4M17 Practical Optimisation

Coursework Assignment 1 – Norm Approximation

Candidate Number: 5628A

Question 1

We have the following norm approximation problem:

$$\text{minimise} \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\| \tag{1}$$

Part 1a

l_1-norm

For $\mathbf{u} \in \mathbb{R}^m$, the l_1-norm is defined as:

$$\|\mathbf{u}\|_1 = \sum_{i=1}^{m} |u_i| \tag{2}$$

The corresponding norm approximation problem is

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 = \min_{\mathbf{x}} \sum_{i=1}^{m} \left|\mathbf{a}_i^T \mathbf{x} - b_i\right| \tag{3}$$

l_∞-norm

The l_∞-norm is defined as:

$$\|\mathbf{u}\|_\infty = \max_{i=1,\dots,m} |u_i| \tag{4}$$

The corresponding norm approximation problem is

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_\infty = \min_{\mathbf{x}} \max_{i=1,\dots,m} |\mathbf{a}_i^T \mathbf{x} - b_i| \tag{5}$$

l_2-norm

The l_2-norm is defined as:

$$\|\mathbf{u}\|_2 = \left(\sum_{i=1}^{m} u_i^2\right)^{\frac{1}{2}} \tag{6}$$

The corresponding norm approximation problem is

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 = \min_{\mathbf{x}} \left(\sum_{i=1}^{m} (\mathbf{a}_i^T \mathbf{x} - b_i)^2\right)^{\frac{1}{2}} \tag{7}$$

Since all the terms inside the summation are non-negative, this is equivalent to minimising the square of the l_2-norm:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \tag{8}$$

Therefore, we have

$$\begin{aligned}\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 &= \min_{\mathbf{x}} \sum_{i=1}^{m} (\mathbf{a}_i^T\mathbf{x}-b_i)^2 \\ &= \min_{\mathbf{x}} (\mathbf{A}\mathbf{x}-\mathbf{b})^T(\mathbf{A}\mathbf{x}-\mathbf{b}) \\ &= \min_{\mathbf{x}} (\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{b})\end{aligned}$$

To show that the quadratic function we are minimising is convex, we simply need to show that **dom** f is convex and the Hessian, $\nabla^2 f(\mathbf{x}) = \mathbf{A}^T\mathbf{A}$, is positive semi-definite (Boyd and Vandenberghe, 2004), which are both true. Thus, we end up with an optimisation problem with a convex quadratic function.

We can compute an analytical solution by differentiating f wrt $\mathbf{x}$ and equating it to zero:

$$\begin{aligned}\nabla f(\mathbf{x}) &= 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{b} = \mathbf{0} \\ \therefore \mathbf{A}^T\mathbf{A}\mathbf{x} &= \mathbf{A}^T\mathbf{b} \qquad (9)\end{aligned}$$

$\mathbf{x}$ can then be found by solving a *linear system of equations.*

Part 1b

We can cast the l_1 and l_∞ norm approximation problems as *linear programming* (LP) problems.

l_1-norm

From (3), the norm approximation problem is

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_1 = \min_{\mathbf{x}} \sum_{i=1}^{m} \left|\mathbf{a}_i^T\mathbf{x}-b_i\right|$$

We can replace the absolute of a quantity by:

$$\begin{aligned}|z_i| &= \max\ \{-z_i, z_i\} \\ &= \min\ \{u_i \mid z_i \leqslant u_i, -z_i \leqslant u_i\}\end{aligned}$$

Thus our problem in (3) can be written as a LP problem:

$$\begin{aligned}\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{i=1}^{m} u_i \qquad (10)\\ \text{s.t.} \quad & \mathbf{a}_i^T\mathbf{x}-b_i \leqslant u_i, \quad i=1,...,m \\ & -(\mathbf{a}_i^T\mathbf{x}-b_i) \leqslant u_i, \quad i=1,...,m\end{aligned}$$

which could also be written as:

$$\begin{aligned}\min_{\tilde{\mathbf{x}}} \quad & \tilde{\mathbf{c}}^T\tilde{\mathbf{x}} \qquad (11)\\ \text{s.t.} \quad & \tilde{\mathbf{A}}\tilde{\mathbf{x}} \leqslant \tilde{\mathbf{b}}\end{aligned}$$

where

$$\tilde{\mathbf{x}} = \begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix}, \quad \tilde{\mathbf{c}} = \begin{bmatrix}\mathbf{0}\\ \mathbf{1}\end{bmatrix}, \quad \tilde{\mathbf{A}} = \begin{bmatrix}\mathbf{A} & -\mathbf{I}\\ -\mathbf{A} & -\mathbf{I}\end{bmatrix}, \quad \tilde{\mathbf{b}} = \begin{bmatrix}\mathbf{b}\\ -\mathbf{b}\end{bmatrix}$$

The dimensions are as follows:

$$\tilde{\mathbf{x}} \in \mathbb{R}^{n+m}, \quad \tilde{\mathbf{c}} \in \mathbb{R}^{n+m}, \quad \tilde{\mathbf{A}} \in \mathbb{R}^{2m\times(m+n)}, \quad \tilde{\mathbf{b}} \in \mathbb{R}^{2m}$$

