Thesis Formulation Report

Plug-and-Play regularisation techniques for inverse problems

Amin Sabir
Supervised by: Dr. Yury Korolev and Dr. Matthias Ehrhardt
July-September 2024

Abstract

This thesis formulation report (TFR) explores plug-and-play (PnP) regularisation techniques for imaging inverse problems, with a particular focus on computed tomography (CT). Inverse problems involve reconstructing a quantity of interest from indirect and often noisy measurements, which can lead to instability when inverting the forward model directly. Regularisation has been widely used to combat these issues with traditional techniques often relying on handcrafted regularisers to impose properties such as smoothness. Recently, data-driven methods, such as PnP, offer an adaptive approach by integrating learned priors from training data. This report reviews variable-splitting algorithms with various denoisers in a PnP framework and studies their fixed-point convergence theorems. Numerical experiments demonstrate PnP's effectiveness in CT, empirically test the fixed-point convergence theorems and analyse the residual noise distribution of denoisers compared to the measurement noise in the inverse problem. Overall, this work lays the foundations for research ideas to be pursued in a forthcoming PhD.

List of abbreviations

- CBSD - Colour Berkley Segmentation Dataset
- CT - Computed Tomography
- FBP - Filtered Back-projection
- KŁ - Kurdyka-Łojasiewicz
- KS - Kolmogorov–Smirnov
- L.S.C/l.s.c - Lower Semi-Continuous
- MNIST - Modified National Institute of Standards and Technology Database
- PDF - Probability Density Function
- SOTA - State-of-the-Art

Optimisation

- ADMM - Alternating Direction Method of Multipliers
- FISTA - Fast Iterative Shrinkage Thresholding Algorithm
- MAP - Maximum A-Posteriori
- MMSE - Minimum Mean Squared Error
- PGD/FBS - Proximal Gradient Descent/Forward-Backward Splitting
- PnP-ADMM/PGD - Plug-and-play ADMM/PGD
- PSNR - Peak signal-to-noise ratio

Priors/denoisers

- BM3D - Block Matching 3D
- CNN - Convolutional Neural Network
- DnCNN - Denoising Convolutional Neural Network
- DRUNet - Dilated Residual U-Network
- GS - Gradient Step
- NN - Neural Network
- PnP - Plug-and-Play
- TV - Total Variation

Contents

1 Introduction

1.1 Background and motivation

In inverse problems, we are interested in reconstructing unknown quantities from an observation. Often a forward model or operator is used to describe the measurement process to obtain this observation. For example, in computed tomography (CT) the task is to reconstruct the image of the object of interest (e.g. an organ in a human body), using a *sinogram* measurement, which is related to the absorption pattern of transmitted X-rays. Other examples include image deblurring and denoising [Scherzer et al. (2009)] each with the main goal of reconstructing the quantities of interest.

We have an unknown ground truth $\boldsymbol{x} \in \mathcal{X}$ given an observation $\boldsymbol{y} \in \mathcal{Y}$, where $\mathcal{X}, \mathcal{Y}$ are Hilbert spaces. Note that in practical applications, such as medical imaging, these spaces are always approximated by finite dimensions for computation. In this instance, the image pixel resolution might be determined by measuring equipment that can only take 1024×1024 pixels for example. The two quantities are related by a forward operator (usually known) $A : \mathcal{X} \to \mathcal{Y}$ such that

$$\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{\epsilon} \tag{1.1}$$

where $\boldsymbol{\epsilon}$ denotes noise for the observation (e.g. additive Gaussian noise). A common forward operator is the Radon transform used in CT, which will be the main application used in this report (see Section 2.3). This report only considers (1.1) in finite dimensions (e.g. $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{y}, \boldsymbol{\epsilon} \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$) and discrete linear inverse problems, where A is a linear operator.

The challenge with inverse problems such as (1.1) is that often these problems are *ill-posed* against **Hadamard's criteria** for *well-posedness* [Hadamard (1902)]. This is where the inverse problem violates the following conditions for the solution $\boldsymbol{x}$ to be well-posed:

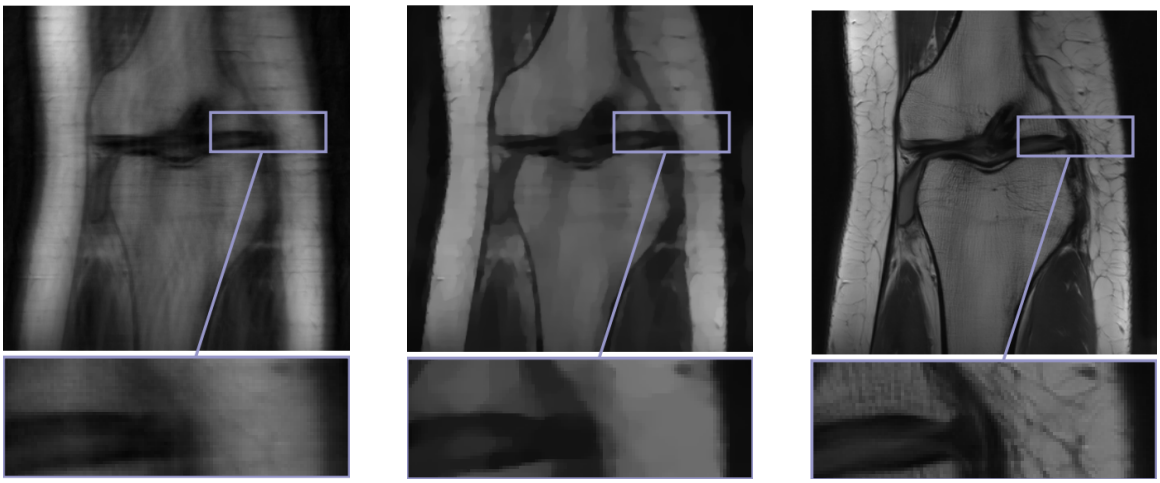
1. A solution exists for all $\boldsymbol{y} \in \mathcal{Y}$
2. The solution is unique
3. The solution depends continuously on data, i.e. small errors in $\boldsymbol{y}$, result in small errors in $\boldsymbol{x}$ (stable to perturbations).

For our purposes, we assume that the first two conditions are satisfied, while the third condition is violated. This implies that our focus lies on the noise present in $\boldsymbol{y}$, which induces instability in our solution $\boldsymbol{x}$. The resulting estimate for $\boldsymbol{x}$ is amplified with noise if the forward model is inverted naively.

A popular method to overcome the ill-posedness is to use variational methods for regularisation, where for (1.1) we solve

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \{ \underbrace{f(\boldsymbol{x})}_{\text{data fidelity}} + \underbrace{\lambda g(\boldsymbol{x})}_{\text{regulariser}} \} \tag{1.2}$$

where $\boldsymbol{x}^*$ is a minimiser and $f(\boldsymbol{x})$ is the data-fidelity term, trying to best approximate the solution, e.g. $f(\boldsymbol{x}) = \frac{1}{2}||A\boldsymbol{x} - \boldsymbol{y}||_2^2$. The term g is a regularisation functional representing a regulariser with a regularisation parameter $\lambda > 0$, to keep (1.2) well-posed. This functional can encode prior knowledge about $\boldsymbol{x}$ in the reconstruction. An example for g is using total variational (TV) regularisation where $g = \mathrm{TV}(\boldsymbol{x}) = ||\nabla \boldsymbol{x}||_1$ [Rudin et al. (1992)]. Often these methods are hand-crafted by being analytically defined to reconstruct the characteristics of a solution, for example, the edges or smoothness of the ground truth (see Figure 1 for an example).



(a) Zero-fill reconstruction **(b)** Hand crafted TV prior **(c)** CNN-based reconstruction

Figure 1: Comparison of different reconstruction methods for magnetic resonance imaging (MRI) from sub-sampled data of a human body. **(a)** Zero-fill - setting unknown data measurements to zero naively, **(b)** total variation (TV), **(c)** a convolutional neural network (CNN) [Habring and Holler (2023)]

Recently, data-driven methods such as *plug-and-play* (PnP) [Venkatakrishnan et al. (2013)], algorithm unrolling [Monga et al. (2020)] and adversarial regularisers [Lunz et al. (2019)], have been used to learn what features of the ground truth to construct, as well as the regularity of the solution from training data (Figure 1). These are often more adaptable to different data than classical methods and have given interesting insights into tackling inverse problems as indicated in a review paper by Habring and Holler (2023) on such data-driven techniques.

1.2 Report structure

The main focus of this report is on PnP methods for data-driven regularisation. We begin with the mathematical preliminaries involving the optimisation framework and algorithms required for PnP, while introducing the CT inverse problem application (Section 2). In the next two sections, we concentrate on the PnP setup from a theoretical and a numerical perspective. First, we concentrate on the PnP literature with its theoretical formulation, involving the adaptations of these algorithms with different plug-in Gaussian denoisers, alongside an overview of theorems regarding the fixed-point convergence of such adapted algorithms (Section 3). When tracking the development of these theorems and denoisers used in PnP, this leads us to a particular interest in gradient-step denoisers (Section 3.4). Then for the numerical experiments, we first put the PnP theory into practice for the CT inverse problem, with investigations using different types of images and empirically testing the fixed-point convergence theorems (Section 4.1).

Next, there is a focus on the denoising operation of classical and deep learning denoisers for this inverse problem (Section 4.2). Here, we investigate the type of noise a denoiser removes within the steps of a PnP algorithm and how this type of noise may change over algorithm iterations when compared to the assumed measurement noise (e.g. Gaussian) of the inverse problem. We study whether

this "residual noise" distribution (the difference between the inversion and denoising steps in the algorithms) matches the assumed measurement noise distribution for the inverse problem. From the PnP literature [Venkatakrishnan et al. (2013); Zhang et al. (2021)], one might expect these noise distributions to match, but this remains an open question. In our experiments, we aim to test the hypothesis that the measurement noise is linked to the noise the denoisers remove across algorithm iterations. For instance, the change of measurement noise (e.g. Gaussian to Poisson) might not be always replicated in the corresponding denoiser's residual noise distribution.

We then conclude this report with a discussion reviewing PnP as a whole with our numerical experiments and look forward to ideas of what the PhD would entail for the immediate future (Section 5). Some of these potential ideas include, further investigating the impact of types of measurement noise affecting the denoisers, taking PnP forward in other formulations such as a consensus-based optimisation framework [Buzzard et al. (2018)] and looking into the wider industry application of PnP compared to other data-driven regularisation methods.

Note that, while this report focuses mainly on PnP regularisation, the overarching PhD theme will be on data-driven regularisation methods in general, for imaging inverse problems.

2 Mathematical preliminaries

Before we discuss the main parts of this report, we will first look at some mathematical preliminaries such as the optimisation framework and algorithm settings for PnP to work.

2.1 Maximum a-posteriori optimisation framework

Data-driven methods for regularisation often build from the Bayesian perspective of (1.1) [Habring and Holler (2023)]. First, let $\mathcal{X}$ and $\mathcal{Y}$ be finite-dimensional. Then, $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{\epsilon}$ are modelled as random variables X and Y (with values in $\mathcal{X}$ and $\mathcal{Y}$ respectively) with $\boldsymbol{\epsilon}$ as Gaussian noise, $\mathcal{N}(\mathbf{0}, \sigma^2 \mathrm{Id})$ for $\mathrm{Id} \in \mathbb{R}^m = \mathcal{Y}$ as the identity operator in finite dimensions. We use Bayes' theorem on the posterior distribution of $\mathbb{P}(\boldsymbol{x}|\boldsymbol{y})$

$$\mathbb{P}(\boldsymbol{x}|\boldsymbol{y}) = \frac{\mathbb{P}(\boldsymbol{y}|\boldsymbol{x})\mathbb{P}(\boldsymbol{x})}{\mathbb{P}(\boldsymbol{y})}, \tag{2.1}$$

$$-\log \underbrace{\mathbb{P}(\boldsymbol{x}|\boldsymbol{y})}_{\text{posterior}} = -\log \underbrace{\mathbb{P}(\boldsymbol{y}|\boldsymbol{x})}_{\text{likelihood}} - \log \underbrace{\mathbb{P}(\boldsymbol{x})}_{\text{prior}} + \log \mathbb{P}(\boldsymbol{y}) \tag{2.2}$$

where $\mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$ is the data likelihood and $\mathbb{P}(\boldsymbol{x})$ is the prior distribution of the ground truth $\boldsymbol{x} \in \mathcal{X}$. We can seek the maximum *a-posteriori* (MAP) estimate where we maximise $\mathbb{P}(\boldsymbol{x}|\boldsymbol{y})$ to find $\boldsymbol{x}^*$

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}} \{\mathbb{P}(\boldsymbol{x}|\boldsymbol{y})\} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \{-\log \mathbb{P}(\boldsymbol{y}|\boldsymbol{x}) - \log \mathbb{P}(\boldsymbol{x})\}. \tag{2.3}$$

By letting the data-fidelity term $f(\boldsymbol{x}) := -\log \mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$ and regularisation term $g(\boldsymbol{x}) := -\log \mathbb{P}(\boldsymbol{x})$ we retrieve (1.2)

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \{f(\boldsymbol{x}) + \lambda g(\boldsymbol{x})\}, \tag{2.4}$$

giving the general MAP estimate formulation for proper (function is not ∞ everywhere) and lower semi-continuous (l.s.c) functions $f, g : \mathbb{R}^n \to \mathbb{R}$. Note that we can consider f to be convex and sometimes differentiable but g is non-convex. For the Gaussian noise case with standard deviation σ, we have differentiable $f(\boldsymbol{x}) = \frac{1}{2\sigma^2}||\boldsymbol{y} - A\boldsymbol{x}||_2^2$ and $g(\boldsymbol{x})$ is weighted by the regularisation parameter $\lambda = \sigma^2$. The relationship between (1.2) and (2.4) exists for other types of noise, where $f(\boldsymbol{x})$ can have different expressions. For instance with Poisson noise, $f(\boldsymbol{x})$ is the Kullback-Leibler divergence

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} y_i \log\left(\frac{y_i}{(A\boldsymbol{x})_i}\right) + (A\boldsymbol{x})_i - y_i, \text{ where } y_i \geq 0 \text{ and } (A\boldsymbol{x})_i > 0, \tag{2.5}$$

for the measurement $\boldsymbol{y} = (y_1, \ldots, y_m)^T \in \mathbb{R}^m$ from (1.1).

Often the data-likelihood term $\mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$ is modelled by the forward model and the prior $\mathbb{P}(\boldsymbol{x})$ is modelled by various methods including classical priors such as total variation. Here we focus on the data-driven approaches in particular PnP, which uses different priors that can often be trained on different datasets than the inverse problem we use the prior for. In the latter case, we would like to train a model that can approximate a quantity related to $\mathbb{P}(\boldsymbol{x})$. The goal is to use this prior distribution to combat the ill-posedness of the problem, by modelling realistic priors to fit the problem setting.

Generally (2.4) cannot be solved analytically and numerical optimisation algorithms are required. In the next section, we discuss such algorithms and later on (Section 3) we see how they can be adapted in the PnP setting.

