Imperial College London

COURSEWORK 1

IMPERIAL COLLEGE LONDON

DEPARTMENT OF MATHEMATICS

MATH60005 Optimisation CID: 01862156

Date: November 23, 2022

Contents

1 Part I: Unconstrained Optimisation

1.1 Part (i)

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be given by

$$f(x_1, x_2) = (x_1+1)^2(x_2+1)^2, \quad (x_1, x_2) \in \mathbb{R}^2$$

which is clearly continuous in both variables as it is a polynomial. To see that this function is not coercive, we observe that along the line $x_2 = -1$, $f(x_1, x_2) = 0, x_1 \in \mathbb{R}$. Suppose for a contradiction that f is coercive, then the fact that

$$\lim_{|x|\to\infty} f(x) = \infty$$

yields that there exists an $M > 0$ such that

$$f(x) > 1, \quad |x| > M$$

We now let $x = (x_1, x_2) = (M, -1)$, and we have that $|x| = \sqrt{M^2+1} > M$, yet $f(x) = 0$, a contradiction hence establishing that f is not coercive. We now fix $\alpha \in \mathbb{R}$ and consider the limits

$$\lim_{|x_1|\to\infty} f(x_1, \alpha x_1) \quad \text{and} \quad \lim_{|x_2|\to\infty} f(\alpha x_2, x_2)$$

By symmetry, we can consider the first limit since when both limits exist, they are equal. The first case is when $\alpha = 0$, where

$$\lim_{|x_1|\to\infty} f(x_1, \alpha x_1) = \lim_{|x_1|\to\infty} f(x_1, 0) = \lim_{|x_1|\to\infty} (x_1+1)^2 = \infty$$

The second case is when $\alpha \in \mathbb{R} \setminus \{0\}$. Now,

$$\begin{aligned} f(x_1, \alpha x_1) &= (x_1+1)^2(\alpha x_1+1)^2 = (x_1^2+2x_1+1)(\alpha^2 x_1^2 + 2\alpha x_1 + 1) \\ &= \alpha^2 x_1^4 + (2\alpha+2\alpha^2)x_1^3 + (1+\alpha^2 4\alpha)x_1^2 + (2+2\alpha)x_1 + 1 \\ &= x_1^4\left[\alpha^2 + (2\alpha+2\alpha^2)\frac{1}{x_1} + (1+\alpha^2 4\alpha)\frac{1}{x_1^2} + (2+2\alpha)\frac{1}{x_1^3} + \frac{1}{x_1^4}\right] \end{aligned}$$

Consider the function

$$h(\alpha, x_1) = (2\alpha+2\alpha^2)\frac{1}{x_1} + (1+\alpha^2 4\alpha)\frac{1}{x_1^2} + (2+2\alpha)\frac{1}{x_1^3} + \frac{1}{x_1^4}$$

Now, since all negative powers of x_1 tend to zero as $|x_1| \to \infty$, we have by the algebra of limits that there exists some $M > 0$ and $|x_1| > M$, we have that $|h| < \frac{\alpha^2}{2}$. This finally gives that

$$f(x_1, \alpha x_1) > x_1^4\left[\alpha^2 - \frac{\alpha^2}{2}\right] = x_1^4\frac{\alpha^2}{2} \geq M^4\frac{\alpha^2}{2}, \quad |x_1| > M.$$

Since, $M > 0$ is arbitrary, we have that as $f(x_1, \alpha x_1) \to \infty$ as $|x_1| \to \infty$. The same argument applied to the second limit gives that

$$\lim_{|x_2|\to\infty} f(\alpha x_2, x_2) = \infty$$

as required.

1.2 Part (ii)

To find all stationary points of f, we compute its gradient

$$\nabla f(x) = [\partial_1 f(x_1,x_2), \partial_2 f(x_1,x_2)] = [16x_1^3 - 8x_1x_2, 2x_2 - 4x_1^2]$$

Setting the above expression to zero yields the system of equations

$$\begin{cases} 16x_1^3 - 8x_1x_2 = 0 \\ 2x_2 - 4x_1^2 = 0 \end{cases}$$

with solution $x_2 = 2x_1^2, x_1 \in \mathbb{R}$. To classify these points, we notice that

$$f(x_1,x_2) = 4x_1^4 + x_2^2 - 4x_1^2x_2 + 4 = (2x_1^2 - x_2)^2 + 4 \geq 4$$

Thus, at the stationary points, $f(x_1, 2x_1^2) = 4$. This means that all stationary points are global minima. They are not strict because at any such point $(x_1, 2x_1^2)$, there is a point

$$(y_1, 2y_1^2) \in B((x_1, 2x_1^2), \delta) := \left\{(y_1, y_2 \in \mathbb{R}^2) : \|(x_1, 2x_1^2) - (y_1, y_2)\|_2 < \delta\right\}$$

for all $\delta > 0$ with $f(y_1, 2y_1^2) = 4$. Finally, this is because the the function $2x_1^2$ is continuous for $x_1 \in \mathbb{R}$.

