Assignment 4

Callum Lau, 19102521

Numerical Optimisation

April 15, 2020

1 EXERCISE 2

1.1 (a)

We have the following constraint minimisation problem

$$\begin{aligned} \min_{(x,y)} \quad & f(x,y) = (x-2y)^2 + (x-2)^2 \\ \text{s.t.} \quad & x - y = 4 \end{aligned}$$

which we can write as

$$\begin{aligned} \min_{(\boldsymbol{x})} \quad & \frac{1}{2}\boldsymbol{x}^\top P \boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x} + r \\ \text{s.t.} \quad & A\boldsymbol{x} = b \end{aligned}$$

where $P = \begin{pmatrix} 4 & -4 \\ -4 & 8 \end{pmatrix}$, $\boldsymbol{q} = \begin{pmatrix} -4 \\ 0 \end{pmatrix}$, $A^\top = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$, $b = 4$ and $r = 0$. Hence following the formula in the lectures the KKT system is formulated as:

$$\begin{bmatrix} 4 & -4 & 1 \\ -4 & 8 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x^* \\ y^* \\ \nu^* \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 4 \end{bmatrix} \tag{1}$$

1.2 (b)

The KKT matrix has determinant -4 hence it is non-singular and the system has a unique solution. We solve this system by first finding the null space N of $A \in \mathbb{R}^{1\times 2}$ of dimension $\mathbb{R}^{2\times 1}$. Note that $A\boldsymbol{x} = \boldsymbol{0} \iff \begin{pmatrix} 1 \\ -1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0 \iff x_1 = x_2$. Hence letting $N = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \in \text{span}\{N(A)\}$, and $\hat{x} = \begin{pmatrix} 6 \\ 2 \end{pmatrix}$ be a solution to $A\boldsymbol{x} = \boldsymbol{b}$, then we solve the unconstrained problem

$$\min_z f(Nz + \hat{x}) = \min_z f(z+6, z+2) = \min_z (2-z)^2 + (z+4)^2$$

which by differentiating gives minimum $z^* = -1$. The solution $\boldsymbol{x}^*$ is given by

$$\boldsymbol{x}^* = N\boldsymbol{z}^* + \hat{x} = -\begin{pmatrix}1\\1\end{pmatrix} + \begin{pmatrix}6\\2\end{pmatrix} = \begin{pmatrix}5\\1\end{pmatrix}$$

This also gives the value of $\nu^* = -12$.

2 EXERCISE 4

Firstly, the original sparse signal x_0 was created and is displayed below.

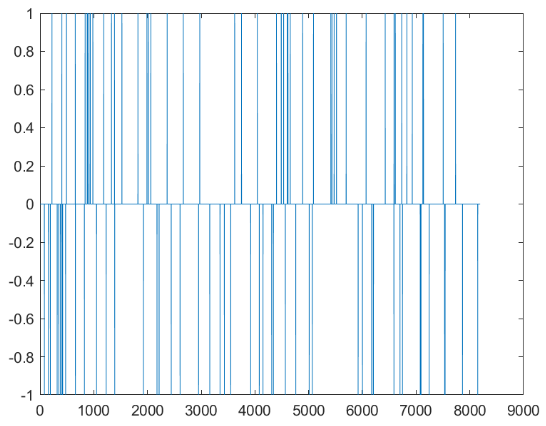
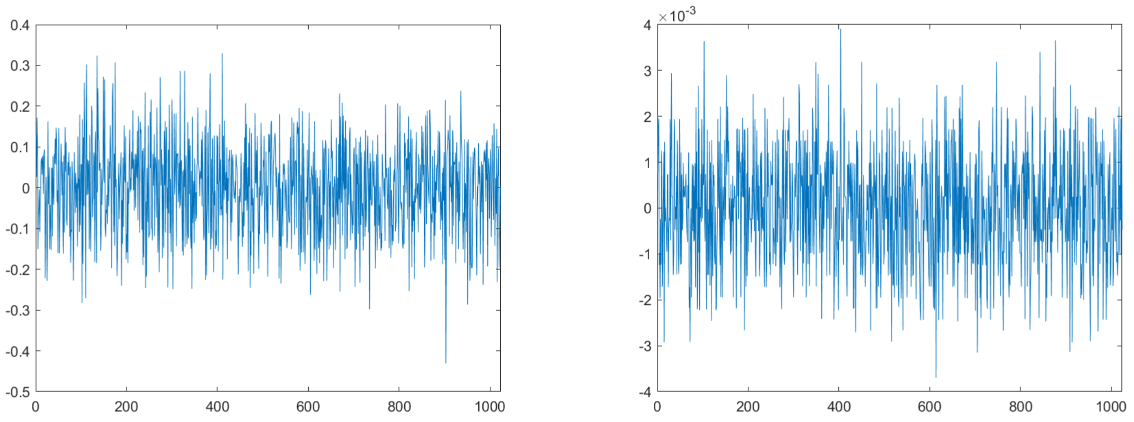