2.2 First-order variable-splitting algorithms

When solving (2.4), we can consider optimising for $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ separately into two sub-problems, one focusing on the data fidelity f and the other on the regulariser g. This splitting into the different sub-problems, allows one to optimise on two separate problems more easily than solving the entire problem itself. The algorithms used for variable-splitting, often stem from the use of a proximal operator.

Definition 2.1. *(Proximal operator)*
For a proper and l.s.c $f : \mathbb{R}^n \to \mathbb{R}$, the proximal operator of f denoted by $\boldsymbol{Prox}_f$ is the point to set mapping defined by

$$\boldsymbol{Prox}_f(\boldsymbol{x}) = \underset{\boldsymbol{y}\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ f(\boldsymbol{y}) + \frac{1}{2}||\boldsymbol{x}-\boldsymbol{y}||_2^2 \right\}. \tag{2.6}$$

The use of the proximal operator is beneficial when either f or g is not smooth. In this report, we extensively use two first-order proximal variable-splitting algorithms, which only contain the first derivative of f, while also using the proximal operator on f and g. These are proximal gradient descent and alternating direction method of multipliers.

2.2.1 Proximal gradient descent (PGD)

Proximal gradient descent (PGD) or forward-backward splitting (FBS) [Chambolle and Pock (2016)] is an optimisation algorithm that alternates between the conventional gradient descent on f ('forward' step), while also using the proximal operator on g ('backward' step, see Algorithm 1).

Algorithm 1: PGD [Chambolle and Pock (2016)]

Initial guess $\boldsymbol{x}^{(0)}$, N, step size $\tau > 0$ and $tol > 0$
For $k = 0, 1, 2, \ldots, N$

$$\boldsymbol{x}^{(k+1)} = \underbrace{\mathbf{Prox}_{\tau g}}_{\text{backward step}} \left(\underbrace{\boldsymbol{x}^{(k)} - \tau\nabla f(\boldsymbol{x}^{(k)})}_{\text{forward step}}\right),$$

Repeat until $||\nabla f(\boldsymbol{x}^{(k)})||_2 \leq tol$ or $k = N$

Note that f has to be differentiable and have Lipschitz gradient L. There is an accelerated version of PGD called the fast iterative shrinkage algorithm (FISTA) [Beck and Teboulle (2009)].

Algorithm 2: FISTA [Beck and Teboulle (2009)]

Initial guess $\boldsymbol{x}^{(0)}, N, tol > 0$
$\boldsymbol{y}^{(0)} = \boldsymbol{x}^{(0)}, t_0 = 1$ (introduces an auxiliary variable and initialises the first step size)
For $k = 0, 1, 2, \ldots, N$

$\boldsymbol{x}^{(k+1)} = \mathbf{Prox}_{(\frac{1}{L})g}(\boldsymbol{y}^{(k+1)} - (\frac{1}{L})\nabla f(\boldsymbol{y}^{(k+1)}))$, L is a Lipschitz constant for ∇f

$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$

$\boldsymbol{y}^{(k+1)} = \boldsymbol{x}^{(k+1)} + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)})$

Repeat until $||\nabla f(\boldsymbol{y}^{(k)})||_2 \leq tol$ or $k = N$

FISTA involves a momentum term, incorporating a weighted average of the previous and current iterations in the algorithm, helping to speed up the algorithm's convergence (see Algorithm 2).

For both PGD and FISTA, there exist fixed-point convergence theorems for convex and non-convex functions. The convex case is more straightforward and so we state the theorem for non-convex $F := f + g$.

Theorem 2.1. *(Fixed-point convergence of PGD and FISTA)*
Assume f and g are bounded from below, proper and lower semi-continuous with L Lipschitz constant for ∇f. Then for step size $\tau < \frac{1}{L}$, the iterates $\boldsymbol{x}^{(k)}$ for $k \in \mathbb{N}$ satisfy

1. *$F(\boldsymbol{x}^{(k)})$ is non-increasing and converges*
2. *The sequence $\sum_{k=1}^{N} \|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2 < \infty$ and $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2 \to 0$ as $k \to \infty$.*

This theorem is proven in Attouch et al. (2011) with the use of semi-algebraic sets and inequalities on f, particularly using the Kurdyka-Łojasiewicz inequality.

Definition 2.2. *(Kurdyka-Łojasiewicz (KŁ) property and inequality on functions, Attouch et al. (2011))*
A function $f : \mathbb{R}^n \to \mathbb{R}$ has the KL property for $\boldsymbol{x}^* \in dom(f)$, if there exists $\eta \in (0, \infty)$ and a continuous concave function $\psi : (0, \eta) \to \mathbb{R}_+$ such that $\psi(0) = 0$, ψ is 1-differentiable continuous $\mathcal{C}^1$ on $(0, \eta)$ and the KL inequality holds

$$\frac{1}{d(0, \partial f(x))} \leq \psi'(f(\boldsymbol{x}) - f(\boldsymbol{x}^*)), \quad \forall \boldsymbol{x} \in U \cap (f(\boldsymbol{x}^*), f(\boldsymbol{x}^*) + \eta) \tag{2.7}$$

where U is a neighbourhood of $\boldsymbol{x}^*$ and $d(\cdot, \cdot)$ denotes a distance function (e.g. 2-norm), in this case between 0 and the subdifferential $\partial f(\boldsymbol{x})$.

This condition can be interpreted as, up to a reparameterisation, the subdifferentials at $\boldsymbol{x}$ are bounded away from 0. This property is a local property on the functions and is powerful for non-convex functions, where the regulariser g can be in (2.3), meaning this property provides a reasonable structure for functions to be used in optimisation problems. Also, the inequality (2.7) is crucial in the convergence of numerical optimisation algorithms in the non-convex setting [Attouch et al. (2011)]. We see this for variable-splitting algorithms and their PnP versions in Section 3.

2.2.2 Alternating directions method of multipliers (ADMM)

Another method is ADMM [Boyd (2010)], which involves more substeps for an iteration than PGD and FISTA. Here, the two sub-problems in (2.3) are split into two variables $(\boldsymbol{x}, \boldsymbol{u})$

$$(\boldsymbol{x}^*, \boldsymbol{u}^*) = \underset{\boldsymbol{x}, \boldsymbol{u}}{\operatorname{argmin}} \left\{ f(\boldsymbol{x}) + \lambda g(\boldsymbol{u}) \right\}, \tag{2.8}$$

where over time we eventually have $\boldsymbol{x} = \boldsymbol{u}$. To do this, we introduce an augmented Lagrangian for (2.8) and optimise over this Lagrangian

$$\begin{aligned} L_\beta(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) &= f(\boldsymbol{x}) + \lambda g(\boldsymbol{u}) + \frac{\beta}{2} ||\boldsymbol{x} - (\boldsymbol{u} - \boldsymbol{v})||_2^2 - \frac{\beta}{2} ||\boldsymbol{v}||_2^2, \\ &= \frac{1}{\beta} f(\boldsymbol{x}) + \sigma^2 g(\boldsymbol{u}) + \frac{1}{2} ||\boldsymbol{x} - (\boldsymbol{u} - \boldsymbol{v})||_2^2 - \frac{1}{2} ||\boldsymbol{v}||_2^2 \end{aligned}$$

$$\min_{\boldsymbol{x},\boldsymbol{u}} \{L_\beta(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v})\} \tag{2.9}$$

where $\beta > 0$ is a moderation parameter for the regularisation of λ and $\sigma^2 = \frac{\lambda}{\beta}$ with $\boldsymbol{v}$ as a scaled dual variable. The ADMM algorithm is then performed as outlined in Algorithm 3.

Algorithm 3: ADMM [Boyd (2010)]

Initial guess $\boldsymbol{x}^{(0)}, \boldsymbol{u}^{(0)}, \boldsymbol{v}^{(0)}$, specify $f(\boldsymbol{x})$ and $g(\boldsymbol{u})$, $tol > 0$

For $k = 0, 1, 2, \ldots, N$

1. $\boldsymbol{x}^{(k+1)} = \mathbf{Prox}_{\frac{1}{\beta}f}(\boldsymbol{u}^{(k)} - \boldsymbol{v}^{(k)}) = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{\beta} f(\boldsymbol{x}) + \frac{1}{2}||\boldsymbol{x} - (\boldsymbol{u}^{(k)} - \boldsymbol{v}^{(k)})||_2^2 \right\}$
2. $\boldsymbol{u}^{(k+1)} = \mathbf{Prox}_{\sigma^2 g}(\boldsymbol{x}^{(k+1)} + \boldsymbol{v}^{(k)}) = \underset{\boldsymbol{u}}{\operatorname{argmin}} \left\{ \frac{1}{2}||\boldsymbol{u} - (\boldsymbol{x}^{(k+1)} + \boldsymbol{v}^{(k)})||_2^2 + \sigma^2 g(\boldsymbol{u}) \right\}$
3. $\boldsymbol{v}^{(k+1)} = \boldsymbol{v}^{(k)} + \boldsymbol{x}^{(k+1)} - \boldsymbol{u}^{(k+1)}$

Repeat until a defined tolerance is met or $k = N$

In line 2 of Algorithm 3, the iteration step for $\boldsymbol{u}^{(k+1)}$ can be recognised as a variational method for denoising $\boldsymbol{x}^{(k+1)} + \boldsymbol{v}^{(k)}$. This is analogous to the variational problem in (1.2) and (2.3) with $f(\boldsymbol{r}) = \frac{1}{2\sigma^2}||\boldsymbol{u} - \boldsymbol{r}||_2^2$ and $\lambda g(\boldsymbol{u}) = \sigma^2 g(\boldsymbol{u})$ where $\boldsymbol{r} = \boldsymbol{x}^{(k+1)} + \boldsymbol{v}^{(k)}$ at the k^{th} iteration. The denoising part comes from recovering the ground truth $\boldsymbol{x}^*$ from noisy observations of $\boldsymbol{r}$

$$\boldsymbol{r} = \boldsymbol{x}^* + \boldsymbol{\zeta}, \quad \boldsymbol{\zeta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathrm{Id}) \tag{2.10}$$

for $\sigma > 0$ and $\mathrm{Id} \in \mathbb{R}^n$.

Similar to PGD and FISTA, there exist fixed-point convergence theorems for convex and non-convex functions of $F =: f + g$ for ADMM outlined in Chambolle and Pock (2016).

2.3 Computed tomography inverse problem

For our numerical experiments in Section 4, we will predominantly use the computed tomography (CT) inverse problem setup. CT is a medical imaging technique used to create detailed cross-sectional images of the inside of the human body. It is a non-invasive method that allows physicians to visualise the internal structures of the body without the need for surgery (see Figure 2). See Figure 2 for an example.

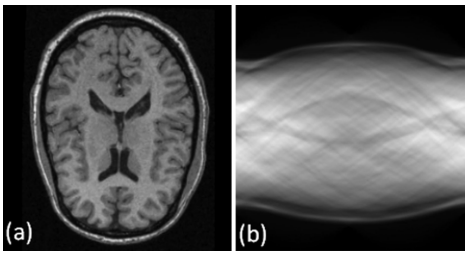


Figure 2: A typical CT scan of a brain with its sinogram [Yang and Fei (2013)]

Mathematically as an inverse problem, the process involves the Radon transform [Radon (1917)] as the forward operator A in (1.1), for an object $\boldsymbol{x}$ to be scanned, which generates a measurement $\boldsymbol{y}$ called a *sinogram*. This sinogram is a collection of sine waves of different amplitudes and phases which gives an indirect reading of a CT scan for a given object. The Radon transform projects the object through a series of line integrals across different angles as the scanner rotates around the object (Figure 3). In our experiments (Section 4), we use the discretised version of the Radon transform, $A \in \mathbb{R}^{m \times n}$, where m is the number of scanner detector elements multiplied by the number of projection angles taken from the scanner and n is the number of pixels in the image $\boldsymbol{x}$.

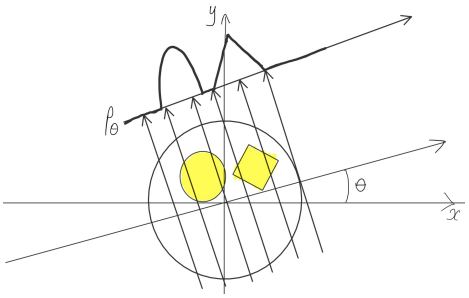


Figure 3: An example of a 2D projection p_θ in the x-y plane for a given angle θ for a circle and square object. The resulting Radon transform is a collection of all projections across all angles.

Inverting the sinogram is the essence of the inverse problem. There is an analytical inverse called the filtered back-projection (FBP). This inversion is quick and often reasonable for reconstructions, however, it suffers when there is noise present in the data, giving low-quality and noisy reconstructions. We see this for a range of Gaussian noise applied to the sinogram of the Shepp-Logan Phantom and FBP applied in Figure 4. For our experiments (Section 4), we investigate different regularisation techniques including classical and PnP methods for obtaining adequate reconstructions of objects with noise present.

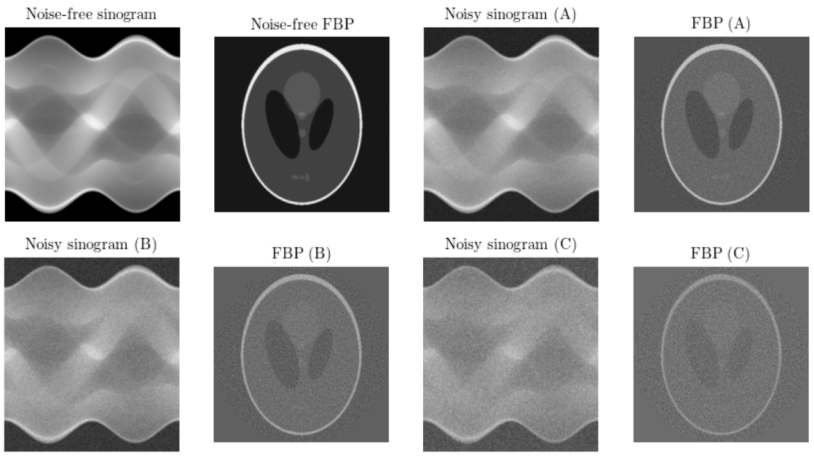


Figure 4: Shepp-Logan Phantom sinograms with their FBP: One noise-free and A, B, C with increasing Gaussian noise (Intensities: 5, 10, 20 $\times \mathcal{N}(0, 1)$ added across each sinograms respectively)

3 Plug-and-play regularisation

Plug-and-play (PnP) regularisation or priors, first proposed by Venkatakrishnan et al. (2013) is a hybrid approach, combining model-based and learning-based reconstruction methods, that exploits the variable-splitting algorithms mentioned in Section 2.2. PnP builds on the variable-splitting by replacing the proximal operator in the regularisation sub-problem with a plug-in denoiser, which may or may not have explicit knowledge of the overall problem that the algorithm is trying to solve. The regularisation step in (1.1) and (2.3) is replaced by learned denoisers. Often these denoisers try to estimate the prior distribution of clean images for the inverse problem and tend to be very successful in a host of tasks to do with image restoration involving denoising and deblurring [Zhang et al. (2017)].

