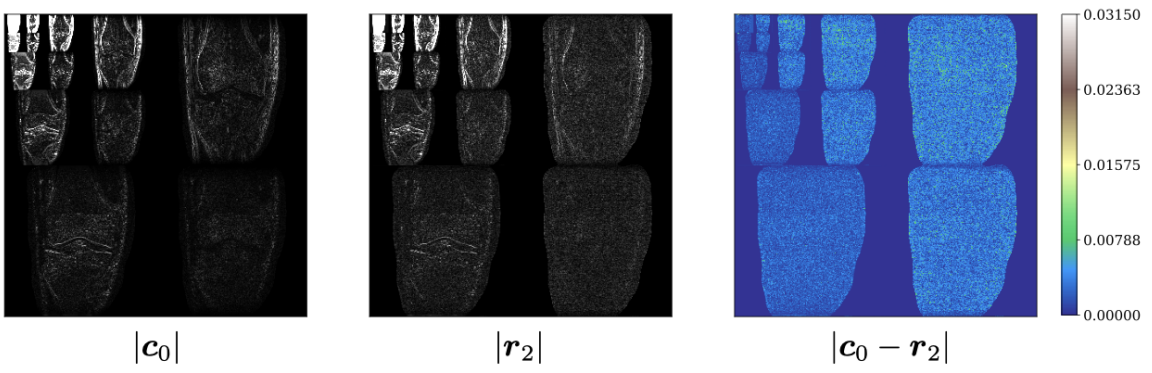


Figure 2.5: Example multicoil knee image recovery: True wavelet coefficient magnitude $|\boldsymbol{c}_0|$, D-GEC's denoiser-input magnitude $|\boldsymbol{r}_2|$ at iteration 10, and the error magnitude $|\boldsymbol{c_0} - \boldsymbol{r}_2|$, for $R = 4$ and measurement SNR $= 40$ dB.

Fig. 2.5 shows the magnitude $|\boldsymbol{c}_{\mathsf{true}}|$ of the corresponding true wavelet coefficients, the magnitude $|\boldsymbol{r}_2|$ of the noisy signal entering the D-GEC denoiser at iteration 10, and the error magnitude $|\boldsymbol{r}_2 - \boldsymbol{c}_{\mathsf{true}}|$. The wavelet subbands are visible as the image tiles in these plots. Here again, we see zero-valued error in the zero-coil region. As anticipated from (2.35), the error maps look like white noise outside the zero-coil region of each wavelet subband, with an error variance that varies across subbands.

To verify the Gaussianity of the wavelet subband errors, Fig. 2.6 shows quantile-quantile (QQ) plots of the real and imaginary parts of the error $\boldsymbol{c}_{\mathsf{true}} - \boldsymbol{r}_2$ outside the zero-coil region of several wavelet subbands at iteration 1, and Fig. 2.7 shows the same at iteration 10. These QQ-plots suggest that the subband errors are indeed Gaussian at all iterations.

To show that the subband precisions $\boldsymbol{\gamma}_2$ predicted by D-GEC match the empirical subband precisions in the error vector $\boldsymbol{e}_2$, Fig. 2.8 plots the $\ell$th subband SD $1/\sqrt{\gamma_\ell}$ versus iteration, along with the SDs empirically estimated from $\boldsymbol{c}_{\mathsf{true}} - \boldsymbol{r}_2$, for several

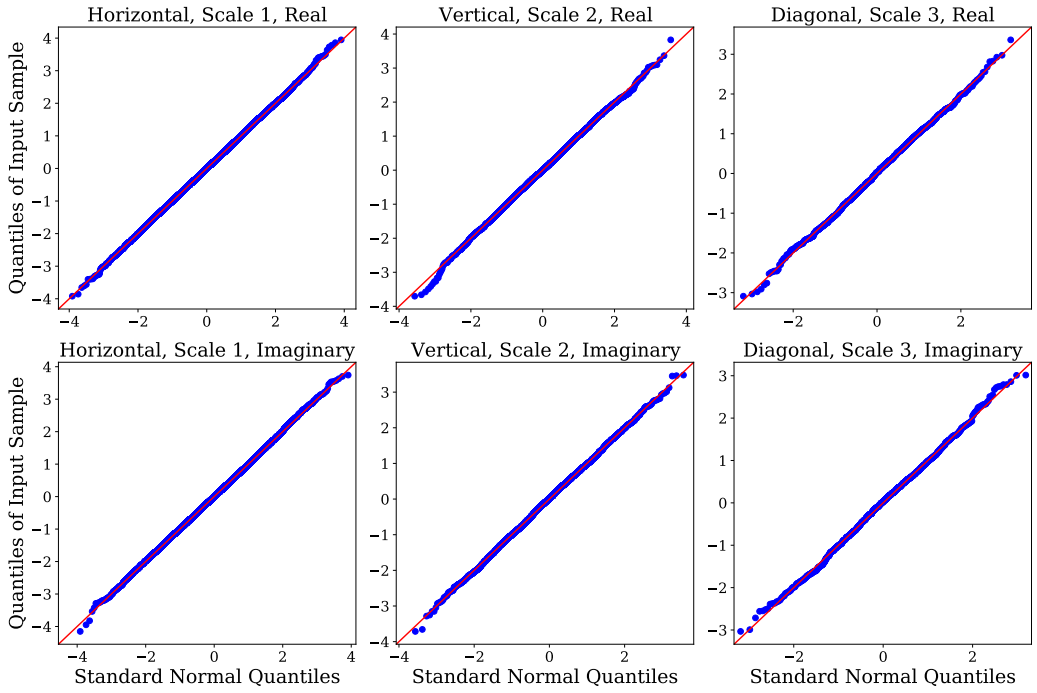


Figure 2.6: QQ-plots of the real and imaginary parts of D-GEC's subband errors $\boldsymbol{c}_{\mathsf{true}} - \boldsymbol{r}_2$ at iteration 1.

subbands $\ell$ and a typical run of the algorithm. It can be seen that the predicted SDs are in close agreement with the empirically estimated SDs.

Finally, to verify that the errors $\boldsymbol{c}_{\mathsf{true}} - \boldsymbol{r}_2$ are zero-mean in each subband of each validation image, we performed a t-test [144] using a significance level of $\alpha = 0.05$ (i.e., if the errors were truly zero mean then the test would fail with probability $\alpha$). At the first iteration, we ran a total of 208 tests (one for each of the 13 subbands in each of the 16 knee validation images at $R = 4$ and SNR $= 40$ dB) and found that 11 tests rejected the zero-mean hypothesis, which is consistent with $\alpha = 0.05$ since $11/208 = 0.0529 \approx 0.05$. At the 10th iteration, 12 tests rejected the zero-mean hypothesis, which is again consistent with $\alpha = 0.05$.

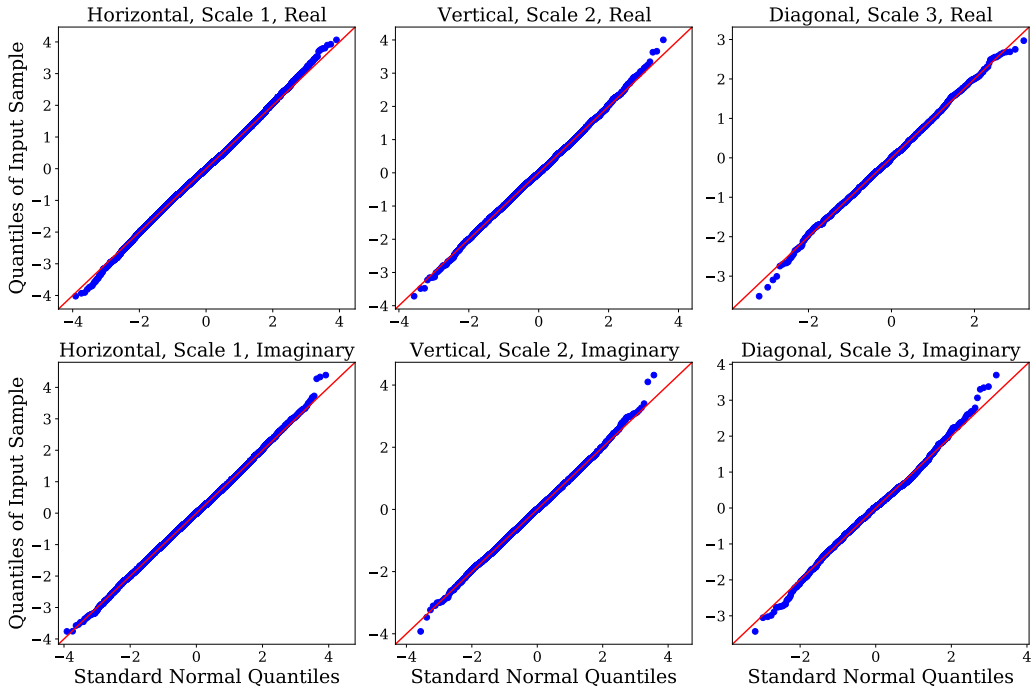


Figure 2.7: QQ-plots of the real and imaginary parts of D-GEC's subband errors $\boldsymbol{c}_{\mathsf{true}} - \boldsymbol{r}_2$ at iteration 10.

### 2.3.3 Multicoil MRI algorithm comparison with a 2D point mask

In this section, we compare the performance of D-GEC to two state-of-the-art algorithms for multicoil MRI image recovery: P-VDAMP [102] and PnP-PDS [108]. We use 2D point masks in this section out of fairness to P-VDAMP, which was designed around 2D point masks. Multicoil experiments with 2D line masks are presented in Section 2.3.4, and single-coil experiments are presented in Section 2.3.5. We examine two acceleration rates, $R = 4$ and $R = 8$, and several measurement SNRs between 20 and 45 dB. As before, we quantify recovery performance using PSNR and SSIM.

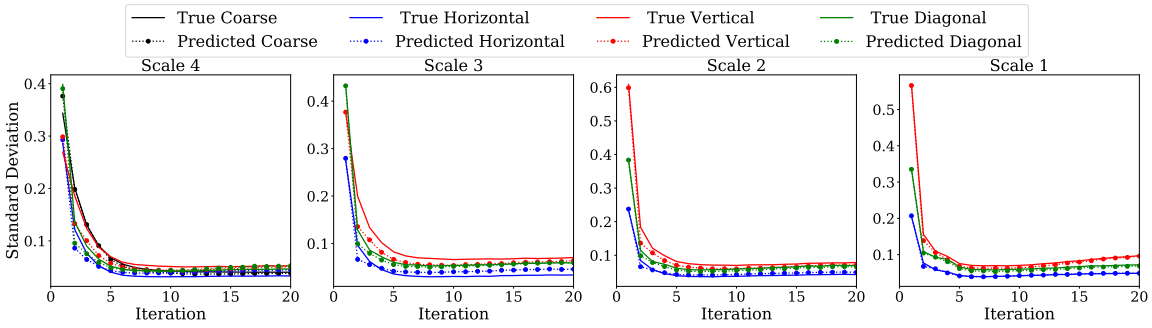


Figure 2.8: Evolution of D-GEC's predicted subband SDs ($1/\sqrt{\gamma_\ell}$) and empirically estimated subband SDs (from $\boldsymbol{c}_{\mathsf{true}} - \boldsymbol{r}_2$) for several subbands $\ell$ over 20 iterations.

For this section, we used both knee and brain fastMRI data. The details of the experimental setup are given in Appendix C-A.

For P-VDAMP, we ran the authors' code from [102] under its default settings. For PnP-PDS, we used a bias-free DnCNN [142] denoiser trained to minimize $\ell_2$ loss when removing WGN with an SD uniformly distributed in the interval $[0, 55/255]$. This bias-free network is known to perform very well over a wide SD range, and so there is no advantage in training multiple denoisers over different SNR ranges [142]. Because PnP-PDS performance strongly depends on the chosen penalty parameter and number of PDS iterations, we separately tuned these parameters for every combination of measurement SNR and acceleration rate to maximize PSNR on the training set. For D-GEC, we used a Haar wavelet transform of depth $D = 4$, which yields $L = 13$ subbands, and a corr+corr bias-free DnCNN denoiser; see Appendix C-A for additional details. For all algorithms, we set the image estimate to zero in the zero-coil region.

For each acceleration rate $R$ and SNR under test, we ran all three algorithms on all images in the brain and knee testing sets. We then computed the average PSNR and SSIM values across those images and summarized the results in Fig. 2.9, using

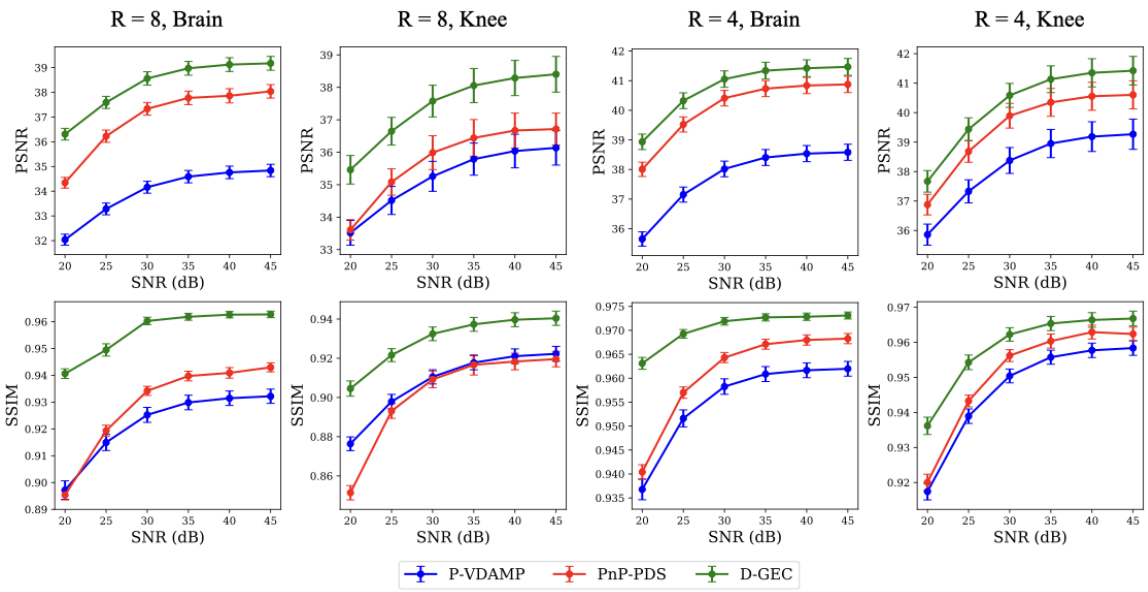


Figure 2.9: Average PSNR and SSIM versus measurement SNR for P-VDAMP, PnP-PDS, and D-GEC.

error bars to show plus/minus one standard error. The figure shows that D-GEC significantly outperformed the other algorithms in all metrics at all combinations of $R$ and measurement SNR.

Figure 2.10 shows image recoveries and error images for a typical fastMRI brain image at acceleration $R = 4$ and measurement SNR $= 35$ dB. In this case, D-GEC outperformed the P-VDAMP and PnP-PDS algorithms in PSNR by 2.6 and 0.76 dB, respectively. Furthermore, D-GEC's error image looks the least structured. Looking at the details of the zoomed plots, we see that D-GEC is able to reconstruct certain fine details better than its competitors.

Figure 2.11 shows PSNR versus iteration for the three algorithms at $R = 4$ and SNR $= 20$ dB. The PSNR values shown are the average over all 16 test images from the brain MRI dataset. The plot shows P-VDAMP, D-GEC, and PnP-PDS taking

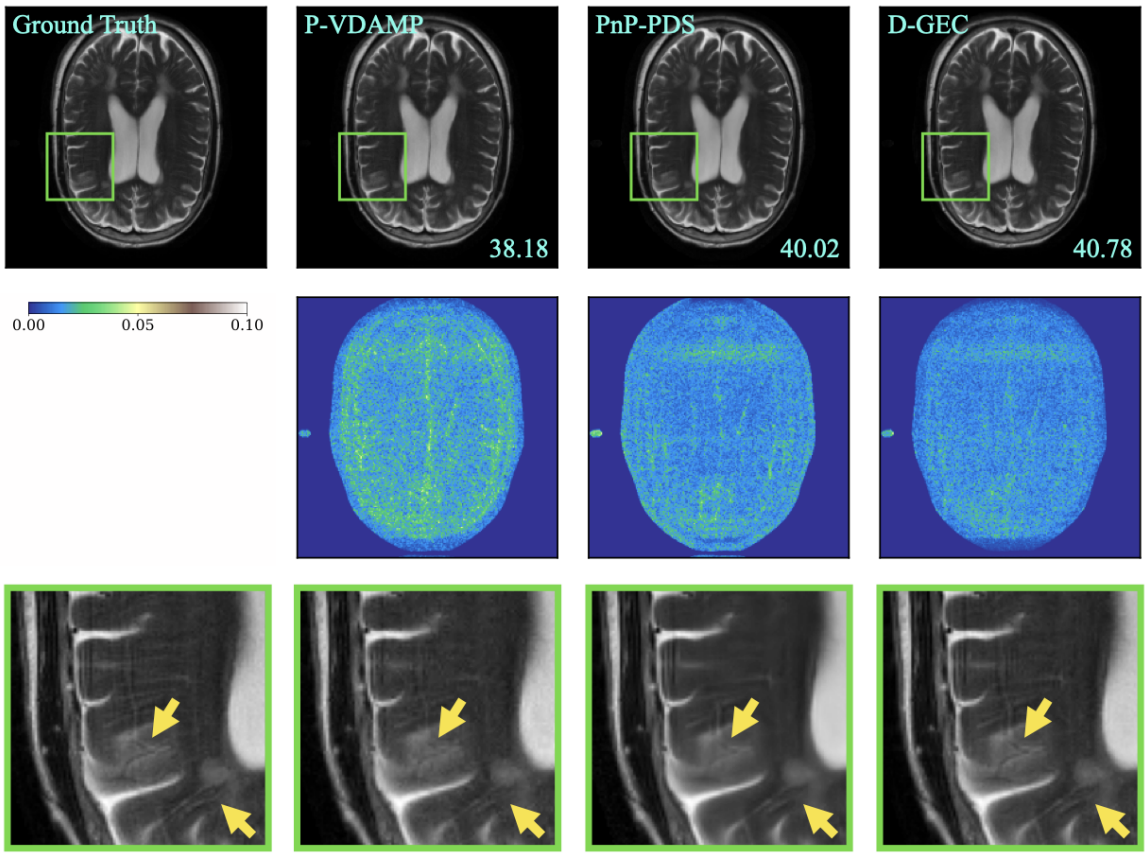


Figure 2.10: Example multicoil MRI brain recoveries and error images at $R = 4$ and SNR $= 35$ dB. The number printed on each recovered image shows its PSNR. The bottom row is a zoomed in version of the green square in the top row. This figure is best viewed in electronic form.

about 7, 8, and 25 iterations to converge, respectively. If we measure the number of iterations taken to reach 35 dB SNR, then D-GEC, PnP-PDS, and P-VDAMP take about 3, 5, and 7 iterations, respectively.

### 2.3.4 Multicoil MRI algorithm comparison with a 2D line mask

In this section, we compare the performance of D-GEC to that of P-VDAMP [102] and PnP-PDS [108] when using a 2D line mask. We examine acceleration rates $R = 4$

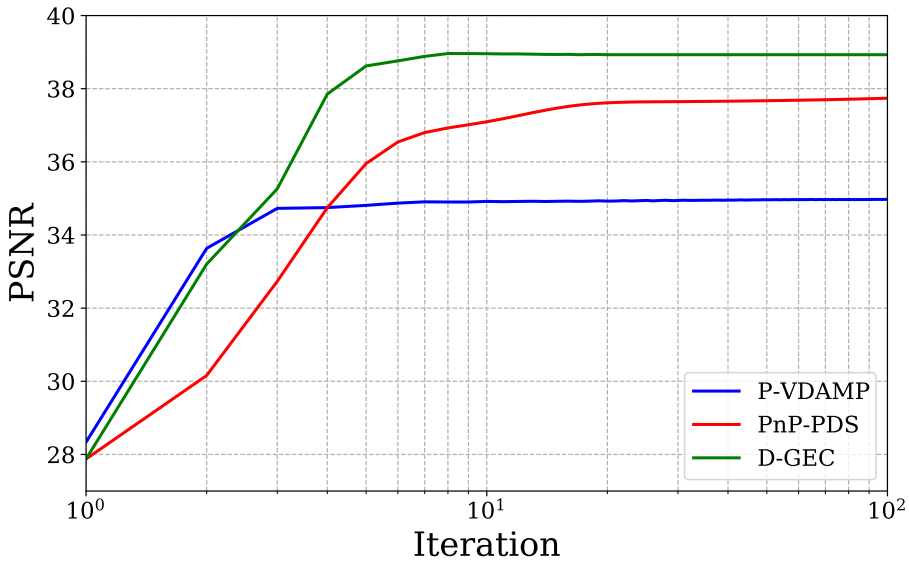


Figure 2.11: PSNR versus iterations for multicoil brain MRI recovery at $R = 4$ and SNR = 20 dB. PSNR was averaged over the 16 test images.

Table 2.2: Multicoil 2D line-mask results at SNR = 40 dB averaged over all test images.

| | Knee | | | | Brain | | | |
|---|---|---|---|---|---|---|---|---|
| | $R = 4$ | | $R = 8$ | | $R = 4$ | | $R = 8$ | |
| method | PSNR ± SE | SSIM ± SE | PSNR ± SE | SSIM ± SE | PSNR ± SE | SSIM ± SE | PSNR ± SE | SSIM ± SE |
| P-VDAMP [102] | 33.84 ± 0.40 | 0.9018 ± 0.0036 | 20.34 ± 0.46 | 0.5614 ± 0.0051 | 30.30 ± 0.16 | 0.8847 ± 0.0021 | 13.51 ± 0.26 | 0.4763 ± 0.0069 |
| PnP-PDS [108] | 36.28 ± 0.38 | 0.9204 ± 0.0028 | 32.34 ± 0.32 | 0.8556 ± 0.0040 | 38.07 ± 0.23 | 0.9501 ± 0.0016 | 28.97 ± 0.13 | 0.8269 ± 0.0031 |
| D-GEC (proposed) | **38.82** ± 0.50 | **0.9504** ± 0.0023 | **33.66** ± 0.28 | **0.8893** ± 0.0028 | **39.04** ± 0.29 | **0.9631** ± 0.0013 | **30.61** ± 0.19 | **0.9015** ± 0.0031 |

and $R = 8$, and a measurement SNR of 40 dB, on the fastMRI brain and knee datasets. With the exception of the sampling mask, the experimental setup was identical to that in Section 2.3.3. Although [102] states that P-VDAMP is not intended to be used for "purely 2D acquisitions" like that associated with a 2D line mask, we show P-VDAMP performance for completeness. To run P-VDAMP, we gave it a 2D sampling density that was uniform along the fully sampled dimension and proportional to the 1D sampling density along the subsampled dimension (recall Figs. 2.1(c)-(d)).

Table 2.2 shows PSNR and SSIM averaged over the test images with the corresponding standard errors. There it can be seen that D-GEC significantly outperformed the other techniques on both datasets at both acceleration rates. For example, D-GEC outperformed its closest competitor, PnP-PDS, by 2.54 and 1.32 dB at $R = 4$ and $R = 8$, respectively, on the knee data.

### 2.3.5 Single-coil MRI algorithm comparison with a 2D point mask

In this section we compare the performance of D-GEC to several other recently proposed algorithms for single-coil MRI recovery using a 2D point mask. We examine two acceleration rates, $R = 4$ and $R = 8$, and a measurement SNR of 45 dB. For this section, we used the Stanford 2D FSE dataset [31] with the test images in Fig. 2.3. The details of the experimental setup are reported in Appendix C-B.

We compared our proposed D-GEC algorithm to D-AMP-MRI [126], VDAMP [100], D-VDAMP [101], and PnP-PDS [108]. We used a 2D point mask out of fairness to VDAMP and D-VDAMP, which were designed around 2D point masks. For VDAMP and D-VDAMP, we ran the authors' implementations at their default settings. For D-AMP-MRI and PnP-PDS, we used a bias-free DnCNN [142] denoiser trained to minimize the $\ell_2$ loss when removing WGN with SDs uniformly distributed in the interval $[0, 55/255]$. This bias-free network is known to perform very well over a wide SD range, and so there is no advantage in training multiple denoisers over different SNR ranges [142]. We ran the D-AMP-MRI and PnP-PDS algorithms for 50 and 300 iterations, respectively. Because the PnP fixed-points strongly depend on the chosen penalty parameter, we carefully tuned the PnP-PDS parameter at each acceleration rate $R$ to maximize PSNR on the validation set. For D-GEC, we used a Haar wavelet

Table 2.3: Single-coil image recovery results averaged over the ten test images.

| | $R = 4$ | | $R = 8$ | |
|---|---|---|---|---|
| method | PSNR ± SE | SSIM ± SE | PSNR ± SE | SSIM ± SE |
| D-AMP-MRI [126] | 33.28 ± 4.62 | 0.7789 ± 0.0900 | 25.83 ± 4.33 | 0.7252 ± 0.1214 |
| VDAMP [100] | 33.10 ± 1.30 | 0.8650 ± 0.0243 | 28.47 ± 0.96 | 0.7378 ± 0.0313 |
| D-VDAMP [101] | 42.57 ± 1.48 | 0.9731 ± 0.0089 | 35.18 ± 1.93 | 0.9023 ± 0.0248 |
| PnP-PDS [108] | 43.36 ± 1.60 | 0.9787 ± 0.0076 | 38.10 ± 1.75 | 0.9527 ± 0.0158 |
| D-GEC (proposed) | **45.17** ± 1.62 | **0.9824** ± 0.0066 | **38.97** ± 1.76 | **0.9570** ± 0.0132 |

transform of depth $D = 4$, which yields $L = 13$ subbands, and a corr+corr bias-free DnCNN denoiser; see Appendix C-B for additional details.

Table 2.3 shows PSNR and SSIM averaged over the 10 test images with the corresponding standard errors. There it can be seen that D-GEC significantly outperformed the other techniques at both tested acceleration rates. For example, D-GEC outperformed its closest competitor, PnP-PDS, by 1.81 and 0.87 dB at $R = 4$ and $R = 8$, respectively.

Figure 2.12 shows PSNR versus iteration for several algorithms at $R = 4$ and SNR = 45 dB. The PSNR value shown is the average over all 10 test images in Fig. 2.3. Two versions of D-VDAMP are shown in Fig. 2.12: the standard version from [101], which includes early stopping, and a modified version without early stopping. The importance of early stopping is clear from the figure. The figure also shows that, for this single-coil dataset, D-GEC took more iterations to converge than the other algorithms but yielded a larger value of PSNR at convergence. In the multicoil case in Fig. 2.11, D-GEC took an order-of-magnitude fewer iterations to converge.

Figure 2.13 shows image recoveries for a typical Stanford 2D FSE MRI image at $R = 4$ and measurement SNR = 45 dB. For this experiment, D-GEC significantly

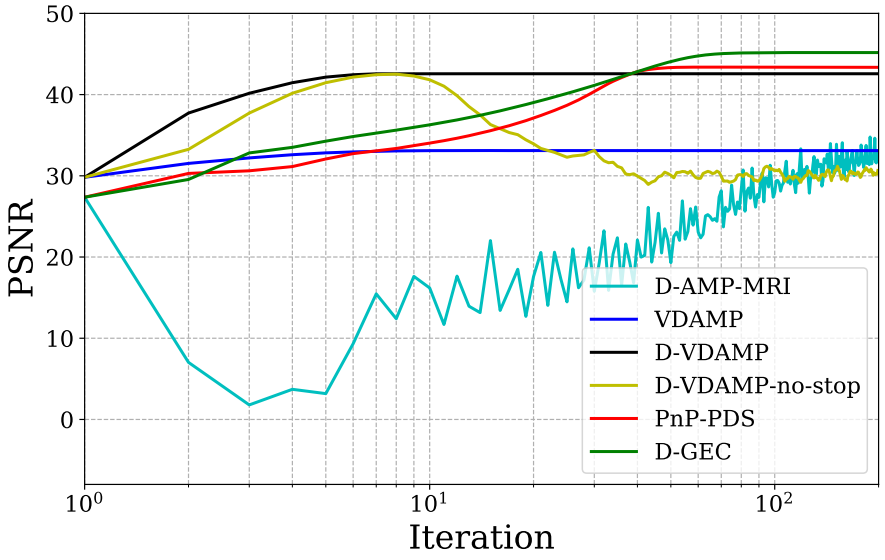


Figure 2.12: PSNR versus iterations for single-coil MRI recovery at $R = 4$ and SNR $= 45$ dB. PSNR was averaged over the 10 test images in Fig. 2.3.

outperformed the competing algorithms in PSNR, and its error image looks the least structured. Also, the zoomed subplots show that D-GEC recovered fine details in the true image that are missed by its competitors.

## 2.4 Conclusion

PnP algorithms require relatively few training images and are insensitive to deviations in the forward model $\boldsymbol{A}$ and measurement noise statistics between training and test. However, PnP can be improved, because the denoisers typically used for PnP are trained to remove white Gaussian noise, whereas the denoiser input errors encountered in PnP are typically non-white and non-Gaussian. In this chapter, we proposed a new PnP algorithm, called Denoising Generalized Expectation-Consistent (D-GEC) approximation, to address this shortcoming for Fourier-structured $\boldsymbol{A}$ and Gaussian

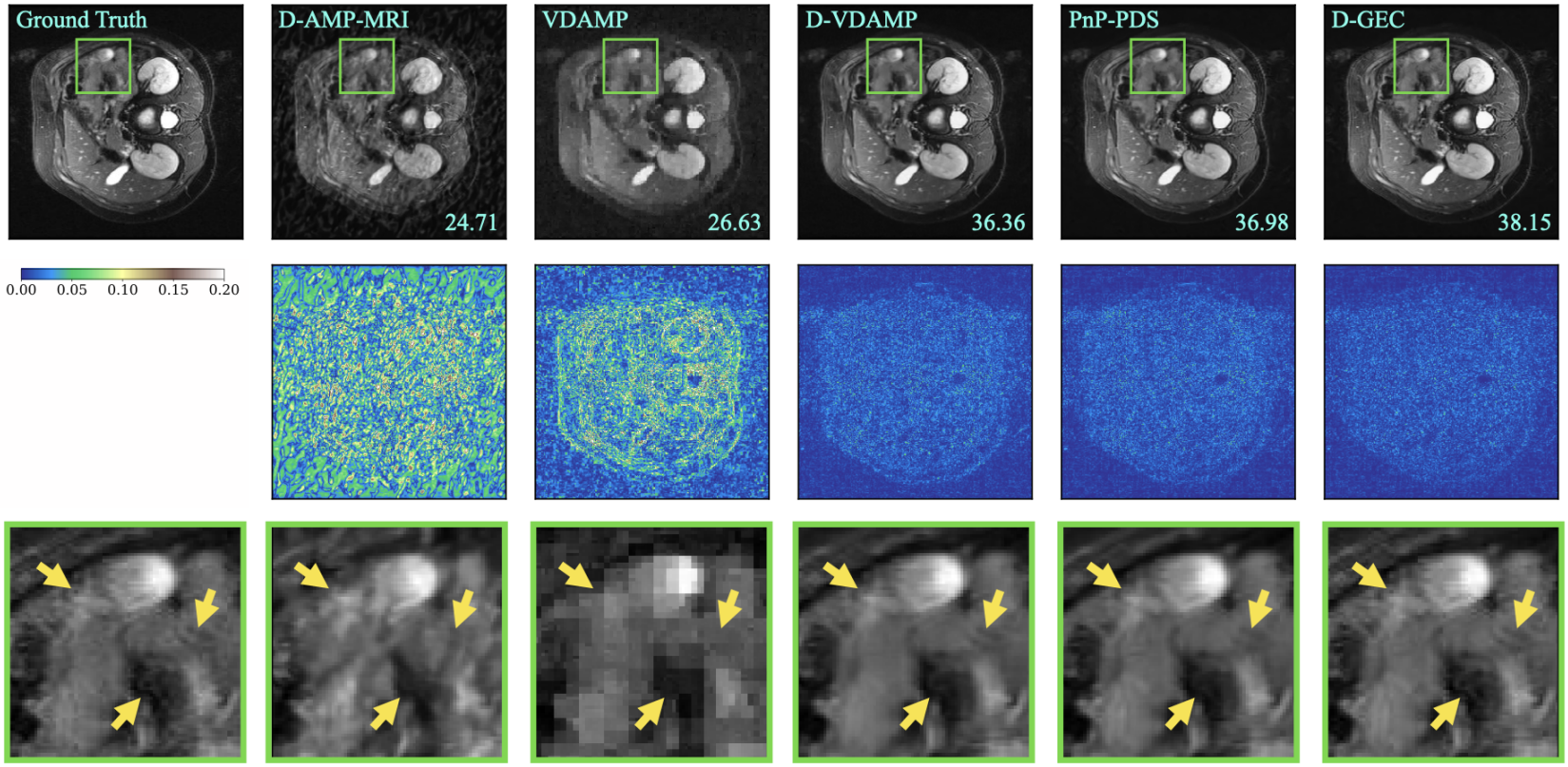


Figure 2.13: Example single-coil MRI image recoveries and error images at $R = 4$ and SNR $= 45$ dB. The number printed on each recovered image shows its PSNR. The bottom row is a zoomed in version of the green square in the top row. This figure is best viewed in electronic form.

measurement noise. In particular, D-GEC is designed to make the denoiser input error white and Gaussian within each wavelet subband with a predictable variance. We then proposed a new DNN denoiser that is capable of exploiting the knowledge of those subband error variances. Our "corr+corr" denoiser takes in a signal corrupted by correlated Gaussian noise, as well as independent realization(s) of the same correlated noise. It then learns how to extract the statistics of the provided noise and then use them productively for denoising the signal. Numerical experiments with single- and multicoil MRI image recovery demonstrate that D-GEC does indeed provide the denoiser with subband errors that are white and Gaussian with a predictable variance. Furthermore, the experiments demonstrate improved recovery accuracy relative to existing state-of-the-art PnP methods for MRI, especially with practical 2D line sampling masks. More work is needed to understand the theoretical properties of the proposed D-GEC and corr+corr denoisers.

# Chapter 3: Fast and Robust Phase Retrieval via Deep Expectation-Consistent Approximation

## 3.1 Background

### 3.1.1 Generalized Linear Model (GLM)

Recall that in Generalized Linear Model, relationship between $\boldsymbol{y} \in \mathcal{Y}^m$ and $\boldsymbol{x} \in \mathbb{R}^d$ or $\mathbb{C}^d$ can be described using a likelihood model of the form

$$p_{\mathsf{y}|\mathsf{x}}(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{m} p_{\mathsf{y}|\mathsf{z}}(y_i|z_i) \text{ for } \boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}, \tag{3.1}$$

where the forward operator $\boldsymbol{A} \in \mathbb{C}^{m\times d}$ and scalar measurement channel $p_{\mathsf{y}|\mathsf{z}}$ are both known. As previously discussed, versions of $p_{\mathsf{y}|\mathsf{z}}$ exist for, e.g., additive noise of an arbitrary distribution, logistic regression [33], Poisson regression [145], noisy quantization [146], and phase retrieval [36, 37]. In this work, we focus on phase retrieval, although many of the ideas that we describe can be applied more generally.

For phase retrieval, several variants of $p_{\mathsf{y}|\mathsf{z}}$ have been employed, such as $p_{\mathsf{y}|\mathsf{z}}(y_i|z_i) = \mathcal{N}(y_i; |z_i|^2, v)$ [53], $p_{\mathsf{y}|\mathsf{z}}(y_i|z_i) = \text{Poisson}(y_i; |z_i|^2)$ [147], the Rician model resulting from $y_i \sim |z_i + w_i|$ with $w_i \sim \mathcal{N}(0, v)$ [60], and

$$p_{\mathsf{y}|\mathsf{z}}(y_i|z_i) = \mathcal{N}(y_i; |z_i|, v). \tag{3.2}$$

For algorithm design, we will focus on (3.2), as it has proven to be effective in practice [148] and is adopted by state-of-the-art techniques like prDeep [56] and Deep-ITA [65].

