UNIVERSITY OF BATH

Efficient Bilevel Optimization for Imaging

Interdisciplinary Research Project

Students:	Jennifer Power
	Matthew Pawley
Supervisors:	Silvia Gazzola
	Matthias Ehrhardt

April 26, 2021

Contents

1 Introduction

The focus of this project is on "Efficient Bilevel Optimisation for Imaging". Image reconstruction has applications in a wide variety of areas, including in medical imaging and astronomical imaging. An image reconstruction problem takes the form of a linear inverse problem $Bu = y$. Given the measured data (y) and the forward operator (B), the aim is to obtain the reconstructed image (u). However, such problems are often ill-posed, and the naive solution $u = B^{-1}y$ is extremely sensitive to noise in the measured data. Instead, regularisation methods are used to suppress noise components in the reconstruction, as well as impose additional *a priori* information about the unknown solution, e.g. smoothness or sparsity. The inverse problem can therefore be formulated as an variational optimisation problem of the form:

$$\min_u \mathcal{D}(Bu, y) + \alpha\mathcal{R}(u) \tag{1}$$

where $\mathcal{D}$ is a data fidelity term (which encourages the reconstruction to fit the data and is chosen appropriately for the noise model of the data), and $\mathcal{R}$ is the regularisation term. The regularisation parameter α controls the trade-off between these two terms. This leads to the question: how should the regularisation parameter be chosen? Qualitatively speaking, the optimal parameter is the parameter such that the reconstruction closely matches the true solution. In many practical problems, we are interested in optimising many parameters, not just one. For example, for magnetic resonance imaging (MRI), taking measurements is time-intensive, expensive, and results in patient discomfort, which can lead to movement and artefacts in the image. Therefore it is of interest to determine the the optimal sparse MRI sampling pattern, so that we can take as few measurements as possible without sacrificing the quality of the reconstruction (Sherry et al. 2020). The potentially large number of parameters renders procedures such as grid search infeasible, due to the curse of dimensionality. Instead, the optimisation is typically performed using gradient-based algorithms such as gradient descent (GD) (Chambolle and Pock 2016).

A natural, data-driven approach for learning the parameters of a variational model is bilevel optimisation (Sherry et al. 2020; Ehrhardt and Roberts 2020). The bilevel approach exploits the hierarchical nature of the underlying problem: there is an upper level problem (parameter selection) and a lower level problem (which may be of the form (1)) nested within it. Given a training set of paired data and corresponding ground truth images, bilevel optimisation determines the optimal parameters which minimise the averaged reconstruction error on the training set.

The goal of this project is to improve the efficiency of existing bilevel optimisation algorithms used for parameter selection in imaging problems. Our main focus lies with the gradient of the objective function of the upper level problem, which poses two particular difficulties.

First, computing the upper level gradient involves inverting the Hessian matrix of the lower level objective function, which is typically very large. Therefore, in practice, this inversion is performed approximately, by solving a linear system using an iterative method. For example, Sherry et al. (2020) use the conjugate gradient method. Section 3 will investigate more efficient ways of solving the linear system. In particular, we propose using recycling Krylov methods to speed up the convergence of the linear system. The second difficulty is that the gradient of the upper level objective is not computed exactly, because it depends on the lower level solution and the inverse of the Hessian, both of which are computed inexactly. Section 4 will explore how inexact gradient-methods can be used to achieve convergence while minimising computational effort. The proposed methods will be illustrated and tested using an example problem, detailed in Section 2. The problem - learning the total variation regularisation parameter for (1D) signal processing - is a simplified version of the problem considered by Sherry et al. (2020). Section 5 concludes with discussion and suggestions for future work.

2 Problem Formulation

2.1 Bilevel optimisation for variational model parameter selection

As explained in Section 1, variational regularisation techniques are commonly used for image reconstruction tasks, and the variational model may involve a large number of parameters, e.g. the MRI sampling pattern (Sherry et al. 2020) or the total variation smoothing parameter (Chambolle and Pock 2016), which will be chosen using bilevel optimisation. Suppose we have access to a labelled training set $\{(y_i, u_i^\star)\}_{i=1}^N$ data y_i and the corresponding ground truths $u_i^\star$. The optimal model parameters $\theta \in \Theta$ are obtained by solving the bilevel problem

$$\min_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} L_{u_i^\star}(\hat{u}_i(\theta)) \qquad \text{subject to} \qquad \hat{u}_i(\theta) = \arg\min_{u\in\mathcal{U}} E_i(u;\theta). \tag{2}$$

The lower level objective function $E_i(u;\theta)$ may be of the form (1) and will generally depend on the data y_i. The upper level objective comprises cost functions $L_{u_i^\star}(\hat{u}_i(\theta))$ which measure the discrepancy between the lower level solution $\hat{u}_i(\theta)$ and the ground truth $u_i^\star$. The goal is to find the parameter θ that generates the best reconstructions. For the sake of notational simplicity, in the remainder of this section we take $N = 1$ and omit the dependence on i.

2.2 Solving the lower level problem

The lower level problem is concerned with finding the optimal reconstruction of the data y for a variational regularisation model with a given parameter θ:

$$\hat{u}(\theta) = \arg\min_{u\in\mathcal{U}} E(u;\theta). \tag{3}$$

We seek to minimise this function using a gradient-based optimisation technique. We choose a simple first-order, gradient descent method, which is known to perform well on a wide class of high-dimensional optimisation problems (Chambolle and Pock 2016). In order to control the accuracy of the lower level solution, we impose some assumptions on the smoothness and convexity of the lower level objective.

Definition 2.1 (M-smooth)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is M-smooth if it is differentiable and its derivative is Lipschitz continuous with constant $M \geq 0$, i.e. for all $x, y \in \mathbb{R}^n$,

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq M \|x - y\|_2 .$$

Definition 2.2 (μ-convex)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is μ-convex ($\mu > 0$) if $f - \frac{\mu}{2} \|\cdot\|^2$ is convex.

We assume that E is M-smooth and μ-convex. Under these assumptions, the gradient descent (GD) algorithm

$$u^{(k+1)} = u^{(k)} - \tau_k \nabla E(u^{(k)}; \theta)$$

with fixed step-size $\tau_k = 2/(M+\mu)$ converges linearly to the unique minimiser $\hat{u}$ of E (Ehrhardt and Roberts 2020):

$$\left\|u^{(k)} - \hat{u}\right\|_2^2 \leq \left(1 - \frac{\mu}{M}\right)^{2k} \left\|u^{(0)} - \hat{u}\right\|_2^2 .$$

By performing sufficiently many GD iterations, the lower level problem can be solved to an arbitrary level of accuracy. Of course, in practice the algorithm must be terminated after finitely many iterations, so the lower level solution is only ever computed approximately.

2.3 Challenges in solving the upper level problem

The upper level problem selects the optimal parameters for the variational model:

$$\theta^\star = \arg\min_{\theta \in \Theta} L_{u^\star}(\hat{u}(\theta))$$

where

$$L_{u^\star}(\hat{u}(\theta)) = \frac{1}{2} \|\hat{u}(\theta) - u^\star\|_2^2 . \tag{4}$$

The gradient of (4) is given by (Sherry et al. 2020)

$$\nabla_\theta L_{u^\star}(\hat{u}(\theta)) = -D_{\theta,u} E(\hat{u}(\theta))[D_u^2 E(\hat{u}(\theta))]^{-1} \nabla_{\hat{u}(\theta)} L_{u^\star}(\hat{u}(\theta))^\top . \tag{5}$$

The upper level problem is inherently more complicated to solve than the lower level problem and presents several difficulties. First, the global Lipschitz constant - if it exists - is unknown, so an adaptive GD step size determined by a two-way backtracking scheme (Truong and Nguyen

2020) is used. Second, the inversion of the Hessian matrix $D_u^2 E(\hat{u}(\theta))$ is prohibitively expensive even for moderately sized problems, e.g. reconstructing a 256×256 image would involve (repeatedly) inverting a $65,536 \times 65,536$ matrix. Instead, the associated linear system is solved using iterative methods. Third, the gradient is not known exactly, since it depends on the lower level solution and the inverse of the Hessian, both of which are only computed approximately. The viability of gradient-based methods in this setting is therefore unclear.

The gradient (5) will be the focal point for the remainder of this report. Section 3 will focus on the inversion of the Hessian matrix $D_u^2 E(\hat{u}(\theta))$. Iterative Krylov techniques for solving this linear system, which involve generating a Krylov subspace, will be implemented, in particular the GMRES algorithm (Saad 2003). However, Krylov methods for solving this linear system are already well established. The real focus of this section will be on ways to improve the convergence of these algorithms. Specifically, we will look at recycling Krylov methods, which are techniques that reuse subspace information generated in previous information to assist in accelerating the convergence of future iterations. For each iteration of gradient descent to solve the upper level problem, a lower level reconstruction is obtained, leading to a new objective function and hence a new linear system system to be solved. However, the new Hessian matrix changes incrementally from one iteration to the next. As there is not much difference between linear systems, the Krylov subspace generated for one system should be similar to the Krylov subspace generated for the next iteration. By this logic, reusing subspace information should save on costs, both in memory and convergence speed. Section 4 will examine how the error in the upper level gradient depends on the errors incurred in solving the lower level problem. This analysis will be exploited to reduce the computational effort involved in solving the bilevel problem.

2.4 A Case Study: Learning the TV regularisation parameter

We demonstrate and test our methods using the problem of learning the optimal total variation (TV) regularisation parameter. For simplicity the data is one-dimensional (so this is strictly a signal processing task). However, the example can be easily extended to 2D images.

Consider the variational regularisation model given by

$$E(u;\alpha) = \frac{1}{2}\left\|Bu - y\right\|_2^2 + \alpha R_\gamma(u) + \frac{\lambda}{2}\left\|u\right\|_2^2, \tag{6}$$

where $\alpha > 0$ (the weighting of the TV regularisation term) is the parameter to be learnt. Both the TV smoothing parameter $\gamma > 0$ and the convex penalty parameter $\lambda > 0$ are fixed, and the forward operator B (e.g. a blurring or subsampling matrix) is known. The regulariser R_γ is an approximation of the discrete total variation, which induces sparsity in the gradients of the signal, removing noise while preserving sharp discontinuities; for a more detailed overview of regularisers used in imaging tasks, see Chambolle and Pock (2016). It is defined as

$$R_\gamma(u) = \sum_{i=1}^{n} \rho_\gamma([\mathcal{A}u]_i),$$

where $\mathcal{A} = \hat{\nabla}$ is the 1D finite difference operator and $\rho_\gamma : \mathbb{R} \to \mathbb{R}$ is a twice differentiable version of the Huber loss function,

$$\rho_\gamma(x) = \begin{cases} -\frac{|x|^3}{3x^2} + \frac{x^2}{\gamma}, & \text{if } |x| \leq \gamma \\ |x| - \frac{\gamma}{3}, & \text{if } |x| > \gamma. \end{cases}, \qquad \gamma > 0,\ x \in \mathbb{R}.$$

The function ρ_γ is a smooth version of the absolute value function; its purpose is to ensure that R_γ is sufficiently smooth. The differentials of the TV regulariser are given by

$$D_u R_\gamma(u) = \mathcal{A}^T \begin{pmatrix} \rho'_\gamma([\mathcal{A}u]_1) \\ \vdots \\ \rho'_\gamma([\mathcal{A}u]_n) \end{pmatrix}, \qquad D_u^2 R_\gamma(u) = \mathcal{A}^T \begin{pmatrix} \rho''_\gamma([\mathcal{A}u]_1) & & \\ & \ddots & \\ & & \rho''_\gamma([\mathcal{A}u]_n) \end{pmatrix} \mathcal{A}.$$

In order to choose an appropriate GD step size for the lower level problem, we require the smoothness and convexity constants for the objective.