3.1 PnP algorithm adaptations

The PnP adaptations of the algorithms, presented in Section 2.2, involve replacing the proximal step with a denoiser $\mathcal{D}_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ which is built to remove Gaussian noise with standard deviation σ. We see this by using the proximal operator definition (2.6) for τg

$$\mathbf{Prox}_{\tau g}(\boldsymbol{y}) = \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \frac{1}{2\tau}||\boldsymbol{y}-\boldsymbol{x}||_2^2 + g(\boldsymbol{x}) \right\} \tag{3.1}$$

where (3.1) is equivalent to the MAP estimate for Gaussian noise (2.3) with prior g and the stepsize $\tau = \sigma^2$. Therefore, we replace the proximal denoising step in the variable-splitting algorithms with a plug-in Gaussian denoiser $\mathcal{D}_\sigma$

$$\mathbf{Prox}_{\tau g} \to \mathcal{D}_\sigma \tag{3.2}$$

that tries to approximate the proximal operator and give greater flexibility to the algorithms. In the case of PGD (Algorithm 1), the backward step becomes

$$\boldsymbol{x}^{k+1} = \mathcal{D}_\sigma(\boldsymbol{x}^{(k)} - \tau\nabla f(\boldsymbol{x}^{(k)})) \tag{3.3}$$

to form PnP-PGD. A similar adjustment is taken to Algorithm 2 to obtain PnP-FISTA. For ADMM, the denoiser replaces the proximal step for g only, as outlined in Algorithm 4.

Algorithm 4: PnP-ADMM [Venkatakrishnan et al. (2013)]

Initial guess $\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{v}_0$, specify $f(\boldsymbol{x})$ and $g(\boldsymbol{u})$, $tol > 0$

For $k = 0, 1, 2, \ldots, N$

1. $\boldsymbol{x}^{(k+1)} = \mathbf{Prox}_{\frac{1}{\beta}f}(\boldsymbol{u}^{(k)} - \boldsymbol{v}^{(k)})$
2. $\boldsymbol{u}^{(k+1)} = \mathcal{D}_\sigma(\boldsymbol{x}^{(k+1)} + \boldsymbol{v}^{(k)})$
3. $\boldsymbol{v}^{(k+1)} = \boldsymbol{v}^{(k)} + \boldsymbol{x}^{(k+1)} - \boldsymbol{u}^{(k+1)}$

Repeat until a defined tolerance is met or $k = N$

Note that, the PnP algorithms work well in the cases where the proximal operator (2.6) has a closed-form solution. For instance, with Gaussian noise inverse problems where $f(\boldsymbol{x}) = \frac{1}{2}||A\boldsymbol{x}-\boldsymbol{y}||_2^2$, we

can compute $\mathbf{Prox}_{\tau f}(\boldsymbol{x})$ explicitly

$$\begin{aligned} &\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{\tau}{2}||A\boldsymbol{x}-\boldsymbol{y}||_2^2 + \frac{1}{2}||\boldsymbol{x}-\boldsymbol{y}||_2^2 \right\}, \\ &\implies 0 = \tau A^T(A\boldsymbol{x}-\boldsymbol{y}) + (\boldsymbol{x}-\boldsymbol{y}), \end{aligned}$$

$$\mathbf{Prox}_{\tau f}(\boldsymbol{x}) = (\tau A^T A + \mathrm{Id})^{-1}(\tau A^T \boldsymbol{y} + \boldsymbol{x}) \tag{3.4}$$

where Id is the identity operator or identity matrix for $\mathbb{R}^{m\times n}$, if our $\mathcal{X}$ and $\mathcal{Y}$ are finite dimensions for $\mathbb{R}^n$ and $\mathbb{R}^m$. The critical calculation is the inverse of $(\tau A^* A + \mathrm{Id})$, where if A has an easily computable singular-value decomposition then (3.4) has a closed-form. For inverse problems such as *inpainting*, which is an image restoration process that fills in damaged or missing parts of an image, the forward operator A is diagonal and therefore its $\mathbf{Prox}_{\tau f}(\boldsymbol{x})$ has a closed-form. Where this is not the case, $\mathbf{Prox}_{\tau f}$ is approximated via numerical optimisation algorithms such as PGD and ADMM (Section 2.2).

Other formulations of PnP can be constructed including half-quadratic splitting and primal-dual methods [Chambolle and Pock (2016)], which try to achieve a similar goal with the separate sub-problems for the data and denoising problems.

3.2 Denoiser choices

The flexibility of choosing between a variety of denoisers makes PnP very versatile. With the replacement of the proximal operator with the plug-in Gaussian denoisers that represent the prior of the ground truth, we can consider a host of denoisers.

The denoisers we consider include:

1. Total variation (**TV**) - $g = ||\nabla \boldsymbol{x}||_1$ in (1.2), used as a baseline for comparison
2. Block-matching 3D (**BM3D**) - classical denoiser
3. Denoising CNN (**DnCNN**) - deep learning denoiser
4. Dilated-residual U-network (**DRUNet**) - deep learning denoiser
5. Gradient-step DRUNet (**GS-DRUNet**) - a modified version of DRUNet (see Section 3.4).

There has been much success with classical denoisers [Venkatakrishnan et al. (2013); Chan et al. (2016)] such as block-matching 3D (BM3D) [Danielyan et al. (2012)] and K-SVD [Aharon et al. (2006)]. For BM3D, the algorithm groups similar image pixels of an image input into matching blocks and applies a denoising filter (e.g. Wiener filter) on each group before aggregating to form the result. In comparison, K-SVD is a generalised version of K-means clustering where sparse coding and dictionary learning are used. Although these classical denoisers often perform well, they can be time-consuming and computationally expensive.

Recently, there has been a focus on using data-driven involving deep learning denoisers to achieve state-of-the-art (SOTA) restoration performance for inverse problem tasks. These often utilise convolutional neural networks (CNNs). There are multiple prominent deep CNN denoisers to choose from [Habring and Holler (2023)] but for our numerical experiments, we focus on two, which are the denoising CNN (DnCNN) [Zhang et al. (2017)] and DRUNet [Zhang et al. (2021)].

Both DnCNN and DRUNet employ the U-Net CNN architecture. This architecture consists of two main paths involving an encoder and a decoder. The encoder path progressively downsamples an image input through a series of convolutional and max pooling layers, extracting meaningful feature maps while reducing the overall resolution of the input image. The decoder path applies the encoder's layers in reverse order. The decoder path upsamples these feature maps back to an image of the same spatial resolution as the image input. The U-Net name is derived from the structure of the symmetric encoder and decoder paths performed to the input image (see Figure 5).

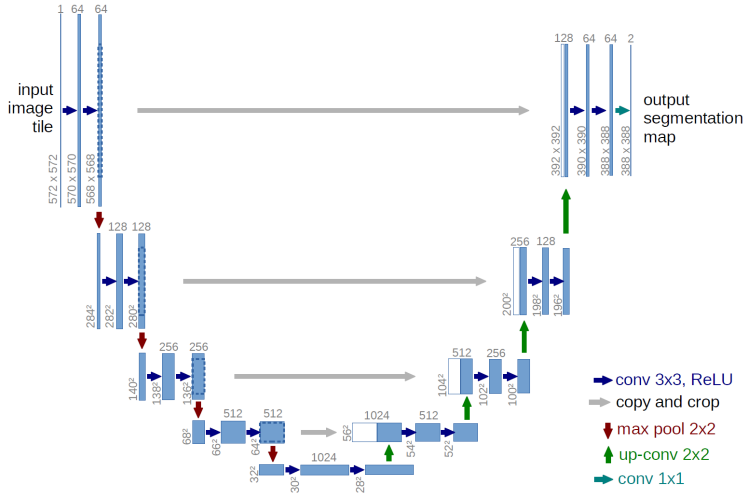


Figure 5: U-Net architecture with a 32×32 pixel image as the lowest resolution, each blue and white box represents a multi-channel and a copied feature map [Ronneberger et al. (2015)]

For DRUNet, we see that the network architecture comes from the dilated-residual U-Network (DRUNet) and it extends DnCNN. The network setup (Figure 6) involves a U-Net and ResNet [He et al. (2016)] with four scales, where each scale has a skip connection between a convolutional downscaling and upscaling layer.

Both denoisers can handle various noise levels using their models. What makes these denoisers very flexible is they can be trained on a different image dataset to the inverse problem, as long as the type of noise (e.g. Gaussian or Poisson) that is removed is equivalent across the training and testing datasets. In Section 4.1, we investigate this using different test images for the CT inverse problem.

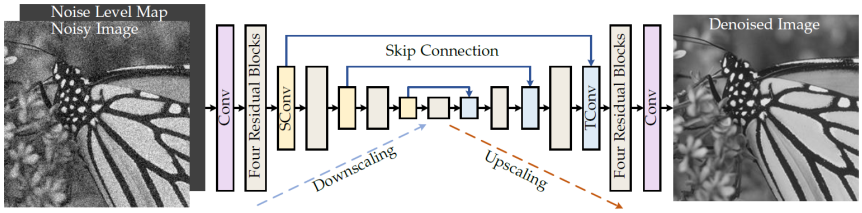


Figure 6: DRUNet architecture for the denoiser prior, taking noise level map inputs and combining U-Net and ResNet. SConv and TConv represent the strided convolution and the transposed convolution [Zhang et al. (2021)].

3.3 Fixed-point convergence theorems for non-expansive denoisers

The initial empirical success of PnP with different plug-in denoisers often lacked theoretical convergence results [Venkatakrishnan et al. (2013)]. Once the prior term (2.3) for the regularisation is replaced, usually denoisers are no longer minimising the explicit optimisation function such as the MAP estimate. The interpretation as an optimisation problem is lost without imposing restrictions on these denoisers.

In the literature, this is subject to ongoing research with papers exploring fixed-point convergence of PnP algorithms [Chan et al. (2016); Ryu et al. (2019)] for a fixed regularisation parameter λ and noise level σ. There is also ongoing research into the convergence of the PnP regularisation as a minimiser [Ebner and Haltmeier (2022)]. The latter focuses on the method of PnP, as $\lambda, \sigma \to 0$ for the inverse problem and whether we can retrieve the solution of a least-squares solution minimiser.

In this section, we look at the restrictions which allow us to achieve results for the former, fixed-point convergence. Here, we are interested in conditions under which the PnP algorithms reach a stationary fixed point in the limit the algorithm iterations go to infinity ($k \to \infty$). This corresponds to our PnP algorithms with PnP-PGD (3.3) where $\boldsymbol{x}^*$ is a fixed point if

$$\boldsymbol{x}^* = \mathcal{D}_\sigma(\boldsymbol{x}^* - \tau \nabla f(\boldsymbol{x}^*)). \tag{3.5}$$

Similarly, for PnP-ADMM (Algorithm 4) $(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{v}^*)$ is a fixed point if

$$\boldsymbol{x}^* = \mathbf{Prox}_{\frac{1}{\beta} f}(\boldsymbol{u}^* - \boldsymbol{v}^*), \quad \boldsymbol{u}^* = \mathcal{D}_\sigma(\boldsymbol{x}^* + \boldsymbol{v}^*), \quad \boldsymbol{v}^* = \boldsymbol{v}^* + \boldsymbol{x}^* - \boldsymbol{u}^*. \tag{3.6}$$

Recall that, these algorithms are balancing the goals of (1.2) with fitting the estimates to the measurement data while decreasing the overall noise of these estimates. This is a trade-off between stability and accuracy for the solution, as is always the case with regularisation methods.

3.3.1 Non-expansive denoisers

First, we define non-expansive and bounded denoisers, under the 2-norm, used for subsequent theoretical results.

Definition 3.1. *(Non-expansive)*
A mapping $\mathcal{D} : \mathbb{R}^n \to \mathbb{R}^n$ is non-expansive when

$$\|\mathcal{D}(\boldsymbol{x}) - \mathcal{D}(\boldsymbol{y})\|_2 \leq \|\boldsymbol{x} - \boldsymbol{y}\|_2, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n. \tag{3.7}$$

This is equivalent to having $\mathcal{D}$ as Lipschitz continuous with Lipschitz constant $L \leq 1$.

Note there is a stronger notion of non-expansiveness.

Remark 3.1. *(Firmly non-expansive)*
*A mapping $\mathcal{D}$ is **firmly** non-expansive when*

$$\|\mathcal{D}(\boldsymbol{x}) - \mathcal{D}(\boldsymbol{y})\|_2^2 + \|(Id - \mathcal{D})(\boldsymbol{x}) - (Id - \mathcal{D})(\boldsymbol{y})\|_2^2 \leq \|\boldsymbol{x} - \boldsymbol{y}\|_2^2, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n. \tag{3.8}$$

Having a denoiser to be non-expansive can give stability within a PnP algorithm and preserve the underlying structure of a quantity, as the denoiser removes noise.

Definition 3.2. *(Bounded denoiser)*
A bounded denoiser with parameter σ is a function $\mathcal{D}_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ such that $\forall \boldsymbol{x} \in \mathbb{R}^n = \mathcal{X}$ (input with noise)

$$\frac{||\mathcal{D}_\sigma(\boldsymbol{x}) - \boldsymbol{x}||_2^2}{n} \leq \sigma^2 C \tag{3.9}$$

for a constant C independent of n and σ. Note that, $\mathcal{D}_\sigma \to \mathcal{I}$ (the identity map) as $\sigma \to 0$.

With these definitions, we state the fixed point convergence theorem result for PnP-ADMM from Chan et al. (2016), which looks at bounded denoisers $\mathcal{D}_\sigma$ combating Gaussian noise with standard deviation σ.

Theorem 3.1. *(Fixed-point convergence for bounded denoisers, PnP-ADMM, Chan et al. (2016))*
If there is a bounded denoiser $\mathcal{D}_\sigma$ and the data-fidelity term $f(\boldsymbol{x})$ from (1.2), has bounded gradients, then the iterates of PnP-ADMM from Algorithm 4 have fixed-point convergence, i.e there exists $(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{v}^*)$ such that $|||\boldsymbol{x}^{(k)} - \boldsymbol{x}^*||_2, |||\boldsymbol{u}^{(k)} - \boldsymbol{u}^*||_2, |||\boldsymbol{v}^{(k)} - \boldsymbol{v}^*||_2 \to 0$ and $\boldsymbol{x}^{(k)} \to \boldsymbol{u}^{(k)}$ as $k \to \infty$.

A more general theorem proven in Ryu et al. (2019), which demonstrates the fixed point convergence of several PnP algorithms including PnP-PGD, PnP-FISTA and PnP-ADMM (Section 3.1).

Theorem 3.2. *(Fixed-point convergence of PnP-PGD, PnP-FISTA and PnP-ADMM, , Ryu et al. (2019))*
Assume the denoiser $\mathcal{D}_\sigma$ is non-expansive and f is convex. Then the PnP algorithms discussed in Section 3.1, with stepsize $\tau > 0$, have fixed points if f is differentiable with Lipschitz gradient L_f:

1. *For PnP-PGD and PnP-FISTA, $0 < \tau L_f < 2$ is required.*
2. *For PnP-ADMM, $\mathcal{D}_\sigma$ is firmly non-expansive (i.e (3.8) is satisfied) and $\tau > 0$ is required.*

All the fixed-point sequences for the algorithm iterates $\left(\boldsymbol{x}^{(k)}\right)_{k=1}^N$ satisfy $\sum_{k=1}^N \|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2 < \infty$ as $N \to \infty$.

