Adaptive Array Signal Processing

[5SSC0]

Assignment Part 2A:

REPORT

Group number: 3

Names including ID:
1: Floris Naber 1253514
2: Shao Hsuan Hung 1723219

Date: 18/03/2023

Q1

The cost function consists of 2 terms, the first term in the least squares term and the second term is the l_2-norm. Minimizing this cost function is a trade-off between fitting the data and reducing the norm of the solution. The least square term pushes the solution towards $\underline{\hat{\mathbf{x}}} \approx \underline{\mathbf{y}}$. For the case, $\underline{\hat{\mathbf{x}}} = \underline{\mathbf{y}}$, the solution is over-fitted to the data. The l_2-norm term pushes the solution towards a magnitude of 0 ($||\underline{\hat{\mathbf{x}}}|| \approx 0$). This term makes sure the the solution does not over-fit to the data. The value λ determines to what extent we find the l_2-norm term important, compared to the least squares term, when we try to find the optimal solution. If λ is larger, then the solution will fit less to the data.

When looking at it from the denoising perspective, we have that the noisy signal consists of some samples that contain a signal and noise component, and some samples that only contain a noise component. The Tikhonov regularization (l_2-norm), which decreases the magnitude of the solution, will decrease the noisy signal samples and the pure noise samples by the same factor. If we assume that the signal component is much larger than the noise component, then after multiplying the signal with this factor, the noise terms will be closer to 0 than before, and thus the signal will appear to be denoised. It should be noted that Tikhonov regularization also compresses the signal components towards 0 with the same factor as the noise components, so the contrast between the noise and signal remains the same. Also the noise will never become 0 (since the noise is projected to the l_2 ball) and thus this method does not enforce sparsity.

Q2

We consider the cost function:

$$f(\underline{\hat{\mathbf{x}}}) = \frac{1}{2}||\underline{\hat{\mathbf{x}}} - \underline{\mathbf{y}}||_2^2 + \lambda\frac{1}{2}||\underline{\hat{\mathbf{x}}}||_2^2, \tag{1}$$

where $\underline{\mathbf{y}} \in \mathbb{R}^n$ is the noisy data and the estimate of the signal is $\underline{\hat{\mathbf{x}}} \in \mathbb{R}^n$. We can write this in vector notation as follows:

$$\begin{aligned} f(\underline{\hat{\mathbf{x}}}) &= \frac{1}{2}(\underline{\hat{\mathbf{x}}} - \underline{\mathbf{y}})^\top(\underline{\hat{\mathbf{x}}} - \underline{\mathbf{y}}) + \lambda\frac{1}{2}\underline{\hat{\mathbf{x}}}^\top\underline{\hat{\mathbf{x}}} \\ &= \frac{1}{2}(\underline{\hat{\mathbf{x}}}^\top\underline{\hat{\mathbf{x}}} - \underline{\hat{\mathbf{x}}}^\top\underline{\mathbf{y}} - \underline{\mathbf{y}}^\top\underline{\hat{\mathbf{x}}} + \underline{\mathbf{y}}^\top\underline{\mathbf{y}}) + \lambda\frac{1}{2}\underline{\hat{\mathbf{x}}}^\top\underline{\hat{\mathbf{x}}} \\ &= \frac{1}{2}(1+\lambda)\underline{\hat{\mathbf{x}}}^\top\underline{\hat{\mathbf{x}}} - \underline{\mathbf{y}}^\top\underline{\hat{\mathbf{x}}} + \frac{1}{2}\underline{\mathbf{y}}^\top\underline{\mathbf{y}}. \end{aligned} \tag{2}$$

To find the optimal solution for $\underline{\hat{\mathbf{x}}}$, we find the gradient of the cost function and set it to 0 and solve for $\underline{\hat{\mathbf{x}}}$. To find the gradient, first we define the Taylor series $\mathbb{R}^n \to \mathbb{R}$, in the point $\underline{\hat{\mathbf{x}}}$ as:

$$f(\underline{\hat{\mathbf{x}}} + \underline{\mathbf{h}}) = f(\underline{\hat{\mathbf{x}}}) + \nabla f(\underline{\hat{\mathbf{x}}})^\top\underline{\mathbf{h}} + \mathcal{O}(||\underline{\mathbf{h}}||^2), \tag{3}$$

where $h \in \mathbb{R}^n$ is a vector with a small magnitude ($\underline{\mathbf{h}} \approx \underline{\mathbf{0}}$) and $\nabla f(\underline{\hat{\mathbf{x}}}) \in \mathbb{R}^n$ is the gradient in the point $\underline{\hat{\mathbf{x}}}$. The term $\mathcal{O}(||\underline{\mathbf{h}}||^2)$ denotes the higher order terms, which will be 0, as the function of concern is a second order multivariate polynomial and we know the gradient will therefore be linear. We can now rewrite the above equation to find the derivative and substitute equation 2 into it:

$$\begin{aligned}\nabla f(\underline{\hat{\mathbf{x}}})^\top \underline{\mathbf{h}} &= f(\underline{\hat{\mathbf{x}}}+\underline{\mathbf{h}}) - f(\underline{\hat{\mathbf{x}}})\\ &= \frac{1}{2}(1+\lambda)(\underline{\hat{\mathbf{x}}}+\underline{\mathbf{h}})^\top(\underline{\hat{\mathbf{x}}}+\underline{\mathbf{h}}) - \underline{\mathbf{y}}^\top(\underline{\hat{\mathbf{x}}}+\underline{\mathbf{h}}) - \frac{1}{2}(1+\lambda)\underline{\hat{\mathbf{x}}}^\top\underline{\hat{\mathbf{x}}} + \underline{\hat{\mathbf{x}}}^\top\underline{\mathbf{y}}\\ &= \frac{1}{2}(1+\lambda)(\underline{\hat{\mathbf{x}}}^\top\underline{\hat{\mathbf{x}}} + \underline{\hat{\mathbf{x}}}^\top\underline{\mathbf{h}} + \underline{\mathbf{h}}^\top\underline{\hat{\mathbf{x}}} + \underline{\mathbf{h}}^\top\underline{\mathbf{h}}) - \underline{\mathbf{y}}^\top\underline{\mathbf{h}} - \frac{1}{2}(1+\lambda)\underline{\hat{\mathbf{x}}}^\top\underline{\hat{\mathbf{x}}}\\ &= (1+\lambda)\underline{\hat{\mathbf{x}}}^\top\underline{\mathbf{h}} + \frac{1}{2}(1+\lambda)\underline{\mathbf{h}}^\top\underline{\mathbf{h}} - \underline{\mathbf{y}}^\top\underline{\mathbf{h}}.\end{aligned} \tag{4}$$

