Recycling Krylov Methods

A STRAND OF EFFICIENT BILEVEL OPTIMISATION FOR IMAGING

IRP INDIVIDUAL REPORT

Student: Jennifer Power

Supervisors: Silvia Gazzola

Matthias Ehrhardt

April 26, 2021

1 Background

This project is on bilevel optimisation, specifically in relation to inverse problems in imaging and signal processing. We are working on a parameter learning problem for image reconstruction. The image reconstruction problem for an image u, forward operator B and collected data y is formulated as a 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 which aims to remove noise and encourage a predefined structure in the solution. The regularisation parameter α controls the trade-off between these two terms. In general, there are a variety of parameters hidden within any variational problem used to recover u which need to be fine tuned such that the error between the reconstructed solution and the true solution is minimised. These parameters vary depending on the applications but can include the choice of regularisation parameter or the sampling pattern at which the data is collected. It is impractical to try to optimally balance these parameters for each individual reconstruction. Instead, it is ideal to know these optimal parameters ahead of time.

A natural, data-driven approach for learning the parameters of the variational problem is bilevel optimisation. As the name suggests, bilevel optimisation consists of two optimisation problems; an upper level problem where the parameters are selected, and a lower level problem, which can take the form of (1), nested within it. Suppose we have access to a labelled training set $\{(y_i, u_i^\star)\}_{i=1}^N$ comprising of collected data y_i and the corresponding ground truths $u_i^\star$. The optimal model parameters $\theta \in \Theta$ are obtained by solving a bilevel optimisation problem of the following form

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

where the lower level objective function for the ith pair of training data $E_i(u;\theta)$ 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 this particular project, we will focus on learning the regularisation parameter for a total variation regulariser. The objective functions of the lower and upper level problems are based on the setting given in [4]. For notational simplicity, we have dropped the dependency on i, and the lower and upper objective functions are given respectively by:

$$E(u;\alpha) = \frac{1}{2}\|Bu - y\|_2^2 + \alpha R(u) + \frac{\lambda}{2}\|u\|_2^2, \tag{3}$$

$$L_{u^*}(u(\alpha)) = \frac{1}{2}\|u - u^*\|_2^2 \tag{4}$$

where u represents an image, y is the obtained data, B is the forward operator which takes the form of a 1D Gaussian blur, λ is the convex penalty parameter, $R(u)$ is a smoothed version of the total variation (TV) functional and α is the regularisation parameter which balances the fit of the image to the given data and the TV of the solution, and is what we would like to optimise for. Both objective functions are minimised using gradient descent [1], an iterative optimization method which uses the gradient to find the minimum of a function. In this section, we focus primarily on the upper level problem, and specifically the computation of its gradient.

Recall that the gradient of the upper level problem is given by

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

This gradient involves the inversion of the Hessian matrix of the lower level objective function with respect to u. Computationally, it is very impractical to invert this matrix. Inverting an $n \times n$ matrix costs $\sim 2n^3/3$ arithmetic operations in general. This can be very costly for large n, especially when dealing with images as these systems are very large. For example, for a 256×256 pixel image, the Hessian matrix is of dimension $65,536 \times 65,536$. Instead, it is much preferred to use an iterative solver to solve the linear system

$$w = [D_u^2E(\hat{u}(\theta))]^{-1}\nabla_{\hat{u}(\theta)}L_{u^\star}(\hat{u}(\theta))^\top \tag{6}$$

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 (6). 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.

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 (6) is of the same form, where $A = D_u^2E(\hat{u}(\theta))$, $b = \nabla_{\hat{u}(\theta)}L_{u^\star}(\hat{u}(\theta))^\top$, and $x = w$.

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

Definition 2.1
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.

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

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 [3] condition that the residual vector be perpendicular a subspace of dimension m, $\mathcal{L}_m$ *i.e.*

$$x_m = x_0 + \mathcal{K}_m, \qquad \text{with} \qquad r_m = b - Ax_m \perp \mathcal{L}_m \tag{7}$$

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 $\mathcal{L}_m = A\mathcal{K}_m$ [3]. The corresponding Krylov subspace for these methods is therefore given by

$$\mathcal{K}_m(A, r_0) = \text{span}(r_0, Ar_0, A^2r_0, A^3r_0, \ldots, A^{m-1}r_0) \tag{8}$$

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.