### 3.1.2 Expectation-Consistent Approximation for GLM

We now consider statistical estimation of $\boldsymbol{x} \sim p_{\mathsf{x}}$ from $\boldsymbol{y}$ under the GLM (3.1). When both $p_{\mathsf{x}}$ and $p_{\mathsf{y}|\mathsf{z}}$ are Gaussian, the maximum a posteriori (MAP) and minimum mean-squared error (MMSE) estimates of $\boldsymbol{x}$ coincide and are analytically computable. But Gaussian $p_{\mathsf{y}|\mathsf{z}}$ is not useful for phase retrieval. When both $p_{\mathsf{x}}$ and $p_{\mathsf{y}|\mathsf{z}}$ are log-concave, the posterior $p_{\mathsf{x}|\mathsf{y}}$ is also log-concave and thus the MAP estimate of $\boldsymbol{x}$ can be computed using standard convex-optimization algorithms [149, 150]. Although the $p_{\mathsf{y}|\mathsf{z}}$ in (3.2) is indeed log-concave, high-fidelity image priors $p_{\mathsf{x}}$ are not. Furthermore, the widespread use of PSNR as a recovery-performance metric suggests the use of MMSE estimation over MAP.

The expectation-consistent (EC) approximation framework [28, 80] is a well established method to approximate the MMSE estimate of $\boldsymbol{x}$ from $\boldsymbol{y}$, i.e., the conditional mean $\mathrm{E}\{\boldsymbol{x}|\boldsymbol{y}\}$. The application of EC to the SLM (1.4) or GLM (3.1) is sometimes referred to as vector AMP (VAMP) [120] and generalized VAMP [61], respectively. Rigorous analyses of VAMP have established its MSE optimality under high-dimensional rotationally invariant $\boldsymbol{A}$ and various assumptions on $p_{\mathsf{x}}$ and $p_{\mathsf{y}|\mathsf{z}}$ [81, 121, 151, 152].

In the above works, EC is employed to MMSE-estimate $\boldsymbol{x}$ from $\boldsymbol{y}$ using the prior $p_{\mathsf{x}}$ and the likelihood $p_{\mathsf{y}|\mathsf{x}}$ from (3.1). This approach, however, fails for many deterministic choices of $\boldsymbol{A}$, such as the Fourier-based operators that arise in phase retrieval (see the discussion in [56, 67]) and magnetic resonance imaging (see the

discussion in [100, 101, 127, 153]). In this work, we instead employ EC to MMSE-estimate $\boldsymbol{z}$ from $\boldsymbol{y}$, using the prior $p_{\mathsf{z}}$ and likelihood $p_{\mathsf{y}|\mathsf{z}}$. Separately, we estimate $\boldsymbol{x}$ from $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}$. Thus, when reviewing EC below, we do so in the context of estimating $\boldsymbol{z}$ using $p_{\mathsf{z}}$ and $p_{\mathsf{y}|\mathsf{z}}$. The more "traditional" application of EC (estimating $\boldsymbol{x}$ using $p_{\mathsf{x}}$ and $p_{\mathsf{y}|\mathsf{x}}$) is identical up to variable substitutions.

To begin our review of EC, notice that one can write the true posterior distribution $p_{\mathsf{z}|\mathsf{y}}(\cdot|\boldsymbol{y})$ without approximation as

$$
\begin{aligned}
p_{\mathsf{z}|\mathsf{y}}(\cdot|\boldsymbol{y}) &= \arg\min_{q} D\big(q\big\|p_{\mathsf{z}|\mathsf{y}}(\cdot|\boldsymbol{y})\big) && (3.3)\\
&= \arg\min_{q} D\big(q\big\|p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y}|\cdot)\big) + D(q\|p_{\mathsf{z}}) + H(q) && (3.4)\\
&= \arg\min_{q_1=q_2=q_3} \underbrace{D\big(q_1\big\|p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y}|\cdot)\big) + D(q_2\|p_{\mathsf{z}}) + H(q_3)}_{\triangleq\, J_{\mathsf{Gibbs}}(q_1,q_2,q_3)}, && (3.5)
\end{aligned}
$$

where $D(q\|p) \triangleq \int q(\boldsymbol{z}) \ln \frac{q(\boldsymbol{z})}{p(\boldsymbol{z})} \,\mathrm{d}\boldsymbol{z}$ denotes Kullback-Leibler (KL) divergence, $H(q) \triangleq -\int q(\boldsymbol{z}) \ln q(\boldsymbol{z}) \,\mathrm{d}\boldsymbol{z}$ denotes differential entropy, and $p_{\mathsf{z}|\mathsf{y}}(\boldsymbol{z}|\boldsymbol{y}) = p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y}|\boldsymbol{z})p_{\mathsf{z}}(\boldsymbol{z})/p_{\mathsf{y}}(\boldsymbol{y})$ via Bayes rule was used for (3.4). Because the optimization in (3.5) is intractable, Opper and Winther [118] proposed to relax the equality constraints to moment-matching constraints, i.e.,

$$
\begin{aligned}
&\arg\min_{q_1,q_2,q_3} J_{\mathsf{Gibbs}}(q_1,q_2,q_3) \text{ such that} && (3.6)\\
&\begin{cases}
\mathrm{E}\{\boldsymbol{z}|q_1\} = \mathrm{E}\{\boldsymbol{z}|q_2\} = \mathrm{E}\{\boldsymbol{z}|q_3\} \triangleq \widehat{\boldsymbol{z}}\\
\mathrm{tr}(\mathrm{Cov}\{\boldsymbol{z}|q_1\}) = \mathrm{tr}(\mathrm{Cov}\{\boldsymbol{z}|q_2\}) = \mathrm{tr}(\mathrm{Cov}\{\boldsymbol{z}|q_3\}) \triangleq m\widehat{v},
\end{cases}
\end{aligned}
$$

where $\mathrm{E}\{\boldsymbol{z}|q_i\}$ and $\mathrm{Cov}\{\boldsymbol{z}|q_i\}$ denote the mean and covariance of $\boldsymbol{z}$ under $\boldsymbol{z} \sim q_i$ and $\mathrm{tr}(\cdot)$ denotes the trace, and then approximate $\mathrm{E}\{\boldsymbol{z}|\boldsymbol{y}\}$ by $\widehat{\boldsymbol{z}}$. The solution to (3.6)

takes the form

$$q_1(\boldsymbol{z}; \overline{\boldsymbol{z}}^{(1)}, \overline{v}^{(1)}) \propto p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y}|\boldsymbol{z}) \mathcal{N}(\boldsymbol{z}; \overline{\boldsymbol{z}}^{(1)}, \overline{v}^{(1)} \boldsymbol{I}) \tag{3.7}$$

$$q_2(\boldsymbol{z}; \overline{\boldsymbol{z}}^{(2)}, \overline{v}^{(2)}) \propto p_{\mathsf{z}}(\boldsymbol{z}) \mathcal{N}(\overline{\boldsymbol{z}}^{(2)}; \boldsymbol{z}, \overline{v}^{(2)} \boldsymbol{I}) \tag{3.8}$$

$$q_3(\boldsymbol{z}; \widehat{\boldsymbol{z}}, \widehat{v}) = \mathcal{N}(\boldsymbol{z}; \widehat{\boldsymbol{z}}, \widehat{v} \boldsymbol{I}) \tag{3.9}$$

for some $\overline{\boldsymbol{z}}^{(1)}, \overline{\boldsymbol{z}}^{(2)}$ and $\overline{v}^{(1)}, \overline{v}^{(2)}$, known as the "extrinsic" means and variances, respectively, that must be computed (see the discussion after (3.12)), and for

$$\widehat{v} = \left(1/\overline{v}^{(1)} + 1/\overline{v}^{(2)}\right)^{-1} \tag{3.10}$$

$$\widehat{\boldsymbol{z}} = \left(\frac{\overline{\boldsymbol{z}}^{(1)}}{\overline{v}^{(1)}} + \frac{\overline{\boldsymbol{z}}^{(2)}}{\overline{v}^{(2)}}\right) \widehat{v}. \tag{3.11}$$

Note that $q_1$ combines the true likelihood $p_{\mathsf{y}|\mathsf{z}}$ with the Gaussian "pseudo-prior" $\boldsymbol{z} \sim \mathcal{N}(\overline{\boldsymbol{z}}^{(1)}, \overline{v}^{(1)} \boldsymbol{I})$, while $q_2$ combines the true prior $p_{\mathsf{z}}$ with the Gaussian "pseudo-likelihood" $\mathcal{N}(\overline{\boldsymbol{z}}^{(2)}; \boldsymbol{z}, \overline{v}^{(2)} \boldsymbol{I})$, i.e., the model

$$\overline{\boldsymbol{z}}^{(2)} = \boldsymbol{z}_{\mathsf{true}} + \boldsymbol{e} \text{ with } \boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \overline{v}^{(2)} \boldsymbol{I}). \tag{3.12}$$

To solve for $(\overline{\boldsymbol{z}}^{(1)}, \overline{\boldsymbol{z}}^{(2)}, \overline{v}^{(1)}, \overline{v}^{(2)})$, Opper and Winther [118] proposed Alg. 4, which iterates the following four steps: compute the posterior mean and trace-covariance of $q_2(\cdot; \overline{\boldsymbol{z}}^{(2)}, \overline{v}^{(2)})$, update the extrinsic quantities $(\overline{\boldsymbol{z}}^{(1)}, \overline{v}^{(1)})$ to obey (3.10)-(3.11), compute the posterior mean and trace-covariance of $q_1(\cdot; \overline{\boldsymbol{z}}^{(1)}, \overline{v}^{(1)})$, and update the extrinsic quantities $(\overline{\boldsymbol{z}}^{(2)}, \overline{v}^{(2)})$ to obey (3.10)-(3.11). The non-informative initialization of $\overline{v}^{(2)}$ and $\overline{\boldsymbol{z}}^{(2)}$ in line 3 ensures that $q_2 = p_{\mathsf{z}}$ initially, so that the initial $\widehat{v}^{(2)}$ and $\widehat{\boldsymbol{z}}^{(2)}$ are the prior variance and mean (as are the initial $\overline{v}^{(1)}$ and $\overline{\boldsymbol{z}}^{(1)}$). Algorithm 4 can be interpreted [118] as an instance of expectation propagation (EP) from [119].

**Algorithm 4** EC to infer $\boldsymbol{z} \sim p_{\mathsf{z}}$ from $\boldsymbol{y} \sim p_{\mathsf{y}|\mathsf{z}}(\cdot|\boldsymbol{z})$

**Require:** $p_{\mathsf{z}}(\cdot), p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y}|\cdot)$

$q_1(\boldsymbol{z}; \overline{\boldsymbol{z}}^{(1)}, \overline{v}^{(1)}) \propto p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y}|\boldsymbol{z}) \mathcal{N}(\boldsymbol{z}; \overline{\boldsymbol{z}}^{(1)}, \overline{v}^{(1)} \boldsymbol{I})$

$q_2(\boldsymbol{z}; \overline{\boldsymbol{z}}^{(2)}, \overline{v}^{(2)}) \propto p_{\mathsf{z}}(\boldsymbol{z}) \mathcal{N}(\boldsymbol{z}; \overline{\boldsymbol{z}}^{(2)}, \overline{v}^{(2)} \boldsymbol{I})$

$\overline{v}^{(2)} \leftarrow \infty$ and $\overline{\boldsymbol{z}}^{(2)} \leftarrow$ arbitrary finite vector in $\mathbb{R}^m$

// Exploit prior $p_{\mathsf{z}}(\cdot)$ and pseudo-likelihood

$\widehat{v}^{(2)} \leftarrow \frac{1}{m} \operatorname{tr}(\operatorname{Cov}\{\boldsymbol{z}|q_2(\cdot; \overline{\boldsymbol{z}}^{(2)}, \overline{v}^{(2)})\})$

$\widehat{\boldsymbol{z}}^{(2)} \leftarrow \mathrm{E}\{\boldsymbol{z}|q_2(\cdot; \overline{\boldsymbol{z}}^{(2)}, \overline{v}^{(2)})\}$

$\overline{v}^{(1)} \leftarrow (1/\widehat{v}^{(2)} - 1/\overline{v}^{(2)})^{-1}$

$\overline{\boldsymbol{z}}^{(1)} \leftarrow (\widehat{\boldsymbol{z}}^{(2)}/\widehat{v}^{(2)} - \overline{\boldsymbol{z}}^{(2)}/\overline{v}^{(2)})\overline{v}^{(1)}$

// Exploit likelihood $p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y}|\cdot)$ and pseudo-prior

$\widehat{v}^{(1)} \leftarrow \frac{1}{m} \operatorname{tr}(\operatorname{Cov}\{\boldsymbol{z}|q_1(\cdot; \overline{\boldsymbol{z}}^{(1)}, \overline{v}^{(1)})\})$

$\widehat{\boldsymbol{z}}^{(1)} \leftarrow \mathrm{E}\{\boldsymbol{z}|q_1(\cdot; \overline{\boldsymbol{z}}^{(1)}, \overline{v}^{(1)})\}$

$\overline{v}^{(2)} \leftarrow (1/\widehat{v}^{(1)} - 1/\overline{v}^{(1)})^{-1}$

$\overline{\boldsymbol{z}}^{(2)} \leftarrow (\widehat{\boldsymbol{z}}^{(1)}/\widehat{v}^{(1)} - \overline{\boldsymbol{z}}^{(1)}/\overline{v}^{(1)})\overline{v}^{(2)}$ **return** $\widehat{\boldsymbol{z}}^{(2)}$ as the EC approximation of $\mathrm{E}\{\boldsymbol{z}|\boldsymbol{y}\}$

## 3.2 Proposed Method

In the following subsections we propose several modifications of the standard EC approach from Alg. 4 that involve stochastic damping, deep networks, and simplifications/approximations specific to phase retrieval.

### 3.2.1 Stochastic Damping

The EC Alg. 4 is not guaranteed to converge for general $p_{\mathsf{z}}$ and $p_{\mathsf{y}|\mathsf{z}}$. One technique that helps promote convergence is "damping" [127, 154], which slows the updates in a way that preserves the EC fixed points. Damping is also referred to as "under-relaxation" in the context of iterative numerical methods and computational fluid dynamics [155]. It typically takes the form $\boldsymbol{\varphi}_{\mathsf{new}} \leftarrow \mu \boldsymbol{\varphi}_{\mathsf{raw}} + (1-\mu)\boldsymbol{\varphi}_{\mathsf{old}}$ when iteratively updating $\boldsymbol{\varphi}$ using the most recent computation $\boldsymbol{\varphi}_{\mathsf{raw}}$. Here, $\mu \in (0, 1]$ is a user-selectable parameter, where lower values increase stability but slow convergence.

The damping technique from [127] considers the $\overline{v}^{(i)}$ and $\overline{\boldsymbol{z}}^{(i)}$ computed in lines 7-8 and 12-13 of Alg. 4 as "raw" quantities and damps them using the additional steps (for $i \in \{1, 2\}$)

$$\overline{v}^{(i)} \leftarrow \left(\mu^{(i)}\sqrt{\overline{v}^{(i)}_{\mathsf{raw}}} + (1-\mu^{(i)})\sqrt{\overline{v}^{(i)}_{\mathsf{old}}}\right)^2 \tag{3.13a}$$

$$\overline{\boldsymbol{z}}^{(i)} \leftarrow \mu^{(i)}\overline{\boldsymbol{z}}^{(i)}_{\mathsf{raw}} + (1-\mu^{(i)})\overline{\boldsymbol{z}}^{(i)}_{\mathsf{old}}, \tag{3.13b}$$

where $\overline{v}^{(i)}_{\mathsf{old}}$ and $\overline{\boldsymbol{z}}^{(i)}_{\mathsf{old}}$ denote the values of $\overline{v}^{(i)}$ and $\overline{\boldsymbol{z}}^{(i)}$ from the previous iteration and $\mu^{(i)} \in (0, 1]$ is a fixed damping constant. When $\mu^{(1)} = 1 = \mu^{(2)}$, the original EC is recovered.

We now propose a "stochastic damping" procedure inspired by diffusion methods like [84]. It damps $\overline{v}^{(2)}$ as in (3.13a) but constructs $\overline{\boldsymbol{z}}^{(2)}$ by adding AWGN of a prescribed variance:

$$\overline{v}^{(2)} \leftarrow \left(\mu^{(2)}\sqrt{\overline{v}^{(2)}_{\mathsf{raw}}} + (1-\mu^{(2)})\sqrt{\overline{v}^{(2)}_{\mathsf{old}}}\right)^2 \tag{3.14a}$$

$$\overline{\boldsymbol{z}}^{(2)} \leftarrow \overline{\boldsymbol{z}}^{(2)}_{\mathsf{raw}} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \max\{\overline{v}^{(2)} - \overline{v}^{(2)}_{\mathsf{raw}}, 0\}\boldsymbol{I}). \tag{3.14b}$$

To better understand (3.14b), recall from (3.12) that, under ideal conditions, $\overline{\boldsymbol{z}}^{(2)} = \boldsymbol{z}_{\mathsf{true}} + \boldsymbol{e}$ for $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \overline{v}^{(2)}\boldsymbol{I})$ and $\overline{\boldsymbol{z}}^{(2)}_{\mathsf{raw}} = \boldsymbol{z}_{\mathsf{true}} + \boldsymbol{\varepsilon}$ for $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \overline{v}^{(2)}_{\mathsf{raw}}\boldsymbol{I})$. This implies that a valid $\overline{\boldsymbol{z}}^{(2)}$ can be constructed from $\overline{\boldsymbol{z}}^{(2)}_{\mathsf{raw}}$ by adding additional AWGN of variance $\overline{v}^{(2)} - \overline{v}^{(2)}_{\mathsf{raw}}$, as done in (3.14b). Here, it is expected that $\overline{v}^{(2)} > \overline{v}^{(2)}_{\mathsf{raw}}$ because the variances decrease over the iterations and damping aims to slow down that decrease. In Section 3.3, we show that (3.14) has advantages over (3.13), perhaps because (3.14) attempts to enforce the EC model (3.12).

### 3.2.2 deepEC for general $\boldsymbol{A}$

As formulated in Alg. 4, EC estimates $\boldsymbol{z} \sim p_{\mathsf{z}}$ from observations $\boldsymbol{y} \sim p_{\mathsf{y}|\mathsf{z}}(\cdot|\boldsymbol{z})$. To solve inverse problems using the GLM model (3.1), we instead want to estimate $\boldsymbol{x} \sim p_{\mathsf{x}}$ from $\boldsymbol{y} \sim p_{\mathsf{y}|\mathsf{z}}(\cdot|\boldsymbol{z})$ where $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}$. In principle, this can be done using a minor modification of Alg. 4 that rewrites $\widehat{\boldsymbol{z}}^{(2)}$ in line 6 as a function of an MMSE estimate of $\boldsymbol{x}$:

$$\widehat{\boldsymbol{z}}^{(2)} = \mathrm{E}\{\boldsymbol{z}|q_2(\cdot;\overline{\boldsymbol{z}}^{(2)},\overline{v}^{(2)})\} \tag{3.15}$$

$$= \mathrm{E}\{\boldsymbol{z}|\overline{\boldsymbol{z}}^{(2)} = \boldsymbol{z}+\boldsymbol{e}\},\ \boldsymbol{z} \sim p_{\mathsf{z}},\ \boldsymbol{e} \sim \mathcal{N}(0,\overline{v}^{(2)}\boldsymbol{I}) \tag{3.16}$$

$$= \boldsymbol{A}\underbrace{\mathrm{E}\{\boldsymbol{x}|\overline{\boldsymbol{z}}^{(2)} = \boldsymbol{A}\boldsymbol{x}+\boldsymbol{e}\},\ \boldsymbol{x} \sim p_{\mathsf{x}},\ \boldsymbol{e} \sim \mathcal{N}(0,\overline{v}^{(2)}\boldsymbol{I})}_{\triangleq\ \widehat{\boldsymbol{x}}^{(2)}}, \tag{3.17}$$

and then returns $\widehat{\boldsymbol{x}}^{(2)}$ as the EC output. The practical challenge, however, is that computing $\widehat{\boldsymbol{x}}^{(2)}$ via (3.17) involves MMSE estimation of $\boldsymbol{x}$ under the SLM (1.4), which is itself non-trivial.

Fortunately, it is now commonplace to train a deep networks to solve SLMs [18]. Writing $\widehat{\boldsymbol{x}}^{(2)} \approx \boldsymbol{d}(\overline{\boldsymbol{z}}^{(2)};\overline{v}^{(2)})$ for deep network $\boldsymbol{d}$, we could train the network parameters $\boldsymbol{\theta}$ via

$$\arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \mathrm{E}\left\{\|\boldsymbol{x}_t - \boldsymbol{d}(\boldsymbol{A}\boldsymbol{x}_t+\boldsymbol{e};\overline{v}^{(2)})\|^2\right\} \tag{3.18}$$

with training data $\{\boldsymbol{x}_t\}_{t=1}^T$, random noise $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0},\overline{v}^{(2)}\boldsymbol{I})$, and random noise variance $\overline{v}^{(2)} \sim \mathrm{Unif}[0,v_{\max}]$ for some $v_{\max}$. Likewise, we could train another deep network $h_{\boldsymbol{\phi}}$ to approximate the posterior variance $\widehat{v}^{(2)}$. Writing $\widehat{v}^{(2)} \approx h_{\boldsymbol{\phi}}(\overline{\boldsymbol{z}}^{(2)};\overline{v}^{(2)})$, its network parameters $\boldsymbol{\phi}$ might be trained via

$$\arg\min_{\boldsymbol{\phi}} \sum_{t=1}^{T} \mathrm{E}\left\{\left|\tfrac{1}{m}\|\boldsymbol{z}_t - \boldsymbol{A}\widehat{\boldsymbol{x}}_t^{(2)}\|^2 - h_{\boldsymbol{\phi}}(\boldsymbol{A}\boldsymbol{x}_t+\boldsymbol{e};\overline{v}^{(2)})\right|\right\}, \tag{3.19}$$

**Algorithm 5** deepECpr

**Require:** transform $\boldsymbol{A}$, channel $p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y},\cdot)$, denoiser $\boldsymbol{d}(\cdot,\cdot)$, denoising factor $\beta \in (0,1)$, image initialization $\widehat{\boldsymbol{x}}_{\mathsf{init}}$, variance initialization $\overline{v}_{\mathsf{init}}$, variance initialization factor $\zeta \approx 1.2$, damping factors $\mu^{(1)} \in (0,1]$ and $\mu^{(2)} \in (0,1]$

initialize: $\overline{\boldsymbol{z}}^{(2)} \leftarrow \boldsymbol{A}\widehat{\boldsymbol{x}}_{\mathsf{init}} + \boldsymbol{n}$ with $\boldsymbol{n} \sim \mathcal{CN}(\boldsymbol{0}, \overline{v}_{\mathsf{init}}\boldsymbol{I})$, and $\overline{v}^{(2)} \leftarrow \zeta \overline{v}_{\mathsf{init}}$

**for** $j = 1, \ldots, J$ **do**

  // Exploit prior $\boldsymbol{x} \sim p_{\mathsf{x}}$ (via denoiser $\boldsymbol{d}$) and pseudo-likelihood $p(\overline{\boldsymbol{z}}^{(2)}|\boldsymbol{x}) = \mathcal{CN}(\overline{\boldsymbol{z}}^{(2)}; \boldsymbol{A}\boldsymbol{x}, \overline{v}^{(2)}\boldsymbol{I})$

  denoise: $\widehat{\boldsymbol{x}}^{(2)} \leftarrow \boldsymbol{d}(\Re\{\boldsymbol{A}^{\mathsf{H}}\overline{\boldsymbol{z}}^{(2)}\}, 0.5\overline{v}^{(2)})$

  deep-network approximated posterior mean and variance: $\widehat{\boldsymbol{z}}^{(2)} \leftarrow \boldsymbol{A}\widehat{\boldsymbol{x}}^{(2)}$ and $\widehat{v}^{(2)} \leftarrow \beta\overline{v}^{(2)}$

  extrinsic variance: $\overline{v}^{(1)}_{\mathsf{raw}} \leftarrow (1/\widehat{v}^{(2)} - 1/\overline{v}^{(2)})^{-1}$ and $\overline{v}^{(1)}_{\mathsf{old}} \leftarrow \overline{v}^{(1)}$

  extrinsic mean: $\overline{\boldsymbol{z}}^{(1)}_{\mathsf{raw}} \leftarrow (\widehat{\boldsymbol{z}}^{(2)}/\widehat{v}^{(2)} - \overline{\boldsymbol{z}}^{(2)}/\overline{v}^{(2)})\overline{v}^{(1)}_{\mathsf{raw}}$ and $\overline{\boldsymbol{z}}^{(1)}_{\mathsf{old}} \leftarrow \overline{\boldsymbol{z}}^{(1)}$

  damping: $\overline{v}^{(1)} \leftarrow \left(\mu^{(1)}\sqrt{\overline{v}^{(1)}_{\mathsf{raw}}} + (1-\mu^{(1)})\sqrt{\overline{v}^{(1)}_{\mathsf{old}}}\right)^2$ and $\overline{\boldsymbol{z}}^{(1)} \leftarrow \mu^{(1)}\overline{\boldsymbol{z}}^{(1)}_{\mathsf{raw}} + (1-\mu^{(1)})\overline{\boldsymbol{z}}^{(1)}_{\mathsf{old}}$

  // Exploit likelihood $p_{\mathsf{y}|\mathsf{z}}(y_i|z_i)$ and pseudo-prior $z_i \sim \mathcal{CN}(\overline{z}_i^{(1)}, \overline{v}^{(1)})$ for $i = 1, \ldots, m$

  Laplace-approximated posterior mean and variance: $(\widehat{z}_i^{(1)}, \widehat{v}_i^{(1)})$ $\forall i$ and set $\widehat{v}^{(1)} \leftarrow \frac{1}{m}\sum_{i=1}^m \widehat{v}_i^{(1)}$

  extrinsic variance: $\overline{v}^{(2)}_{\mathsf{raw}} \leftarrow (1/\widehat{v}^{(1)} - 1/\overline{v}^{(1)})^{-1}$ and $\overline{v}^{(2)}_{\mathsf{old}} \leftarrow \overline{v}^{(2)}$

  extrinsic mean: $\overline{\boldsymbol{z}}^{(2)}_{\mathsf{raw}} \leftarrow (\widehat{\boldsymbol{z}}^{(1)}/\widehat{v}^{(1)} - \overline{\boldsymbol{z}}^{(1)}/\overline{v}^{(1)})\overline{v}^{(2)}_{\mathsf{raw}}$

  stochastic damping: $\overline{v}^{(2)} \leftarrow \left(\mu^{(2)}\sqrt{\overline{v}^{(2)}_{\mathsf{raw}}} + (1-\mu^{(2)})\sqrt{\overline{v}^{(2)}_{\mathsf{old}}}\right)^2$ and $\overline{\boldsymbol{z}}^{(2)} \leftarrow \overline{\boldsymbol{z}}^{(2)}_{\mathsf{raw}} + \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{CN}(\boldsymbol{0}, [\overline{v}^{(2)} - \overline{v}^{(2)}_{\mathsf{raw}}]\boldsymbol{I})$

**return** $\widehat{\boldsymbol{x}}^{(2)}$

where $\widehat{\boldsymbol{x}}_t^{(2)} \triangleq \boldsymbol{d}(\boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{e}; \overline{v}^{(2)})$ and where $\boldsymbol{x}_t$, $\boldsymbol{e}$, and $\overline{v}^{(2)}$ are constructed as they were for (3.18).

We will refer to EC with these deep-network approximations as "deepEC." To summarize, deepEC is EC Algorithm 4 but with $\widehat{\boldsymbol{z}}^{(2)}$ computed via $\boldsymbol{A}\boldsymbol{f}_{\boldsymbol{\theta}}(\overline{\boldsymbol{z}}^{(2)}; \overline{v}^{(2)})$ and $\widehat{v}^{(2)}$ computed via $h_{\boldsymbol{\phi}}(\overline{\boldsymbol{z}}^{(2)}; \overline{v}^{(2)})$. As such, deepEC solves GLM problems by plugging deep SLM networks into EC Alg. 4.

### 3.2.3 deepEC when $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} = \boldsymbol{I}$

One practical drawback to deepEC is that the deep networks $\boldsymbol{d}$ and $h_{\boldsymbol{\phi}}$ are dependent on the forward operator $\boldsymbol{A}$, and so the networks must be retrained for each new choice of $\boldsymbol{A}$. There is, however, an important exception to this rule, which is when $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} = \boldsymbol{I}$. For such column-orthogonal $\boldsymbol{A}$, a sufficient statistic [105] for the estimation of $\widehat{\boldsymbol{x}}^{(2)}$ in (3.17) is

$$\overline{\boldsymbol{r}}^{(2)} \triangleq \boldsymbol{A}^{\mathsf{H}}\overline{\boldsymbol{z}}^{(2)} = \boldsymbol{x} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \overline{v}^{(2)}\boldsymbol{I}), \tag{3.20}$$

in which case $\widehat{\boldsymbol{x}}^{(2)}$ solves the MMSE denoising problem

$$\widehat{\boldsymbol{x}}^{(2)} = \mathrm{E}\{\boldsymbol{x}|\overline{\boldsymbol{r}}^{(2)} = \boldsymbol{x} + \boldsymbol{\epsilon}\}, \ \boldsymbol{x} \sim p_{\mathsf{x}}, \ \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \overline{v}^{(2)}\boldsymbol{I}), \tag{3.21}$$

and we can approximate $\widehat{\boldsymbol{x}}^{(2)}$ using a deep *denoising* network $\boldsymbol{d}(\overline{\boldsymbol{r}}^{(2)}, \overline{v}^{(2)})$ that is invariant to $\boldsymbol{A}$. Likewise, we have that

$$\begin{aligned}
\widehat{v}^{(2)} &= \tfrac{1}{m}\operatorname{tr}(\operatorname{Cov}\{\boldsymbol{z}|\overline{\boldsymbol{z}}^{(2)}, \overline{v}^{(2)}\}) && (3.22)\\
&= \tfrac{1}{m}\operatorname{tr}(\boldsymbol{A}\operatorname{Cov}\{\boldsymbol{x}|\overline{\boldsymbol{z}}^{(2)}, \overline{v}^{(2)}\}\boldsymbol{A}^{\mathsf{H}}) && (3.23)\\
&= \tfrac{1}{m}\operatorname{tr}(\boldsymbol{A}\operatorname{Cov}\{\boldsymbol{x}|\overline{\boldsymbol{r}}^{(2)}, \overline{v}^{(2)}\}\boldsymbol{A}^{\mathsf{H}}) && (3.24)\\
&= \tfrac{1}{m}\operatorname{tr}(\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A}\operatorname{Cov}\{\boldsymbol{x}|\overline{\boldsymbol{r}}^{(2)}, \overline{v}^{(2)}\}) && (3.25)\\
&= \tfrac{1}{m}\operatorname{tr}(\operatorname{Cov}\{\boldsymbol{x}|\overline{\boldsymbol{r}}^{(2)}, \overline{v}^{(2)}\}), && (3.26)
\end{aligned}$$

and so the network $h_{\boldsymbol{\phi}}$ that deepEC uses to approximate $\widehat{v}^{(2)}$ can also be invariant to $\boldsymbol{A}$. One way to construct $h_{\boldsymbol{\phi}}$ would be to use the Monte-Carlo scheme from [117], which employs random $\{\boldsymbol{p}_c\}_{c=1}^{C}$ such that $\mathrm{E}\{\boldsymbol{p}_c\boldsymbol{p}_c^{\top}\} = \boldsymbol{I}$ and small $\delta > 0$ in

$$\begin{aligned}
&\operatorname{tr}(\operatorname{Cov}\{\boldsymbol{x}|\overline{\boldsymbol{r}}^{(2)}, \overline{v}^{(2)}\})\\
&\approx \frac{1}{C}\sum_{c=1}^{C}\frac{\boldsymbol{p}_c^{\top}[\boldsymbol{d}(\overline{\boldsymbol{r}}^{(2)}+\delta\boldsymbol{p}_c, \overline{v}^{(2)}) - \boldsymbol{d}(\overline{\boldsymbol{r}}^{(2)}, \overline{v}^{(2)})]}{\delta}
\end{aligned} \tag{3.27}$$

at the expense of $C \geq 1$ additional calls of the denoiser $\boldsymbol{d}$.

In summary, for any forward operator $\boldsymbol{A}$ that obeys $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} = \boldsymbol{I}$, the proposed deepEC scheme computes approximate-MMSE solutions to GLM problems by iteratively calling a deep denoiser, similar to the PnP [21] or RED [22, 23] schemes. But it requires far fewer iterations, as we demonstrate in Section 3.3.

For phase retrieval, $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} = \boldsymbol{I}$ holds for the two classes of $\boldsymbol{A}$ that dominate the literature: i) oversampled-Fourier (OSF) and ii) coded diffraction pattern (CDP) [156]. For OSF,

$$\boldsymbol{A} = \boldsymbol{F}_m \boldsymbol{O} \quad \text{for} \quad \boldsymbol{O} \triangleq \begin{bmatrix} \boldsymbol{I}_d \\ \boldsymbol{0}_{m-d \times d} \end{bmatrix} \in \mathbb{R}^{m \times d}, \tag{3.28}$$

where $\boldsymbol{F}_m \in \mathbb{C}^{m \times m}$ is a unitary 2D Fourier transform and $\boldsymbol{O}$ pads the vectorized image with zeros. For CDP,

$$\boldsymbol{A} = \frac{1}{\sqrt{K}} \begin{bmatrix} \boldsymbol{F}_d \operatorname{Diag}(\boldsymbol{c}_1) \\ \vdots \\ \boldsymbol{F}_d \operatorname{Diag}(\boldsymbol{c}_K) \end{bmatrix}, \tag{3.29}$$

where $\boldsymbol{F}_d \in \mathbb{C}^{d \times d}$ is a unitary 2D Fourier transform and $\boldsymbol{c}_k \in \mathbb{C}^d$ are random code vectors with entries drawn independently and uniformly from the unit circle in $\mathbb{C}$. By inspection, $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} = \boldsymbol{I}$ under (3.28) with $m \geq d$ and (3.29) with $K \geq 1$.

### 3.2.4 deepEC for Phase Retrieval

For PR, we propose to apply the deepEC algorithm from Section 3.2.3 with the stochastic damping scheme from Section 3.2.1. In addition, we use the minor modifications described below. The overall scheme, "deepECpr," is detailed in Alg. 5.

*Initialization:* Unlike the non-informative $\overline{\boldsymbol{z}}^{(2)}$ initialization used in Alg. 4, deepECpr uses an informative initialization. For OSF $\boldsymbol{A}$, we first run the HIO algorithm [49] to produce $\widehat{\boldsymbol{x}}_{\mathsf{init}}$, as in prDeep [56] and Deep-ITA [65]. For CDP $\boldsymbol{A}$, we find that it

suffices to run only the first iteration of HIO, and so $\widehat{\boldsymbol{x}}_{\mathsf{init}} = \boldsymbol{A}^{\mathsf{H}}(\boldsymbol{y}e^{j\angle \boldsymbol{A1}})$, where $e^{j\angle \boldsymbol{A1}}$ is computed and applied element-wise. Then, to initialize $\overline{\boldsymbol{z}}^{(2)}$, deepECpr transforms $\widehat{\boldsymbol{x}}_{\mathsf{init}}$ to the $\boldsymbol{z}$-domain and adds circular-complex AWGN:

$$\overline{\boldsymbol{z}}^{(2)} \leftarrow \boldsymbol{A}\widehat{\boldsymbol{x}}_{\mathsf{init}} + \boldsymbol{n} \text{ with } \boldsymbol{n} \sim \mathcal{CN}(\boldsymbol{0}, \overline{v}_{\mathsf{init}}\boldsymbol{I}). \tag{3.30}$$

By using a large noise variance (nominally $\overline{v}_{\mathsf{init}} \approx 100^2$ for pixel amplitudes in $[0, 255]$) the structured error artifacts in $\boldsymbol{A}\widehat{\boldsymbol{x}}_{\mathsf{init}}$ are suppressed. Because the error variance of $\overline{\boldsymbol{z}}^{(2)}$ is larger than $\overline{v}_{\mathsf{init}}$ due to the error in $\boldsymbol{A}\widehat{\boldsymbol{x}}_{\mathsf{init}}$, the initial $\overline{v}^{(2)}$ is set to $\zeta\overline{v}_{\mathsf{init}}$ for some $\zeta > 1$ (nominally $\zeta = 1.2$).