Now as we are interested in the gradient in the point $\underline{\hat{\mathbf{x}}}$ and thus $\underline{\mathbf{h}}$ approaches $\underline{\mathbf{0}}$. Thus the term $\underline{\mathbf{h}}^\top\underline{\mathbf{h}}$ approaches 0. We then have that:

$$\begin{aligned}\nabla f(\underline{\hat{\mathbf{x}}})^\top \underline{\mathbf{h}} &= (1+\lambda)\underline{\hat{\mathbf{x}}}^\top\underline{\mathbf{h}} - \underline{\mathbf{y}}^\top\underline{\mathbf{h}}\\ &= \big((1+\lambda)\underline{\hat{\mathbf{x}}} - \underline{\mathbf{y}}\big)^\top\underline{\mathbf{h}}\\ \implies \nabla f(\underline{\hat{\mathbf{x}}}) &= (1+\lambda)\underline{\hat{\mathbf{x}}} - \underline{\mathbf{y}}.\end{aligned} \tag{5}$$

Now that we have the gradient, we simply set it to $\underline{\mathbf{0}}$, as we know the minimum cost is achieved when the gradient is zero, because it is an unconstrained convex optimization problem. Now we solve for $\underline{\hat{\mathbf{x}}}$:

$$\begin{aligned}\nabla f(\underline{\hat{\mathbf{x}}}) = \underline{\mathbf{0}} &= (1+\lambda)\underline{\hat{\mathbf{x}}} - \underline{\mathbf{y}}\\ \implies (1+\lambda)\underline{\hat{\mathbf{x}}} &= \underline{\mathbf{y}}\\ \implies \underline{\hat{\mathbf{x}}} &= \frac{1}{1+\lambda}\underline{\mathbf{y}}.\end{aligned} \tag{6}$$

Q3

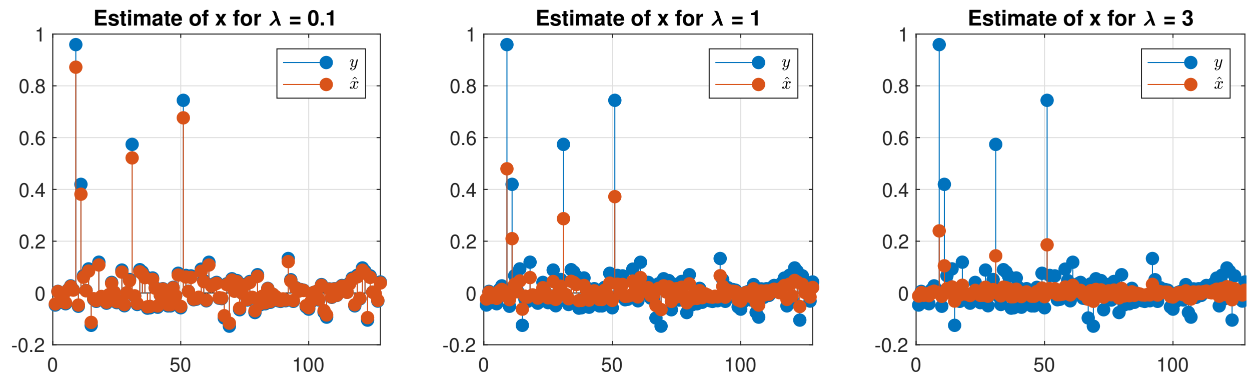


Figure 1: Tikhonov regularization for different λ, for a sparse signal with Gaussian noise.

From equation 6, it can be seen that the optimal solution is simply the noisy signal multiplied by a factor $\frac{1}{1+\lambda}$. If there is no Tikhonov regularization ($\lambda = 0$), then $\underline{\hat{\mathbf{x}}} = \underline{\mathbf{y}}$, which is the least squares solution. If $\lambda \to \infty$, we have that the only relevant term is the Tikhonov regularization term and thus $\underline{\hat{\mathbf{x}}} \to \underline{\mathbf{0}}$.

For $\lambda > 0$, we have that $0 < \frac{1}{1+\lambda} < 1$ and thus $\underline{\mathbf{0}} < \underline{\hat{\mathbf{x}}} < \underline{\mathbf{y}}$. The larger λ is, the smaller $\frac{1}{1+\lambda}$ is. The whole noisy signal is suppressed by this factor, and thus for larger λ there is more suppression of the noise, but also suppression of the signal components. In figure 1 this relation can be seen. It can be seen that both the noise and signal components are suppressed and also that the noise does not fully disappear using Tikhonov regularization, also for large λ.

Q4

The difficulty in finding the optimal solution for a cost function with the l_1-norm is that the l_1-norm has terms with absolute values, which are not differentiable. However, we can separate these terms in different domains and calculate the sub-differentials of the l_1 norm. To deal with this problem we define the vectors $\underline{\hat{\mathbf{x}}} = \begin{bmatrix}\hat{x}_1 & \dots & \hat{x}_n\end{bmatrix}^\top \in \mathbb{R}^n$ and $\underline{\mathbf{y}} = \begin{bmatrix}y_1 & \dots & y_n\end{bmatrix}^\top \in \mathbb{R}^n$. We then write the cost function as a multivariate equation, not in vector notation:

$$\begin{aligned} f(\underline{\hat{\mathbf{x}}}) &= \frac{1}{2}||\underline{\hat{\mathbf{x}}} - \underline{\mathbf{y}}||_2^2 + \lambda||\underline{\hat{\mathbf{x}}}||_1 \\ &= \sum_{i=1}^{n} \frac{1}{2}(\hat{x}_i - y_i)^2 + \lambda|\hat{x}_i|. \end{aligned} \tag{7}$$