Figure 1: Original (sparse) signal

The signal was then convolved with the Gaussian random matrix and the subsampled Walsh-Hadamard transform and noise added to the result:



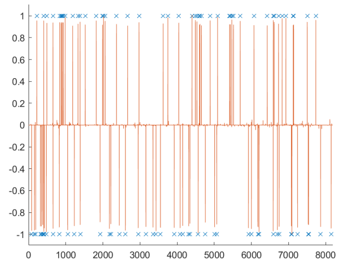
(a) Gaussian measurement signal

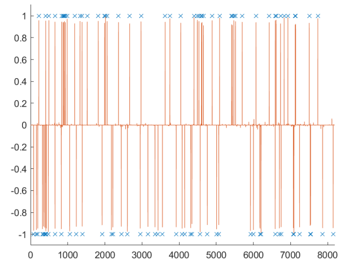
(b) Welsh-Hadamard signal

The ISTA, FISTA, and ADMM methods were then applied to the noisy signals in order to recover x_0. They all attempt to solve compressed sensing recovery problem:

$$x_{\text{CS}} = \underset{x}{\operatorname{argmin}} \frac{1}{2}\|Ax - y\|_2^2 + \lambda\|x\|_1$$

The quality of the reconstruction depends heavily on the parameter λ - which controls the tradeoff between the data fit term $\|Ax-y\|_2^2$ and the sparsity inducing term $\|x\|_1$. The only way to assess the quality of the reconstruction (without knowing the true data-generating signal x_0) is by visual inspection. A reliable initial guess for the value of λ is the take $0.01 \times \max |A^\top y|$ - i.e. 0.01 percent of the maximum entry of the absolute value of the backprojection. From there we can search for a suitable λ till the reconstruction is optimal. Below are plots of the reconstructions (orange) over the original signal (blue crosses).


(a) ISTA recovered signa


(b) FISTA recovered signa

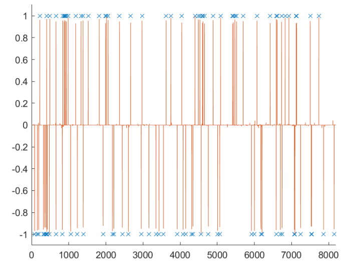
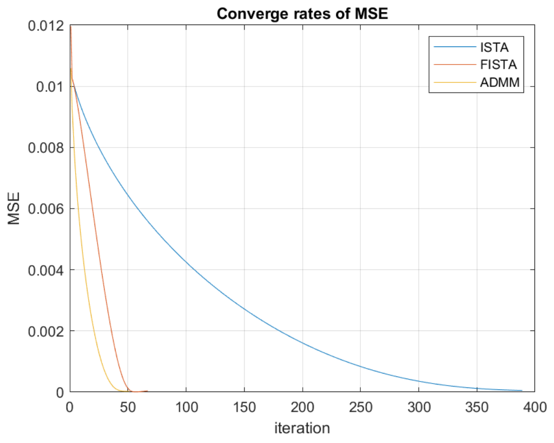
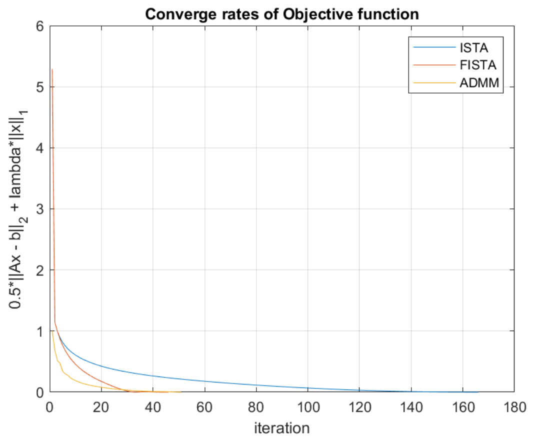

(c) ADMM recovered signal

Figure 3: Gaussian matrix recovered signals, $\lambda = 0.02 \times \max |A^\top y| = 0.0044$

We found that a value of $\lambda = 0.0044$ seemed to provide a good reconstruction for all three methods. The convergence rates of the MSE values and Objective Function values are plotted below:



(a) ISTA/FISTA/ADMM MSE



(b) ISTA/FISTA Objective function values $F(x_k) - F(x^*)$

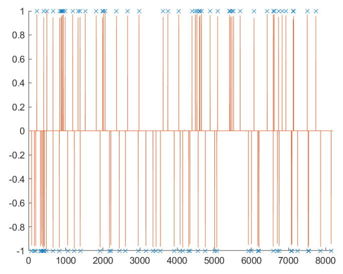
Figure 4: Gaussian matrix convergence

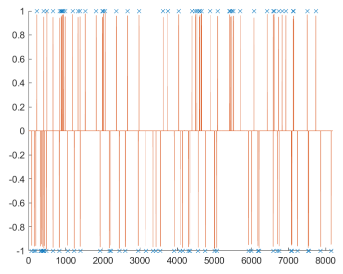
Firstly, we observe that the ISTA method takes many more iterations ($\approx$ 400) to converge, compared to FISTA ($\approx$ 50). From the theory we know that ISTA has convergence rate in the function values $F(x_k) - F(x) \approx O(1/k)$ and FISTA has convergence rate $F(x_k) - F(x) \approx O(1/k^2)$ [1]. This is verified by the Objective Function convergence rate figure, where we observe that ISTA roughly has shape $1/k$ and FISTA $1/k^2$. Experimentally we also observe that ADMM seems

[1] A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems, Amir Beck and Marc Teboulle

to converge in fewer iterations. The convergence theory relating to ADMM simply states that convergence is guarenteed under the assumption of proper, closed and convexity of the objective functions.

We repeat the same process for the subsampled Walsh-Hadamard signal. Firstly, we attempt to find the optimal λ by plotting the reconstructions:


(a) ISTA, $\lambda = 0.0659$


(b) FISTA, $\lambda = 0.0659$

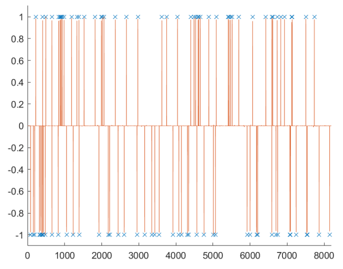
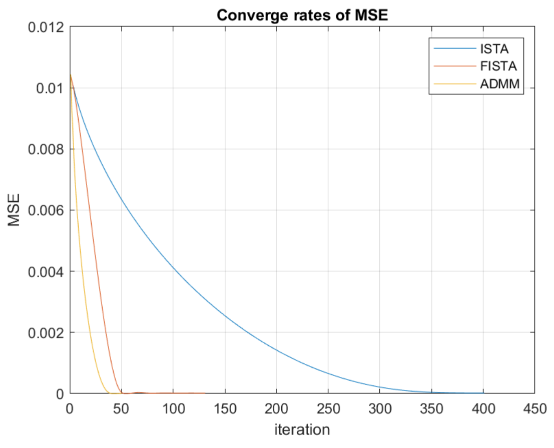
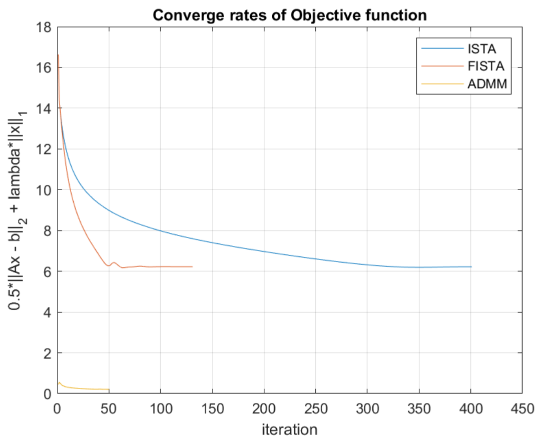

(c) ADMM, $\lambda = 0.0022$

Figure 5: Welsh-Hadamard Recovered signals

In this case, however, we found that ADMM required a much smaller value $\lambda = 0.0022$ compared to FISTA and ISTA $\lambda = 0.0659$ to produce an accurate reconstruction. Secondly, we plot the MSE and Objective function converge plots:



(a) ISTA/FISTA/ADMM MSE



(b) ISTA/FISTA/ADMM Objective function

Figure 6: Welsh-Hadamard convergence

Again, we note in both the MSE plot and Objective Function plot ISTA converges slower than FISTA and seems to demonstrate $O(1/k)$ rate of convergence compared to $O(1/k^2)$ of FISTA. Note the objective function values of ADMM are much lower because the ADMM is minimising a different objective (with a smaller λ value).