2 Part II: Linear Least Squares

2.1 Part (i)

The recurrence relation

$$\begin{cases} x_0 = \bar{x} \\ x_i = ax_{i-1} + du_i, \quad i = 1, \ldots, N. \end{cases} \tag{1}$$

can be used to express $\mathbf{x} := \{x_1, \ldots, x_n\}$ in terms of $\mathbf{u}$ giving:

$$x_k = a^k\bar{x} + d\left(\sum_{j=1}^{k-1} a^{k-j}u_j\right), \quad k = 1, \ldots, N. \tag{2}$$

This is readily shown by induction. The base case k = 1 simply reduces to the recurrence relation (1). Now for the inductive step, suppose for k in $2, \ldots, N-1$ that

$$x_k = a^k\bar{x} + d\left(\sum_{j=1}^{k-1} a^{k-j}u_j\right)$$

then, (1) yields that

$$x_{k+1} = ax_k + du_{k+1} = a^{k+1}\bar{x} + d\left(\sum_{j=1}^{k-1} a^{k+1-j}u_j\right) + du_{k+1} = a^{k+1}\bar{x} + d\left(\sum_{j=1}^{k} a^{k+1-j}u_j\right)$$

which is in the desired form, thus completing the proof. Rewriting this in matrix form, we obtain:

$$\mathbf{x} = \mathbf{A}\mathbf{u} - \mathbf{b}$$

where

$$\mathbf{A} = d\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ a & 1 & 0 & \dots & 0 \\ a^2 & a & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ a^{N-1} & \dots & a^2 & a & 1 \end{bmatrix} \in \mathbb{R}^{N\times N} \quad \text{and} \quad \mathbf{b} = -\bar{x}\begin{bmatrix} a \\ a^2 \\ \vdots \\ a^N \end{bmatrix} \in \mathbb{R}^{\mathbb{N}} \tag{3}$$

The code listing 1 contains functions that initialise **A**, **b** and invert **A**; these functions are routinely used in the implementation of the gradient descent algorithms in the Appendix.

Further, note that

$$\mathbf{x}_{\bar{x}}^{\mathbf{u}} = [\bar{x} \quad \mathbf{x}]^T \in \mathbb{R}^{N+1}$$

Since $\mathbf{x}_{\bar{x}}^{\mathbf{u}}$ only introduces the constant term $\bar{x}^2$ in the objective functions (4), (7), (8) as compared to $\mathbf{x}$, it can thus be use interchangeably with $\mathbf{x}$ with regards to finding the optimal control signal. Thus, the minimisation problem can be reformulated as the following least squares problem:

$$\min_{\mathbf{u}\in\mathbb{R}^N} ||\mathbf{A}\mathbf{u} - \mathbf{b}||_2^2 + \frac{\gamma}{2}||\mathbf{u}||_2^2 \tag{4}$$

Now, for $\gamma > 0$, the matrix $(\mathbf{A}^\mathbf{T}\mathbf{A}) + \frac{\gamma}{2}\mathbf{I_N}$ is positive definite since

$$\mathbf{x}^\mathbf{T}\left((\mathbf{A}^\mathbf{T}\mathbf{A}) + \frac{\gamma}{2}\mathbf{I_N}\right)\mathbf{x} = ||\mathbf{A}\mathbf{x}||_2^2 + \frac{\gamma}{2}||\mathbf{x}||_2^2 \geq \frac{\gamma}{2}||\mathbf{x}||_2^2 > 0 \quad \text{for} \quad \mathbf{x} \neq \mathbf{0}$$

implying that it is invertible. Thus from lectures, there exists a unique minimiser

$$\mathbf{u}_{\mathbf{RLS}}^* = \left((\mathbf{A}^\mathbf{T}\mathbf{A}) + \frac{\gamma}{2}\mathbf{I_N}\right)^{-1}\mathbf{A}^\mathbf{T}\mathbf{b} \tag{5}$$

to the least squares problem (4).

Now, suppose $\mathbf{u_{LS}}$ solves the the least squares problem (4) with $\gamma = 0$ and let $\mathbf{u}_{\mathbf{RLS}}^*$ be the optimal solution to (4) for some $\gamma > 0$. There are two cases to consider.

2.1.1 $\mathbf{d = 0}$:

Here, $\mathbf{A} = \mathbf{O}_{N\times N}$, reducing (4) to:

$$\min_{\mathbf{u}\in\mathbb{R}^N} ||\mathbf{b}||_2^2 + \frac{\gamma}{2}||\mathbf{u}||_2^2$$

which is clearly minimised for $\mathbf{u}^*_{\mathbf{RLS}} = \mathbf{0}$. Now, $0 = ||\mathbf{u}^*_{\mathbf{RLS}}|| \leq ||\mathbf{u}_{\mathbf{LS}}||$, since the norm is non-negative.

2.1.2 $\mathbf{d \neq 0}$:

In this case, $\mathbf{A}$ is invertible with inverse:

$$\mathbf{A^{-1}} = \frac{1}{d}\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -a & 1 & 0 & \dots & 0 \\ 0 & -a & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & \dots & -a & 1 \end{bmatrix} \in \mathbb{R}^{N\times N}$$

which can easily be seen by performing the matrix multiplication. It follows that $\mathbf{A^T A}$ is positive definite. From lectures, this means that the objective function in (4) with $\gamma = 0$, namely

$$f(\mathbf{u}) = (\mathbf{Au} - \mathbf{b})^{\mathbf{T}}(\mathbf{Au} - \mathbf{b}) \quad \mathbf{u} \in \mathbb{R}^N$$

which is twice continuously differentiable, is convex since its Hessian

$$\mathbf{H}f(\mathbf{u}) = 2\mathbf{A^T A} \succ 0, \quad \mathbf{u} \in \mathbb{R}^N$$