Lemma 2.3. *The lower level objective (6) is M-smooth and μ-convex with*

$$M = \|B\|_2^2 + \frac{8\alpha}{\gamma} + \lambda, \qquad \mu = \|B\|_2^2 + \lambda. \tag{7}$$

Proof. The gradient of the upper level objective is

$$\nabla E(u;\alpha) = B^\top(Bu - y) + \alpha D_u R_\gamma(u) + \lambda u.$$

Note that $\|\mathcal{A}\|_2 \leq 2$ and $\rho'_\gamma(x)$ is $(2/\gamma)$-Lipschitz. Thus

$$\begin{aligned}
\|D_u R_\gamma(u) - D_u R_\gamma(v)\|_2 &= \left\| \mathcal{A}^\top \begin{pmatrix} \rho'_\gamma([\mathcal{A}u]_1) \\ \vdots \\ \rho'_\gamma([\mathcal{A}u]_n) \end{pmatrix} - \mathcal{A}^\top \begin{pmatrix} \rho'_\gamma([\mathcal{A}v]_1) \\ \vdots \\ \rho'_\gamma([\mathcal{A}v]_n) \end{pmatrix} \right\|_2 \\
&\leq \|\mathcal{A}^\top\|_2 \left(\sum_{i=1}^n \left(\rho'_\gamma([\mathcal{A}u]_i) - \rho'_\gamma([\mathcal{A}v]_i)\right)^2 \right)^{1/2} \\
&\leq \|\mathcal{A}^\top\|_2 \cdot \frac{2}{\gamma} \cdot \|\mathcal{A}(u-v)\|_2 \\
&\leq \|\mathcal{A}^\top\|_2 \cdot \frac{2}{\gamma} \cdot \|\mathcal{A}\| \|u-v\|_2 \\
&\leq \frac{8}{\gamma} \|u-v\|_2.
\end{aligned}$$

It follows that

$$\begin{aligned}
\|\nabla E(u) - \nabla E(v)\|_2 &= \left\| B^\top B(u-y) + \alpha(D_u R_\gamma(u) - D_u R_\gamma(v)) + \lambda(u-v) \right\|_2 \\
&\leq \|B\|^2 \|u-v\|_2 + \alpha \|D_u R_\gamma(u) - D_u R_\gamma(v)\|_2 + \lambda \|u-v\|_2 \\
&\leq \left(\|B\|_2^2 + \frac{8\alpha}{\gamma} + \lambda \right) \|u-v\|_2.
\end{aligned}$$

The strong convexity constants for each of the individual terms in (6) are given by $\|B\|^2$, 0 and λ respectively. It follows that their sum is strongly convex with $\mu = \|B\|_2^2 + \lambda$. □

The lower level problem is solved using GD with step-size as specified in Section 2.2. Finally, we specify the constituent components of the upper level gradient:

$$D_{\alpha,u} E(\hat{u}(\alpha)) = D_u R_\gamma(\hat{u}(\alpha)) \tag{8}$$

$$D_u^2 E(\hat{u}(\alpha)) = B^\top B + D_u^2 R_\gamma(\hat{u}(\alpha)) + \lambda I \tag{9}$$

$$\nabla_{\hat{u}} L_{u^\star}(\hat{u}(\alpha))^\top = \hat{u}(\alpha) - u^\star. \tag{10}$$

This example will be used to perform numerical experiments in the next two sections, in which we examine ways to improve the convergence speed of the problem and utilise inexact gradients.

3 Recycling Krylov Methods

3.1 Introduction

Recall the gradient of the upper level problem given by (5)

$$g = -D_{\theta,u}E(\hat{u}(\theta))[D_u^2 E(\hat{u}(\theta))]^{-1}\nabla_{\hat{u}(\theta)} L_{u^\star}(\hat{u}(\theta))^\top.$$

This gradient involves the inversion of the Hessian matrix of the lower level objective function with respect to u. As stated previously, its is computationally impractical to invert this matrix, as images lead to very large linear system to solve, and aside from this, the linear system has to be solved repeatedly as it is solved in every iteration of gradient descent for minimising the upper level objective function.

Instead, it is much preferred to use an iterative solver to solve the linear system

$$w = [D_u^2 E(\hat{u}(\theta))]^{-1}\nabla_{\hat{u}(\theta)} L_{u^\star}(\hat{u}(\theta))^\top \tag{11}$$

as these methods are much more cost effective and are useful for large systems where it is reasonable to trade-off the precision of the solution for a shorter run time.

For this problem, the Hessian matrix is positive definite and sparse: its structure is determined by the form of the regularisation functional. This makes Krylov methods the perfect candidate for solving this system, as repeatably performing matrix-vector products, which is the main operation in Krylov methods, is computationally convenient when the matrix is sparse. This section will look at Krylov subspace methods for solving the linear system (11). Due to the structure of the bilevel problem, during the optimization process, different instances of this linear system are solved many times (once for each upper level gradient step), with not much variation between the linear systems. For this reason it is possible to speed up convergence using recycling Krylov methods, which reuse previous subspace information when solving the new system. This will be looked at in this section also.

3.2 Krylov Methods

Krylov subspace methods are types of iterative techniques for solving linear systems of the form

$$Ax = b$$

where $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and $x \in \mathbb{R}^n$. The linear system in (11) is of the same form, where $A = D_u^2 E(\hat{u}(\theta))$, $b = \nabla_{\hat{u}(\theta)} L_{u^\star}(\hat{u}(\theta))^\top$, and $x = w$.

Krylov subspace methods are based on projections onto Krylov subspaces. A Krylov subspace is defined as follows:

Definition 3.1. A Krylov Subspace of dimension m is a subspace which is spanned by vectors of the form $q_{m-1}(A)v$ where q_{m-1} is a polynomial of degree $m-1$, *i.e.*

$$\mathcal{K}_m(A, v) = \text{span}(v, Ab, A^2v, A^3v, \dots, A^{m-1}v)$$

Krylov subspace methods aim to minimise the residual vector $r = b - Ax$. For Krylov subspace methods, an approximation x_m to the true solution is extracted from the subspace $x_0 + \mathcal{K}_m$, where x_0 represents an arbitrary initial guess to the solution, by imposing the Petrov-Galerkin (Saad 2003) condition that the residual vector be perpendicular a subspace of dimension m, $\mathcal{L}_m$ *i.e.*

$$x_m = x_0 + \mathcal{K}_m, \qquad \text{with} \qquad r_m = b - Ax_m \perp \mathcal{L}_m \tag{12}$$

where $\mathcal{L}_m$ is known as the constraint space, and the choice of this subspace is dependent on the iterative technique. For this project, we focus solely on minimum residual variation methods, where its is necessary for the residual vector to be orthogonal to the Krylov space, and choose $\mathcal{L}_m = A\mathcal{K}_m$ (Saad 2003). The corresponding Krylov subspace for these methods is therefore given by

$$\mathcal{K}_m(A, r_0) = \text{span}(r_0, Ar_0, A^2r_0, A^3r_0, \dots, A^{m-1}r_0) \tag{13}$$

where $r_0 = b - Ax_0$. It is clear to see that the approximations obtained from a Krylov subspace method is of the form

$$A^{-1}b \approx x_m = x_0 + q_{m-1}(A)r_0.$$

In each iteration m of the approximation process, the dimension of the Krylov subspace increases by one. The Krylov space spans the initial residual vector multiplied by a polynomial of the matrix A. As the computed solution for Krylov iterative methods is extracted from the Krylov subspace, it is necessary to generate a basis for the Krylov subspace. The method that will be used to generate this basis is the Arnoldi-Modified Gram-Schmidt method.

3.2.1 Arnoldi's Method for Orthogonalization

Arnoldi's method is an orthogonal projection method onto $\mathcal{K}_m$ for general square non-hermitian matrices (Saad 2003). It is a procedure for building an orthonormal basis of the Krylov subspace, which is comprised of vectors, v_i, known as the Arnoldi vectors. The method works in the following way: at each step, the algorithm multiplies the previous Arnoldi vector v_j by the matrix A, and then normalises their resulting product, w_j, against all of the previous v_i's by a standard Gram-Schmidt procedure.

These Arnoldi vectors can be stored in a matrix $V_m \in \mathbb{R}^{n\times m}$, whose columns contain $v_1, \dots, v_m$. It should be noted that V_m is an orthogonal matrix and $V_m^\top V_m = I_m$. Arnoldi's algorithm was originally developed as a means of reducing a dense matrix into Hessenberg form with a unitary transformation. A Hessenberg matrix is one which is 'almost' triangular; it has zero entries below its first subdiagonal, or above its first superdiagonal, depending on whether it is lower or upper Hessenberg respectively. This leads to a relation known as the **Arnoldi Relation**:

Let $\bar{H}_m$ be the $(m+1)\times m$ upper Hessenberg matrix, whose non-zero entries h_{ij} and $h_{j+1,j}$ for all $i, = 1, \dots, m$, are defined in Algorithm 1. Let H_m denote the matrix obtained by deleting the last row from $\bar{H}_m$. The following relations hold

$$AV_m = V_{m+1}\bar{H}_m \tag{14}$$
$$V_m^\top AV_m = H_m \tag{15}$$

The method described above works for exact arithmetic, however, in practice it is preferred to use the Modified Gram-Schmidt instead of the standard Gram-Schmidt algorithm as it is more reliable when dealing with rounding error. The Arnoldi- Modified Gram-Schmidt procedure is outlined in Algorithm 1.

Now that we have developed a way of constructing an orthonormal basis for the Krylov subspace, we look at ways of solving the linear system $Ax = b$. The particular method that we will be looking at is known as the Generalised Minimum Residual Method (GMRES).

Algorithm 1: Arnoldi-Modified Gram-Schmidt (Saad 2003)

```
1  Choose a vector $v_1$, such that $\|v_1\|_2 = 1$
2  for $j = 1, 2, \dots, m$ do
3      Compute $w_j := Av_j$
4      for $i = 1, \dots, j$ do
5          $h_{ij} = \langle w_j, v_i \rangle$
6          $w_j := w_j - h_{ij}v_i$
7      $h_{j+1,j} = \|w_j\|_2$
8      If $h_{j+1,j} = 0$ then Stop
9      $v_{j+1} = w_j / h_{j+1,j}$
10 End
```

3.2.2 GMRES

GMRES is a projection method which minimizes the residual norm over all vectors in the space $x_0 + \mathcal{K}_m$. It uses Arnoldi orthogonalization to construct the basis for the Krylov space. We note that any vector x in $x_0 + \mathcal{K}_m$ can be written as

$$x = x_0 + V_m y, \quad \text{where } y \in \mathbb{R}^m. \tag{16}$$

We define the norm of the residual of x to be

$$J(y) = \|b - Ax\|_2 = \|b - A(x_0 + V_m y)\|_2. \tag{17}$$

Looking more closely at the residual itself, we set the first Arnoldi vector v_1 to be $v_1 = r_0/\|r_0\|_2$ and set $\beta = \|r_0\|_2$, and obtain the following

$$\begin{aligned} b - Ax &= b - A(x_0 + V_m y) \\ &= b - Ax_0 - AV_m y \\ &= r_0 - AV_m y \\ &= r_0 - (V_{m+1}\bar{H}_m)y, \quad \text{by applying (14)} \\ &= \beta v_1 - (V_{m+1}\bar{H}_m)y \\ &= V_{m+1}(\beta e_1 - \bar{H}_m y) \end{aligned}$$

where e_1 is the canonical basis vector of $\mathbb{R}^{m+1}$ with the fist element being 1. This last relationship is obtained since the Arnoldi vector v_1 is orthogonal to all of the other Arnoldi vectors, and $(V_m)e_1 = v_1$.

Going back to (17)

$$J(y) = \|b - A(x_0 + V_m y)\|_2 = \|V_{m+1}(\beta e_1 - \bar{H}_m y)\|_2 = \|(\beta e_1 - \bar{H}_m y)\|_2 \tag{18}$$

since the columns of V_{m+1} are orthonormal. The aim is to compute the unique vector in $x_0 + \mathcal{K}_m$ that minimises the residual $r_m - b - Ax_m$ and satisfies the Petrov-Galerkin condition stated in (12). This approximation can be computed simply by

$$x_m = x_0 + V_m y_m, \quad \text{where} \tag{19}$$

$$y_m = \arg\min_y \|\beta e_1 - \bar{H}_m y\|_2. \tag{20}$$

The minimizer y_m is inexpensive to compute as it is the solution to the $(m+1) \times m$ least squares problem, where m is typically small. In the implementation, this can be solved using MATLAB's backslash operator. Typically $m \ll n$, and therefore solving the linear system in (20) directly is much cheaper computationally than directly solving (11). The GMRES algorithm is given below in Algorithm 2

Algorithm 2: GMRES (Saad 2003)

```
Input: $A$, $b$, $x_0$, $m$ = maxiter
Output: $x_m$
1 Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$. and $v_1 := r_0/\beta$
2 for $j$ = 1,2,...,$m$ do
3     Run the Arnoldi/Modified Gram-Schmidt procedure detailed in Algorithm 1. If
      $h_{j+1,j} = 0$ during this process, set $m := j$ and go to 4
4 Define the $(m+1) \times m$ Hessenberg matrix $\bar{H}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$
5 Compute $y_m$, the minimizer of $\|\beta e_1 - \bar{H}_m y\|_2$
6 $x_m = x_0 + V_m y_m$
```

3.3 Recycling Methods

In the previous section, we looked at Krylov methods, and the standard implementation of the Krylov iterative solver, GMRES. In this section, we now turn our attention to an adaption of these methods, known as **Recycling Krylov Methods**. These methods are based on the concept of subspace recycling, where subspace information is preserved between iteration cycles for a single linear system solved with restarting or between different linear systems. This is done to mitigate the effects of discarding basis vectors due to memory requirements, as well as to accelerate the convergence of an iterative method Soodhalter, Sturler, and Kilmer 2020. Recycling Krylov methods are ideal for solving multiple linear systems

$$A^{(i)}x^{(i)} = b^{(i)}, \quad i = 1, 2, \dots \tag{21}$$

where $A^{(i)} \in \mathbb{R}^{n \times n}$, $b^{(i)} \in \mathbb{R}^n$, $x^{(i)} \in \mathbb{R}^n$ change from one consecutive system to the next.