The gradient is defined by $\nabla f(\underline{\hat{\mathbf{x}}}) = \begin{bmatrix}\frac{\partial f(\underline{\hat{\mathbf{x}}})}{\partial \hat{x}_1} & \dots & \frac{\partial f(\underline{\hat{\mathbf{x}}})}{\partial \hat{x}_n}\end{bmatrix}^\top \in \mathbb{R}^n$. Now let us define the derivative of $|\hat{x}_i|$ as a set of sub-differentials as follows:

$$\frac{d|\hat{x}_i|}{d\hat{x}_i} = \begin{cases} \text{sign}(\hat{x}_i), & \text{if } \hat{x}_i \neq 0 \\ [-1,1], & \text{if } \hat{x}_i = 0, \end{cases} \tag{8}$$

where sign$(\hat{x}_i)$ is the sign function and where [-1,1] denotes an undefined value that is in the interval -1 to and including 1, as the derivative in the point $\hat{x}_i = 0$ is undefined. Now let us solve the partial derivative $\frac{\partial f(\underline{\hat{\mathbf{x}}})}{\partial \hat{x}_i} \equiv \nabla f(\underline{\hat{\mathbf{x}}})_i$:

$$\begin{aligned} \nabla f(\underline{\hat{\mathbf{x}}})_i &= \hat{x}_i - y_i + \lambda\frac{\partial|\hat{x}_i|}{\partial \hat{x}_i} \\ &= \begin{cases} \hat{x}_i - y_i + \lambda \cdot \text{sign}(\hat{x}_i), & \text{if } \hat{x}_i \neq 0 \\ \hat{x}_i - y_i + \lambda[-1,1], & \text{if } \hat{x}_i = 0 \end{cases} \\ &= \begin{cases} \hat{x}_i - y_i + \lambda, & \text{if } \hat{x}_i > 0 \\ \hat{x}_i - y_i - \lambda, & \text{if } \hat{x}_i < 0 \\ -y_i + [-\lambda, \lambda], & \text{if } \hat{x}_i = 0. \end{cases} \end{aligned} \tag{9}$$

The solution is obtained when the gradient is zero: $\nabla f(\underline{\hat{\mathbf{x}}}) = \underline{\mathbf{0}}$. From this it follows that the optimal solution is reached when $\nabla f(\underline{\hat{\mathbf{x}}})_i = 0$ for $i = \{1, \ \dots, \ n\}$. For the case $\hat{x}_i = 0$, we can find the relation between y_i and λ by formally writing the optimality condition as:

$$0 \in -y_i + [-\lambda, \lambda] \implies y_i \in [-\lambda, \lambda] \implies |y_i| \leq \lambda. \tag{10}$$

For the cases where $x_i \neq 0$, the solution is straightforward:

$$\begin{aligned} \nabla f(\underline{\hat{\mathbf{x}}})_i &= 0 = \hat{x}_i - y_i + \lambda && \implies && \hat{x}_i = y_i - \lambda, && \text{if } \hat{x}_i > 0 \\ \nabla f(\underline{\hat{\mathbf{x}}})_i &= 0 = \hat{x}_i - y_i - \lambda && \implies && \hat{x}_i = y_i + \lambda, && \text{if } \hat{x}_i < 0. \end{aligned} \tag{11}$$

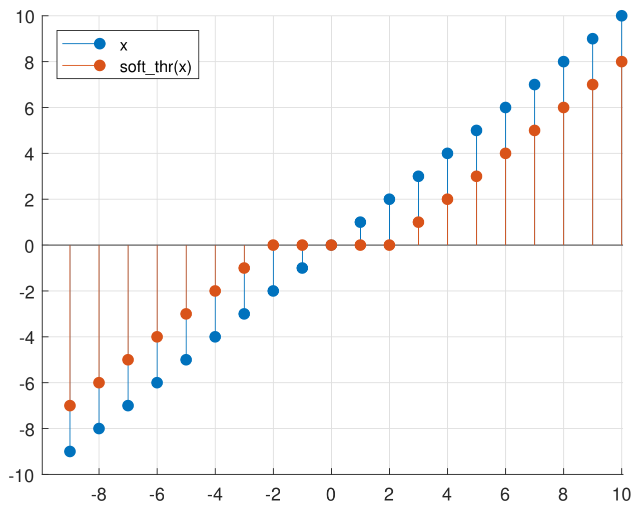
Thus we have the set of solutions for $\hat{x}_i$, where we can substitute the solutions for $\hat{x}_i$ into the conditional statements to find the relation between λ and y_i:

$$\hat{x}_i = \begin{cases} y_i + \lambda, & \text{if } \hat{x}_i < 0 \\ 0, & \text{if } \hat{x}_i = 0 \\ y_i - \lambda, & \text{if } \hat{x}_i > 0 \end{cases} \implies \hat{x}_i = \begin{cases} y_i + \lambda, & \text{if } y_i + \lambda < 0 \\ 0, & \text{if } |y_i| \leq \lambda \\ y_i - \lambda, & \text{if } y_i - \lambda > 0. \end{cases} \tag{12}$$

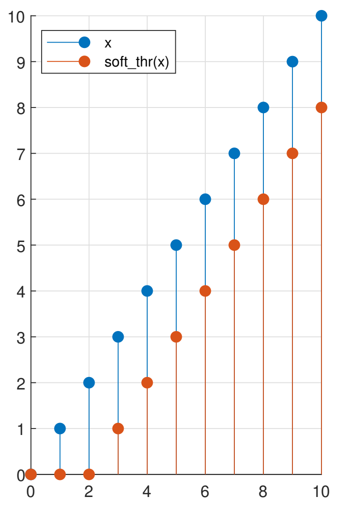
Now that we have a solution for all elements in the vector $\underline{\hat{\mathbf{x}}}$, we can write the solution back in vector notation:

$$\underline{\hat{\mathbf{x}}} = \begin{cases} \underline{\mathbf{y}} + \lambda, & \text{if } \underline{\mathbf{y}} < -\lambda \\ 0, & \text{if } |\underline{\mathbf{y}}| \leq \lambda \\ \underline{\mathbf{y}} - \lambda, & \text{if } \underline{\mathbf{y}} > \lambda. \end{cases} \tag{13}$$

It should be noted that these conditions hold element-wise and thus the calculations should all be performed element-wise, which is not immediately obvious from the above equation.



