Optimization and Algorithms
Project report

Group 28
Franisco Melo 84053 and Rodrigo Rego 89213

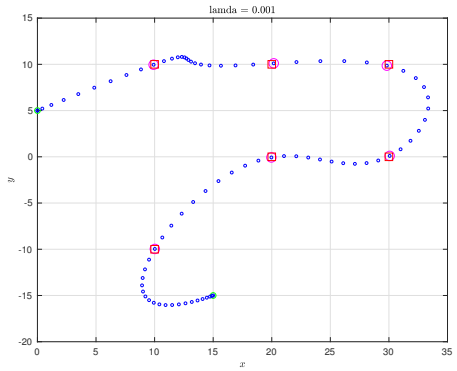
Contents

1 P1 - Task 1

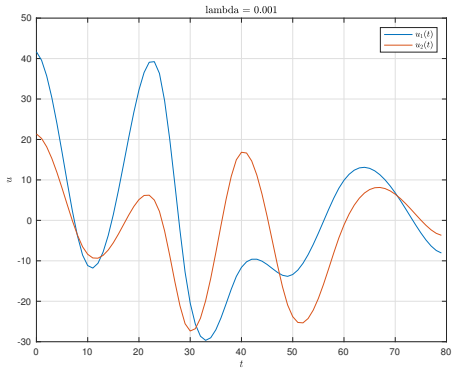
Listing 1: Task 1 Code

```
%% Task1
close all; clear all; clc;

load Initial_Constants;

lambdas=[0.001 0.01 0.1 1 10 100 1000];

for l=1:length(lambdas)

    lambda=lambdas(l);

    cvx_begin quiet
    variable x(4,T+1);
    variable u(2,T+1);

    f1=0;
    for i=1:K
        f1=f1+square_pos(norm(E*x(:,tau(i)+1)-W(:,i)));
    end

    f2=0;
    for j=2:T
        f2=f2 + square_pos(norm(u(:,j)-u(:,j-1)));
    end

    f= f1+ lambda*f2;

    minimize(f);

    subject to
    x(:,1)==[pinitial';0;0];
    x(:,T+1)==[pfinal';0;0];

    for a=1:T
        norm(u(:,a)) <= Umax;
        x(:,a+1)==A*x(:,a)+ B*u(:,a);
    end

    cvx_end;

    figure();
    plot(pinitial(1),pinitial(2),'go',pfinal(1), pfinal(2),'go');

    hold on;
    plot(x(1,:), x(2,:),'bo','MarkerSize',3);
    hold on;
```

```
47
48      for q=1:K
49          plot(x(1,tau(q)+1), x(2,tau(q)+1),'mo','MarkerSize',10);
50          hold on;
51      end
52
53      hold on;
54      plot(W(1,:),W(2,:),'rs','MarkerSize',10);
55      grid on;
56      title( sprintf('lambda = %g', lambda),'interpreter','latex');
57      xlabel('$x$','interpreter','latex');
58      ylabel('$y$','interpreter','latex');
59
60
61      time=linspace(0,T-1,T);
62
63      figure();
64      plot(time,u(1,1:end-1),time,u(2,1:end-1));
65      grid on;
66      title(sprintf('lambda = %g', lambda),'interpreter','latex');
67      xlabel('$t$','interpreter','latex');
68      ylabel('$u$','interpreter','latex');
69      le=legend('$u_{1}(t)$', '$u_{2}(t)$');
70      set(le,'interpreter','latex');
71
72      counter=0;
73
74      for t=2:T
75          if norm(u(:,t)-u(:,t-1)) > 10^-6
76              counter=counter+1;
77          end
78      end
79
80      fprintf('N of times that optimal control changes (lambda= %g)- %g\n',...
81          lambda,counter);
82
83      mean_sum=0;
84      for r=1:K
85          mean_sum=mean_sum + norm(E*x(:,tau(r)+1)-W(:,r));
86      end
87
88      mean_deviation=(1/K)*mean_sum;
89      fprintf('Mean Deviantion (lambda=%g) - %g\n\n', lambda, mean_deviation);
90  end;
```

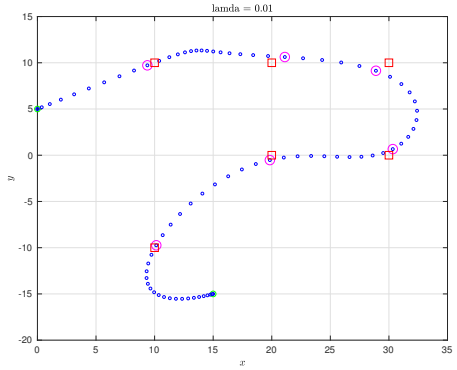


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10^{-3}$ with ℓ_2^2 regularizer.

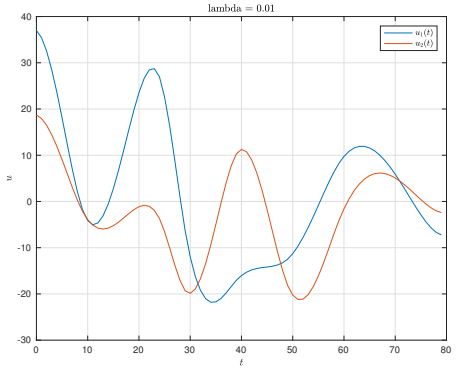


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-3}$ with ℓ_2^2 regularizer.

Figure 1: Case $\lambda = 10^{-3}$. Position of the robot (a) and the control signal (b).

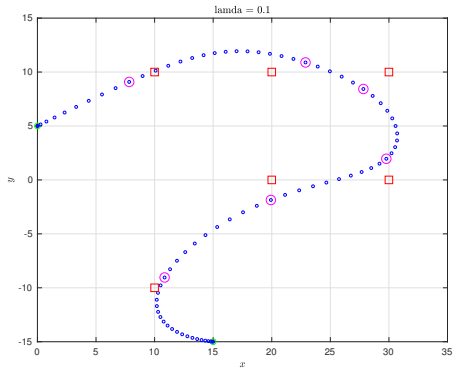


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10^{-2}$ with ℓ_2^2 regularizer.

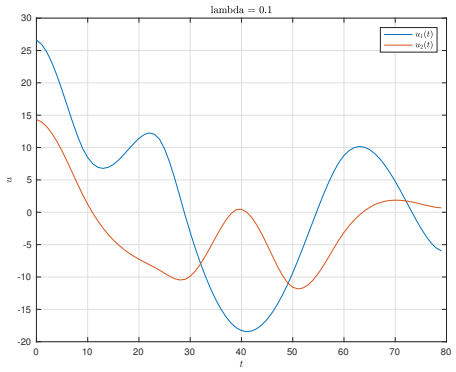


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-2}$ with ℓ_2^2 regularizer.

Figure 2: Case $\lambda = 10^{-2}$. Position of the robot (a) and the control signal (b).

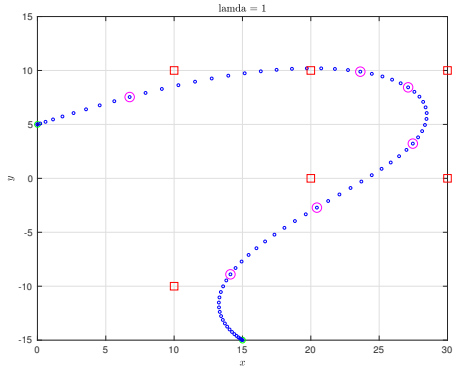


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10^{-1}$ with ℓ_2^2 regularizer.

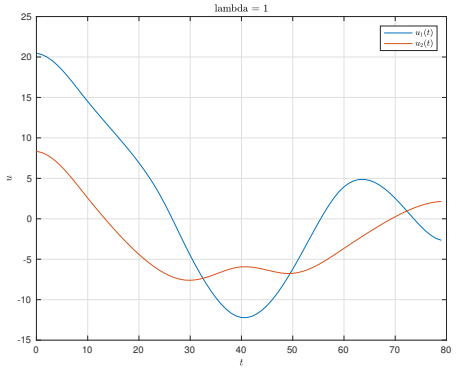


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-1}$ with ℓ_2^2 regularizer.

Figure 3: Case $\lambda = 10^{-1}$. Position of the robot and the control signal for.

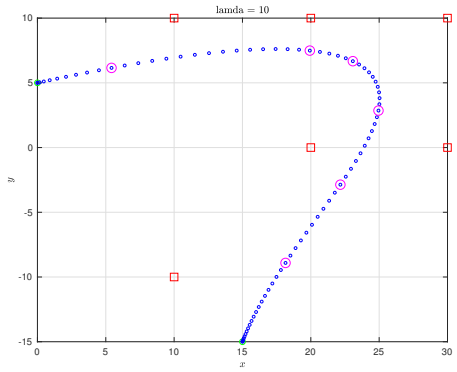


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 1$ with ℓ_2^2 regularizer.

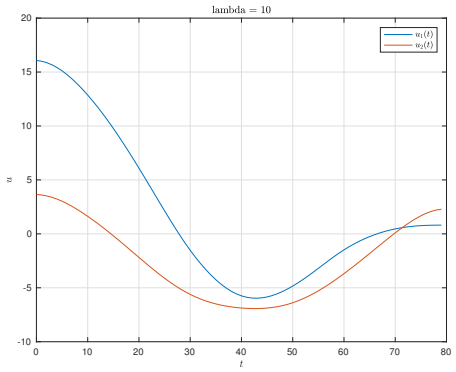


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 1$ with ℓ_2^2 regularizer.

Figure 4: Case $\lambda = 1$. Position of the robot and the control signal for.

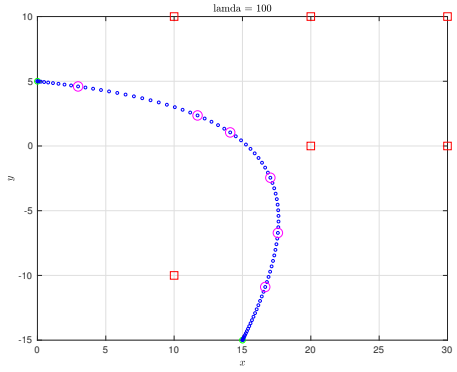


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10$ with ℓ_2^2 regularizer.