*Real-valued $\boldsymbol{x}$ and complex-valued $\boldsymbol{A}$:* The sufficient statistic $\overline{\boldsymbol{r}}^{(2)} = \boldsymbol{A}^{\mathsf{H}}\overline{\boldsymbol{z}}^{(2)}$ in (3.20) would be appropriate if both $\boldsymbol{x}$ and $\boldsymbol{A}$ were real-valued, or if both were complex-valued. However, for the PR problems that we consider, $\boldsymbol{x}$ is real-valued and $\boldsymbol{A}$ is complex-valued, and so we form the sufficient statistic as

$$\overline{\boldsymbol{r}}^{(2)} \triangleq \Re\{\boldsymbol{A}^{\mathsf{H}}\overline{\boldsymbol{z}}^{(2)}\} = \boldsymbol{x} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, 0.5\overline{v}^{(2)}\boldsymbol{I}), \tag{3.31}$$

and call the deepECpr denoiser as $\widehat{\boldsymbol{z}}^{(2)} = \boldsymbol{A}\boldsymbol{d}(\overline{\boldsymbol{r}}^{(2)}, 0.5\overline{v}^{(2)})$.

*Simplified variance-prediction:* As described in Section 3.2.2, the deep variance predictor $h_{\boldsymbol{\phi}}$ aims to estimate the error variance $\widehat{v}^{(2)}$ of the transformed denoiser output $\widehat{\boldsymbol{z}}^{(2)}$. Although the Monte-Carlo scheme (3.27) is an option, it involves additional denoiser calls, which would increase runtime. We thus propose to use the simple approximation

$$\widehat{v}^{(2)} \approx \beta\overline{v}^{(2)}, \tag{3.32}$$

where $\beta \in (0, 1)$ is a denoiser-specific constant. In words, (3.32) models the denoiser's output error variance $\widehat{v}^{(2)}$ as a fixed fraction of its input error variance $\overline{v}^{(2)}$.

*Laplace approximation:* For PR, we adopt the likelihood

$$p_{\mathsf{y}|\mathsf{z}}(y_i|z_i) = \mathcal{N}(y_i; |z_i|, v), \quad i = 1, \ldots, m, \tag{3.33}$$

due to its excellent performance in practice [148]. However, the resulting EC posterior

$$q_2(z_i; \overline{z}_i^{(1)}, \overline{v}^{(1)}) = \frac{p_{\mathsf{y}|\mathsf{z}}(y_i|z_i)\mathcal{CN}(z_i; \overline{z}_i^{(1)}, \overline{v}^{(1)})}{\int p_{\mathsf{y}|\mathsf{z}}(y_i|z_i')\mathcal{CN}(z_i'; \overline{z}_i^{(1)}, \overline{v}^{(1)})\,\mathrm{d}z_i'} \tag{3.34}$$

$$\triangleq p_{\mathsf{z}|\mathsf{y}}(z_i|y_i; \overline{z}_i^{(1)}, \overline{v}^{(1)}) \tag{3.35}$$

does not have analytically tractable mean or variance. Consequently, we employ the Laplace approximation [157], which assigns $\widehat{z}_i^{(1)}$ to the posterior mode (i.e., the MAP estimate) and $\widehat{v}_i^{(1)}$ to the trace of the inverse Hessian of $-\ln p_{\mathsf{z}|\mathsf{y}}(z_i|y_i; \overline{z}_i^{(1)}, \overline{v}^{(1)})$ at $z_i = \widehat{z}_i^{(1)}$, both of which have closed-form expressions. In particular, the MAP estimate of $z_i$ is given by [65]

$$\widehat{z}_i^{(1)} = \frac{\overline{v}^{(1)} y_i + 2v|\overline{z}_i^{(1)}|}{\overline{v}^{(1)} + 2v} e^{j\angle \overline{z}_i^{(1)}}, \tag{3.36}$$

and the trace-inverse-Hessian at $z_i = \widehat{z}_i^{(1)}$ is

$$\widehat{v}_i^{(1)} = \frac{\overline{v}^{(1)}(\overline{v}^{(1)} y_i + 4v|\overline{z}_i^{(1)}|)}{2|\overline{z}_i^{(1)}|(\overline{v}^{(1)} + 2v)} \tag{3.37}$$

as derived in Appendix E. The overall posterior variance $\widehat{v}^{(1)}$ is then set to the average value over all $i$, i.e., $\widehat{v}^{(1)} = \frac{1}{m}\sum_{i=1}^m \widehat{v}_i^{(1)}$.

## 3.3 Numerical Experiments

In this section, we present numerical experiments with Poisson-shot-noise-corrupted OSF and CDP phaseless measurements, comparing the performance of the proposed deepECpr algorithm to the classical HIO [49] and state-of-the-art prDeep [56], DeepITA (both "F" and "S" variants) [65], Diffusion Posterior Sampling (DPS) [76], and Diffusion mOdeL for PHase retrieval (DOLPH) [77] approaches. Our code is available at `https://github.com/Saurav-K-Shastri/deepECpr`.

Figure 3.1: The thirty $256 \times 256$ FFHQ test images.

### 3.3.1 Test Data and Denoisers

We test on three image datasets. As in the DPS paper [76], we use $256 \times 256$ color images from the FFHQ test partition [158]. In particular, we use the first thirty images, shown in Fig. 3.1. As in the Deep-ITA paper [65], we use the six "natural" $128 \times 128$ grayscale images shown in Fig. 3.2, and as in the prDeep paper [56], we use the six "unnatural" $128 \times 128$ grayscale images shown in Fig. 3.2.

Figure 3.2: The natural (left) and unnatural (right) grayscale test images.

When recovering the FFHQ test images, we use a denoiser that was constructed by rescaling (see [76, eq. (9)]) an unconditional DDPM diffusion model [84] trained on 49 000 FFHQ training images [159]. The resulting diffusion denoiser "$\boldsymbol{f}_{\mathsf{diff}}(\boldsymbol{r}_{\mathsf{noisy}}, v_{\mathsf{in}})$" accepts a noisy image $\boldsymbol{r}_{\mathsf{noisy}}$ and an input-noise variance $v_{\mathsf{in}}$, and thus is directly compatible with deepECpr, DOLPH, and DPS. The prDeep and Deep-ITA methods, however, do not have a way to estimate $v_{\mathsf{in}}$ and instead sequence through a bank of four "blind" denoisers designed to work at input-noise standard deviations (SD) of 60, 40, 20, 10 (relative to pixel values in $[0, 255]$), respectively. Thus, for prDeep and Deep-ITA, we use $\boldsymbol{f}_{\mathsf{diff}}(\boldsymbol{r}_{\mathsf{noisy}}, v_{\mathsf{in}})$ with $v_{\mathsf{in}} \in \{60^2, 40^2, 20^2, 10^2\}$ for FFHQ image recovery.

When recovering the grayscale images, we follow prDeep and Deep-ITA in using a bank of four blind DnCNN [27] denoisers $\{\boldsymbol{f}_{\mathsf{dncnn\text{-}60}}(\cdot), \boldsymbol{f}_{\mathsf{dncnn\text{-}40}}(\cdot), \boldsymbol{f}_{\mathsf{dncnn\text{-}20}}(\cdot), \boldsymbol{f}_{\mathsf{dncnn\text{-}10}}(\cdot)\}$. Although deepECpr would likely work better with a non-blind denoiser, we wanted a fair comparison with prDeep and Deep-ITA. These DnCNN denoisers were trained over the SD intervals $\{[40, 60], [20, 40], [10, 20], [0, 10]\}$, respectively, on the BSD400 [160] dataset using the bias-free approach from [142]. Note that our natural grayscale test

images are similar to (but distinct from) the BSD400 training data, while our unnatural grayscale test images could be considered out-of-distribution for this denoiser. For DOLPH and DPS, which are incompatible with blind denoisers, we use a pretrained ImageNet diffusion denoiser [161] and copied the grayscale measurements into all three color channels.

### 3.3.2 Measurement Generation

Like the prDeep and Deep-ITA papers, we employ OSF and CDP forward operators $\boldsymbol{A}$ (recall (3.28)-(3.29)) with $4\times$ oversampling and the (Gaussian approximated) Poisson shot-noise mechanism from [56]

$$y_i^2 = |z_i|^2 + w_i \text{ with } w_i \sim \mathcal{N}(0, \alpha^2|z_i|^2), \tag{3.38}$$

where $y_i^2/\alpha^2 \sim \mathcal{N}(|z_i|^2/\alpha^2, |z_i|^2/\alpha^2)$ is approximately Poisson$((|z_i|/\alpha)^2)$ for sufficiently small noise levels $\alpha$. Note that, because we scale $\boldsymbol{A}$ to have unit-norm columns, our $\alpha$ may have a different meaning than the $\alpha$ used in other papers. For example, with OSF measurements, our $\alpha$ must be scaled by $1/\sqrt{K} = 0.5$ to match the prDeep paper [56] or by $\sqrt{K} = 2$ to match the Deep-ITA paper [65]. With CDP measurements, our $\alpha$ must be scaled by $\sqrt{K} = 2$ to match the prDeep paper.

### 3.3.3 Performance Evaluation

We quantify performance using PSNR and SSIM [141] after resolving certain ambiguities fundamental to PR.

Phaseless Fourier measurements are unaffected by spatial translations, conjugate flips, and global phase [162]. The phase ambiguity is circumvented when the image pixels are non-negative, as with all of our test images. Furthermore, the translation

ambiguity is avoided when oversampling is used with images that have a sufficient number of non-zero edge pixels, as with our color FFHQ test images, but not our grayscale test images. Thus, when recovering the grayscale test images from OSF measurements, we correct for both translations and flips, but when recovering the color FFHQ test images, we correct for flips only. With color images, however, one further complication arises with HIO. Because HIO separately recovers each color channel, the flip ambiguity can manifest differently in each channel. To address this latter problem, we fix the first channel of the HIO recovery and flip of each remaining channel to maximize its correlation with the first channel.

Phaseless CDP measurements experience only a global-phase ambiguity [163], which is inconsequential with our non-negative test images, and so we perform no ambiguity resolution.

### 3.3.4 Algorithm Setup

For HIO, we ported the MATLAB implementation from [164] to Python and set the tunable step-size parameter to 0.9. For prDeep, we use the Python implementation from [165]. For Deep-ITA and DOLPH, we implemented the algorithm in Python/PyTorch since no public information is available. For DPS, we use the Python implementation from [159]. As in the prDeep and Deep-ITA papers, we use $p_{\mathsf{y}|\mathsf{z}}$ from (3.2) for all algorithms (except HIO, which does not use a likelihood). For deepECpr, we use damping factors $\mu^{(1)} = 0.3$ and $\mu^{(2)} = 0.075$ and initialization factor $\zeta = 1.2$. EM-based auto-tuning [166] is used to re-evaluate $\overline{v}^{(2)}$ for the first 3 and 10 iterations in the FFHQ and grayscale experiments, respectively. The initial noise level $\overline{v}_{\mathsf{init}}$ is set to $120^2$

for the FFHQ experiments and $70^2$ for the grayscale experiments. Experiment-specific settings are described later.

*Initialization/Reporting for OSF:* For the OSF experiments, we apply the initialization/reporting protocol used in both the prDeep and Deep-ITA papers to those algorithms, as well as to HIO and deepECpr. First, HIO is run 50 times, for 50 iterations each, from a random initialization. The candidate $\widehat{\boldsymbol{x}}$ with the lowest measurement residual $\|\boldsymbol{y} - |\boldsymbol{A}\boldsymbol{x}|\|$ is then selected as the HIO initialization, after which HIO is run for 1000 iterations. In the case of color images, the second and third channels are flipped to best match the first. Finally, the resulting HIO output is used to initialize prDeep, Deep-ITA, and deepECpr. The entire procedure is repeated three times, and the reconstruction with the lowest measurement residual is reported as the final output for each algorithm. For DPS and DOLPH, we follow the DPS authors' recommendation to run the algorithm four times and report the reconstruction with the lowest measurement residual.

*Initialization/Reporting for CDP:* For the CDP experiments, we use the initialization $\widehat{\boldsymbol{x}}_{\mathsf{init}} = \boldsymbol{A}^{\mathsf{H}}(\boldsymbol{y}e^{j\angle \boldsymbol{A}\mathbf{1}})$ for HIO, prDeep, Deep-ITA, and deepECpr, where $e^{j\angle \boldsymbol{A}\mathbf{1}}$ is computed and applied element-wise. Each algorithm was run only once.

### 3.3.5 OSF Phase Retrieval of FFHQ Images

We first investigate the recovery of FFHQ test images from noisy, phaseless OSF measurements. For prDeep and Deep-ITA, we sequence through the denoisers $\boldsymbol{f}_{\mathsf{diff}}(\cdot, 60^2)$, $\boldsymbol{f}_{\mathsf{diff}}(\cdot, 40^2)$, $\boldsymbol{f}_{\mathsf{diff}}(\cdot, 20^2)$, and $\boldsymbol{f}_{\mathsf{diff}}(\cdot, 10^2)$ for 300 iterations each, for a total of 1200 iterations, as recommended by the authors. Also, we set the $\lambda$ parameter to $0.1\sqrt{v}$ for prDeep, and $0.05\sqrt{v}$ for Deep-ITA, to maximize validation PSNR. For DPS,

Table 3.1: Average PSNR and SSIM for FFHQ phase retrieval with OSF and CDP operators and noise level $\alpha$.

| method | OSF | | | | | | CDP | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\alpha = 4$ | | $\alpha = 6$ | | $\alpha = 8$ | | $\alpha = 5$ | | $\alpha = 15$ | | $\alpha = 45$ | |
| | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| HIO | 27.37 | 0.6759 | 26.08 | 0.6163 | 25.03 | 0.5664 | 36.48 | 0.9140 | 26.79 | 0.6064 | 17.58 | 0.2330 |
| Deep-ITA-F | 35.05 | 0.9420 | 34.94 | 0.9374 | 34.50 | 0.9321 | 41.69 | 0.9786 | 37.20 | 0.9556 | 29.93 | 0.8402 |
| Deep-ITA-S | 34.01 | 0.9365 | 34.54 | 0.9367 | **35.15** | **0.9412** | 41.87 | 0.9795 | 37.15 | 0.9531 | 27.28 | 0.7817 |
| prDeep | 37.69 | 0.9654 | 35.28 | 0.9523 | 33.68 | 0.9410 | 42.43 | 0.9816 | 37.48 | 0.9584 | 23.14 | 0.4477 |
| DOLPH | 14.65 | 0.3426 | 14.65 | 0.3420 | 14.64 | 0.3393 | 40.74 | 0.9755 | 33.84 | 0.8806 | 20.98 | 0.3541 |
| DPS | 27.22 | 0.7674 | 25.84 | 0.7530 | 24.57 | 0.7408 | 41.52 | 0.9786 | 35.68 | 0.9381 | 30.12 | 0.8428 |
| deepECpr (proposed) | **39.75** | **0.9720** | **37.01** | **0.9567** | 34.86 | 0.9404 | **43.12** | **0.9846** | **37.55** | **0.9589** | **32.15** | **0.8941** |

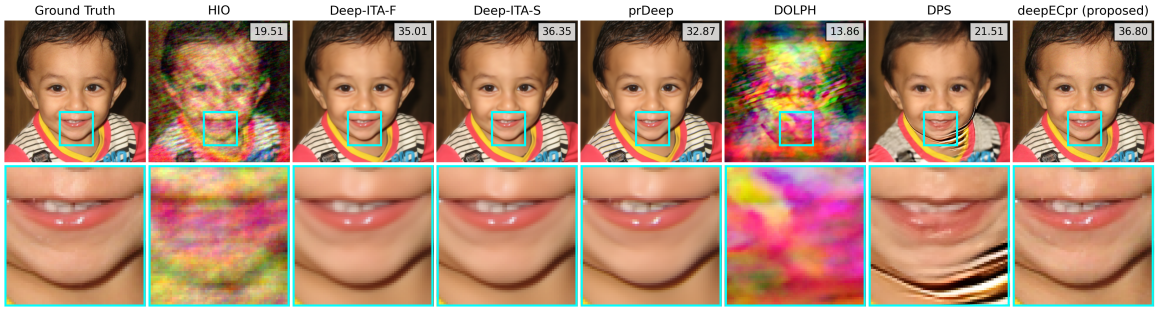


Figure 3.3: Top: Example FFHQ image recoveries from phaseless OSF measurements at noise level $\alpha = 8$, with PSNR indicated in the top right corner of each image. Bottom: Zoomed versions of the cyan regions in the top row. Note that the HIO, DOLPH, and DPS recoveries contain strong artifacts and that Deep-ITA and prDeep show oversmoothing.

we use 1000 steps as recommended by the authors and set the scale parameter to 0.0075 to maximize validation PSNR. Similarly, we use 1000 steps for DOLPH and set the step-size parameter to $10^{-6}$ to maximize validation PSNR. For deepECpr, we use $\boldsymbol{f}_{\mathsf{diff}}$ as specified in Alg. 5 for 200 iterations with denoising factor $\beta = 0.15$.

Table 3.1 shows the average PSNR and SSIM at noise levels $\alpha \in \{4, 6, 8\}$. There, deepECpr outperforms the other methods in both PSNR and SSIM at $\alpha \in \{4, 6\}$, where the PSNR advantage is significant ($\approx 2$ dB). At $\alpha = 8$, deepECpr trails the

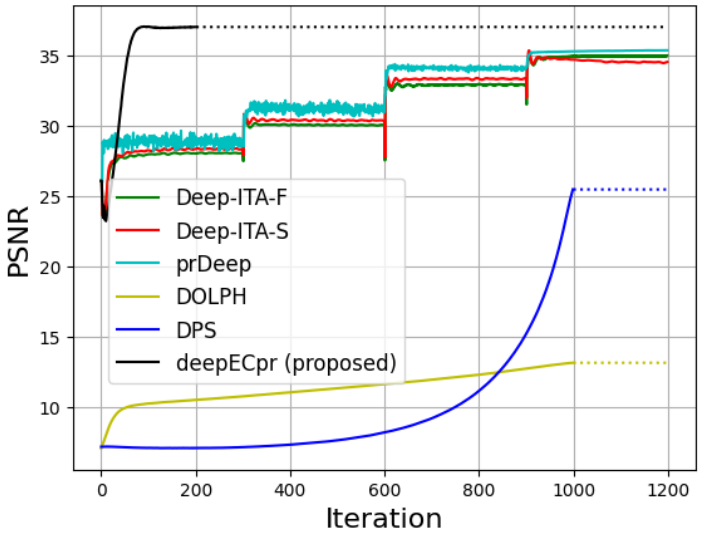


Figure 3.4: Average PSNR versus iteration for FFHQ phase retrieval from OSF measurements at noise level $\alpha = 6$.

best-performing technique, Deep-ITA-S, by a small margin: 0.29 dB in PSNR and 0.0008 in SSIM.

Figure 3.3 shows example recoveries at $\alpha = 8$. There we see severe artifacts in the HIO, DOLPH, and DPS recoveries. Although the prDeep, Deep-ITA-F, Deep-ITA-S, and deepECpr approaches avoid unwanted artifacts, the proposed deepECpr does the best job of recovering fine details in the ground-truth image.

Figure 3.4 plots PSNR (averaged over the FFHQ test images) versus iteration at $\alpha = 6$. Note that the prDeep, Deep-ITA, DOLPH, DPS, and deepECpr algorithms all call the denoiser once per iteration and thus have similar per-iteration complexities. From the figure, we see that deepECpr converges significantly faster than its competitors.

Figure 3.5 plots the standard-deviations (SDs) of the deepECpr-estimated and true errors in the $\boldsymbol{z}$ estimates for an typical OSF recovery at noise level $\alpha = 8$. In particular, it plots $\sqrt{\overline{v}^{(i)}}$ in comparison to the true error $m^{-1/2}\|\overline{\boldsymbol{z}}^{(i)} - \boldsymbol{z}_{\mathsf{true}}\|$, and $\sqrt{\widehat{v}^{(i)}}$

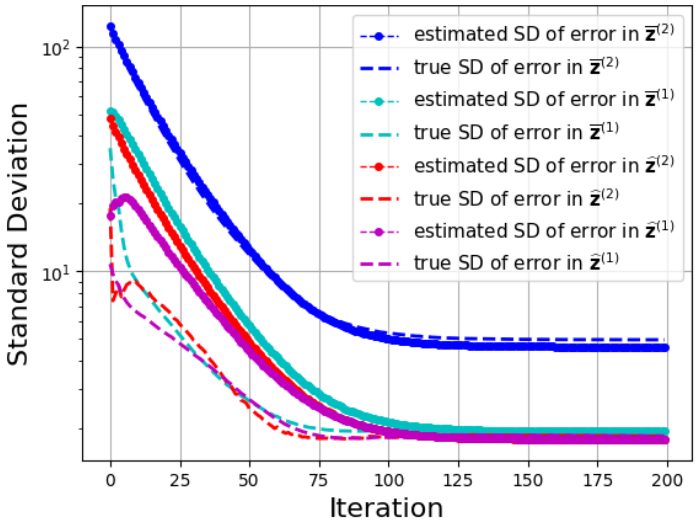


Figure 3.5: Evolution of the true and estimated $\boldsymbol{z}$ errors in deepECpr for OSF phase retrieval at noise level $\alpha = 8$.

in comparison to the true error $m^{-1/2}\|\widehat{\boldsymbol{z}}^{(i)} - \boldsymbol{z}_{\text{true}}\|$, for $i \in \{1, 2\}$. The figure shows the accuracy of $\overline{v}^{(2)}$ over all iterations and the accuracy of the other estimates after iteration 100. Furthermore, it shows the EC fixed-point condition $\widehat{v}^{(1)} = \widehat{v}^{(2)}$ being satisfied.

Table 3.2 presents an ablation study comparing the proposed stochastic damping (3.14) to the deterministic damping (3.13) when used in line 13 of Alg. 5. With stochastic damping, deepECpr yields higher PSNR and SSIM values at all tested noise levels $\alpha$. Figure 3.7 plots the SD of the AWGN noise injected by stochastic damping as a function of the deepECpr iteration. The plot shows that the SD starts high but decreases to zero with the iterations.

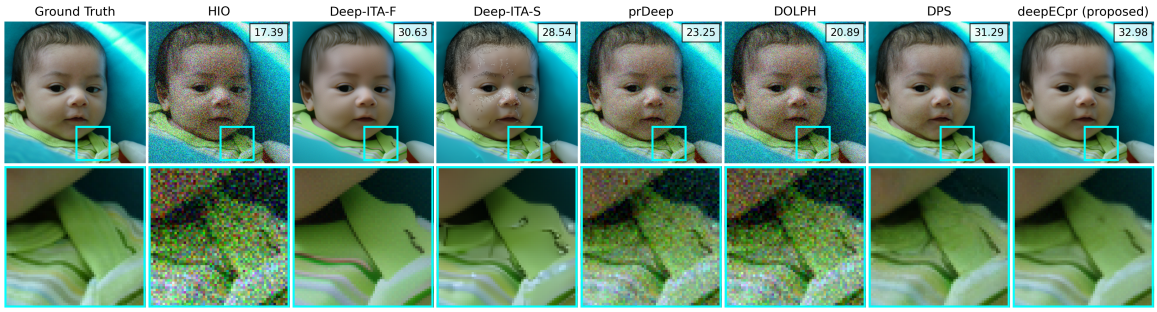


Figure 3.6: Top: Example FFHQ image recoveries from phaseless CDP measurements at noise level $\alpha = 45$, with PSNR indicated in the top right corner of each image. Bottom: Zoomed versions of the cyan regions in the top row. Note that the HIO, prDeep, DOLPH recoveries are very noisy and that Deep-ITA is plagued by both oversmoothing and hallucinations.

Table 3.2: PSNR and SSIM of deterministic versus stochastic damping for FFHQ OSF phase retrieval under noise level $\alpha$

| type of damping used in line 13 of Alg. 5 | $\alpha = 4$ | | $\alpha = 6$ | | $\alpha = 8$ | |
|---|---|---|---|---|---|---|
| | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| deterministic damping (3.13) | 38.05 | 0.9565 | 36.3 | 0.9468 | 34.20 | 0.9267 |
| stochastic damping (3.14) | 39.75 | 0.9720 | 37.01 | 0.9567 | 34.86 | 0.9404 |

### 3.3.6 CDP Phase Retrieval of FFHQ Images

We now investigate the recovery of FFHQ test images from noisy, phaseless CDP measurements. Deep-ITA, prDeep, DOLPH, DPS, and deepECpr are configured as described in Section 3.3.5, except that the Deep-ITA uses $\lambda = 0.01\sqrt{v}$, DOLPH uses stepsize $5 \times 10^{-6}$, and DPS uses scale parameter 0.05, in all three cases to maximize validation PSNR. deepECpr uses the denoising factor $\beta = 0.002$.

Table 3.1 shows the average PSNR and SSIM values at noise levels $\alpha \in \{5, 15, 45\}$. There we see that deepECpr outperforms the other methods in both PSNR and SSIM at all $\alpha$ and gives a PSNR advantage of $> 2$ dB at $\alpha = 45$.

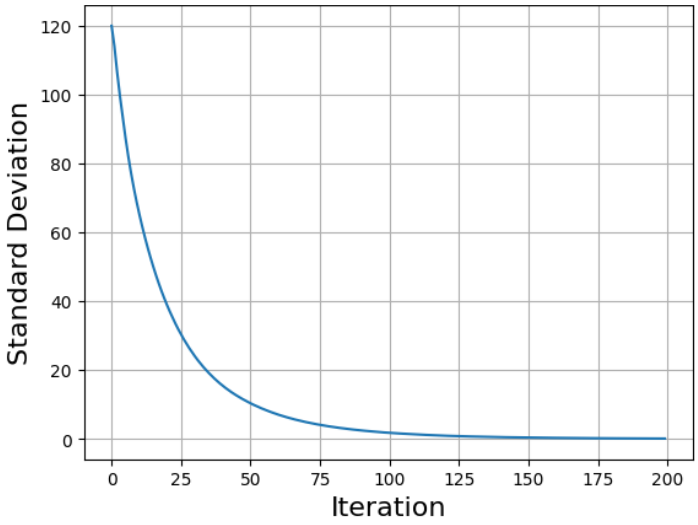


Figure 3.7: Standard deviation of the noise added by deepECpr's stochastic damping scheme versus iteration for an example of FFHQ phase retrieval with OSF measurements at noise level $\alpha = 8$.

Figure 3.6 shows example recoveries at $\alpha = 45$. There we see that HIO, prDeep, and DOLPH fall prey to noise-like artifacts. Meanwhile, Deep-ITA-F over-smooths the image, while Deep-ITA-S and DPS introduce hallucinations. Finally, deepECpr does the best job of faithfully recovering the true image.

Figure 3.8 plots the PSNR (averaged over the FFHQ test images) versus iteration at $\alpha = 5$. As before, deepECpr converges significantly faster than its competitors.

### 3.3.7 CDP Phase Retrieval of Grayscale Images

We now investigate the recovery of natural and unnatural grayscale test images from noisy, phaseless CDF measurements. The algorithms under test are configured as described in Section 3.3.4. In addition, prDeep uses the author-recommended $\lambda = 0.1\sqrt{v}$, while Deep-ITA uses $\lambda = 0.25\sqrt{v}$ to maximize validation PSNR. For deepECpr, we use the same blind DnCNN denoisers as prDeep and Deep-ITA, with $\beta =$

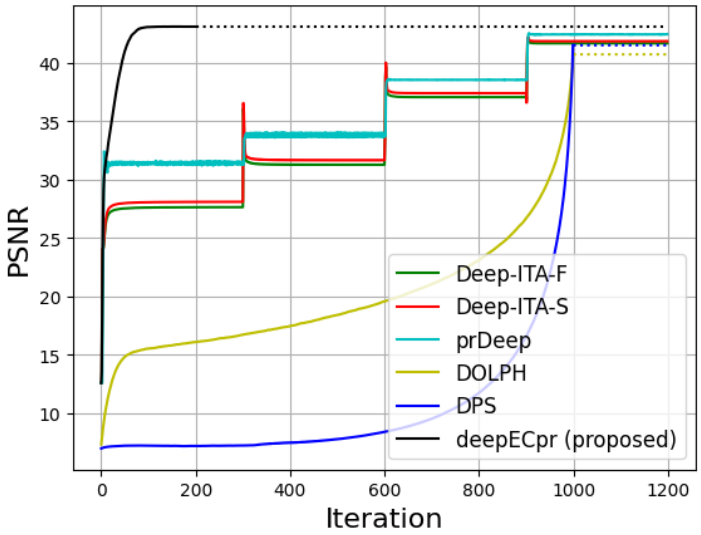


Figure 3.8: Average PSNR versus iteration for FFHQ phase retrieval from CDP measurements at noise level $\alpha = 5$.

$0.045, 0.035, 0.025, 0.020$ for $\boldsymbol{f}_{\text{dncnn-60}}, \boldsymbol{f}_{\text{dncnn-40}}, \boldsymbol{f}_{\text{dncnn-20}}, \boldsymbol{f}_{\text{dncnn-10}}$, respectively. DOLPH and DPS utilize the ImageNet diffusion denoiser [161] with step-size of $5 \times 10^{-6}$ and a scale parameter of 0.025, respectively, to maximize validation PSNR.

Table 3.3 shows the average PSNR and SSIM values at noise levels $\alpha \in \{5, 15, 45\}$. There deepECpr achieves the highest PSNR and SSIM values at all tested noise levels for both natural and unnatural grayscale images.

Figure 3.9 shows example recoveries at $\alpha = 15$. There we see that deepECpr's recovery looks cleaner and sharper than those of the competing methods.

### 3.3.8 OSF Phase Retrieval of Grayscale Images

Finally, we investigate the recovery of natural and unnatural grayscale test images from noisy, phaseless OSF measurements. The algorithms are set up as described in Section 3.3.7, except that prDeep and Deep-ITA use the author-recommended choices

Table 3.3: Average PSNR and SSIM for natural and unnatural grayscale image phase retrieval with CDP measurements at noise level $\alpha$.

| method | natural | | | | | | unnatural | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\alpha = 5$ | | $\alpha = 15$ | | $\alpha = 45$ | | $\alpha = 5$ | | $\alpha = 15$ | | $\alpha = 45$ | |
| | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| HIO | 36.39 | 0.9541 | 26.65 | 0.7663 | 17.13 | 0.4155 | 36.45 | 0.8891 | 26.94 | 0.6133 | 18.55 | 0.3175 |
| Deep-ITA-F | 38.67 | 0.9783 | 29.57 | 0.8595 | 22.77 | 0.6301 | 39.42 | 0.9607 | 29.80 | 0.7526 | 21.40 | 0.4352 |
| Deep-ITA-S | 38.80 | 0.9797 | 28.56 | 0.8357 | 17.25 | 0.4260 | 39.80 | 0.9679 | 28.56 | 0.7077 | 18.74 | 0.3211 |
| prDeep | 38.73 | 0.9785 | 32.49 | 0.9388 | 26.38 | 0.8167 | 39.63 | 0.9660 | 33.84 | 0.9267 | 27.57 | 0.8220 |
| DOLPH | 39.19 | 0.9738 | 28.37 | 0.7435 | 17.63 | 0.3336 | 40.40 | 0.9708 | 29.11 | 0.6571 | 19.28 | 0.2797 |
| DPS | 37.66 | 0.9617 | 32.07 | 0.9038 | 26.76 | 0.7634 | 39.05 | 0.9603 | 33.15 | 0.8978 | 27.22 | 0.7423 |
| deepECpr (proposed) | **39.46** | **0.9845** | **32.75** | **0.9439** | **27.06** | **0.8428** | **40.92** | **0.9808** | **34.17** | **0.9356** | **28.14** | **0.8292** |

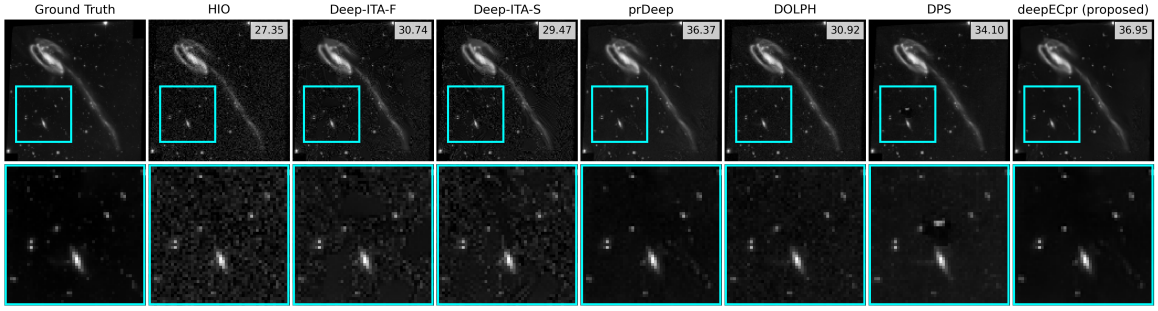


Figure 3.9: Top: Example unnatural image recoveries from phaseless CDP measurements at noise level $\alpha = 15$, with PSNR indicated in the top right corner of each image. Bottom: Zoomed versions of the cyan regions in the top row. Note that the HIO, Deep-ITA, and DOLPH recoveries are visibly noisy, and that DPS has hallucinated a bright spot in the middle of the image.

$\lambda = \sqrt{v}$ and $\lambda = 0.025\sqrt{v}$, respectively, and deepECpr uses $\beta = 0.45, 0.35, 0.25, 0.20$ for $\boldsymbol{f}_{\mathsf{dncnn\text{-}60}}, \boldsymbol{f}_{\mathsf{dncnn\text{-}40}}, \boldsymbol{f}_{\mathsf{dncnn\text{-}20}}, \boldsymbol{f}_{\mathsf{dncnn\text{-}10}}$ over $150, 100, 75, 75$ iterations, respectively. Additionally, DOLPH employs a step-size of $5 \times 10^{-5}$, while DPS uses a scale parameter of 0.05, each to maximize validation PSNR.