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 [3]. 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 $v_1, \ldots, v_m$. 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 $i, = 1, \ldots, m$, 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

$$AV_m = V_{m+1}\bar{H}_m \tag{9}$$

$$V_m^\top AV_m = H_m \tag{10}$$

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 [3]

```
Choose a vector v_1, such that ||v_1||_2 = 1
for j = 1, 2, ..., m do
    Compute w_j := A v_j
    for i = 1, ..., j do
        h_ij = <w_j, v_i>
        w_j := w_j - h_ij v_i
    h_{j+1,j} = ||w_j||_2
    If h_{j+1,j} = 0 then Stop
    v_{j+1} = w_j / h_{j+1,j}
End
```

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

$$x = x_0 + V_m y, \quad \text{where } y \in \mathbb{R}^m. \tag{11}$$

We define the norm of the residual of x to be

$$J(y) = \|b - Ax\|_2 = \|b - A(x_0 + V_m y)\|_2. \tag{12}$$

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

$$\begin{aligned} b - Ax &= b - A(x_0 + V_m y) \\ &= b - Ax_0 - AV_m y \\ &= r_0 - AV_m y \\ &= r_0 - (V_{m+1}\bar{H}_m)y, \text{ by applying (9)} \\ &= \beta v_1 - (V_{m+1}\bar{H}_m)y \\ &= V_{m+1}(\beta e_1 - \bar{H}_m y) \end{aligned}$$

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 (12)

$$J(y) = \|b - A(x_0 + V_m y)\|_2 = \|V_{m+1}(\beta e_1 - \bar{H}_m y)\|_2 = \|(\beta e_1 - \bar{H}_m y)\|_2 \tag{13}$$

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 (7). This approximation can be computed simply by

$$x_m = x_0 + V_m y_m, \text{ where} \tag{14}$$

$$y_m = \arg\min_y \|\beta e_1 - \bar{H}_m y\|_2. \tag{15}$$

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 (15) directly is much cheaper computationally than directly solving (6). The GMRES algorithm is given below in algorithm 2

Algorithm 2: GMRES [3]

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 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 [5]. Recycling Krylov methods are ideal for solving multiple linear systems

$$A^{(i)}x^{(i)} = b^{(i)}, \quad i = 1, 2, \dots \tag{16}$$

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 (6) 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.

3.1 Recycling GMRES

Recall the Petrov-Galerkin condition given in (7). 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 (7) 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:

$$\text{select } s_j \in \mathcal{U} \text{ and } t_j \in \mathcal{V}_j, \text{ such that } r_m = b - A(x_0 + s_j + t_j) \perp A(\mathcal{U} + \mathcal{V}_j) \tag{17}$$

where the solution to the linear system is given by

$$x_j = x_0 + s_j + t_j. \tag{18}$$

For the sequence of linear systems defined in (16), $A = A^{(i)}$ 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 $C^\top C = I$, range$(C) = A\mathcal{U}$ and $Q = CC^\top$. The Arnoldi relation (9) becomes

$$(I-Q)AV_j = V_{j+1}\bar{H}_j \iff AV_j = CB_j + V_{j+1}\bar{H}_j, \quad \text{where } B_j = C^\top AV_j \tag{19}$$

This yields the following equation known as the **modified Arnoldi relation** [5]

$$A\begin{bmatrix} U & V_j \end{bmatrix} = \begin{bmatrix} C & V_{j+1} \end{bmatrix} \begin{bmatrix} I & B_j \\ 0 & \bar{H}_j \end{bmatrix} \tag{20}$$

Therefore, the recycled GMRES algorithm leads to the following solution

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

where

$$(z_j, y_j) = \arg\min_{z,y} \left\| \begin{bmatrix} I & B_j \\ 0 & \bar{H}_j \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} - \begin{bmatrix} C^0 & e_1\beta \end{bmatrix} \right\| \tag{21}$$

with $s_j = Uz_j$, $t_j = V_j y_j$, and $\beta = \|(I-Q)r_0\|$. For practical implementation, it is convenient to first solve (21) for y_j, and then use this to solve for z_j. This is the approach that has been implemented in the accompanying code.

3.2 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* [2] 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:

$$H_m + h_{m+1,m}^2 H_m^{-T} e_m e_m^\top \tag{22}$$

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 (22) is a more efficient process than simply choosing the Arnoldi vectors themselves. The full algorithm for Recycled GMRES is given in algorithm 3.