(a) real values



(b) complex values

Figure 2: Soft-thresholding for $\lambda = 2$.

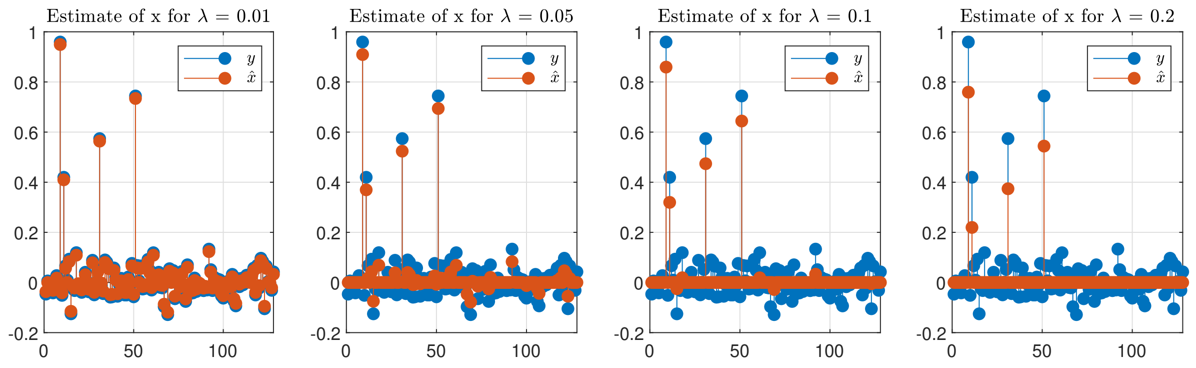


Figure 3: Soft-thresholding for different values of λ, for a sparse signal with Gaussian noise.

Q5

The implementation of the soft-thresholding function is shown below:

```
function [x_thr] = soft_thresholding(y, lambda)
    x_thr = zeros(length(y),1);
    x_thr = (abs(y)-lambda) .* exp(1i*angle(y)) .*(abs(y)>lambda);
end
```

It can be seen from figure 2, that the soft-thresholding pushes the measurements closer to 0, but every element of the signal is pushed to 0 with the same value of λ, whereas with the l_2-norm, every component was pushes to 0 with an equal factor (bigger value results in bigger step size). With the l_1-norm, the values of the signal that are below the threshold λ are set to 0. After denoising, the signal contains more exact 0-values, thus enforcing sparsity.

Q6

It can be seen in figure 3 how the soft thresholding function works on a sparse signal with small Gaussian noise. It can be seen that for larger values of λ the noise is suppressed more and thus the sparsity is enforced. Meanwhile, the signal components are also decreased in magnitude.

Q7

The l_0-norm is another regularization term that can be used to enforce sparsity, as this norm counts the amount of non-zero values in a vector. The problem with using this term is that the cost function that includes an l_0-norm is not convex and NP-hard. This makes it hard to find the optimal solution. For sparse signal reconstruction, the l_1-norm is better than the

l_2-norm, because it enforces sparsity by soft-thresholding. After reconstruction the signal will be more sparse than before reconstruction. For the l_2-norm, this is not the case.

The better performance of the l_1-norm, compared to the l_2-norm, can be seen visually when comparing figure 1 and figure 3. For figure 1, the l_2-norm pushes the solution closer to zero with increasing λ, but the signal is also compressed more (note that it will also not become more sparse). For figure 3, the l_1-norm pushes the solution also closer to 0 with increasing λ, while the signal components of the measurement are compressed much less. Components close to 0 will actually be set to 0 and the signal becomes more sparse. To be brief, the l_1-norm is the better choice, because the l_1-norm enforces sparsity, unlike thel_2-norm.

Q8

Advantages of reducing sampling rate is that the memory and power requirements of the system are lower. Especially for wireless sensors, more bandwidth on communication channels is preserved if a sensor samples less.

Q9

In short, it is not possible to find a solution analytically, because the system is undetermined. There are more unknowns ($\hat{\underline{\mathbf{x}}}$) than measurements ($\underline{\mathbf{y}}$). As a result, there are multiple possible solutions that minimize the cost function.

When looking at it from the mathematical perspective we also arrive at the same conclusion: Let us consider that it may be possible to have a closed form solution and that the optimality condition is that the gradient of the cost function is 0 in the optimal point (unconstrained problem). Now we use the Taylor expansion again (same as equation 3) to find the gradient of the cost function of the differentiable part (excluding the l_1-norm). The cost function is defined as:

$$\begin{aligned} f(\hat{\underline{\mathbf{x}}}) &= \frac{1}{2}||\mathbf{A}\hat{\underline{\mathbf{x}}} - \underline{\mathbf{y}}||_2^2 + \lambda\frac{1}{2}||\hat{\underline{\mathbf{x}}}||_1 \\ f(\hat{\underline{\mathbf{x}}}) &= f_1(\hat{\underline{\mathbf{x}}}) + f_2(\hat{\underline{\mathbf{x}}}), \end{aligned} \tag{14}$$

where $f_1(\hat{\underline{\mathbf{x}}}) = \frac{1}{2}||\mathbf{A}\hat{\underline{\mathbf{x}}} - \underline{\mathbf{y}}||_2^2$ and $f_2(\hat{\underline{\mathbf{x}}}) = \lambda\frac{1}{2}||\hat{\underline{\mathbf{x}}}||_1$. We then have that the gradient of the differentiable part ($f_1(\hat{\underline{\mathbf{x}}})$) is described by (we know this gradient is linear):