Hence $\mathbf{u_{LS}}$ is the unique minimum

$$\mathbf{u_{LS}} = \left(\mathbf{A^T A}\right)^{-1} \mathbf{A^T b}$$

By the spectral theorem for real symmetric matrices, $\mathbf{A^T A}$ has an orthonormal eigen-basis $(\mathbf{v_i})_{i=1,\dots,N}$ with respective eigenvalues $(\lambda_\mathbf{i} > 0)_{i=1,\dots,N}$. Thus, $(\mathbf{A^T A}) + \frac{\gamma}{2}\mathbf{I_N}$ has the same eigen-basis with corresponding eigenvalues $\left(\tilde{\lambda}_\mathbf{i} = \lambda_\mathbf{i} + \frac{\gamma}{2} > 0\right)_{i=1,\dots,N}$ and $\left((\mathbf{A^T A}) + \frac{\gamma}{2}\mathbf{I_N}\right)^{-1}$ has the same eigen-basis as above with respective eigenvalues

$$\left(\left(\lambda_i + \frac{\gamma}{2}\right)^{-1} > 0\right)_{i=1,\dots,N}$$

Now,

$$\mathbf{A^T b} = \sum_{i=1}^{N} \beta_i \mathbf{v_i}, \quad \beta_i \in \mathbb{R}, \quad \text{for} \quad i = 1, \dots, N$$

in terms of the eigen-basis and using (5) we compute:

$$\begin{aligned}||\mathbf{u}^*_{\mathbf{RLS}}||_2^2 &= \left|\left|\left((\mathbf{A}^{\mathbf{T}}\mathbf{A}) + \frac{\gamma}{2}\mathbf{I_N}\right)^{-1}\mathbf{A}^{\mathbf{T}}\mathbf{b}\right|\right|_2^2 = \left|\left|\sum_{i=1}^{N}\left((\mathbf{A}^{\mathbf{T}}\mathbf{A}) + \frac{\gamma}{2}\mathbf{I_N}\right)^{-1}\beta_i\mathbf{v_i}\right|\right|_2^2 \\ &= \left|\left|\sum_{i=1}^{N}\frac{\beta_i}{\lambda_i+\frac{\gamma}{2}}\mathbf{v_i}\right|\right|_2^2 = \sum_{i=1}^{N}\frac{\beta_i^2}{\left(\lambda_i+\frac{\gamma}{2}\right)^2} \leq \sum_{i=1}^{N}\frac{\beta_i^2}{\lambda_i^2} \\ &= \left|\left|\sum_{i=1}^{N}\frac{\beta_i}{\lambda_i}\mathbf{v_i}\right|\right|_2^2 = \left|\left|\sum_{i=1}^{N}\left(\mathbf{A}^{\mathbf{T}}\mathbf{A}\right)^{-1}\beta_i\mathbf{v_i}\right|\right|_2^2 = \left|\left|\left(\mathbf{A}^{\mathbf{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathbf{T}}\mathbf{b}\right|\right|_2^2 = ||\mathbf{u_{LS}}||_2^2\end{aligned}$$

finally yielding

$$||\mathbf{u}^*_{\mathbf{RLS}}||_2 \leq ||\mathbf{u_{LS}}||_2$$

as required.

2.2 Part ii)

The objective function to minimise is

$$||\mathbf{Au}-\mathbf{b}||_2^2 + \frac{\gamma}{2}||\mathbf{u}||_2^2, \quad \mathbf{u} \in \mathbb{R}^N \tag{6}$$

with parameters

$$N = 40, a = 1, d = -0.01, \bar{x} = 1, \gamma \in \{0.001, 0.01, 0.1, 1\}.$$

In light of the discussion global minimsers in the form (5) can be found for the above settings and are plotted in figure 1. The code used to compute and plot the above optimal control signals and their respective trajectories can be found in code listing 1 in the appendix.

With regards to the optimal Control Signals, figure 1 shows that increasing γ has the effect of penalising large controls (high energy usage). This is reflected in the signals taking smaller and smaller values at earlier times, with all signal curves decaying in time to zero since the objective function also includes the ℓ_2 norm of $\mathbf{x}^{\mathbf{u}_{\bar{x}}}$.

Additionally, for the optimal trajectories, figure 2 shows that increasing γ has the effect of slowing the dampening of the optimal trajectory to zero. This is consistent with the effect on the optimal controls as for higher values of γ, smaller controls are allowed, thereby mostly allowing for minor corrections to the optimal trajectories.

For the code computing the exact least square solutions, please see code listing 2 in the appendix.