In the context of bilevel optimisation (2), the linear system (11) is solved when minimising the upper level objective function using gradient descent. For each iteration of gradient descent, a lower level reconstruction is obtained, leading to a new objective function and hence a new linear system to be solved. However, between iterations of gradient descent, the new linear system is slowly changing. Assuming this linear system is solved with GMRES, as there is not much difference between linear systems (at least when gradient descent is starting to converge), the Arnoldi basis generated for one system should be similar to the Arnoldi basis generated for the next iteration. By this logic, reusing the subspace information should save on costs, both in memory and amount of iterations. This makes bilevel optimisation an ideal candidate for subspace recycling. We will adapt Algorithm 2 to include subspace recycling.

3.3.1 Recycling GMRES

Recall the Petrov-Galerkin condition given in (12). For standard GMRES, we chose the constraint subspace to be $\mathcal{L}_m = A\mathcal{K}_m$. For performing recycling, we use an augmented projection method, *i.e.* for the projection space, we use a sum correction space $\mathcal{U} + \mathcal{V}_j$, where $\mathcal{V}_j$ is the Krylov space generated at the jth iteration of the Arnoldi process, and $\mathcal{U}$ is the fixed augmentation space, recycled from the previously generated, but discarded, subspace. Let $\mathcal{U}$ have fixed dimension $\dim\mathcal{U} = k$ and $\mathcal{V}_j$ have dimension $\dim\mathcal{V}_j = j$. We can define an analogous condition to (12) in the context of subspace recycling for the GMRES algorithm, where the constraint space is therefore chosen to be $\mathcal{L}_m = A(\mathcal{U} + \mathcal{V}_j)$.We have the following condition:

$$\text{select } s_j \in \mathcal{U} \text{ and } t_j \in \mathcal{V}_j, \text{ such that } r_m = b - A(x_0 + s_j + t_j) \perp A(\mathcal{U} + \mathcal{V}_j) \tag{22}$$

where the solution to the linear system is given by

$$x_j = x_0 + s_j + t_j. \tag{23}$$

For the sequence of linear systems defined in (21), $A = A^{(i)}$ and $\mathcal{U}$ was the subspace generated to solve $A^{(i-1)}x^{(i-1)} = b^{(i-1)}$.

We now focus our attention on deriving the method required for Recycling GMRES. We choose the Krylov subspace to be $\mathcal{V}_j = \mathcal{K}_j((I-Q)A, (I-Q)r_0)$, where Q is the orthogonal projector onto $A\mathcal{U}$. Therefore, it is necessary to use the Arnoldi process to generate the orthonormal process for $\mathcal{K}_j((I-Q)A, (I-Q)r_0)$. We define the orthogonal matrix $C \in \mathbb{R}^{n\times k}$ such that $C^{=}I$, range$(C) = A\mathcal{U}$ and $Q = CC^\top$. The Arnoldi relation (14) becomes

$$(I-Q)AV_j = V_{j+1}\bar{H}_j \iff AV_j = CB_j + V_{j+1}\bar{H}_j, \quad \text{where } B_j = C^\top AV_j \tag{24}$$

This yields the following equation known as the **modified Arnoldi relation** (Soodhalter, Sturler, and Kilmer 2020)

$$A\begin{bmatrix} U & V_j \end{bmatrix} = \begin{bmatrix} C & V_{j+1} \end{bmatrix}\begin{bmatrix} I & B_j \\ 0 & \bar{H}_j \end{bmatrix} \tag{25}$$

Therefore, the recycled GMRES algorithm leads to the following solution

$$x_j = x_0 + s_j + t_j.$$

where

$$(z_j, y_j) = \arg\min_{z,y} \left\| \begin{bmatrix} I & B_j \\ 0 & \bar{H}_j \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} - \begin{bmatrix} C^0 & e_1\beta \end{bmatrix} \right\|_2 \tag{26}$$

with $s_j = Uz_j$, $t_j = V_j y_j$, and $\beta = \|(I - Q)r_0\|_2$. For practical implementation, it is convenient to first solve (26) for y_j, and then use this to solve for z_j. This is the approach that has been implemented in the accompanying code.

3.3.2 Choosing Recycling Vectors

This leads to the question however, of how to choose the subspace $\mathcal{U}$? The amount of subspace information that can be recycled is typically dictated by available memory. For solving these large linear systems, it is necessary to balance the memory cost with the cost of computation. The more information is recycled, the less computations needed for the recycling GMRES method, but the more storage needed, and vice versa. We consider two different approaches for judiciously selecting this subspace.

The first approach is simply to choose a fixed number of the Arnoldi basis vectors. This can be done by multiplying the basis V_m by a matrix E_k which contains canonical basis vectors, where the position of the 1 corresponds to the vector to be recycled.

Another way of choosing the subspace information is to recycle the *harmonic Ritz vectors* (Parks et al. 2006) which correspond to the harmonic Ritz values of smallest magnitude. In this method, the recycled subspace is chosen to be a linear combination of former basis vectors corresponding to the smallest eigenvalues of the matrix given by:

$$H_m + h_{m+1,m}^2 H_m^{-T} e_m e_m^\top \tag{27}$$

This method is a more intelligent choice than the previous method of arbitrarily choosing a fixed number of Arnoldi vectors. We will validate this with a numerical experiment.

Both methods were tested on a simple test problem. Consider the linear system $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is the 1D blurring forward operator and $b \in \mathbb{R}^n$ is the blurred, noisy data. We would like to use recycling GMRES to solve this system for x. In this experiment $n = 256$, and an initial 20 iterations of GMRES were performed. From the Arnoldi basis V_m generated from the procedure, a fixed number $k = 3$ basis vectors were chosen. Both methods for choosing recycling vectors were tested. The convergence of the recycling procedure for the different choices of subspace information is shown in Figure 1. As expected, choosing subspace information based on (27) is a more efficient process than simply choosing the Arnoldi vectors themselves. The full algorithm for Recycled GMRES is given in Algorithm 3.

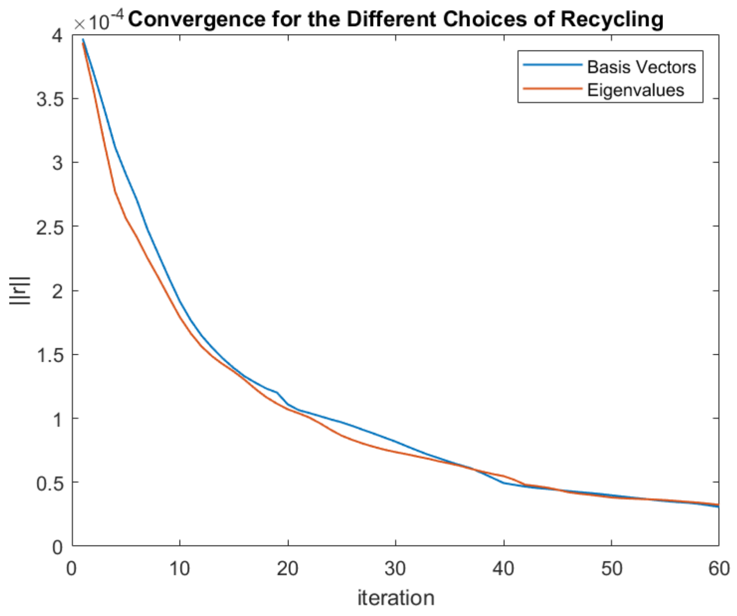


Figure 1: The convergence information for the two different methods of choosing recycling vectors.

Algorithm 3: Recycled GMRES (Parks et al. 2006)

```
    Input: $A$, $b$, $x_0$, $V_{m+1}$, $\bar{H}_m$, $m = \texttt{maxiter}$, $k$
    Output: $x_m$, $\bar{H}_m$, $V_m$
1   Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := r_0/\beta$
2   Define $H_m = \bar{H}_m$ with the last row deleted.
3   Compute the $k$ eigenvectors $\tilde{z}_j$ of $(H_m + h^2_{m+1,m} H_m^{-T} e_m e_m^\top)\tilde{z}_j = \tilde{\theta}_j \tilde{z}_j$ associated with the
     smallest magnitude of eigenvalues $\tilde{\theta}_j$ and store in $P_k$.
4   $V_k = V_m P_k$
5   Let $[Q, R]$ be the reduced QR factorisation of $\bar{H}_m P_k$.
6   $C = V_{m+1} Q$
7   $U = V_k R^{-1}$
8   for $\|r_i\|_2 <$ tol do
9       Perform $m - k$ Arnoldi steps with the linear operator $(I - CC^\top)A$, letting
         $v_1 = r_0/\|r_0\|_2$ and generate $V_{m-k+1}$ and $\bar{H}_{m-k}$.
10      $B = C^\top A V_{m-k}$
11      Let $\beta = \|r_0\|_2$ and solve $y = \arg\min_y \|\bar{H}_{m-k} y - \beta e_1\|$
12      Solve $z = C^\top r_0 - By$
13      $s = Uz$
14      $t = V_{m-k} y$
15      Obtain $x = x_0 + s + t$ with $r = \|b - Ax\|$
16  end
```

3.4 Experimentation

3.4.1 Comparing convergences for GMRES vs Recycled GMRES

In this section, we implement the algorithms described in the previous sections, and perform some numerical experiments comparing the performances of algorithms 2 and 3. The performance of GMRES and recycled GMRES will be tested on the Total Variation case study described in Section 2.4, where the upper level objective function is given in (4) and the lower level objective function is given by (6).

All of the experimentation was complemented using MATLAB 2020b. The functions and code developed for this project can be found in this GitHub repository, which includes a full description of the purpose of each function, and the hierarchy of the bilevel code.

The signals were chosen to have length $n = 256$, for all of the experiments. The training ground truth signals were generated using a "signal generator function", which creates piecewise continuous signals of a given length, whose range varies from [-2 2], with a random number of jumps. The corresponding blurred noisy data was generated by applying the 1D Gaussian Blur operator B to these signals, and adding some Gaussian noise with standard deviation $\sigma = 0.03$.

The main solver for both the lower level problem and the upper level problem is gradient descent. The bilevel optimisation procedure is structured in the following way: an initial guess α_0 for the regularisation parameter is chosen. This initial guess for α is used as the starting point for the upper level gradient descent procedure. Within each iteration of gradient descent for the upper level function, a lower level reconstruction is obtained by using the current version of α as the regularisation parameter, and minimising (6) with respect to u using gradient descent. Both gradient descent functions terminate after a maximum number of iterations has been reach, or the gradient reaches a sufficient accuracy (more details on this accuracy can be found in Section 4). The value of the objective function L and its gradient are also computed during this step, and these are used to perform the gradient descent algorithm for the upper level problem. Within the computation of the gradient of L is the inversion of the Hessian matrix. This inversion is done either using GMRES and recycled GMRES.

The specific parameter values that were used for this experimentation are shown in Table 1. The bilevel code was run twice; once using GMRES as the iterative solver for computing the gradient and once using recycled GMRES. The time taken for each method was measured using MATLAB's `tic toc` functions. The experiments were performed 6 times, with the number of training data increasing in each experiment. For both methods, the bilevel optimisation method was set to run until the tolerance criterion for the gradient was reached.

The step size for gradient descent for the lower level problem was determined based on the theory outlined in Section 2. For the lower level problem, the step size was chosen to be $\tau = 2/(M+\mu)$ where M is the Lipschitz constant and μ is the strong convexity constant as described in Section 2.4. As the upper level problem is not convex, the step size had to be chosen adaptively by using a two way backtracking scheme, which is detailed in Truong and Nguyen (2020).

Table 1: The parameter values that were used for experimentation.