We see that, under conditions such as (3.1) and (3.9), PnP algorithms have theoretical results to back up their numerical implementations. Recall that, these algorithms try to approximate the proximal operator with a denoiser which follows from a useful theorem proposed by Jean Moreau in 1965. This is where a non-expansive mapping is equivalent to a proximal operator of a convex function.

Theorem 3.3. *(Approximation for proximal operators of convex functions as denoisers, Moreau (1965), Corollary 10c))*
An operator, such as a denoiser, $\mathcal{D} : \mathbb{R}^n \to \mathbb{R}^n$ is the proximal operator of a proper, lower semi-continuous and convex function $\phi : \mathbb{R}^n \to \mathbb{R}$, i.e. $\mathcal{D} = \boldsymbol{Prox}_\phi$, if and only if:

1. *$\mathcal{D}$ is non-expansive*
2. *There exists a proper, lower semi-continuous and convex function h such that $\mathcal{D} \in \partial h$ (sub-differential).*

With this theorem, we can characterise continuous proximal operators as sub-gradients (gradients if f is continuous) of convex functions. The denoisers can be defined in this way such as with DnCNN and DRUNet.

In practice, imposing these conditions, in particular, the non-expansiveness (3.1) on the denoisers can severely degrade performance [Chan et al. (2016); Ryu et al. (2019)] and is often an unrealistic setup for many denoisers discussed in Section 3.2. Other considerations, including $\mathcal{D}_\sigma - \mathrm{Id}$ to be non-expansive instead of $\mathcal{D}_\sigma$, give more flexibility for the denoiser, but again it is hard to maintain the condition for a range of denoisers [Ryu et al. (2019)]. Also, these theorems are limited to convex data-fidelity terms f. The setup yields no convergence results for non-convex f [Ryu et al. (2019)], where such f can occur in some noise model setups of imaging inverse problems (e.g. Poisson or Laplacian noise). Finally, the convergence of iterates, namely Theorems 3.1 and 3.2, cannot be characterised as a form of a minimum or critical point of any functional such as the MAP estimate (2.3).

However, the denoisers we have discussed (Section 3.2) and many more used in PnP, have shown SOTA image restoration performance for many imaging inverse problems.

3.4 Gradient-step proximal denoisers

A recent alternative outlook on denoisers that does not enforce the non-expansive condition, directly on the denoiser, has been proposed in Hurault et al. (2022a, 2023a) called a gradient-step (GS) proximal denoiser. The objective is to have the plug-in denoiser minimise an optimisation functional that mimics (2.3) while maintaining state-of-the-art performance among different denoisers. The general premise of replacing the proximal operator with the denoiser still applies, however, the finer details are adapted to help make the overall setup more flexible.

Before discussing the setup details, we note the main theoretical change for the denoiser compared to Section 3.3. The approach uses an adaptation of Theorem 3.3, where the non-expansive condition for the denoisers is dropped in exchange for approximating the proximal operator for a non-convex function.

Theorem 3.4. *(Approximation for proximal operators of non-convex functions as denoisers, Gribonval and Nikolova (2020))*
An operator $\mathcal{D} : \mathbb{R}^n \to \mathbb{R}^n$ is the proximal operator of a proper, l.s.c and non-convex function $\phi : \mathbb{R}^n \to \mathbb{R}$, i.e. $\mathcal{D} = \boldsymbol{Prox}_\phi$, if and only if there exists a proper, l.s.c and convex function h where $\mathcal{D} \in \partial h$.

This subtle change helps give more flexibility in the fixed-point convergence results for GS denoisers while maintaining state-of-the-art performance as will be discussed in Section 3.4.2.

3.4.1 Gradient-step setup

The denoiser $\mathcal{D}_\sigma$ in (3.2) is amended from the direct use of a plug-in denoiser to one with a gradient descent step

$$\mathcal{D}_\sigma = \mathrm{Id} - \nabla g_\sigma \tag{3.10}$$

where $g_\sigma : \mathbb{R}^n \to \mathbb{R}$ is parametrised by a neural network (NN). Note that g_σ does not have to exactly be equivalent to a neural network (NN) and in Hurault et al. (2022a) the denoiser is defined as

$$g_\sigma(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - N_\sigma(\boldsymbol{x})\|_2^2, \tag{3.11}$$

$$\mathcal{D}_\sigma(\boldsymbol{x}) = N_\sigma(\boldsymbol{x}) + J_{N_\sigma}^T(\boldsymbol{x})(\boldsymbol{x} - N_\sigma(\boldsymbol{x})) \tag{3.12}$$

where $N_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ is a neural network and $J_{N_\sigma}(\boldsymbol{x})$ is the Jacobian of N_σ evaluated at $\boldsymbol{x}$. This formulation from Hurault et al. (2022a) allows for **any differentiable** NN architecture to be parametrised for N_σ. For example, DnCNN and DRUNet architectures (Section 3.2) can be used where the non-differentiable rectified linear unit (ReLU) activation function is replaced with a smooth differentiable exponential linear unit (ELU) activation function. For example, the element-wise ELU for $\boldsymbol{x} = (x_1, \dots, x_n)^T \in \mathbb{R}^n$ is

$$ELU(\boldsymbol{x})_i = \begin{cases} x_i, & x_i \geq 0 \\ \mathrm{e}^{x_i} - 1, & x_i < 0 \end{cases} \tag{3.13}$$

for $i = 1, \dots, n$.

The denoiser (3.10) is trained for Gaussian noise to minimise the mean squared loss (MSE) loss functions for $\mathcal{D}_\sigma$ and g_σ

$$L(\mathcal{D}_\sigma) = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}(\boldsymbol{x}),\gamma\sim\mathcal{N}(\mathbf{0},\sigma^2\mathrm{Id})}[\|\mathcal{D}_\sigma(\boldsymbol{x}+\gamma) - \boldsymbol{x}\|_2^2], \tag{3.14}$$

$$L(g_\sigma) = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}(\boldsymbol{x}),\gamma\sim\mathcal{N}(\mathbf{0},\sigma^2\mathrm{Id})}[\|g_\sigma(\boldsymbol{x}+\gamma) - \gamma\|_2^2]. \tag{3.15}$$

Note that (3.14) is related to an estimate, similar to the MAP estimate, called the minimum mean square error (MMSE) estimate. The MMSE estimate selects the mean of the posterior distribution $\mathbb{P}(\boldsymbol{x}|\boldsymbol{y})$

$$\mathcal{D}_\sigma^{\mathrm{MMSE}}(\boldsymbol{y}) = \mathbb{E}[X|Y=\boldsymbol{y}] = \int_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{x}\mathbb{P}(\boldsymbol{x}|\boldsymbol{y})d\boldsymbol{x} \tag{3.16}$$

where X and Y are random variables that represent $\boldsymbol{x}$ and $\boldsymbol{y}$. The MMSE estimate is also the optimal Bayes estimator for L^2 loss

$$\mathcal{D}_\sigma^{\mathrm{MMSE}}(Y) = \mathbb{E}_{(X,Y)}[\|\mathcal{D}_\sigma(Y) - X\|_2^2] \tag{3.17}$$

and it is also related to the *score* $\nabla \log(p_\sigma)$ via Tweedie's identity [Efron (2011)]

$$\mathcal{D}_\sigma^{\mathrm{MMSE}} = \mathrm{Id} + \sigma^2 \nabla p_\sigma. \tag{3.18}$$

where $p_\sigma \equiv p_Y$ is the distribution of observations $\boldsymbol{y}$ with Gaussian noise σ from (1.1). Therefore, using the loss function for (3.15) coupled with definition of the GS denoiser (3.4), the optimal solution for g_σ^* is of the form $\nabla g_\sigma^* = -\sigma^2 \nabla \log(p_\sigma)$ i.e $g_\sigma^* = -\sigma^2 \log(p_\sigma) + M$, for a constant $M \in \mathbb{R}$. Like the MAP estimate, generally the MMSE estimate cannot be obtained explicitly, however, (3.16) and (3.18) have been used in frameworks called *denoising score matching*, that have proven fruitful in image generation and other research fields [Habring and Holler (2023)].

Overall, looking at the structure of (3.10), we see that the formation of PnP-PGD (3.3), PnP-FISTA and PnP-ADMM (Algorithm 4) follows in the same way as before, with the proximal operator replaced by the denoiser in the corresponding steps of the algorithms. However, there are now important distinctions between PnP with deep denoisers (DnCNN, DRUNet) and GS-PnP with the same deep denoisers (GS-DnCNN, GS-DRUNet). This is due to the change in the construction of $\mathcal{D}_\sigma$ (3.10) with (3.11) - (3.12).

3.4.2 Fixed-point convergence results

Similar to the theorems in Section 3.3, there are fixed-point convergence theorems for this gradient-step proximal denoiser when applied to various variable-splitting algorithms such as PGD and

ADMM. As mentioned before, Theorem 3.3 is adapted with Theorem 3.4 to help approximate the proximal operator as a plug-in denoiser, by not enforcing the non-expansive condition.

We adapt (2.4) to include g_σ

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + \lambda g_\sigma(\boldsymbol{x}) \tag{3.19}$$

where using GS denoiser for g_σ when minimising the functional F is the same as using an explicit regularisation prior for (2.4). This leads to a global minimisation problem, whereas, the PnP algorithms in Section 3.3 do not have this characteristic.

We state the fixed-point convergence theorem for GS-PnP-PGD, where there is a similar theorem for GS-PnP-ADMM (same fixed-point convergence rate) [Hurault et al. (2022b)] as well, which is omitted.

Theorem 3.5. *(Fixed-point convergence of GS-PnP-PGD, Hurault et al. (2022b))*
Let f be taken from (2.3) and be differentiable with L_f Lipschitz gradient and bounded below. Assume ∇g_σ, taken from (3.10), has a Lipschitz constant $L < 1$. Also, assume that F is bounded from below and satisfies the KŁ property (2.7). Then for $\lambda L_f < \frac{L+2}{L+1}$, the iterates $\left(\boldsymbol{x}^{(k)}\right)_{k=1}^{N}$, as $N \to \infty$, of PnP-PGD (3.3) with stepsize $\tau = 1$:

1. *$F(\boldsymbol{x}^{(k)})$ is non-increasing and converges $< \infty$*
2. *All the fixed-point sequences of $\boldsymbol{x}^{(k)}$ satisfy $\sum_{k=1}^{\infty} \|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2 < \infty$ and converges to 0 at a rate $\min_{k<K\in\mathbb{N}} \|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2 = \mathcal{O}(\frac{1}{\sqrt{K}})$*
3. *If f is non-negative and g_σ is coercive, then the iterates $\left(\boldsymbol{x}^{(k)}\right)_{k=1}^{N}$ converge towards a critical point of F.*

The combination of Theorem 3.4, KŁ property for F and conditions on the Lipschitz constants of f and ∇g_σ allow the GS denoisers to achieve SOTA image restoration performance while also having convergence guarantees. Note that, unlike in Section 3.3, the GS denoiser setup also works for non-convex data-fidelity f.

In Section 4, we aim to show the performance of GS-DRUNet against the other denoisers we have considered, numerically testing the fixed-point convergence ($\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2$) across the denoisers.

3.5 PnP advantages and disadvantages

Now that we have seen the general outlook on PnP, with its setup for various algorithms and their theoretical convergence results, we now consider the advantages and disadvantages of PnP in the context of data-driven regularisation methods in general.

Advantages

A major upside for PnP is the flexibility to use different off-the-shelf denoisers that achieve state-of-the-art performance for various image recovery tasks in inverse problems [Kamilov et al. (2017); Zhang et al. (2021)]. The notion of replacing the proximal operator in variable-splitting algorithms (3.2) allows for many denoisers to be used and is modular, with the main structure of the algorithms being kept intact.

Training the deep denoisers, especially for Gaussian denoising, is often stable and does not require a large number of training images to achieve good denoising performance. For example, DnCNN was trained only on 400 images [Zhang et al. (2017)]. Other deep priors such as generative adversarial networks (GANs) or variational autoencoders (VAEs) can be much tougher to train and less flexible than PnP.

Another advantage of the PnP framework is the reusability across different datasets and inverse problem tasks. Although, for different measurement noise models like Poisson, the algorithms would have to be adapted with extensions of the proximal operator using the Bregman distance [Hurault et al. (2023b)], the denoisers themselves would not have to be retrained. This is further discussed in Section 5.2.

The framework is also unsupervised, where the training process for the denoisers does not need to be trained on a specific problem with ground-truth data. For a deep denoiser, the only requirement is for the neural network to be trained to remove a specific type of noise (e.g. Gaussian) from a dataset of clean images. This is appealing when the inverse problem does not have significant ground-truth data or cannot simulate data for its problem. For other data-driven methods such as algorithm unrolling, the training process can be dataset-specific, where if the methods are tested on a different inverse problem task with similarities in the noise but uses a different dataset, the performance is of lower quality compared to the PnP framework. In the case of algorithm unrolling, the training process depends on the inverse problem itself and the forward operator A, where the neural network is specifically trained. Therefore the neural network would have to be retrained when used on another inverse problem with a different forward operator, whereas PnP denoisers would not have to be retrained.

Lastly, subject to certain conditions as we saw in Sections 3.3 and 3.4.2, there are convergence results of PnP algorithms that give justification of SOTA performance in a range of inverse problem tasks [Zhang et al. (2021); Hurault et al. (2022b)].

Disadvantages

Usually, plug-in denoisers $\mathcal{D}_\sigma$, especially deep denoisers, are not variational, i.e. there is no explicit relation on the regulariser $\lambda g(\boldsymbol{x})$ in (1.1). If this is the case, it is difficult to find what the denoiser is trying to minimise overall. A possible remedy regarding consensus-based optimisation for PnP is discussed in Section 5.2. In general, it is difficult to exactly replicate the proximal operator $\mathbf{Prox}_g$ or the gradient descent step $\mathrm{Id} - \nabla g_\sigma$ for the optimisation algorithms with $\mathcal{D}_\sigma$, without enforcing conditions such as $\mathcal{D}_\sigma$ being non-expansive (Theorems 3.3 and 3.4) or the KŁ property on ∇g_σ. As discussed in Section 3.3, it is hard to maintain these conditions in practice.

PnP priors with CNNs have become very popular recently with many different types of CNNs used in the denoising processes. These denoisers can handle a range of different noise levels when applied to clean images for training, but when it comes to noise levels, outside of the training range, the denoisers can struggle to cope. There is also a trade-off usually between the number of network layers and the effectiveness of a denoiser's performance.

When training the NN denoisers, the computational time usually takes anything from several hours to days. For instance, the DRUNet denoiser used in Zhang et al. (2021) took four days to train. So there is a substantial amount of effort required to build and train the denoiser models such that they

perform well against other data-driven regularisation methods. However, as discussed, once trained they can be widely used.