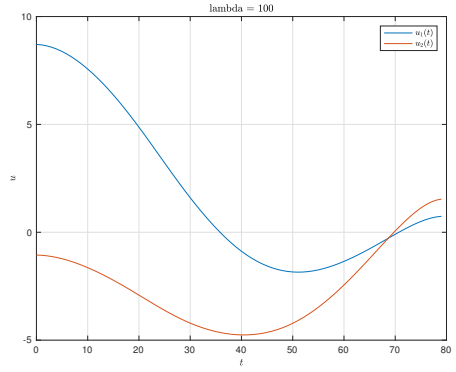


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10$ with ℓ_2^2 regularizer.

Figure 5: Case $\lambda = 10$. Position of the robot and the control signal for.

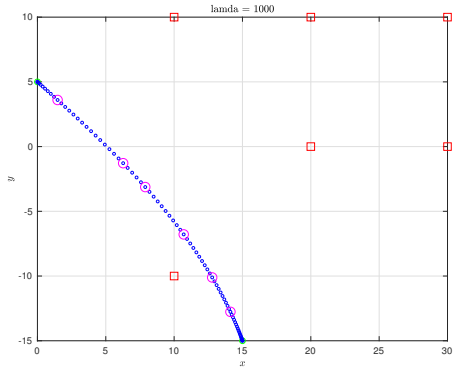


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 100$ with ℓ_2^2 regularizer.

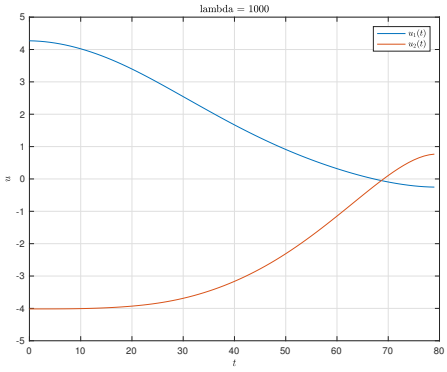


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 100$ with ℓ_2^2 regularizer.

Figure 6: Case $\lambda = 100$. Position of the robot and the control signal for.



(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 1000$ with ℓ_2^2 regularizer.



(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 1000$ with ℓ_2^2 regularizer.

Figure 7: Case $\lambda = 1000$. Position of the robot and the control signal for.

λ	Control Changes	Mean Deviation
10^{-3}	79	0.1257
10^{-2}	79	0.8242
10^{-1}	79	2.1958
1	79	3.6826
10	79	5.6317
100	79	10.9042
1000	79	15.3304

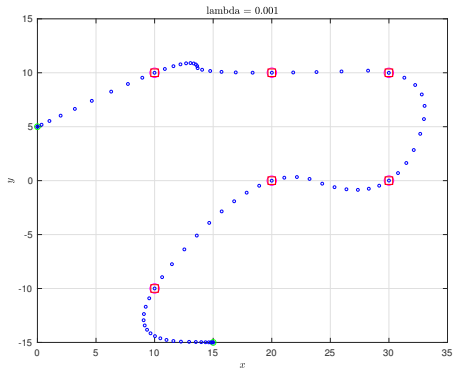
Table 1: Number of times that the optimal control signal changes and the mean deviation from the waypoints for all the values of λ

2 P1 - Task 2

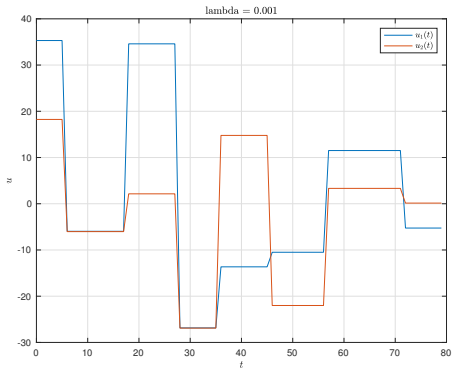
Listing 2: Task 2 Code

```
%% Task2
close all; clear all; clc;

load Initial_Constants;

lambdas=[0.001 0.01 0.1 1 10 100 1000];

for l=1:length(lambdas)

    lambda=lambdas(l);
    cvx_begin quiet
    variable x(4,T+1);
    variable u(2,T+1);

    f1=0;
    for i=1:K
        f1=f1+square_pos(norm(E*x(:,tau(i)+1)-W(:,i)));
    end

    f2=0;
    for j=2:T
        f2=f2 + norm(u(:,j)-u(:,j-1));
    end

    f= f1+ lambda*f2;

    minimize(f);

    subject to
    x(:,1)==[pinitial';0;0];
    x(:,T+1)==[pfinal';0;0];

    for a=1:T
        norm(u(:,a)) <= Umax;

        x(:,a+1)==A*x(:,a)+ B*u(:,a);
    end

    cvx_end;

    figure();
    plot(pinitial(1),pinitial(2),'go',pfinal(1), pfinal(2),'go');
    hold on;
    plot(x(1,:), x(2,:),'bo','MarkerSize',3);
    hold on;

```

```
47      for q=1:K
48          plot(x(1,tau(q)+1), x(2,tau(q)+1),'mo','MarkerSize',10);
49          hold on;
50      end
51
52      hold on;
53      plot(W(1,:),W(2,:),'rs','MarkerSize',10);
54      grid on;
55      title( sprintf('lambda = %g', lambda),'interpreter','latex');
56      xlabel('$x$','interpreter','latex');
57      ylabel('$y$','interpreter','latex');
58
59      time=linspace(0,T-1,T);
60
61      figure();
62      plot(time,u(1,1:end-1),time,u(2,1:end-1));
63      grid on;
64      title(sprintf('lambda = %g', lambda),'interpreter','latex');
65      xlabel('$t$','interpreter','latex');
66      ylabel('$u$','interpreter','latex');
67      le=legend('$u_{1}(t)$', '$u_{2}(t)$');
68      set(le,'interpreter','latex');
69
70      counter=0;
71
72      for t=2:T
73          if norm(u(:,t)-u(:,t-1)) > 10^-6
74              counter=counter+1;
75          end
76      end
77
78      fprintf("N of times that optimal control changes (lambda= %g) - %g\n",...
79          lambda,counter);
80
81      mean_sum=0;
82      for r=1:K
83          mean_sum=mean_sum + norm(E*x(:,tau(r)+1)-W(:,r));
84      end
85
86      mean_deviation=(1/K)*mean_sum;
87      fprintf('Mean Deviantion (lambda=%g) - %g\n \n', lambda, mean_deviation);
88  end;
```

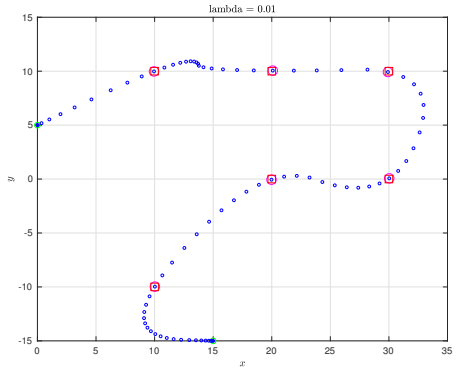


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10^{-3}$ with ℓ_2 regularizer.

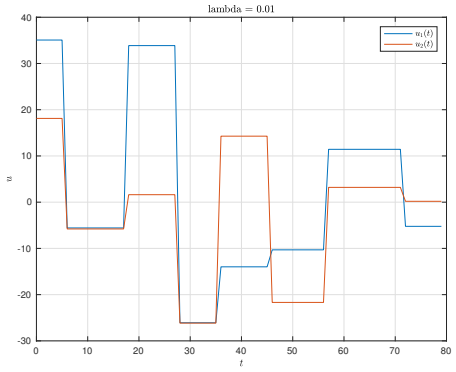


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-3}$ with ℓ_2 regularizer.

Figure 8: Case $\lambda = 10^{-3}$. Position of the robot (a) and the control signal (b).

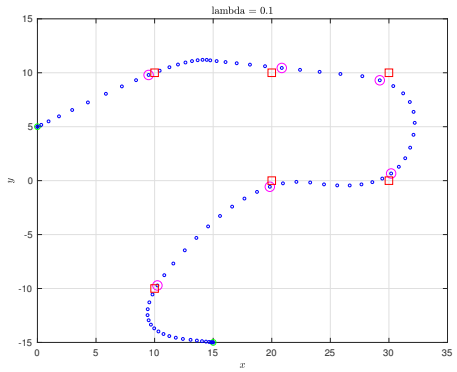


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10^{-2}$ with ℓ_2 regularizer.

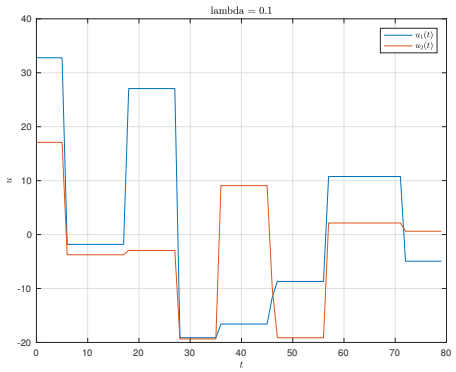


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-2}$ with ℓ_2 regularizer.

Figure 9: Case $\lambda = 10^{-2}$. Position of the robot (a) and the control signal (b).

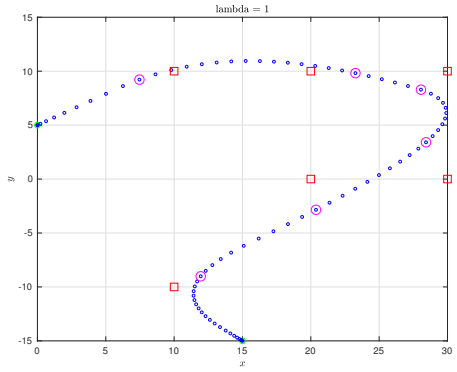


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10^{-1}$ with ℓ_2 regularizer.