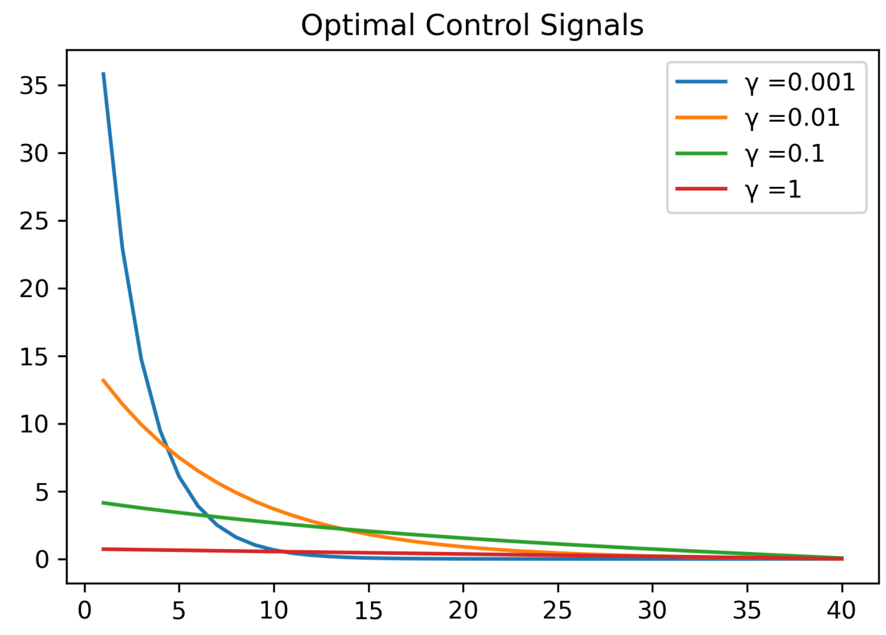

Figure 1: Optimal Control Signals
$N = 40, a = 1, d = -0.01, \bar{x} = 1$

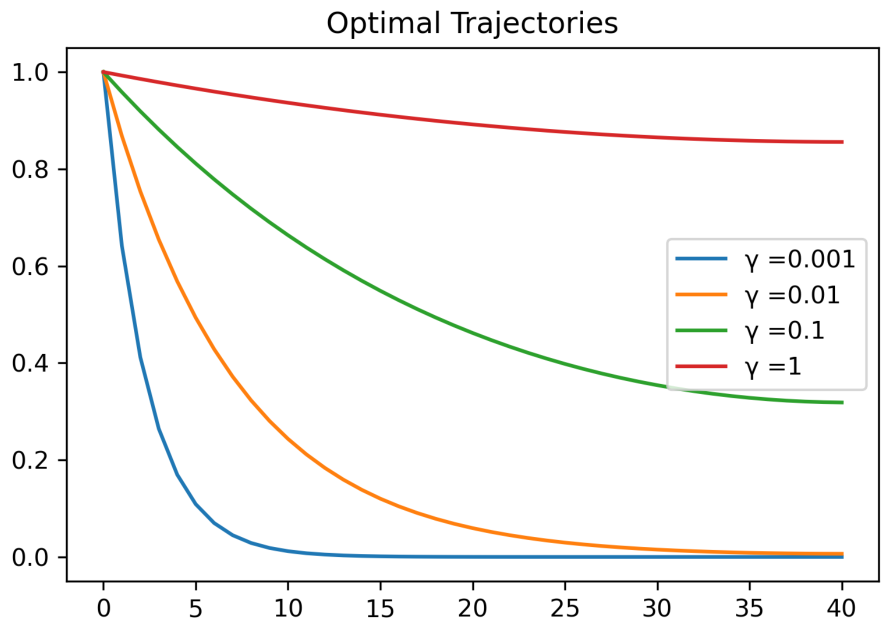

Figure 2: Optimal Trajectories
$N = 40, a = 1, d = -0.01, \bar{x} = 1$

2.3 Part iii)

In order to minimise the objective function

$$\|\mathbf{x}_{\bar{x}}^{\mathbf{u}}\|_2^2 + \frac{\gamma}{2}\|\mathbf{u}\|_2^2 - \sum_{i=1}^{N} \log(u_{max-u} - u_i), \quad \gamma, \delta > 0 \tag{7}$$

with settings

$$N = 40, a = 1, d = -0.01, \bar{x} = 1, u_{max} = 8, \gamma = \delta = 10^{-2},$$

gradient descent with a constant step size of $t = 1$, an initial value of

$$\mathbf{u} = \left[\frac{u_{max}}{2}, \dots, \frac{u_{max}}{2}\right] \in \mathbb{R}^N$$

and a tolerance of 10^{-3} was used. Please refer to the code listing 3 in the appendix for the implementation of the algorithm itself.

One observes from figure 3 that the optimal controls are bounded above by $u_{max} = 8$. This is expected from the fact that the cost of approaching u_{max} in the objective function 7 becomes infinite. Also, the controls stay approximately constant, with a gradual decrease below the maximal value u_{max} for the first five or so time steps and subsequently a more rapid decay to zero is observed.

Similarly, from figure 4, the trajectory initialised is eventually driven close to zero (and attains small negative values for the final values of time) at an almost linear rate for the first five or so time steps (corresponding to the control signals being approximately constant, as can be seen from the relation 1) and then one observes a 'hyperbolic' decay for times thereafter.

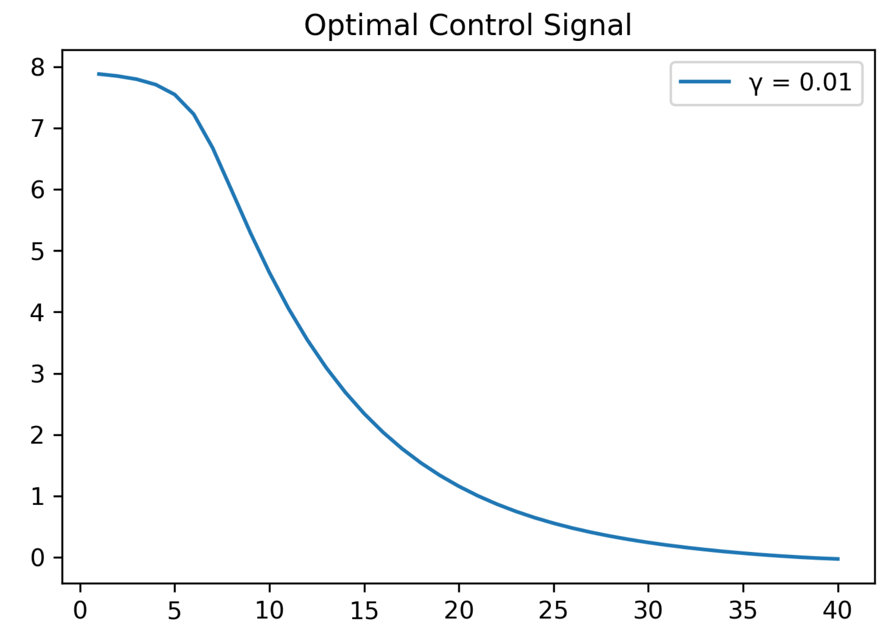

Figure 3: Optimal Control Signal
$N = 40, a = 1, d = -0.01, \bar{x} = 1, u_{max} = 8, \gamma = \delta = 10^{-2}$