$$\begin{aligned} \nabla f_1(\hat{\underline{\mathbf{x}}})^\top \underline{\mathbf{h}} &= f_1(\hat{\underline{\mathbf{x}}} + \underline{\mathbf{h}}) - f_1(\hat{\underline{\mathbf{x}}}) \\ &= \frac{1}{2}||\mathbf{A}(\hat{\underline{\mathbf{x}}} + \underline{\mathbf{h}}) - \underline{\mathbf{y}}||_2^2 - \frac{1}{2}||\mathbf{A}\hat{\underline{\mathbf{x}}} - \underline{\mathbf{y}}||_2^2 \\ &= \frac{1}{2}\left(\mathbf{A}(\hat{\underline{\mathbf{x}}} + \underline{\mathbf{h}}) - \underline{\mathbf{y}}\right)^\top\left(\mathbf{A}(\hat{\underline{\mathbf{x}}} + \underline{\mathbf{h}}) - \underline{\mathbf{y}}\right) - \frac{1}{2}\left(\mathbf{A}\hat{\underline{\mathbf{x}}} - \underline{\mathbf{y}}\right)^\top\left(\mathbf{A}\hat{\underline{\mathbf{x}}} - \underline{\mathbf{y}}\right) \\ &= \frac{1}{2}\left(\underline{\mathbf{h}}^\top\mathbf{A}^\top\mathbf{A}\hat{\underline{\mathbf{x}}} + \hat{\underline{\mathbf{x}}}^\top\mathbf{A}^\top\mathbf{A}\underline{\mathbf{h}} + \underline{\mathbf{h}}^\top\mathbf{A}^\top\mathbf{A}\underline{\mathbf{h}} - \underline{\mathbf{y}}^\top\mathbf{A}\underline{\mathbf{h}} - \underline{\mathbf{h}}^\top\mathbf{A}^\top\underline{\mathbf{y}}\right). \end{aligned} \tag{15}$$

We are interested in the gradient in the point $\hat{\underline{\mathbf{x}}}$ and thus $\underline{\mathbf{h}}$ approaches $\underline{\mathbf{0}}$. Thus the constant term $\underline{\mathbf{h}}^\top \mathbf{A}^\top \mathbf{A}\underline{\mathbf{h}}$ approaches 0. We then have that:

$$\begin{aligned} \nabla f_1(\hat{\underline{\mathbf{x}}})^\top \underline{\mathbf{h}} &= \frac{1}{2}\big(\underline{\mathbf{h}}^\top \mathbf{A}^\top \mathbf{A}\hat{\underline{\mathbf{x}}} + \hat{\underline{\mathbf{x}}}^\top \mathbf{A}^\top \mathbf{A}\underline{\mathbf{h}} - \underline{\mathbf{y}}^\top \mathbf{A}\underline{\mathbf{h}} - \underline{\mathbf{h}}^\top \mathbf{A}^\top \underline{\mathbf{y}}\big) \\ &= \hat{\underline{\mathbf{x}}}^\top \mathbf{A}^\top \mathbf{A}\underline{\mathbf{h}} - \underline{\mathbf{y}}^\top \mathbf{A}\underline{\mathbf{h}} \\ \implies \nabla f_1 &= \hat{\underline{\mathbf{x}}}^\top \mathbf{A}^\top \mathbf{A} - \underline{\mathbf{y}}^\top \mathbf{A}. \end{aligned} \tag{16}$$

The gradient $\nabla f_2(\hat{\underline{\mathbf{x}}})$ can be calculated in a similar way as in Q4, but we leave it as it is, as we can demonstrate without calculating it if the solution is closed form. This can be done because $\nabla f_2(\hat{\underline{\mathbf{x}}})$ does not depend on the magnitude or direction of $\hat{\underline{\mathbf{x}}}$, but only the sign. Combining the gradients and setting it to 0 to get the optimal solution $\hat{\underline{\mathbf{x}}}^*$, we have that:

$$\begin{aligned} \nabla f(\hat{\underline{\mathbf{x}}}^*) = 0 &= \nabla f_1(\hat{\underline{\mathbf{x}}}^*) + \nabla f_2(\hat{\underline{\mathbf{x}}}^*) \\ 0 &= \underline{\hat{\mathbf{x}^*}}^\top \mathbf{A}^\top \mathbf{A} - \underline{\mathbf{y}}^\top \mathbf{A} + \nabla f_2(\hat{\underline{\mathbf{x}}}^*) \\ \implies \mathbf{A}^\top \mathbf{A}\hat{\underline{\mathbf{x}}}^* &= \mathbf{A}^\top \underline{\mathbf{y}} - \nabla f_2(\hat{\underline{\mathbf{x}}}^*) \\ \implies \hat{\underline{\mathbf{x}}}^* &= \big(\mathbf{A}^\top \mathbf{A}\big)^{-1}\big(\mathbf{A}^\top \underline{\mathbf{y}} - \nabla f_2(\hat{\underline{\mathbf{x}}}^*)\big). \end{aligned} \tag{17}$$

From this equation it can be seen that this equation cannot be calculated. Since $\mathbf{A} \in \{0,1\}^{M\times N}$ and $M < N$, which contain exactly one 1 per row (the rest elements are zeros, and all rows are unique. Then we have that $\mathbf{A}^\top \mathbf{A} \in \{0,1\}^{N\times N}$ and because at least 1 column in $\mathbf{A}$ contains 0's only, we have that at least 1 column and 1 row in $\mathbf{A}^\top \mathbf{A}$ contains only 0's, which makes this matrix not invertible. Thus we can not find a closed-form solution for this optimization problem.

Q10

As shown in figure 4, the reconstructed signal is periodic, with a period of 32 samples, since we sample 32 equi-spaced components in the frequency domain. Undersampling (equidistantly) in the frequency domain leads to aliasing of the original signal in the time domain. It is

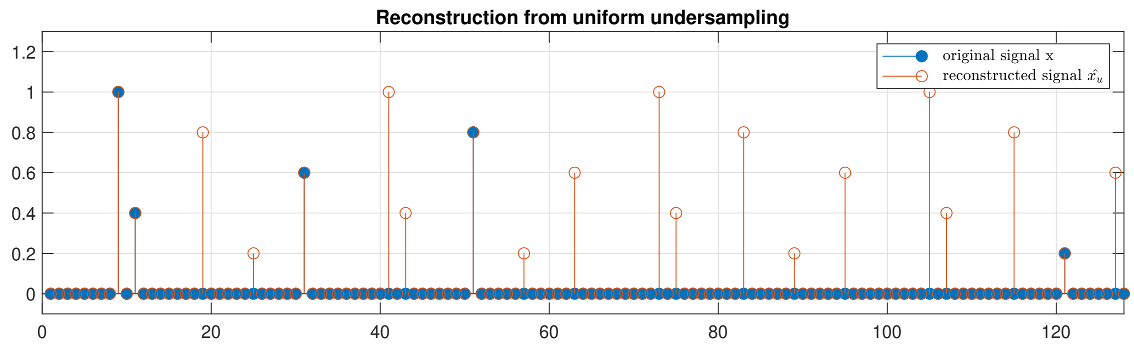


Figure 4: Reconstruction of $\underline{\mathbf{x}}$ after uniform undersampling in frequency domain.