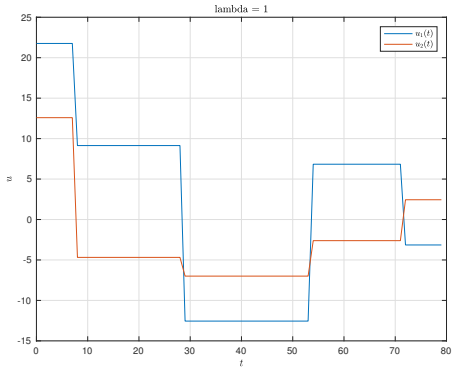


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-1}$ with ℓ_2 regularizer.

Figure 10: Case $\lambda = 10^{-1}$. Position of the robot (a) and the control signal (b).

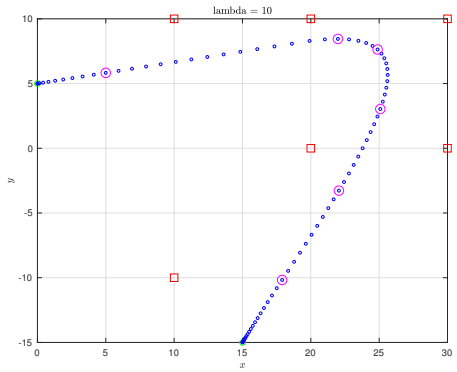


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 1$ with ℓ_2 regularizer.

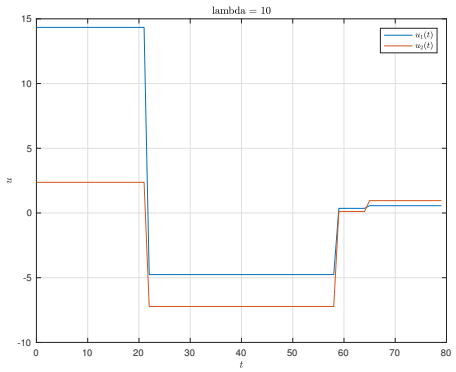


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 1$ with ℓ_2 regularizer.

Figure 11: Case $\lambda = 1$. Position of the robot (a) and the control signal (b).

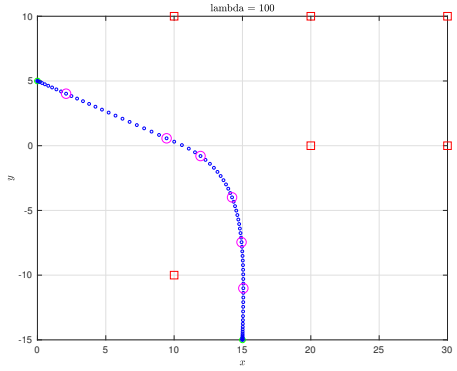


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10$ with ℓ_2 regularizer.

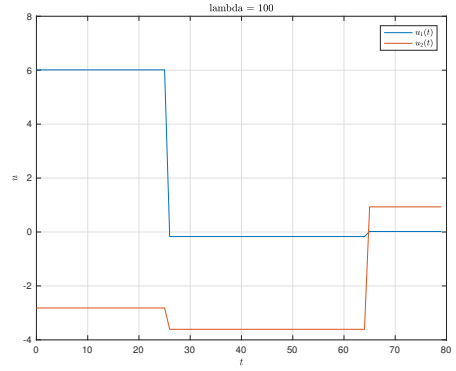


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10$ with ℓ_2 regularizer.

Figure 12: Case $\lambda = 10$. Position of the robot (a) and the control signal (b).

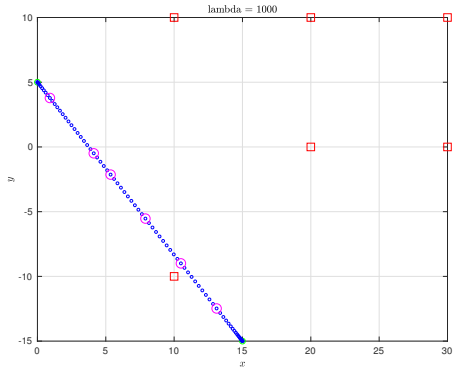


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 100$ with ℓ_2 regularizer.

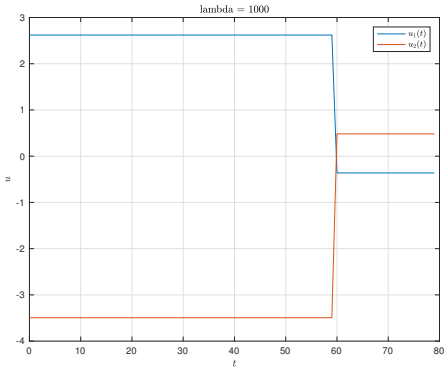


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 100$ with ℓ_2 regularizer.

Figure 13: Case $\lambda = 100$. Position of the robot (a) and the control signal (b).



(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 1000$ with ℓ_2 regularizer.



(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T - 1$. Case $\lambda = 1000$ with ℓ_2 regularizer.

Figure 14: Case $\lambda = 1000$. Position of the robot (a) and the control signal (b).

λ	Control Changes	Mean Deviation
10^{-3}	10	0.0075
10^{-2}	8	0.0747
10^{-1}	11	0.7021
1	4	2.8876
10	4	5.3689
100	4	12.5914
1000	1	16.2266

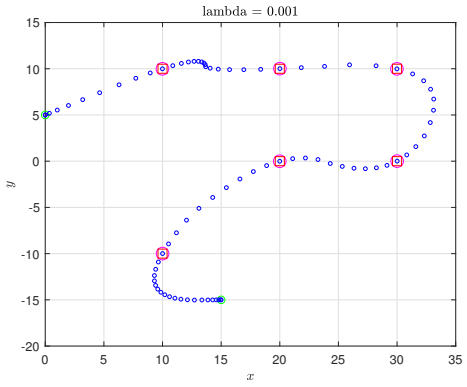
Table 2: Number of times that the optimal control signal changes and the mean deviation from the waypoints for all the values of λ

3 P1 - Task 3

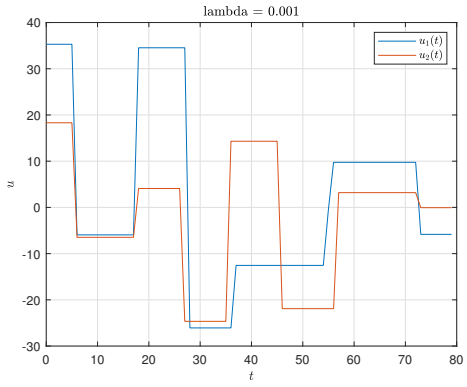
Listing 3: Task 3 Code

```
%% Task3
close all; clear all; clc;

load Initial_Constants;

lambdas=[0.001 0.01 0.1 1 10 100 1000];

for l=1:length(lambdas)

    lambda=lambdas(l);
    cvx_begin quiet
    variable x(4,T+1);
    variable u(2,T+1);

    f1=0;
    for i=1:K
        f1=f1+square_pos(norm(E*x(:,tau(i)+1)-W(:,i)));
    end

    f2=0;
    for j=2:T
        f2=f2 + norm(u(:,j)-u(:,j-1),1);%l1 norm
    end

    f= f1+ lambda*f2;

    minimize(f);

    subject to
    x(:,1)==[pinitial';0;0];
    x(:,T+1)==[pfinal';0;0];

    for a=1:T
        norm(u(:,a)) <= Umax;
        x(:,a+1)==A*x(:,a)+ B*u(:,a);
    end


    cvx_end;

    figure();
    plot(pinitial(1),pinitial(2),'go',pfinal(1), pfinal(2),'go');
    hold on;
    plot(x(1,:), x(2,:),'bo','MarkerSize',3);
    hold on;

```

```
47      for q=1:K
48          plot(x(1,tau(q)+1), x(2,tau(q)+1),'mo','MarkerSize',10);
49          hold on;
50      end
51
52      hold on;
53      plot(W(1,:),W(2,:),'rs','MarkerSize',10);
54      grid on;
55      title( sprintf('lambda = %g', lambda),'interpreter','latex');
56      xlabel('$x$','interpreter','latex');
57      ylabel('$y$','interpreter','latex');
58
59
60      time=linspace(0,T-1,T);
61      figure();
62
63      plot(time,u(1,1:end-1),time,u(2,1:end-1));
64      grid on;
65      title(sprintf('lambda = %g', lambda),'interpreter','latex');
66      xlabel('$t$','interpreter','latex');
67      ylabel('$u$','interpreter','latex');
68      le=legend('$u_{1}(t)$', '$u_{2}(t)$');
69      set(le,'interpreter','latex');
70
71      counter=0;
72
73      for t=2:T
74          if norm(u(:,t)-u(:,t-1)) > 10^-6
75              counter=counter+1;
76          end
77      end
78
79      fprintf("N of times that optimal control changes (lambda= %g) - %g\n",...
80          lambda,counter);
81      mean_sum=0;
82      for r=1:K
83          mean_sum=mean_sum + norm(E*x(:,tau(r)+1)-W(:,r));
84      end
85
86      mean_deviation=(1/K)*mean_sum;
87      fprintf('Mean Deviantion (lambda=%g) - %g\n\n', lambda, mean_deviation);
88  end;
```

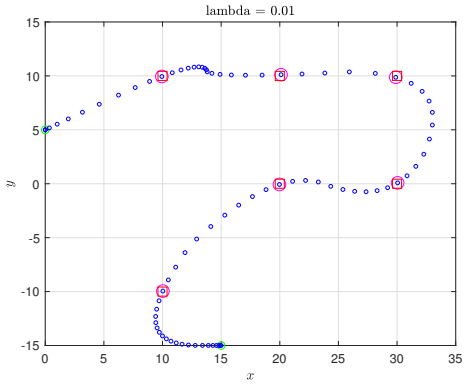


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10^{-3}$ with ℓ_1 regularizer.

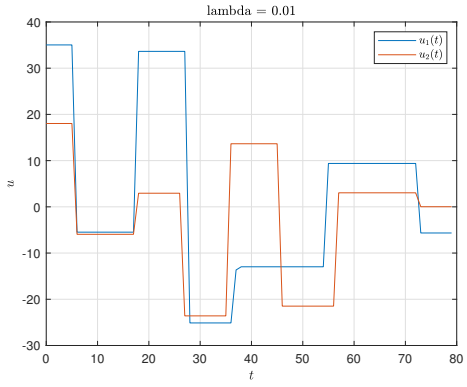


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-3}$ with ℓ_1 regularizer.