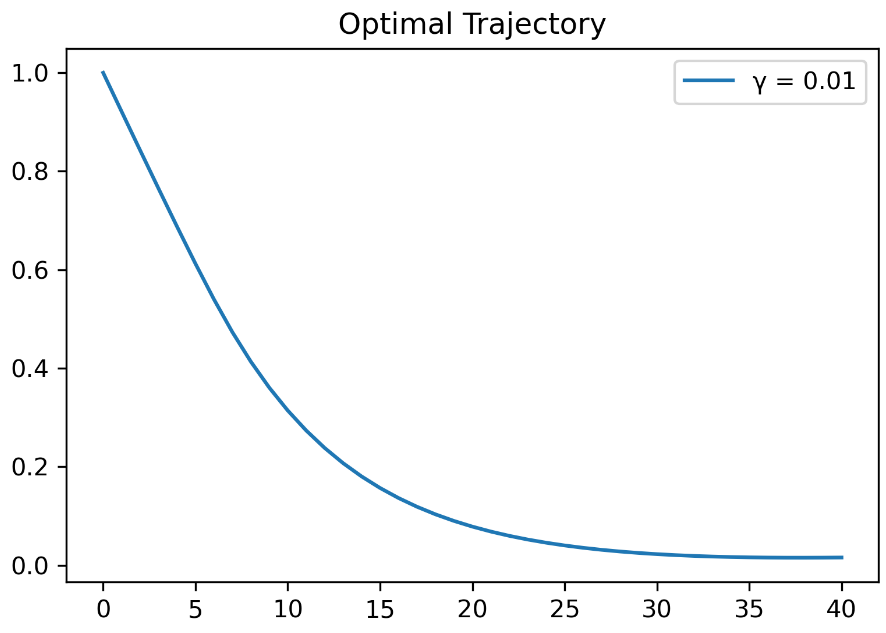

Figure 4: Optimal Trajectory
$N = 40, a = 1, d = -0.01, \bar{x} = 1, u_{max} = 8, \gamma = \delta = 10^{-2}$

2.4 Part iv)

The objective function under consideration is

$$||\mathbf{x}_{\bar{x}}^{\mathbf{u}}||_2^2 + \frac{\gamma_2}{2}||\mathbf{u}||_2^2 + \gamma_1 \cdot \mathcal{L}_\epsilon(\mathbf{u}), \quad \gamma_2, \gamma_1, \epsilon > 0 \tag{8}$$

where

$$\mathcal{L}_\epsilon(\mathbf{u}) = \sum_{i=1}^{N} f_i(\mathbf{u}) \tag{9}$$

and

$$f_i(\mathbf{u}) = \begin{cases} \frac{1}{2}u_i^2, & |u_i| \leq \epsilon \\ \epsilon(|u_i| - \frac{1}{2}\epsilon), & \text{otherwise} \end{cases}$$

It is indeed true that $\mathcal{L}_\epsilon(\mathbf{u})$ is differentiable. To show this, it is sufficient to show that the f_i are all continuously differentiable. Thus, fix i from $1,\ldots,N$ and consider f_i. Now, it is clear that

$$\frac{\partial f_i}{\partial u_j}(\mathbf{u}) = 0, \quad j \neq i, \mathbf{u} \in \mathbb{R}^N$$

since only u_i is used in the definition of f_i. In the case where $j = i$, we have that if $|u_i| < \epsilon$, $f_i(\mathbf{u}) = \frac{1}{2}u_i^2$ (hence smooth) in a neighborhood of that point,

$$\frac{\partial f_i}{\partial u_i}(\mathbf{u}) = u_i, \quad |u_i| < \epsilon$$

Similarly, if $|u_i| > \epsilon$, $f_i(\mathbf{u}) = \epsilon(|u_i| - \frac{1}{2}\epsilon)$ (hence smooth since $|u_i|$ is bounded away from zero) in a neighborhood of that point,

$$\frac{\partial f_i}{\partial u_i}(\mathbf{u}) = \epsilon \cdot \text{sign}(u_i), \quad |u_i| > \epsilon$$

Now, in the boundary case where $|u_i| = \epsilon$ (in particular the case where $u_i = \epsilon$, the case $u_i = -\epsilon$ is similar), the difference quotients

$$\frac{f_i(\mathbf{u} + h\mathbf{e_i}) - f_i(\mathbf{u})}{h} \quad \text{and} \quad \frac{f_i(\mathbf{u} - h\mathbf{e_i}) - f_i(\mathbf{u})}{-h}, \quad h > 0$$

are equal to

$$\frac{\epsilon(\epsilon + h - \frac{1}{2}\epsilon) - \frac{1}{2}\epsilon^2}{h} \quad \text{and} \quad \frac{\frac{1}{2}(\epsilon - h)^2 - \frac{1}{2}\epsilon^2}{-h}, \quad h > 0$$

respectively and both converge to ϵ as $h \to 0^+$. Thus $f_i(u)$ is continuously differentiable for $|u_i| = \epsilon$, hence for all $\mathbf{u} \in \mathbb{R}^N$ by the above. This implies that $\mathcal{L}_\epsilon(\mathbf{u})$ is continuously differentiable for all $\mathbf{u} \in \mathbb{R}^N$ with gradient

$$\nabla\mathcal{L}_\epsilon(\mathbf{u}) = \sum_{i=1}^{N} \left\{ \begin{array}{ll} u_i, & |u_i| \leq \epsilon \\ \epsilon \cdot \text{sign}(u_i), & |u_i| > \epsilon \end{array} \right\} \mathbf{e_i} \in \mathbb{R}^N \tag{10}$$