Table 3.4 shows the average PSNR and SSIM values at noise levels $\alpha \in \{4, 6, 8\}$. There, for both natural and unnatural test images, deepECpr achieves the highest

Table 3.4: Average PSNR and SSIM for natural and unnatural grayscale test images with OSF phase retrieval at noise level $\alpha$.

| method | natural | | | | | | unnatural | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\alpha = 4$ | | $\alpha = 6$ | | $\alpha = 8$ | | $\alpha = 4$ | | $\alpha = 6$ | | $\alpha = 8$ | |
| | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| HIO | 24.32 | 0.6932 | 22.75 | 0.6267 | 21.63 | 0.5702 | 26.66 | 0.6221 | 25.10 | 0.5653 | 24.04 | 0.5231 |
| Deep-ITA-F | 31.19 | 0.8996 | 29.64 | 0.8642 | 28.35 | 0.8278 | 28.30 | 0.6904 | 26.76 | 0.6466 | 26.58 | 0.6450 |
| Deep-ITA-S | 30.98 | 0.8933 | 29.71 | 0.8603 | 28.78 | 0.8343 | 28.20 | 0.6894 | 26.96 | 0.6472 | 26.75 | 0.6390 |
| prDeep | 35.71 | 0.9632 | 32.56 | 0.9242 | 29.57 | 0.8670 | 30.22 | 0.7533 | 26.66 | 0.6630 | 25.96 | 0.6297 |
| DOLPH | 16.65 | 0.3331 | 16.63 | 0.3234 | 16.61 | 0.3208 | 19.25 | 0.4226 | 19.27 | 0.4025 | 19.08 | 0.3833 |
| DPS | 14.65 | 0.2177 | 14.22 | 0.2014 | 14.19 | 0.1851 | 17.19 | 0.3168 | 15.66 | 0.2742 | 15.75 | 0.2774 |
| deepECpr (proposed) | **37.30** | **0.9766** | **34.06** | **0.9546** | **31.85** | **0.9302** | **30.98** | **0.7675** | **27.45** | **0.6897** | **27.09** | **0.6780** |

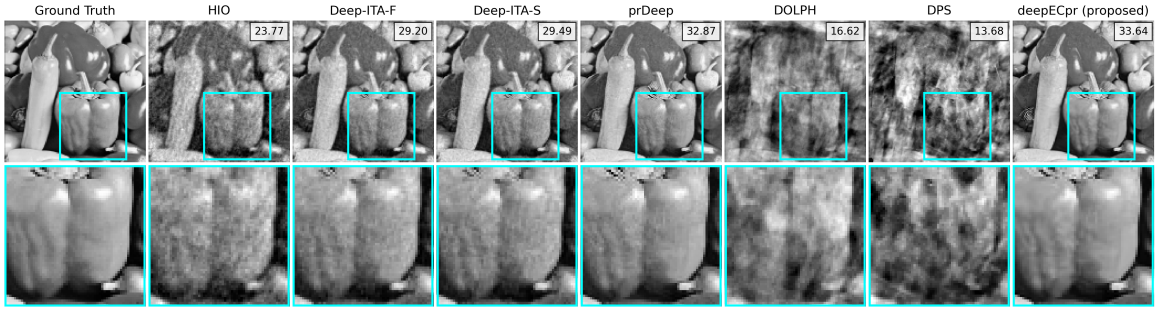


Figure 3.10: Top: Example natural image recoveries from phaseless OSF measurements at noise level $\alpha = 8$, with PSNR indicated in the top right corner of each image. Bottom: Zoomed versions of the cyan regions in the top row. Note that the HIO, Deep-ITA, DOLPH, and DPS recoveries contain strong artifacts, while the prDeep recovery shows an overly rough texture.

PSNR and SSIM at all tested noise levels. For natural images, deepECpr's PSNR exceeds the second-best PSNR by $\geq 1.5$ dB.

Figure 3.10 shows example recoveries at $\alpha = 8$. There we see that deepECpr recovers the fine details and texture of the original image, while the other approaches show noise-like artifacts and overly rough textures.

## 3.4 Conclusion

In this chapter, we proposed "deepECpr," which combines expectation-consistent (EC) approximation with deep denoising networks to advance the state-of-the-art in phase retrieval. In addition to a non-traditional application of EC, deepECpr employs a novel stochastic damping scheme. Experimental results with oversampled-Fourier and coded-diffraction-pattern operators, and with color, natural, and unnatural grayscale images, demonstrate deepECpr's advantages in image recovery from noise-corrupted phaseless measurements. In almost all cases, deepECpr yielded improved PSNR and SSIM over the state-of-the-art prDeep, Deep-ITA, DOLPH, and DPS methods with significantly fewer denoiser calls. Although not explicitly tested, the proposed deepEC approach should be directly applicable to other generalized-linear-model problems like dequantization, logistic regression, and image recovery in non-Gaussian noise.

# Chapter 4: Solving Inverse Problem using Recursive Annealed Posterior Sampling

## 4.1 Background

In diffusion models, the forward process transforms a complex distribution $p(\boldsymbol{x}_0)$ a simple distribution (e.g., Gaussian) by gradually adding noise, and the reverse process learns to gradually denoise a sample from that simple distribution to finally obtain a sample from $p(\boldsymbol{x}_0)$. The forward process is often defined by a stochastic differential equation (SDE) [85] of the form

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{w}_t, \tag{4.1}$$

for some choices of $\boldsymbol{f}(\cdot, t)$ and $g(\cdot)$. In (4.1), $\boldsymbol{x}_t \in \mathbb{R}^d$ is the sample at time $t \in [0, T]$ and $\boldsymbol{w}_t$ is the standard Wiener process (SWP) (i.e., Brownian motion). The reverse process can then be described by [167]

$$\mathrm{d}\boldsymbol{x}_t = \big(\boldsymbol{f}(\boldsymbol{x}_t, t) - g^2(t)\nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{x}_t)\big)\,\mathrm{d}t + g(t)\,\mathrm{d}\overline{\boldsymbol{w}}_t, \tag{4.2}$$

where $p(\boldsymbol{x}_t)$ is the distribution of $\boldsymbol{x}_t$ and $\overline{\boldsymbol{w}}_t$ is the SWP run backwards. The so-called score function $\nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{x}_t)$ can be approximated using a neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ trained by score matching [168]

$$\arg\min_{\boldsymbol{\theta}} \mathrm{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0), \boldsymbol{x}_t \sim p(\boldsymbol{x}_t|\boldsymbol{x}_0), t}\{\lambda_t \|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{x}_t)\|^2\} \tag{4.3}$$

for some positive weighting function $\lambda_t$.

Above, $\boldsymbol{f}(\cdot, t)$ and $g(\cdot)$ are design choices. One famous example is the variance preserving (VP) SDE, where $\boldsymbol{f}(\boldsymbol{x}_t, t) = -\frac{1}{2}\beta_t \boldsymbol{x}_t$ and $g_t = \sqrt{\beta_t}$ for some variance schedule $\beta_t$. In practice, the VP-SDE is discretized over $i = 0, \ldots, N$ with forward and reverse processes

$$\boldsymbol{x}_i = \sqrt{1-\beta_i}\boldsymbol{x}_{i-1} + \sqrt{\beta_i}\boldsymbol{z}_{i-1}, \quad \{\boldsymbol{z}_i\} \overset{i.i.d}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{4.4}$$

$$\boldsymbol{x}_{i-1} = \frac{1}{\sqrt{\alpha_i}}\big[\boldsymbol{x}_i + \beta_i \nabla_{\boldsymbol{x}_i} \ln p(\boldsymbol{x}_i)\big] + \sigma_i \boldsymbol{z}_i, \tag{4.5}$$

with $\alpha_i \triangleq 1 - \beta_i$ and $\overline{\alpha}_i \triangleq \prod_{j=1}^{i} \alpha_j$ and $\sigma_i^2 \triangleq \frac{1-\overline{\alpha}_{i-1}}{1-\overline{\alpha}_i}\beta_i$, where it is known as DDPM [84]. Another choice is the variance exploding (VE) SDE, where $\boldsymbol{f}(\boldsymbol{x}_t, t) = \boldsymbol{0}$ and $g(t) = \sqrt{\mathrm{d}[\sigma_t^2]/\,\mathrm{d}t}$ for some variance schedule $\sigma_t^2$. In practice, the VE-SDE is discretized over $i = 0, \ldots, N$ with forward and reverse processes

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2}\boldsymbol{z}_i, \quad \{\boldsymbol{z}_i\} \overset{i.i.d}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{4.6}$$

$$\boldsymbol{x}_i = \boldsymbol{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2)\nabla_{\boldsymbol{x}_{i+1}} \ln p(\boldsymbol{x}_{i+1}) + \sqrt{\frac{\sigma_i^2(\sigma_{i+1}^2 - \sigma_i^2)}{\sigma_{i+1}^2}}\boldsymbol{z}_{i+1}, \tag{4.7}$$

where it is known as SMLD [83].

For either the VP-SDE or VE-SDE (and in fact for any affine $\boldsymbol{f}(\cdot, t)$), the pdf $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ is Gaussian. For example, with the VE-SDE we have $p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}_0, \sigma_t^2\boldsymbol{I})$. In this case, the score can be expressed using Tweedie's formula [94] as

$$\nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{x}_t) = \frac{\mathrm{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t\} - \boldsymbol{x}_t}{\sigma_t^2}, \tag{4.8}$$

where $\mathrm{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t\}$ is the minimum mean-squared error (MMSE) denoiser of $\boldsymbol{x}_t = \boldsymbol{x}_0 + \sigma_t\boldsymbol{n}$ for $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Thus, as an alternative to score-matching (4.3), one can approximate

$\mathrm{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t\}$ by a denoising network $\boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \sigma_t)$ trained via [169]

$$\arg\min_{\boldsymbol{\theta}} \mathrm{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0),\boldsymbol{n}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),t} \left\{\lambda_t' \|\boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_0 + \sigma_t\boldsymbol{n}, \sigma_t) - \boldsymbol{x}_0\|^2\right\}, \tag{4.9}$$

for positive weighting function $\lambda_t' = \lambda_t/\sigma_t^4$, and then set the score approximation as

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t) = \frac{\boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \sigma_t) - \boldsymbol{x}_t}{\sigma_t^2}. \tag{4.10}$$

When applied in discretized form to the SMLD reverse process, this yields

$$\boldsymbol{x}_i = \boldsymbol{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2)\left(\frac{\boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i+1}, \sigma_{i+1}) - \boldsymbol{x}_{i+1}}{\sigma_{i+1}^2}\right) + \sqrt{\frac{\sigma_i^2(\sigma_{i+1}^2 - \sigma_i^2)}{\sigma_{i+1}^2}}\boldsymbol{z}_{i+1} \tag{4.11}$$

$$= \frac{\sigma_i^2}{\sigma_{i+1}^2}\boldsymbol{x}_{i+1} + \left(1 - \frac{\sigma_i^2}{\sigma_{i+1}^2}\right)\boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i+1}, \sigma_{i+1}) + \sqrt{\frac{\sigma_i^2(\sigma_{i+1}^2 - \sigma_i^2)}{\sigma_{i+1}^2}}\boldsymbol{z}_{i+1}. \tag{4.12}$$

Another widely used approach to diffusion involves a deterministic formulation. For every forward process defined in (4.1), Song et al. [83] introduce a corresponding deterministic process:

$$\mathrm{d}\boldsymbol{x}_t = \left(\boldsymbol{f}(\boldsymbol{x}_t, t) - \tfrac{1}{2}g^2(t)\nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{x}_t)\right)\mathrm{d}t, \tag{4.13}$$

which preserves the same marginal distribution $p(\boldsymbol{x}_t)$ at all times $t$ as the SDE in (4.2). This deterministic process, described by (4.13), is referred to as the probability flow ordinary differential equation (ODE) since it governs the continuous evolution of probability distributions in a non-stochastic manner. For a specific choice of $\boldsymbol{f}(\boldsymbol{x}_t, t) = \mathbf{0}$ and $g(t) = \sqrt{2\sigma_t\,\mathrm{d}\sigma_t/\,\mathrm{d}t}$, where $\sigma_t^2$ follows a predefined variance schedule, the probability flow ODE simplifies to:

$$\mathrm{d}\boldsymbol{x}_t = -\frac{\mathrm{d}\sigma_t}{\mathrm{d}t}\sigma_t\nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{x}_t)\,\mathrm{d}t. \tag{4.14}$$

This ODE can be solved using various numerical integration schemes and choices of discrete sampling time steps [85–87]. When $\mathrm{d}t$ is discretized over $i = 0, \ldots, N$ using

the Euler method, the reverse process becomes [85]

$$\boldsymbol{x}_i = \boldsymbol{x}_{i+1} + \frac{1}{2}(\sigma_i^2 - \sigma_{i+1}^2)\nabla_{\boldsymbol{x}_{i+1}} \ln p(\boldsymbol{x}_{i+1}). \tag{4.15}$$

If instead $\mathrm{d}\sigma_t$ is discretized using the Euler method, one gets the deterministic variant of DDIM [86]:

$$\boldsymbol{x}_i = \boldsymbol{x}_{i+1} + (\sigma_i - \sigma_{i+1})\sigma_{i+1}\nabla_{\boldsymbol{x}_{i+1}} \ln p(\boldsymbol{x}_{i+1}). \tag{4.16}$$

When (4.10) is applied to the ODE reverse process (4.16), we get

$$\boldsymbol{x}_i = \frac{\sigma_i}{\sigma_{i+1}}\boldsymbol{x}_{i+1} + \left(1 - \frac{\sigma_i}{\sigma_{i+1}}\right)\boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i+1}, \sigma_{i+1}). \tag{4.17}$$

The stochastic EDM algorithm [87] generalizes this deterministic ODE reverse process (4.17) to a stochastic rule parameterized by $\gamma_i \geq 0$:

$$\boldsymbol{x}_i = \frac{\sigma_i}{(1+\gamma_i)\sigma_{i+1}}\boldsymbol{x}_{i+1} + \left(1 - \frac{\sigma_i}{(1+\gamma_i)\sigma_{i+1}}\right)\boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i+1}, \sigma_{i+1}) + \sqrt{\sigma_i^2 - \frac{\sigma_i^2}{(1+\gamma_i)^2}}\boldsymbol{z}_{i+1}, \tag{4.18}$$

where $\gamma_i = 0$ recovers the deterministic update (4.17).

Equations (4.12), (4.17), and (4.18) are all self-consistent, in that $p(\boldsymbol{x}_i|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{x}_0, \sigma_i^2\boldsymbol{I})$ guarantees $p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{i+1}; \boldsymbol{x}_0, \sigma_{i+1}^2\boldsymbol{I})$ when the denoiser is perfect, i.e., when $\boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i+1}, \sigma_{i+1}) = \boldsymbol{x}_0$. Thus stochastic EDM (4.18) can also be seen as a generalization of SMLD (4.12) that agrees with SMLD when

$$\frac{\sigma_i}{(1+\gamma_i)\sigma_{i+1}} = \frac{\sigma_i^2}{\sigma_{i+1}^2} \quad \Leftrightarrow \quad \gamma_i = \frac{\sigma_{i+1}}{\sigma_i} - 1. \tag{4.19}$$

Through the choice of $\gamma_i$, stochastic EDM can traverse a range of stochasticities from none (i.e., ODE) to fully stochastic (i.e., much more stochastic than SMLD).

Diffusion methods can be used to solve inverse problems, where one observes measurements $\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{x}_0)$ and seeks to sample from the posterior $p(\boldsymbol{x}_0|\boldsymbol{y})$. For

this application, the methodology is similar to above, but with the score function $\nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{x}_t)$ replaced by the conditional score function $\nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{x}_t|\boldsymbol{y})$, which can be decomposed using Bayes rule as

$$\nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{x}_t|\boldsymbol{y}) = \nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \ln p(\boldsymbol{y}|\boldsymbol{x}_t). \tag{4.20}$$

The latter term is non-trivial to compute because $p(\boldsymbol{y}|\boldsymbol{x}_t) = \int p(\boldsymbol{y}|\boldsymbol{x}_0)p(\boldsymbol{x}_0|\boldsymbol{x}_t)\,\mathrm{d}\boldsymbol{x}_t$ with a $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ that is difficult to characterize. Hence, various approximations have been made, such as $p(\boldsymbol{x}_0|\boldsymbol{x}_t) \approx \delta\big(\boldsymbol{x}_0 - \boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \sigma_t)\big)$ in DPS [76] and $p(\boldsymbol{x}_0|\boldsymbol{x}_t) \approx \mathcal{N}(\boldsymbol{x}_0; \boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \sigma_t), \sigma_{0|t}^2\boldsymbol{I})$ for some $\sigma_{0|t}^2$ in ΠGDM [93]. Many other approximation exist; see [88] for a survey.

An important observation from (4.4) and (4.6) is that, in the typical case that $N$ is large, both $\beta_i$ and $\sigma_{i+1}^2 - \sigma_i^2$ are small, and so the amount of stochasticity injected into $\boldsymbol{x}_i$ relative to $\boldsymbol{x}_{i-1}$ is quite small. Consequently, the $\{\boldsymbol{x}_i\}$ trajectories that arise in a given instance of the reverse process don't explore the full range of possibilities afforded by $p(\boldsymbol{x}_t|\boldsymbol{y})$. The authors of [97] conjectured that this behavior leads to a lack of diversity when existing diffusion methods are used as approximate posterior samplers. As an alternative, they proposed a "decoupled" forward process over $i \in \{1, \ldots, N\}$ with the form

$$\boldsymbol{x}_i = \boldsymbol{x}_0 + \sigma_i \boldsymbol{n}_i, \quad \{\boldsymbol{n}_i\}_{i=1}^N \overset{\text{i.i.d}}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{4.21}$$

with $\sigma_i$ decreasing in $i$. Importantly, $\{\boldsymbol{x}_i\}_{i=1}^N$ in (4.21) are mutually independent conditional on $\boldsymbol{x}_0$, i.e., $p(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) \propto p(\boldsymbol{x}_0) \prod_{i=1}^N p(\boldsymbol{x}_i|\boldsymbol{x}_0)$. This dependency structure is markedly different from traditional forward processes like (4.4) and (4.6), where $p(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) \propto p(\boldsymbol{x}_0) \prod_{i=1}^N p(\boldsymbol{x}_i|\boldsymbol{x}_{i-1})$. Consequently, the design of the reverse process must be reconsidered.

The authors of [97] observed that, for any steps $i$ and $k \neq i$,

$$p(\boldsymbol{x}_i|\boldsymbol{y}) = \int\int \underbrace{p(\boldsymbol{x}_i|\boldsymbol{x}_0, \boldsymbol{x}_k, \boldsymbol{y})}_{\mathcal{N}(\boldsymbol{x}_i; \boldsymbol{x}_0, \sigma_i^2 \boldsymbol{I})} p(\boldsymbol{x}_0|\boldsymbol{x}_k, \boldsymbol{y}) p(\boldsymbol{x}_k|\boldsymbol{y}) \, \mathrm{d}\boldsymbol{x}_0 \, \mathrm{d}\boldsymbol{x}_k \tag{4.22}$$

which suggests a three-step approach to drawing a sample $\widetilde{\boldsymbol{x}}_i$ from $p(\boldsymbol{x}_i|\boldsymbol{y})$: first draw $\widetilde{\boldsymbol{x}}_k \sim p(\boldsymbol{x}_k|\boldsymbol{y})$, then draw $\widetilde{\boldsymbol{x}}_{0|k}$ from $p(\boldsymbol{x}_0|\boldsymbol{x}_k, \boldsymbol{y})|_{\boldsymbol{x}_k = \widetilde{\boldsymbol{x}}_k}$, and finally draw $\widetilde{\boldsymbol{x}}_i$ from $\mathcal{N}(\widetilde{\boldsymbol{x}}_{0|k}, \sigma_i^2 \boldsymbol{I})$. This led them to propose a reverse process that starts by sampling $\widetilde{\boldsymbol{x}}_N$ from $\mathcal{N}(\boldsymbol{0}, \sigma_N^2 \boldsymbol{I}) \approx p(\boldsymbol{x}_N|\boldsymbol{y})$ and then recursively samples $\widetilde{\boldsymbol{x}}_{i-1}$ using $\widetilde{\boldsymbol{x}}_i$ for $i = N, N-1, \ldots, 1$. We argue, however, that using only $\widetilde{\boldsymbol{x}}_i$ to construct $\widetilde{\boldsymbol{x}}_{i-1}$ is suboptimal, and is in fact guided by the intuition from traditional diffusion, like (4.4) or (4.6), where

$$p(\boldsymbol{x}_0|\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) = p(\boldsymbol{x}_0|\boldsymbol{x}_i, \boldsymbol{y}). \tag{4.23}$$

But under the decoupled forward process (4.21), one can show that

$$p(\boldsymbol{x}_0|\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) \neq p(\boldsymbol{x}_0|\boldsymbol{x}_i, \boldsymbol{y}), \tag{4.24}$$

and so we claim that *all* previous samples $\{\widetilde{\boldsymbol{x}}_k\}_{k=i}^N$ should be used to construct $\widetilde{\boldsymbol{x}}_{i-1}$. The details are presented below.

## 4.2 Proposed Approaches

### 4.2.1 Exact RAPS

Suppose that the goal is to sample from the posterior $p(\boldsymbol{x}_0|\boldsymbol{y})$. Notice that, under the forward model (4.21),

$$p(\boldsymbol{x}_0|\boldsymbol{y}) = \int \cdots \int p(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\boldsymbol{y}) \, \mathrm{d}\boldsymbol{x}_1 \cdots \mathrm{d}\boldsymbol{x}_N \tag{4.25}$$

$$= \int \cdots \int p(\boldsymbol{x}_0|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\boldsymbol{y}) \, \mathrm{d}\boldsymbol{x}_1 \cdots \mathrm{d}\boldsymbol{x}_N, \tag{4.26}$$

where $p(\boldsymbol{x}_0|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{y})$ does not simplify. So, to sample from $p(\boldsymbol{x}_0|\boldsymbol{y})$, we could first draw $(\widetilde{\boldsymbol{x}}_1, \ldots, \widetilde{\boldsymbol{x}}_N) \sim p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\boldsymbol{y})$ and then sample from $p(\boldsymbol{x}_0|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{y})$ evaluated at $\boldsymbol{x}_1 = \widetilde{\boldsymbol{x}}_1, \ldots, \boldsymbol{x}_N = \widetilde{\boldsymbol{x}}_N$.

As for how to draw $(\widetilde{\boldsymbol{x}}_1, \ldots, \widetilde{\boldsymbol{x}}_N) \sim p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\boldsymbol{y})$, notice that

$$
\begin{aligned}
p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\boldsymbol{y}) &= p(\boldsymbol{x}_1|\boldsymbol{x}_2, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) p(\boldsymbol{x}_2, \ldots, \boldsymbol{x}_N|\boldsymbol{y}) && (4.27)\\
&= p(\boldsymbol{x}_1|\boldsymbol{x}_2, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) p(\boldsymbol{x}_2|\boldsymbol{x}_3, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) p(\boldsymbol{x}_3, \ldots, \boldsymbol{x}_N|\boldsymbol{y}) && (4.28)\\
&= p(\boldsymbol{x}_1|\boldsymbol{x}_2, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) p(\boldsymbol{x}_2|\boldsymbol{x}_3, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) \cdots p(\boldsymbol{x}_N|\boldsymbol{y}) && (4.29)\\
&= \prod_{i=1}^{N} p(\boldsymbol{x}_i|\boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y}). && (4.30)
\end{aligned}
$$

This implies that we can sample the tuple $(\widetilde{\boldsymbol{x}}_1, \ldots, \widetilde{\boldsymbol{x}}_N)$ sequentially: we first sample $\widetilde{\boldsymbol{x}}_N \sim p(\boldsymbol{x}_N|\boldsymbol{y})$, and then $\widetilde{\boldsymbol{x}}_{N-1} \sim p(\boldsymbol{x}_{N-1}|\boldsymbol{x}_N, \boldsymbol{y})|_{\boldsymbol{x}_N = \widetilde{\boldsymbol{x}}_N}$, and then $\widetilde{\boldsymbol{x}}_{N-2} \sim p(\boldsymbol{x}_{N-2}|\boldsymbol{x}_{N-1}, \boldsymbol{x}_N, \boldsymbol{y})|_{\boldsymbol{x}_{N-1} = \widetilde{\boldsymbol{x}}_{N-1}, \boldsymbol{x}_N = \widetilde{\boldsymbol{x}}_N}$, and so on. This suggests a reverse process over the indices $i = N, N-1, \ldots, 1$. To initialize the reverse process, we can approximate $p(\boldsymbol{x}_N|\boldsymbol{y}) \approx \mathcal{N}(\boldsymbol{x}_N; \boldsymbol{0}, \sigma_N^2 \boldsymbol{I})$ when $\sigma_N^2$ is very large.

At index $i$ of this reverse process, we want to draw $\widetilde{\boldsymbol{x}}_i \sim p(\boldsymbol{x}_i|\boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y})$ evaluated at $\boldsymbol{x}_{i+1} = \widetilde{\boldsymbol{x}}_{i+1}, \ldots, \boldsymbol{x}_N = \widetilde{\boldsymbol{x}}_N$. For intuition on how to perform this sampling, notice that, for any $i \in \{1, \ldots, N\}$,

$$
\begin{aligned}
p(\boldsymbol{x}_i|\boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) &= \int p(\boldsymbol{x}_i, \boldsymbol{x}_0|\boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x}_0 && (4.31)\\
&= \int p(\boldsymbol{x}_i|\boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y}, \boldsymbol{x}_0) p(\boldsymbol{x}_0|\boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x}_0 && (4.32)\\
&= \int p(\boldsymbol{x}_i|\boldsymbol{x}_0) p(\boldsymbol{x}_0|\boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x}_0 && (4.33)
\end{aligned}
$$

because $\boldsymbol{x}_i$ is independent of $\{\boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y}\}$ when conditioned on $\boldsymbol{x}_0$. Furthermore, because $p(\boldsymbol{x}_i|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{x}_0, \sigma_i^2 \boldsymbol{I})$, we can draw $\widetilde{\boldsymbol{x}}_i$ via

$$
\widetilde{\boldsymbol{x}}_i = \widetilde{\boldsymbol{x}}_{0|i+1} + \sigma_i \boldsymbol{n}_i, \quad \boldsymbol{n}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \quad \widetilde{\boldsymbol{x}}_{0|i+1} \sim p(\boldsymbol{x}_0|\boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N, \boldsymbol{y})|_{\boldsymbol{x}_{i+1} = \widetilde{\boldsymbol{x}}_{i+1}, \ldots, \boldsymbol{x}_N = \widetilde{\boldsymbol{x}}_N}. \tag{4.34}
$$

As for how to construct $\widetilde{\boldsymbol{x}}_{0|i+1}$, notice that

$$p(\boldsymbol{x}_0|\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N,\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N,\boldsymbol{x}_0)p(\boldsymbol{x}_0|\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N)}{p(\boldsymbol{y}|\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N)} \tag{4.35}$$

$$\propto p(\boldsymbol{y}|\boldsymbol{x}_0)p(\boldsymbol{x}_0|\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N) \tag{4.36}$$

because $\boldsymbol{y}$ is independent of $\{\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N\}$ when conditioned on $\boldsymbol{x}_0$. Importantly, because $\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N$ are independent when conditioned on $\boldsymbol{x}_0$, we can write

$$p(\boldsymbol{x}_0|\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N) \propto p(\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N|\boldsymbol{x}_0)p(\boldsymbol{x}_0) \tag{4.37}$$

$$p(\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N|\boldsymbol{x}_0) = \prod_{j=1}^{N-i} p(\boldsymbol{x}_{i+j}|\boldsymbol{x}_0) = \prod_{j=1}^{N-i} \mathcal{N}(\boldsymbol{x}_{i+j};\boldsymbol{x}_0,\sigma_{i+j}^2\boldsymbol{I}) = \prod_{j=1}^{N-i} \mathcal{N}(\boldsymbol{x}_0;\boldsymbol{x}_{i+j},\sigma_{i+j}^2\boldsymbol{I}) \tag{4.38}$$

$$\propto \mathcal{N}(\boldsymbol{x}_0;\overline{\boldsymbol{x}}_{i+1},\overline{\sigma}_{i+1}^2\boldsymbol{I}) \quad \text{for} \quad \overline{\sigma}_{i+1}^2 \triangleq \frac{1}{\sum_{j=1}^{N-i}\sigma_{i+j}^{-2}} \quad \text{and} \quad \overline{\boldsymbol{x}}_{i+1} \triangleq \overline{\sigma}_{i+1}^2 \sum_{j=1}^{N-i} \frac{\boldsymbol{x}_{i+j}}{\sigma_{i+j}^2} \tag{4.39}$$

$$= \mathcal{N}(\overline{\boldsymbol{x}}_{i+1};\boldsymbol{x}_0,\overline{\sigma}_{i+1}^2\boldsymbol{I}) \tag{4.40}$$

and thus, from (4.37) and (4.40)

$$p(\boldsymbol{x}_0|\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N) \propto \mathcal{N}(\overline{\boldsymbol{x}}_{i+1};\boldsymbol{x}_0,\overline{\sigma}_{i+1}^2\boldsymbol{I})p(\boldsymbol{x}_0) \propto p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_{i+1}) \tag{4.41}$$

for

$$p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_{i+1}) \triangleq Z^{-1}\mathcal{N}(\overline{\boldsymbol{x}}_{i+1};\boldsymbol{x}_0,\overline{\sigma}_{i+1}^2\boldsymbol{I})p(\boldsymbol{x}_0) \quad \text{with} \quad Z \triangleq \int \mathcal{N}(\overline{\boldsymbol{x}}_{i+1};\boldsymbol{x}_0,\overline{\sigma}_{i+1}^2\boldsymbol{I})p(\boldsymbol{x}_0)\,\mathrm{d}\boldsymbol{x}_0. \tag{4.42}$$

The implication of (4.41) is that, for the purpose of estimating $\boldsymbol{x}_0$, knowing $\overline{\boldsymbol{x}}_{i+1}$ is equivalent to knowing $\boldsymbol{x}_{i+1},\ldots,\boldsymbol{x}_N$. Furthermore, (4.42) shows that $\overline{\boldsymbol{x}}_{i+1}$ can be interpreted as a Gaussian-noise corrupted measurement of $\boldsymbol{x}_0$. Finally, based on (4.39), the quantities $\overline{\sigma}_{i+1}^{-2}$ and $\overline{\boldsymbol{x}}_{i+1}$ can be recursively computed:

$$\overline{\sigma}_{i+1}^{-2} = \sum_{j=1}^{N-i} \sigma_{i+j}^{-2} = \sigma_{i+1}^{-2} + \sum_{j=2}^{N-i} \sigma_{i+j}^{-2} = \sigma_{i+1}^{-2} + \sum_{j'=1}^{N-i-1} \sigma_{i+1+j'}^{-2} = \sigma_{i+1}^{-2} + \overline{\sigma}_{i+2}^{-2} \tag{4.43}$$

$$\frac{\overline{\boldsymbol{x}}_{i+1}}{\overline{\sigma}_{i+1}^2} = \sum_{j=1}^{N-i} \frac{\boldsymbol{x}_{i+j}}{\sigma_{i+j}^2} = \frac{\boldsymbol{x}_{i+1}}{\sigma_{i+1}^2} + \sum_{j=2}^{N-i} \frac{\boldsymbol{x}_{i+j}}{\sigma_{i+j}^2} = \frac{\boldsymbol{x}_{i+1}}{\sigma_{i+1}^2} + \sum_{j'=1}^{N-i-1} \frac{\boldsymbol{x}_{i+1+j'}}{\sigma_{i+1+j'}^2} = \frac{\boldsymbol{x}_{i+1}}{\sigma_{i+1}^2} + \frac{\overline{\boldsymbol{x}}_{i+2}}{\overline{\sigma}_{i+2}^2}. \tag{4.44}$$

Plugging (4.41) into (4.36) then gives

$$p(\boldsymbol{x}_0|\boldsymbol{x}_{i+1},\dots,\boldsymbol{x}_N,\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{x}_0)p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_{i+1}). \tag{4.45}$$

To sample from (4.45), one option is to use MCMC methods such as Langevin dynamics [170]:

$$\widetilde{\boldsymbol{x}}[k+1] = \widetilde{\boldsymbol{x}}[k] - \eta_{\mathsf{lan}}\big[\nabla_{\boldsymbol{x}_0}\ln p(\boldsymbol{y}|\boldsymbol{x}_0) + \nabla_{\boldsymbol{x}_0}\ln p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_{i+1})\big]_{\boldsymbol{x}_0=\widetilde{\boldsymbol{x}}[k]} + \sqrt{2\eta_{\mathsf{lan}}}\boldsymbol{\epsilon}[k] \tag{4.46}$$

for $\boldsymbol{\epsilon}[k] \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I})$. As $\eta_{\mathsf{lan}} \to 0$ and $k \to \infty$, the quantity $\widetilde{\boldsymbol{x}}[k]$ is an exact sample from (4.45) under certain regularity conditions. In practice, $\nabla_{\boldsymbol{x}_0}\ln p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_{i+1})$ would be approximated by a noise-conditional score network (NCSN) [83] trained by denoising score matching [169], as described around (4.3).

Algorithm 6 summarizes the reverse process described thus far, which we refer to as "exact" recursive annealed posterior sampling (RAPS). It generates exact samples from $p(\boldsymbol{x}_0|\boldsymbol{y})$ when Langevin dynamics is correct and when $\sigma_N^2 \to \infty$, in which case $p(\boldsymbol{x}_N|\boldsymbol{y})$ is well approximated by $\mathcal{N}(\boldsymbol{x}_N;\boldsymbol{0},\sigma_N^2)$.

### 4.2.2 Equivalence between RAPS and SMLD

An interesting observation is that if we consider $\overline{\boldsymbol{x}}_i$ as the "main" iterate of the RAPS reverse process, then we can write its update as

$$\overline{\boldsymbol{x}}_i = \frac{\overline{\sigma}_i^2}{\sigma_i^2}\widetilde{\boldsymbol{x}}_i + \frac{\overline{\sigma}_i^2}{\overline{\sigma}_{i+1}^2}\overline{\boldsymbol{x}}_{i+1} = \frac{\sigma_i^{-2}}{\overline{\sigma}_i^{-2}}(\widetilde{\boldsymbol{x}}_{0|i+1} + \sigma_i\boldsymbol{n}_i) + \frac{\overline{\sigma}_{i+1}^{-2}}{\overline{\sigma}_i^{-2}}\overline{\boldsymbol{x}}_{i+1} \tag{4.47}$$

$$= \frac{\overline{\sigma}_i^{-2} - \overline{\sigma}_{i+1}^{-2}}{\overline{\sigma}_i^{-2}}\widetilde{\boldsymbol{x}}_{0|i+1} + \frac{\overline{\sigma}_{i+1}^{-2}}{\overline{\sigma}_i^{-2}}\overline{\boldsymbol{x}}_{i+1} + \frac{\overline{\sigma}_i^2}{\sigma_i}\boldsymbol{n}_i = \Big(1 - \frac{\overline{\sigma}_{i+1}^{-2}}{\overline{\sigma}_i^{-2}}\Big)\widetilde{\boldsymbol{x}}_{0|i+1} + \frac{\overline{\sigma}_{i+1}^{-2}}{\overline{\sigma}_i^{-2}}\overline{\boldsymbol{x}}_{i+1} + \frac{\overline{\sigma}_i^2}{\sigma_i}\boldsymbol{n}_i \tag{4.48}$$

$$= \frac{\overline{\sigma}_i^2}{\overline{\sigma}_{i+1}^2}\overline{\boldsymbol{x}}_{i+1} + \Big(1 - \frac{\overline{\sigma}_i^2}{\overline{\sigma}_{i+1}^2}\Big)\widetilde{\boldsymbol{x}}_{0|i+1} + \frac{\overline{\sigma}_i^2}{\sigma_i}\boldsymbol{n}_i. \tag{4.49}$$

We might wonder whether (4.49) is a valid recursion. In other words, if $\overline{\boldsymbol{x}}_{i+1} = \boldsymbol{x}_0 + \overline{\sigma}_{i+1}\boldsymbol{e}_{i+1}$ for $\boldsymbol{e}_{i+1} \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I})$ and $\widetilde{\boldsymbol{x}}_{0|i+1} = \boldsymbol{x}_0$ (i.e., the conditional denoising is

**Algorithm 6** Exact RAPS

**Require:** $p(\boldsymbol{y}|\boldsymbol{x}_0), \{\sigma_i\}_{i=1}^N, \sigma_0 = 0, N_{\mathsf{lan}}, \eta_{\mathsf{lan}}$

$\widetilde{\boldsymbol{x}}_N \sim \mathcal{N}(\boldsymbol{0}, \sigma_N^2 \boldsymbol{I})$

$\overline{\sigma}_{N+1} = \infty, \quad \overline{\boldsymbol{x}}_{N+1} = \boldsymbol{0}$

**for** $i = N, N-1, \ldots, 1$ **do**

  // recursive update

  $\overline{\sigma}_i^2 = 1/(\sigma_i^{-2} + \overline{\sigma}_{i+1}^{-2})$

  $\overline{\boldsymbol{x}}_i = \overline{\sigma}_i^2(\widetilde{\boldsymbol{x}}_i/\sigma_i^2 + \overline{\boldsymbol{x}}_{i+1}/\overline{\sigma}_{i+1}^2)$

  // Langevin sampling from $p(\boldsymbol{y}|\boldsymbol{x}_0)p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_i)$

  $\widetilde{\boldsymbol{x}}_{0|i} \leftarrow \overline{\boldsymbol{x}}_i$

  **for** $k = 0, \ldots, N_{\mathsf{lan}} - 1$ **do**

    $\boldsymbol{\epsilon}_{ik} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

    $\widetilde{\boldsymbol{x}}_{0|i} \leftarrow \widetilde{\boldsymbol{x}}_{0|i} - \eta_{\mathsf{lan}}\big[\nabla_{\boldsymbol{x}_0} \ln p(\boldsymbol{y}|\boldsymbol{x}_0) + \nabla_{\boldsymbol{x}_0} \ln p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_i)\big]_{\boldsymbol{x}_0 = \widetilde{\boldsymbol{x}}_{0|i}} + \sqrt{2\eta_{\mathsf{lan}}}\boldsymbol{\epsilon}_{ik}$

  // annealing

  $\widetilde{\boldsymbol{x}}_{i-1} = \widetilde{\boldsymbol{x}}_{0|i} + \sigma_{i-1}\boldsymbol{n}_{i-1}, \quad \boldsymbol{n}_{i-1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

**return** $\widetilde{\boldsymbol{x}}_0$

perfect), then is $\overline{\boldsymbol{x}}_i = \boldsymbol{x}_0 + \overline{\sigma}_i \boldsymbol{e}_i$ for $\boldsymbol{e}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$? We can answer "yes" since, under these conditions,

$$\overline{\boldsymbol{x}}_i = \frac{\overline{\sigma}_i^2}{\overline{\sigma}_{i+1}^2}(\boldsymbol{x}_0 + \overline{\sigma}_{i+1}\boldsymbol{e}_{i+1}) + \Big(1 - \frac{\overline{\sigma}_i^2}{\overline{\sigma}_{i+1}^2}\Big)\boldsymbol{x}_0 + \frac{\overline{\sigma}_i^2}{\sigma_i}\boldsymbol{n}_i \tag{4.50}$$

$$= \boldsymbol{x}_0 + \frac{\overline{\sigma}_i^2}{\overline{\sigma}_{i+1}}\boldsymbol{e}_{i+1} + \frac{\overline{\sigma}_i^2}{\sigma_i}\boldsymbol{n}_i. \tag{4.51}$$

Since $\boldsymbol{n}_i$ is independent of $\boldsymbol{e}_{i+1}$, the total variance of the noise above will equal

$$\frac{\overline{\sigma}_i^4}{\overline{\sigma}_{i+1}^2} + \frac{\overline{\sigma}_i^4}{\sigma_i^2} = \overline{\sigma}_i^4(\overline{\sigma}_{i+1}^{-2} + \sigma_i^{-2}) = \overline{\sigma}_i^4(\overline{\sigma}_i^{-2}) = \overline{\sigma}_i^2, \tag{4.52}$$

which is the expected result.