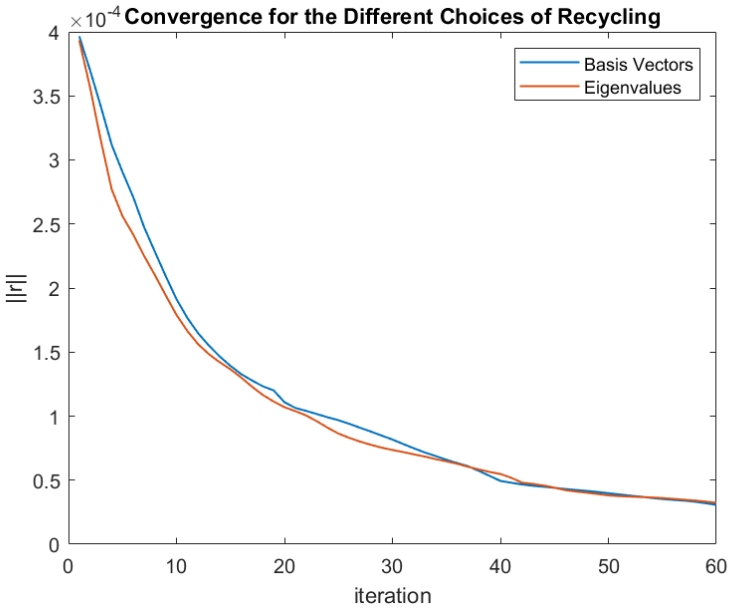


Figure 1: The convergence information for the two different methods of choosing recycling vectors.

Algorithm 3: Recycled GMRES [2]

Input: A, b, x_0, V_{m+1}, $\bar{H}_m$, $m = \texttt{maxiter}$, k
Output: x_m, $\bar{H}_m$, V_m

Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := r_0/\beta$
Define $H_m = \bar{H}_m$ with the last row deleted.
Compute the k eigenvectors $\tilde{z}_j$ of $(H_m + h_{m+1,m}^2 H_m^{-T} e_m e_m^\top)\tilde{z}_j = \tilde{\theta}_j \tilde{z}_j$ associated with the smallest magnitude of eigenvalues $\tilde{\theta}_j$ and store in P_k.
$V_k = V_m P_k$
Let $[Q, R]$ be the reduced QR factorisation of $\bar{H}_m P_k$.
$C = V_{m+1} Q$
$U = V_k R^{-1}$
for $\|r_i\|_2 < \texttt{tol}$ **do**
    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 $\bar{H}_{m-k}$.
    $B = C^\top A V_{m-k}$
    Let $\beta = \|r_0\|_2$ and solve $y = \arg\min_y \|\bar{H}_{m-k} y - \beta e_1\|$
    Solve $z = C^\top r_0 - By$
    $s = Uz$
    $t = V_{m-k} y$
    Obtain $x = x_0 + s + t$ with $r = \|b - Ax\|$
end

4 Experimentation

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 where the upper level objective function is given in (4) and the lower level objective function is given by (3).

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 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 value of α as the regularisation parameter, and minimising (3) 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. 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 [1]. 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 for the lower level problem, where $M = \|B\|_2^2 + \frac{8\alpha}{\gamma} + \lambda$ and $\mu = \|B\|_2^2 + \lambda$. 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 [6].

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 the best approach for learning the parameters 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.

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

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.

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

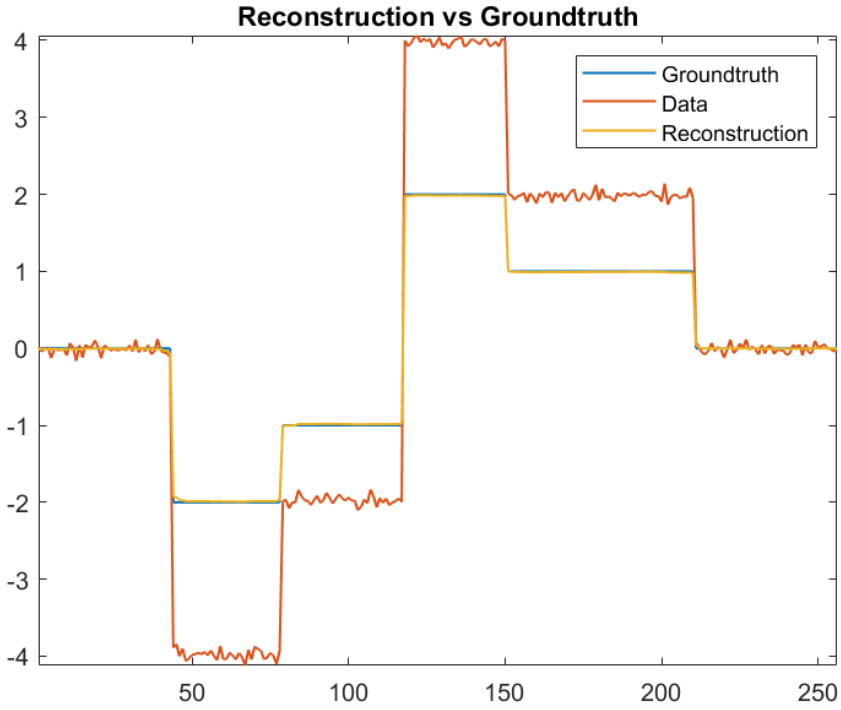
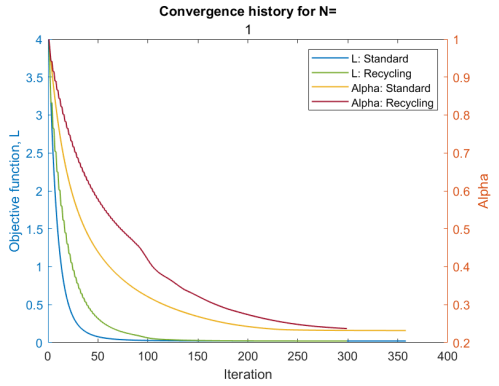