Now, the function $\mathcal{L}_\epsilon(\mathbf{u})$ for $\epsilon > 0$ is an approximation the the ℓ_1 norm in the sense that in the ℓ_∞ ball of radius ϵ about the origin (9) is precisely the ℓ_2 norm squared of $\mathbf{u}$. Outside of that domain, $\mathcal{L}_\epsilon(\mathbf{u})$ gets mixed ℓ_2 from the $|u_i| \leq \epsilon$ and affinely transformed ℓ_1 norm contributions from precisely the $|u_i| > \epsilon$, chosen so that the values and the one sided derivatives of $\mathcal{L}_\epsilon(\mathbf{u})$ match when some $|u_i| = \epsilon$.

Note the case where all $|u_i| > \epsilon$, $\mathcal{L}_\epsilon(\mathbf{u})$ just becomes

$$\mathcal{L}_\epsilon(\mathbf{u}) = \epsilon\left(||\mathbf{u}||_1 - N\frac{\epsilon}{2}\right)$$

that is the ℓ_1 norm of $\mathbf{u}$ up to a linear transformation. Since, the ℓ_1 norm promotes sparsity in the entries of $\mathbf{u}$ (see (1)), it follows that $\mathcal{L}_\epsilon(\mathbf{u})$ acts similarly as a regulariser penalising lack of sparsity in the optimal control signals.

Now, in order to minimise the objective function (8) with parameters

$$N = 40, a = 1, d = -0.01, \bar{x} = 1, \epsilon = 3, \delta = 10^{-2}, \gamma_1 = \gamma_1 = 0, \gamma_2 = 0.01$$

gradient descent with backtracking was used with parameters: $\alpha = 0.1, \beta = 0.1, s = 1$, initial value for $\mathbf{u}$

$$\mathbf{u} = [1, \ldots, 1] \in \mathbb{R}^N$$

and a tolerance of 10^{-3}. In the second case with parameters

$$N = 40, a = 1, d = -0.01, \bar{x} = 1, \epsilon = 3, \delta = 10^{-2}, \gamma_1 = 0.01, \gamma_2 = 0$$

gradient descent with backtracking was used with parameters: $\alpha = 0.1, \beta = 0.1, s = 1$, initial value for $\mathbf{u}$

$$\mathbf{u} = [1, \ldots, 1] \in \mathbb{R}^N$$

and a tolerance of 10^{-3}. Please refer to code listing 4 in the appendix for the exact implementation of the algorithm.

Comparing the two cases $\gamma_1 = 0, \gamma_2 = 0.01$ and $\gamma_1 = 0.01, \gamma_2 = 0$ (blue and orange curves respectively in figures 5, 6), we see that in the former case, figure 5 shows the optimal control signal rapidly decaying to the $\epsilon = 3$ value and as a result one observes the sparsity in the signal with a few large signals for small times, but otherwise near zero controls for further times. In the latter case, since the regualrisation is always the ℓ_2 norm squared, sparsity is nor penalised heavily and instead, on observes a more gradual decrease in the control signal to zero as time progresses. In other words, signals that exceed the $\epsilon = 3$ threshold are more heavily penalised, and this can be seen in the optimal control plot 5.

The above is reflected in the optimal trajectories as can be seen in figure 6. The very large controls for the latter case (orange curve in figure 5) drive the optimal trajectory's value close to zero rapidly, as one would have expected (by inspecting relation 2) and then the small control signals that follow drive the $\mathbf{x}_{\bar{x}}^{\mathbf{u}}$ more slowly. Finally, the optimal trajectories for the former case (blue curve in figure 6), smaller

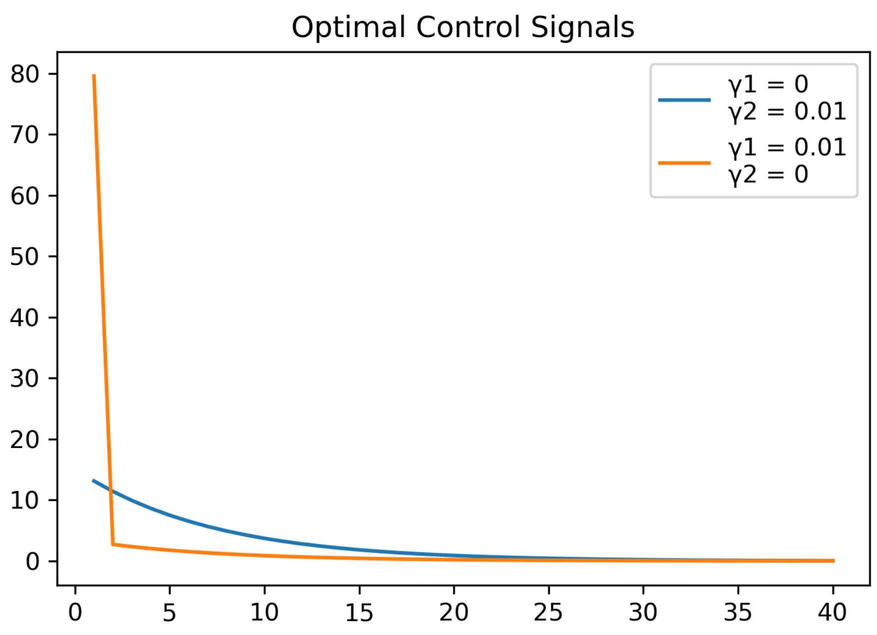

Figure 5: Optimal Control Signals
$N = 40, a = 1, d = -0.01, \bar{x} = 1, \epsilon = 3, \delta = 10^{-2}$

but slower to decrease signals yield a curve that decays to zero more slowly yielding that the values on the ℓ_2 regularisation's optimal trajectory (blue) are greater than those on the approximate ℓ_1 regularisation's trajectory (orange).