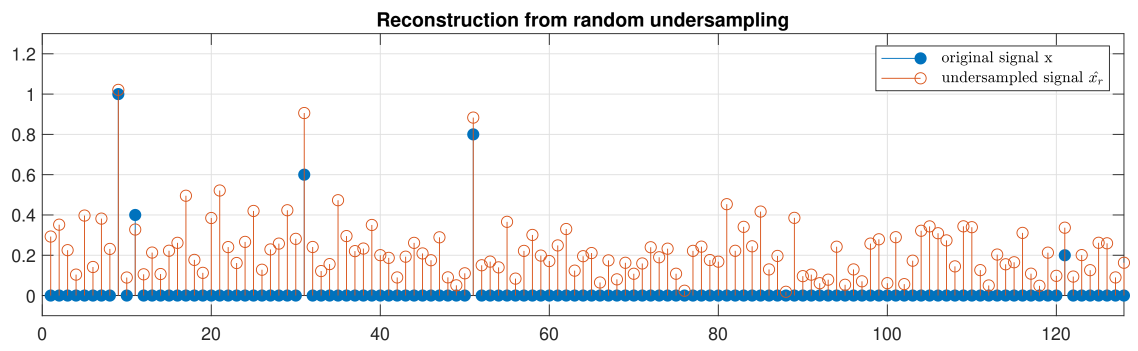


Figure 5: Reconstruction of $\underline{\mathbf{x}}$ after random undersampling in frequency domain.

impossible to reconstruct the original signal as there is no way to distinguish from which of the 4 32-sample periods the original signal components came from.

Q11

As can be seen in figure 5, for non-uniform undersampling the large signal components of the original measurement can still be distinguished, as the aliasing is gone, although their exact height estimation is incorrect. However, due to the non-uniform undersampling, a noise component in the time domain is now visible for all samples, making it hard to distinguish the low amplitude signal components from the noise components. This is very similar to the denoising problem, because there we also consider a sparse signal that is cluttered with noise after measuring. Non-uniform undersampling in the frequency domain for a sparse signal in the time domain leads to a very similar problem as the denoising problem, thus we can use denoising techniques to attempt to recover the original signal (at least partly), such as an l_1-norm (soft-threholding).

Q12

The implementation of the POCS algorithm is shown in the following code and the result is shown in figure 6. The input `X` is the non-zero-filled undersampled measurement (`A*fft(x')`, where `x` is the generated sparse test signal). To make it zero-filled we can multiply it with the transpose of the measurement matrix `A`: `A'*X`.

```
function [x_hat_new] = pocs(X, A, lambda, eps)
    i=0; X0 = A'*X;
    x_hat_new=zeros(size(X0); x_hat_prev=zeros(size(X0)); X_hat = X0;
    while (i == 0 || norm(x_hat_new-x_hat_prev)>eps)
        x_hat_prev = x_hat_new;
        x_hat_new = soft_thresholding(ifft(X_hat),lambda);
        X_hat = fft(x_hat_new); X_hat(X0~=0) = X0(X0~=0);
        i = i+1;
    end
end
```

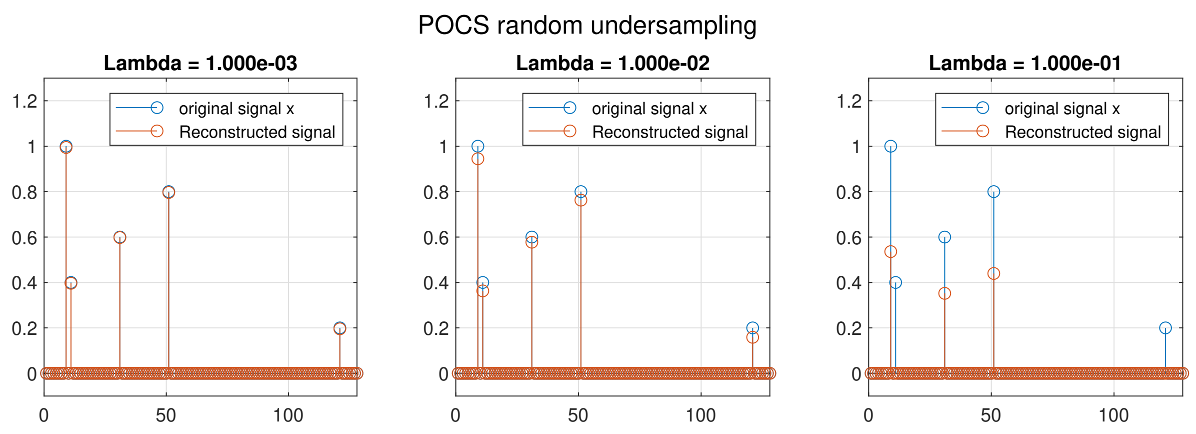


Figure 6: Reconstruction of $\underline{\mathbf{x}}$ using the POCS algorithm for different values of λ.

Q13

The implementation of the ISTA algorithm is shown in the following code and the result is shown in figure 7. For this algorithm there is a multiplication with `size(A,2)` in the update step. This is because in the original ISTA algorithm there is a multiplication with F^H, the conjugate transpose of the matrix transformation corresponding to a discrete Fourier transform. As $F^H = F^{-1} \cdot N$, where $F \in \mathbb{C}^{N \times N}$, we have that we can use the Matlab function `ifft()` as F^-1 and `size(A,2)` as N.

```
function [x_hat_new] = ISTA(X, A, lambda, beta, eps)
    i = 0; x_hat_prev = 4.*ifft(A'*X); x_hat_new = x_hat_prev;
    while (i == 0 || norm(x_hat_new-x_hat_prev)>eps)
        x_hat_prev = x_hat_new;
        x_hat_new = x_hat_prev - beta*ifft(A'*(A*fft(x_hat_prev)-X))*size(A,2);
        x_hat_new = soft_thresholding(x_hat_new,lambda);
        i=i+1;
    end
end
```

Q14

The higher the value of λ, the more noise is suppressed as mentioned in Q6. Since larger values of λ means a wider range of noise is suppressed, it can happen that signal components are suppressed too much, if λ is chosen too large. A smaller value of λ is therefore desired to get an accurate reconstruction. However, a smaller value of λ means that update steps of the iterative algorithms are smaller and as a result it takes much longer to converge.