Parameter	Value
Length of signal n	256
Huber loss parameter γ	0.002
Convex penalty coefficient λ	0.01
Initial guess for α (α_0)	1
Maximum iterations for GD for Upper Level	10000
Tolerance for GD for Upper Level	1e-9
Backtracking parameter β	0.1
Maximum iterations for GD for Lower Level	1000
Tolerance for GD Lower Level	1e-10
Number of Arnoldi iterations for GMRES (m)	20
Number of recycled vectors (k)	3
Number of Arnoldi iterations for recycled GMRES ($m-k$)	17

The number of iterations the method took and the final values for α and L were also recorded. The results can be seen in Table 2. As can be seen from observing the results, the recycling GMRES method consistently took less iterations to converge to the optimal α. Recycled GMRES took, an average, 78% of the iterations GMRES needed to converge. As the number of pairs of training data increased, recycled GMRES also took less time to converge to the optimal regularisation parameter. GMRES was only faster in time in one instance, which was when there was only one pair of training data. It should be noted however that it optimising over a large set of training data is desired as the aim is to learn the optimal parameters for a certain class of functions. On average, recycled GMRES took 89% of the time it took GMRES to converge. We can conclude that for this experiment, the performance of recycled GMRES is better than the performance of standard GMRES.

It should be noted that the optimal regularisation parameters don't vary by much as the size of the training set is increased. This implies that the experimental setting is quite robust. This is a very desirable property but may not hold with different experimental settings. Figure 2 shows the reconstruction of the first element in the training data, using the optimal α found from using a training set of $N = 20$, compared to the ground truth and the blurred noisy data. it is clear to see that the reconstruction is a good match to the true solution.

Table 2: The results from the bilevel experimentation, comparing the speed of GMRES vs GMRES with Recycling. Here N represents the number of pairs of training data, t(s) is the time for the upper level problem to be solved, measured in seconds. The optimal regularisation parameter α is given, as well as the corresponding minimised value of the upper level objective function, L.

	GMRES				Recycling GMRES			
N	t (s)	k	Alpha	Final L	t (s)	k	Alpha	Final L
1	99.704298	358	0.231802	0.0212898	133.528209	299	0.237055	0.021407
3	283.317005	351	0.294294	0.020328	215.610202	259	0.296188	0.020467
5	388.215209	285	0.280443	0.019913	369.855036	257	0.294357	0.020797
7	483.372372	253	0.269374	0.017339	329.740327	166	0.267476	0.017276
10	683.352752	243	0.265825	0.016966	601.784125	212	0.265426	0.016952
20	1519.902028	287	0.288042	0.017385	1112.114020	203	0.290147	0.017501

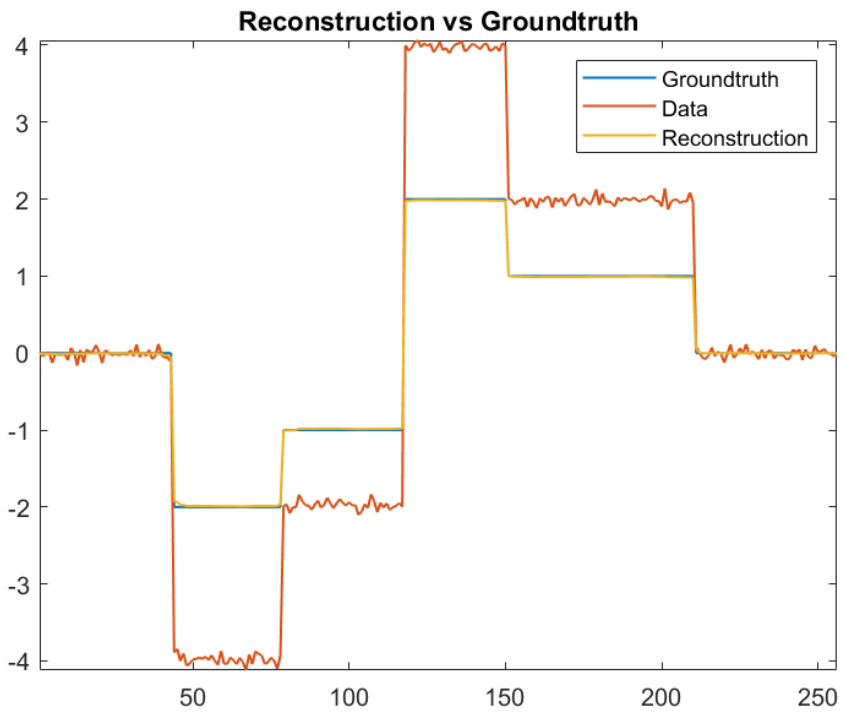
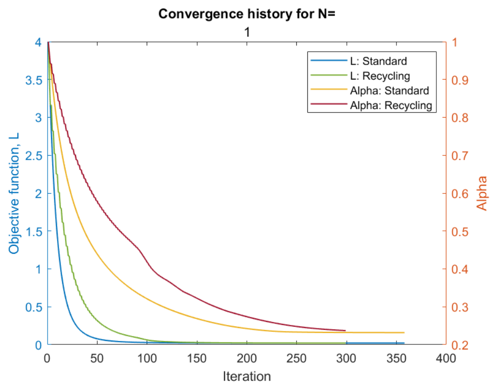
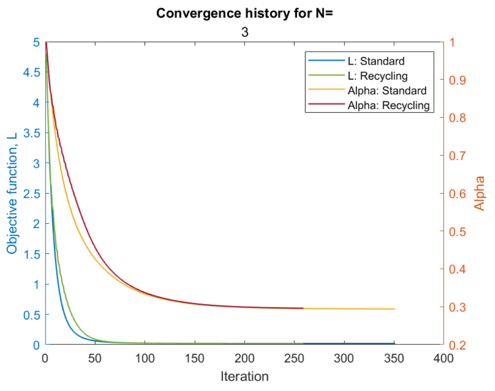


Figure 2: The reconstruction of the signal using $\alpha = 0.290147$ obtained from using a training set of size $N = 20$, compared to the corresponding ground truth and the data from which this reconstruction was made from.

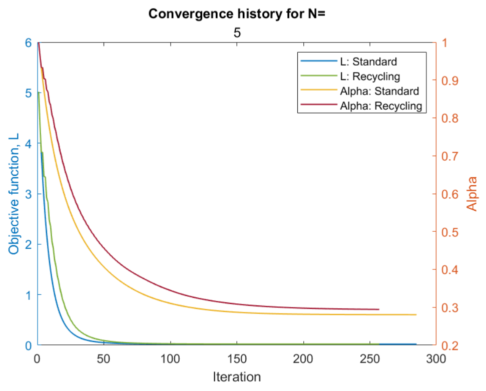
Graphs showing the values of the objective function L and the value of α for each iteration, for both methods, for all N pairs of training data are shown in Figure 3. As can be seen from observing these graphs, it appears that the standard GMRES method minimises L quicker than the recycling GMRES method. However, the recycled GMRES method reaches the stopping criterion for terminating the bilevel procedure faster than for standard GMRES. As the size of the pairs of training data increase, L appears to decrease at the same rate, with the recycled method taking less iterations. A



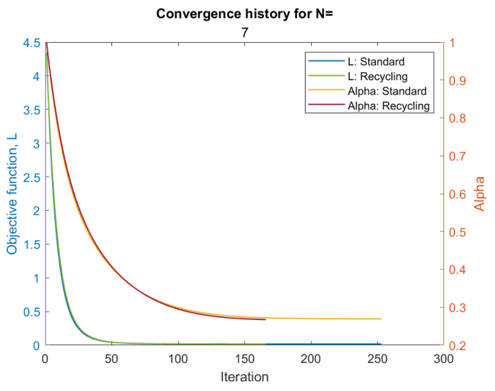
(a) $N = 1$



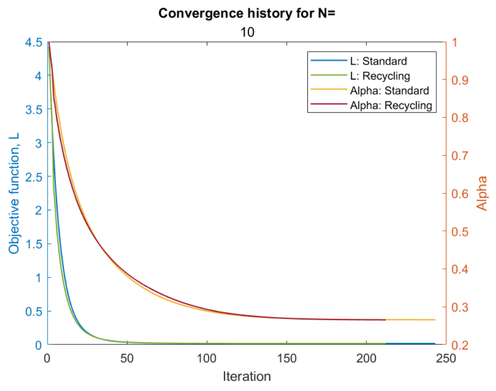
(b) $N = 3$



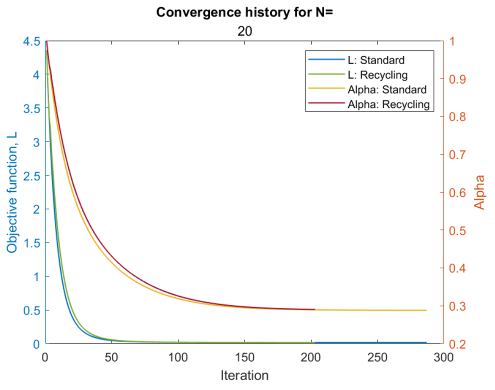
(c) $N = 5$



(d) $N = 7$



(e) $N = 10$



(f) $N = 20$

Figure 3: The convergence graphs for the bilevel optimisation test problem using both GMRES and Recycled GMRES. The left y-axis measures the objective function L and the right y-axis measures the value of the regularisation parameter α.

3.4.2 Applying the optimal regularisation parameter to a signal not in the training set

In this experiment, we look to see if the computed optimal regularisation parameter from the previous experiment is a good choice to regularise a signal which was not in the training data. The premise of learning the optimal parameters is so they can be used to regularise any given signal of a certain type (for this project we are considering piecewise constant signals). We will now investigate whether this is valid.

We will consider the regularisation parameter obtained from using recycling for a training set of $N = 20$. The larger the training set, the better the computed regularisation is for regularising the class of functions. Figure 4 shows the training set used for that experiment. The new ground truth signal which is not a part of the original $N = 20$ training data is shown in Figure 5a. The data for this signal was generated in the same manner as in Section 3.4.1. The reconstruction procedure was then applied to the data, using a regularisation parameter of $\alpha = 0.290147$, which is the value computed solving the bilevel optimisation problem. The result is shown in Figure 5b. It is clear to see that this reconstruction is a good match to the true solution, and that this regularisation parameter works well with a signal which is not in the training set. This experiment validates the reason for doing bilevel optimisation.

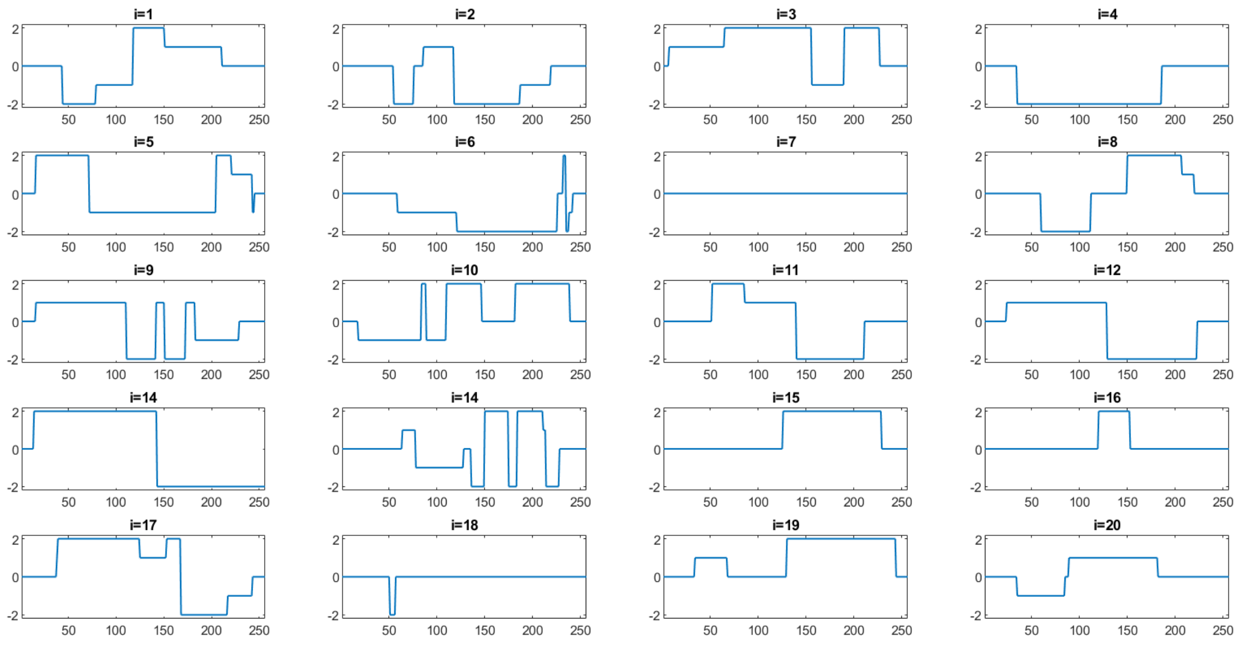
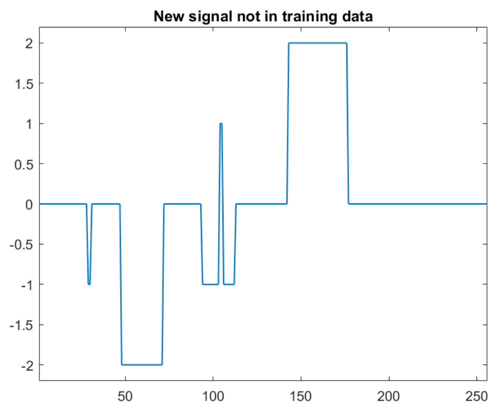
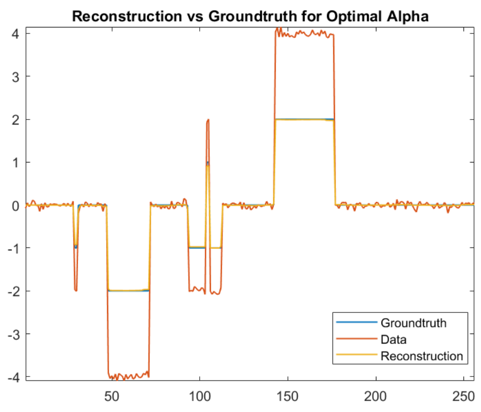


Figure 4: The training set of ground truth signals for $N = 20$.