We now argue that (4.49) is exactly the SMLD reverse process (4.12) when "$\overline{\sigma}_i$" is used in place of the standard SMLD quantity "$\sigma_i$" and when $\widetilde{\boldsymbol{x}}_{0|i+1}$ is considered the denoiser output $\boldsymbol{d}_{\boldsymbol{\theta}}$. The first two terms in (4.49) and (4.12) clearly agree, an so we

need to show that the last term agrees as well. Towards this aim, notice that

$$\frac{\overline{\sigma}_i^2}{\sigma_i} = \sqrt{\frac{\overline{\sigma}_i^4}{\sigma_i^2}} = \sqrt{\overline{\sigma}_i^4 \sigma_i^{-2}} = \sqrt{\overline{\sigma}_i^4 (\overline{\sigma}_i^{-2} - \overline{\sigma}_{i+1}^{-2})} = \sqrt{\overline{\sigma}_i^2 - \frac{\overline{\sigma}_i^4}{\overline{\sigma}_{i+1}^2}} = \sqrt{\frac{\overline{\sigma}_i^2 (\overline{\sigma}_{i+1}^2 - \overline{\sigma}_i^2)}{\overline{\sigma}_{i+1}^2}} \tag{4.53}$$

which exactly agrees with the last term in (4.12). Thus RAPS is equivalent to SMLD.

### 4.2.3 Choice of variance schedule $\{\sigma_i\}$

In RAPS, the choice of $\{\sigma_i\}$ determines $\{\overline{\sigma}_i\}$, which in turn governs the dynamics of the reverse process via (4.49). For example, by changing the ratio $\overline{\sigma}_i^2/\sigma_i$, we can control how much stochasticity there is in the reverse process. Given the connection to the SMLD, it makes sense to choose a schedule $\{\overline{\sigma}_i\}$ that transitions from $\sigma_{\max}$ down to $\sigma_{\min}$ in either a geometric manner

$$\overline{\sigma}_i = \sigma_{\min}(\sigma_{\max}/\sigma_{\min})^{\frac{i-1}{N-1}} \tag{4.54}$$

or like deterministic EDM [87]

$$\overline{\sigma}_i = \left(\sigma_{\max}^{1/\rho} + \tfrac{N-i}{N-1}(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho})\right)^{\rho}. \tag{4.55}$$

In either case, we would set

$$\sigma_i^2 = \begin{cases} \overline{\sigma}_i^2 & i = N \\ \left[\overline{\sigma}_i^{-2} - \overline{\sigma}_{i+1}^{-2}\right]^{-1} & i \in \{1, \dots, N-1\}. \end{cases} \tag{4.56}$$

### 4.2.4 Practical RAPS

The exact RAPS approach in Alg. 6 is computationally impractical because it computes the score function within the inner loop. To circumvent this computation, we propose a Gaussian approximation

$$p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_{i+1}) \approx \mathcal{N}(\boldsymbol{x}_0; \widehat{\boldsymbol{x}}_{0|i+1}, \widehat{\sigma}_{0|i+1}^2 \boldsymbol{I}), \tag{4.57}$$

where $\widehat{\boldsymbol{x}}_{0|i+1}$ either approximates $\mathrm{E}\{\boldsymbol{x}_0|\overline{\boldsymbol{x}}_{i+1}\}$ or is an approximate sample from $p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_{i+1})$ computed using a few steps of DDIM. Although similar Gaussian approximations have been used in the past (see [88]), ours differs in the construction of $\widehat{\sigma}^2_{0|i+1}$. We view the goal of $\widehat{\sigma}^2_{0|i+1}$ as approximating $\mathrm{E}\{\|\boldsymbol{x}_0 - \widehat{\boldsymbol{x}}_{0|i+1}\|^2/d\}$. To do this, we propose to train the parameters $\phi$ of a model $f_\phi(\cdot)$ such that

$$f_\phi(\sigma) \approx \mathrm{E}_{\boldsymbol{x}_0,\boldsymbol{y},\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\{\ln p(\boldsymbol{y}|\boldsymbol{x}_0 + \sigma\boldsymbol{\epsilon})\} \tag{4.58}$$

and then, in the reverse process, set

$$\widehat{\sigma}_{0|i+1} = \arg\min_{\sigma} \left|[\ln p(\boldsymbol{y}|\boldsymbol{x}_0)]_{\boldsymbol{x}_0=\widehat{\boldsymbol{x}}_{0|i+1}} - f_\phi(\sigma)\right|. \tag{4.59}$$

In the special case of a linear-Gaussian measurement model of the form

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \sigma_{\mathsf{w}}\boldsymbol{w} \in \mathbb{R}^m, \quad \boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{4.60}$$

we have $\ln p(\boldsymbol{y}|\boldsymbol{x}_0) = -\frac{1}{2\sigma^2_{\mathsf{w}}}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_0\|^2 - C$ for $C \triangleq \frac{m}{2}\ln[2\pi\sigma^2_{\mathsf{w}}]$, and so we can actually design $f_\phi(\cdot)$ in closed-form as

$$f_\phi(\sigma) = \mathrm{E}_{\boldsymbol{x}_0,\boldsymbol{y},\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\{\ln p(\boldsymbol{y}|\boldsymbol{x}_0 + \sigma\boldsymbol{\epsilon})\} = -\frac{1}{2\sigma^2_{\mathsf{w}}}\,\mathrm{E}_{\boldsymbol{w},\boldsymbol{\epsilon}}\left\{\|\sigma_{\mathsf{w}}\boldsymbol{w} + \sigma\boldsymbol{A}\boldsymbol{\epsilon}\|^2\right\} - C \tag{4.61}$$

$$= -\frac{1}{2\sigma^2_{\mathsf{w}}}\left(m\sigma^2_{\mathsf{w}} + \sigma^2\|\boldsymbol{A}\|_F^2\right) - C, \tag{4.62}$$

after which (4.59) reduces to

$$\widehat{\sigma}_{0|i+1} = \arg\min_{\sigma} \left|m\sigma^2_{\mathsf{w}} + \sigma^2\|\boldsymbol{A}\|_F^2 - \|\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{0|i+1}\|^2\right| \tag{4.63}$$

$$= (\|\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{0|i+1}\|^2 - m\sigma^2_{\mathsf{w}})/\|\boldsymbol{A}\|_F^2, \tag{4.64}$$

which matches the approach proposed in DDfire [171]. The approach that we propose for RAPS is more general, however, in that it can handle arbitrary $p(\boldsymbol{y}|\boldsymbol{x}_0)$.

We summarize this "Gaussian" variant of RAPS in Alg. 7. Note that if $N_{\mathsf{ode}} = 1$, then $\widehat{\boldsymbol{x}}_{0|i} = \boldsymbol{d}_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}_i; \overline{\sigma}_i)$.

**Algorithm 7** Gaussian RAPS

**Require:** $\{\sigma_i\}_{i=1}^N, \sigma_0 = 0, \boldsymbol{d}_{\boldsymbol{\theta}}(\cdot|\sigma), N_{\mathsf{ode}}, \sigma_{\min}, f_\phi(\cdot), p(\boldsymbol{y}|\boldsymbol{x}_0), N_{\mathsf{lan}}, \eta_{\mathsf{lan}}$

$\widetilde{\boldsymbol{x}}_N \sim \mathcal{N}(\boldsymbol{0}, \sigma_N^2 \boldsymbol{I})$

$\overline{\sigma}_{N+1} = \infty, \quad \overline{\boldsymbol{x}}_{N+1} = \boldsymbol{0}$

**for** $i = N, N-1, \ldots, 1$ **do**

  // recursive update

  $\overline{\sigma}_i^2 = 1/(\sigma_i^{-2} + \overline{\sigma}_{i+1}^{-2})$

  $\overline{\boldsymbol{x}}_i = \overline{\sigma}_i^2(\widetilde{\boldsymbol{x}}_i/\sigma_i^2 + \overline{\boldsymbol{x}}_{i+1}/\overline{\sigma}_{i+1}^2)$

  // unconditional DDIM starting from variance $\overline{\sigma}_i^2$

  $\widehat{\boldsymbol{x}}_{0|i} \leftarrow \overline{\boldsymbol{x}}_i$

  **for** $j = N_{\mathsf{ode}}, N_{\mathsf{ode}} - 1, \ldots, 1$ **do**

    $\sigma_{ij}^2 = \overline{\sigma}_i^2(\sigma_{\min}^2/\overline{\sigma}_i^2)^{\frac{N_{\mathsf{ode}}-j}{N_{\mathsf{ode}}-1}}$ ▷ *geometric with* $\sigma_{i,N_{\mathsf{ode}}} = \overline{\sigma}_i$ *and* $\sigma_{i,1}|_{N_{\mathsf{ode}}>1} = \sigma_{\min}$

    $\widehat{\boldsymbol{x}}_{0|i} \leftarrow \frac{\sigma_{i,j-1}}{\sigma_{ij}}\widehat{\boldsymbol{x}}_{0|i} + (1 - \frac{\sigma_{i,j-1}}{\sigma_{ij}})\boldsymbol{d}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{x}}_{0|i}; \sigma_{ij})$ ▷ *where* $\sigma_{i,0} \triangleq 0$

  $\widehat{\sigma}_{0|i}^2 = \arg\min_\sigma \left|[\ln p(\boldsymbol{y}|\boldsymbol{x}_0)]_{\boldsymbol{x}_0 = \widehat{\boldsymbol{x}}_{0|i}} - f_\phi(\sigma)\right|$

  // Langevin sampling from $p(\boldsymbol{y}|\boldsymbol{x}_0)p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_i)$

  **for** $j = 0, \ldots, N_{\mathsf{lan}} - 1$ **do**

    $\boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

    $\widetilde{\boldsymbol{x}}_{0|i} \leftarrow \widetilde{\boldsymbol{x}}_{0|i} - \eta_{\mathsf{lan}}\left[\nabla_{\boldsymbol{x}_0} \ln p(\boldsymbol{y}|\boldsymbol{x}_0) + \nabla_{\boldsymbol{x}_0} \frac{1}{2\widehat{\sigma}_{0|i}^2}\|\boldsymbol{x}_0 - \widehat{\boldsymbol{x}}_{0|i}\|^2\right]_{\boldsymbol{x}_0 = \widetilde{\boldsymbol{x}}_{0|i}} + \sqrt{2\eta_{\mathsf{lan}}}\boldsymbol{\epsilon}_{ij}$

  // annealing

  $\widetilde{\boldsymbol{x}}_{i-1} = \widetilde{\boldsymbol{x}}_{0|i} + \sigma_{i-1}\boldsymbol{n}_{i-1}, \quad \boldsymbol{n}_{i-1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

**return** $\widetilde{\boldsymbol{x}}_0$

- For a given NFE budget, how beneficial is it to use multiple inner EDM iterations at the expense of fewer outer diffusion iterations? Since the inner iterations don't use $\boldsymbol{y}$, it seems that data consistency would be lost. Are they trying to keep the MCMC cost down? It seems they use 100 MCMC iterations by default.

- Expanding on the last point, is it essential to add the colored noise in (4.69)? The colored noise contributes appropriate levels of stochasticity to the measured and non-measured spaces, but since the noise injected by the subsequent diffusion step is so large, it is not clear that the colored noise would have a significant effect

on $\boldsymbol{x}_{i-1}$. (Also, it may degrade the performance of the subsequent denoising step.)

### 4.2.5 Simplifications and Stochastic Enhancement

We now consider a simplification of Gaussian RAPS for the case of the standard linear models (SLM), where the likelihood is given by $p(\boldsymbol{y}|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{A}\boldsymbol{x}_0, \sigma_{\mathsf{w}}^2\boldsymbol{I})$. Under this model, the approximate posterior $\propto p(\boldsymbol{y}|\boldsymbol{x}_0)\mathcal{N}(\boldsymbol{x}_0; \widehat{\boldsymbol{x}}_{0|i}, \widehat{\sigma}_{0|i}^2\boldsymbol{I})$ takes the Gaussian form $\mathcal{N}(\boldsymbol{x}_0; \boldsymbol{\mu}_i, \boldsymbol{C}_i)$, where

$$\boldsymbol{\mu}_i \triangleq \arg\min_{\boldsymbol{x}} \Big\{ \frac{1}{2\sigma_{\mathsf{w}}^2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \frac{1}{2\widehat{\sigma}_{0|i}^2}\|\boldsymbol{x} - \widehat{\boldsymbol{x}}_{0|i}\|^2 \Big\} \tag{4.65}$$

$$= \widehat{\boldsymbol{x}}_{0|i} + \boldsymbol{A}^\top \Big( \boldsymbol{A}\boldsymbol{A}^\top + \frac{\sigma_{\mathsf{w}}^2}{\widehat{\sigma}_{0|i}^2}\boldsymbol{I} \Big)^{-1} (\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{0|i}) \tag{4.66}$$

and

$$\boldsymbol{C}_i = \widehat{\sigma}_{0|i}^2\boldsymbol{I} - \widehat{\sigma}_{0|i}^2\boldsymbol{A}^\top \Big( \widehat{\sigma}_{0|i}^2\boldsymbol{A}\boldsymbol{A}^\top + \sigma_{\mathsf{w}}^2\boldsymbol{I} \Big)^{-1} \boldsymbol{A}\widehat{\sigma}_{0|i}^2 = \widehat{\sigma}_{0|i}^2 \Big( \boldsymbol{I} - \boldsymbol{A}^\top \Big( \boldsymbol{A}\boldsymbol{A}^\top + \frac{\sigma_{\mathsf{w}}^2}{\widehat{\sigma}_{0|i}^2}\boldsymbol{I} \Big)^{-1} \boldsymbol{A} \Big) \tag{4.67}$$

$$= \widehat{\sigma}_{0|i}^2 \Big( \boldsymbol{I} + \frac{\widehat{\sigma}_{0|i}^2}{\sigma_{\mathsf{w}}^2}\boldsymbol{A}^\top\boldsymbol{A} \Big)^{-1} = \Big( \frac{1}{\widehat{\sigma}_{0|i}^2}\boldsymbol{I} + \frac{1}{\sigma_{\mathsf{w}}^2}\boldsymbol{A}^\top\boldsymbol{A} \Big)^{-1}. \tag{4.68}$$

Thus, in place of MCMC, a posterior sample could be drawn using

$$\widetilde{\boldsymbol{x}}_{0|i} = \boldsymbol{\mu}_i + \boldsymbol{C}_i^{1/2}\boldsymbol{v}_i, \quad \boldsymbol{v}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{4.69}$$

For nonlinear inverse problems modeled as GLMs, a similar approach can be adopted using expectation propagation, as we will discuss later.

The colored noise in (4.69) contributes appropriate levels of stochasticity to the measured and non-measured spaces. However, since the noise injected by the subsequent annealing step is substantially larger, the effect of the colored noise is overwhelmed.

Empirically, it was observed to have no significant impact on $\widetilde{\boldsymbol{x}}_{i-1}$, and hence we choose to set

$$\widetilde{\boldsymbol{x}}_{0|i} = \boldsymbol{\mu}_i = \widehat{\boldsymbol{x}}_{0|i} + \boldsymbol{A}^\top \Big( \boldsymbol{A}\boldsymbol{A}^\top + \frac{\sigma_{\mathsf{w}}^2}{\widehat{\sigma}_{0|i}^2} \boldsymbol{I} \Big)^{-1} (\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{0|i}) \tag{4.70}$$

$$= \Big( \widehat{\sigma}_{0|i}^2 \boldsymbol{A}^\top \boldsymbol{A} + \sigma_{\mathsf{w}}^2 \boldsymbol{I} \Big)^{-1} \Big( \widehat{\sigma}_{0|i}^2 \boldsymbol{A}^\top \boldsymbol{y} + \sigma_{\mathsf{w}}^2 \widehat{\boldsymbol{x}}_{0|i} \Big). \tag{4.71}$$

Equation (4.71) can be computed using conjugate gradients (CG) or, if practical, the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$ via

$$\widetilde{\boldsymbol{x}}_{0|i} = \boldsymbol{V} \Big( \widehat{\sigma}_{0|i}^2 \boldsymbol{S}^\top \boldsymbol{S} + \sigma_{\mathsf{w}}^2 \boldsymbol{I} \Big)^{-1} \Big( \widehat{\sigma}_{0|i}^2 \boldsymbol{S}^\top \boldsymbol{U}^\top \boldsymbol{y} + \sigma_{\mathsf{w}}^2 \boldsymbol{V}^\top \widehat{\boldsymbol{x}}_{0|i} \Big). \tag{4.72}$$

In addition, given a fixed NFE budget, using multiple inner unconditional DDIM iterations comes at the cost of fewer outer diffusion iterations. Since the inner iterations do not incorporate the measurements $\boldsymbol{y}$, relying heavily on them can lead to a loss of data consistency. To mitigate this, we choose to use $N_{\mathsf{ode}} = 1$, resulting in the update $\widehat{\boldsymbol{x}}_{0|i} = \boldsymbol{d}_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}_i; \overline{\sigma}_i)$.

The practical refinements introduced up to this point perform well across a range of linear inverse problems. However, we observe a notable degradation in performance for inpainting task. To address this, we draw inspiration from the stochastic EDM algorithm (4.18), parameterized by $\gamma_i \geq 0$, which introduces noise into the ODE (4.17), yielding a generalized algorithm with adjustable stochasticity. By selecting appropriate values of $\gamma_i$, stochastic EDM can span a spectrum of stochastic behaviors, ranging from purely deterministic (i.e., ODE) to highly stochastic (i.e., exceeding the stochasticity of SMLD). As shown in prior work [87], injecting controlled stochasticity can improve robustness by mitigating errors arising from discretization and approximations introduced in earlier sampling steps. Motivated by these insights, we adapt a

similar strategy within RAPS by augmenting the update steps with additional noise, effectively enhancing the overall stochasticity of the sampling process.

By incorporating all of the aforementioned refinements—including the simplification for SLM (cf. Eq. (4.71)), the use of a single inner DDIM iteration i.e. $N_{\mathsf{ode}} = 1$, and the introduction of controlled stochasticity inspired by stochastic EDM—we arrive at the following updated set of update rules that modify Alg. 7 for any $\gamma_i \geq 0$:

$$\frac{1}{\overline{\sigma}_i^2} = \frac{1}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2} + \frac{1}{\sigma_i^2} \tag{4.73}$$

$$\overline{\boldsymbol{x}}_i = \frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2}\breve{\boldsymbol{x}}_{i+1} + \left(1 - \frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2}\right)\big(\widetilde{\boldsymbol{x}}_{0|i+1} + \sigma_i \boldsymbol{n}_i\big), \quad \boldsymbol{n}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{4.74}$$

$$\breve{\boldsymbol{x}}_i = \overline{\boldsymbol{x}}_i + \sqrt{(1+\gamma_i)^2\overline{\sigma}_i^2 - \overline{\sigma}_i^2}\boldsymbol{w}_i, \quad \boldsymbol{w}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{4.75}$$

$$\widehat{\boldsymbol{x}}_{0|i} = \boldsymbol{d}_{\boldsymbol{\theta}}(\breve{\boldsymbol{x}}_i, (1+\gamma_i)\overline{\sigma}_i) \tag{4.76}$$

$$\widetilde{\boldsymbol{x}}_{0|i} = \left(\widehat{\sigma}_{0|i}^2 \boldsymbol{A}^\top \boldsymbol{A} + \sigma_{\mathsf{w}}^2 \boldsymbol{I}\right)^{-1}\left(\widehat{\sigma}_{0|i}^2 \boldsymbol{A}^\top \boldsymbol{y} + \sigma_{\mathsf{w}}^2 \widehat{\boldsymbol{x}}_{0|i}\right) \tag{4.77}$$

$$\widetilde{\boldsymbol{x}}_{i-1} = \widetilde{\boldsymbol{x}}_{0|i} + \sigma_{i-1}\boldsymbol{n}_{i-1}, \quad \boldsymbol{n}_{i-1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{4.78}$$

This new formulation is self-consistent, in that, when $\breve{\boldsymbol{x}}_{i+1} \sim \mathcal{N}(\boldsymbol{x}_0, (1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2\boldsymbol{I})$ and $\widetilde{\boldsymbol{x}}_{0|i+1} = \boldsymbol{x}_0$, then $\overline{\boldsymbol{x}}_i \sim \mathcal{N}(\boldsymbol{x}_0, \overline{\sigma}_i^2\boldsymbol{I})$ and $\breve{\boldsymbol{x}}_i \sim \mathcal{N}(\boldsymbol{x}_0, (1+\gamma_i)^2\overline{\sigma}_i^2\boldsymbol{I})$ at each iteration $i$. We now prove this by writing $\breve{\boldsymbol{x}}_{i+1} = \boldsymbol{x}_0 + (1+\gamma_{i+1})\overline{\sigma}_{i+1}\breve{\boldsymbol{e}}_{i+1}$ with $\breve{\boldsymbol{e}}_{i+1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, in which case (4.74) gives

$$\overline{\boldsymbol{x}}_i - \boldsymbol{x}_0 = \frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2}(1+\gamma_{i+1})\overline{\sigma}_{i+1}\breve{\boldsymbol{e}}_{i+1} + \left(1 - \frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2}\right)\sigma_i\boldsymbol{n}_i. \tag{4.79}$$

The variance on the first term in (4.79) is

$$\frac{\overline{\sigma}_i^4}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2} \tag{4.80}$$

and noting from (4.73) that

$$\sigma_i^2 = \overline{\sigma}_i^2\left(1 - \frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2}\right)^{-1} \tag{4.81}$$

the variance on the second term in (4.79) is

$$\left(1-\frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2}\right)^2\sigma_i^2=\left(1-\frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2}\right)^2\overline{\sigma}_i^2\left(1-\frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2}\right)^{-1} \tag{4.82}$$

$$=\left(1-\frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2}\right)\overline{\sigma}_i^2 \tag{4.83}$$

$$=\overline{\sigma}_i^2-\frac{\overline{\sigma}_i^4}{(1+\gamma_{i+1})^2\overline{\sigma}_{i+1}^2}, \tag{4.84}$$

and so, due to the independence between $\breve{\boldsymbol{e}}_{i+1}$ and $\boldsymbol{n}_i$, the total variance of (4.79) is $\overline{\sigma}_i^2$. Thus we see that $\overline{\boldsymbol{x}}_i \sim \mathcal{N}(\boldsymbol{x}_0, \overline{\sigma}_i^2\boldsymbol{I})$. Applying that to (4.75), it's straightforward to see that $\breve{\boldsymbol{x}}_i$ will have error variance $(1+\gamma_i)\overline{\sigma}_i^2$, and so $\breve{\boldsymbol{x}}_i \sim \mathcal{N}(\boldsymbol{x}_0, (1+\gamma_i)^2\overline{\sigma}_i^2\boldsymbol{I})$.

We summarize this new simplified and stochastically enhanced variant of RAPS for SLM inverse problems in Alg. 8. We refer to this algorithm as Stochastic RAPS, or StRAPS for short.

### 4.2.6 Extension to GLM Inverse Problems

We now propose to extend the SLM-StRAPS from Section 4.2.5 to the generalized linear model (GLM)

$$\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{z}_0) = \prod_{j=1}^{m} p_{\mathsf{y}|\mathsf{z}}(y_j|z_{0,j}) \quad \text{with} \quad \boldsymbol{z}_0 \triangleq \boldsymbol{A}\boldsymbol{x}_0 \tag{4.85}$$

where $p_{\mathsf{y}|\mathsf{z}}$ is some scalar "measurement channel." Examples include $p_{\mathsf{y}|\mathsf{z}}(y|z) = \mathcal{N}(y; |z|, \sigma_{\mathsf{w}}^2)$ for phase retrieval, $p_{\mathsf{y}|\mathsf{z}}(y|z) = z^y e^{-z}/y!$ for Poisson regression, and $p_{\mathsf{y}|\mathsf{z}}(y|z) = \int_{\tau_y}^{\tau_{y+1}} \mathcal{N}(\tau; z, \sigma_{\mathsf{w}}^2)\,\mathrm{d}\tau$ for dequantization.

Our extension is inspired by expectation propagation (EP) [119, 157] and its application to GLMs [61, 172]. The idea is to iterate between i) constructing "pseudo-measurements" $\overline{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{x}_0 + \overline{\boldsymbol{w}}$ with $\overline{\boldsymbol{w}} \sim \mathcal{N}(\boldsymbol{0}, \overline{\sigma}_{\mathsf{w}}^2\boldsymbol{I})$ using $p_{\mathsf{y}|\mathsf{z}}$ and an SLM-StRAPS-constructed belief that $\boldsymbol{z}_0 \sim \mathcal{N}(\overline{\boldsymbol{z}}_0, \overline{\sigma}_{\mathsf{z}}^2\boldsymbol{I})$, and then ii) running SLM-StRAPS with

**Algorithm 8** StRAPS for the SLM

**Require:** $\boldsymbol{y}, \boldsymbol{A}, \sigma_{\mathsf{w}}, \{\sigma_i\}_{i=1}^N, \sigma_0 = 0, \{\gamma_i\}, \boldsymbol{d}_{\boldsymbol{\theta}}(\cdot|\sigma)$,

$\widetilde{\boldsymbol{x}}_N \sim \mathcal{N}(\boldsymbol{0}, \sigma_N^2 \boldsymbol{I})$

$\overline{\sigma}_{N+1} = \infty, \quad \overline{\boldsymbol{x}}_{N+1} = \boldsymbol{0}$

**for** $i = N, N-1, \dots, 1$ **do**

  // stochastic recursive update

  $\overline{\sigma}_i^2 = \left( \frac{1}{(1+\gamma_{i+1})^2 \overline{\sigma}_{i+1}^2} + \frac{1}{\sigma_i^2} \right)^{-1}$

  $\overline{\boldsymbol{x}}_i = \frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2 \overline{\sigma}_{i+1}^2} \breve{\boldsymbol{x}}_{i+1} + \left( 1 - \frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2 \overline{\sigma}_{i+1}^2} \right) \widetilde{\boldsymbol{x}}_i$

  $\breve{\boldsymbol{x}}_i = \overline{\boldsymbol{x}}_i + \sqrt{(1+\gamma_i)^2 \overline{\sigma}_i^2 - \overline{\sigma}_i^2} \boldsymbol{w}_i, \quad \boldsymbol{w}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

  // unconditional denoising

  $\widehat{\boldsymbol{x}}_{0|i} = \boldsymbol{d}_{\boldsymbol{\theta}}(\breve{\boldsymbol{x}}_i; (1+\gamma_i)\overline{\sigma}_i)$

  $\widehat{\sigma}_{0|i}^2 = (\|\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{0|i}\|^2 - m\sigma_{\mathsf{w}}^2) / \|\boldsymbol{A}\|_F^2$

  // approximate sampling from $p(\boldsymbol{y}|\boldsymbol{x}_0) p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_i)$

  $\widetilde{\boldsymbol{x}}_{0|i} = \arg\min_{\boldsymbol{x}} \left\{ \frac{1}{2\sigma_{\mathsf{w}}^2} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2 + \frac{1}{2\widehat{\sigma}_{0|i}^2} \|\boldsymbol{x} - \widehat{\boldsymbol{x}}_{0|i}\|^2 \right\}$

  // annealing

  $\widetilde{\boldsymbol{x}}_{i-1} = \widetilde{\boldsymbol{x}}_{0|i} + \sigma_{i-1} \boldsymbol{n}_{i-1}, \quad \boldsymbol{n}_{i-1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

**return** $\widetilde{\boldsymbol{x}}_0$

those pseudo-measurements and updating its belief on $\boldsymbol{z}_0$. Figure 4.1 shows a high-level summary. Details are given below.

First we assume that the denoiser output error is white and Gaussian, i.e., $\boldsymbol{x}_0 \sim \mathcal{N}(\widehat{\boldsymbol{x}}_{0|i}, \widehat{\sigma}_{0|i}^2 \boldsymbol{I})$. We estimate the variance $\widehat{\sigma}_{0|i}^2$ quantity from the denoiser input variance $\sigma^2$ by training a predictor of the form

$$\widehat{\nu}_{\boldsymbol{\phi}}(\sigma) \approx \mathrm{E}\{\|\boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{x}_0 + \sigma\boldsymbol{\epsilon}, \sigma) - \boldsymbol{x}_0\|^2 / d\}, \tag{4.86}$$

where the expectation is over $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and validation images $\boldsymbol{x}_0 \sim p_0$. Recall that $d$ is the dimension of $\boldsymbol{x}_0$. In our experiments, $\widehat{\nu}_{\boldsymbol{\phi}}(\cdot)$ is implemented using a lookup table or zero-order spline.

Next, we construct the belief on $\boldsymbol{z}_0$. Because $\boldsymbol{z}_0 = \boldsymbol{A}\boldsymbol{x}_0$, we see that $\boldsymbol{z}_0 \sim \mathcal{N}(\overline{\boldsymbol{z}}, \widehat{\sigma}_{0|i}^2 \boldsymbol{A}\boldsymbol{A}^\top)$, where $\overline{\boldsymbol{z}} \triangleq \boldsymbol{A}\widehat{\boldsymbol{x}}_{0|i}$. For simplicity, however, we use the white-noise

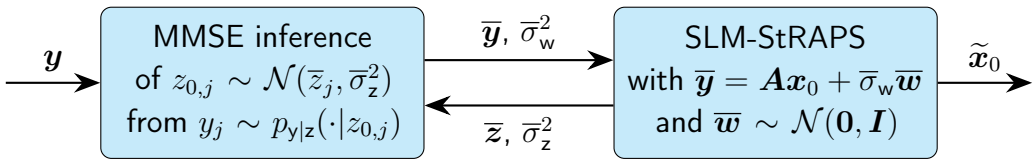


Figure 4.1: High-level overview of GLM-StRAPS, which uses EP-style iterations between SLM-StRAPS and an MMSE inference stage that involves the scalar measurement channel $p_{\mathsf{y}|\mathsf{z}}$.

approximation $\boldsymbol{z}_0 \sim \mathcal{N}(\overline{\boldsymbol{z}}, \overline{\sigma}_{\mathsf{z}}^2\boldsymbol{I})$, where $\overline{\sigma}_{\mathsf{z}}^2 \triangleq \widehat{\sigma}_{0|l}^2\|\boldsymbol{A}\|_F^2/m$. Using the scalar belief $z_{0,j} \sim \mathcal{N}(\overline{z}_j, \overline{\sigma}_{\mathsf{z}}^2)$ and the likelihood model $y_j \sim p_{\mathsf{y}|\mathsf{z}}(\cdot|z_{0,j})$, EP suggests to first compute the posterior mean $\mathrm{E}\{z_{0,j}|y_j; \overline{z}_j, \overline{\sigma}_{\mathsf{z}}^2\} \triangleq \widehat{z}_j$ and variance $\frac{1}{m}\sum_{j=1}^m \mathrm{var}\{z_{0,j}|y_j; \overline{z}_j, \overline{\sigma}_{\mathsf{z}}^2\} \triangleq \widehat{\sigma}_{\mathsf{z}}^2$, and then pass the "extrinsic" versions of those quantities:

$$\overline{\sigma}_{\mathsf{w}}^2 \triangleq [1/\widehat{\sigma}_{\mathsf{z}}^2 - 1/\overline{\sigma}_{\mathsf{z}}^2]^{-1}, \quad \overline{\boldsymbol{y}} \triangleq \overline{\sigma}_{\mathsf{w}}^2(\widehat{\boldsymbol{z}}/\widehat{\sigma}_{\mathsf{z}}^2 - \overline{\boldsymbol{z}}/\overline{\sigma}_{\mathsf{z}}^2) \tag{4.87}$$

back to SLM-StRAPS, where they are used to construct the pseudo-measurement model

$$\overline{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{x}_0 + \overline{\sigma}_{\mathsf{w}}\overline{\boldsymbol{w}}, \quad \overline{\boldsymbol{w}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{4.88}$$

The GLM-StRAPS algorithm is summarized as Alg. 9. There, the dashed blue box surrounds the lines used for the EP update; all other lines are taken directly from SLM-StRAPS. As can be seen, GLM-StRAPS performs one EP update per SLM-StRAPS iteration. When $p_{\mathsf{y}|\mathsf{z}}(y|z) = \mathcal{N}(y; z, \sigma_{\mathsf{w}}^2)$, is it straightforward to show that $\overline{\boldsymbol{y}} = \boldsymbol{y}$ and $\overline{\sigma}_{\mathsf{w}}^2 = \sigma_{\mathsf{w}}^2$ for any $\overline{\boldsymbol{z}}$ and $\overline{\sigma}_{\mathsf{z}}^2$, in which case GLM-StRAPS reduces to SLM-StRAPS.