Note that the inequality constraints automatically imply $\mathbf{u} \geqslant \mathbf{0}$.

l_∞-norm

From (5), the norm approximation problem is:

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_\infty = \min_{\mathbf{x}} \max_{i=1,\dots,m} |\mathbf{a}_i^T\mathbf{x} - b_i| \tag{12}$$

The maximum of a set of absolute values can be written as:

$$\begin{aligned}\max_i \ |z_i| &= \max_i \ \{z_i, -z_i\} \\ &= \min_i \ \{u \mid z_i \leqslant u, -z_i \leqslant u\}\end{aligned}$$

Therefore, the l_∞ approximation problem can be written as a LP problem:

$$\begin{aligned}\min_{\mathbf{x},u} \quad & t \\ \text{s.t.} \quad & \mathbf{a}_i^T\mathbf{x} - b_i \leqslant u, \quad i = 1, \dots, m \\ & -(\mathbf{a}_i^T\mathbf{x} - b_i) \leqslant u, \quad i = 1, \dots, m\end{aligned}$$

Alternatively, we can write:

$$\begin{aligned}\min_{\tilde{\mathbf{x}}} \quad & \tilde{\mathbf{c}}^T\tilde{\mathbf{x}} \\ \text{s.t.} \quad & \tilde{\mathbf{A}}\tilde{\mathbf{x}} \leqslant \tilde{\mathbf{b}}\end{aligned}$$

where

$$\tilde{\mathbf{x}} = \begin{bmatrix}\mathbf{x} \\ u\end{bmatrix}, \quad \tilde{\mathbf{c}} = \begin{bmatrix}\mathbf{0} \\ 1\end{bmatrix}, \quad \tilde{\mathbf{A}} = \begin{bmatrix}\mathbf{A} & -\mathbf{1} \\ -\mathbf{A} & -\mathbf{1}\end{bmatrix}, \quad \tilde{\mathbf{b}} = \begin{bmatrix}\mathbf{b} \\ -\mathbf{b}\end{bmatrix}$$

The dimensions are as follows:

$$\tilde{\mathbf{x}} \in \mathbb{R}^{n+1}, \quad \tilde{\mathbf{c}} \in \mathbb{R}^{n+1}, \quad \tilde{\mathbf{A}} \in \mathbb{R}^{2m\times(n+1)}, \quad \tilde{\mathbf{b}} \in \mathbb{R}^{2m}$$

Part 1c

For the 5 pairs of problem data, the dual-simplex algorithm is applied to solve the l_1- and l_∞-norm minimisation problems, and a simple linear solver is used to solve the l_2-norm minimisation problem. The optimised l-norms and the corresponding running times are recorded in Table 1. The running time generally increases with n for the same optimisation problem. l_2-norm approximation is much faster than l_1 and l_∞ since it can be solved directly without iterative methods. l_1-norm approximation is slower than l_∞ because it has a higher dimension ($\mathbb{R}^{n+m}$ vs $\mathbb{R}^{n+1}$ for $\tilde{\mathbf{x}}$).

		optimised norm			**runtime (s)**		
dataset	n	l_1	l_2	l_∞	l_1	l_2	l_∞
`(A1,b1)`	16	10.26	2.389	0.6081	0.1297	1.00×10^4	0.0480
`(A2,b2)`	64	33.61	4.404	0.5796	0.0648	2.00×10^4	0.0517
`(A3,b3)`	256	143.26	9.390	0.6135	1.6055	0.0015	1.0052
`(A4,b4)`	512	277.18	12.906	0.5936	16.8054	0.0088	10.2556
`(A5,b5)`	1024	571.64	18.553	0.6035	275.9477	0.0367	151.4514

Table 1: Optimised l-norms and the corresponding running times for the 5 datasets

Part 1d

Histograms of residuals of the norm approximation problems for data pair `(A5,b5)` for the l_1, l_2 and l_∞-norms are shown in Figure 1.