All of the code developed and used for this experimentation can be found in the Github repository: `https://github.com/jip30/Efficient-Bilevel-Optimisation-for-Imaging.git`.



(a) The new signal which is not part of the $N = 20$ training data.



(b) The reconstructed solution with $\alpha = 0.290147$, compared to the ground truth and the blurred, noisy data.

4 Inexact Gradient Methods

4.1 Introduction

Consider the general bilevel learning problem (2). The number of parameters of interest is potentially very large, which renders procedures such as grid search infeasible due to the curse of dimensionality. Instead, the optimisation is typically performed using gradient-based algorithms such as gradient descent (GD) (Chambolle and Pock 2016). However, the nested structure of the bilevel problem creates some challenges in this respect. In particular, the gradient of the upper level objective (5) depends on the lower level solution and the inverse of the Hessian of the lower level problem, neither of which are computed exactly in practice. The viability of gradient-based methods in this setting is therefore dubious.

There are two pathways that could be taken at this juncture. First, we could avoid gradient methods altogether. This avenue is explored by Ehrhardt and Roberts (2020), who use derivative-free methods, which do not require access to the gradient, in order to solve the upper level problem. Such methods are efficient in low-dimensional problems ($<$ 100 dimensions), but scale poorly to much higher dimensions, where the convergence rate tends to be slow or the per-iteration computational cost prohibitive (Qian, Hu, and Yu 2016). The second approach - to be explored this section - is to use inexact gradient methods, i.e. gradient methods that permit inexact gradient computation. This approach is better suited to large-scale problems, and will improve the efficiency of the bilevel optimisation algorithm by reducing the computational effort expended in solving the lower level problem and the linear system, particularly in the early stages of the algorithm when even an erroneous gradient is likely to be sufficient to achieve descent.

This section is organised as follows: Section 4.2 outlines how the upper level gradient is approximated and develops a notational framework for keeping track of the errors that occur at each stage. Section 4.3 establishes an *a posteriori* bound for the error in the gradient. The proposed algorithm for bilevel learning with inexact gradients is described in Section 4.4. Empirical experiments based on the example problem from Section 2.4 are performed in Section 4.5. Throughout this section, $\|\cdot\|$ denotes $\|\cdot\|_2$.

The code implementing our approach and performing the experimentation that follows is available at https://github.com/jip30/Efficient-Bilevel-Optimisation-for-Imaging.git.

4.2 Approximating the upper level gradient

By imposing smoothness and convexity assumptions on the lower level objective, we can solve the lower level problem to arbitrary accuracy (Section 2.2). In particular, for any $\delta > 0$, the accuracy $\|u^{(k)} - \hat{u}\|^2 \leq \delta$ is guaranteed by terminating GD when the stopping criterion

$$\left\|\nabla E(u^{(k)})\right\|^2 \leq \delta \mu^2$$

is satisfied (Ehrhardt and Roberts 2020). We will denote by $\tilde{u}_\delta$ the approximate solution to the lower level problem (3) that satisfies

$$\|\tilde{u}_\delta(\theta) - \hat{u}(\theta)\| \leq \delta.$$

For brevity, we rewrite the upper level gradient (5) in a more compact form,

$$\nabla L_{u^\star}(\hat{u}(\theta)) = \Phi(\theta) H^{-1}(\theta) b(\theta),$$

where

$$\Phi(\theta) = -D_{\theta,u} E(\hat{u}(\theta);\theta), \qquad H(\theta) = D_u^2 E(\hat{u}(\theta);\theta), \qquad b(\theta) = \nabla_{\hat{u}(\theta)} L_{u^\star}(\hat{u}(\theta))^\top. \tag{28}$$

Defining

$$w(\theta) = H^{-1}(\theta) b(\theta),$$

as the solution of the linear system, the (exact) gradient is given by $\nabla L_{u^\star}(\hat{u}(\theta)) = \Phi(\theta) w(\theta)$.

Carrying forward this notation, we seek approximations for Φ and w. Let $\boldsymbol{\delta} = (\delta_1, \delta_2)$ for arbitrary $\delta_1, \delta_2 > 0$. Suppose we solve the lower level problem to accuracy δ_1, i.e. find the approximate solution $\tilde{u}_{\delta_1}(\theta)$. Approximations for Φ, H and b (denoted $\tilde{\Phi}$, $\tilde{H}$ and $\tilde{b}$, respectively) are obtained by substituting the approximate lower level solution in place of the exact solution within the expressions in (28):

$$\tilde{\Phi}_{\delta_1}(\theta) = -D_{\theta,u} E(\tilde{u}_{\delta_1}(\theta);\theta), \qquad \tilde{H}_{\delta_1}(\theta) = D_u^2 E(\tilde{u}_{\delta_1}(\theta);\theta), \qquad \tilde{b}_{\delta_1}(\theta) = \nabla_{\tilde{u}_{\delta_1}(\theta)} L_{u^\star}(\tilde{u}_{\delta_1}(\theta))^\top.$$

The approximation of w depends on both δ_1 and δ_2, because it represents the approximate solution of a linear system which itself is constructed from the approximate lower level solution. We define $\tilde{w}_{\boldsymbol{\delta}}(\theta)$ as the vector obtained by solving the lower level problem to accuracy δ_1 and the resulting linear system to accuracy δ_2 (in terms of residual norm), so that $\tilde{w}_{\boldsymbol{\delta}}(\theta)$ satisfies

$$\left\| \tilde{H}_{\delta_1}(\theta)\tilde{w}_{\boldsymbol{\delta}}(\theta) - \tilde{b}_{\delta_1}(\theta) \right\| \leq \delta_2.$$

The approximation of the upper level gradient is denoted

$$\tilde{\nabla}^{(\boldsymbol{\delta})} L_{u^\star}(\hat{u}(\theta)) = \tilde{\Phi}_{\delta_1}(\theta)\tilde{w}_{\boldsymbol{\delta}}(\theta).$$

4.3 Estimating the error in the upper level gradient

When the errors in solving the lower level problem and linear system are accounted for, the error in the upper level gradient can be written

$$e_{\boldsymbol{\delta}}(\theta) = \nabla L_{u^\star}(\hat{u}(\theta)) - \tilde{\nabla}^{(\boldsymbol{\delta})} L_{u^\star}(\hat{u}(\theta)) = \Phi(\theta)w(\theta) - \tilde{\Phi}_{\delta_1}(\theta)\tilde{w}_{\boldsymbol{\delta}}(\theta).$$

Of course, this error is unobservable since the exact gradient is unknown. Instead, we find an estimate (bound) for the error as a function of $\boldsymbol{\delta}$ and θ.

Lemma 4.1. *For any $\boldsymbol{\delta} = (\delta_1, \delta_2)$,*

$$\|e_{\boldsymbol{\delta}}(\theta)\| \leq \|H^{-1}\| \left(\|\Phi - \tilde{\Phi}_{\delta_1}\| + \|\tilde{\Phi}_{\delta_1}\| \right) \left(\delta_1 + \delta_2 + \|H - \tilde{H}_{\delta_1}\| \|\tilde{w}_{\boldsymbol{\delta}}\| \right) + \|\tilde{w}_{\boldsymbol{\delta}}\| \|\Phi - \tilde{\Phi}_{\delta_1}\|. \quad (29)$$

where the dependence on θ is omitted on the right-hand side for brevity. Assume that Φ and H (considered as functions of u) are Lipschitz continuous with constants M_Φ and M_H respectively, and that there exists a constant $C_H > 0$ such that $\|H^{-1}\| \leq C_H$. Then

$$\|e_{\boldsymbol{\delta}}(\theta)\| \leq C_H \left(M_\Phi \delta_1 + \|\tilde{\Phi}_{\delta_1}\| \right) \left(\delta_1 + \delta_2 + M_H \delta_1 \|\tilde{w}_{\boldsymbol{\delta}}\| \right) + M_\Phi \delta_1 \|\tilde{w}_{\boldsymbol{\delta}}\|. \quad (30)$$

Proof. For brevity, dependencies on θ and $\boldsymbol{\delta}$ are omitted. By repeated application of the triangle inequality,

$$\begin{aligned} \|e\| &= \|\Phi w - \tilde{\Phi}\tilde{w}\| \\ &\leq \|\Phi\| \|w - \tilde{w}\| + \|\tilde{w}\| \|\Phi - \tilde{\Phi}\| \\ &\leq \left(\|\Phi - \tilde{\Phi}\| + \|\tilde{\Phi}\| \right) \|w - \tilde{w}\| + \|\tilde{w}\| \|\Phi - \tilde{\Phi}\|. \end{aligned}$$

The exact solution w of the (exact) linear system is unknown, so further work is required:

$$\begin{aligned}
\|w - \tilde{w}\| &= \|H^{-1}(Hw - H\tilde{w})\| \\
&\leq \|H^{-1}\|\|b - H\tilde{w}\| \\
&\leq \|H^{-1}\| \left(\|b - \tilde{b}\| + \|\tilde{b} - H\tilde{w}\|\right) \\
&\leq \|H^{-1}\| \left(\|b - \tilde{b}\| + \|\tilde{b} - \tilde{H}\tilde{w}\| + \|\tilde{H}\tilde{w} - H\tilde{w}\|\right) \\
&\leq \|H^{-1}\| \left(\|b - \tilde{b}\| + \delta_2 + \|\tilde{H} - H\|\|\tilde{w}\|\right) \\
&\leq \|H^{-1}\| \left(\delta_1 + \delta_2 + \|\tilde{H} - H\|\|\tilde{w}\|\right),
\end{aligned}$$

where the last line follows from the fact that $b = \hat{u} - u^\star$ and so

$$\|b - \tilde{b}\| = \|(\hat{u} - u^\star) - (\tilde{u} - u^\star)\| = \|\hat{u} - \tilde{u}\| \leq \delta_1.$$

Combining these results yields (29). For (30), we simply note that (by assumption)

$$\begin{aligned}
\|\Phi - \tilde{\Phi}\| &\leq M_\Phi \|\hat{u} - \tilde{u}\| \leq M_\Phi \delta_1, \\
\|H - \tilde{H}\| &\leq M_H \|\hat{u} - \tilde{u}\| \leq M_H \delta_1.
\end{aligned}$$

□

The estimate (30) provides an *a posteriori* bound for the error in the gradient comprising solely computable terms. That is, for any $\boldsymbol{\delta}$ and θ, an inexact gradient $\tilde{\nabla}^{(\boldsymbol{\delta})} L_{u^\star}(\hat{u}(\theta))$ can be computed and its error estimated. A limitation of the *a posteriori* bound is that it depends on the quantities $\|\tilde{\Phi}_{\delta_1}\|$ and $\|\tilde{w}_{\boldsymbol{\delta}}\|$, which are unknown until we solve the lower level problem and linear system to $\boldsymbol{\delta}$ accuracy. Thus, we can only know that superfluous computations have been performed after the event. Nevertheless, the *a posteriori* bound can be used to improve computational efficiency.