Figure 2: The reconstruction of the signal using $\alpha = 0.290147$ obtained from using a training set of size $N = 20$, compared to the corresponding ground truth and the data from which this reconstruction was made from.

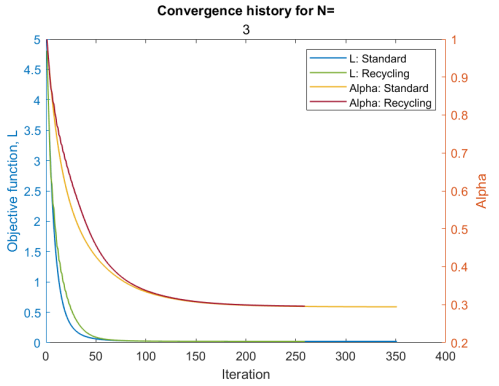
4.2 Applying the optimal regularisation parameter to a signal not in the training set

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.

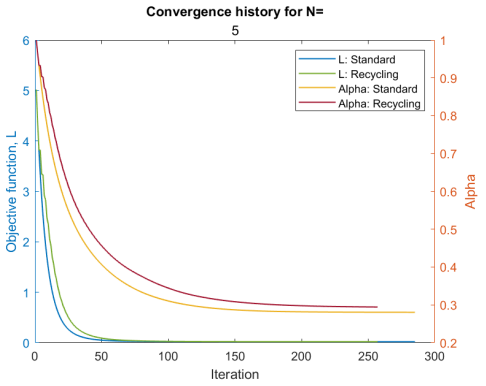
We will consider the regularisation parameter obtained from using recycling for a training set of $N = 20$. The larger the training set, the better the computed regularisation is for regularising the class of functions. Figure 4 shows the training set used for that experiment. The new ground truth signal 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 subsection 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.



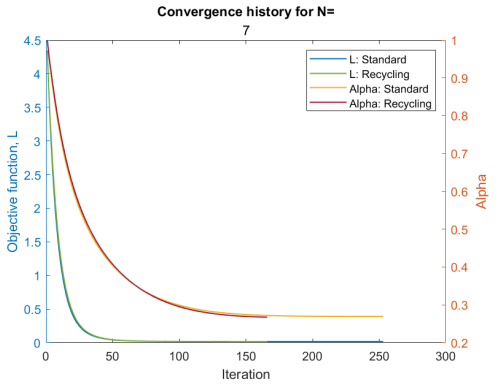
(a) $N = 1$



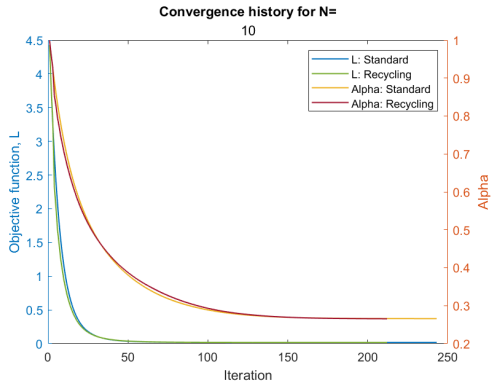
(b) $N = 3$



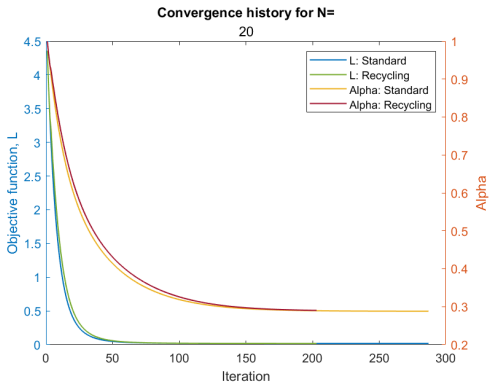
(c) $N = 5$



(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.