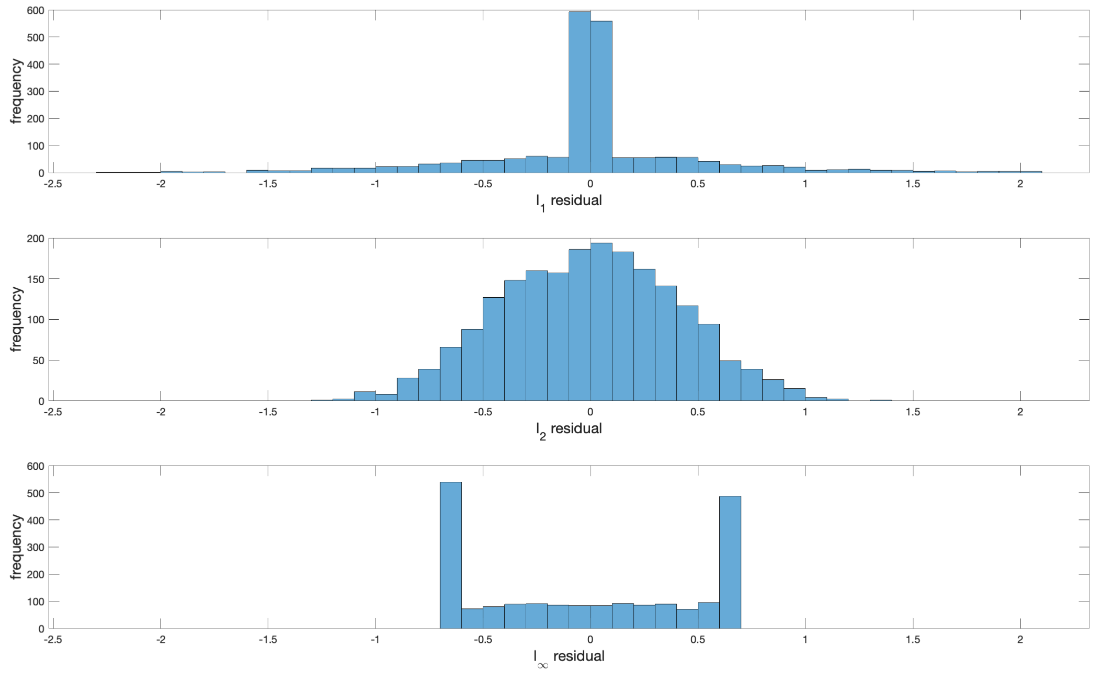

Figure 1: Histograms of residuals of the norm approximation problems for data pair `(A5,b5)` for the l_1, l_2 and l_∞-norms

Informally, in a norm-approximation problem, the choice of norm determines what values of residuals are emphasised which is then reflected in the amplitude distribution (histogram) of the optimal residuals. Having a small relative weighting on a value u means that we care little about residuals of values u, and vice versa.

l_1-norm

For l_1-norm, most weight ($> 50\%$) is concentrated around small residuals ($|u| < 0.1$) meaning the residuals are mostly very small or even zero. However, it has the biggest residual range amongst the three norms and it also has more large residuals that l_2-norm, although the relative weightings of the large residuals are very small.

l_1-norm puts relatively large weight on small residuals, and small weight on large residuals, meaning that we would expect many of the optimal residuals to have values close to or equal to zero. In other words, many of the equations $\mathbf{a}_i^T \mathbf{x} = b_i$ are exactly or close to satisfied. Moreover, it is less sensitive to outliers (large residuals) since there is little incentive to drive large residuals smaller.

l_2-norm

For l_2-norm, the weight is more evenly distributed – the decrease in relative weighting away from the peak is much milder compared to l_1-norm. The residual range is smaller and it has relatively few large residuals compared to l_1-norm.

l_2-norm puts very small weight on small residuals – u^2 is very small when u is small. We would expect the optimal residuals to be small, but not as small as those of l_1-norm since l_2-norm has less incentive to drive small residuals smaller. Conversely, l_2-norm is much more sensitive to outliers since their effect is amplified by the squaring. We therefore see relatively fewer large residuals in the optimal solution to l_2-norm approximation.

l_∞-norm

For l_∞-norm, most weight is concentrated on the residuals with the largest amplitudes. The relative weightings of the other components are much smaller.

l_∞-norm only cares about the maximum absolute value of the residual and completely ignores the rest. To gain more insights, the actual values of the residuals are plotted in Figure 2. We see that many residuals are in fact *at* the maximum amplitude (± 0.6135). This is because when we minimise the maximum residual, we search for $\mathbf{x}$ that would result in the smallest l_∞-norm ball which could contain all the residuals. Note that there could be multiple residuals touching the face of the norm ball. Starting with only one point on the face of the norm ball, we search for $\mathbf{x}$ such that the size of the norm ball is reduced, until another point inside the ball meets the face of the ball. This is repeated until the norm ball cannot be shrunk any further. Thus, at the optimal solution, there will be many points touching the face of the norm ball with constant distance to the centre (the radius) which is equal to the optimised l_∞-norm. We therefore have the bounded residual histogram and the components in-between are the residuals inside the norm ball.

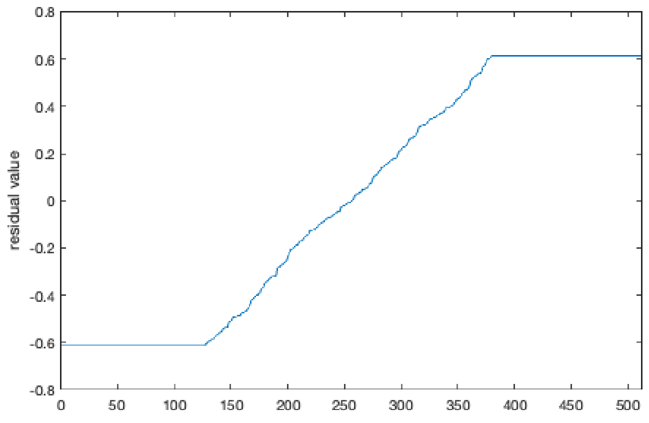

Figure 2: Plot of sorted optimised residuals for l_∞-norm approximation

Question 2

Part 2a

The LP formulation of the l_1-norm approximation problem is derived in Part 1b. We can convert the problem to an unconstrained minimisation problem by adding a *logarithmic barrier function*:

$$\min_{\mathbf{x},\mathbf{u}} \quad f(\mathbf{x},\mathbf{u}) = t\sum_{i=1}^{m} u_i - \sum_{i=1}^{m} \log\left(-\mathbf{a}_i^T\mathbf{x} + b_i + u_i\right) - \sum_{i=1}^{m} \log\left(\mathbf{a}_i^T\mathbf{x} - b_i + u_i\right) \tag{13}$$