4.4 Algorithm for bilevel optimisation with inexact gradients

The rationale behind our approach is as follows: starting with conservatively chosen (i.e. large) values of δ_1 and δ_2, we can obtain a cheap estimate of the gradient from relatively few iterations of GD and the linear solver; if the error, as estimated by (30), is larger than some specified level $\mathcal{E}$, then reduce δ_1 and/or δ_2 and perform further iterations; repeat this process until the accuracy of the gradient reaches the desired level and perform the upper level GD update in the direction of the inexact gradient. This procedure is formalised in Algorithm 4.

The multi-index of errors $\boldsymbol{\delta}$ is determined by a two-way backtracking scheme. If the desired error $\mathcal{E}_k$ has not been attained, then $\boldsymbol{\delta}^{(k)}$ is rescaled by a factor of $\beta_1 < 1$; this process is repeated until the error becomes sufficiently small. Once the error $\mathcal{E}_k$ is reached, the GD update is

Algorithm 4: Parameter selection using bilevel optimisation with inexact gradients.

Input : Ground truth signal $u^\star \in \mathbb{R}^n$; data $y \in \mathbb{R}^m$; matrix $B \in \mathbb{R}^{m\times n}$; initial parameter $\theta^{(0)} \in \Theta$; a sequence of gradient errors $\{\mathcal{E}^{(k)}\}_{k=0}^{\infty}$; initial accuracies $\boldsymbol{\delta}^{(0)} = (\delta_1^{(0)}, \delta_2^{(0)})$; tolerance for the stopping criterion, `tol` > 0; accuracy line search parameters $0 < \beta_1 < 1 < \beta_2$.

Output: The learnt parameter $\theta \in \Theta$.

```
1  Set $u_0^{(0)} = \mathbf{0}$ ;
2  for $k = 0, 1, 2, \ldots$ do
3      Solve the lower level problem initialised at $u_0^{(k)}$ to $\delta_1^{(k)}$ accuracy; obtain
         $\tilde{u} = \tilde{u}_{\delta_1^{(k)}}(\theta^{(k)})$.
4      Compute $\tilde{\Phi}_{\delta_1^{(k)}}(\theta^{(k)})$, $\tilde{H}_{\delta_1^{(k)}}(\theta^{(k)})$ and $\tilde{b}_{\delta_1^{(k)}}(\theta^{(k)})$.
5      Compute $w_{\boldsymbol{\delta}^{(k)}}(\theta^{(k)})$ by solving the linear system to accuracy $\delta_2^{(k)}$.
6      Compute an estimate of the gradient error $\tilde{e}_{\boldsymbol{\delta}^{(k)}}(\theta^{(k)})$ using (30).
7      if $\tilde{e}_{\boldsymbol{\delta}^{(k)}}(\theta^{(k)}) \le \mathcal{E}^{(k)}$ then
8          Compute the inexact gradient $\tilde{\nabla}^{(\boldsymbol{\delta}^{(k)})} L_{u^\star}(\tilde{u}) = \tilde{\Phi}_{\delta_1^{(k)}}(\theta^{(k)}) w_{\boldsymbol{\delta}^{(k)}}(\theta^{(k)})$.
9          Set $\theta^{(k+1)} = \theta^{(k)} - \tau^{(k)} \tilde{\nabla}^{(\boldsymbol{\delta}^{(k)})} L_{u^\star}(\tilde{u})$, where $\tau^{(k)}$ is chosen by backtracking
             scheme.
10         if $\|\tilde{\nabla}^{(\boldsymbol{\delta}^{(k)})} L_{u^\star}(\tilde{u})\|^2 <$ tol then
11             return $\theta^{(k+1)}$
12         end
13         Set $\boldsymbol{\delta}^{(k+1)} = \beta_2 \boldsymbol{\delta}^{(k)}$.
14     else
15         Set $u_0^{(k)} = \tilde{u}$.
16         Set $\boldsymbol{\delta}^{(k)} \leftarrow \beta_1 \boldsymbol{\delta}^{(k)}$.
17         Return to line 3.
18     end
19 end
```

performed and we take $\beta_2\boldsymbol{\delta}^{(k)}$ as the starting $\boldsymbol{\delta}$ for the (k + 1)th iteration, where $\beta_2 > 1$. This approach is likely to lead to efficient choices of $\boldsymbol{\delta}$: the $\boldsymbol{\delta}$ required in one iteration is likely to be close to that which is required in the next iteration (assuming that the error sequence $\{\mathcal{E}_k\}$ does not vary significantly from term to term), but starting the next iteration with a slightly more conservative $\boldsymbol{\delta}$ affords an opportunity to cut superfluous computations where possible. This approach could be extended further by specifying separate β_1 and β_2 values for δ_1 and δ_2. In this report, we do not concern ourselves with how best to choose these values, setting $\beta_1 = 0.9$ and $\beta_2 = 10$ throughout. Choosing $\beta_1 \approx 1$ means that the gradient error criterion is checked regularly, and choosing $\beta_2 \gg 1$ means that $\boldsymbol{\delta}$ is chosen conservatively at the start of each upper level iteration. Both of these factors contribute to reducing the number of superfluous computations.

Experiments based on the problem of learning the TV regularisation parameter introduced in

Section 2.4 will be used to illustrate and test the algorithm in the following section.

4.5 Experiments: learning the TV regularisation parameter

Unless stated otherwise, the experiments in this section are based on the following: the training set is of size $N = 1$; the ground truth signal $u^\star$ has size $n = 64$; the forward operator $B \in \mathbb{R}^{m\times n}$ is a subsampling matrix with $m = 32$; the observed data is generated by $y = Bu^\star + \xi$ where ξ is Gaussian white noise with standard deviation $\sigma = 0.03$; the TV smoothing parameter is $\gamma = 10^{-3}$ and the convex penalty parameter is $\lambda = 10^{-3}$. The ground truth, observed data and some reconstructions are shown in Figure 6. A large value TV regularisation parameter, say $\alpha = 1$, leads to poor results because sparse gradients are favoured over data fit. Conversely, when $\alpha = 0.001$ the data fit and convex penalty terms dominate, so the solution fits to the data where it exists and is zero elsewhere. Numerical experiments suggest that $\alpha = 0.043$ gives the best reconstruction. This reconstruction captures sharp discontinuities in the signal but does not fit to the noise in the data. Note that it is impossible to reconstruct the left-most discontinuity due to the absence of data at this point.

4.5.1 Estimating the error in the upper level gradient

Using the expressions for the terms in the upper level gradient given in (8), we have that

$$\begin{aligned} \Phi(\alpha) &= -D_u R_\gamma(\hat{u}(\alpha)) & \tilde{\Phi}_{\delta_1}(\alpha) &= -D_u R_\gamma(\tilde{u}_{\delta_1}(\alpha)) \\ H(\alpha) &= B^\top B + D_u^2 R_\gamma(\hat{u}(\alpha)) + \lambda I & \tilde{H}_{\delta_1}(\alpha) &= B^\top B + D_u^2 R_\gamma(\tilde{u}_{\delta_1}(\alpha)) + \lambda I \\ b(\alpha) &= \hat{u}(\alpha) - u^\star & \tilde{b}_{\delta_1}(\alpha) &= \tilde{u}_{\delta_1}(\alpha) - u^\star. \end{aligned} \tag{31}$$

Considered as functions of u, the function Φ and H are Lipschitz continuous with constants $M_\Phi = 8/\gamma$ and $M_H = 16/\gamma^2$. The derivation of M_Φ follows directly from the calculations in the proof of Lemma 2.3:

$$\|\Phi - \tilde{\Phi}\| = \|D_u R_\gamma(\hat{u}) - D_u R_\gamma(\tilde{u})\| \leq \frac{8}{\gamma}\|\hat{u} - \tilde{u}\|.$$

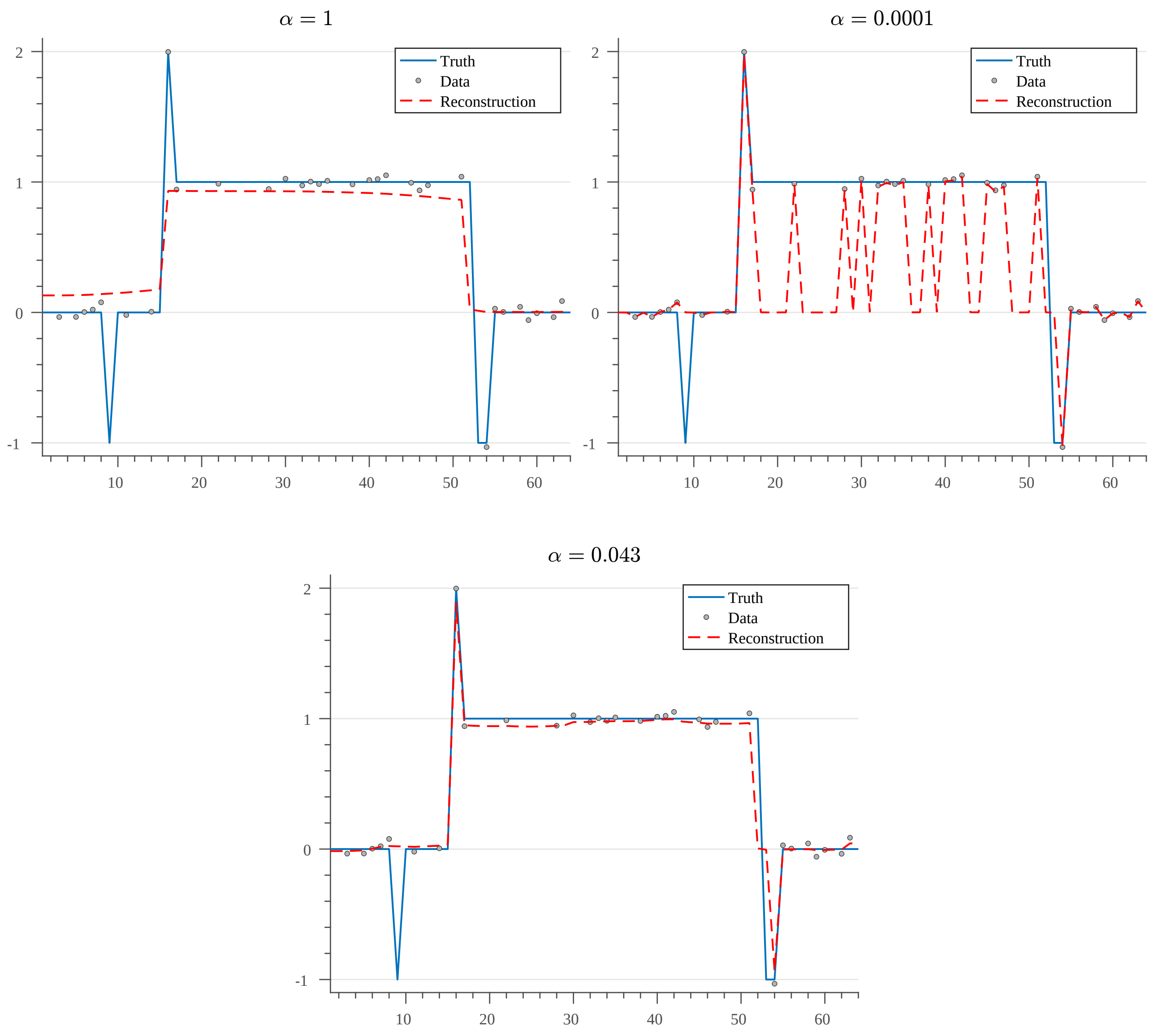


Figure 6: The reconstructions corresponding to three values of the TV regularisation parameter α.