Figure 15: Case $\lambda = 10^{-3}$. Position of the robot (a) and the control signal (b).

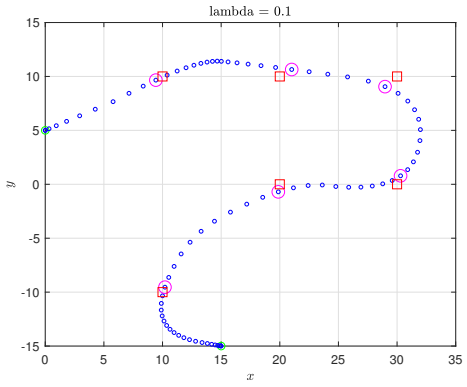


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10^{-2}$ with ℓ_1 regularizer.

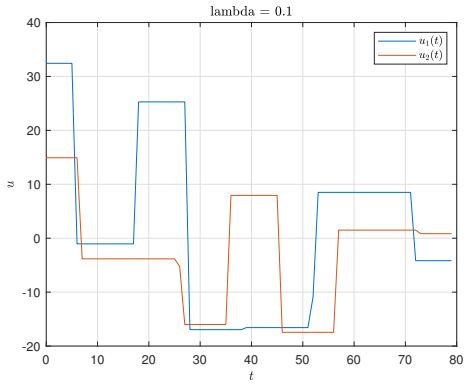


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-2}$ with ℓ_1 regularizer

Figure 16: Case $\lambda = 10^{-2}$. Position of the robot (a) and the control signal (b).

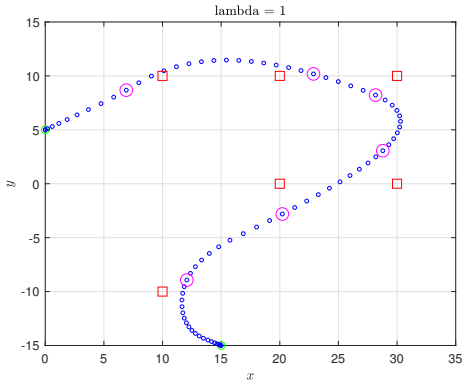


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10^{-1}$ with ℓ_1 regularizer.

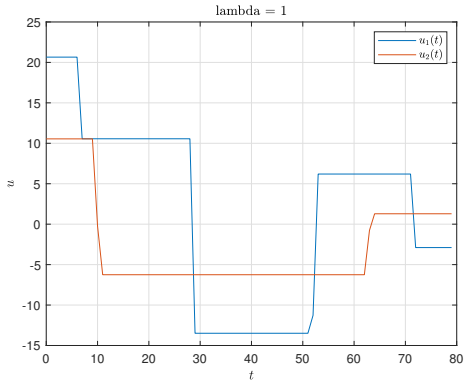


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-1}$ with ℓ_1 regularizer

Figure 17: Case $\lambda = 10^{-1}$. Position of the robot (a) and the control signal (b).

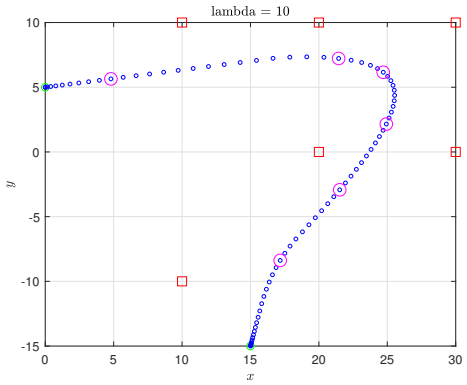


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 1$ with ℓ_1 regularizer.

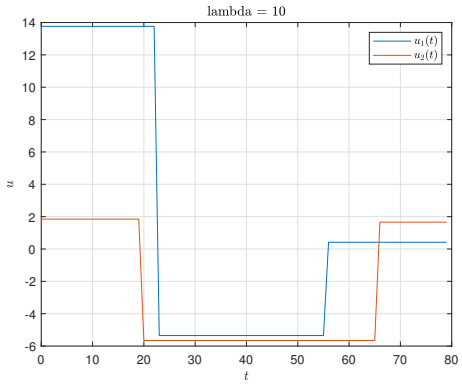


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 1$ with ℓ_1 regularizer

Figure 18: Case $\lambda = 1$. Position of the robot (a) and the control signal (b).

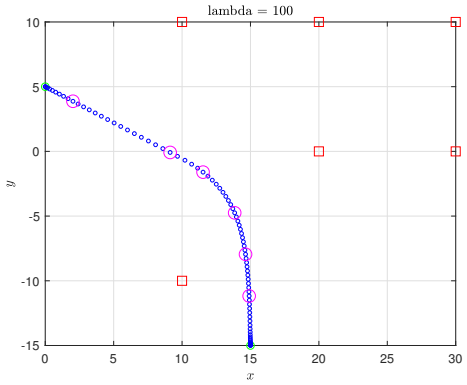


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 10$ with ℓ_1 regularizer.

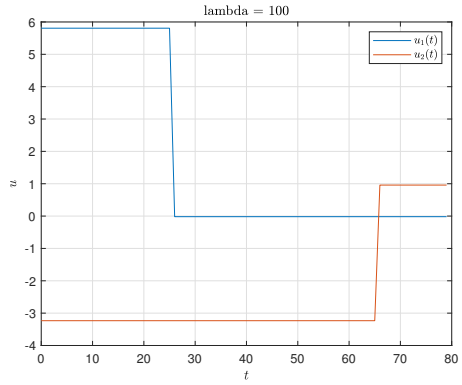


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10$ with ℓ_1 regularizer

Figure 19: Case $\lambda = 10$. Position of the robot (a) and the control signal (b).

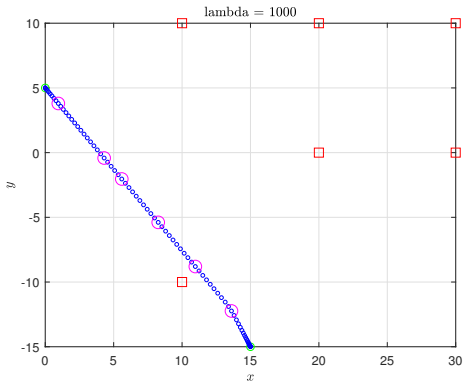


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 100$ with ℓ_1 regularizer.

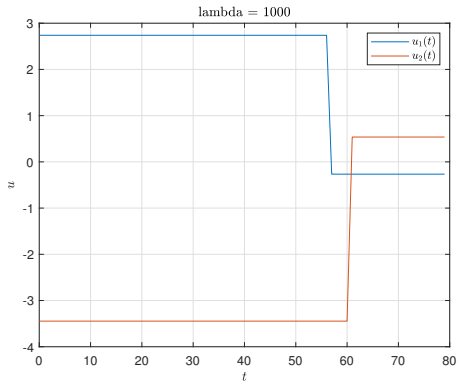


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 100$ with ℓ_1 regularizer

Figure 20: Case $\lambda = 100$. Position of the robot (a) and the control signal (b).



(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares. Case $\lambda = 1000$ with ℓ_1 regularizer.



(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T - 1$. Case $\lambda = 1000$ with ℓ_1 regularizer

Figure 21: Case $\lambda = 1000$. Position of the robot (a) and the control signal (b).

λ	Control Changes	Mean Deviation
10^{-3}	13	0.0107
10^{-2}	12	0.1055
10^{-1}	14	0.8863
1	11	2.8732
10	5	5.4362
100	3	13.0273
1000	2	16.0463

Table 3: Number of times that the optimal control signal changes and the mean deviation from the waypoints for all the values of λ

4 P1 - Task 4

- The impact of the regularisation parameter λ:

Within each task the value of λ is increased from 10^{-3} to 10^{3}. Increasing λ gives more importance to the fourth wish, having a simple control signal which translates to less consumption of energy by the motors, at the expense of playing down the third wish, passing as close as possible to the *waypoints*.

Therefore, giving more importance to the fourth wish by increasing the value of λ results in higher values of mean deviation, as a consequence of a navigation less interested on passing close to the *waypoints*. The opposite happens for smaller values of λ, the values of mean deviation decrease since the robot passes closer to the *waypoints*, the problem considers more the third wish.

In addition, increasing the value of λ prioritises the minimisation of the control signal deviations for consecutive instants of time, making the control signals smoother, as we can observe in the sequence of figures of the control signals within each task.

- The impact of using ℓ_2^2, ℓ_2 or ℓ_1 *regulariser*:

The ℓ_2^2 *regulariser* squares the norm of the difference between the control signals in consecutive instants of time, which means that the penalisation of the deviations of the control signals is stronger for higher deviations (to the power of 2), than it is for smaller deviations. For Task 1, the number of control signal changes was always 79, since the *regulariser* has a quadratic behaviour, for none of the deviations it is possible to minimise them to a piecewise constant signal, the ideal result, due to the penalisation focused more on the higher deviations.

On the other hand, using the ℓ_2 or ℓ_1 *regularisers* leads to better results in simplifying the control signal, considering that for both cases, piecewise constant signals were obtained. As a consequence of both of these cases not squaring the norm of the deviations, which means that either small deviations and high deviations will be considered in the minimisation problem.