For fixed $t \geqslant 0$, the expression for the gradient of the cost is then found by differentiation:

$$\nabla f(\mathbf{x},\mathbf{u}) = \begin{bmatrix} -\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2 \\ t\mathbf{1} - \boldsymbol{\gamma}_1 - \boldsymbol{\gamma}_2 \end{bmatrix} \tag{14}$$

where

$$\boldsymbol{\theta}_1 = \sum_{i=1}^{m} \frac{-1}{-\mathbf{a}_i^T\mathbf{x} + b_i + u_i}\mathbf{a}_i \quad , \quad \boldsymbol{\theta}_2 = \sum_{i=1}^{m} \frac{1}{\mathbf{a}_i^T\mathbf{x} - b_i + u_i}\mathbf{a}_i$$

$\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$ are vectors where

$$(\boldsymbol{\gamma}_1)_i = \frac{1}{-\mathbf{a}_i^T\mathbf{x} + b_i + u_i} \quad , \quad (\boldsymbol{\gamma}_2)_i = \frac{1}{\mathbf{a}_i^T\mathbf{x} - b_i + u_i}$$

Part 2b

To solve the problem data `(A3,b3)` in Part 2a, we apply the first-order gradient method:

$$\text{decent direction } \Delta\tilde{\mathbf{x}} = -\nabla f(\mathbf{x},\mathbf{u}) \tag{15}$$

with backtracking linesearch, for $t = 1$. The stopping criterion is $\|f(\tilde{\mathbf{x}})\|_2 < \epsilon$ for some $\epsilon > 0$. The parameters used for backtracking linesearch are $\alpha = 0.01$ and $\beta = 0.8$.

Using this algorithm, the l_1-norm corresponding to the optimised $\mathbf{x}$ is 169.34 which is different from the one obtained in Part 1c using dual-simplex (143.26). This is because the problem is a LP problem where the optimal solution is at a vertex where some constraints are active. During the search using Newton with logarithmic barrier function, as we approach a constraint, the function value increases and will reach infinity if it is at the boundary. Therefore, we can never reach the true optimal point of the original objective function due to the penalty by the barrier function. We can obtain a closer approximation by increasing t but this would also increase the runtime.

The runtime for this search is 91.31 s which is much longer than dual-simplex (1.61 s). Thus, we conclude that the dual-simplex is a much better algorithm in terms of accuracy and runtime for this problem.

Part 2c

Figure 3 shows the semilogarithmic plot of the minimisation error, $(f(\tilde{\mathbf{x}}^{(k)}) - p^*)$, where k is the iteration number and p^* is the minimum of $f(\tilde{\mathbf{x}})$. In our case, we approximate p^* as the function value at the last iteration, i.e., $p^* = f(\tilde{\mathbf{x}}^{(N)})$ where N is the total number of iterations.

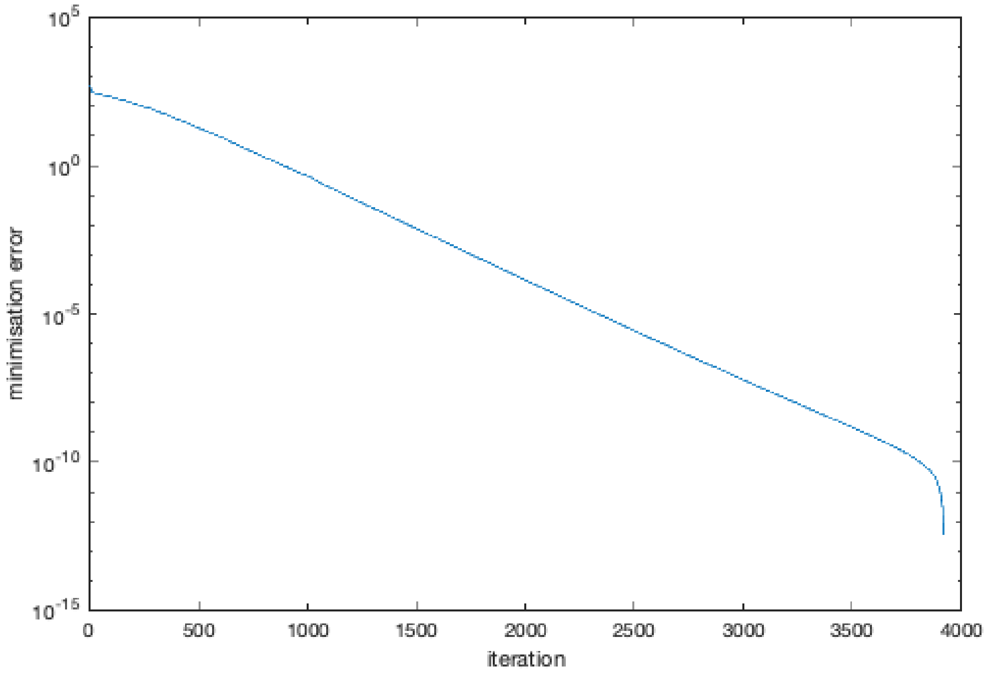

Figure 3: Semilogarithmic plot of the minimisation error against the number of iteration using the first-order gradient method with backtracking (Part 2b). The red dotted line ($x = 3700$) represents roughly the point where the algorithm transitions from the damped Newton phase to the quadratically convergent phase.