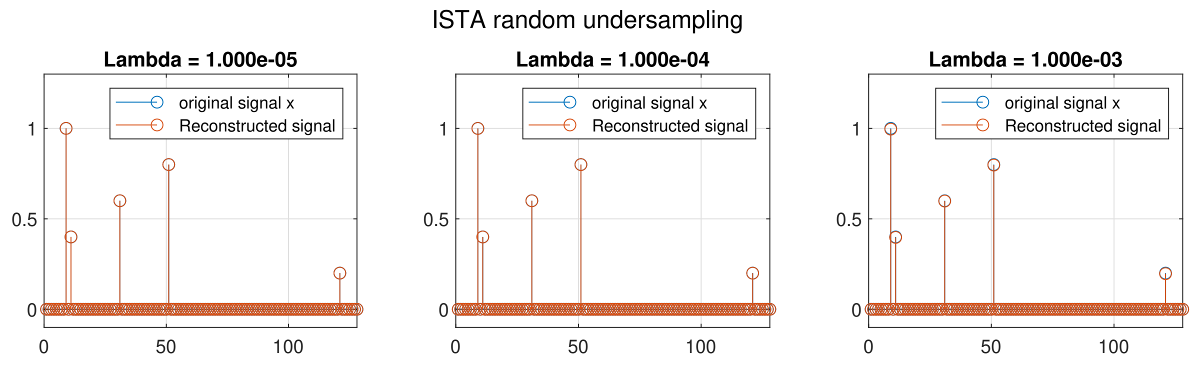


Figure 7: Reconstruction of $\underline{\mathbf{x}}$ using the ISTA algorithm for different values of λ.

Q15

The biggest conceptual algorithmic difference between the POCS and the ISTA algorithm is how the algorithms iteratively calculate the reconstruction of the sparse signals. The POCS algorithm iteratively sparsifies the reconstructed signal and then makes sure the reconstruction of the signal adheres to the sparsely measured frequency data, until convergence. On the other hand, the ISTA algorithm iteratively calculates the proximal gradient of a cost function that consists of a sparsifying norm (l_1-norm) and the Euclidean norm between the measured data and the reconstruction. The algorithm alternatingly steps in the negative gradient direction of both norms, until convergence.

Q16

The most important assumption that must hold for compressed sensing (the form $y = A\Psi x$) to be successful, is that the desired signal (x) must be sparse in some domain (with sparsifying basis Ψ, square matrix, such that Ψx is sparse). The second most important assumption that must hold is that the measurement matrix (A, matrix with much more columns than rows, such that y in $y = A\Psi x$ is much smaller than x) should satisfy the restricted isometry property (RIP), which means that the matrix should be nearly orthonormal, (and have only non-zero elements,) when operating on sparse vectors (The example used in this report is a random combination of frequency components).

Q17

We propose two efficient methods, the first one is more complicated to program, but is approximately twice as fast in execution due to less matrix operation. The first method is creating a zero-vector that contains the same amount of elements as the measurement matrix, then randomly setting an element to 1 in every vector, that corresponds to different columns in the measurement matrix and finally reshaping the vector to a matrix. The second

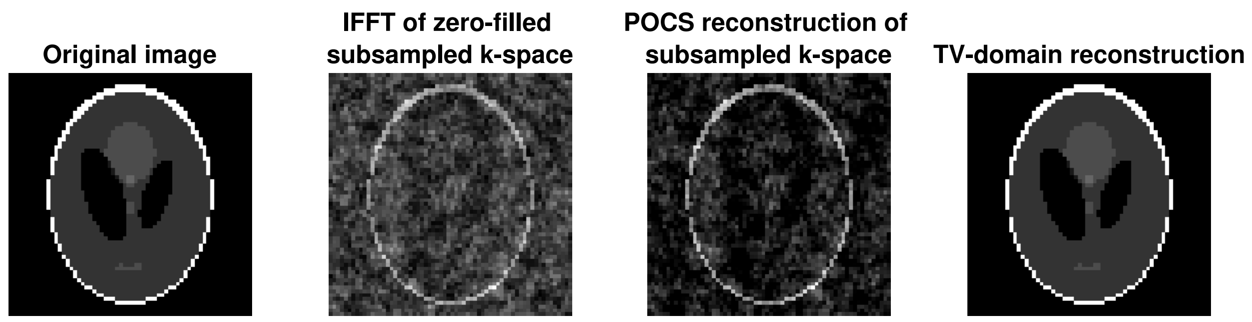

Original image
IFFT of zero-filled subsampled k-space
POCS reconstruction of subsampled k-space
TV-domain reconstruction

Figure 8: Comparison of original image (left), IFFT of zero-filled subsampled K-space (left middle), POCS reconstruction of subsampled k-space (right middle) and TV based reconstruction (right).

method is first creating the measurement matrix first by concatenating an identity matrix with a zero matrix and then randomly ordering the columns of this matrix.