Noting that $\rho''_\gamma(x)$ is $(2/\gamma^2)$-Lipschitz, the derivation of M_H proceeds similarly:

$$\begin{aligned}
\|H - \tilde{H}\| &= \|(B^\top B + D_u^2 R_\gamma(\hat{u}) + \lambda I) - (B^\top B + D_u^2 R_\gamma(\tilde{u}) + \lambda I)\| \\
&= \|D_u^2 R_\gamma(\hat{u}) - D_u^2 R_\gamma(\tilde{u})\| \\
&= \left\| \mathcal{A}^\top \begin{pmatrix} \rho''_\gamma([\mathcal{A}\hat{u}]_1) & & \\ & \ddots & \\ & & \rho''_\gamma([\mathcal{A}\hat{u}]_n) \end{pmatrix} \mathcal{A} - \mathcal{A}^\top \begin{pmatrix} \rho''_\gamma([\mathcal{A}\tilde{u}]_1) & & \\ & \ddots & \\ & & \rho''_\gamma([\mathcal{A}\tilde{u}]_n) \end{pmatrix} \mathcal{A} \right\| \\
&\le \|\mathcal{A}^\top\| \left\| \begin{pmatrix} \rho''_\gamma([\mathcal{A}\hat{u}]_1) - \rho''_\gamma([\mathcal{A}\tilde{u}]_1) & & \\ & \ddots & \\ & & \rho''_\gamma([\mathcal{A}\hat{u}]_n) - \rho''_\gamma([\mathcal{A}\tilde{u}]_n) \end{pmatrix} \right\| \|\mathcal{A}\| \\
&= \|\mathcal{A}^\top\| \max_{i=1,\ldots,n} \left| \rho''_\gamma([\mathcal{A}\hat{u}]_i) - \rho''_\gamma([\mathcal{A}\tilde{u}]_i) \right| \|\mathcal{A}\| \\
&\le \|\mathcal{A}^\top\| \max_{i=1,\ldots,n} \frac{2}{\gamma^2} |[\mathcal{A}(\hat{u} - \tilde{u})]_i| \, \|\mathcal{A}\| \\
&\le \|\mathcal{A}^\top\| \cdot \frac{2}{\gamma^2} \cdot \|\mathcal{A}(\hat{u} - \tilde{u})\| \cdot \|\mathcal{A}\| \\
&\le \|\mathcal{A}^\top\| \cdot \frac{2}{\gamma^2} \cdot \|\mathcal{A}\| \|\hat{u} - \tilde{u}\| \cdot \|\mathcal{A}\| \\
&\le \frac{16}{\gamma^2} \|\hat{u} - \tilde{u}\|.
\end{aligned}$$

Finally, note that $\|H^{-1}\| \le C_H = (\sigma_{\min}(B^\top B) + \lambda)^{-1}$, where $\sigma_{\min}(B^\top B)$ is the smallest singular value of the (known) matrix B. Substituting these constants into (30) yields (omitting dependencies on α)

$$\|e_{\boldsymbol{\delta}}\| \le (\sigma_{\min}(B^\top B) + \lambda)^{-1} \left(\frac{8\delta_1}{\gamma} + \|\tilde{\Phi}_{\delta_1}\| \right) \left(\delta_1 + \delta_2 + \frac{16\delta_1}{\gamma^2} \|\tilde{w}_{\boldsymbol{\delta}}\| \right) + \frac{8\delta_1}{\gamma} \|\tilde{w}_{\boldsymbol{\delta}}\|. \tag{32}$$

4.5.2 Tightness of the error bound

One of the key premises of this approach is that (30) provides a reasonable estimate for the error in the gradient. If the bound is not tight and the true error is much smaller than our estimate suggests, then excessively many computations will be performed. Therefore, it is useful to compare the estimated error against the true error. Of course, the true error is unknown, since the exact gradient is unknown. As a proxy for the exact gradient, we use the gradient that results from computing the lower level solution to accuracy $\delta_1 = 10^{-10}$ and inverting the Hessian exactly. We examine the tightness of the bound at various points in the parameter space and various values of $\boldsymbol{\delta} = (\delta_1, \delta_2)$. The results are shown in Figure 7. The tightness of the bound is poor: the estimated error and the true error differ by several orders of magnitude.

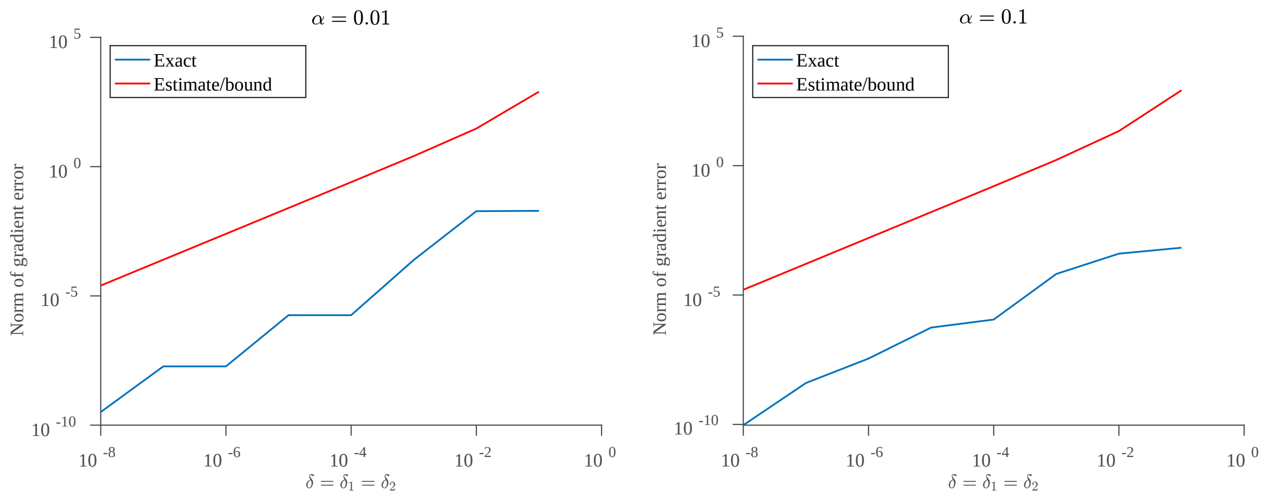


Figure 7: Examining the tightness of the bound at $\alpha = 0.01$ (left) and $\alpha = 0.1$ (right) for various values of $\boldsymbol{\delta} = (\delta_1, \delta_2)$ with $\delta_1 = \delta_2$. The 'exact' gradient was computed by computing the lower level solution to accuracy $\delta_1 = 10^{-10}$ and inverting the Hessian exactly. The estimate of the error is computed via (32).

This suggests that further theoretical work to improve the bound is necessary. However, the plot also shows that the estimated error differs from the true error by a constant multiplicative factor that is approximately independent of $\boldsymbol{\delta}$ (since the curves are vertically shifted on the log scale) and also of α (comparing the left- and right-hand plots). We can proceed to test our approach on the proviso that, if a maximum error $\mathcal{E}$ in the gradient is desired, then we should instead input $C\mathcal{E}$ as the required error for the purposes of Algorithm 4, where C is the aforementioned multiplicative constant.

4.5.3 Choosing the error sequence $\{\mathcal{E}_k\}$

The error sequence $\{\mathcal{E}_k\}_{k=0}^{\infty}$ in Algorithm 4 determines how accurately the gradient is computed at each GD iteration for the upper level problem. To be precise, the $\mathcal{E}_k$ should more accurately be interpreted as being proportional to the error in the gradient at each iteration (see Section 4.5.2). Intuitively, efficiency gains can be achieved by using a decreasing error sequence. In the early stages of the algorithm, even highly erroneous gradients are likely to be sufficient to achieve descent. This is especially true in our one-dimensional problem, where the algorithm only needs to learn the correct sign of the gradient in order to move in the current direction in the parameter space. As the algorithm progresses towards the optimum, more precise gradient computations are needed to facilitate the final fine-tuning. There are two classes of error sequences we can consider: vanishing and non-vanishing. Vanishing error assumptions, e.g.

$$\lim_{k\to\infty} \mathcal{E}_k = 0, \qquad \sum_{k=0}^{\infty} \mathcal{E}_k < \infty,$$

are stipulated by most analyses of inexact gradient methods (Bertsekas 2017). However, Sra (2012) suggests that non-vanishing but uniformly bounded error sequences may lead to im-

Table 3: The computational cost and convergence status associated with each of the error sequences.

Error sequence, $\mathcal{E}_k$	Lower level iterations	Linear solver iterations	Upper level iterations	α
Constant large	43,300	28,528	≥ 50	0.040
Constant small	416,746	22,121	25	0.043
Vanishing, non-summable	67,766	23,396	34	0.043
Exponentially decreasing	295,492	23,241	23	0.043

proved scalability. Clearly exact stationarity cannot be guaranteed when the errors are non-vanishing, but the iterates may progress towards inexact stationary points, i.e. points where the null gradient condition holds to within the current error level.

We test experimentally the performance of a variety of error sequences: constant (large), constant (small), vanishing but non-summable, and exponentially decreasing,

$$\mathcal{E}_k = C_{\text{large}}, \qquad \mathcal{E}_k = C_{\text{small}}, \qquad \mathcal{E}_k = Ck^{-1}, \qquad \mathcal{E}_k = C^{-\rho k},$$

for constants $C, \rho > 0$. The convergence history for each error sequence when applied to our example problem is shown in Figure 8. The learnt parameter and the number of lower level, linear solver, and upper level iterations required in each case is given in Table 3. When the error is fixed and large, the algorithm performs well initially, but has difficulty locating the precise location of the optimum later on. The errors prevent stationarity from being achieved, so the stopping criterion was not attained and the learnt parameter is not quite the objective minimiser. However, this scheme was very inexpensive computationally, requiring very few lower level iterations. Using a very small error threshold throughout results in no convergence issues, demonstrating a clear benefit in performing more precise computations as the algorithm progresses. However, both the large and small constant error sequences show similar initial performance (descending to an objective value of 0.4 after approximately 20 iterations), so using a small error from the outset involved significantly more computations for no discernible gain. We can combine the advantages of these two cases by specifying a decreasing error schedule, so that the initial gradient computations are cheap and the later estimates are precise. While the exponentially decreasing error sequence converged in fewer upper level iterations than the non-summable sequence, the number of lower level iterations was comparatively high. We conclude that the more gradually decreasing sequence performed best, requiring few lower level iterations and resolving the convergence issues encountered in the first sequence. Comparing this against the performance of the constant small error sequence, the advantage of our inexact gradients approach becomes evident: the number of lower level iterations was reduced by 84%.

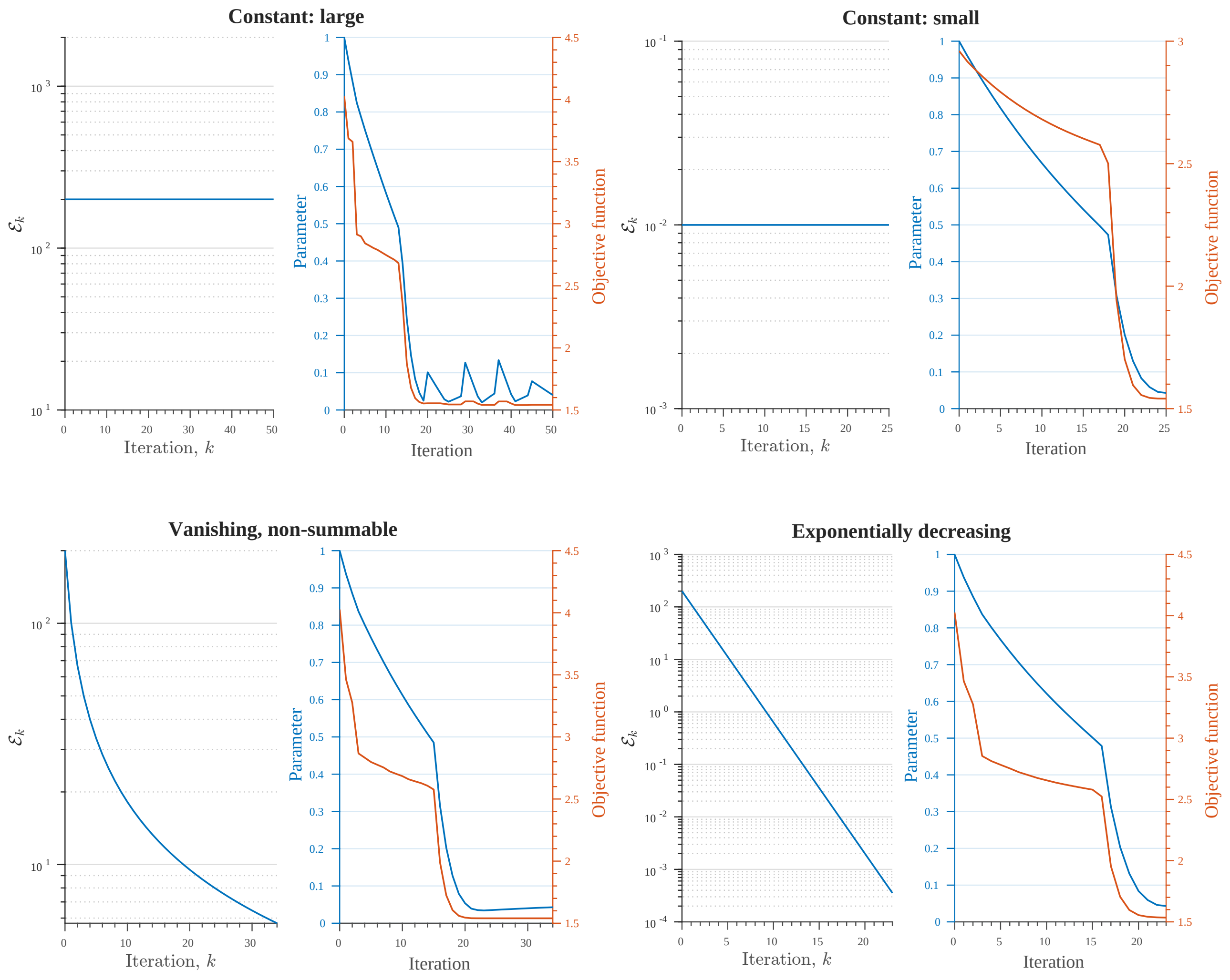


Figure 8: The convergence histories based on various error sequences: constant large, $\mathcal{E}_k = 200$; constant small, $\mathcal{E}_k = 10^{-2}$; vanishing, non-summable, $\mathcal{E}_k = 200/(k+1)$; exponentially decreasing, $\mathcal{E}_k = 200 \times 10^{-0.25k}$. Algorithm initialised at $\alpha^{(0)} = 1$ with stopping criterion `tol` $= 10^{-6}$, which was achieved in all cases except 'constant large'.