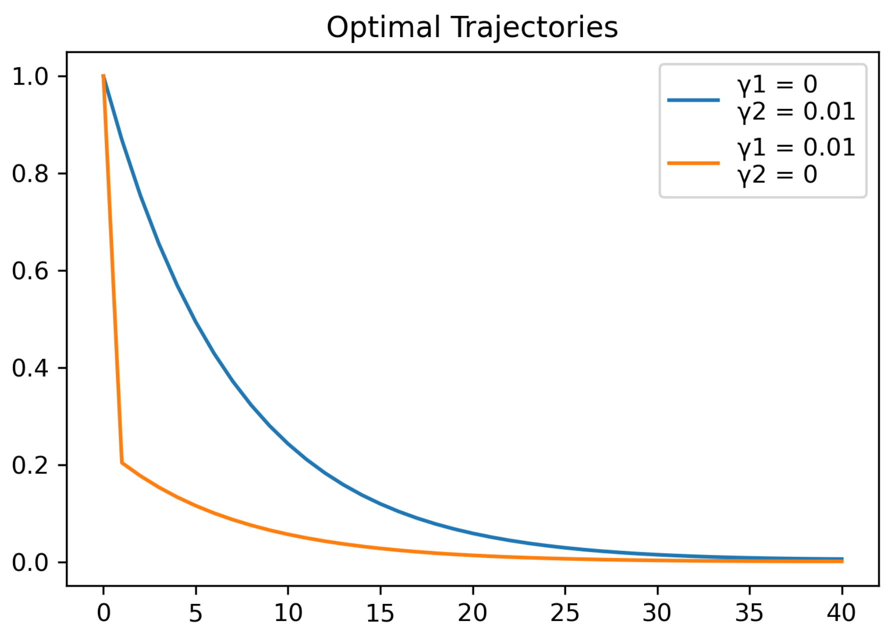

Figure 6: Optimal Trajectories
$N = 40, a = 1, d = -0.01, \bar{x} = 1, \epsilon = 3, \delta = 10^{-2}$

3 Appendix

Please refer to my github repository, for a runnable version of the code. For completeness, I have included the code below in latex form.

```python
import numpy as np

def A_inv(N,a,d):
    A = np.zeros((N, N))
    for i in range(N-1):
        A[i+1,i] = -a
    for i in range(N):
        A[i,i] = 1
    return A/d

def b(N, x_bar, a):
    return np.array([-x_bar*(a)**n for n in range(1,N+1)])
```

Listing 1: Auxiliary code for returning $\mathbf{x}_{\bar{\mathbf{x}}}^{\mathbf{u}}$ in terms of control signal $\mathbf{u}$

```python
import numpy as np
#Regularised least squares
N = 40
a = 1
d = -0.01
x_bar = 1
γ = [10**(-3), 10**(-2), 0.1, 1]

```

```
A = np.linalg.inv(A_inv(N,a,d))
b = b(N,x_bar,a)
#The columns of U correspond to the control signals for each value of
    γ
U = np.zeros((N,4))
# X_u are the optimal trajectories:
X_u = np.zeros((N+1,4))

for i in range(4):
    #U[:,i] is the regularised least squares solution for γ[i]
    U[:,i] = np.matmul(np.matmul(np.linalg.inv(np.matmul(np.transpose
   (A),A)+γ[i]/2*np.identity(N)), np.transpose(A)), b)
    #compute the optimal Trajectories X_u
    X_u[1:,i] = np.matmul(A, U[:,i])-b
    X_u[0, i] = x_bar

import matplotlib.pyplot as plt


plt.title('Optimal Control Signals')
for i in range(4):
    plt.plot(np.array(range(1, N+1)), U[:,i], label = 'γ =' + str(γ[i
   ]), markersize = 1)

leg = plt.legend()
plt.savefig('2ii) Optimal Control Signals.png',dpi=300)

plt.show()

plt.title('Optimal Trajectories')
for i in range(4):
    plt.plot(X_u[:,i], label = 'γ =' + str(γ[i]), markersize = 1)

leg = plt.legend()
plt.savefig('2ii) Optimal Trajectories.png',dpi=300)
plt.show()
```

Listing 2: Section II part ii)

```
#Gradient descent with fixed time step
import numpy as np
N = 40
a = 1
d = -0.01
x_bar = 1
umax = 8
γ = 10**(-2)
δ = 10**(-2)
ϵ = 10**(-3)
u0 = [umax/2]*N

A = np.linalg.inv(A_inv(N,a,d))
b = b(N,x_bar,a)


def descent(s, u0, Gf):
    #use params: timestep t = 1, ϵ tolerance 10**-3
    u = u0
    grad = Gf(u)
    G = np.linalg.norm(grad)
    while G > ϵ  :
        u = u - s*grad
        grad = Gf(u)
        G = np.linalg.norm(grad)
    return u

#The columns of U conrrespond to the control signals for each value of γ
U = [0]*N
# X_u are the optimal trajectories:
X_u = [0]*(N+1)

#Gradient of objective function
Gf = lambda u: np.array([2*np.matmul(np.matmul(np.transpose(A),A),u)[i]-2*(np.matmul(np.transpose(A),b)[i])+δ/(umax-u[i])+γ*u[i] for i in range(N)])
#computing optimal controls U
U = descent(1, u0, Gf)
#computing Optimal Trajectories X_u
X_u[1:] = np.matmul(A, U)-b
X_u[0] = x_bar

import matplotlib.pyplot as plt

plt.title('Optimal Control Signal')
plt.plot(np.array(range(1, N+1)),U, label = 'γ = 0.01', markersize = 1)
leg = plt.legend()
plt.savefig('2iii) Optimal Control Signal.png',dpi=300)
plt.show()

plt.title('Optimal Trajectory')
plt.plot(X_u, label = 'γ = 0.01', markersize = 1)
leg = plt.legend()
plt.savefig('2iii) Optimal Trajectories.png',dpi=300)
plt.show()
```