Finally, the computation of PnP algorithms can be very costly compared to other data-driven methods such as the algorithm unrolling. Usually, hundreds of iterations of the PnP algorithms are required to achieve similar performance compared to very few iterations of algorithm unrolling (e.g. < 50 iterations).

4 Numerical experiments

In this section, we conduct various numerical experiments using PnP methods discussed in Section 3. We explore using several denoisers and implement them with PnP-PGD and PnP-ADMM algorithms. We also aim to numerically test the fixed-point convergence theorem results for imaging inverse problems and investigate interesting ideas for PnP.

There are two main experiments to be performed:

1. CT inverse problem for different types of images
2. Investigation of the denoiser in the PnP-PGD algorithm - what noise does the denoiser remove compared to the measurement noise in the inverse problem?

Each experiment uses the same denoisers discussed in Sections 3.2 and 3.4 including TV as a baseline with BM3D, DnCNN, DRUNet and GS-DRUNet as a mix of classical and deep denoisers. Note that, all the deep learning denoisers (DnCNN, DRUNet, GS-DRUNet) are pre-trained on various large-scale image datasets of natural images (see Figure 7) including ImageNet [Deng et al. (2009)] and the Berkeley segmentation dataset [CBSD, Martin et al. (2001)]. This follows the practice of recent PnP papers that use these same datasets for training the denoisers [Zhang et al. (2021); Hurault et al. (2022a)].

Figure 7: Examples of natural images used for training [Martin et al. (2001); Deng et al. (2009)]

All the experiments use the pre-trained deep denoiser network weights from the `deepinv` open-source Python library [Tachella et al. (2023)] and our code for these experiments is available as a GitHub repository, see Appendix A. Also, all of these experiments used the performance metric, peak signal-to-noise ratio (PSNR), which is measured in decibels (dB). It is comprised of the ratio between the maximum pixel intensity of the ground truth signal $\boldsymbol{x}^*$ ($MAX_I(\boldsymbol{x}^*) \in [0, 255]$ or $[0, 1]$) and the pixel intensity of the noise in the reconstructed signal k^{th} approximate $\boldsymbol{x}^{(k)}$ through the mean squared error (MSE)

$$\text{PSNR} = 10 \log \left(\frac{MAX_I(\boldsymbol{x}^*)}{MSE(\boldsymbol{x}^*, \boldsymbol{x}^{(k)})} \right). \tag{4.1}$$

Values for PSNR include anything from $0 - 100$ dB, with the ground truth having the highest PSNR. In our case, depending on the noise degradation of an image, values between $25 - 35$ dB can indicate reasonable reconstructions of the ground truth. In general, the higher the PSNR, the more similar the reconstruction is towards the ground truth.

4.1 Computed tomography (CT)

For this experiment, we considered the CT inverse problem (Section 2.3). We recall that the forward operator A is the Radon transform and corresponding measurements with Gaussian noise are observed for $\boldsymbol{y}$ as the sinogram. We investigated two types of images for the experiments, MNIST digits and Shepp-Logan phantom. Each image's observed sinogram is subject to additive Gaussian noise, $\sigma \times \mathcal{N}(0, 1)$, where $\sigma = 0.1$ (see Appendix A.2). This value of σ is in line with typical noise levels for CT [Diwakar and Kumar (2018)] and represents 10% of the maximum of the ground truth intensity that is affected (e.g. 25.5/255 for an 8-bit image).

We used FBP (analytical inverse of the Radon transform, Section 2.3) as a baseline comparison against the five denoisers. Also, each PnP algorithm was executed for a maximum of 200 iterations (λ, PnP-PGD and PnP-ADMM algorithm parameter setups in Appendices A.2 and A.3). The step-size τ of PnP-PGD was bounded by $\frac{1}{L_f}$, where $L_f = \|A^T A\|_2 = \lambda_{\max}(A^T A)$ (the largest eigenvalue of $A^T A$, Hessian matrix of f). This was estimated via the power method (see Appendix B).

Note that the results of these experiments are not intended to represent a complete performance evaluation of PnP, but rather to demonstrate the practical application of PnP.

4.1.1 MNIST digits

For MNIST, the images are 28×28 pixels and the Radon transform is taken for 60 different angles with 40 detector elements. The respective forward operator is the discretised Radon transform $A \in \mathbb{R}^{m \times n}$ with $\boldsymbol{x} \in \mathbb{R}^n$ where $m = 60 \times 40 = 2400$ and $n = 28^2 = 784$. We seek the ground truth image $\boldsymbol{x}^*$ from the noisy sinogram $\boldsymbol{y}$, see Figure 8.

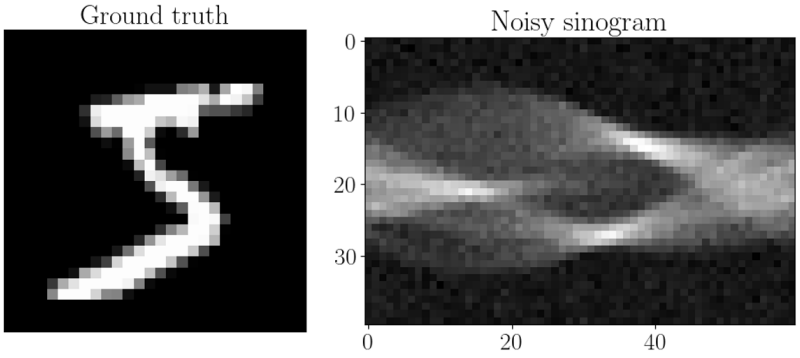


Figure 8: Ground truth $\boldsymbol{x}^*$ (MNIST digit 5) and noisy sinogram $\boldsymbol{y}$

For PnP-PGD, we have the results of the various reconstructions in Figure 9. FBP yielded a noisy reconstruction as expected, with several denoisers (TV, DnCNN, DRUNet, GS-DRUNet) all producing better PSNR values and achieving SOTA reconstruction performance. The best reconstructions of DnCNN and TV respectively, managed to capture the finer details of the MNIST digit image, while not losing details compared to BM3D and DRUNet. As this image was piecewise constant, TV thrived on this structure and performed very well, giving a baseline reconstruction compared to the other denoisers.

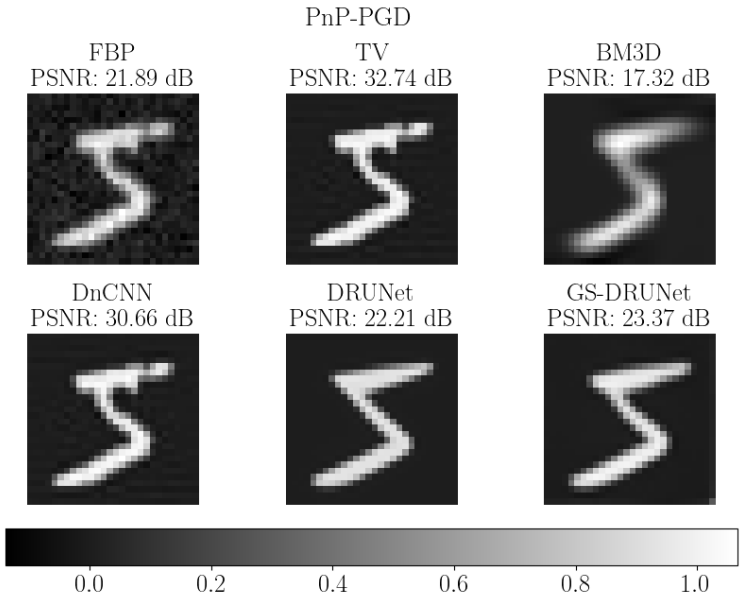


Figure 9: MNIST digit 5 PnP-PGD reconstructions with FBP and five different denoisers

Looking at the progression over the algorithm iterations k for the PSNR and fixed-point convergence in Figure 10, we see that all five denoisers improve their PSNR and there is a significant decrease between algorithm iterates. GS-DRUNet empirically satisfies the fixed-point convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$, following the results of Theorem 3.5, alongside all the other denoisers. DRUNet had the best fixed-point convergence rate. However, we see that both GS-DRUNet and BM3D exhibited oscillatory behaviour as the k increases.

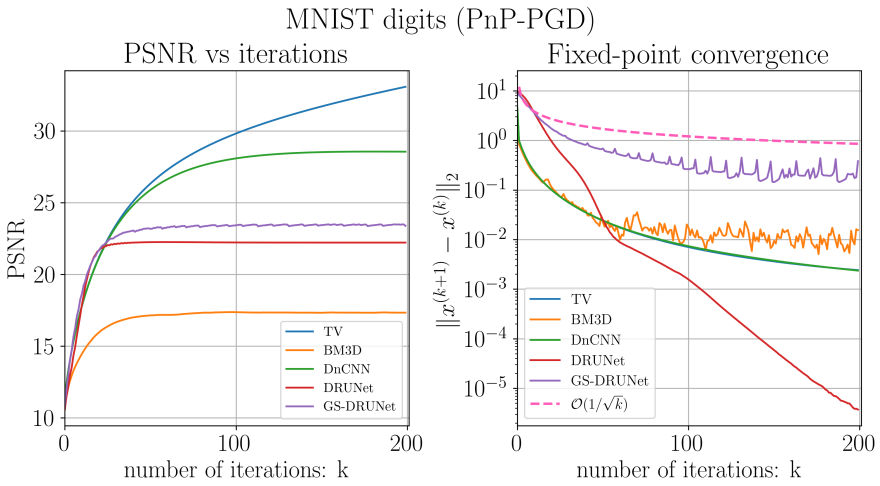


Figure 10: PnP-PGD: PSNR and $\|x^{(k+1)} - x^{(k)}\|_2$ vs number of iterations

Similarly for the results of PnP-ADMM in Figures 11 and 12, the reconstructions and the performance over time have the same trend as for PnP-PGD. This time there was little difference among all the denoisers, with GS-DRUNet marginally having the best PSNR. All denoisers managed to capture the details of the ground truth to a reasonable level. Note that, the number of iterations k was restricted to 50, due to PSNR performance levelling off for the denoisers after a small number of iterations. In Figure 12, all denoisers including GS-DRUNet, again followed the fixed-point convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$.

There were not any restrictions on making the denoisers non-expansive, but the performance across the two algorithms showed interesting and practical results for PnP. Overall, there was an alignment with the fixed-point convergence theory outlined in Sections 3.3 and 3.4.2.

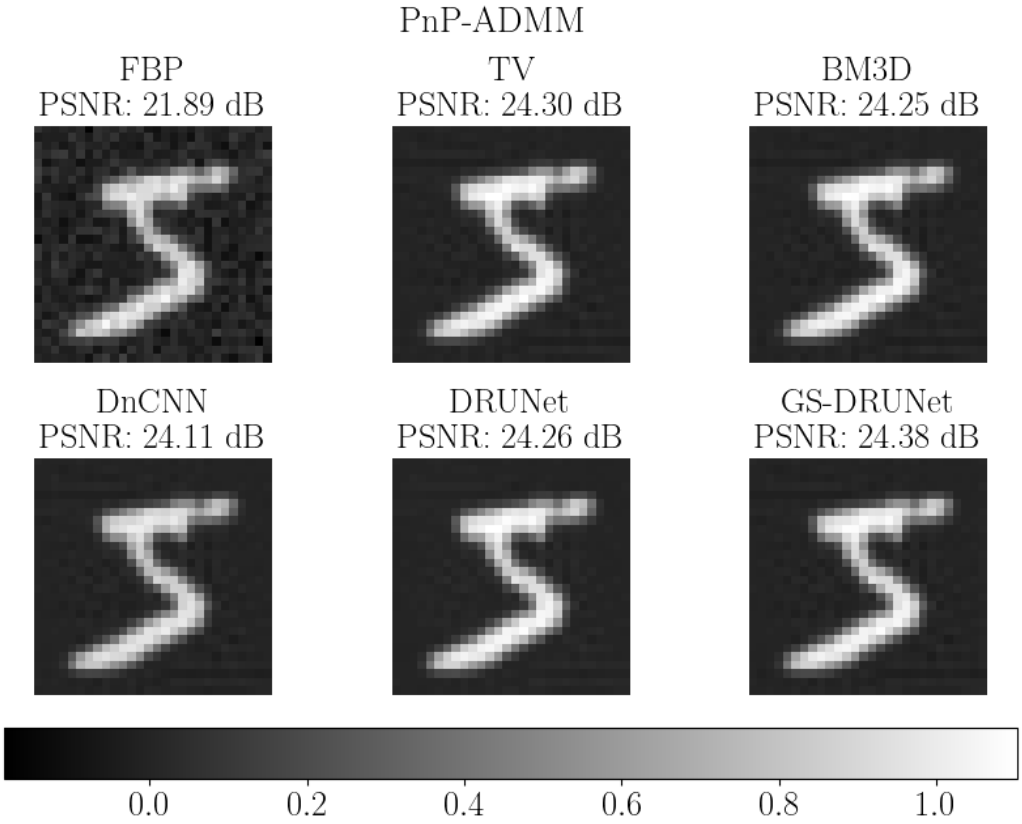


Figure 11: MNIST digit 5 PnP-ADMM reconstructions with FBP and five different denoisers

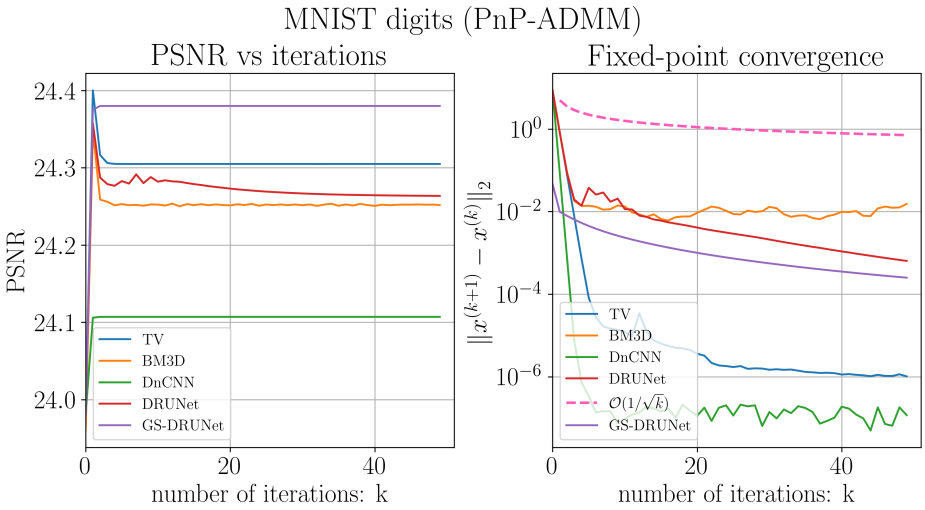


Figure 12: PnP-ADMM: PSNR and $\|x^{(k+1)} - x^{(k)}\|_2$ vs number of iterations

4.1.2 Shepp-Logan Phantom

For the Shepp-Logan Phantom, the image is 84×84 pixels and the Radon transform is again taken across 60 different angles with 114 detector elements. The dimensions of the forward operator $A \in \mathbb{R}^{m \times n}$ and $\boldsymbol{x} \in \mathbb{R}^n$ are adjusted, where $m = 60 \times 119 = 7140$ and $n = 84^2 = 7056$. We seek the ground truth image $\boldsymbol{x}^*$ from the noisy sinogram $\boldsymbol{y}$, see Figure 13.