3 EXERCISE 5

The trajectories of the iterates from the infeasible starting point $\boldsymbol{x}_0 = (0, 2)$ are shown in Figure 7. The minimum found took vaalue $\boldsymbol{x}^* = (0.8571, 1.1249)$. It appears that Augmented Lagrangian takes a path that more stongly adheres to the equality constraint (shown by the elliptical solid red line). The two algorithms took slightly different number of iterations with Quadratic Penalty (QP) taking 40 and Augmented Lagrangian (AL) taking 51.

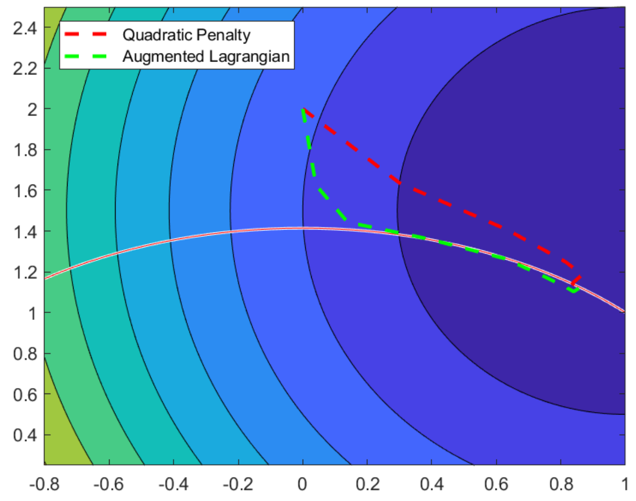


Figure 7: Iterate trajectories

The trajectories of the iterates from the feasible starting point $\boldsymbol{x}_0 = (-1, -1)$ are shown in Figure 8. Again the AL method takes a path that more strongly satisfies the equality constraints. Additionally the QP method took 52 iterations, whereas the AL method took 63 iterations. The convergence rate of the QP method from a feasible starting point therefore seems to be faster than the AL method.

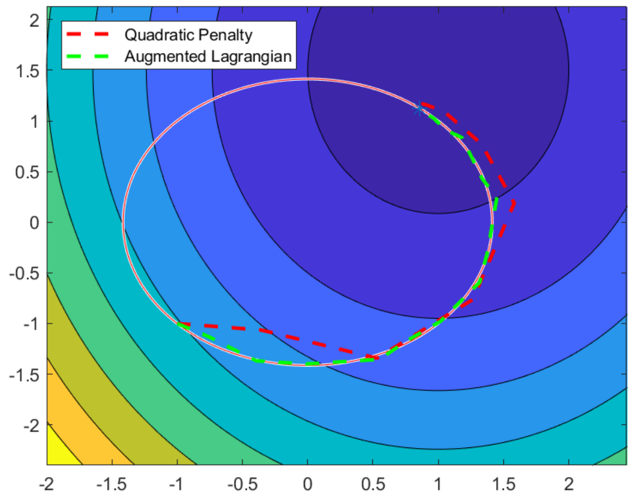
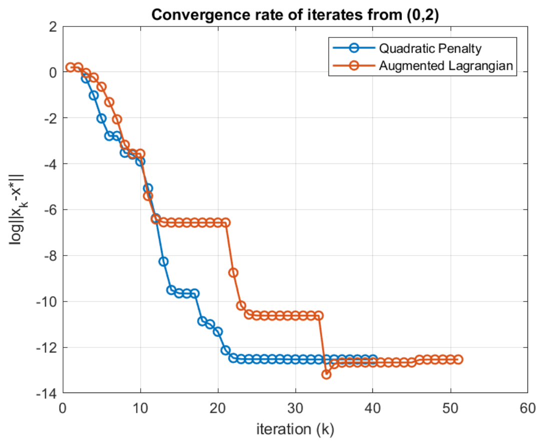
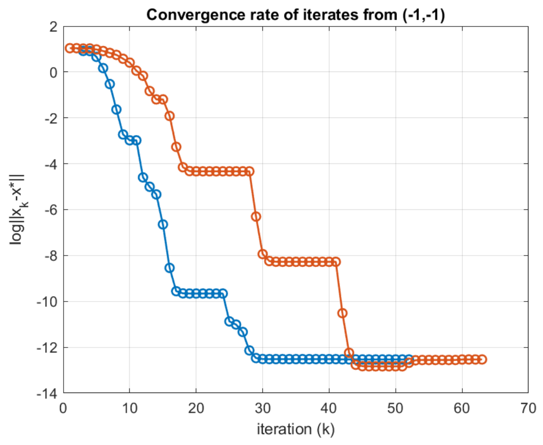