5 Conclusions and Future Work

5.1 Conclusions

Section 3 focused on improving the efficiency of solving the square linear systems given by (11). Due to the sparsity of the Hessian matrix, solving the system with iterative Krylov solvers is computationally efficient. This strand looked at finding ways of making the process even more efficient. The Krylov method chosen was GMRES, and this was adapted to include the recycling of previous subspace information. The numerical experiments show that using subspace recycling is indeed a more efficient way of approaching (11). For the particular experiments performed, taking the arithmetic mean over all 6 of the experiments shown in Table 2, recycling took on average 78% of the iterations of GMRES to converge, and converged in 89% of GMRES time. We can conclude that in this strand, a more efficient method for solving the linear system was implemented.

Section 4 exploited inexact gradient methods to minimise the computational effort expended in solving the lower level problem and the linear system. An *a posteriori* bound for the error in the gradient was established, from which we proposed an algorithm which computes the gradient to within some specified error threshold at each iteration. A well-chosen sequence of error thresholds is shown to lead to significant computational savings while retaining good convergence properties. In our toy example, we find that a gradually decreasing error sequence performs best reducing the number of lower level iterations by 84% as compared with the naive approach of computing gradients to high accuracy at every iteration.

The overall aim of this project was to develop a more efficient approach for the bilevel optimisation procedure used for parameter selection tasks in image reconstruction. It has been shown through numerical experimentation that the modifications discussed in sections 3 and 4 indeed improve upon existing approaches for solving (2). We achieved efficiency gains by using subspace recycling to solve the linear system more efficiently and allowing for inexactness in the gradient computations so that fewer lower level iterations are required at each step.

5.2 Future Work

5.2.1 Recycling Krylov methods

Thus far, a subspace recycling scheme has been implemented for the bilevel problem. This scheme uses an adaption of the iterative Krylov method, GMRES, to solve the linear system given in equation (11). As the linear system is slowly varying from iteration to iteration of the upper level problem, especially when approaching a stationary point of L in (1), it is possible, and, as seen from the experimental results, quite useful to recycle subspace information generated from previous iterations to aid in generating the Arnoldi basis for the new iteration, and as a consequence, speeding up the convergence of the bilevel method. There are however still some questions left to investigate, leaving room for future work on this problem.

One aspect to investigate is how the number of iterations affects both the speed of the convergence of this method, and also the accuracy of the computed gradient. The estimate for the solution obtained from GMRES, with and without recycling, involves the product of the matrix containing the Arnoldi vectors V_m and the vector y, as defined in Algorithm 2 and Algorithm 3. For all of the experimentation, the choice of `maxiter` and k, the number of recycled vectors, were fixed to be 20 and 3 respectively. This meant that the standard GMRES algorithm ran for 20 iterations, and the recycled GMRES ran for 17 iterations, computing V_m of dimension 20 and 17 respectively. These numbers were chosen by doing some trial an error experiments and observing how the speed and accuracy of the solution vary, and choosing values which seemed to balance both quantities in a reasonable way, delivering fast computations without compromising the accuracy of the solution. As one would expect, as the number of iterations increases, the speed of convergence decreases, but the computed gradient also behaves more accurately. It would be highly useful to do some rigorous experimentation, preferably backed up by mathematical analysis, on which combination of maximum number of iterations and recycled vectors is optimal.

Aside from determining the optimal number of maximum iterations, it would also be useful to implement a stopping criterion for both GMRES and recycled GMRES, based on the norm of the residual, with the intent that the algorithm would terminate once the residual norm is low enough. Some investigation into deciding on a suitable value for this tolerance would be a useful thing to look into, potentially developing an a priori error bound on the discrepancy between the true solution and the solution computed using GMRES and also recycled GMRES.

Another aspect to consider is the choice of Krylov method. At present we have only considered GMRES. It could be possible that a method such as CGLS may be more useful for this type of system and may provide better convergence rates.

Another avenue for future research is in deciding which subspace vectors to recycle. In the current implementation, a fixed number of vectors are chosen each time, which correspond to the smallest eigenvalues of the system defined in equation (27). A very interesting route to look at is if there are smarter ways of choosing which vectors to recycle, particularly if there are adaptive ways to choose which vectors to recycle. As described earlier, the reason it is useful to use recycling methods for this system is because the linear system may not be changing by much from each iteration to the next. Therefore, the Arnoldi basis generated in the new iteration does not vary much from the Arnoldi basis from the previous iteration, and the recycled vectors resemble the vectors that would have been generated from the Arnoldi process, had there been no recycling. However, it must be noted that the linear system does not change by a fixed amount each time. There are some iterations where there may be no change between linear systems (typically close to gradient descent convergence) and some iterations where there may be great change in the linear system (typically at the start of the gradient descent iterations). It would be useful to develop an adaptive strategy for choosing the recycled vectors, so that when there is practically no change between linear systems, more basis vectors are recycled, speeding up the GMRES process, and when the linear system differ significantly, no vectors are recycled as the basis vectors from the previous iteration are not relevant to the new linear system.

5.2.2 Inexact gradient methods

We establish an *a posteriori* bound on the error in the gradient, from which we can verify (post hoc) that a specified accuracy in the gradient has been obtained. Numerical experiments suggest the current bound is not very tight; our approach would benefit from further theoretical analysis to improve the tightness. A more significant outcome would be to establish an *a priori* bound, so that a sufficiently small $\boldsymbol{\delta}$ can be found in advance of performing any computations. This would afford two main advantages. First, the number of superfluous computations would be reduced. Using the *a posteriori* bound we can only know that the number of computations was more than required once they have already been performed. Second, it would permit a more systematic approach to choosing $\boldsymbol{\delta}$, instead of the line search procedure adopted presently.

The example considered in this report involved learning only a single parameter. It would be useful to test our approach for more complicated problems where the number of parameters is large. The key motivating reason for choosing a gradient-based approach in the first place was that they perform well on high-dimensional optimisation problems. Our example did not serve to illustrate this. A multi-parameter learning problem, e.g. learning the optimal MRI sampling pattern (Ehrhardt and Roberts 2020; Sherry et al. 2020), would also provide a more interesting test case. In the one parameter problem, errors in the gradient may not have a significant effect: provided the approximation of the gradient has the correct sign, the algorithm will move in the correct direction within the parameter space. However, if the parameter space is high-dimensional, then errors in the gradient are more likely to result in poor updates and convergence issues. As we consider tasks of varying dimensionality, it would be interesting to compare the performance of our methods with the derivative-free approach of Ehrhardt and Roberts (2020). We expect that gradient methods will perform better when the number of parameters is very large.

Our experimentation revealed some interesting conclusions about the best choice for the error sequence $\{\mathcal{E}_k\}$. It is not *a priori* clear how these conclusions would generalise to higher dimensional problems, where precise gradient computations may be necessary at an earlier stage in order to navigate the more complex objective landscape. Another future objective is to ground these conclusions in sound mathematical theory. Sra (2012) provides a useful starting point for deriving convergence results for inexact gradient methods, even under the weak assumption of non-vanishing computational errors.

Additional future work includes: expanding the framework to less tractable bilevel problems, e.g. relaxing the smoothness or convexity assumptions on the lower level objective; performing a thorough convergence analysis to derive convergence guarantees and convergence rates; testing the approach for large training sets and considering stochastic gradient descent with inexact gradient computations; formulating, experimentally or otherwise, some heuristics for choosing β_1 and β_2 used in the $\boldsymbol{\delta}$ update step.

5.2.3 Connections between the strands and overall project

As well as room for further exploration in each individual strand, there is also lots of scope for further research into connecting both strands and developing the project further as a whole.

Unifying the strands within the overall algorithm is straightforward, since the inexact gradient methods do not depend on the method used to solve the linear system. During the analysis performed in Section 4, the linear system was solved using the conjugate gradient method. The recycling GMRES solver can be substituted in its place with no ramifications.

The ability to solve the linear system more efficiently using subspace recycling methods may lead to additional computational savings within the inexact gradients framework. If the linear solver converges at a faster rate, then the computational effort required to attain a given accuracy δ_2 in the solution of the linear system is diminished. Thus, δ_2 can be reduced at little expense, and consequently δ_1 can be increased with no corresponding increase in the total error in the gradient. In this way, we would achieve a further reduction in the number of lower level iterations.

Extending the case study described Section 2.4 problem to 2D images, rather than the 1D signals which are currently implemented. This would involve setting $\mathcal{A}$ to be the 2D finite difference operator, and modifying the Lipschitz constants accordingly. It would also involve creating a new "signal generator function" which rather than computing piecewise continuous signals, computes piecewise continuous images. This would be a relatively straightforward extension. It would be interesting to investigate whether the same experiments from sections 3 and 4 achieve similar results in the 2D setting as in the 1D.

References

Saad, Yousef (2003). *Iterative Methods for Sparse Linear Systems.* SIAM, 2003.

Parks, Michael L et al. (2006). "Recycling Krylov subspaces for sequences of linear systems". In: *SIAM Journal on Scientific Computing* 28.5 (2006), pp. 1651–1674.

Sra, Suvrit (2012). "Scalable nonconvex inexact proximal splitting". en. In: *Advances in Neural Information Processing Systems* 25 (2012), p. 9.

Chambolle, Antonin and Thomas Pock (2016). "An introduction to continuous optimization for imaging". en. In: *Acta Numerica* 25 (May 2016), pp. 161–319. ISSN: 0962-4929, 1474-0508. DOI: 10.1017/S096249291600009X. URL: https://www.cambridge.org/core/product/identifier/S096249291600009X/type/journal_article (visited on 04/17/2021).

Qian, Hong, Yi-Qi Hu, and Yang Yu (2016). “Derivative-Free Optimization of High-Dimensional Non-Convex Functions by Sequential Random Embeddings”. en. In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (2016), pp. 1946–1952.

Bertsekas, Dimitri P. (2017). “Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey”. en. In: *arXiv:1507.01030 [cs, math]* (Dec. 2017). arXiv: 1507.01030. URL: http://arxiv.org/abs/1507.01030 (visited on 04/26/2021).

Ehrhardt, Matthias J and Lindon Roberts (2020). “Inexact Derivative-free Optimization for Bilevel Learning”. In: *arXiv preprint arXiv:2006.12674* (2020).

Sherry, Ferdia et al. (2020). “Learning the Sampling Pattern for MRI”. In: *IEEE Transactions on Medical Imaging* 39.12 (2020), pp. 4310–4321.

Soodhalter, Kirk M, Eric de Sturler, and Misha Kilmer (2020). “A Survey of Subspace Recycling Iterative Methods”. In: *arXiv preprint arXiv:2001.10347* (2020).

Truong, Tuyen Trung and Hang-Tuan Nguyen (2020). “Backtracking Gradient Descent Method and Some Applications in Large Scale Optimisation. Part 2: Algorithms and Experiments”. en. In: *Applied Mathematics & Optimization* (Sept. 2020). ISSN: 0095-4616, 1432-0606. DOI: 10.1007/s00245-020-09718-8. URL: http://link.springer.com/10.1007/s00245-020-09718-8 (visited on 04/23/2021).