Between the use of the ℓ_2 and the ℓ_1 *regularisers*, the best choice for this problem would be the one that considers more equally the minimisation of the norm of higher deviations relatively to smaller deviations. As corroborated by the results, the ℓ_2 *regulariser* yields better results, since a smaller number of control signal changes is obtained, because it considers more equally either the norm of higher deviations as well as of smaller deviations, considering the square root operation with the ℓ_2 *regulariser* versus the sum of absolute values with the ℓ_1 *regulariser*.

$$deviation = d = u(t) - u(t-1) \tag{1}$$

$$\|d\|_2^2 = \sum_{k=1}^{K} d_k^2 \tag{2}$$

$$\|d\|_2 = \sqrt{\sum_{k=1}^{K} d_k^2} \tag{3}$$

$$\|d\|_1 = \sum_{k=1}^{K} |d_k| \tag{4}$$

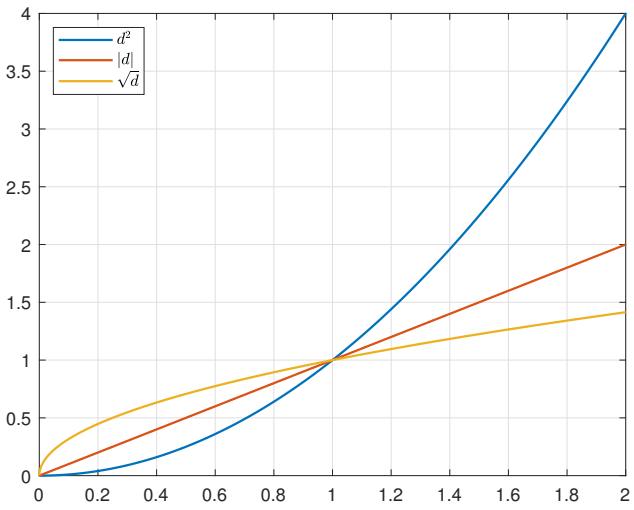


Figure 22: Graphical view of the three major operations for each type of norm.

As it can be observed in figure 22, the ℓ_2 norm is the one that normalises big deviations to smaller ones and small deviations (less than 1) to bigger ones (considering also that the square root is a strictly monotonically increasing function), and thus attributing a more equal penalisation which translates into a piecewise constant control signal. The ℓ_2^2 norm actually diminishes even more the impact of deviations smaller than 1 in the minimisation problem, and raises the impact considerably (to the power of 2) of bigger deviations.

5 P1 - Task 5

In this problem, it is asked to give a closed form expression for $d(p, D(c,r))$, where $d(p, D(c,r))$ is the distance of the point $p \in \mathbf{R}^2$ to the disk $D(c,r)$:

$$d(p, D(c,r)) = \min \{\|p - y\|_2 \, : \, y \in D(c,r)\}. \tag{5}$$

A disk, denoted $D(c,r)$, is a set of the form $\{x \in \mathbf{R}^2 \, : \, \|x - c\|_2 \leq r\}$, where $c \in \mathbf{R}^2$ is the center of the disk and r its radius.

The norm of the difference between a point $p \in \mathbf{R}^2$ and the center of the disk, is equal to the radius of the disk:

$$\|p - c\|_2 \, = r \tag{6}$$

If the point p is outside the disk, the closest point of the disk will be in the boundary and this distance will be given by :

$$d(p, D(c,r)) = \|p - c\|_2 \, - r \tag{7}$$

In figure 23 it is possible to see more intuitively what was described.

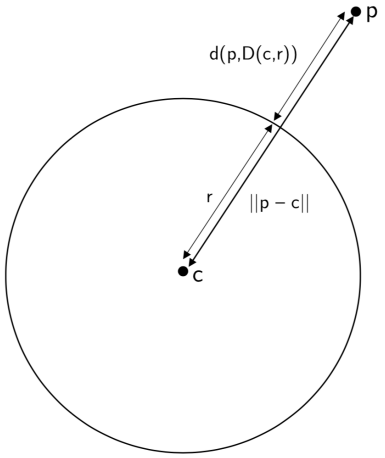


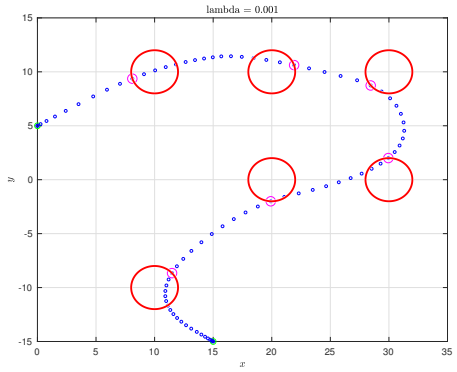
Figure 23: Graphical representation of the solution.

6 P1 - Task 6

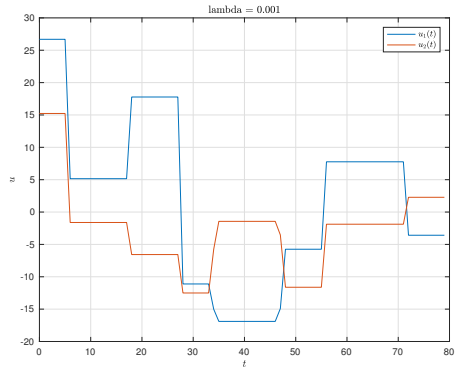
Listing 4: Task 6 Code

```
%% Task 6
close all; clear all; clc;

load Initial_Constants;
C=W; %center of the disks
R=2; % radius of the disks

lambdas=[0.001 0.01 0.1 1 10 100 1000];

for l=1:length(lambdas)

    lambda=lambdas(l);
    cvx_begin quiet
    variable x(4,T+1);
    variable u(2,T+1);

    f1=0;
    for i=1:K
        f1=f1+square_pos(norm(E*x(:,tau(i)+1)- C(:,i))-R);
    end

    f2=0;
    for j=2:T
        f2=f2 + norm(u(:,j)-u(:,j-1));
    end

    f= f1+ lambda*f2;

    minimize(f);

    subject to
    x(:,1)==[pinitial';0;0];
    x(:,T+1)==[pfinal';0;0];

    for a=1:T
        norm(u(:,a)) <= Umax;
        x(:,a+1)==A*x(:,a)+ B*u(:,a);
    end



    cvx_end;
    figure();

    plot(pinitial(1),pinitial(2),'go',pfinal(1), pfinal(2),'go');
    hold on;
```

```
47      plot(x(1,:), x(2,:),'bo','MarkerSize',3);
48      hold on;
49
50      for q=1:K
51          plot(x(1,tau(q)+1), x(2,tau(q)+1),'mo','MarkerSize',10);
52          hold on;
53      end
54
55      hold on;
56      vR = ones(6,1)*R;
57      viscircles(C',vR);
58
59      grid on;
60      title( sprintf('lambda = %g', lambda),'interpreter','latex');
61      xlabel('$x$','interpreter','latex');
62      ylabel('$y$','interpreter','latex');
63
64      time=linspace(0,T-1,T);
65
66      figure();
67
68      plot(time,u(1,1:end-1),time,u(2,1:end-1));
69      grid on;
70      title(sprintf('lambda = %g', lambda),'interpreter','latex');
71      xlabel('$t$','interpreter','latex');
72      ylabel('$u$','interpreter','latex');
73      le=legend('$u_{1}(t)$', '$u_{2}(t)$');
74      set(le,'interpreter','latex');
75
76      counter=0;
77
78      for t=2:T
79          if norm(u(:,t)-u(:,t-1)) > 10^-6
80              counter=counter+1;
81          end
82      end
83
84      fprintf("N of times that optimal control changes (lambda= %g) - %g\n",...
85          lambda,counter);
86
87      mean_sum=0;
88      for r=1:K
89          mean_sum=mean_sum + norm(E*x(:,tau(r)+1)-W(:,r));
90      end
91
92      mean_deviation=(1/K)*mean_sum;
93      fprintf('Mean Deviantion (lambda=%g) - %g\n\n', lambda, mean_deviation);
94  end;
```

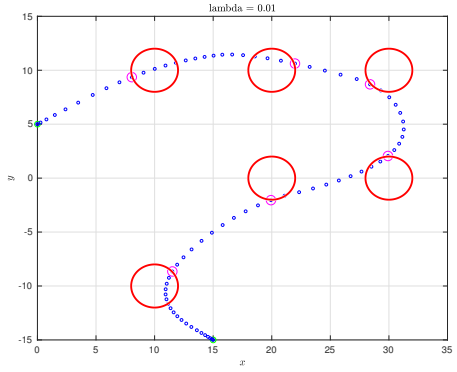


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The approximate intermediate regions are the red circles. Case $\lambda = 10^{-3}$ with ℓ_2 regularizer.

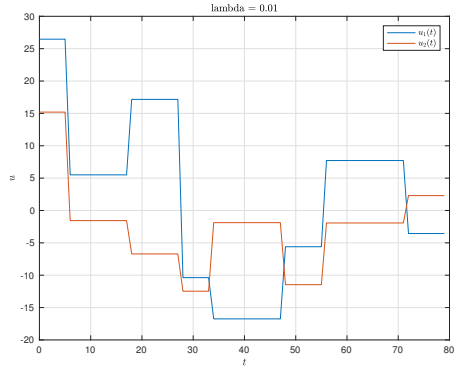


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-3}$ with ℓ_2 regularizer

Figure 24: Case $\lambda = 10^{-3}$. Position of the robot (a) and the control signal (b).

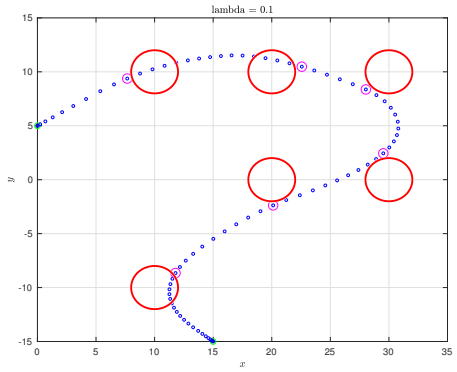


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The approximate intermediate regions are the red circles. Case $\lambda = 10^{-2}$ with ℓ_2 regularizer.

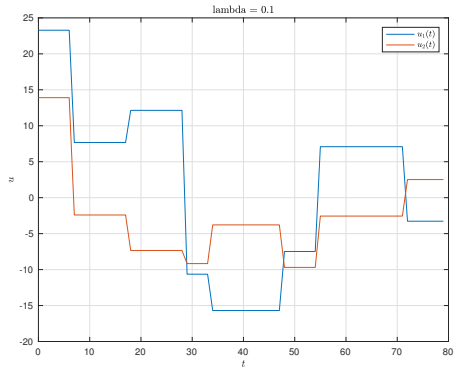


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-2}$ with ℓ_2 regularizer

Figure 25: Case $\lambda = 10^{-2}$. Position of the robot (a) and the control signal (b).