$f(\mathbf{x}, \mathbf{u})$ is self-concordant because $\tilde{\mathbf{c}}^T \tilde{\mathbf{x}}$ is linear and the logarithmic barrier function is self-concordant. Hence, gradient descent with backtracking linesearch could at best converge to the optimal solution linearly. From Figure 3, the overall convergence rate is indeed approximately linear, with a slightly slower convergence at the beginning for about 400 iterations, and extremely rapid convergence at the end when it approaches the optimal solution. Excluding the rapid drop at the end, the error is reduced by a factor of around 10^{11} within 3500 iterations, giving an average error reduction of a factor of $10^{-11/3500} = 0.993$ per iteration, which is rather slow.

Question 3

We have the following l_1-regularised least squares problem:

$$\min_{\mathbf{x}} \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_1 = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \sum_{i=1}^{n} |x_i| \tag{16}$$

where $\lambda > 0$ is a regularisation parameters.

Part 3a

Similar to what we did in Part 1b, we can write the absolute of a quantity as:

$$|x_i| = \max\{-x_i, x_i\} = \min(u_i | -x_i \leqslant u_i, x_i \leqslant u_i) \ , \ i = 1, ..., n$$

Therefore, (16) can be written as a constrained optimisation problem:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \lambda \sum_{i=1}^{n} u_i \\ \text{s.t.} \quad & -x_i \leqslant u_i \ , \ i = 1, ..., n \\ & x_i \leqslant u_i \ , \ i = 1, ..., n \end{aligned} \tag{17}$$

Apply the following logarithmic barrier function which corresponds to the inequality constraints:

$$\Phi(\mathbf{x},\mathbf{u}) = -\sum_{i=1}^{n} \log(u_i + x_i) - \sum_{i=1}^{n} \log(u_i - x_i) \tag{18}$$

Finally, we have our *central path formulation*:

$$\begin{aligned} \phi_t(\mathbf{x},\mathbf{u}) &= t\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + t\lambda \sum_{i=1}^{n} u_i - \sum_{i=1}^{n} \log(u_i + x_i) - \sum_{i=1}^{n} \log(u_i - x_i) & (19) \\ &= t\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + t\lambda \sum_{i=1}^{n} u_i - \sum_{i=1}^{n} \log(u_i^2 - x_i^2) & (20) \end{aligned}$$

Part 3b

The expression for the gradient is obtained as follows:

$$\nabla \phi_t(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \nabla_{\mathbf{x}} \phi_t \\ \nabla_{\mathbf{u}} \phi_t \end{bmatrix} \tag{21}$$

where $\nabla_{\mathbf{x}}\phi_t$ and $\nabla_{\mathbf{u}}\phi_t$ are the derivatives of ϕ_t w.r.t. $\mathbf{x}$ and $\mathbf{u}$ respectively. Defining $\boldsymbol{\theta}, \boldsymbol{\gamma} \in \mathbb{R}^n$ as vectors where

$$\begin{aligned} \theta_i &= \frac{\partial}{\partial x_i}\left(-\sum_{i=1}^{n} \log(u_i^2 - x_i^2)\right) = \frac{2x_i}{u_i^2 - x_i^2} \\ \gamma_i &= \frac{\partial}{\partial u_i}\left(-\sum_{i=1}^{n} \log(u_i^2 - x_i^2)\right) = -\frac{2u_i}{u_i^2 - x_i^2} \end{aligned}$$

we have

$$\begin{aligned} \nabla_{\mathbf{x}}\phi_t &= t(2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{b}) + \boldsymbol{\theta} & (22) \\ \nabla_{\mathbf{x}}\phi_t &= t\lambda\mathbf{1} + \boldsymbol{\gamma} & (23) \end{aligned}$$

$$\therefore \nabla \phi_t(\mathbf{x},\mathbf{u}) = t\begin{bmatrix} 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{b} \\ \lambda\mathbf{1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\theta} \\ \boldsymbol{\gamma} \end{bmatrix} \tag{24}$$