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.

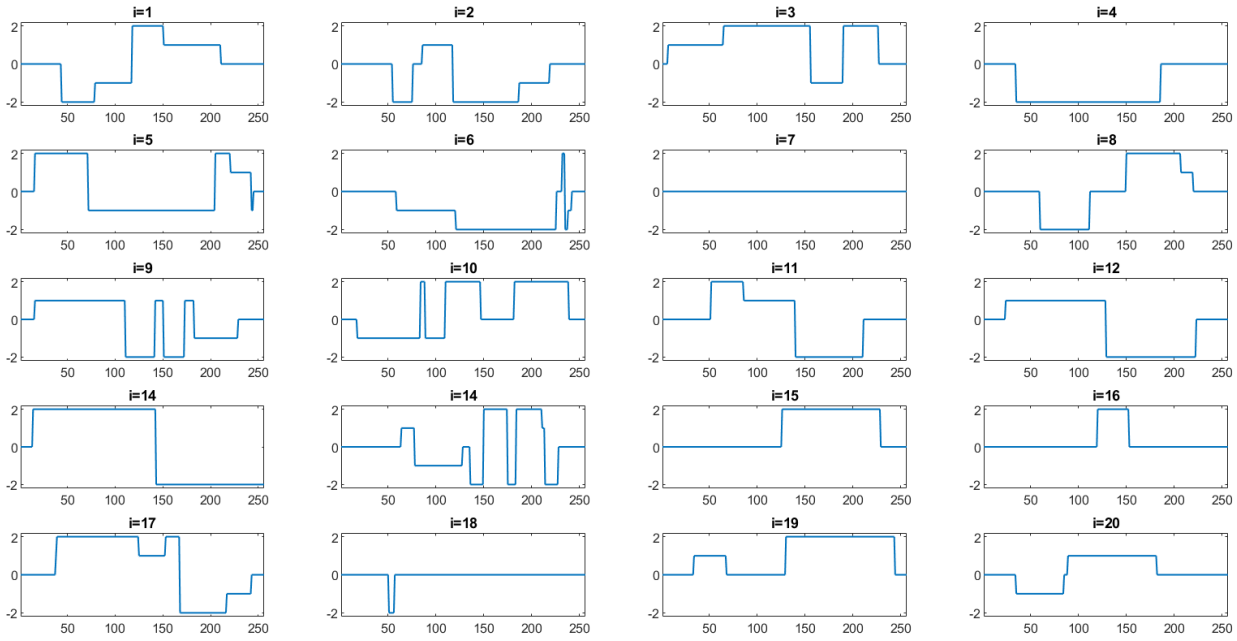
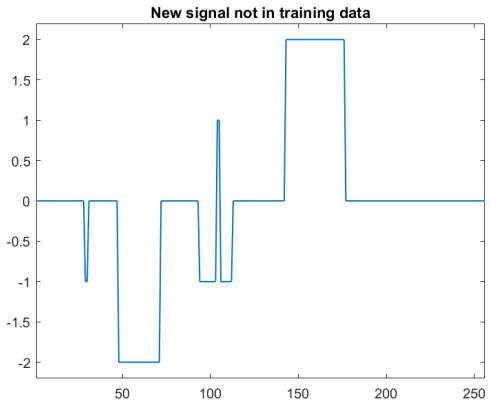
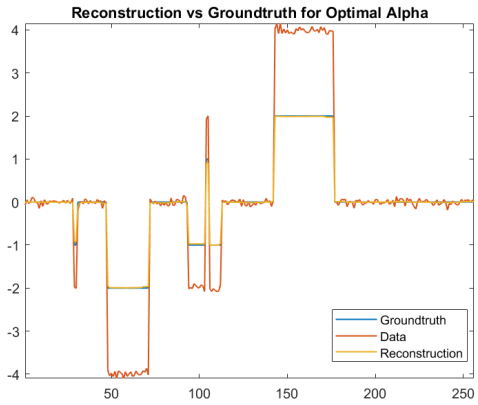


Figure 4: The training set of ground truth signals for $N = 20$.



(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.

5 Conclusions and Future Work

This strand focused on improving the efficiency of solving the square linear systems given by (6). 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 (6). 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.

There are many areas for future work for this strand. 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 (6). 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 (22). 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.

References

[1] Antonin Chambolle and Thomas Pock. "An Introduction to Continuous Optimization for Imaging". In: *Acta Numerica* 25 (2016), pp. 161–319.

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

[3] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

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

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

[6] Tuyen Trung Truong and Hang-Tuan Nguyen. "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).