Method 1:

```
M = 1024;
A_1d = zeros(1,M*64^2); % Make a 1d array with length M*64^2
A_1d((0:M-1)*64^2 + randperm(64^2,M)) = 1; % randomly set an element to 1 in each row
A = reshape(A_1d,[64^2,M])'; % create matrix A
```

Method 2:

```
M = 1024;
A = [eye(M) zeros(M,64^2-M)]; % create matrix A
A = A(:,randperm(64^2)); % randomize matrix A
```

Q18

The implementation of randomly undersampling the k-space in 1D and subsequently reconstructing the image in 2D can be seen below and the resulting image is shown in figure 8. It can be seen that this method of reconstructing directly after undersampling in k-space is not a successful method for compressed sensing, as many image features are lost using this method.

```
im = phantom('Modified Shepp-Logan', 64); im_k = fft2(im); % k-space
% Steps: flatten > sample > reshape > ifft > scale correction:
invReconstruction = 64^2/M*ifft2(reshape(A'*(A*im_k(:)),64,64));
```

Q19

The implementation of TV based reconstruction image is shown below, and the resulting image is shown in figure 8. It can clearly be seen that this image reconstruction is very accurate. The differences in reconstruction quality between the inverse Fourier reconstruction and TV-based reconstruction is explained by the sparsity assumption. For the Fourier reconstruction, we have that a similar thing occurs as in Q11, namely, the image gets cluttered with noise and only very strong signal components get preserved (white oval), the rest features disappear with the noise. Furthermore we have that this Fourier reconstruction relies on the fact that the image is sparse, but this phantom image is not very sparse and thus reconstructing the image with a compressed sensing technique will not be very successful. When using a 2D version of the POCS algorithm (see Appendix A for implementation, see figure 8 for the reconstruction), we can see that the reconstruction is better than the naive Fourier reconstruction, as it preserves slightly more image features. However, it is still not very accurate, demonstrating that the image is not very sparse as POCS is optimized for sparse signals. On the other hand, the TV-based reconstruction uses the total variation domain as the sparsifying domain instead of the image domain. The image is actually sparse in this TV domain, so image reconstruction using the l_1-norm in this domain can successfully be used.

```
N = 64; F = dftmtx(N); lambda = 0.1;
y = A*(im_k(:)); % Measurment data y in k-space
cvx_begin
  cvx_precision low
  variable imhat(N,N)
  Dx_imhat = imhat(:,2:end)-imhat(:,1:end-1);
  Dy_imhat = imhat(2:end,:)-imhat(1:end-1,:);
  F_imhat = F*imhat*F;
  minimize(norm(A*F_imhat(:)-y)+lambda*(norm(Dx_imhat(:),1)+norm(Dy_imhat(:),1)) );
cvx_end
TVreconPhantom = imhat;
```

Q20

The results of the reconstructed image using Fourier based and TV based reconstruction are shown in figure 9. The inverse Fourier transform of the zero-filled undersampled image in k-space looks noisy, as with the earlier signals using this method. The TV based reconstruction image can reconstruct the original image better, but lacks some of the details. The reason that the TV based reconstruction can not reconstruct the image very well, is simply because the MRI image has a lot of variation, especially in the cerebral cortex part. Thus the image is not sparse in the TV domain for this low resolution (64 by 64 pixel), unlike the phantom image used earlier. It can be seen that the loss of detail results in the reconstruction having

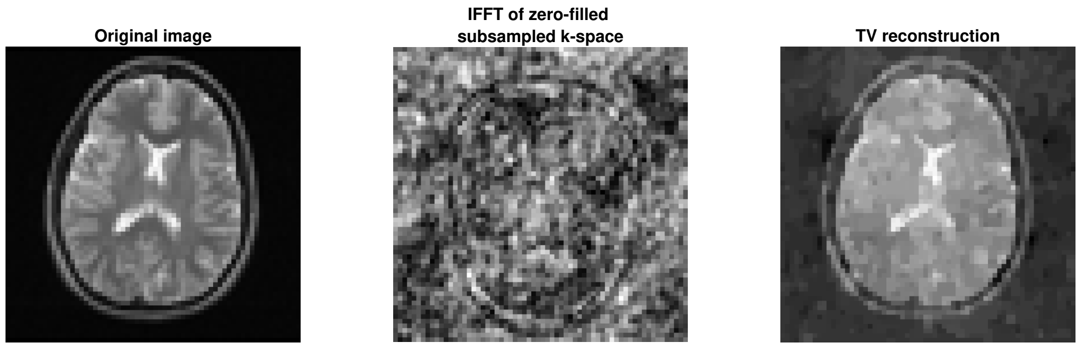


Figure 9: Comparison of original image, IFFT of zero-filled subsampled K-space, and TV based reconstruction.

less variation. This makes sense, given that the cost function of the optimization problem contains an l_1-norm in the TV domain and as a result the solution is pushed towards a sparse solution in this domain.

Q21

One of the domains in which MRI images are sparse is in the wavelet domain[1]. As an MRI image is sparse in this domain, it can be used as a sparsifying domain, for reconstruction after compressed sensing.

Q22

If we want to use the POCS algorithm with the wavelet domain as the sparsifying domain, besides using the 2D Fourier transform and its inverse instead of the 1D transforms (see Appendix A for implementation), we need to make 2 other major changes to the POCS algorithm. Firstly, instead of uniformly random sampling the k-space (like the first POCS implementation), we can use different distribution such as radial sampling or zig-zag sampling to sample the k space. Furthermore the second major change that needs to be made is that the image needs to be transformed into the wavelet domain when soft-thresholding, as that is the sparsifying domain. After soft-thresholding we convert the image back to k-space for the data consistency step.

References

[1] N. Dwork, D. O'Connor, C. A. Baron, E. M. I. Johnson, A. B. Kerr, J. M. Pauly, and P. E. Z. Larson, "Utilizing the wavelet transform's structure in compressed sensing," *Signal, Image and Video Processing*, vol. 15, no. 7, pp. 1407–1414, mar 2021.

Appendix A

POCS algorithm for general 2D data:

```
im = phantom('Modified Shepp-Logan', 64); im_k=fft2(im);
y = A*im_k(:); % sparse measurement in k-space
eps = 1e-10; lambda = 0.01; % POCS variables
x_hat_pocs = pocs2d(y, A, lambda, eps); % see function below
pocsReconstruction = reshape(x_hat_pocs,64,64); % reshape to image

%% POCS ALGORITHM FOR IMAGE DATA %%
function [x_hat_new] = pocs2d(X, A, lambda, eps)
    i=0; x_hat_new=0; x_hat_prev=0; X0 = reshape(A'*X,64,64); X_hat = X0;
    while (i == 0 || norm(x_hat_new-x_hat_prev)>eps)
        x_hat_prev = x_hat_new;
        x_hat_temp = ifft2(X_hat);
        x_hat_new  = soft_thresholding(x_hat_temp(:),lambda);
        X_hat = fft2(reshape(x_hat_new,64,64));
        X_hat(X0~=0) = X0(X0~=0);
        i = i+1;
    end
end
```