The Hessian $\nabla^2\phi_t(\mathbf{x},\mathbf{u})$ is obtained by calculating the individual Hessians term-by-term from (20) and summing the results:

$$\nabla^2 \left(t\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2\right) = \begin{bmatrix} 2\mathbf{A}^T\mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{25}$$

$$\nabla^2 \left(t\lambda\sum_{i=1}^{n} u_i\right) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{26}$$

$$\nabla^2 \left(-\sum_{i=1}^{n} \log\left(u_i^2 - x_i^2\right)\right) = \begin{bmatrix} \boldsymbol{\Theta} & \boldsymbol{\Gamma} \\ \boldsymbol{\Gamma} & \boldsymbol{\Theta} \end{bmatrix} \tag{27}$$

$$\therefore \quad \nabla^2\phi_t(\mathbf{x},\mathbf{u}) = \begin{bmatrix} 2t\mathbf{A}^T\mathbf{A} + \boldsymbol{\Theta} & \boldsymbol{\Gamma} \\ \boldsymbol{\Gamma} & \boldsymbol{\Theta} \end{bmatrix} \tag{28}$$

$$\text{where} \quad \boldsymbol{\Theta} = \operatorname{diag}\left(\left\{\frac{2(u_i^2 + x_i^2)}{(u_i^2 - x_i^2)^2}\right\}_{i=1}^{n}\right) \quad \text{and} \quad \boldsymbol{\Gamma} = \operatorname{diag}\left(\left\{\frac{-4(u_i x_i)}{(u_i^2 - x_i^2)^2}\right\}_{i=1}^{n}\right)$$

Part 3c

Sparse signal reconstruction on the data `A` and `b=A*x0` with central path formulation in (20) is done using an exact Newton interior-point method. The result is shown in Figure 5.

Part 3d

For minimum energy reconstruction, the recovered signal $\mathbf{x}^*$ is the solution to the following optimisation problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|\mathbf{x}\|_2^2 = \mathbf{x}^T\mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned} \tag{29}$$

This can be solved analytically using Lagrange multipliers:

$$\text{Lagrangian} \quad L(\mathbf{x},\boldsymbol{\lambda}) = \mathbf{x}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{30}$$

$$\nabla_{\mathbf{x}} L(\mathbf{x},\boldsymbol{\lambda}) = 2\mathbf{x} + \mathbf{A}^T\boldsymbol{\lambda} = 0 \tag{31}$$

$$\Rightarrow \quad \mathbf{x}^* = -\frac{1}{2}\mathbf{A}^T\boldsymbol{\lambda} \tag{32}$$

Substituting (32) into constraints $\mathbf{A}\mathbf{x} = \mathbf{b}$, we have

$$-\frac{1}{2}\mathbf{A}^T\mathbf{A}\boldsymbol{\lambda} = \mathbf{b} \quad \Rightarrow \quad \boldsymbol{\lambda} = -2(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \tag{33}$$

Finally, substituting (33) back into (32), the optimal $\mathbf{x}^*$ is found:

$$\mathbf{x}^* = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \tag{34}$$

This solution is the minimum since $\nabla_{\mathbf{x}}^2 L = 2\mathbf{I}$ which is positive definite. The reconstructed signal using this method is displayed in Figure 6.

Results comparison

Sparse signal reconstruction (Figure 5) gives a close-to-perfect signal, with slight deviation at the spikes and very small fluctuation in the flat regions. All the spikes are located perfectly. Minimum energy reconstruction (Figure 6) returns a much worse signal which does not resemble the original signal. It identifies all the spikes but the amplitudes of the spikes (~ 0.2) are much smaller than the original and the other parts of the signal are of similar magnitudes. The minimum energy reconstruction is not able to capture the low-cardinality pattern of the signal.

From Part 1d, we see that l_1-norm approximation puts relatively large weight on small residuals, meaning that the optimal residuals will tend to have many entries that are close to or equal to zero. Therefore, the l_1-regularised least squares problem will tend to produce sparse solutions which is suitable for this signal reconstruction problem. When the l_1-regularisor is applied, we essentially put in prior information/assumption that the actual signal has low cardinality.

Conversely, l_2-norm puts very small weight on small residuals as explained in Part 1d. In minimium energy reconstruction where $\|\mathbf{x}\|_2^2$ is minimised, the contribution to the l_2-norm from points that are close to zero is very small. Thus, the norm has little incentive to drive the small values even smaller, but instead puts emphasis on reducing large values including outliers. It is therefore hard for the reconstruction method to produce a signal with many zero/close to zero entries (i.e. sparse); we instead obtain a signal that has roughly the same variation amplitude throughout.