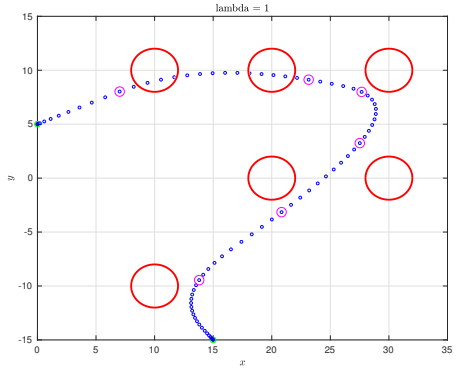


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The approximate intermediate regions are the red circles. Case $\lambda = 10^{-1}$ with ℓ_2 regularizer.

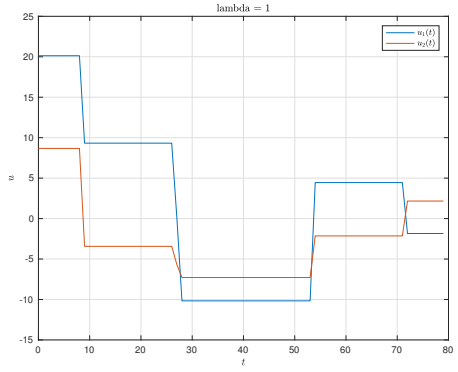


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10^{-1}$ with ℓ_2 regularizer

Figure 26: Case $\lambda = 10^{-1}$. Position of the robot (a) and the control signal (b).

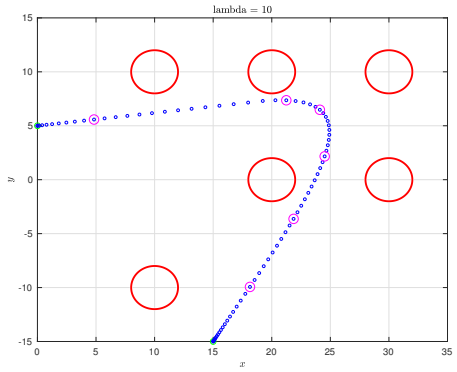


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The approximate intermediate regions are the red circles. Case $\lambda = 1$ with ℓ_2 regularizer.

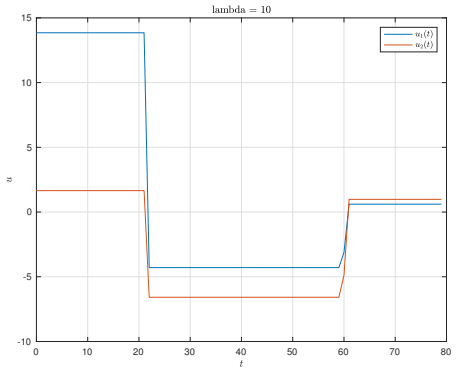


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 1$ with ℓ_2 regularizer

Figure 27: Case $\lambda = 1$. Position of the robot (a) and the control signal (b).

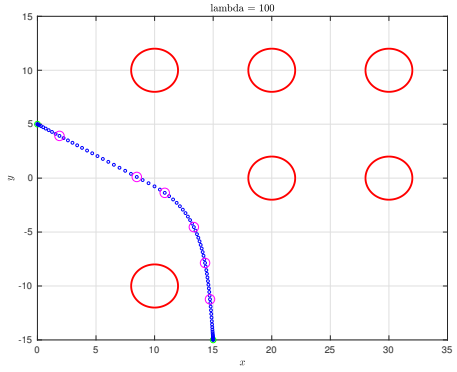


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The approximate intermediate regions are the red circles. Case $\lambda = 10$ with ℓ_2 regularizer.

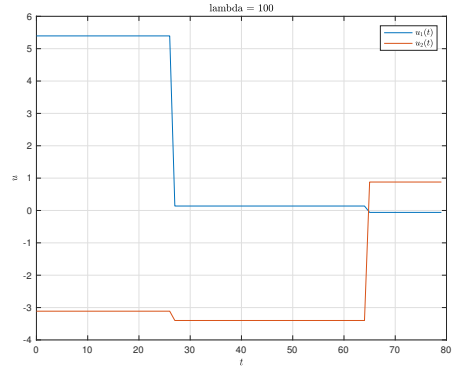


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 10$ with ℓ_2 regularizer

Figure 28: Case $\lambda = 10$. Position of the robot (a) and the control signal (b).

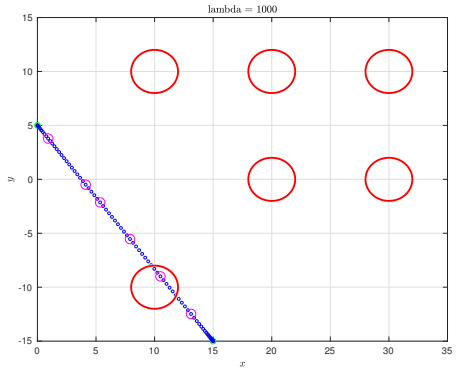


(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The approximate intermediate regions are the red circles. Case $\lambda = 100$ with ℓ_2 regularizer.

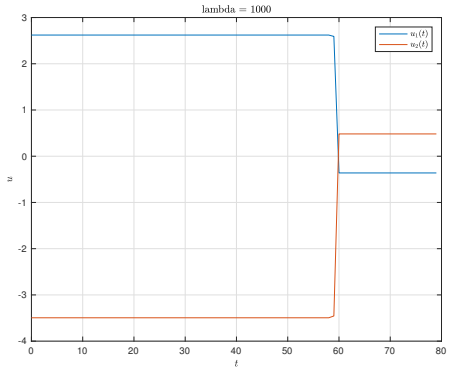


(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T-1$. Case $\lambda = 100$ with ℓ_2 regularizer

Figure 29: Case $\lambda = 100$. Position of the robot (a) and the control signal (b).



(a) Positions of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The approximate intermediate regions are the red circles. Case $\lambda = 1000$ with ℓ_2 regularizer.



(b) The two components, $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T - 1$. Case $\lambda = 1000$ with ℓ_2 regularizer

Figure 30: Case $\lambda = 1000$. Position of the robot (a) and the control signal (b).

λ	Control Changes	Mean Deviation
10^{-3}	13	2.0055
10^{-2}	11	2.05414
10^{-1}	7	2.44833
1	5	3.51969
10	5	5.7842
100	2	13.2397
1000	2	16.227

Table 4: Number of times that the optimal control signal changes and the mean deviation from the waypoints for all the values of λ

Comments:

In Task 2 we wanted do minimise the distances of the robot to the waypoints, which can be seen as disks with radius zero.

In task 6 we wanted to minimise the distances of the robot to disks where the disks are centred in the waypoints.

Comparing the positions of the robots from $t = 0$ to $t = T$, for low values of λ, it is possible to see that the trajectory of the robot in Task 6 is smoother than the trajectory in Task 1. This can be explained because we only want to minimise to an approximate region instead of a specific point, that means that the robot needs to make less variations to reach the goal. On the other hand this will increase the mean deviation, because if the robot is outside the disk the nearest point to the disk will be in the boundary, instead of the waypoint. This results in a higher mean deviation for Task 6 since the mean deviation is defined by

$$\frac{1}{K}\sum_{k=1}^{K}\left\|Ex(\tau_k) - w_k\right\|_2, \tag{8}$$

where the distance is defined between the robot and the waypoint, and not between the robot and the nearest points of the disk (boundary if the robot is outside the disk).

Looking for the control signal $u(t)$ from $t = 0$ to $t = T$, for the same reasons as before, it is possible to conclude that the control signal in task 6 has less changes than in task 2 for lower values of λ.

For higher values of λ, the third wish is ignored for both tasks, meaning that the robot will give more importance to decrease the number of control changes instead of crossing the waypoints or disks. This translates into a smaller number of control changes but also in a higher mean deviation.

7 P1 - Task 7

Listing 5: Task 7 Code

```
%% Task 7
close all; clear all; clc;

load Initial_Constants;
Umax = 15;

cvx_begin
variable x(4,T+1);
variable u(2,T+1);

f= 0;
minimize(f);

subject to
x(:,1)==[pinitial';0;0];
x(:,T+1)==[pfinal';0;0];

for a=1:T
    norm(u(:,a)) ≤ Umax;
    x(:,a+1)==A*x(:,a)+ B*u(:,a);
end
for b=1:K
    E*x(:,tau(b)+1) == W(b);
end

cvx_end;
```

In this task, a *feasibility problem* was run to check if there is a point that satisfies all the constraints of the optimisation problem in question.

CVX returned that this problem, with reduced U_{max} to 15, **is infeasible**, due to the fact that a lower U_{max} restricts the region of possible solutions so much so that it doesn't exist a solution in the region defined by the constraints that solves this optimisation problem.

From a practical perspective, reducing U_{max} means that the robot has a weaker motor. As a result of the problem being infeasible, it is confirmed that this weaker motor is so limited that it can't make the robot go from its initial position to the desired final position, passing close to the defined *waypoints*, accordingly to its dynamics. In short, a better motor is required to accomplish this task.

8 P1 - Task 8

$f : \mathbf{R^n} \to \mathbf{R}$ is convex if

$$f[(1-\alpha)x + \alpha y] \leq (1-\alpha)f(x) + \alpha f(y) \tag{9}$$

for all $x, y \in \mathbf{R^n}$ and $0 \leq \alpha \leq 1$

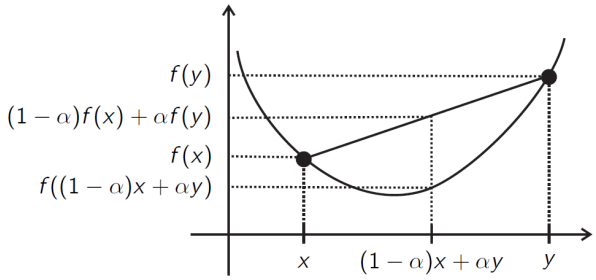


Figure 31: Graphical representation of the convexity condition (1) for two arbitrary points $x, y \in \mathbf{R^n}$.

Choosing two random points and inputting them in the convexity condition (1), until the condition is violated, can show that the function $\phi : \mathbf{R^2} \to \mathbf{R}$ is nonconvex.