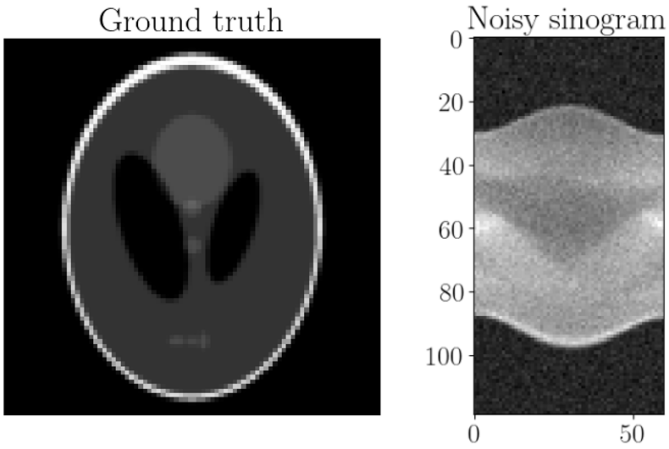


Figure 13: Ground truth $\boldsymbol{x}^*$ and noisy sinogram $\boldsymbol{y}$

For PnP-PGD, we see all the reconstructions in Figure 14. Again, FBP had a very noisy reconstruction and overall the denoisers produced SOTA reconstructions of the Phantom. Both DRUNet and GS-DRUNet have lost some finer details of the Phantom but still produced a clear image. Unlike the MNIST digit result (Figure 9), BM3D achieved the best results, followed by DnCNN and TV.

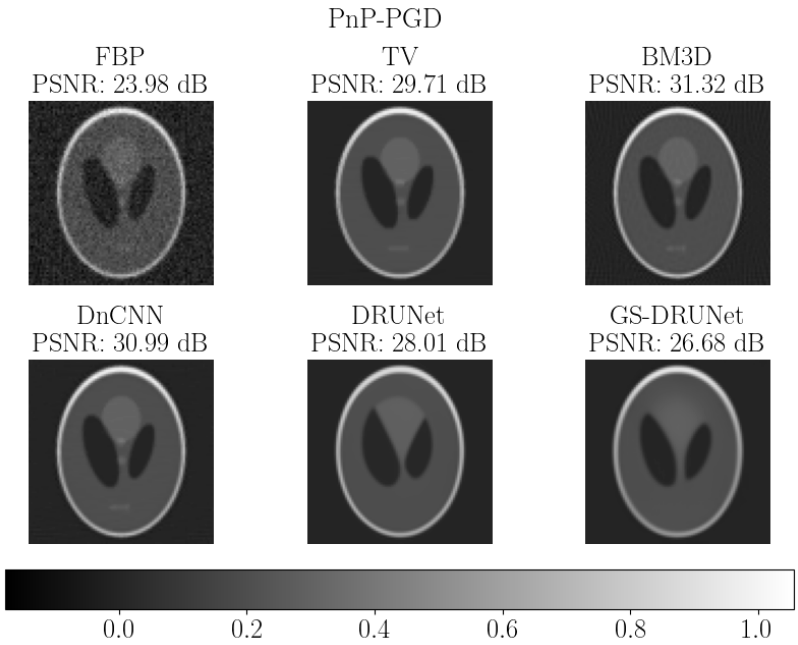


Figure 14: Shepp-Logan Phantom PnP-PGD reconstructions with FBP and five denoisers

Investigating the PSNR and fixed-point convergence in Figure 15, we see that all denoisers satisfied the convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$ (Theorem 3.5) with GS-DRUNet having the fastest fixed-point convergence rate. There was no oscillatory behaviour, as seen in Figure 10, with all five denoisers exhibiting similar improvements over the algorithm iterations (TV, BM3D, and DnCNN had identical fixed-point convergence rates).

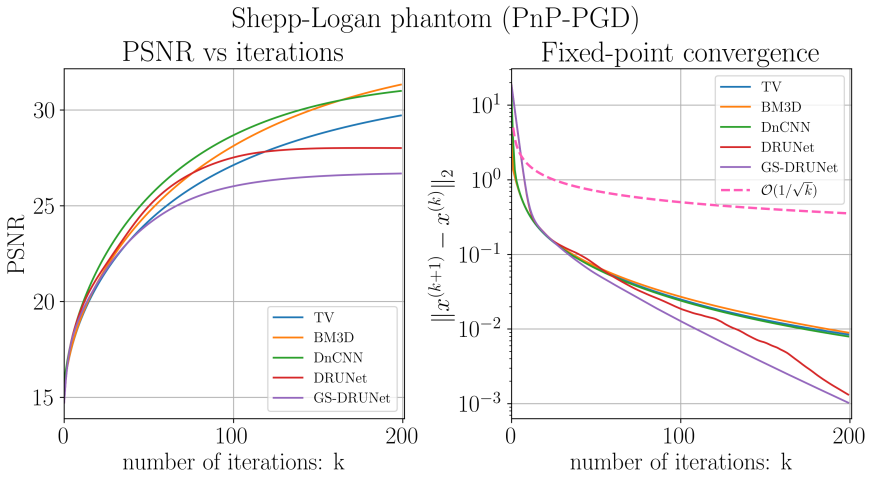


Figure 15: PnP-PGD: PSNR and $\|x^{(k+1)} - x^{(k)}\|_2$ vs number of iterations

Considering the results for PnP-ADMM, in Figures 16 and 17, the reconstructions again follow similar trend to PnP-PGD. Apart from TV, which had small imperfections in its reconstruction, the other denoisers were very close in PSNR, with GS-DRUNet having the best reconstruction. Similar to MNIST results for PnP-ADMM (Figure 12), the algorithm iterations k were restricted to 50 and there was an odd initial decrease in overall quality (see Figure 17) until the PSNR levelled off for all the denoisers. Note that, this may have been due to the implementation of the PnP-ADMM algorithm for this image (for MNIST and natural images this effect did not occur, see Figure 12 and Appendix A). GS-DRUNet exhibited oscillatory behaviour, similar to the results in Figure 10, which may indicate instability as the PnP-ADMM algorithm progressed. Following the results of all the previous numerical experiments, the fixed-point convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$ was satisfied by all denoisers again in Figure 17, with DnCNN having the fastest fixed-point convergence rate among the five denoisers.

In general, the results in Sections 4.1.1 and 4.1.2 yielded SOTA image reconstruction performance across the two PnP algorithms. Even though no conditions for the denoisers being non-expansive were enforced, the fixed-point convergence results were empirically satisfied, particularly for the GS-DRUNet denoiser (Section 3.4.2).

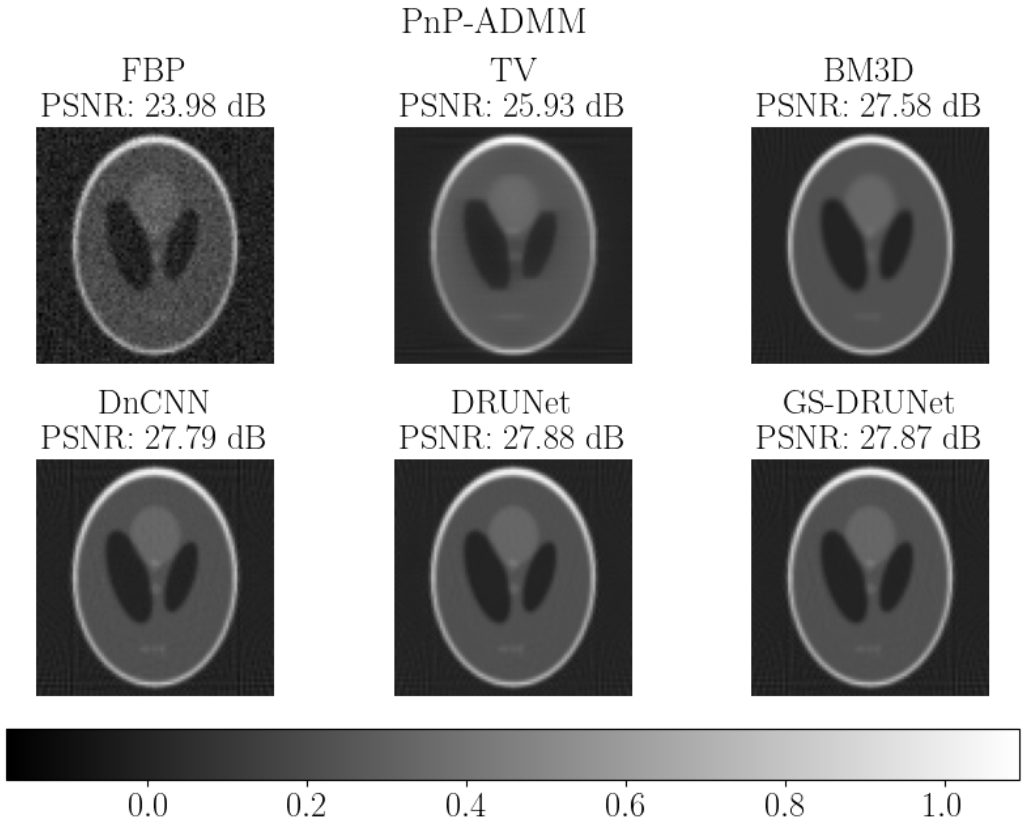


Figure 16: Shepp-Logan Phantom PnP-ADMM reconstructions with FBP and five denoisers

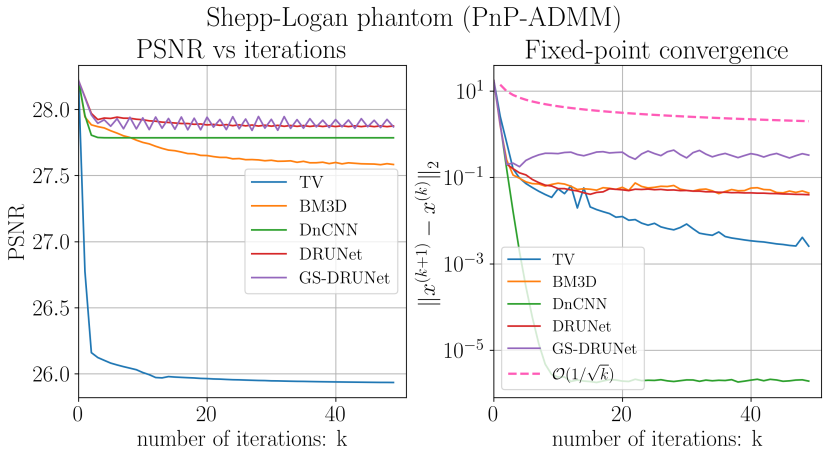


Figure 17: PnP-ADMM: PSNR and $\|x^{(k+1)} - x^{(k)}\|_2$ vs number of iterations

4.2 What does a denoiser actually do?

In this experiment, we investigated the noise distributions between algorithm iterates for the different denoisers with the same PnP-PGD setup (Appendix A.3) for the Shepp-Logan Phantom (Section 4.1.2). As outlined in Sections 2.2 and 3.1, in the variable-splitting algorithms there is a cycle of inverting via the data-fidelity term f and performing the regularising/denoising step with g. We denoted these in the results as the inverted $\boldsymbol{x}^{(k)}$ and denoised $\mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$ iterates respectively and studied the "residual noise" between the two, denoted by $\boldsymbol{x}^{(k)} - \mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$. To visualise this, there are image plots (an example in Figure 18) of $\boldsymbol{x}^{(k)}$ and $\mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$ aiming to show the effect of different denoisers on the k^{th} iterate, while $\boldsymbol{x}^{(k)} - \mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$ shows the difference across the two operations for an algorithm iterate, with a key depicting their respective colour bars.

The idea, as presented in Section 1.2, is that based on the noise assumption of the inverse problem, in this case, Gaussian, one might expect the residual noise between the inverted iterate and the denoised iterate in the algorithm to resemble the same type of noise as the inverse problem. However, the priors of g are often non-linear denoisers, such as neural networks, which may exhibit different types of probability distributions for the residual noise compared to the measurement noise assumption for the inverse problem. Also, due to the setup of PnP, with the replacement of the proximal operator via the MAP estimate for Gaussian noise (3.1), the denoisers are trained to remove artificial Gaussian noise, so we hope to see if this residual noise itself is Gaussian.

To test for this, we conducted the Kolmogorov-Smirnov (KS) hypothesis test [Kolmogorov (1933)] on three of the denoisers (TV, BM3D, DRUNet), with Gaussian measurement noise from the CT inverse problem (Figure 13). Both DnCNN and GS-DRUNet exhibited similar results to DRUNet so were omitted (full residual noise code for all denoisers is available in Appendix A). KS is a non-parametric, goodness of fit test that assesses whether a given sample probability distribution is Gaussian, comparing the maximum distances between the sample and Gaussian probability distributions. In our case, the sample distribution is the residual noise's probability distribution $\boldsymbol{x}^{(k)} - \mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$, which allowed us to see if there was a link to the measurement noise assumption of the inverse problem. Note that, we had $n = 7056$ samples (number of pixels) for the residual noise, with sample mean μ and standard deviation γ, at each k^{th} iterate of the PnP-PGD algorithm.

We conducted the KS hypothesis test with a significant level of 5% and the null hypothesis (H_0) as

$$H_0\text{: } (\boldsymbol{x}^{(k)} - \mathcal{D}_\sigma(\boldsymbol{x}^{(k)})) \sim \mathcal{N}(\mu, \gamma^2), \tag{4.2}$$

where for each test for the algorithm iterates, a KS test statistic value and p-value were produced to compare against the significance level ($p > 0.05$, accept H_0 and $p < 0.05$, reject H_0). Note that, we used the kernel density estimation to estimate the sample probability distributions of the respective residual noise for the k^{th} iterates. For each denoiser, we studied each algorithm iterate, but our results focused primarily on the first and last iterates ($k = 1$ and 200).

Firstly for TV, we see the results for the algorithm iterates in Figures 18 and 19, with the effect of applying the denoiser to each $\boldsymbol{x}^{(k)}$. We see the noise removal in the respective $\boldsymbol{x}^{(k)} - \mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$ and how for $k = 1$, the changes from $\boldsymbol{x}^{(k)}$ and $\mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$ are more pronounced than for $k = 200$. The corresponding residual noise probability density functions (PDFs) in Figure 20, showed that both residual noise PDFs do not follow a Gaussian distribution and therefore rejected the null hypothesis (4.2), $p < 0.05$. Although for $k = 1$, there was a more familiar Gaussian shape, the overall distribution of the pixel intensities from Figure 18 was not Gaussian. This was the same for all the other iterates, indicating there was not a clear link between the residual noise of the TV denoiser and the measurement noise.