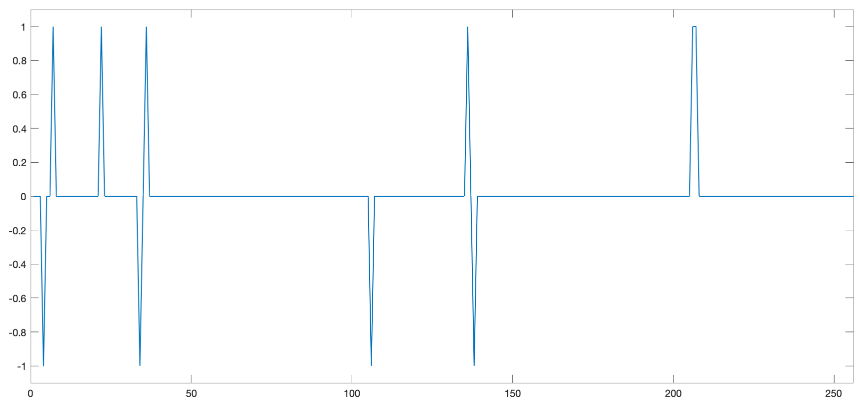

Figure 4: Original signal `x0` given in the coursework data

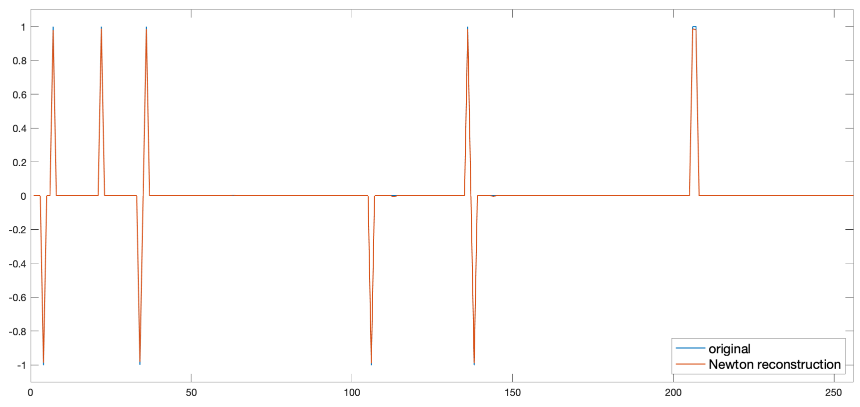

Figure 5: Recovered signal using *sparse signal reconstruction* in Part 3c, overlaid on the original signal

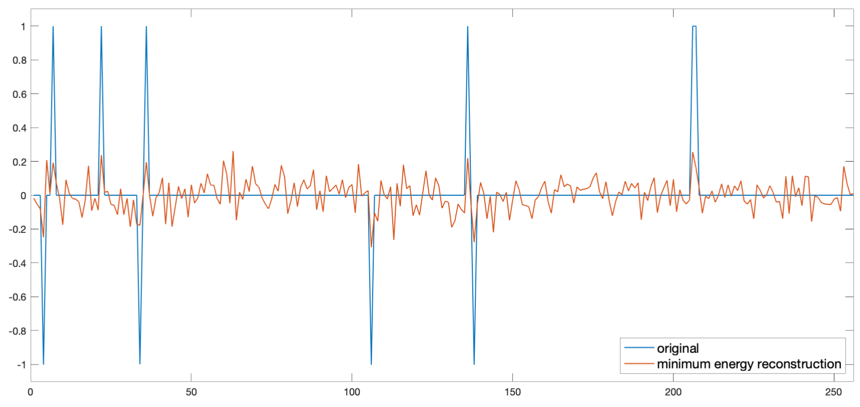

Figure 6: Recovered signal using *minimum energy reconstruction* in Part 3d, overlaid on the original signal

References

S. Boyd and L. Vandenberghe (2004), *Convex Optimization*. Cambridge University Press.

Question 1 MATLAB Code

```
clear all;

% arrays for recording the time needed to solve the problems
l1_times = zeros(5,1);
l2_times = zeros(5,1);
lin_times = zeros(5,1);
% arrays for storing the optimised l-norms
l1_opts = zeros(5,1);
l2_opts = zeros(5,1);
lin_opts = zeros(5,1);

for i=3:3
    A = eval(sprintf("load('q1_data/A%u.mat').A%u", i, i));
    b = eval(sprintf("load('q1_data/b%u.mat').b%u", i, i));

    [m, n] = size(A);

    % l1-norm
    I = eye(m);
    A_ = [A -I; -A -I];
    b_ = [b; -b];
    c_ = [zeros(n,1); ones(m,1)];

    tic;
    [x_opt_l1, l1_opt, exitflag_l1, output_l1] = linprog(c_, A_, b_);
    l1_times(i) = toc;
    l1_opts(i) = l1_opt;
    fprintf('dataset %u l1 completed', i);

    % l_infty-norm
    A_ = [A -ones(m,1); -A -ones(m,1)];
    b_ = [b; -b];
    c_ = [zeros(n,1); 1];

    tic;
    [x_opt_lin, lin_opt, exitflag_lin, output_lin] = linprog(c_, A_, b_);
    lin_times(i) = toc;
    lin_opts(i) = lin_opt;
    fprintf('dataset %u lin completed', i);

    % l2-norm
    tic;
    x_opt_l2 = linsolve(A'*A, A'*b);
    l2_times(i) = toc;
    % compute actual l2-norm
    l2_opt = sqrt(sum((A*x_opt_l2 - b).^2));
    l2_opts(i) = l2_opt;
    fprintf('dataset %u l2 completed', i);
end

% plot residuals
x_opt_l1 = x_opt_l1(1:n);   % only keep x, discard t
```

```
x_opt_lin = x_opt_lin(1:n);

figure();
ax1 = subplot(3,1,1);
histogram(A*x_opt_l1 - b);
ylabel('frequency', 'fontSize', 14); xlabel('l_1 residual', 'fontSize', 14);
ax2 = subplot(3,1,2);
histogram(A*x_opt_l2 - b);
ylabel('frequency', 'fontSize', 14); xlabel('l_2 residual', 'fontSize', 14);
ax3 = subplot(3,1,3);
histogram(A*x_opt_lin - b);
ylabel('frequency', 'fontSize', 14); xlabel('l_\infty residual', 'fontSize',
14);

linkaxes([ax1, ax2, ax3], 'x');
```