**Algorithm 9** StRAPS for the GLM

**Require:** $\boldsymbol{y}, \boldsymbol{A}, p_{\mathsf{y}|\mathsf{z}}, \{\sigma_i\}_{i=1}^N, \sigma_0 = 0, \{\gamma_i\}, \boldsymbol{d}_{\boldsymbol{\theta}}(\cdot|\sigma),$

$\widetilde{\boldsymbol{x}}_N \sim \mathcal{N}(\mathbf{0}, \sigma_N^2 \boldsymbol{I})$

$\overline{\sigma}_{N+1} = \infty, \quad \overline{\boldsymbol{x}}_{N+1} = \mathbf{0}$

**for** $i = N, N-1, \dots, 1$ **do**

  // stochastic recursive update

  $\overline{\sigma}_i^2 = \left( \frac{1}{(1+\gamma_{i+1})^2 \overline{\sigma}_{i+1}^2} + \frac{1}{\sigma_i^2} \right)^{-1}$

  $\overline{\boldsymbol{x}}_i = \frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2 \overline{\sigma}_{i+1}^2} \breve{\boldsymbol{x}}_{i+1} + \left( 1 - \frac{\overline{\sigma}_i^2}{(1+\gamma_{i+1})^2 \overline{\sigma}_{i+1}^2} \right) \widetilde{\boldsymbol{x}}_i$

  $\breve{\boldsymbol{x}}_i = \overline{\boldsymbol{x}}_i + \sqrt{(1+\gamma_i)^2 \overline{\sigma}_i^2 - \overline{\sigma}_i^2} \boldsymbol{w}_i, \quad \boldsymbol{w}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

  // unconditional denoising

  $\widehat{\boldsymbol{x}}_{0|i} = \boldsymbol{d}_{\boldsymbol{\theta}}(\breve{\boldsymbol{x}}_i; (1+\gamma_i)\overline{\sigma}_i)$

  $\widehat{\sigma}_{0|i}^2 \leftarrow \widehat{\nu}_{\boldsymbol{\phi}}((1+\gamma_i)\overline{\sigma}_i)$

  // EP update

  $\overline{\boldsymbol{z}} \leftarrow \boldsymbol{A} \widehat{\boldsymbol{x}}_{0|i}$

  $\overline{\sigma}_{\mathsf{z}}^2 \leftarrow \widehat{\sigma}_{0|i}^2 \, \|\boldsymbol{A}\|_F^2 / m$ ▷ *Error variance of* $\overline{\boldsymbol{z}}$

  $\widehat{z}_j \leftarrow \mathrm{E}\{z_{0,j} | y_j; \overline{z}_j, \overline{\sigma}_{\mathsf{z}}^2\} \; \forall j = 1, \dots, m$ ▷ *Posterior mean estimation*

  $\widehat{\sigma}_{\mathsf{z}}^2 \leftarrow \frac{1}{m} \sum_{j=1}^m \mathrm{var}\{z_{0,j} | y_j; \overline{z}_j, \overline{\sigma}_{\mathsf{z}}^2\}$ ▷ *Averaged posterior variance of* $\{z_{0,j}\}$

  $\overline{\sigma}_{\mathsf{w}}^2 \leftarrow [1/\widehat{\sigma}_{\mathsf{z}}^2 - 1/\overline{\sigma}_{\mathsf{z}}^2]^{-1}$ ▷ *Extrinsic variance*

  $\overline{\boldsymbol{y}} \leftarrow \overline{\sigma}_{\mathsf{w}}^2 (\widehat{\boldsymbol{z}}/\widehat{\sigma}_{\mathsf{z}}^2 - \overline{\boldsymbol{z}}/\overline{\sigma}_{\mathsf{z}}^2)$ ▷ *Extrinsic mean*

  $\widehat{\sigma}_{0|i}^2 = (\|\overline{\boldsymbol{y}} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{0|i}\|^2 - m\overline{\sigma}_{\mathsf{w}}^2) / \|\boldsymbol{A}\|_F^2$

  // approximate sampling from $p(\boldsymbol{y}|\boldsymbol{x}_0) p(\boldsymbol{x}_0|\overline{\boldsymbol{x}}_i)$

  $\widetilde{\boldsymbol{x}}_{0|i} = \arg\min_{\boldsymbol{x}} \left\{ \frac{1}{2\overline{\sigma}_{\mathsf{w}}^2} \|\overline{\boldsymbol{y}} - \boldsymbol{A}\boldsymbol{x}\|^2 + \frac{1}{2\widehat{\sigma}_{0|i}^2} \|\boldsymbol{x} - \widehat{\boldsymbol{x}}_{0|i}\|^2 \right\}$

  // annealing

  $\widetilde{\boldsymbol{x}}_{i-1} = \widetilde{\boldsymbol{x}}_{0|i} + \sigma_{i-1} \boldsymbol{n}_{i-1}, \quad \boldsymbol{n}_{i-1} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

**return** $\widetilde{\boldsymbol{x}}_0$

## 4.3 Numerical Experiments

We use the $256 \times 256$ FFHQ dataset [158] with pretrained diffusion models from [76]. For linear inverse problems, we consider the following tasks: box inpainting with a $128 \times 128$ mask, Gaussian deblurring using a $61 \times 61$ blur kernel with a standard deviation of 3 pixels, motion deblurring with a $61 \times 61$ blur kernel of intensity 0.5, generated using [173], and $4\times$ bicubic super-resolution.

Table 4.1: Noisy FFHQ results with measurement noise standard deviation $\sigma_{\mathsf{w}} = 0.05$.

| | | Inpaint (box) | | | Deblur (Gaussian) | | | Deblur (Motion) | | | 4× Super-resolution | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # NFEs | Model | PSNR↑ | LPIPS↓ | FID↓ | PSNR↑ | LPIPS↓ | FID↓ | PSNR↑ | LPIPS↓ | FID↓ | PSNR↑ | LPIPS↓ | FID↓ |
| 20 | DiffPIR | 21.86 | 0.2152 | 41.50 | 23.55 | 0.2720 | **31.21** | **27.32** | 0.2031 | **28.27** | 22.26 | 0.2979 | 44.69 |
| | DDRM | 21.71 | 0.1551 | 40.61 | 25.35 | 0.2223 | 51.70 | - | - | - | **27.32** | **0.1764** | 45.82 |
| | StRAPS | **21.99** | **0.1335** | **35.70** | **26.68** | **0.1712** | 32.25 | 26.77 | **0.1622** | 30.07 | 26.76 | 0.1822 | **35.49** |
| 100 | DiffPIR | 22.43 | 0.1883 | 31.98 | 24.57 | 0.2394 | 26.78 | **26.91** | 0.1952 | **24.67** | 24.89 | 0.2486 | 32.26 |
| | ΠGDM | 21.41 | 0.2009 | 44.41 | 23.66 | 0.2525 | 45.33 | 25.14 | 0.2082 | 41.95 | 24.40 | 0.2520 | 51.40 |
| | StRAPS | **22.74** | **0.1195** | **31.91** | **26.57** | **0.1584** | **26.20** | 26.82 | **0.1548** | 26.51 | **26.86** | **0.1701** | **29.65** |
| 1000 | DPS | 22.54 | 0.1368 | 29.97 | 25.70 | 0.1774 | **25.18** | 26.74 | 0.1655 | 27.17 | 26.30 | 0.1830 | 27.38 |
| | DAPS | **23.61** | 0.1415 | 31.51 | **26.97** | 0.1827 | 31.10 | 27.13 | 0.1718 | 30.74 | **26.91** | 0.1885 | 30.83 |
| | StRAPS | 23.36 | **0.1097** | **27.95** | 26.86 | **0.1527** | 26.85 | **27.14** | **0.1479** | **25.74** | 26.83 | **0.1635** | **27.21** |

We compare our approach against various baselines: DDRM [91] and DiffPIR [90] at 20 NFEs, ΠGDM [93] and DiffPIR at 100 NFEs, and DPS [76] and DAPS [97] at 1000 NFEs.

We also consider phase retrieval with the shot-noise corruption mechanism from [56] for both oversampled Fourier (OSF) and coded diffraction pattern (CDP) [156] $\boldsymbol{A}$ at 4× oversampling with $\alpha_{\mathsf{shot}} = 8$ and 45, respectively. Here, $\alpha_{\mathsf{shot}}$ is the shot-noise strength, as detailed in Appendix F. We compare to prDeep [56], DOLPH [77], DPS, DAPS, and the classical hybrid input-output (HIO) algorithm [49], using $p_{\mathsf{y}|\mathsf{z}}(y|z) = \mathcal{N}(y; |z|, \sigma_{\mathsf{w}}^2)$ for all algorithms that accept a likelihood function. Since the chosen likelihood renders the conditional mean and variance in lines 15-14 of Alg. 9 intractable, we employ the Laplace approximation [157].

For StRAPS, we use a fixed $\gamma_i$ across all iterations, tuned to minimize LPIPS [174] on a separate 100-sample validation set. Appendix F provides the tuned $\gamma_i$ values along with additional implementation details for StRAPS and the competing methods.

For noisy linear inverse problems, Table 4.1 show PSNR, LPIPS, and FID [175] on a 1000-sample test set for FFHQ data. DDRM was not applied to motion deblurring

Table 4.2: Noisy FFHQ phase retrieval results

| | | OSF | | | CDP | | |
|---|---|---|---|---|---|---|---|
| # NFEs | Model | PSNR↑ | LPIPS↓ | FID↓ | PSNR↑ | LPIPS↓ | FID↓ |
| - | HIO | 23.66 | 0.4706 | 130.58 | 17.59 | 0.5430 | 84.87 |
| 1000 | DOLPH | 14.73 | 0.7220 | 389.88 | 25.76 | 0.1686 | 32.93 |
| 1000 | DPS | 23.63 | 0.2908 | 53.91 | 29.19 | 0.1394 | 27.87 |
| 1000 | DAPS | 24.10 | 0.2891 | 57.73 | 28.26 | 0.1927 | 34.97 |
| 800 | prDeep | 30.90 | 0.1132 | 31.51 | 19.24 | 0.4183 | 59.44 |
| 800 | StRAPS | **33.52** | **0.0698** | **24.86** | 29.72 | **0.1367** | **24.52** |
| 100 | StRAPS | 27.91 | 0.1808 | 38.63 | **29.79** | 0.1395 | 26.42 |

due to the lack of an SVD. Table 4.1 show that, when comparing to competitors at equal NFEs, StRAPS wins in most cases and otherwise performs well.

Fig. 4.2 shows image examples for inpainting and Gaussian deblurring, and Fig. 4.3 presents image examples for motion deblurring and 4× super-resolution on FFHQ. The zoomed regions illustrate that StRAPS more effectively recovers fine details while avoiding artifacts.

Table 4.2 shows performance on noisy FFHQ phase retrieval. For OSF $\boldsymbol{A}$, we see that StRAPS at 800 NFEs outperforms prDeep by a small margin and surpasses DAPS, DPS, and DOLPH by a large margin. Also, StRAPS at 100 NFEs beats DAPS, DPS, and DOLPH at 1000 NFEs. For CDP $\boldsymbol{A}$, we see that StRAPS performs similarly at 100 and 800 NFEs and that both outperform the other methods. Example reconstructions for phase retrieval can be found in Fig. 4.4.

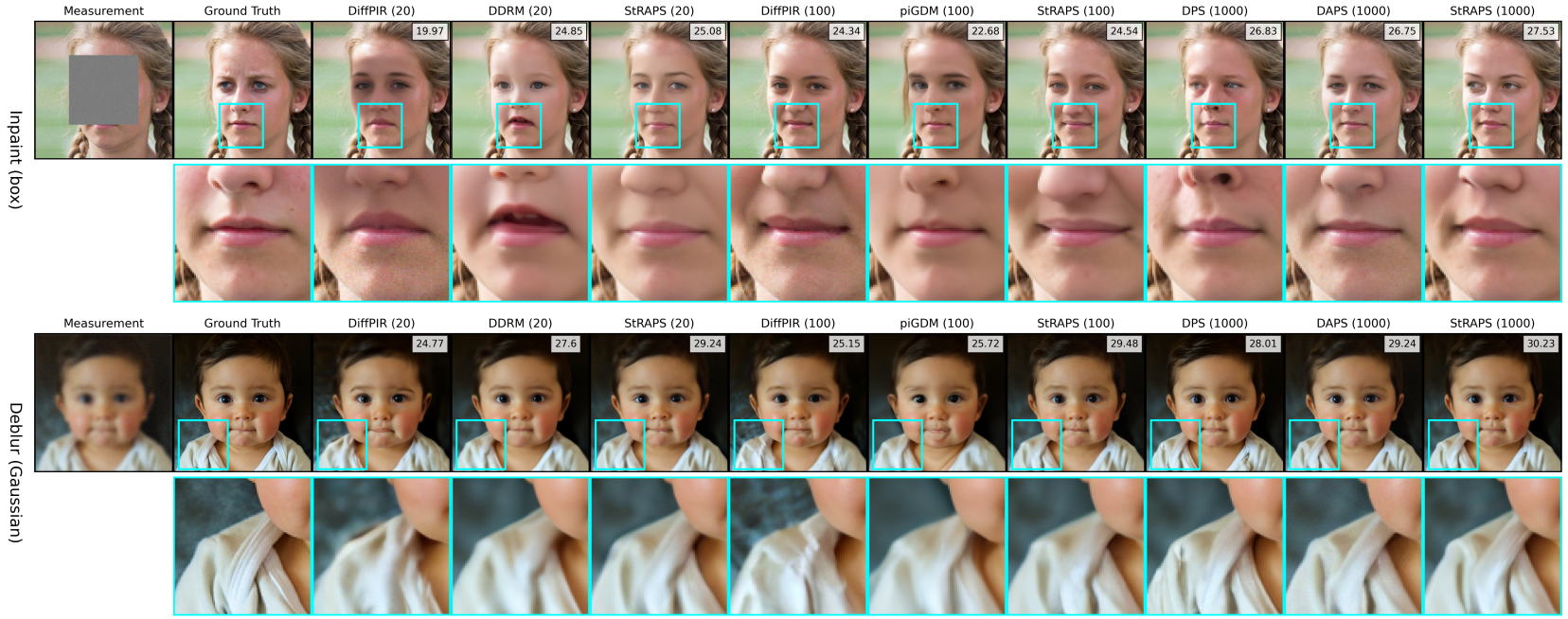


Figure 4.2: Example recoveries from noisy linear inverse problems (inpainting and Gaussian deblurring) with FFHQ images.

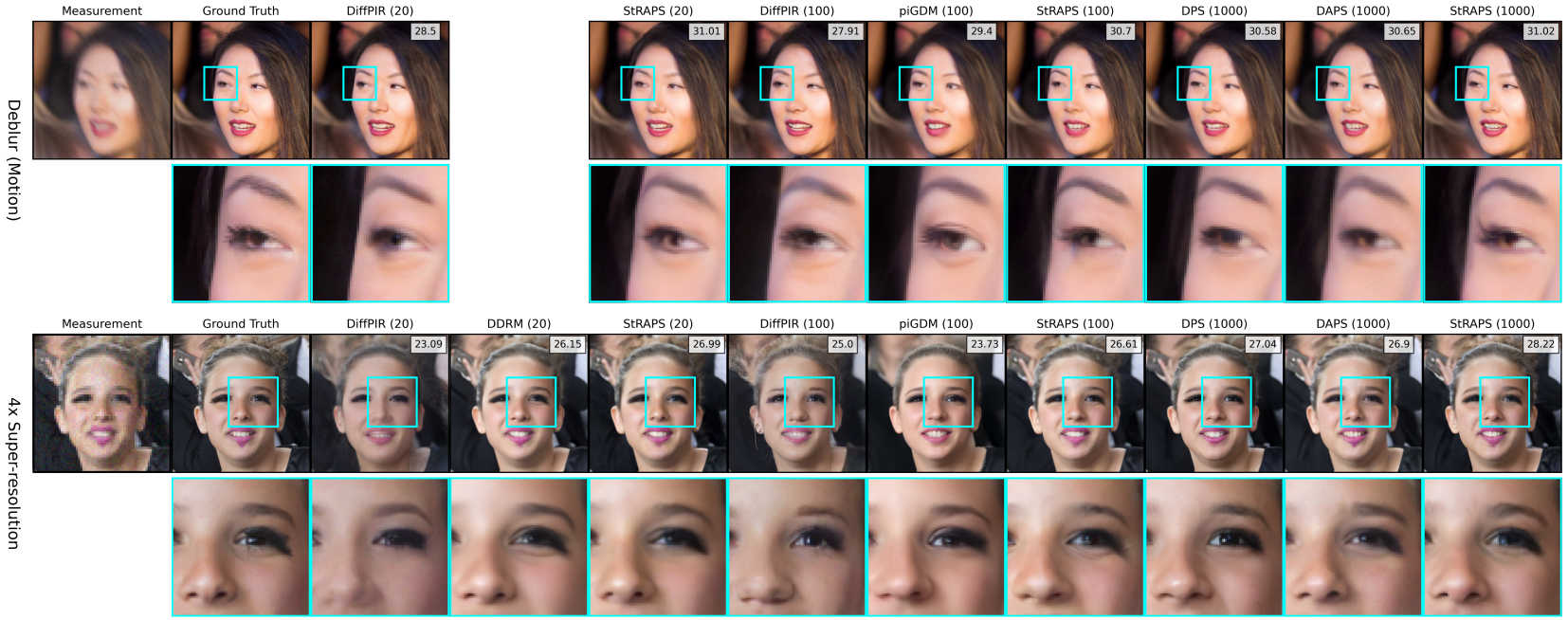


Figure 4.3: Example recoveries from noisy linear inverse problems (motion deblurring and 4× super-resolution) with FFHQ images.

## 4.4 Conclusion

Inspired by the idea of using a decoupled forward process during reverse diffusion, we introduced a novel unsupervised diffusion posterior sampling approach called Recursive Annealed Posterior Sampling (RAPS). By carefully restructuring the reverse process update equations, we demonstrated that RAPS aligns with the widely used SMLD framework. We further extended RAPS by proposing StRAPS, which incorporated key simplifications and enhancements, including automatic tuning of the regularization weight and the introduction of stochasticity to mitigate discretization and approximation errors during sampling. Our proposed framework can be utilized to solve both linear and generalized-linear inverse problems, such as phase retrieval. Experiments on box inpainting, Gaussian and motion deblurring, and 4× super-resolution with FFHQ images demonstrated that StRAPS outperformed DAPS, DPS, ΠGDM, DiffPIR, and DDRM at equal NFEs in terms of PSNR, LPIPS, and FID in most cases. Additionally, evaluations on noisy FFHQ phase retrieval, including both OSF and CDP variants, showed that StRAPS consistently surpassed DAPS, DPS, DOLPH, prDeep, and HIO. Beyond its accuracy, StRAPS is computationally efficient, supports batch generation, and benefits from SVD-based acceleration when available, making it a practical and scalable solution for inverse problems. Future work will extend StRAPS to nonlinear inverse problems beyond those that conform to generalized-linear models.

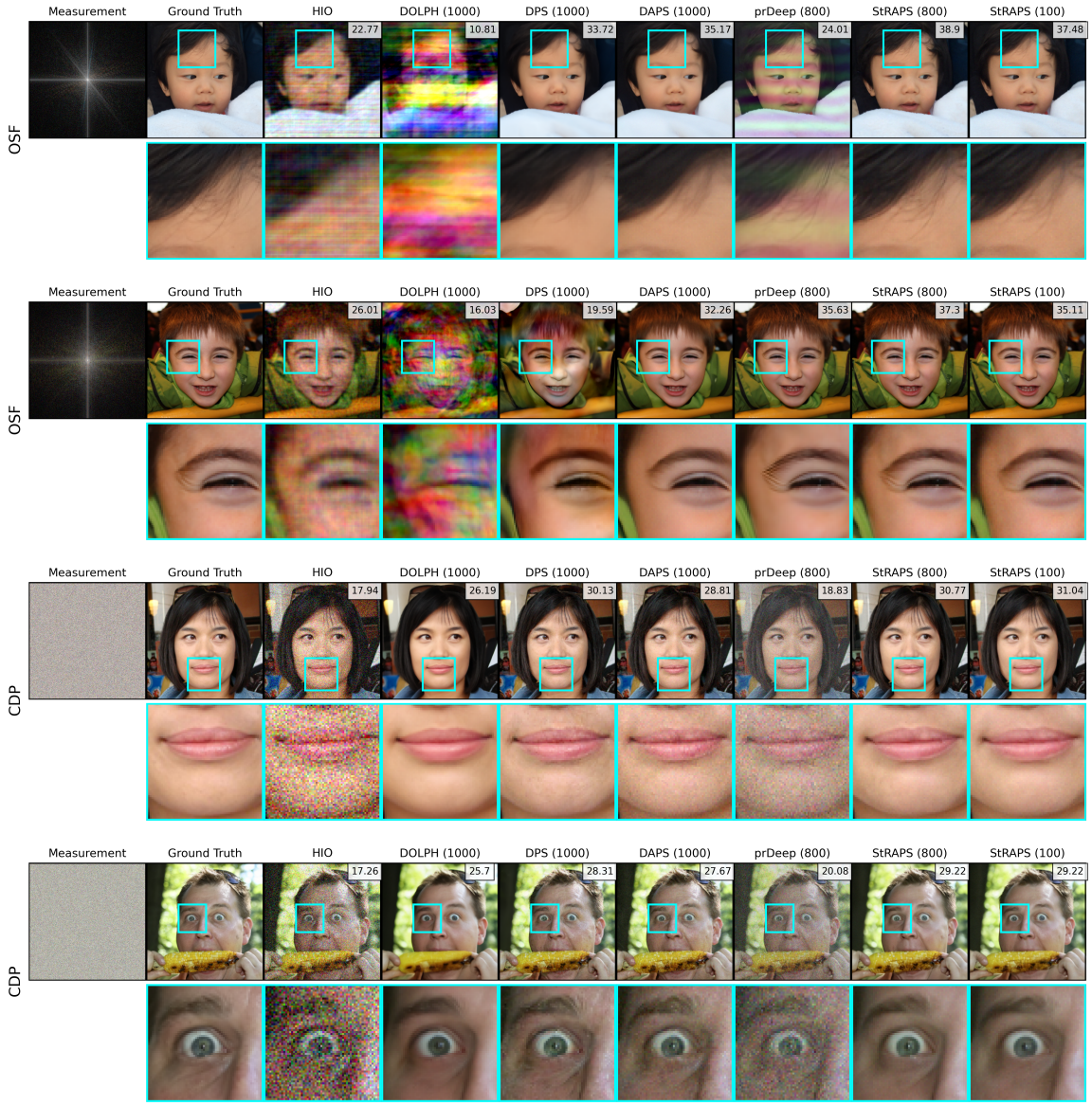


Figure 4.4: Example recoveries from noisy phase retrieval with FFHQ images.

# Bibliography


[1] R. Wang and D. Tao, "Recent progress in image deblurring," *arXiv:1409.6838*, 2014.

[2] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: A technical overview," *IEEE Signal Process. Mag.*, vol. 20, pp. 21–36, May 2003.

[3] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao, "Deep learning for single image super-resolution: A brief review," *IEEE Trans. Multimedia*, vol. 21, pp. 3106–3121, Dec. 2019.

[4] C. Guillemot and O. Le Meur, "Image inpainting: Overview and recent advances," *IEEE Signal Process. Mag.*, vol. 31, pp. 127–144, Jan. 2014.

[5] F. Knoll, K. Hammernik, C. Zhang, S. Moeller, T. Pock, D. K. Sodickson, and M. Akcakaya, "Deep-learning methods for parallel magnetic resonance imaging reconstruction: A survey of the current approaches, trends, and issues," *IEEE Signal Process. Mag.*, vol. 37, pp. 128–140, Jan. 2020.

[6] M. Unser and M. T. McCann, "Biomedical image reconstruction: From the foundations to deep neural networks," *Found. Trends Signal Process.*, vol. 13, pp. 280–359, Jan. 2019.

[7] F. Soulez, L. Denis, É. Thiébaut, C. Fournier, and C. Goepfert, "Inverse problem approach in particle digital holography: Out-of-field particle detection made possible," *J. Opt. Soc. Amer. A*, vol. 24, no. 12, pp. 3708–3716, 2007.

[8] R. Venkataramanan, S. Tatikonda, and A. Barron, "Sparse regression codes," *Found. Trends Commun. Info. Thy.*, vol. 15, no. 1-2, pp. 1–195, 2019.

[9] J. A. Fessler, "Optimization methods for magnetic resonance image reconstruction," *IEEE Signal Process. Mag.*, vol. 37, pp. 33–40, Jan. 2020.

[10] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D Nonlinear Phenom.*, vol. 60, no. 1-4, pp. 259–268, 1992.

[11] Y. Shi and Q. Chang, "Efficient algorithm for isotropic and anisotropic total variation deblurring and denoising," *J. Appl. Math.*, vol. 2013, Mar. 2013. Art. no. 797239.

[12] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Trans. Image Process.*, vol. 26, pp. 4509–4522, Sept. 2017.

[13] G. Yang, S. Yu, H. Dong, G. Slabaugh, P. L. Dragotti, X. Ye, F. Liu, S. Arridge, J. Keegan, Y. Guo, *et al.*, "DAGAN: Deep de-aliasing generative adversarial networks for fast compressed sensing MRI reconstruction," *IEEE Trans. Med. Imag.*, vol. 37, pp. 1310–1321, June 2018.

[14] K. Hammernik, T. Klatzer, E. Kobler, M. P. Recht, D. K. Sodickson, T. Pock, and F. Knoll, "Learning a variational network for reconstruction of accelerated MRI data," *Magn. Reson. Med.*, vol. 79, no. 6, pp. 3055–3071, 2018.

[15] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, pp. 18–44, Mar. 2021.

[16] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, "Compressed sensing using generative models," in *Proc. Intl. Conf. Mach. Learn.*, pp. 537–546, 2017.

[17] P. Hand and V. Voroninski, "Global guarantees for enforcing deep generative priors by empirical risk," in *Proc. Conf. Learn. Theory*, pp. 970–978, 2018.

[18] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, "Solving inverse problems using data-driven models," *Acta Numerica*, vol. 28, pp. 1–174, June 2019.

[19] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, "Deep learning techniques for inverse problems in imaging," *IEEE J. Sel. Areas Info. Theory*, vol. 1, pp. 39–56, May 2020.

[20] K. Hammernik, T. Küstner, B. Yaman, Z. Huang, D. Rueckert, F. Knoll, and M. Akçakaya, "Physics-driven deep learning for computational magnetic resonance imaging," *arXiv:2203.12215*, 2022.

[21] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Proc. IEEE Global Conf. Signal Info. Process.*, pp. 945–948, 2013.

[22] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1804–1844, 2017.

[23] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Trans. Comput. Imag.*, vol. 5, pp. 52–67, Mar. 2019.

[24] R. Ahmad, C. A. Bouman, G. T. Buzzard, S. Chan, S. Liu, E. T. Reehorst, and P. Schniter, "Plug and play methods for magnetic resonance imaging," *IEEE Signal Process. Mag.*, vol. 37, pp. 105–116, Mar. 2020.

[25] D. Gilton, G. Ongie, and R. Willett, "Deep equilibrium architectures for inverse problems in imaging," *IEEE Trans. Comput. Imag.*, vol. 7, pp. 1123–1133, 2021.

[26] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.*, vol. 106, pp. 18914–18919, Nov. 2009.

[27] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, pp. 3142–3155, July 2017.

[28] A. K. Fletcher, M. Sahraee-Ardakan, S. Rangan, and P. Schniter, "Expectation consistent approximate inference: Generalizations and convergence," in *Proc. IEEE Intl. Symp. Info. Theory*, pp. 190–194, 2016.

[29] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*. San Diego, CA, USA: Academic Press, 3rd ed., 2008.

[30] J. Zbontar, F. Knoll, A. Sriram, M. J. Muckley, M. Bruno, A. Defazio, M. Parente, K. J. Geras, J. Katsnelson, H. Chandarana, Z. Zhang, M. Drozdzal, A. Romero, M. Rabbat, P. Vincent, J. Pinkerton, D. Wang, N. Yakubova, E. Owens, C. L. Zitnick, M. P. Recht, D. K. Sodickson, and Y. W. Lui, "fastMRI: An open dataset and benchmarks for accelerated MRI," *arXiv:1811.08839*, 2018.

[31] F. Ong, S. Amin, S. Vasanawala, and M. Lustig, "Mridata.org: An open archive for sharing MRI raw data," in *Proc. Annu. Meeting ISMRM*, vol. 26, 2018.

[32] P. McCullagh and J. A. Nelder, *Generalized Linear Models*. London: Chapman & Hall/CRC, 2nd ed., 1989.

[33] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd ed., 2009.

[34] M. A. T. Figueiredo and J. M. Bioucas-Dias, "Restoration of Poissonian images using alternating direction optimization," *IEEE Trans. Image Process.*, vol. 19, no. 12, pp. 3133–3145, 2010.

[35] A. Zymnis, S. Boyd, and E. Candès, "Compressed sensing with quantized measurements," *IEEE Signal Process. Lett.*, vol. 17, no. 2, pp. 149–152, 2010.

[36] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, "Phase retrieval with application to optical imaging: A contemporary overview," *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 87–109, 2015.

[37] J. Dong, L. Valzania, A. Maillard, T.-A. Pham, S. Gigan, and M. Unser, "Phase retrieval: From computational imaging to machine learning: A tutorial," *IEEE Signal Process. Mag.*, vol. 40, no. 1, pp. 45–57, 2023.

[38] A. Walther, "The question of phase retrieval in optics," *Optica Acta*, vol. 10, no. 1, pp. 41–49, 1963.

[39] M. Stefik, "Inferring DNA structures from segmentation data," *Artificial Intelligence*, vol. 11, no. 1-2, pp. 85–114, 1978.

[40] J. C. Dainty and J. R. Fienup, "Phase retrieval and image construction for astronomy," in *Image Recovery: Theory and Application* (H. Stark, ed.), ch. 7, pp. 231–275, New York: Academic Press, 1987.

[41] R. P. Millane, "Phase retrieval in crystallography and optics," *J. Opt. Soc. Amer. A*, vol. 7, pp. 394–411, Mar. 1990.

[42] R. W. Harrison, "Phase problem in crystallography," *J. Opt. Soc. Amer. A*, vol. 10, no. 5, pp. 1046–1055, 1993.

[43] J. Miao, P. Charalambous, J. Kirz, and D. Sayre, "Extending the methodology of X-ray crystallography to allow imaging of micrometre-sized non-crystalline specimens," *Nature*, vol. 400, no. 6742, pp. 342–344, 1999.

[44] R. Balan, P. G. Casazza, and D. Edidin, "On signal reconstruction without phase," *Appl. Comput. Harmonic Anal.*, vol. 20, pp. 345–356, May 2006.

[45] F. Hüe, J. Rodenburg, A. Maiden, F. Sweeney, and P. Midgley, "Wave-front phase retrieval in transmission electron microscopy via ptychography," *Physical Rev. B*, vol. 82, no. 12, p. 121415, 2010.

[46] J. Zhang, N. Pégard, J. Zhong, H. Adesnik, and L. Waller, "3D computer-generated holography by non-convex optimization," *Optica*, vol. 4, no. 10, pp. 1306–1313, 2017.

[47] C. A. Metzler, F. Heide, P. Rangarajan, M. M. Balaji, A. Viswanath, A. Veeraraghavan, and R. G. Baraniuk, "Deep-inverse correlography: Towards real-time high-resolution non-line-of-sight imaging," *Optica*, vol. 7, no. 1, pp. 63–71, 2020.

[48] R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of the phase from image and diffraction plane pictures," *Optika*, vol. 35, no. 2, pp. 237–246, 1972.

[49] J. R. Fienup, "Phase retrieval algorithms: A comparison," *Appl. Optics*, vol. 21, pp. 2758–2769, Aug. 1982.

[50] H. H. Bauschke, P. L. Combettes, and D. R. Luke, "Hybrid projection-reflection method for phase retrieval," *J. Opt. Soc. Amer. A*, vol. 20, no. 6, pp. 1025–1034, 2003.

[51] V. Elser, "Phase retrieval by iterated projections," *J. Opt. Soc. Amer. A*, vol. 20, no. 1, pp. 40–55, 2003.

[52] C.-C. Chen, J. Miao, C. Wang, and T. Lee, "Application of optimization technique to noncrystalline X-ray diffraction microscopy: Guided hybrid input-output method," *Physical Rev. B*, vol. 76, no. 6, p. 064113, 2007.

[53] E. J. Candès, X. Li, and M. Soltanolkotabi, "Phase retrieval via Wirtinger flow: Theory and algorithms," *IEEE Trans. Info. Theory*, vol. 61, no. 4, pp. 1985–2007, 2015.

[54] M. Mondelli and A. Montanari, "Fundamental limits of weak recovery with applications to phase retrieval," in *Proc. Conf. Learn. Theory*, pp. 1445–1450, 2018.

[55] W. Luo, W. Alghamdi, and Y. M. Lu, "Optimal spectral initialization for signal recovery with applications to phase retrieval," *IEEE Trans. Signal Process.*, vol. 67, no. 9, pp. 2347–2356, 2019.

[56] C. A. Metzler, P. Schniter, A. Veeraraghavan, and R. G. Baraniuk, "prDeep: Robust phase retrieval with flexible deep neural networks," in *Proc. Intl. Conf. Mach. Learn.*, pp. 3501–3510, 2018.

[57] E. J. Candès, T. Strohmer, and V. Voroninski, "PhaseLift: Exact and stable signal recovery from magnitude measurements via convex programming," *Commun. Pure & Appl. Math.*, vol. 66, no. 8, pp. 1241–1274, 2013.

[58] T. Goldstein and C. Studer, "Phasemax: Convex phase retrieval via basis pursuit," *IEEE Trans. Info. Theory*, vol. 64, no. 4, pp. 2675–2689, 2018.

[59] A. Fannjiang and T. Strohmer, "The numerics of phase retrieval," *Acta Numerica*, vol. 29, pp. 125–228, 2020.

[60] P. Schniter and S. Rangan, "Compressive phase retrieval via generalized approximate message passing," *IEEE Trans. Signal Process.*, vol. 63, pp. 1043–1055, Feb. 2015.

[61] P. Schniter, S. Rangan, and A. K. Fletcher, "Vector approximate message passing for the generalized linear model," in *Proc. Asilomar Conf. Signals Syst. Comput.*, pp. 1525–1529, 2016.

[62] A. Maillard, B. Loureiro, F. Krzakala, and L. Zdeborová, "Phase retrieval in high dimensions: Statistical and computational phase transitions," in *Proc. Neural Info. Process. Syst. Conf.*, vol. 33, pp. 11071–11082, 2020.

[63] M. Mondelli and R. Venkataramanan, "Approximate message passing with spectral initialization for generalized linear models," in *Proc. Intl. Conf. Artificial Intell. Statist.*, pp. 397–405, 2021.

[64] F. Heide, S. Diamond, M. Nießner, J. Ragan-Kelley, W. Heidrich, and G. Wetzstein, "Proximal: Efficient image optimization using proximal algorithms," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–15, 2016.

[65] Y. Wang, X. Sun, and J. Fleischer, "When deep denoising meets iterative phase retrieval," in *Proc. Intl. Conf. Mach. Learn.*, pp. 10007–10017, 2020.

[66] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "BM3D-PRGAMP: Compressive phase retrieval based on BM3D denoising," in *Proc. IEEE Intl. Conf. Image Process.*, pp. 2504–2508, 2016.

[67] C. A. Metzler, M. K. Sharma, S. Nagesh, R. G. Baraniuk, O. Cossairt, and A. Veeraraghavan, "Coherent inverse scattering via transmission matrices: Efficient phase retrieval algorithms and a public dataset," in *Proc. Intl. Conf. Comput. Photog.*, pp. 1–16, 2017.

[68] P. Hand, O. Leong, and V. Voroninski, "Phase retrieval under a generative prior," in *Proc. Neural Info. Process. Syst. Conf.*, vol. 31, 2018.

[69] F. Wang, Y. Bian, H. Wang, M. Lyu, G. Pedrini, W. Osten, G. Barbastathis, and G. Situ, "Phase imaging with an untrained neural network," *Light: Science & Applications*, vol. 9, no. 1, p. 77, 2020.

[70] E. Bostan, R. Heckel, M. Chen, M. Kellman, and L. Waller, "Deep phase decoder: Self-calibrating phase microscopy with an untrained deep neural network," *Optica*, vol. 7, no. 6, pp. 559–562, 2020.

[71] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, pp. 9446–9454, 2018.

[72] M. Chen, P. Lin, Y. Quan, T. Pang, and H. Ji, "Unsupervised phase retrieval using deep approximate MMSE estimation," *IEEE Trans. Signal Process.*, vol. 70, pp. 2239–2252, 2022.

[73] A. Sinha, J. Lee, S. Li, and G. Barbastathis, "Lensless computational imaging through deep learning," *Optica*, vol. 4, no. 9, pp. 1117–1125, 2017.