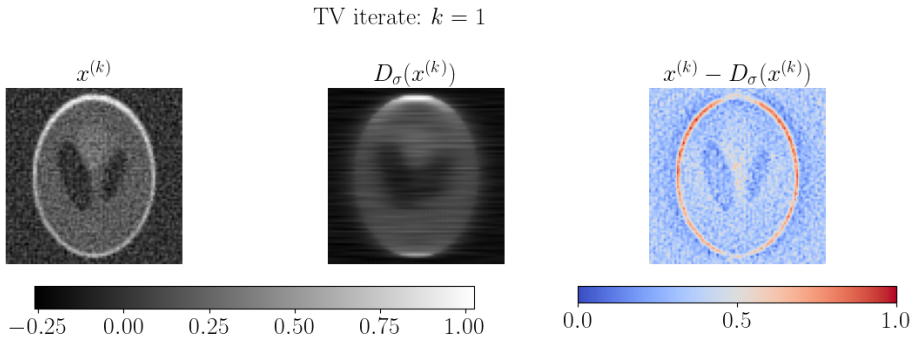


Figure 18: TV results of inverted, denoised iterates and residual noise for $k = 1$

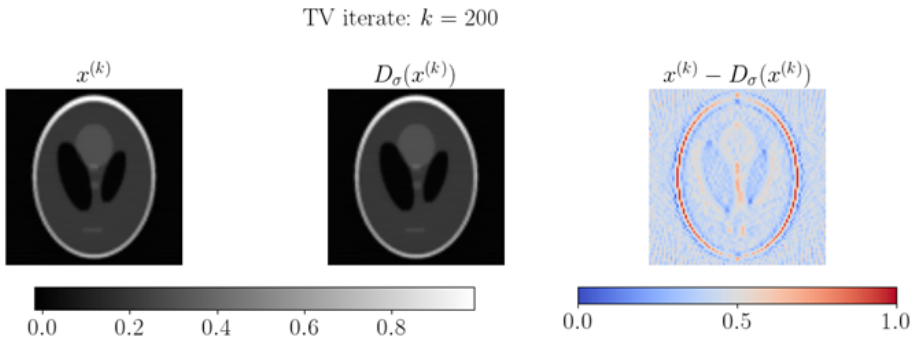
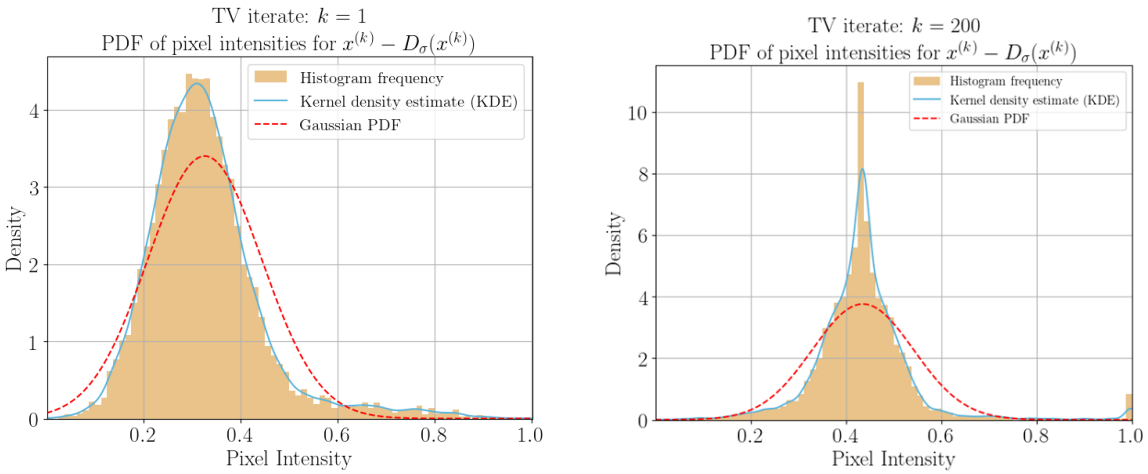


Figure 19: TV results of inverted, denoised iterates and residual noise for $k = 200$



(a) $k = 1, \mu = 0.33, \gamma = 0.12$ $(p < 0.05)$ **(b)** $k = 200, \mu = 0.44, \gamma = 0.11$ $(p < 0.05)$

Figure 20: TV PDFs of the pixel intensities for the residual noise $\boldsymbol{x}^{(k)} - \mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$ with their reference Gaussian PDFs

Next for BM3D, we see the algorithm iterates results in Figures 21 and 22, where for $k = 1$ applying BM3D, actively removed the Gaussian noise from $\boldsymbol{x}^{(k)}$ and the corresponding residual noise plot indicates this. This was confirmed in Figure 23a, with the PDF of the pixel intensities closely resembling a Gaussian with the same mean and standard deviation as the samples. Therefore, there was not enough evidence to reject the null hypothesis (4.2), $p > 0.05$. However, for all the other iterates including $k = 200$, the null hypothesis was rejected. For $k = 200$, we see in Figure 23b, that while the PDF takes the general Gaussian shape, the peak density in the PDF was too high for a Gaussian. This characteristic was present across most of the BM3D algorithm iterates after $k = 1$.

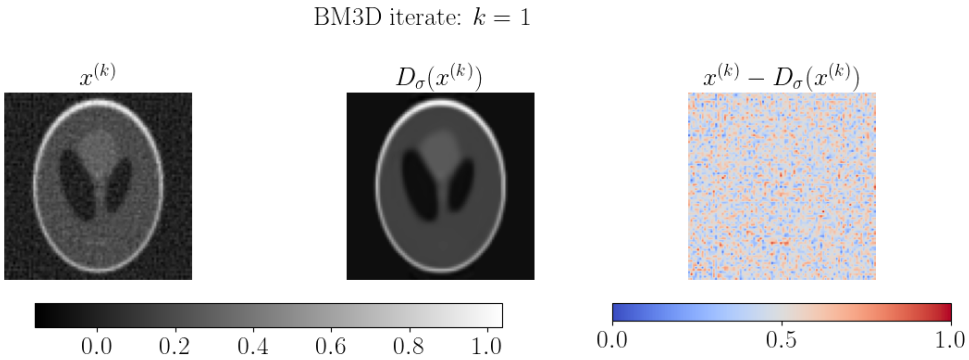


Figure 21: BM3D results of inverted, denoised iterates and residual noise for $k = 1$

Finally, for DRUNet we see the results in Figures 24 and 25. For $k = 1$, there was a smoothing process between $\boldsymbol{x}^{(k)}$ and $\mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$, with the general structure of the Phantom remaining intact in the residual noise. Over time, the reconstruction becomes more similar to the ground truth, resulting in

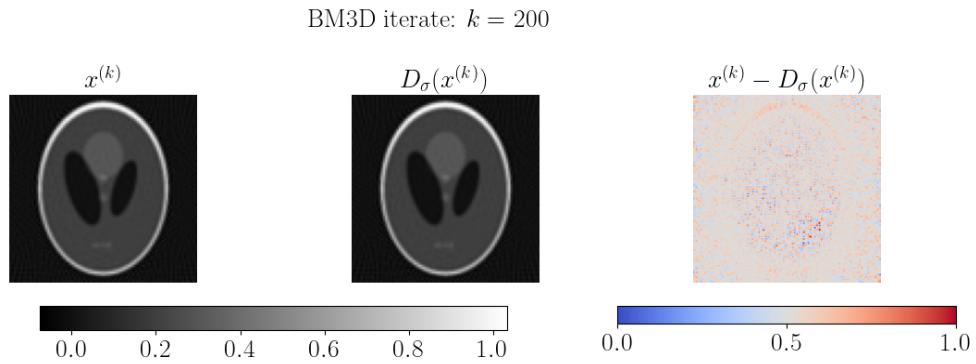
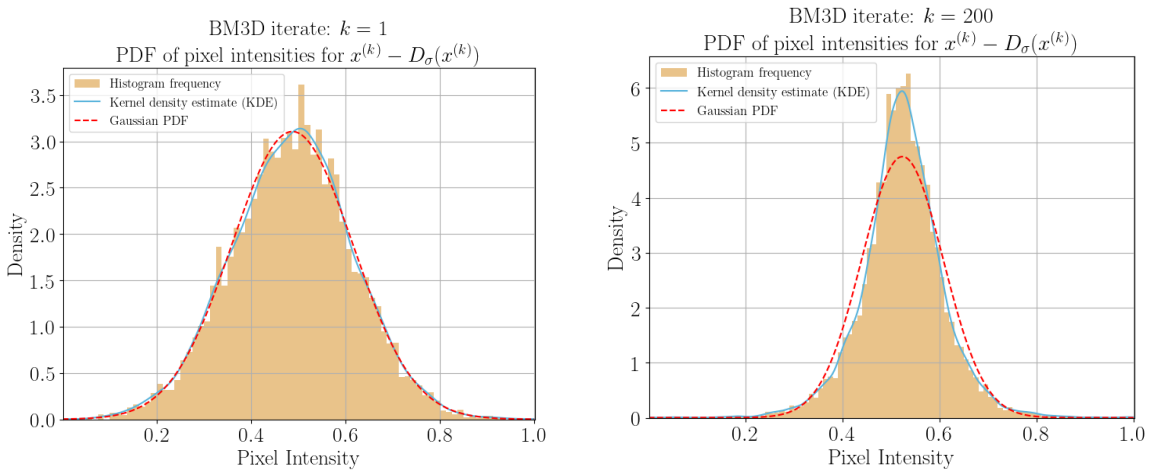


Figure 22: BM3D results of inverted, denoised iterates and residual noise for $k = 200$



(a) $k = 1, \mu = 0.42, \gamma = 0.11$ $(p > 0.05)$ **(b)** $k = 200, \mu = 0.52, \gamma = 0.08$ $(p < 0.05)$

Figure 23: BM3D of the pixel intensities for the residual noise $\boldsymbol{x}^{(k)} - \mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$ with their reference Gaussian PDFs

the $k = 200$ iterate, where some finer details are lost in the Phantom. The corresponding residual noise PDFs in Figure 26 are similar to TV's results (Figure 20) with both iterates rejecting the null hypothesis (4.2), $p < 0.05$. In parallel to TV for $k = 1$, there was a similar Gaussian shape, but again the peak density was too high for a Gaussian. This theme was present among most of the algorithm iterates for DRUNet and also had similar results for GS-DRUNet.

In general, the results suggest there was not a strong relationship between the residual noise of the denoisers and the Gaussian measurement noise of the CT inverse problem. Also, although the denoisers are trained to remove artificial Gaussian noise, there is no representation for this with the residual noise between algorithm iterates. There were some exceptions, such as with BM3D (Figure 23a) and the general Gaussian shapes from TV and DRUNet (Figures 20a and 26a). However, more needs to be investigated regarding the interpretation of the denoiser removing noise for other assumptions of measurement noise (e.g. Poisson). This is discussed in Section 5.2.

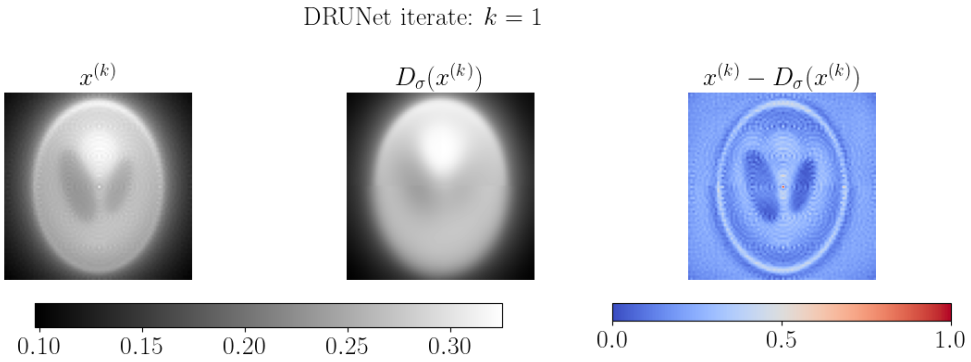


Figure 24: DRUNet results of inverted, denoised iterates and residual noise for $k = 1$

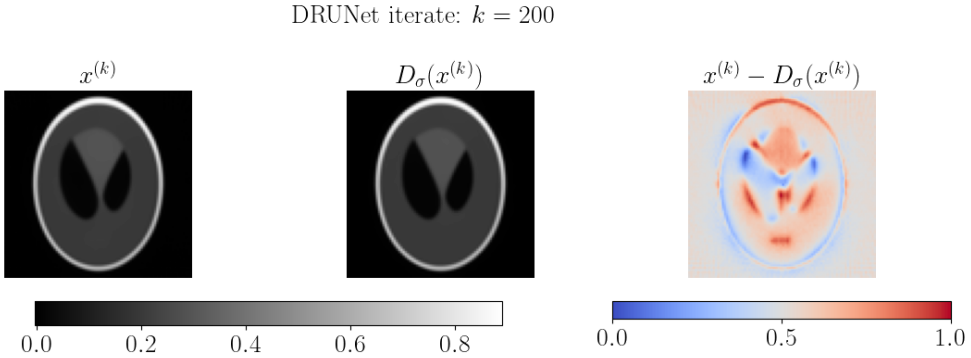
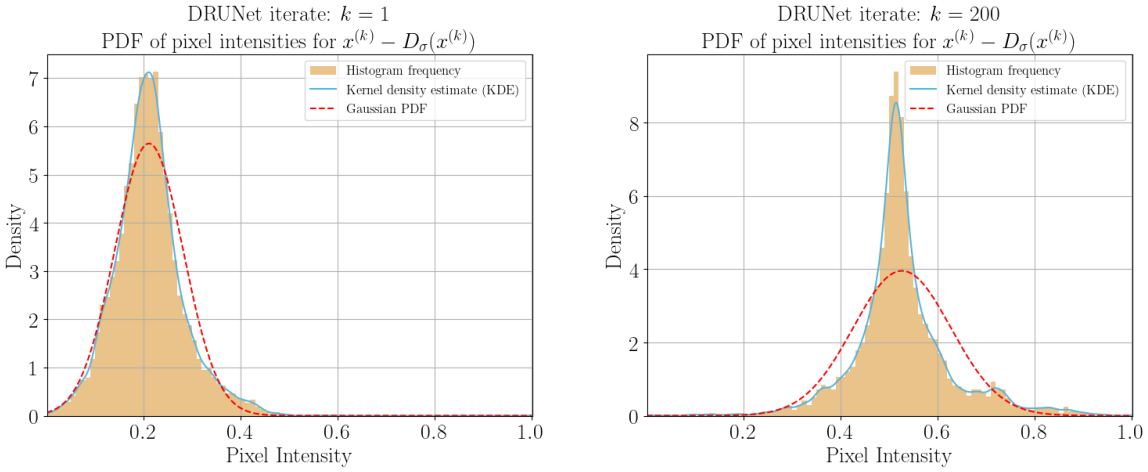


Figure 25: DRUNet results of inverted, denoised iterates and residual noise for $k = 200$



(a) $k = 1, \mu = 0.21, \gamma = 0.07$ $(p < 0.05)$

(b) $k = 200, \mu = 0.53, \gamma = 0.10$ $(p < 0.05)$

Figure 26: DRUNet PDFs of the pixel intensities for the residual noise $\boldsymbol{x}^{(k)} - \mathcal{D}_\sigma(\boldsymbol{x}^{(k)})$ with their reference Gaussian PDFs

5 Summary and future work

In this final section, we summarise the report and discuss possible research directions for the forthcoming PhD.