Question 2 MATLAB Code – Script

```
clear all;

% load data
A0 = load('q1_data/A3.mat').A3;
b0 = load('q1_data/b3.mat').b3;
[m, n] = size(A0);

c = [zeros(n,1); ones(m,1)];
A = [A0 -eye(m); -A0 -eye(m)];
b = [b0; -b0];

% parameters
tp = 1;
alpha = 0.1;
beta = 0.5;
eps = 1e-5;  % stopping criteria

% initialisation
s0 = [zeros(n,1); ones(m,1)];
s = s0;
grad = q2_grad(s,A,b,c,tp);

f_history = [q2_fval(s,A,b,c,tp)];

tic
while norm(grad) > eps
    tb = 1;
    grad = q2_grad(s,A,b,c,tp);
    ds = -grad;

    % backtracking linesearch
    while q2_fval(s+tb*ds,A,b,c,tp) > q2_fval(s,A,b,c,tp) +
alpha*tb*(grad'*ds)
        tb = tb * beta;
    end

    f = q2_fval(s,A,b,c,tp);
    f_history = [f_history f];

    s = s + tb*ds;
    disp(norm(grad));
end
toc

figure
semilogy(f_history - f_history(end))
xlabel('iteration')
ylabel('minimisation error')
```

Question 2 MATLAB Code – Functions

```matlab
function f = q2_fval(s,A,b,c,tp)
    f = tp*c'*s - sum(log(-A*s + b));
    if imag(f) ~= 0
        f = inf;
    end
end

function grad = q2_grad(s,A,b,c,tp)
    grad = tp*c - sum( A ./ (A*s - b) )';
end
```

Question 3 MATLAB Code – Script

```
clear all

% config
max_tp_order = 9;

% load data
A = load('q3_data/A.mat').A;
x0 = load('q3_data/x0.mat').x;
[m,n] = size(A);
b = A*x0;

% parameters
alpha = 0.01;
beta = 0.5;
eps = 1e-1;
tp = 1;

lambda = 0.01*max(abs(2*A'*b));

% initialise x
x = [zeros(n,1); ones(n,1)];

g = q3_grad(x,A,b,tp,lambda);
H = q3_hess(x,A,tp);
L = chol(H)';
dx = -inv(L')*inv(L)*g;
stopping_check = norm(inv(L)*g);

f_history = [];
stopping_check_history = [];

for j=1:max_tp_order
    while stopping_check / 2 > eps
        stopping_check_history = [stopping_check_history; stopping_check];

        % backtracking linesearch
        tb = 1;
        while q3_fval(x+tb*dx,A,b,tp,lambda) > q3_fval(x,A,b,tp,lambda) +
alpha*tb*(g'*dx)
            tb = tb * beta;
        end

        x = x + tb*dx;

        g = q3_grad(x,A,b,tp,lambda);
        H = q3_hess(x,A,tp);
        L = chol(H)';
        dx = -inv(L')*inv(L)*g;
        stopping_check = norm(inv(L)*g);

        fval = q3_fval(x,A,b,tp,lambda);
        f_history = [f_history; fval];
    end

    tp = tp * 10;
    tp
    stopping_check = inf;  % reset so that it doesn't meet the stopping
criteria
end

x_newton = x(1:n);
figure
hold on
plot(x0, 'linewidth', 1)
plot(x_newton, 'linewidth', 1)
xlim([0, length(x0)])
ylim([-1.1, 1.1])
legend('original', 'Newton reconstruction', 'Location', 'southeast',
'FontSize', 13)

% Q3d
x_min_energy = A'*inv(A*A')*b;
figure
hold on
plot(x0, 'linewidth', 1)
plot(x_min_energy, 'linewidth', 1)
xlim([0, length(x0)])
ylim([-1.1, 1.1])
legend('original', 'minimum energy reconstruction', 'Location', 'southeast',
'FontSize', 13)
```

Question 3 MATLAB Code – Functions

```
function f = q3_fval(x,A,b,t,lambda)
    [~,n] = size(A);
    u = x(n+1:2*n);
    x = x(1:n);

    f = t*sum((A*x-b).^2) + t*lambda*sum(u) - sum(log(u+x)) - sum(log(u-x));

    if (imag(f) ~= 0)
        f = inf;
    end
end

function grad = q3_grad(x,A,b,t,lambda)
    [~,n] = size(A);
    u = x(n+1:2*n);
    x = x(1:n);

    grad = [2*(A'*A)*x - 2*A'*b; zeros(n,1)] + [zeros(n,1);
lambda*ones(n,1)];
    denom = u.^2 - x.^2;
    grad = t*grad + 2*[x./denom; -u./denom];
end

function H = q3_hess(x,A,t)
    [~,n] = size(A);
    u = x(n+1:2*n);
    x = x(1:n);

    diag1 = diag( (u.^2 + x.^2) ./ (u.^2 - x.^2).^2 );
    diag2 = diag( (u .* x) ./ (u.^2 - x.^2).^2 );
    H = [2*t*(A'*A) + 2*diag1, -4*diag2; -4*diag2, 2*diag1];
end
```