To do that the following MATLAB code was used

Listing 6: Task 8 Code

```
%% Task 8
clear all; close all; clc;

% Show that the function in (11) is nonconvex
% The function is defined in F.m file

iter = 1;
logic = 1;

v = -10:0.5:10;
valfa = 0:0.01:1;

tic
while logic ~= 0
    x = [datasample(v, 1); datasample(v, 1)];
    y = [datasample(v, 1); datasample(v, 1)];
    alfa = datasample(valfa, 1);

    if x ~= y
        m1 = F((1-alfa)*x + alfa*y);
        fx = F(x);
        fy = F(y);
```

```
23          m2 = (1-alfa)*fx + alfa*fy;
24      end
25
26      logic = m1 <= m2;
27
28      if logic == 0
29          fprintf('Non-convex! Number of iterations: %g \n', iter);
30      end
31      iter = iter + 1;
32  end
33  toc
34
35  function answ = F(arg)
36
37  if arg(1) == 0 && arg(2) == 0
38      answ = 0;
39  else
40      answ = 1;
41  end
42
43  end
```

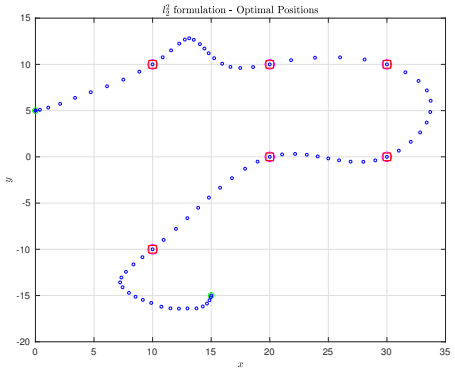
The convexity condition was violated for $x = (0, 0)$ and $y = (9, -2.5)$, where $x, y \in \mathbf{R^2}$.

9 P1 - Task 9

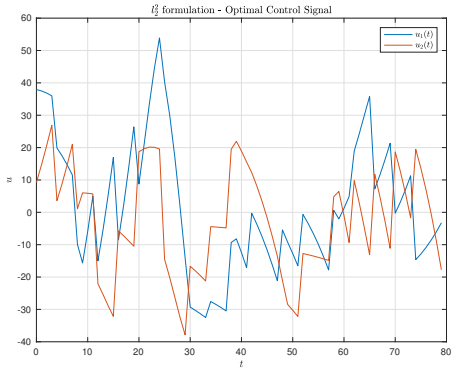
Listing 7: Task 9 Code

```
%% Task 9
close all; clear all; clc;

load Initial_Constants;

cvx_begin quiet
variable x(4,T+1);
variable u(2,T+1);

f1=0;
for i=1:K
    f1=f1+square_pos(norm(E*x(:,tau(i)+1)-W(:,i)));
end

f= f1;
minimize(f);

subject to
x(:,1)==[pinitial';0;0];
x(:,T+1)==[pfinal';0;0];

for a=1:T
    norm(u(:,a)) <= Umax;
    x(:,a+1)==A*x(:,a)+ B*u(:,a);
end

cvx_end;

figure();
plot(pinitial(1),pinitial(2),'go',pfinal(1), pfinal(2),'go');

hold on;
plot(x(1,:), x(2,:),'bo','MarkerSize',3);
hold on;

for q=1:K
    plot(x(1,tau(q)+1), x(2,tau(q)+1),'mo','MarkerSize',10);
    hold on;
end

hold on;
plot(W(1,:),W(2,:),'rs','MarkerSize',10);
grid on;
title( '$l_{2}^{2}$ formulation - Optimal Positions','interpreter','latex');
xlabel('$x$','interpreter','latex');
ylabel('$y$','interpreter','latex');
```

```
47
48
49 time=linspace(0,T-1,T);
50 figure();
51
52 plot(time,u(1,1:end-1),time,u(2,1:end-1));
53 grid on;
54 title('$l_{2}^{2}$ formulation - Optimal Control Signal','interpreter',...
55     'latex');
56 xlabel('$t$','interpreter','latex');
57 ylabel('$u$','interpreter','latex');
58 le=legend('$u_{1}(t)$', '$u_{2}(t)$');
59 set(le,'interpreter','latex');
60
61 counter=0;
62 for t=1:K
63     if norm(E*x(:,tau(t)+1)-W(:,t)) <= 10^-6
64         counter=counter+1;
65     end
66 end
67
68 fprintf("Number of waypoints captured by the robot : %g\n",counter);
```



(a) Optimal positions, ℓ_2^2 formulation, of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares.



(b) The two optimal components with ℓ_2^2 regulizer , $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T - 1$.

Figure 32: Exact waypoints with an ℓ_2^2 formulation.

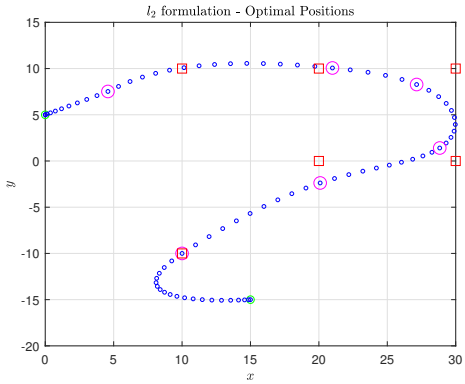
Number of waypoints captured by the robot : 6

10 P1 - Task 10

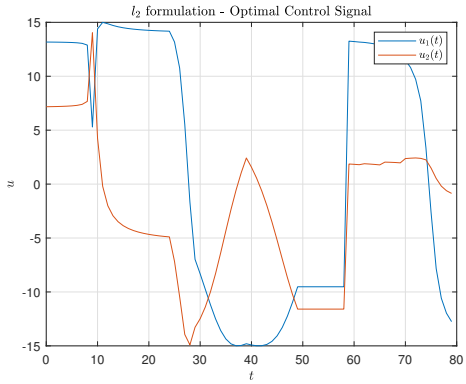
Listing 8: Task 10 Code

```
%% Task 10
close all; clear all; clc;

load Initial_Constants;

Umax=15;

cvx_begin quiet
variable x(4,T+1);
variable u(2,T+1);

f1=0;
for i=1:K
    f1=f1+norm(E*x(:,tau(i)+1)-W(:,i));
end

f= f1;
minimize(f);

subject to
x(:,1)==[pinitial';0;0];
x(:,T+1)==[pfinal';0;0];

for a=1:T
    norm(u(:,a)) <= Umax;
    x(:,a+1)==A*x(:,a)+ B*u(:,a);
end

cvx_end;

figure();
plot(pinitial(1),pinitial(2),'go',pfinal(1), pfinal(2),'go');

hold on;
plot(x(1,:), x(2,:),'bo','MarkerSize',3);
hold on;

for q=1:K
    plot(x(1,tau(q)+1), x(2,tau(q)+1),'mo','MarkerSize',10);
    hold on;
end

hold on;
plot(W(1,:),W(2,:),'rs','MarkerSize',10);
grid on;
title( '$l_{2}$ formulation - Optimal Positions','interpreter','latex');
```

```
xlabel('$x$','interpreter','latex');
ylabel('$y$','interpreter','latex');


time=linspace(0,T-1,T);
figure();

plot(time,u(1,1:end-1),time,u(2,1:end-1));
grid on;
title('$l_{2}$ formulation - Optimal Control Signal','interpreter','latex');
xlabel('$t$','interpreter','latex');
ylabel('$u$','interpreter','latex');
le=legend('$u_{1}(t)$', '$u_{2}(t)$');
set(le,'interpreter','latex');

counter=0;
for t=1:K
    if norm(E*x(:,tau(t)+1)-W(:,t)) <= 10^-6
        counter=counter+1;
    end
end

fprintf("Number of waypoints captured by the robot : %g\n",counter);
x0 = x;
u0 = u;

save('inicialization_task11', 'x0', 'u0');
```



(a) Optimal positions, ℓ_2 formulation, of the robot from $t = 0$ to $t = T$ are the small blue circles; the positions at appointed times τ_k, for $1 \leq k \leq K$, are the large magenta circles. The waypoints are the red squares.



(b) The two optimal components with ℓ_2 regulizer , $u_1(t)$ and $u_2(t)$, of the control signal $u(t)$ from $t = 0$ to $t = T - 1$.

Figure 33: Exact waypoints with an ℓ_2 formulation.

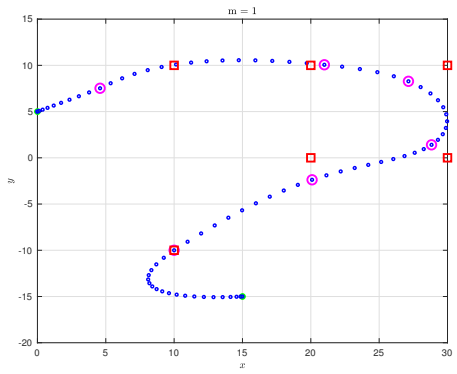
Number of waypoints captured by the robot: 1