5.1 Summary

In this report, we have seen an overview of the general landscape for plug-and-play regularisation for imaging inverse problems. There has been a substantial emphasis on the setup of variable-splitting algorithms with different denoisers and their fixed-point convergence theorems (Sections 2 and 3). Interesting numerical experiments were conducted on the CT inverse problem, where we empirically justified these fixed-point convergence theorems. Also, we investigated the type of noise that the denoisers have removed in the algorithms, with their different residual noise distributions (Section 4).

5.2 Future work

There are various research directions to explore regarding PnP and data-driven regularisation to build the foundations of a PhD programme in the field. As a starting point, we would like to continue exploring the noise assumptions of inverse problems (other than Gaussian) and how different types of noises affect the overall PnP regularisation structure and performance. This theme, alongside two other themes that are more exploratory, are discussed in greater depth below.

1. **The effect of noise assumptions in the inverse problem and PnP algorithms on the denoiser's performance.**
   In the setup of PnP, we interpreted the proximal operators as the MAP estimate for Gaussian noise and replaced it with Gaussian plug-in denoisers (3.2). This interpretation is independent of the measurement noise assumption and only focuses on the characteristic of the proximal operator itself. This interpretation could imply that the denoiser's residual noise (the noise removed for the inverted algorithm iterate) is that of Gaussian denoising. Yet, in our experiments, in Section 4.2, there was no clear link of a Gaussian in the residual noise distributions of these denoisers or towards the Gaussian measurement noise of the CT inverse problem.

   An immediate question is to continue this investigation and explore different measurement noise assumptions (e.g. Poisson or Laplacian noise) and see if the residual noise distributions still do not show this Gaussian denoising. From this, we can assess if the Gaussian plug-in denoiser assumption is important. If we violate this assumption, does the behaviour of the PnP algorithm change? Will we converge to a sub-optimal solution or even have non-convergence? Is it worthwhile to train a different type of denoiser such as a Poisson denoiser?

   For the Poisson measurement noise, recent work has been done by Hurault et al. (2023b), where they modified PnP algorithms by extending the use of the proximal operator with the Bregman distance $\mathcal{B}$. This distance is defined for a function $M : \Omega \to \mathbb{R}$, where M is a continuously-differentiable, strictly convex function defined on a convex set Ω and $\langle \cdot, \cdot \rangle$ is an inner product on $\mathbb{R}$ such that

   $$\mathcal{B}(p, q) = M(p) - M(q) - \langle \nabla M(q), p - q \rangle, \quad p, q \in \Omega. \tag{5.1}$$

   This modification has worked well for Gaussian denoisers with Poisson measurement noise assumptions, but one can explore whether it is useful to train a Poisson denoiser over a Gaussian one.

2. **Extensions and alternative setups of PnP**.
   The general PnP landscape (Section 3) is subject to ongoing research and it would be interesting to examine alternative versions and possible extensions of PnP. These versions include having PnP as a consensus-based optimisation framework [Buzzard et al. (2018)] and as a hybrid prior setup [Cascarano et al. (2022)].

   For the former, this explores the case where there are more than two proximal operator steps in an algorithm and PnP then forms a system that helps all these steps converge to related fixed points as part of a consensus. This framework has been seen as a possible bridge between moving from the well-defined optimisation estimates such as MAP (2.3) and MMSE (3.16), to one where denoisers, without explicit knowledge of the inverse problem, can continue to operate in a mathematical framework. The hope is that the mathematical interpretability can remain intact, regardless of the different denoisers one would use. For example, this might give more understanding as to how non-expansive denoisers (Section 3.3) would perform in a given situation.

   For the hybrid PnP setup, Cascarano et al. (2022) proposed the use of an internal prior governed by TV and an external prior governed by a CNN-based denoiser. They found a promising use in CT and were able to match standard PnP and other data-driven regularisation methods for image reconstruction. One could explore how this setup can be used in other inverse problems and when is it best to use this approach over the standard PnP method.

3. **Applications of PnP in industry**
   As discussed in Section 3.5, PnP is very promising and useful in the area of data-driven regularisation for imaging inverse problems. In particular, for areas where there is no supervised data (no known ground truths to the corresponding measurements). One could possibly use PnP in these applications, where other data-driven methods such as algorithm unrolling that requires supervised data, might be limited. Even in situations, where supervised data is available, often these other methods rely on a forward operator and can be computationally expensive to train to achieve SOTA results. However, for PnP, this could be bypassed in its algorithms and offers a potential alternative in such situations.

   A potential application of this is for positron emission tomography (PET), where ground truth data is limited. Current research [Reader et al. (2021)] uses methods such as algorithm unrolling and variational encoders, which can achieve fast reconstructions. However, to improve the quality of reconstruction, there is potential to use PnP over such methods. One could try to integrate PnP via the primal-dual hybrid gradient (PDHG) algorithm [Chambolle and Pock (2016)], which uses proximal operators and has had success for PET [Ehrhardt et al. (2019)].

In conclusion, there are numerous intriguing research avenues to explore regarding PnP and data-driven regularisation as a whole, many of which can build a strong foundation for a forthcoming PhD.

References

Aharon, M., Elad, M., and Bruckstein, A. (2006). K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322.

Attouch, H., Bolte, J., and Svaiter, B. F. (2011). Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. *Mathematical Programming, Series A*, 137(1):91–124. Publisher: Springer.

Beck, A. and Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

Boyd, S. (2010). Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Buzzard, G. T., Chan, S. H., Sreehari, S., and Bouman, C. A. (2018). Plug-and-Play Unplugged: Optimization Free Reconstruction using Consensus Equilibrium. arXiv:1705.08983.

Cascarano, P., Piccolomini, E. L., Morotti, E., and Sebastiani, A. (2022). Plug-and-Play gradient-based denoisers applied to CT image enhancement. *Applied Mathematics and Computation*, 422:126967.

Chambolle, A. and Pock, T. (2016). An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319.

Chan, S. H., Wang, X., and Elgendy, O. A. (2016). Plug-and-Play ADMM for Image Restoration: Fixed Point Convergence and Applications. arXiv:1605.01710.

Danielyan, A., Katkovnik, V., and Egiazarian, K. (2012). BM3D Frames and Variational Image Deblurring. *IEEE Transactions on Image Processing*, 21(4):1715–1728. arXiv:1106.6180.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. ISSN: 1063-6919.

Diwakar, M. and Kumar, M. (2018). A review on CT image noise and its denoising. *Biomedical Signal Processing and Control*, 42:73–88.

Ebner, A. and Haltmeier, M. (2022). Plug-and-Play image reconstruction is a convergent regularization method. arXiv:2212.06881.

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

Ehrhardt, M. J., Markiewicz, P., and Schönlieb, C.-B. (2019). Faster PET reconstruction with non-smooth priors by randomization and preconditioning. *Physics in Medicine & Biology*, 64(22):225019.

Gribonval, R. and Nikolova, M. (2020). A characterization of proximity operators. arXiv:1807.04014.

Habring, A. and Holler, M. (2023). Neural-network-based regularization methods for inverse problems in imaging. arXiv:2312.14849.

Hadamard, J. (1902). Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*, pages 49–52.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA. IEEE.

Hurault, S., Chambolle, A., Leclaire, A., and Papadakis, N. (2023a). A relaxed proximal gradient descent algorithm for convergent plug-and-play with proximal denoiser. arXiv:2301.13731.

Hurault, S., Kamilov, U., Leclaire, A., and Papadakis, N. (2023b). Convergent Bregman Plug-and-Play Image Restoration for Poisson Inverse Problems. arXiv:2306.03466 [cs, eess, math].

Hurault, S., Leclaire, A., and Papadakis, N. (2022a). Gradient Step Denoiser for convergent Plug-and-Play. arXiv:2110.03220.

Hurault, S., Leclaire, A., and Papadakis, N. (2022b). Proximal Denoiser for Convergent Plug-and-Play Optimization with Nonconvex Regularization. arXiv:2201.13256.

Kamilov, U. S., Mansour, H., and Wohlberg, B. (2017). A Plug-and-Play Priors Approach for Solving Nonlinear Imaging Inverse Problems. *IEEE Signal Processing Letters*, 24(12):1872–1876.

Kolmogorov (1933). Sulla determinazione empirica di una legge didistribuzione. *Giorn Dell'inst Ital Degli Att*, 4:89–91.

Lunz, S., Öktem, O., and Schönlieb, C.-B. (2019). Adversarial Regularizers in Inverse Problems. arXiv:1805.11572.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423 vol.2.

Monga, V., Li, Y., and Eldar, Y. C. (2020). Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing. arXiv:1912.10557.

Moreau, J. (1965). Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 79:273–299.

Radon, J. (1917). 1.1 über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten. *Classic papers in modern diagnostic radiology*, 5(21):124. Publisher: Springer Berlin–Heidelberg–New York.

Reader, A. J., Corda, G., Mehranian, A., Costa-Luis, C. d., Ellis, S., and Schnabel, J. A. (2021). Deep Learning for PET Image Reconstruction. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 5(1):1–25. Conference Name: IEEE Transactions on Radiation and Plasma Medical Sciences.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268.

Ryu, E. K., Liu, J., Wang, S., Chen, X., Wang, Z., and Yin, W. (2019). Plug-and-Play Methods Provably Converge with Properly Trained Denoisers. arXiv:1905.05406.

Scherzer, O., Grasmair, M., Grossauer, H., Haltmeier, M., and Lenzen, F. (2009). *Variational methods in imaging*, volume 167. Springer.

Tachella, J., Chen, D., Hurault, S., and Terris, M. (2023). Deepinverse: A deep learning framework for inverse problems in imaging.

Venkatakrishnan, S. V., Bouman, C. A., and Wohlberg, B. (2013). Plug-and-Play priors for model based reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 945–948, Austin, TX, USA. IEEE.

Yang, X. and Fei, B. (2013). Multiscale segmentation of the skull in MR images for MRI-based attenuation correction of combined MR/PET. *Journal of the American Medical Informatics Association*, 20(6):1037–1045.

Zhang, K., Li, Y., Zuo, W., Zhang, L., Van Gool, L., and Timofte, R. (2021). Plug-and-Play Image Restoration with Deep Denoiser Prior. arXiv:2008.13751.

Zhang, K., Zuo, W., Gu, S., and Zhang, L. (2017). Learning Deep CNN Denoiser Prior for Image Restoration. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2808–2817, Honolulu, HI. IEEE.

A Code availability and algorithm setups

The code for this report is available in a GitHub repository `https://github.com/Amino21786/TfRPnP` called `TfRPnP` that reproduces all the plots and results in this report, including the Python files with the construction of methods. In the folder, there are a `README.md` and a `dependencies.yml`, which outline what this code folder entails and a guide to download the required Python libraries.

There are three main Python files for the CT inverse problem, the first two of which consist of the PnP-PGD and PnP-ADMM for MNIST digits and Shepp-Logan phantom respectively (`MNISTCT.ipynb`,`PhantomCT.ipynb`). These contain the results used in Sections 4.1.1 and 4.1.2. The third is a trial file (not used in Section 4) experimenting with natural images for the same algorithms (`NaturalCT.ipynb`). These three files have the same structure.

As highlighted in Section 4.1.2 with the results seen in Figure 17, PnP-ADMM algorithm behaved strangely for the Shepp-Logan Phantom (decrease in the initial PSNR as the algorithm progressed). Amendments to try and fix this are ongoing with the latest available results available in the Github repository.

There is an extension of the Phantom CT file that contains the results in Section 4.2 (at the end of `PhantomCT.ipynb`), which produces the relevant plots for all five denoisers. There are also plotting and algorithms `.py` files that are used in the aforementioned `.ipynb` files. Full details are explained in the `README.md` file.

A.1 Algorithm parameter descriptions

Algorithm parameter descriptions used for Algorithms 1, 2, 3 and 4.

Parameter	Description
L	Lipschitz constant for ∇f, equal to $\|A^TA\|_2 = \lambda_{\max}(A^TA)$
τ	Stepsize for PGD (Algorithms 1 and 3.3), bounded by L
λ	Regularisation parameter
β	Penalty parameter for ADMM (Algorithms 3 and 4)
σ	Measurement noise level for CT inverse problem

Table 1: Description of algorithm parameters

A.2 PnP-PGD algorithm parameters

The base algorithm parameters setup for the results in Section 4.1.1 and 4.1.2. Note that the σ values are the same for PnP-PGD and PnP-ADMM.

Dataset	L	σ
MNIST digit 5 (28×28 pixels)	1621	0.1
Shepp-Logan Phantom (84×84 pixels)	4867	0.1

Table 2: Model setups for PnP-PGD (same σ as PnP-ADMM)

Denoiser/prior	λ	$\frac{1}{\tau}$
TV	$(10^{-4}, 10^{-4})$	(L, L)
BM3D	$(10^{-5}, 10^{-4})$	(L, L)
DnCNN	$(10^{-5}, 10^{-4})$	$(L, 4000)$
DRUNet	$(10^{-5}, 10^{-4})$	$(750, 2900)$
GS-DRUNet	$(10^{-5}, 10^{-4})$	$(800, 2900)$

Table 3: Model parameters for PnP-PGD (MNIST, Phantom)

A.3 PnP-ADMM algorithm parameters

The base algorithm parameters setup for the results in Section 4.1.1 and 4.1.2.

Denoiser/prior	λ	β
TV	$(10^{-3}, 10^{-1})$	$(10, 6)$
BM3D	$(10^{-3}, 1)$	$(10, 10)$
DnCNN	$(10^{-3}, 1)$	$(100, 10)$
DRUNet	$(10^{-3}, 1)$	$(10, 10)$
GS-DRUNet	$(10^{-3}, 1)$	$(100, 10)$

Table 4: Model parameters for PnP-ADMM (MNIST, Phantom)

B Power method

Power method algorithm for estimating the largest eigenvalue of a square matrix, $H = A^T A \in \mathbb{R}^{n \times n}$, used in Section 4.1.

Algorithm 5: Power method for H

Initial guess $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$ such that $\|\boldsymbol{x}^{(0)}\|_2 = 1$, choose $tol > 0$
For $k = 0, 1, 2, \ldots,$
$\boldsymbol{m}^{(k+1)} = H\boldsymbol{x}^{(k)}$
$\boldsymbol{x}^{(k+1)} = \frac{\boldsymbol{m}^{(k+1)}}{\|\boldsymbol{m}^{(k+1)}\|_2}$
Eigenvalue estimate: $\lambda^{(k)} = \left(\boldsymbol{x}^{(k)}\right)^T \boldsymbol{m}^{(k+1)}$
Residual: $\boldsymbol{r}^k = \boldsymbol{m}^{(k+1)} - \lambda^{(k)}\boldsymbol{x}^{(k)}$
Repeat until $\|\boldsymbol{r}^{(k)}\|_2 < tol$
Largest eigenvalue estimate: $\lambda^{(k)}$