Figure 8: Trajectories from $\boldsymbol{x}_0 = (1, 1)$ (feasible)

Convergence plots of both methods from both starting points are given in Figure 9. The iterate values on the log scale: $\log \|x_k - x^*\|$ are plotted. From the infeasible starting point Figure 9a we see that the convergence rates of both algorithms are similar with QL taking 40 iterations to converge and AL 51. From the feasible starting point Figure 9b the Convergence rates are also similar, however the Augmented Lagrangian takes 63 iterations to converge, whereas the Quadratic Penalty only taking 52. From the graphs we can observe that from both starting points, the convergence rates of both algorithms are roughly **sublinear**. We would expect the AL method to converge slower than the QP method since it introduces an addtional equality constraint penalty term into the Lagrangian. This additionally explains why it stays closer to the equality constraint in the trajectory - the objective function is penalised more heavily for deviating from it.



(a) Rate from $\boldsymbol{x}_0 = (0, 2)$ (infeasible)



(b) Rate from $\boldsymbol{x}_0 = (-1, -1)$ (feasible)

Figure 9: Convergence rates

Quadratic Penalty - Code

```
% Define Q function and derivatives
Q.q = @(x,mu)  (x(1)-1)^2+0.5*(x(2)-1.5)^2-1+mu/2*(x(1)^2 + x(2)^2 -2)^2;
Q.dq = @(x,mu)  [2*(x(1)-1)  + 2*mu*x(1)*(x(1)^2 + x(2)^2 -2);  (x(2)-1.5) + ...
    2*mu*x(2)*(x(1)^2+x(2)^2-2)];
Q.d2q = @(x,mu)  [2 + 2*mu*(x(1)^2 + x(2)^2 - 2) + 4*mu*x(1)^2,  4*mu*x(1)*x(2);
                 4*mu*x(1)*x(2), 1 + 2*mu*(x(1)^2 + x(2)^2 - 2) + 4*mu*x(2)^2];

% Quadratic Penalty function
function [x_k, info] = quadraticPenalty(Q,tau0,mu0,alpha0,x0,tol,ls,maxIter)

% initialise params
x_k = x0; tau_k = tau0; mu_k = mu0; nIter = 1;
x_k_1 = [0;0]

info.xs = x0;
while nIter < maxIter && norm(x_k - x_k_1) > tol

    x_k_1 = x_k;

    % QP term with fixed mu during iteration
    F.f = @(x) Q.q(x,mu0);
    F.df = @(x) Q.dq(x,mu0);
    F.d2f = @(x) Q.d2q(x,mu0);

    % Newton step using backtracking line search
    [x_k, ¬, ¬, ¬]= descentLineSearch(F, ls, alpha0, x_k, tau_k, 10);

    % Decrease tau
    tau_k = tau0^-nIter;
    % Increase mu
    mu_k = 2*mu_k;

    info.xs = [info.xs x_k];
    nIter = nIter + 1;
end
info.it = nIter;
end
```

Augmented Lagrangian - Code

```
% Define Augmented Lagrangian and derivatives
L.h = @(x) x(1)^2 + x(2)^2 - 2;
L.l = @(x,mu,nu) Q.q(x, mu) + nu*L.h(x);
L.dl = @(x,mu,nu) Q.dq(x, mu) + nu.*[2*x(1); 2*x(2)];
L.d2l = @(x,mu,nu) Q.d2q(x, mu) + nu.*[2, 0; 0, 2];

function [x_k, info] = augLagrangian(L,tau0,mu,nu0,alpha0,x0,tol,ls,maxIter)
x_k = x0; tau_k = tau0; nu_k = nu0; nIter = 1;
x_k_1 = [0;0];

info.xs = x0;

while nIter < maxIter && norm(x_k - x_k_1) > tol

    x_k_1 = x_k;

    % Define Lagrangian with fixed mu and nu for iteration
    F.f = @(x) L.l(x,mu,nu_k);
    F.df = @(x) L.dl(x,mu,nu_k);
    F.d2f = @(x) L.d2l(x,mu,nu_k);

    % Newton step using backtracking line search
    x_k = descentLineSearch(F, ls, alpha0, x_k_1, tau_k, maxIter);

    % Decrease tau
    tau_k = tau0^-nIter;
    % Update nu according to formula in Nocedal and Wright
    nu_k = nu_k + (mu)*F.h(x_k);

    nIter = nIter + 1;
    info.xs = [info.xs x_k];
end
info.iter = nIter;
end
```