[74] Y. Rivenson, Y. Zhang, H. Günaydın, D. Teng, and A. Ozcan, "Phase recovery and holographic image reconstruction using deep learning in neural networks," *Light: Science & Applications*, vol. 7, no. 2, p. 17141, 2018.

[75] N. Torem, R. Ronen, Y. Y. Schechner, and M. Elad, "Complex-valued retrievals from noisy images using diffusion models," in *Proc. IEEE Intl. Conf. Comput. Vis.*, pp. 3810–3820, 2023.

[76] H. Chung, J. Kim, M. T. McCann, M. L. Klasky, and J. C. Ye, "Diffusion posterior sampling for general noisy inverse problems," in *Proc. Intl. Conf. Learn. Rep.*, 2023.

[77] S. Shoushtari, J. Liu, and U. S. Kamilov, "Diffusion models for phase retrieval in computational imaging," in *Proc. Asilomar Conf. Signals Syst. Comput.*, pp. 779–783, 2023.

[78] X. Xu and Y. Chi, "Provably robust score-based diffusion posterior sampling for plug-and-play image reconstruction," in *Proc. Neural Info. Process. Syst. Conf.*, 2024.

[79] Z. Wu, Y. Sun, Y. Chen, B. Zhang, Y. Yue, and K. Bouman, "Principled probabilistic imaging using diffusion models as plug-and-play priors," in *Proc. Neural Info. Process. Syst. Conf.*, 2024.

[80] M. Opper and O. Winther, "Expectation consistent approximate inference," *J. Mach. Learn. Res.*, vol. 1, pp. 2177–2204, 2005.

[81] S. Rangan, P. Schniter, and A. K. Fletcher, "Vector approximate message passing," *IEEE Trans. Info. Theory*, vol. 65, pp. 6664–6684, Oct. 2019.

[82] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. Intl. Conf. Mach. Learn.*, pp. 2256–2265, 2015.

[83] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Proc. Neural Info. Process. Syst. Conf.*, 2019.

[84] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Neural Info. Process. Syst. Conf.*, vol. 33, pp. 6840–6851, 2020.

[85] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *Proc. Intl. Conf. Learn. Rep.*, 2021.

[86] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Proc. Intl. Conf. Learn. Rep.*, 2021.

[87] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," in *Proc. Neural Info. Process. Syst. Conf.*, vol. 35, pp. 26565–26577, 2022.

[88] G. Daras, H. Chung, C.-H. Lai, Y. Mitsufuji, J. C. Ye, P. Milanfar, A. G. Dimakis, and M. Delbracio, "A survey on diffusion models for inverse problems," *arXiv:2410.00083*, 2024.

[89] Y. Wang, J. Yu, and J. Zhang, "Zero-shot image restoration using denoising diffusion null-space model," in *Proc. Intl. Conf. Learn. Rep.*, 2023.

[90] Y. Zhu, K. Zhang, J. Liang, J. Cao, B. Wen, R. Timofte, and L. Van Gool, "Denoising diffusion models for plug-and-play image restoration," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, pp. 1219–1229, 2023.

[91] B. Kawar, M. Elad, S. Ermon, and J. Song, "Denoising diffusion restoration models," in *Proc. Neural Info. Process. Syst. Conf.*, 2022.

[92] H. Chung, S. Lee, and J. C. Ye, "Decomposed diffusion sampler for accelerating large-scale inverse problems," in *Proc. Intl. Conf. Learn. Rep.*, 2024.

[93] J. Song, A. Vahdat, M. Mardani, and J. Kautz, "Pseudoinverse-guided diffusion models for inverse problems," in *Proc. Intl. Conf. Learn. Rep.*, 2023.

[94] B. Efron, "Tweedie's formula and selection bias," *J. Am. Statist. Assoc.*, vol. 106, no. 496, pp. 1602–1614, 2011.

[95] M. Mardani, J. Song, J. Kautz, and A. Vahdat, "A variational perspective on solving inverse problems with diffusion models," in *Proc. Intl. Conf. Learn. Rep.*, 2024.

[96] F. Coeurdoux, N. Dobigeon, and P. Chainais, "Plug-and-play split Gibbs sampler: Embedding deep generative priors in Bayesian inference," *IEEE Trans. Image Process.*, vol. 33, pp. 3496–3507, 2024.

[97] B. Zhang, W. Chu, J. Berner, C. Meng, A. Anandkumar, and Y. Song, "Improving diffusion inverse problem solving with decoupled noise annealing," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, 2025.

[98] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. New York, NY, USA: Birkhäuser, 2013.

[99] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magn. Reson. Med.*, vol. 58, no. 6, pp. 1182–1195, 2007.

[100] C. Millard, A. T. Hess, B. Mailhé, and J. Tanner, "Approximate message passing with a colored aliasing model for variable density Fourier sampled images," *IEEE Open J. Signal Process.*, vol. 1, pp. 146–158, 2020.

[101] C. A. Metzler and G. Wetzstein, "D-VDAMP: Denoising-based approximate message passing for compressive MRI," in *Proc. IEEE Intl. Conf. Acoust. Speech Signal Process.*, pp. 1410–1414, 2021.

[102] C. Millard, M. Chiew, J. Tanner, A. T. Hess, and B. Mailhe, "Tuning-free multi-coil compressed sensing MRI with parallel variable density approximate message passing (P-VDAMP)," *arXiv:2203.04180*, 2022.

[103] C. Millard, A. Hess, J. Tanner, and B. Mailhe, "Deep plug-and-play multi-coil compressed sensing MRI with matched aliasing: The denoising-P-VDAMP algorithm," in *Proc. Annu. Meeting ISMRM*, pp. 1–9, 2022. See also arXiv:2203.04180.

[104] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[105] H. V. Poor, *An Introduction to Signal Detection and Estimation*. New York, NY, USA: Springer, 2nd ed., 1994.

[106] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, pp. 2080–2095, Aug. 2007.

[107] T. Meinhardt, M. Möller, C. Hazirbas, and D. Cremers, "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems," in *Proc. IEEE Intl. Conf. Comput. Vis.*, pp. 1781–1790, 2017.

[108] S. Ono, "Primal-dual plug-and-play image restoration," *IEEE Signal Process. Lett.*, vol. 24, pp. 1108–1112, Aug. 2017.

[109] U. Kamilov, H. Mansour, and B. Wohlberg, "A plug-and-play priors approach for solving nonlinear imaging inverse problems," *IEEE Signal Process. Lett.*, vol. 24, pp. 1872–1876, May 2017.

[110] S. K. Shastri, R. Ahmad, and P. Schniter, "Autotuning plug-and-play algorithms for MRI," in *Proc. Asilomar Conf. Signals Syst. Comput.*, pp. 1400–1404, 2020.

[111] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Intl. Symp. Info. Theory*, pp. 2168–2172, Aug. 2011. (full version at *arXiv:1010.5141*).

[112] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. Motivation and construction," in *Proc. Info. Theory Workshop*, (Cairo, Egypt), pp. 1–5, Jan. 2010.

[113] H. V. Poor and G. W. Wornell, eds., *Wireless Communications: Signal Processing Perspectives*. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.

[114] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Info. Theory*, vol. 57, pp. 764–785, Feb. 2011.

[115] R. Berthier, A. Montanari, and P.-M. Nguyen, "State evolution for approximate message passing with non-separable functions," *Info. Inference*, vol. 9, no. 1, pp. 33–79, 2019.

[116] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "BM3D-AMP: A new image recovery algorithm based on BM3D denoising," in *Proc. IEEE Intl. Conf. Image Process.*, pp. 3116–3120, 2015.

[117] S. Ramani, T. Blu, and M. Unser, "Monte-Carlo SURE: A black-box optimization of regularization parameters for general denoising algorithms," *IEEE Trans. Image Process.*, vol. 17, pp. 1540–1554, Sept. 2008.

[118] M. Opper and O. Winther, "Expectation consistent free energies for approximate inference," in *Proc. Neural Info. Process. Syst. Conf.*, pp. 1001–1008, 2005.

[119] T. Minka, *A Family of Approximate Algorithms for Bayesian Inference*. PhD thesis, Dept. Comp. Sci. Eng., MIT, Cambridge, MA, USA, 2001.

[120] S. Rangan, P. Schniter, and A. K. Fletcher, "Vector approximate message passing," in *Proc. IEEE Intl. Symp. Info. Theory*, pp. 1588–1592, 2017.

[121] A. K. Fletcher, P. Pandit, S. Rangan, S. Sarkar, and P. Schniter, "Plug-in estimation in high-dimensional linear inverse problems: A rigorous analysis," in *Proc. Neural Info. Process. Syst. Conf.*, pp. 7440–7449, 2018.

[122] K. Takeuchi, "Rigorous dynamics of expectation-propagation-based signal recovery from unitarily invariant measurements," in *Proc. IEEE Intl. Symp. Info. Theory*, pp. 501–505, 2017.

[123] B. He, H. Liu, Z. Wang, and X. Yuan, "A strictly contractive Peaceman-Rachford splitting method for convex programming," *SIAM J. Optim.*, vol. 24, no. 3, pp. 1011–1040, 2014.

[124] B. He, F. Ma, and X. Yuan, "Convergence study on the symmetric version of ADMM with larger step sizes," *SIAM J. Imag. Sci.*, vol. 9, no. 3, pp. 1467–1501, 2016.

[125] P. Schniter, S. Rangan, and A. K. Fletcher, "Denoising-based vector approximate message passing," in *Proc. Intl. Biomed. Astronom. Signal Process. (BASP) Front. Workshop*, p. 77, 2017.

[126] E. M. Eksioglu and A. K. Tanc, "Denoising AMP for MRI reconstruction: BM3D-AMP-MRI," *SIAM J. Imag. Sci.*, vol. 11, no. 3, pp. 2090–2109, 2018.

[127] S. Sarkar, R. Ahmad, and P. Schniter, "MRI image recovery using damped denoising vector AMP," in *Proc. IEEE Intl. Conf. Acoust. Speech Signal Process.*, pp. 8108–8112, 2021.

[128] J. G. Pipe and P. Menon, "Sampling density compensation in MRI: Rationale and an iterative numerical solution," *Magn. Reson. Med.*, vol. 41, no. 1, pp. 179–186, 1999.

[129] V. Edupuganti, M. Mardani, S. Vasanawala, and J. Pauly, "Uncertainty quantification in deep MRI reconstruction," *IEEE Trans. Med. Imag.*, vol. 40, pp. 239–250, Jan. 2021.

[130] C. Millard, *Approximate message passing for compressed sensing magnetic resonance imaging*. PhD thesis, Dept. Comput. Sci., Oxford Univ., Oxford, U.K., 2021.

[131] B. Adcock, A. C. Hansen, C. Poon, and B. Roman, "Breaking the coherence barrier: A new theory for compressed sensing," *Forum Math. Sigma*, vol. 5, Feb. 2017. doi:10.1017/fms.2016.32.

[132] P. Schniter, S. Rangan, and A. K. Fletcher, "Plug-and-play image recovery using vector AMP." presented at the Intl. Biomedical and Astronomical Signal Processing (BASP) Frontiers Workshop, Villars-sur-Ollon, Switzerland, Jan. 2017.

[133] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD, USA: John Hopkins Univ. Press, 3rd ed., 1996.

[134] A. Ahmadzadegan, P. Simidzija, M. Li, and A. Kempf, "Neural networks can learn to utilize correlated auxiliary noise," *Sci. Rep.*, vol. 11, pp. 1–8, July 2021.

[135] Y. Chang, L. Yan, M. Chen, H. Fang, and S. Zhong, "Two-stage convolutional neural network for medical noise removal via image decomposition," *IEEE Trans. Instrum. Meas.*, vol. 69, pp. 2707–2721, June 2020.

[136] J. Tiirola, "A learning based approach to additive, correlated noise removal," *J. Vis. Commun. Image Rep.*, vol. 62, pp. 286–294, July 2019.

[137] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNN-based image denoising," *IEEE Trans. Image Process.*, vol. 27, pp. 4608–4622, Sept. 2018.

[138] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Intl. Conf. Med. Image Comput. Comput. Assist. Intervent.*, pp. 234–241, 2015.

[139] X. Zhang, Y. Lu, J. Liu, and B. Dong, "Dynamically unfolding recurrent restorer: A moving endpoint control method for image restoration," in *Proc. Intl. Conf. Learn. Rep.*, 2019.

[140] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Trans. Comput. Imag.*, vol. 3, pp. 47–57, Mar. 2017.

[141] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, pp. 600–612, Apr. 2004.

[142] S. Mohan, Z. Kadkhodaie, E. P. Simoncelli, and C. Fernandez-Granda, "Robust and interpretable blind image denoising via bias-free convolutional neural networks," in *Proc. Intl. Conf. Learn. Rep.*, 2020.

[143] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig, "ESPIRiT–an eigenvalue approach to autocalibrating parallel MRI: Where SENSE meets GRAPPA," *Magn. Reson. Med.*, vol. 71, no. 3, pp. 990–1001, 2014.

[144] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probability and Statistics for Engineers and Scientists.* New York, NY, USA: Macmillan, 9th ed., 2016.

[145] R. M. Willett, R. F. Marcia, and J. M. Nichols, "Compressed sensing for practical optical imaging systems: A tutorial," *Optical Eng.*, vol. 50, July 2011.

[146] N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice-Hall, 1984.

[147] V. Katkovnik and J. Astola, "Phase retrieval via spatial light modulator phase modulation in 4f optical setup: Numerical inverse imaging with sparse regularization for phase and amplitude," *J. Opt. Soc. Amer. A*, vol. 29, no. 1, pp. 105–116, 2012.

[148] L.-H. Yeh, J. Dong, J. Zhong, L. Tian, M. Chen, G. Tang, M. Soltanolkotabi, and L. Waller, "Experimental robustness of Fourier ptychography phase retrieval algorithms," *Opt. Express*, vol. 23, no. 26, pp. 33214–33240, 2015.

[149] S. Boyd and L. Vandenberghe, *Convex Optimization*. Providence, RI: Cambridge, 2004.

[150] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering* (H. Bauschke, R. Burachik, P. Combettes, V. Elser, D. Luke, and H. Wolkowicz, eds.), pp. 185–212, Springer, 2011.

[151] P. Pandit, M. Sahraee-Ardakan, S. Rangan, P. Schniter, and A. K. Fletcher, "Inference with deep generative priors in high dimensions," *IEEE J. Sel. Areas Info. Theory*, vol. 1, no. 1, pp. 336–347, 2020.

[152] K. Takeuchi, "Rigorous dynamics of expectation-propagation-based signal recovery from unitarily invariant measurements," *IEEE Trans. Info. Theory*, vol. 66, pp. 368–386, Jan. 2020.

[153] S. K. Shastri, R. Ahmad, C. A. Metzler, and P. Schniter, "Denoising generalized expectation-consistent approximation for mr image recovery," *IEEE J. Sel. Areas Info. Theory*, vol. 3, no. 3, pp. 528–542, 2022.

[154] S. Sarkar, S. Rangan, A. K. Fletcher, and P. Schniter, "Bilinear recovery using adaptive vector-AMP," *IEEE Trans. Signal Process.*, vol. 67, no. 13, pp. 3383–3396, 2019.

[155] J. H. Ferziger, M. Perić, and R. L. Street, *Computational Methods for Fluid Dynamics*. Springer, 2019.

[156] E. J. Candès, X. Li, and M. Soltanolkotabi, "Phase retrieval from coded diffraction patterns," *Appl. Comput. Harmonic Anal.*, vol. 39, no. 2, pp. 277–299, 2015.

[157] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2007.

[158] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, pp. 4396–4405, 2019.

[159] H. Chung, J. Kim, M. T. McCann, M. L. Klasky, and J. C. Ye, "diffusion-posterior-sampling." `https://github.com/DPS2022/diffusion-posterior-sampling`, Mar. 2023.

[160] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Intl. Conf. Comput. Vis.*, vol. 2, pp. 416–423, July 2001.

[161] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Proc. Neural Info. Process. Syst. Conf.*, vol. 34, pp. 8780–8794, 2021.

[162] T. Bendory, R. Beinert, and Y. C. Eldar, "Fourier phase retrieval: Uniqueness and algorithms," in *MATHEON Conf. Compressed Sensing and its Applications*, pp. 55–91, 2015.

[163] A. Fannjiang, "Absolute uniqueness of phase retrieval with random illumination," *Inverse Problems*, vol. 28, no. 7, p. 075008, 2012.

[164] C. A. Metzler, "prdeep." `https://github.com/ricedsp/prDeep/tree/master`, 2018.

[165] D. Hekstra, I. Hunt-Isaak, J. Greisman, and J. Russell, "phase-retrieval." `https://github.com/Hekstra-Lab/phase-retrieval`, 2018.

[166] A. K. Fletcher, M. Sahraee-Ardakan, S. Rangan, and P. Schniter, "Rigorous dynamics and consistent estimation in arbitrarily conditioned linear systems," in *Proc. Neural Info. Process. Syst. Conf.*, pp. 2542–2551, 2017.

[167] B. D. O. Anderson, "Reverse-time diffusion equation models," *Stochastic Processes and their Applications*, vol. 12, no. 3, pp. 313–326, 1982.

[168] A. Hyvärinen, "Estimation of non-normalized statistical models by score matching," *J. Mach. Learn. Res.*, vol. 6, pp. 695–709, 2005.

[169] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, 2011.

[170] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proc. Intl. Conf. Mach. Learn.*, pp. 681–688, 2011.

[171] M. Bendel, S. K. Shastri, R. Ahmad, and P. Schniter, "Solving inverse problems using diffusion with fast iterative renoising," *arXiv:2501.17468*, 2025.

[172] X. Meng, S. Wu, and J. Zhu, "A unified Bayesian inference framework for generalized linear models," *IEEE Signal Process. Lett.*, vol. 25, no. 3, pp. 398–402, 2018.

[173] L. Borodenko, "motionblur." Downloaded from `https://github.com/LeviBorodenko/motionblur`, 2020.

[174] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, pp. 586–595, 2018.

[175] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. Neural Info. Process. Syst. Conf.*, vol. 30, 2017.

[176] B. Collins and S. Matsumoto, "On some properties of orthogonal Weingarten functions," *J. Math. Phys.*, vol. 50, no. 11, p. 113516, 2009.

[177] M. Buehrer, K. P. Pruessmann, P. Boesiger, and S. Kozerke, "Array compression for MRI with large coil arrays," *Magn. Reson. Med.*, vol. 57, no. 6, pp. 1131–1139, 2007.

[178] T. Zhang, J. M. Pauly, S. S. Vasanawala, and M. Lustig, "Coil compression for accelerated imaging with Cartesian sampling," *Magn. Reson. Med.*, vol. 69, no. 2, pp. 571–582, 2013.

[179] B. Kawar, M. Elad, S. Ermon, and J. Song, "Denoising diffusion restoration models." Downloaded from `https://github.com/bahjat-kawar/ddrm`, May 2022.

[180] Y. Zhu, K. Zhang, J. Liang, J. Cao, B. Wen, R. Timofte, and L. V. Gool, "Diffpir." Downloaded from `https://github.com/yuanzhi-zhu/DiffPIR`, July 2024.

[181] NVlabs, "RED-diff." Downloaded from `https://github.com/NVlabs/RED-diff`, 2023.

[182] B. Zhang, W. Chu, J. Berner, C. Meng, A. Anandkumar, and Y. Song, "Improving diffusion inverse problem solving with decoupled noise annealing." Downloaded from `https://github.com/zhangbingliang2019/DAPS`, 2024.

# Appendix A: EC/VAMP error recursion

In this appendix, we establish the error iteration

$$\boldsymbol{e}_2 = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{\mathsf{H}}\boldsymbol{e}_1 + \boldsymbol{u}. \tag{A.1}$$

To begin, we write the estimation function $\boldsymbol{f}_1$ as

$$\boldsymbol{f}_1(\boldsymbol{r}_1;\gamma_1) = \left(\gamma_w \boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} + \gamma_1 \boldsymbol{I}\right)^{-1}\left(\gamma_w \boldsymbol{A}^{\mathsf{H}}\boldsymbol{y} + \gamma_1 \boldsymbol{r}_1\right) \tag{A.2}$$

$$= \boldsymbol{r}_1 + \left(\gamma_w \boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} + \gamma_1 \boldsymbol{I}_N\right)^{-1}\left(\gamma_w \boldsymbol{A}^{\mathsf{H}}\boldsymbol{y} - \gamma_w \boldsymbol{A}^{\mathsf{H}}\boldsymbol{A}\boldsymbol{r}_1\right) \tag{A.3}$$

$$= \boldsymbol{r}_1 + \gamma_w \left(\boldsymbol{C} + \gamma_1 \boldsymbol{I}_N\right)^{-1} \boldsymbol{A}^{\mathsf{H}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{r}_1) \tag{A.4}$$

for

$$\boldsymbol{C} \triangleq \gamma_w \boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathsf{H}}. \tag{A.5}$$

The right side of (A.5) is an eigendecomposition where $\boldsymbol{V}\boldsymbol{V}^{\mathsf{H}} = \boldsymbol{V}^{\mathsf{H}}\boldsymbol{V} = \boldsymbol{I}$ and $\boldsymbol{\Lambda} = \mathrm{Diag}([\lambda_1, \ldots, \lambda_N])$ is real-valued. Note also that $\boldsymbol{V}$ is the right singular vector

matrix of $\boldsymbol{A}$. Using this eigendecomposition, we can write

$$\operatorname{tr}(\nabla \boldsymbol{f}_1(\boldsymbol{r}_1;\gamma_1)) = \operatorname{tr}(\boldsymbol{I} - (\boldsymbol{C} + \gamma_1 \boldsymbol{I}_N)^{-1}\boldsymbol{C}) \tag{A.6}$$

$$= \operatorname{tr}(\boldsymbol{I} - (\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathsf{H}} + \gamma_1 \boldsymbol{I}_N)^{-1}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathsf{H}}) \tag{A.7}$$

$$= \operatorname{tr}(\boldsymbol{I} - (\boldsymbol{\Lambda} + \gamma_1 \boldsymbol{I}_N)^{-1}\boldsymbol{\Lambda}) \tag{A.8}$$

$$= N - \sum_{n=1}^{N} \frac{\lambda_n}{\lambda_n + \gamma_1} \tag{A.9}$$

$$= N(1-\alpha) \quad \text{for} \quad \alpha \triangleq \frac{1}{N}\sum_{n=1}^{N} \frac{\lambda_n}{\lambda_n + \gamma_1}. \tag{A.10}$$

Thus, lines 4-5 of Alg. 1 can be written as

$$\widehat{\boldsymbol{x}}_1 = \boldsymbol{r}_1 + \gamma_w \left(\boldsymbol{C} + \gamma_1 \boldsymbol{I}_N\right)^{-1} \boldsymbol{A}^{\mathsf{H}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{r}_1) \tag{A.11}$$

$$\eta_1 = \frac{\gamma_1 N}{\operatorname{tr}(\nabla \boldsymbol{f}_1(\boldsymbol{r}_1;\gamma_1))} = \frac{\gamma_1}{1-\alpha} \tag{A.12}$$

and lines 7-8 as

$$\gamma_2 = \eta_1 - \gamma_1 = \gamma_1 \left(\frac{1}{1-\alpha} - 1\right) = \gamma_1 \frac{\alpha}{1-\alpha} \tag{A.13}$$

$$\boldsymbol{r}_2 = \frac{\eta_1 \widehat{\boldsymbol{x}}_1 - \gamma_1 \boldsymbol{r}_1}{\gamma_2} = \frac{1}{\alpha}\widehat{\boldsymbol{x}}_1 - \frac{1-\alpha}{\alpha}\boldsymbol{r}_1. \tag{A.14}$$

Plugging (A.11) into (A.14), we get

$$\boldsymbol{r}_2 = \boldsymbol{r}_1 + \frac{\gamma_w}{\alpha}\left(\boldsymbol{C} + \gamma_1 \boldsymbol{I}_N\right)^{-1} \boldsymbol{A}^{\mathsf{H}}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{r}_1). \tag{A.15}$$

Next, we express (A.15) in terms of the error vectors $\boldsymbol{e}_i \triangleq \boldsymbol{r}_i - \boldsymbol{x}_{\mathsf{true}}$ for $i = 1, 2$. Subtracting $\boldsymbol{x}_{\mathsf{true}}$ from both sides of (A.15) and applying $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_{\mathsf{true}} + \boldsymbol{w}$ and the

definition of $\boldsymbol{C}$ from (A.5), we get

$$\begin{aligned}
\boldsymbol{e}_2 &= \boldsymbol{e}_1 + \frac{\gamma_w}{\alpha}\left(\boldsymbol{C} + \gamma_1 \boldsymbol{I}_N\right)^{-1} \boldsymbol{A}^{\mathsf{H}}(\boldsymbol{A}\boldsymbol{x}_{\mathsf{true}} + \boldsymbol{w} - \boldsymbol{A}\boldsymbol{r}_1) \\
&= \boldsymbol{e}_1 - \frac{1}{\alpha}\left(\boldsymbol{C} + \gamma_1 \boldsymbol{I}_N\right)^{-1} \boldsymbol{C}\boldsymbol{e}_1 + \boldsymbol{u} \qquad \text{(A.16)} \\
&= \boldsymbol{e}_1 - \frac{1}{\alpha}\boldsymbol{V}\left(\boldsymbol{\Lambda} + \gamma_1 \boldsymbol{I}_N\right)^{-1} \boldsymbol{\Lambda}\boldsymbol{V}^{\mathsf{H}}\boldsymbol{e}_1 + \boldsymbol{u} \qquad \text{(A.17)} \\
&= \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{\mathsf{H}}\boldsymbol{e}_1 + \boldsymbol{u}, \qquad \text{(A.18)}
\end{aligned}$$

where

$$\boldsymbol{u} \triangleq \frac{\gamma_w}{\alpha}\big(\boldsymbol{C} + \gamma_1 \boldsymbol{I}_N\big)^{-1} \boldsymbol{A}^{\mathsf{H}}\boldsymbol{w} \qquad \text{(A.19)}$$

$$\boldsymbol{D} \triangleq \boldsymbol{I}_N - \frac{1}{\alpha}\left(\boldsymbol{\Lambda} + \gamma_1 \boldsymbol{I}_N\right)^{-1} \boldsymbol{\Lambda} \qquad \text{(A.20)}$$

Notice that $\mathrm{tr}(\boldsymbol{D}) = 0$ due to the definition of $\alpha$ in (A.10).

# Appendix B: EC/VAMP error analysis

We start with the fact [176] that, for any $N \geq 2$, the elements $v_{nj}$ of uniformly distributed orthogonal $\boldsymbol{V} \in \mathbb{R}^{N\times N}$ obey

$$\mathbb{E}(v_{nj}) = 0 \tag{B.1a}$$

$$\mathbb{E}(v_{nj}v_{mk}) = \tfrac{1}{N}\delta_{n-m}\delta_{j-k} \tag{B.1b}$$

$$\mathbb{E}(v_{nj}^2 v_{mk}^2) = \begin{cases} \frac{3}{N(N+2)} & n = m \;\&\; j = k \\ \frac{1}{N(N+2)} & n = m \;\&\; j \neq k \\ \frac{1}{N(N+2)} & n \neq m \;\&\; j = k \\ \frac{N+1}{N(N+2)(N-1)} & n \neq m \;\&\; j \neq k \end{cases}, \tag{B.1c}$$

where $\delta_n$ is the Kronecker delta (i.e., $\delta_0 = 1$ and $\delta_n\big|_{n\neq 0} = 0$). Equations (B.1) will be used to establish the following lemma.

**Lemma B.0.1.** *Suppose that* $\boldsymbol{f} = \boldsymbol{V}\,\mathrm{Diag}(\boldsymbol{d})\boldsymbol{V}^\top\boldsymbol{e} \in \mathbb{R}^N$ *where* $\boldsymbol{d}$ *is deterministic with elements obeying* $\sum_{j=1}^N d_j = 0$ *and* $\mathcal{D} \triangleq \lim_{N\to\infty}\frac{1}{N}\sum_{j=1}^N d_j^2 < \infty$*;* $\boldsymbol{e}$ *is random with elements of finite mean and variance obeying* $\varepsilon \triangleq \lim_{N\to\infty}\frac{1}{N}\sum_{j=1}^N e_j^2 < \infty$*; and* $\boldsymbol{V}$ *is uniformly distributed over the set of orthogonal matrices and independent of* $\boldsymbol{e}$ *up to the fourth moment, i.e.,* $\mathbb{E}(v_{nj}v_{mk}v_{n'j'}v_{m'k'}|\boldsymbol{e}) = \mathbb{E}(v_{nj}v_{mk}v_{n'j'}v_{m'k'})$*. Then, as* $N \to \infty$*,*

$$\mathbb{E}(\boldsymbol{f}|\boldsymbol{e}) = \boldsymbol{0} \tag{B.2}$$

$$\mathrm{Cov}(\boldsymbol{f}|\boldsymbol{e}) = \varepsilon\mathcal{D}\boldsymbol{I}_N. \tag{B.3}$$

*Proof.* Writing the $n$th element of $\boldsymbol{f}$ as

$$f_n = \sum_{j=1}^{N} v_{nj} d_j \sum_{k=1}^{N} v_{kj} e_k \tag{B.4}$$

we can establish (B.2) via

$$\mathbb{E}(f_n|\boldsymbol{e}) = \sum_{j=1}^{N}\sum_{k=1}^{N} d_j e_k \, \mathbb{E}(v_{nj} v_{kj}|\boldsymbol{e}) \tag{B.5}$$

$$\stackrel{(a)}{=} \sum_{j=1}^{N}\sum_{k=1}^{N} d_j e_k \delta_{n-k} \frac{1}{N} \tag{B.6}$$

$$= e_n \frac{1}{N} \sum_{j=1}^{N} d_j \stackrel{(b)}{=} 0 \ \forall n, \tag{B.7}$$

where (a) used (B.1b) and the assumed independence of $\boldsymbol{V}$ and $\boldsymbol{e}$ and (b) used $\sum_j d_j = 0$.

To establish (B.3), we begin by using (B.4) and the assumed independence of $\boldsymbol{V}$ and $\boldsymbol{e}$ to write

$$\mathbb{E}(f_n^2|\boldsymbol{e}) = \sum_{j}\sum_{k}\sum_{j'}\sum_{k'} d_j d_{j'} e_k e_{k'} \, \mathbb{E}(v_{nj} v_{kj} v_{nj'} v_{k'j'}). \tag{B.8}$$

When $k = n$, the expectation will vanish unless $k' = n$, and when $k \neq n$, the expectation will vanish unless $k' = k$ and $j' = j$. Thus we have

$$\begin{aligned}
&\mathbb{E}(f_n^2|\boldsymbol{e}) \\
&= e_n^2 \sum_j \sum_{j'} d_j d_{j'} \, \mathbb{E}(v_{nj}^2 v_{nj'}^2) + \sum_{k \neq n} \sum_j d_j^2 k^2 \, \mathbb{E}(v_{nj}^2 v_{kj}^2) && \text{(B.9)} \\
&= e_n^2 \sum_j d_j^2 \, \mathbb{E}(v_{nj}^4) + e_n^2 \sum_j \sum_{j' \neq j} d_j d_{j'} \, \mathbb{E}(v_{nj}^2 v_{nj'}^2) \\
&\quad + \sum_{k \neq n} \sum_j d_j^2 e_k^2 \, \mathbb{E}(v_{nj}^2 v_{kj}^2) && \text{(B.10)} \\
&\overset{(a)}{=} \frac{3e_n^2}{N(N+2)} \sum_j d_j^2 + \frac{e_n^2}{N(N+2)} \sum_j d_j \sum_{j' \neq j} d_{j'} \\
&\quad + \frac{1}{N(N+2)} \sum_j d_j^2 \sum_{k \neq n} e_k^2 && \text{(B.11)} \\
&\overset{(b)}{=} \frac{e_n^2}{N+2} \big( \frac{1}{N} \sum_j d_j^2 \big) + \frac{N}{N+2} \big( \frac{1}{N} \sum_j d_j^2 \big) \big( \frac{1}{N} \sum_k e_k^2 \big) && \text{(B.12)} \\
&\overset{N \to \infty}{=} \mathcal{D}\varepsilon, && \text{(B.13)}
\end{aligned}$$

where (a) used (B.1c) and where (b) used $\sum_{j' \neq j} d_{j'} = (\sum_{j'} d_{j'}) - d_j = -d_j$ and $\sum_{k \neq n} e_k^2 = \|\boldsymbol{e}\|^2 - e_n^2$. The limit as $N \to \infty$ follows from the definitions of $\mathcal{D}$ and $\varepsilon$, and the fact that $\lim_{N \to \infty} e_n^2 / N = 0$ due to the finite mean and variance of $e_n$. Thus we have established the diagonal terms in (B.3).

The off-diagonal terms in (B.3) follow from analyzing

$$\begin{aligned}
&\mathbb{E}(f_n f_m|\boldsymbol{e})\big|_{n \neq m} \\
&= \sum_j \sum_k \sum_{j'} \sum_{k'} d_j d_{j'} e_k e_{k'} \, \mathbb{E}(v_{nj} v_{kj} v_{mj'} v_{k'j'}). && \text{(B.14)}
\end{aligned}$$

In this case, the expectation will vanish unless $k = n$ or $k = m$. When $k = n$, we also need $k' = m$, and when $k = m$, we also need $k' = n$ and $j = j'$. Thus we can write

$$\begin{aligned}
&\mathbb{E}(f_n f_m|\boldsymbol{e})\big|_{n\neq m} \\
&= e_n e_m \sum_j \sum_{j'} d_j d_{j'} \,\mathbb{E}(v_{nj}^2 v_{mj'}^2) + e_n e_m \sum_j d_j^2 \,\mathbb{E}(v_{nj}^2 v_{mj}^2) && \text{(B.15)} \\
&= 2e_n e_m \sum_j d_j^2 \,\mathbb{E}(v_{nj}^2 v_{mj}^2) + e_n e_m \sum_j \sum_{j'\neq j} d_j d_{j'} \,\mathbb{E}(v_{nj}^2 v_{mj'}^2) && \text{(B.16)} \\
&\overset{(a)}{=} \frac{2e_n e_m}{N(N+2)} \sum_j d_j^2 + \frac{e_n e_m (N+1)}{N(N+2)(N-1)} \sum_j d_j \sum_{j'\neq j} d_{j'} && \text{(B.17)} \\
&\overset{(b)}{=} \frac{N-3}{(N+2)(N-1)} e_n e_m \frac{1}{N} \sum_j d_j^2 && \text{(B.18)} \\
&\overset{(c)}{=} O(1/N) \overset{N\to\infty}{=} 0, && \text{(B.19)}
\end{aligned}$$

where (a) used (B.1c), (b) used $\sum_{j'\neq j} d_{j'} = (\sum_{j'} d_{j'}) - d_j = -d_j$, and (c) used $\frac{1}{N}\sum_j d_j^2 = O(1)$ from the definition of $\mathcal{D}$ and $e_n e_m = O(1)$ from the finite mean and variance of $e_n$. This establishes the off-diagonal terms in (B.3). □

Lemma B.0.1 will now be used to establish

$$\mathbb{E}(\boldsymbol{e}_2|\boldsymbol{e}_1) \overset{N\to\infty}{=} \mathbf{0} \tag{B.20}$$

$$\mathrm{Cov}(\boldsymbol{e}_2|\boldsymbol{e}_1) \overset{N\to\infty}{=} \varepsilon_2 \boldsymbol{I} \tag{B.21}$$

for some $\varepsilon_2 > 0$. To simplify the derivation, we first write (A.1) as

$$\boldsymbol{e}_2 = \boldsymbol{f} + \boldsymbol{u} \quad \text{for} \quad \boldsymbol{f} \triangleq \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top \boldsymbol{e}_1, \tag{B.22}$$

and recall that $\mathrm{tr}(\boldsymbol{D}) = 0$. For the mean of $\boldsymbol{e}_2|\boldsymbol{e}_1$, we immediately have that

$$\mathbb{E}(\boldsymbol{e}_2|\boldsymbol{e}_1) = \mathbb{E}(\boldsymbol{f}|\boldsymbol{e}_1) + \mathbb{E}(\boldsymbol{u}|\boldsymbol{e}_1) = \mathbf{0} \tag{B.23}$$

since $\mathbb{E}(\boldsymbol{f}|\boldsymbol{e}_1) = \mathbf{0}$ due to (B.2). Also, $\mathbb{E}(\boldsymbol{u}|\boldsymbol{e}_1) = \mathbf{0}$ from definition (A.19) and $\mathbb{E}(\boldsymbol{w}|\boldsymbol{e}_2) = \mathbf{0}$. This establishes (B.20).