Listing 3: Section II Part iii)

```
#Gradient descent with backtracking
import numpy as np
N = 40
a = 1
d = -0.01
x_bar = 1
ϵ = 3
#tolerance parameter
ω = 10**(-3)

γ1 = [0, 10**(-2)]
γ2 = [10**(-2), 0]
u0 = [1]*N
```

```


A = np.linalg.inv(A_inv(N,a,d))
b = b(N,x_bar,a)

#l1 regulariser
def L_ϵ(u):
    s = 0
    for i in range(N):
        if abs(u[i]) <=ϵ:
            s+= 0.5*(u[i])**2
        else:
            s+= ϵ*(abs(u[i])-ϵ/2)
    return s

#Gradient of ℓ1 regualriser
def GL_ϵ(u):
    v = [0]*N
    for i in range(N):
        if abs(u[i]) <=ϵ:
            v[i] = u[i]
        else:
            v[i] = ϵ*np.sign(u[i])

    return np.array(v)


#Gradient descent with backtracking
def backtracking(α, β, s, u0, f, Gf):
    # parameters: α = 0.1, β = 1, s = 1, u0 = [1]*N
    u = u0
    G = np.linalg.norm(Gf(u))
    grad = Gf(u)
    func = f(u)
    while G > ω :
        t = s
        d = - grad
        while (f(u)-f(u+t*d)+α*t*np.dot(Gf(u),d)) < 0:
            t = β*t
        u = u + t*d
        func = f(u)
        G = np.linalg.norm(Gf(u))
        grad = Gf(u)
    return u

#The columns of U correspond to the control signals for each value of
    γ
U2 = [0]*N
# X_u are the optimal trajectories:
X_u2 = [0]*(N+1)


#objective function for case γ1 = 0,γ2 = 0.01
f1 = lambda u: np.linalg.norm(np.matmul(A,u)-b)**2+γ2/2*np.linalg.
    norm(u)**2+γ1[0]*L_ϵ(u)
```

```
#gradient of objective function for case $\gamma_1 = 0, \gamma_2 = 0.01$
Gf1 = lambda u: np.array([2*np.matmul(np.matmul(np.transpose(A),A),u)[i]-2*(np.matmul(np.transpose(A),b)[i])+$\gamma_2$[0]*u[i]+$\gamma_1$[0]*GL_$\epsilon$(u)[i] for i in range(N)])
#optimal controls for case $\gamma_1 = 0, \gamma_2 = 0.01$
U1 = backtracking(0.1, 0.1, 1, u0, f1, Gf1)
#optimal trajectory for case $\gamma_1 = 0, \gamma_2 = 0.01$
X_u1[1:] = np.matmul(A, U1)- b
X_u1[0] = x_bar
#objective function for case $\gamma_1 = 0.01, \gamma_2 = 0$
f2 = lambda u: np.linalg.norm(np.matmul(A,u)-b)**2+$\gamma_2$[1]/2*np.linalg.norm(u)**2+$\gamma_1$[1]*L_$\epsilon$(u)
#gradient of objective function for case $\gamma_1 = 0.01, \gamma_2 = 0$
Gf2 = lambda u: np.array([2*np.matmul(np.matmul(np.transpose(A),A),u)[i]-2*(np.matmul(np.transpose(A),b)[i])+$\gamma_2$[1]*u[i]+$\gamma_1$[1]*GL_$\epsilon$(u)[i] for i in range(N)])
#optimal controls for case $\gamma_1 = 0.01, \gamma_2 = 0$
U2 = backtracking(0.1, 0.1, 1, u0, f2, Gf2)
#optimal trajectory for case $\gamma_1 = 0.01, \gamma_2 = 0$
X_u2[1:] = np.matmul(A, U2)-b
X_u2[0] = x_bar

import matplotlib.pyplot as plt

plt.title('Optimal Control Signals')
plt.plot(np.array(range(1, N+1)), U1, label = '$\gamma_1$ = 0\n$\gamma_2$ = '+str(10**(-2)), markersize = 1)
plt.plot(np.array(range(1, N+1)) ,U2, label = '$\gamma_1$ = '+str(10**(-2))+'\n$\gamma_2$ = 0', markersize = 1)
leg = plt.legend()
plt.savefig('2iv) Optimal Control Signals.png',dpi=300)

plt.show()

plt.title('Optimal Trajectories')
plt.plot(X_u1, label = '$\gamma_1$ = 0\ n  2 = '+str(10**(-2)), markersize = 1)
plt.plot(X_u2, label = '$\gamma_1$ = '+str(10**(-2))+'\n$\gamma_2$ = 0', markersize = 1)
leg = plt.legend()
plt.savefig('2iv) Optimal Trajectories.png',dpi=300)

plt.show()
```

Listing 4: Section II Part iv)

4 References

[1] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009. pages 11