To characterize the covariance of $\boldsymbol{e}_2|\boldsymbol{e}_1$, we write

$$\begin{aligned}\mathrm{Cov}(\boldsymbol{e}_2|\boldsymbol{e}_1) = \mathrm{Cov}(\boldsymbol{f}) &+ \mathbb{E}\left[\boldsymbol{f}\boldsymbol{u}^\top\middle|\boldsymbol{e}_1\right] \\ &+ \mathbb{E}\left[\boldsymbol{u}\boldsymbol{f}^\top\middle|\boldsymbol{e}_1\right] + \mathrm{Cov}(\boldsymbol{u}|\boldsymbol{e}_1) \end{aligned} \tag{B.24}$$

and investigate each term separately. For the first term in (B.24), equation (B.3) and definition (B.22) imply that

$$\mathrm{Cov}(\boldsymbol{f}|\boldsymbol{e}_1) \overset{N\to\infty}{=} \frac{\varepsilon_1}{N}\,\mathrm{tr}\left[\boldsymbol{D}^2\right]\boldsymbol{I}_N, \tag{B.25}$$

for $\varepsilon_1 \triangleq \lim_{n\to\infty}\frac{1}{N}\sum_{n=1}^{N} e_{1n}^2$. For the second and third terms in (B.24), equation (B.2) and definition (B.22) imply

$$\mathbb{E}\left[\boldsymbol{f}\boldsymbol{u}^\top\middle|\boldsymbol{e}_1,\boldsymbol{u}\right] \overset{N\to\infty}{=} \boldsymbol{0}. \tag{B.26}$$

For the last term in (B.24), we can use (A.5) and $\mathrm{Cov}(\boldsymbol{w}|\boldsymbol{e}_1) = \boldsymbol{I}_M/\gamma_w$ to obtain

$$\begin{aligned}&\mathrm{Cov}(\boldsymbol{u}|\boldsymbol{V},\boldsymbol{e}_1) \\ &= \frac{1}{\alpha}\big(\boldsymbol{C}+\gamma_1\boldsymbol{I}_N\big)^{-1}\boldsymbol{C}\big(\boldsymbol{C}+\gamma_1\boldsymbol{I}_N\big)^{-1}\frac{1}{\alpha} &\text{(B.27)}\\ &= \frac{1}{\alpha}\boldsymbol{V}\big(\boldsymbol{\Lambda}+\gamma_1\boldsymbol{I}_N\big)^{-1}\boldsymbol{\Lambda}\big(\boldsymbol{\Lambda}+\gamma_1\boldsymbol{I}_N\big)^{-1}\boldsymbol{V}^\top\frac{1}{\alpha} &\text{(B.28)}\\ &= \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Sigma}\boldsymbol{V}^\top &\text{(B.29)}\end{aligned}$$

for

$$\boldsymbol{\Sigma} \triangleq \frac{1}{\alpha}\big(\boldsymbol{\Lambda}+\gamma_1\boldsymbol{I}_N\big)^{-1}\boldsymbol{\Lambda} = \boldsymbol{I}_N - \boldsymbol{D}. \tag{B.30}$$

Then we take the expectation of (B.29) over $\boldsymbol{V}$ to obtain

$$\begin{aligned}&[\mathrm{Cov}(\boldsymbol{u}|\boldsymbol{e}_1)]_{n,m} \\ &= \sum_{j=1}^{N}\frac{(\sigma_j)^2}{\lambda_j}\,\mathbb{E}(v_{nj}v_{mj}|\boldsymbol{e}_1) \overset{(a)}{=} \delta_{n-m}\frac{1}{N}\sum_{j=1}^{N}\frac{(\sigma_j)^2}{\lambda_j},\end{aligned} \tag{B.31}$$

where $\sigma_j \triangleq [\boldsymbol{\Sigma}]_{jj}$ and where (a) follows from (B.1b) and the assumed independence of $\boldsymbol{V}$ and $\boldsymbol{e}_1$. Consequently,

$$\mathrm{Cov}(\boldsymbol{u}|\boldsymbol{e}_1) = \frac{1}{N}\,\mathrm{tr}\left[\boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Sigma}\right]\boldsymbol{I}_N. \tag{B.32}$$

Combining (B.24)–(B.32), we have

$$\mathrm{Cov}(\boldsymbol{e}_2|\boldsymbol{e}_1) = \varepsilon_2 \boldsymbol{I}_N \tag{B.33}$$

for

$$\varepsilon_2 \triangleq \frac{\varepsilon_1}{N}\,\mathrm{tr}\left[(\boldsymbol{I}_N - \boldsymbol{\Sigma})^2\right] + \frac{1}{N}\,\mathrm{tr}\left[\boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Sigma}\right]. \tag{B.34}$$

The expression for $\varepsilon_2$ can be simplified as follows.

$$\begin{aligned}
\varepsilon_2 &= \frac{(\varepsilon_1 - 1/\gamma_1)}{N}\,\mathrm{tr}\left[(\boldsymbol{I}_N - \boldsymbol{\Sigma})^2\right] \\
&\quad + \frac{1}{\gamma_1 N}\,\mathrm{tr}\left[(\boldsymbol{I}_N - \boldsymbol{\Sigma})^2 + \gamma_1 \boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Sigma}\right] &\text{(B.35)} \\
&= \frac{(\varepsilon_1 - 1/\gamma_1)}{N}\sum_{n=1}^{N}\left(1 - \frac{\lambda_n/\alpha}{\lambda_n + \gamma_1}\right)^2 \\
&\quad + \frac{1}{\gamma_1 N}\,\mathrm{tr}\left[\boldsymbol{I}_N - 2\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\big(\boldsymbol{I}_N + \gamma_1\boldsymbol{\Lambda}^{-1}\big)\boldsymbol{\Sigma}\right]. &\text{(B.36)}
\end{aligned}$$

Leveraging (B.30) to simplify the last term, we get

$$\begin{aligned}
\varepsilon_2 &= \frac{(\varepsilon_1 - 1/\gamma_1)}{N}\sum_{n=1}^{N}\left(\frac{\lambda_n(1-1/\alpha)+\gamma_1}{\lambda_n + \gamma_1}\right)^2 \\
&\quad + \frac{1}{\gamma_1 N}\,\mathrm{tr}\left[\boldsymbol{I}_N + (1/\alpha - 2)\boldsymbol{\Sigma}\right] &\text{(B.37)} \\
&\overset{(a)}{=} \frac{(\varepsilon_1 - 1/\gamma_1)}{N}\sum_{n=1}^{N}\left(\frac{\lambda_n(1-1/\alpha)+\gamma_1}{\lambda_n + \gamma_1}\right)^2 \\
&\quad + \frac{1}{\gamma_1}\left(\frac{1}{\alpha} - 1\right) &\text{(B.38)} \\
&\overset{(b)}{=} \frac{(\varepsilon_1 - 1/\gamma_1)}{N}\sum_{n=1}^{N}\left(\frac{1-\lambda_n/\gamma_2}{1+\lambda_n/\gamma_1}\right)^2 + \frac{1}{\gamma_2}, &\text{(B.39)}
\end{aligned}$$

where (a) used the fact that $\mathrm{tr}(\boldsymbol{\Sigma}) = N$ and (b) used (A.10).

Finally, notice that the elements of $\boldsymbol{e}_2$ come from a sum of the form

$$e_{2n} = u_n + \sum_{j=1}^{N} \xi_{nj} e_j \text{ for } \xi_{nj} = [\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top]_{nj}, \tag{B.40}$$

where, for any fixed $\boldsymbol{e}_1$, the elements $\{\xi_{nj}\}_{j=1}^{N}$ are zero mean, $O(1/N)$ variance, and uncorrelated. Because $u_n$ are Gaussian, it can be argued using the central limit theorem that the elements of $\boldsymbol{e}_2$ become Gaussian as $N \to \infty$. Combining this result with (B.20)–(B.21), we have that, given $\boldsymbol{e}_1$, as $N \to \infty$, the elements of $\boldsymbol{e}_2$ are marginally zero-mean Gaussian and uncorrelated.

# Appendix C: Experimental Setup for MR Image Recovery

## C.1 Multicoil MRI experiments

In this section we detail the experimental setup for the multicoil experiments in Sections 2.3.2, 2.3.3, and 2.3.4.

### C.1.1 Data

For our multicoil experiments, we used 3T knee and brain data from fastMRI [30]. For knee training data, we randomly picked 28 volumes and used the middle 8 slices from each volume, while for knee testing data we randomly picked 4 other volumes and used the middle 4 slices from each. Only non-fat-suppressed knee data was used. For brain training data, we randomly picked 28 volumes and used the bottom 8 slices from each volume, while for brain testing data we randomly picked 4 other brain volumes and used the bottom 4 slides from each. Only axial T2-weighted brain data was used. Starting with the raw fastMRI data, we first applied a standard PCA-based coil-compression technique [177, 178] to reduce the number of coils from $C = 15$ to $C = 8$. Then we Fourier-transformed each fully-sampled coil measurement to the pixel domain, center-cropped down to size $368 \times 368$ so that all images had the same size, and Fourier-transformed back to k-space, yielding fully sampled multicoil k-space measurement vectors $\boldsymbol{y}_{\mathsf{full}} \in \mathbb{C}^{NC}$ with $N = 368^2 = 135424$ entries.

### C.1.2 Ground-truth extraction

To extract the ground-truth image $\boldsymbol{x}_{\mathsf{true}}$ from $\boldsymbol{y}_{\mathsf{full}}$, we first estimated the coil sensitivity maps $\{\boldsymbol{s}_c\}_{c=1}^{C}$ from the central 24×24 region of k-space using ESPIRiT[4] [143]. We then modeled $\boldsymbol{y}_{\mathsf{full}} \approx \boldsymbol{A}_{\mathsf{full}}\boldsymbol{x}_{\mathsf{true}}$, where according to the definition of $\boldsymbol{A}$ we have

$$\boldsymbol{A}_{\mathsf{full}} \triangleq \begin{bmatrix} \boldsymbol{F}\,\mathrm{Diag}(\boldsymbol{s}_1) \\ \vdots \\ \boldsymbol{F}\,\mathrm{Diag}(\boldsymbol{s}_C) \end{bmatrix} = (\boldsymbol{I}_C \otimes \boldsymbol{F})\boldsymbol{S} \quad \text{for} \quad \boldsymbol{S} \triangleq \begin{bmatrix} \mathrm{Diag}(\boldsymbol{s}_1) \\ \vdots \\ \mathrm{Diag}(\boldsymbol{s}_C) \end{bmatrix}, \tag{C.1}$$

and we used least-squares to extract the ground-truth images as follows:

$$\begin{aligned} \boldsymbol{x}_{\mathsf{true}} &\triangleq (\boldsymbol{A}_{\mathsf{full}}^{\mathsf{H}}\boldsymbol{A}_{\mathsf{full}})^{+}\boldsymbol{A}_{\mathsf{full}}^{\mathsf{H}}\boldsymbol{y}_{\mathsf{full}} && \text{(C.2)} \\ &= (\boldsymbol{S}^{\mathsf{H}}\boldsymbol{S})^{+}\boldsymbol{S}^{\mathsf{H}}(\boldsymbol{I}_c \otimes \boldsymbol{F}^{\mathsf{H}})\boldsymbol{y}_{\mathsf{full}} && \text{(C.3)} \\ &\overset{(a)}{=} \boldsymbol{S}^{\mathsf{H}}(\boldsymbol{I}_c \otimes \boldsymbol{F}^{\mathsf{H}})\boldsymbol{y}_{\mathsf{full}} && \text{(C.4)} \\ &= \boldsymbol{A}_{\mathsf{full}}^{\mathsf{H}}\boldsymbol{y}_{\mathsf{full}}, && \text{(C.5)} \end{aligned}$$

where (a) holds because ESPIRiT guarantees that, for each index pixel index $n$, the coil maps are either all zero (i.e., $[\boldsymbol{s}_c]_n = 0\ \forall c$) or they have a sum-squared value of one (i.e., $\sum_{c=1}^{C} |[\boldsymbol{s}_c]_n|^2 = 1$).

### C.1.3 Noisy, subsampled, k-space measurements

To create the noisy subsampled k-space measurements, we started with the fully sampled fastMRI $\boldsymbol{y}_{\mathsf{full}}$ from above, applied a sampling mask $\boldsymbol{M}$ of acceleration rate $R$, and added circularly symmetric complex-valued WGN $\boldsymbol{w}$ to obtain $\boldsymbol{y}$. The sampling densities that generated the 2D point and 2D line masks were obtained from the `genPDF` function of the SparseMRI package[5] with the same settings used in the VDAMP code[6],

[4]We used the default ESPIRiT settings from:
https://sigpy.readthedocs.io/en/latest/generated/sigpy.mri.app.EspiritCalib.html.

[5]`http://people.eecs.berkeley.edu/~mlustig/Software.html`

[6]`https://github.com/charlesmillard/VDAMP`

except that the 2D line masks used a 1D sampling density while the 2D point masks used a 2D sampling density. The variance on the noise was adjusted to reach a desired signal-to-noise ratio (SNR), where SNR $\triangleq \|\boldsymbol{y} - \boldsymbol{w}\|^2/\|\boldsymbol{w}\|^2$. With multicoil data, we used masks with a fully sampled central $24 \times 24$ autocalibration region, as in Fig. 2.1(b)-(c) to facilitate the use of ESPIRiT for coil estimation.

### C.1.4 Algorithm details

For D-GEC, we used the 2D Haar wavelet transform of depth $D = 4$, giving $L = 13$ wavelet subbands. When evaluating $\boldsymbol{f}_1$, we use 150 CG iterations in Section 2.3.2 and 10 in Sections 2.3.3 and 2.3.4. Also, we use the damping scheme from [127] with a damping factor of 0.3 and run the D-GEC algorithm for 20 iterations. For the experiments in Section 2.3.2, we used the auto-tuning scheme from [166] to adjust $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$.

### C.1.5 Denoiser details

As described in Section 2.3.1, our corr+corr denoiser was built on bias-free DnCNN [142]. For the multicoil experiments, the images were complex-valued and so DnCNN used two input and output channels: one for the real part and one for the imaginary part. When extending DnCNN to corr+corr, we added a single noise channel, since we assumed that the real and imaginary parts of the noise had the same noise statistics. Prior to training, each ground-truth image was scaled so that the 98th percentile of its pixel magnitudes equaled 1. While training, we used standard deviations $\{1/\sqrt{\gamma_\ell}\}_{\ell=1}^{L}$ drawn independently from a uniform distribution over a specified interval $[\mathrm{SD}_{\min}, \mathrm{SD}_{\max}]$. Despite the use of a bias-free DNN, we found that it did not work well to train a single denoiser over a very wide range of SDs, and so we trained five different

denoisers, each over a different range of subband SDs: $[0, 10/255]$, $[10/255, 20/255]$, $[20/255, 50/255]$, $[50/255, 120/255]$, and $[120/255, 500/255]$. In each case, we used the training procedure described in Section 2.3.1, with $\ell_2$ loss, 20 epochs, a minibatch size of 128, the Adam optimizer, and a learning rate that started at $10^{-3}$ and was reduced by a factor of 2 at the 8th, 12th, 14th, 16th, 18th, and 19th epochs. The denoisers were trained using $64 \times 64$ image patches, of which we obtained 645 792 from the training images using a stride of $10 \times 10$ and standard data-augmentation techniques like rotation and flipping. Although we cannot guarantee that the test images will be scaled in the same way, this is not a problem because bias-free DnCNN obeys $\boldsymbol{f}_2(\alpha \boldsymbol{u}, \alpha \boldsymbol{N}) = \alpha \boldsymbol{f}_2(\boldsymbol{u}, \boldsymbol{N})$ for all $\alpha > 0$. It took approximately 24 hours to train each denoiser on a workstation with a single NVIDIA RTX-A6000 GPU.

## C.2 Single-coil MRI experiments

In this section we detail the experimental setup for the single-coil experiments used in Section 2.3.5.

### C.2.1 Data

For our single-coil experiments, we used MRI images from the Stanford 2D FSE dataset [31]. We used the same train/test/validation split from [101]: for testing, we used the 10 images shown in Fig. 2.3, for training we used 70 other images, and for validation we used 8 remaining images. All images were real-valued and $352 \times 352$. For each ground-truth image, the fully sampled k-space data was created via $\boldsymbol{y}_{\mathsf{full}} = \boldsymbol{F}\boldsymbol{x}_{\mathsf{true}}$ using 2D discrete Fourier transform $\boldsymbol{F}$.

### C.2.2 Noisy, subsampled, k-space measurements

To create the noisy subsampled k-space measurements, we started with the full sampled Stanford $\boldsymbol{y}_{\mathsf{full}}$ from above, applied a 2D point sampling mask $\boldsymbol{M}$ of acceleration rate $R$, and added circularly symmetric complex-valued WGN to obtain $\boldsymbol{y}$. The variance on the noise was adjusted to reach an SNR of 45 dB. With single-coil data, we do not need a fully sampled central autocalibration region and so we use masks similar to that shown in Fig. 2.1(a).

### C.2.3 Algorithm details

For D-GEC, we used the 2D Haar wavelet transform of depth $D = 4$, giving $L = 13$ wavelet subbands. When evaluating $\boldsymbol{f}_1$, we used 10 CG iterations. Also, we used the auto-tuning scheme from [166] to adjust $\boldsymbol{\gamma}_1$ and the damping scheme from [127] with a damping factor of 0.5. We ran the D-GEC algorithm for a maximum of 200 iterations.

### C.2.4 Denoiser details

As described in Section 2.3.1, our corr+corr denoiser was built on bias-free DnCNN [142]. For the single-coil experiments, the images were real-valued and so the standard DnCNN uses one input and output channel. When extending that DnCNN to corr+corr, we added a single noise channel. The training of the denoiser was identical to that used in the multicoil case, described in Appendix C.1.

# Appendix D: Variance of $p(z_i|y_i; \overline{z}_i^{(1)}, \overline{v}^{(1)})$

For brevity, we omit the variable index "$i$" since the development is identical for all indices $i$, as well as the superscript $(\cdot)^{(1)}$. We set $\widehat{v}$ at the approximation of the posterior variance given by the Laplace approximation [157], which is

$$\mathrm{tr}\{\boldsymbol{H}(\widehat{z})^{-1}\} \quad \text{for} \quad \boldsymbol{H}(\widehat{z}) \triangleq \nabla_z^2(-\ln p(z|y;\overline{z},\overline{v}))\big|_{z=\widehat{z}}, \tag{D.1}$$

where $\widehat{z}$ is the MAP estimate given in (3.36), i.e.,

$$\widehat{z} = \frac{\overline{v}y + 2v|\overline{z}|}{\overline{v}+2v} e^{j\angle \overline{z}}, \tag{D.2}$$

and $\nabla_z^2$ denotes the Hessian with respect to the real and imaginary parts of $z$.

In Appendix E, $\mathrm{tr}\{\boldsymbol{H}(\widehat{z})^{-1}\}$ is derived to be

$$\mathrm{tr}\left\{\boldsymbol{H}(\widehat{z})^{-1}\right\} = v\frac{2\frac{|\widehat{z}|}{y} - \frac{\overline{v}}{\overline{v}+2v}}{\frac{\overline{v}+2v}{\overline{v}}\frac{|\widehat{z}|}{y} - 1} \tag{D.3}$$

Assuming that $y \geq -2v|\overline{z}|/\overline{v}$, we have

$$2\frac{|\widehat{z}|}{y} - \frac{\overline{v}}{\overline{v}+2v} = \frac{2\overline{v}+4v|\overline{z}|/y}{\overline{v}+2v} - \frac{\overline{v}}{\overline{v}+2v} = \frac{\overline{v}+4v|\overline{z}|/y}{\overline{v}+2v} \tag{D.4}$$

and

$$\frac{\overline{v}+2v}{\overline{v}}\frac{|\widehat{z}|}{y} - 1 = \frac{\overline{v}+2v}{\overline{v}} \cdot \frac{\overline{v}+2v|\overline{z}|/y}{\overline{v}+2v} - 1 = \frac{2v|\overline{z}|/y}{\overline{v}} \tag{D.5}$$

so that

$$\mathrm{tr}\left\{\boldsymbol{H}(\widehat{z})^{-1}\right\} = v\frac{\overline{v}+4v|\overline{z}|/y}{\overline{v}+2v} \cdot \frac{\overline{v}}{2v|\overline{z}|/y} = \frac{\overline{v}(\overline{v}y+4v|\overline{z}|)}{2|\overline{z}|(\overline{v}+2v)}. \tag{D.6}$$

## Appendix E: Derivation of the Trace Inverse Hessian

Here we derive $\nabla_z^2(-\ln p(z|y;\overline{z},\overline{v}))$, the Hessian of $-\ln p(z|y;\overline{z},\overline{v})$ with respect to the real and imaginary components of $z \in \mathbb{C}$. We will use $z_r$ and $z_j$ to denote the real and imaginary components of $z$, and $\overline{z}_r$ and $\overline{z}_j$ to denote the real and imaginary components of $\overline{z}$, respectively. We have

$$-\ln p(z|y;\overline{z},\overline{v}) = \frac{1}{2v}(y-|z|)^2 + \frac{1}{\overline{v}}|z-\overline{z}|^2 + \text{const} \tag{E.1}$$

$$= \frac{1}{2v}\left(y - \sqrt{z_r^2+z_j^2}\right)^2 + \frac{1}{\overline{v}}\left((z_r-\overline{z}_r)^2 + (z_j-\overline{z}_j)^2\right) + \text{const}. \tag{E.2}$$

This implies that

$$-\frac{\partial}{\partial z_r}\ln p(z|y;\overline{z},\overline{v})$$

$$= \frac{1}{v}\left(\sqrt{z_r^2+z_j^2}-y\right)\frac{1}{2}(z_r^2+z_j^2)^{-1/2}2z_r + \frac{2}{\overline{v}}(z_r-\overline{z}_r) \tag{E.3}$$

$$= \frac{1}{v}\left(1 - \frac{y}{\sqrt{z_r^2+z_j^2}}\right)z_r + \frac{2}{\overline{v}}(z_r-\overline{z}_r), \tag{E.4}$$

and similarly that

$$-\frac{\partial}{\partial z_j}\ln p(z|y;\overline{z},\overline{v}) = \frac{1}{v}\left(1 - \frac{y}{\sqrt{z_r^2+z_j^2}}\right)z_j + \frac{2}{\overline{v}}(z_j-\overline{z}_j). \tag{E.5}$$

The second derivatives are

$$
\begin{aligned}
&-\frac{\partial^2}{\partial z_r^2}\ln p(z|y;\overline{z},\overline{v}) \\
&=\frac{\partial}{\partial z_r}\left(\frac{1}{v}\left(1-\frac{y}{\sqrt{z_r^2+z_j^2}}\right)z_r+\frac{2}{\overline{v}}(z_r-\overline{z}_r)\right) && \text{(E.6)}\\
&=\frac{1}{v}\left(1-\frac{y}{\sqrt{z_r^2+z_j^2}}+\frac{yz_r}{2[z_r^2+z_j^2]^{3/2}}2z_r\right)+\frac{2}{\overline{v}} && \text{(E.7)}\\
&=\frac{1}{v}\left(1-\frac{y(z_r^2+z_j^2)}{[z_r^2+z_j^2]^{3/2}}+\frac{yz_r^2}{[z_r^2+z_j^2]^{3/2}}\right)+\frac{2}{\overline{v}} && \text{(E.8)}\\
&=\frac{1}{v}\left(1-\frac{yz_j^2}{|z|^3}\right)+\frac{2}{\overline{v}}, && \text{(E.9)}
\end{aligned}
$$

and

$$
-\frac{\partial^2}{\partial z_j^2}\ln p(z|y;\overline{z},\overline{v})=\frac{1}{v}\left(1-\frac{yz_r^2}{|z|^3}\right)+\frac{2}{\overline{v}}, \tag{E.10}
$$

and

$$
\begin{aligned}
&-\frac{\partial^2}{\partial z_j\partial z_r}\ln p(z|y;\overline{z},\overline{v}) \\
&=\frac{\partial}{\partial z_j}\left(\frac{1}{v}\left(1-\frac{y}{\sqrt{z_r^2+z_j^2}}\right)z_r+\frac{2}{\overline{v}}(z_r-\overline{z}_r)\right) && \text{(E.11)}\\
&=\frac{1}{v}\frac{yz_r}{2[z_r^2+z_j^2]^{3/2}}2z_j=\frac{yz_rz_j}{v|z|^3}, && \text{(E.12)}
\end{aligned}
$$

and so the Hessian matrix is

$$
\begin{aligned}
\boldsymbol{H}(z)&=\begin{bmatrix}-\frac{\partial^2}{\partial z_r^2}\ln p(z|y;\overline{z},\overline{v}) & -\frac{\partial^2}{\partial z_j\partial z_r}\ln p(z|y;\overline{z},\overline{v})\\ -\frac{\partial^2}{\partial z_j\partial z_r}\ln p(z|y;\overline{z},\overline{v}) & -\frac{\partial^2}{\partial z_j^2}\ln p(z|y;\overline{z},\overline{v})\end{bmatrix}\\
&=\begin{bmatrix}\frac{1}{v}\left(1-\frac{yz_j^2}{|z|^3}\right)+\frac{2}{\overline{v}} & \frac{yz_rz_j}{v|z|^3}\\ \frac{yz_rz_j}{v|z|^3} & \frac{1}{v}\left(1-\frac{yz_r^2}{|z|^3}\right)+\frac{2}{\overline{v}}\end{bmatrix} && \text{(E.13)}\\
&=\left(\frac{1}{v}+\frac{2}{\overline{v}}\right)\boldsymbol{I}_2-\frac{y}{v|z|^3}\begin{bmatrix}z_j\\-z_r\end{bmatrix}\begin{bmatrix}z_j & -z_r\end{bmatrix}. && \text{(E.14)}
\end{aligned}
$$

Using the matrix inversion lemma,

$$
\begin{aligned}
&\operatorname{tr}\left\{(a\boldsymbol{I}_2+b\boldsymbol{c}\boldsymbol{c}^\top)^{-1}\right\}=\operatorname{tr}\left\{\frac{1}{a}\left(\boldsymbol{I}_2-\frac{b\boldsymbol{c}\boldsymbol{c}^\top}{a+b\boldsymbol{c}^\top\boldsymbol{c}}\right)\right\} && \text{(E.15)}\\
&=\frac{1}{a}\left(2-\frac{b\boldsymbol{c}^\top\boldsymbol{c}}{a+b\boldsymbol{c}^\top\boldsymbol{c}}\right)=\frac{2+a^{-1}b\boldsymbol{c}^\top\boldsymbol{c}}{a+b\boldsymbol{c}^\top\boldsymbol{c}}. && \text{(E.16)}
\end{aligned}
$$

Applying this to (E.14) using

$$a = \frac{1}{v} + \frac{2}{\overline{v}} = \frac{\overline{v} + 2v}{\overline{v}v} \tag{E.17}$$

$$b\boldsymbol{c}^\top \boldsymbol{c} = -\frac{y}{v|z|^3} \begin{bmatrix} z_j & -z_r \end{bmatrix} \begin{bmatrix} z_j \\ -z_r \end{bmatrix} = -\frac{y}{v|z|}, \tag{E.18}$$

we get

$$\operatorname{tr}\left\{\boldsymbol{H}(z)^{-1}\right\} = \frac{2 - \frac{\overline{v}v}{\overline{v}+2v}\frac{y}{v|z|}}{\frac{\overline{v}+2v}{\overline{v}v} - \frac{y}{v|z|}} = v\frac{2\frac{|z|}{y} - \frac{\overline{v}}{\overline{v}+2v}}{\frac{\overline{v}+2v}{\overline{v}}\frac{|z|}{y} - 1}. \tag{E.19}$$

# Appendix F: Implementation details of StRAPS and baseline methods

## F.1 Inverse problems

For the linear inverse problems, the measurements were generated as

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \sigma_{\mathsf{w}}\boldsymbol{w}, \quad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{F.1}$$

with appropriate $\boldsymbol{A}$. For box inpainting, Gaussian deblurring, and super-resolution we used the $\boldsymbol{A}$ and $\boldsymbol{A}^\top$ implementations from [179]. For motion deblurring, we implemented our own $\boldsymbol{A}$ and $\boldsymbol{A}^\top$ with reflect padding. All methods used these operators implementations except DiffPIR, which used the authors' implementations. Motion-blur kernels were generated using [173].

For phase retrieval, the measurements were generated using the method from [56]:

$$y_j^2 = |z_{0,j}|^2 + w_j, \quad w_j \sim \mathcal{N}(0, \alpha_{\mathsf{shot}}^2 |z_{0,j}|^2), \quad j = 1, \ldots, m, \tag{F.2}$$

where $\alpha_{\mathsf{shot}}$ controls the noise level and $\boldsymbol{z}_0 = \boldsymbol{A}\boldsymbol{x}_0$, with the values of $\boldsymbol{x}_0$ scaled to lie in the range $[0, 255]$. This is an approximation of the Poisson shot-noise corruption model in that the intensity $y_j^2/\alpha_{\mathsf{shot}}^2$ is approximately Poisson$((|z_{0,j}|/\alpha_{\mathsf{shot}})^2)$ distributed for sufficiently small values of $\alpha_{\mathsf{shot}}$. We implemented the oversampled-Fourier $\boldsymbol{A}$ by zero-padding the image by $2\times$ in each direction and then passing the result through a unitary

FFT. For CDP phase retrieval, we set $\boldsymbol{A} = [\boldsymbol{A}_1^\top, \ldots, \boldsymbol{A}_L^\top]^\top$ for $\boldsymbol{A}_l = L^{-1/2}\boldsymbol{F}\,\mathrm{Diag}(\boldsymbol{c}_l)$, where $\boldsymbol{F}$ is a $d \times d$ FFT and $\boldsymbol{c}_l$ contain i.i.d. random entries uniformly distributed on the unit circle in the complex plane, and where $L = 4$. In both cases, $\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{I}$.

## F.2 Evaluation protocol

For the linear inverse problems, we run each method once for each measurement $\boldsymbol{y}$ in the 1000-sample test set and compute average PSNR, average LPIPS, and FID from the resulting recoveries.

For OSF phase retrieval, following [76], we run each algorithm four times and keep the reconstruction $\widehat{\boldsymbol{x}}$ that minimizes the measurement residual $\|\boldsymbol{y} - |\boldsymbol{A}\widetilde{\boldsymbol{x}}_0|\|$. Performance metrics are then evaluated after resolving the inherent spatial shift and conjugate flip ambiguities associated with phase retrieval (see, e.g., [162]). Note global phase ambiguity is not an issue due to the non-negativity of our images. For the CDP experiments, we run each algorithm only once and don't perform ambiguity resolution, because it is unnecessary.

## F.3 Unconditional diffusion model

For the FFHQ experiments, all methods used the pretrained model from [76] with $T = 1000$.

## F.4 Recovery methods

**StRAPS.** Our Python/Pytorch codebase is a modification of the DPS codebase from [159]. For $\widehat{\sigma}^2_{0|i}$ estimation, we approximate $\|\boldsymbol{A}\|_F^2$ using $\frac{1}{L}\sum_{l=1}^{L}\|\boldsymbol{A}\boldsymbol{w}_l\|^2$ with i.i.d. $\boldsymbol{w}_l \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $L = 25$. For the linear inverse problems, we use conjugate gradient (CG) without SVD, and a fixed $\gamma_i$ across all iterations. The values are selected from a

grid of 15 logarithmically spaced points ranging from $10^{-1.5} \approx 0.0316$ to $10^{0.5} \approx 3.1622$, tuned to minimize LPIPS [174] on a 100-sample validation set (see Table F.1). For phase retrieval, we use 800 NFEs (to ensure fair comparison with prDeep) along with stochastic denoising [171], and similarly employ a fixed $\gamma_i$ across all iterations, selected from a logarithmic grid spanning $10^{-1.5} \approx 0.0316$ to $10^{0.75} \approx 5.6234$ (see Table F.2). Neither CG nor SVD is required, as (4.71) can be solved analytically.

Table F.1: Hyperparameter $\gamma$ values used for SLM-StRAPS.

| # NFEs | Inpaint (box) | Deblur (Gaussian) | Deblur (Motion) | 4× Super-resolution |
|---|---|---|---|---|
| 20 | 0.4394 | 0.0316 | 0.0316 | 0.0316 |
| 100 | 0.2276 | 0.0439 | 0.0611 | 0.0848 |
| 1000 | 0.0848 | 0.0316 | 0.0316 | 0.0316 |

Table F.2: Hyperparameter $\gamma$ values used for Phase Retrieval StRAPS.

| # NFEs | OSF | CDP |
|---|---|---|
| 100 | 1.2798 | 0.8840 |
| 800 | 0.4217 | 1.2798 |

**DDRM.** We use the authors' implementation from [179] with minor changes to work in our codebase.

**DiffPIR.** We use the authors' implementation from [180] without modification. Hyperparameters were set according to the reported values in [90].

**ΠGDM.** Since the authors do not provide a ΠGDM implementation for noisy problems in [181], we wrote our own ΠGDM implementation that computed $(\boldsymbol{A}\boldsymbol{A}^\top + \zeta_k \boldsymbol{I})^{-1}$ using the efficient SVD implementation of $\boldsymbol{A}$ from the DDRM codebase at [179] on problems for which an SVD is available, and otherwise used CG.

**DPS.** For the linear inverse problems, we use the authors' original implementation from [159] without modification. For all problems, we use the suggested scale factors from [76, Sec. D.1].

For phase retrieval, we made minor adjustments to the DPS authors' implementation to accommodate the likelihood $p_{\mathsf{y}|\mathsf{z}}(y|z) = \mathcal{N}(y; |z|, \sigma_{\mathsf{w}}^2)$ (used by all methods for fairness). We used grid-search to find the scale factor that minimized LPIPS on a 100-image validation set. The resulting values were 0.0075 for OSF and 0.05 for CDP.

**DAPS.** For linear inverse problems, we use the authors' implementation from [182] with minor modifications to integrate it into our codebase.

For phase retrieval, we apply similar adjustments as those made for DPS to ensure fairness.

**DOLPH.** As the DOLPH implementation is not publicly available, we implemented DOLPH in Python/PyTorch and used grid-search to find the step-size that minimized LPIPS on a 100-image validation set. The resulting step-sizes were $5 \times 10^{-6}$ for OSF and $5 \times 10^{-7}$ for CDP.

**HIO.** We translated the MATLAB implementation of HIO from [164] to Python and set the step-size parameter to 0.9. We then followed the runtime procedure described in [56]: For the OSF experiments, HIO is first run 50 times, for 50 iterations

each, from a random initialization. The estimate $\widehat{\boldsymbol{x}}$ with the lowest measurement residual $\|\boldsymbol{y} - |\boldsymbol{A}\widehat{\boldsymbol{x}}|\|$ is then used to reinitialize HIO, after which it is run for 1000 more iterations. Finally, the second and third color channels in the result are shifted and flipped as needed to best match the first color channel. For the CDP experiments, HIO is run once for 200 iterations from a random initialization.

**prDeep.** We used the Python implementation from [165]. As recommended in [56], we initialized prDeep with the HIO estimate for OSF experiments and with an all-ones initialization for CDP experiments. (Note that only prDeep uses the HIO initialization; DPS, DAPS, DOLPH, and StRAPS do not.) We tuned $\lambda$ on a grid to minimize LPIPS on a 100-image validation set, which led to $\lambda = 0.1\sigma_{\mathsf{w}}$ for OSF and $\lambda = 0.01\sigma_{\mathsf{w}}$ for CDP.