11 P1 - Task 11

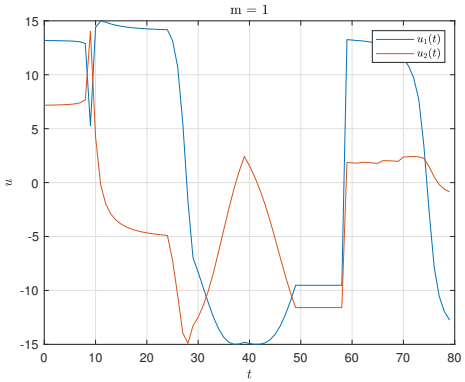
Listing 9: Task 11 Code

```
%% Task 11
clear all; close all; clc;
load Initial_Constants
load inicialization_task11


Umax = 15;
e = 10^-6;
M = 10;

xm = x0;
um=u0;

for l = 1:M

    cvx_begin quiet
    variable x(4,T+1);
    variable u(2,T+1);


    f1 = 0;
    for i = 1:K
        weight(l,i) = 1/(norm(E*xm(:,tau(i)+1)-W(:,i))+e);
        f1 = f1 + weight(l,i)*norm(E*x(:,tau(i)+1)-W(:,i));
    end

    f = f1;
    minimize(f);

    subject to
    x(:,1) == [pinitial'; 0; 0];
    x(:,T+1) == [pfinal'; 0; 0];
    for a = 1:T
        norm(u(:,a)) <= Umax;
        x(:,a+1) == A*x(:,a)+B*u(:,a);
    end
    cvx_end;

    figure()
    plot(pinitial(1), pinitial(2), 'go', 'MarkerFaceColor', 'g');
    hold on;
    plot(pfinal(1), pfinal(2), 'go', 'MarkerFaceColor', 'g'); hold on;
    for z = 1:K
        plot(xm(1, tau(z)+1), xm(2, tau(z)+1), 'mo', 'MarkerSize', 10, ...
            'LineWidth', 2); hold on;
    end
```

```
47      plot(xm(1,:), xm(2,:), 'bo', 'MarkerSize', 3, 'LineWidth', 1.5);
48      hold on;
49      plot(W(1,:), W(2,:), 'rs', 'MarkerSize', 10, 'LineWidth', 2);
50      xlabel('$x$', 'interpreter','latex');
51      ylabel('$y$','interpreter','latex');
52      title(sprintf('m = %d', l),'interpreter','latex');
53      grid on;
54
55      time = 0:1:T-1;
56
57      figure()
58      plot(time, um(1, 1:end-1)); hold on;
59      plot(time, um(2, 1:end-1));
60      xlabel('$t$','interpreter','latex');
61      ylabel('$u$','interpreter','latex');
62      le=legend('$u_{1}(t)$', '$u_{2}(t)$');
63      set(le,'interpreter','latex');
64
65      title(sprintf('m = %d', l), 'interpreter', 'latex');
66      grid on;
67
68      cw = 0;
69      for r = 1:K
70          test = norm(E*xm(:,tau(r)+1)-W(:,r));
71          if test <= 10^-6
72              cw = cw + 1;
73          end
74      end
75
76      fprintf('Number of Captured Waypoints: %g\n\n', cw);
77
78      xm = x;
79      um=u;
80  end
```

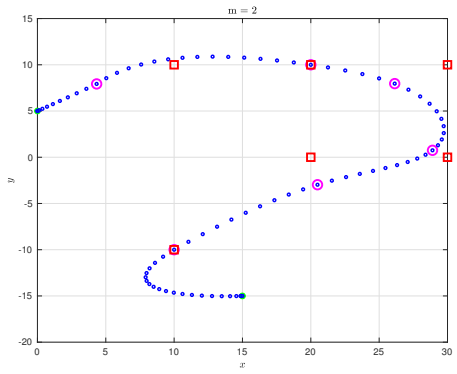


(a) Positions of the robot in $x^{(1)}$ from $t = 0$ to $t = T$.

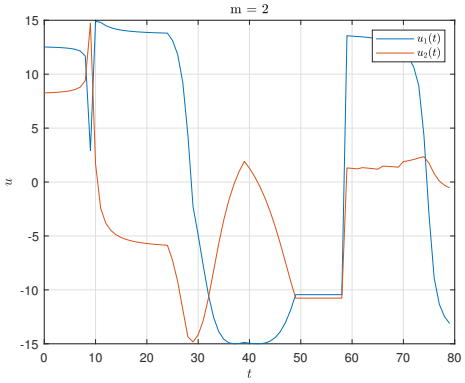


(b) Control signal $u^{(1)}(t) = (u_1^{(1)}(t), u_2^{(1)}(t))$, from $t = 0$ to $t = T - 1$.

Figure 34: Waypoints with the iterative reweighting technique ($m = 1$).

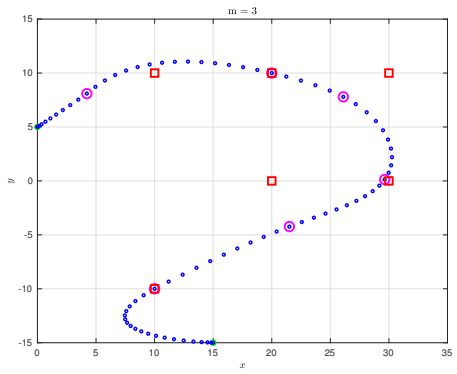


(a) Positions of the robot in $x^{(2)}$ from $t = 0$ to $t = T$.

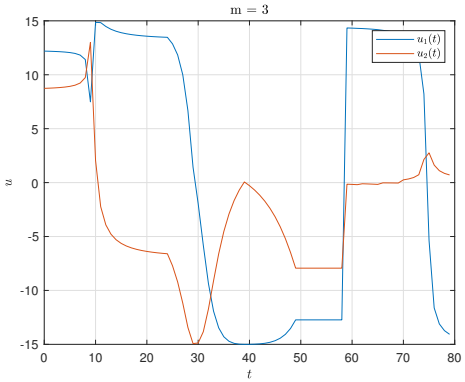


(b) Control signal $u^{(2)}(t) = (u_1^{(2)}(t), u_2^{(2)}(t))$, from $t = 0$ to $t = T - 1$.

Figure 35: Waypoints with the iterative reweighting technique ($m = 2$).

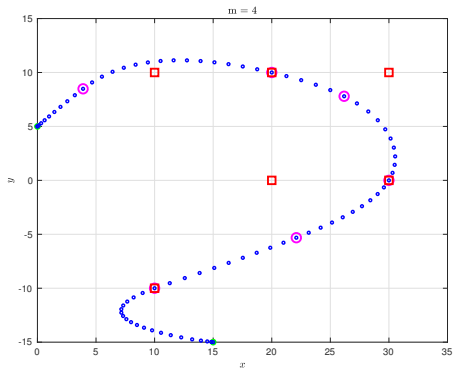


(a) Positions of the robot in $x^{(3)}$ from $t = 0$ to $t = T$.

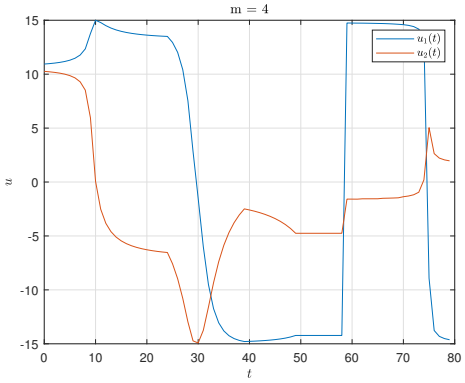


(b) Control signal $u^{(3)}(t) = (u_1^{(3)}(t), u_2^{(3)}(t))$, from $t = 0$ to $t = T - 1$.

Figure 36: Waypoints with the iterative reweighting technique ($m = 3$).

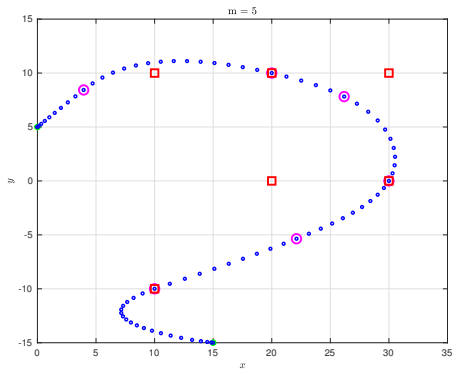


(a) Positions of the robot in $x^{(4)}$ from $t = 0$ to $t = T$.

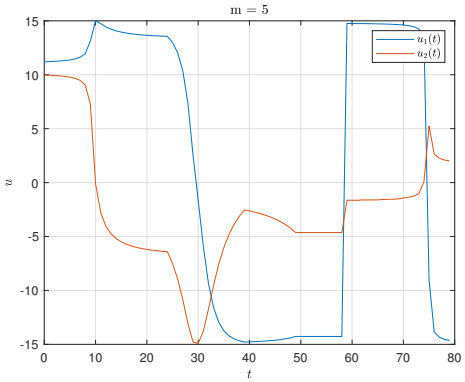


(b) Control signal $u^{(4)}(t) = (u_1^{(4)}(t), u_2^{(4)}(t))$, from $t = 0$ to $t = T - 1$.

Figure 37: Waypoints with the iterative reweighting technique ($m = 4$).

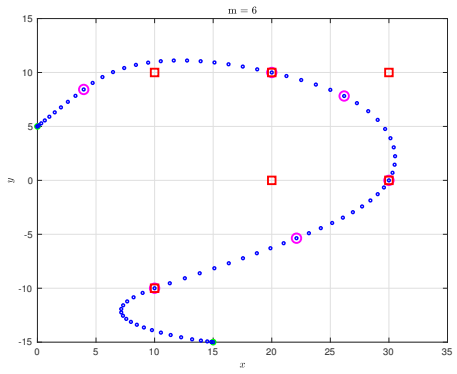


(a) Positions of the robot in $x^{(5)}$ from $t = 0$ to $t = T$.

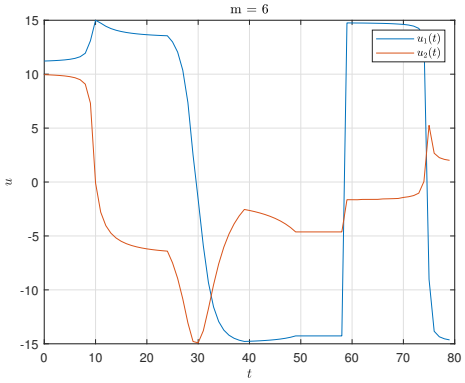


(b) Control signal $u^{(5)}(t) = (u_1^{(5)}(t), u_2^{(5)}(t))$, from $t = 0$ to $t = T - 1$.

Figure 38: Waypoints with the iterative reweighting technique ($m = 5$).

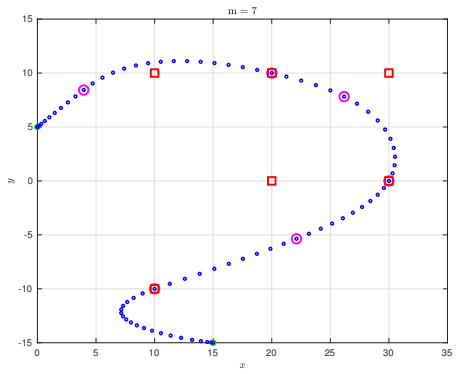


(a) Positions of the robot in $x^{(6)}$ from $t = 0$ to $t = T$.

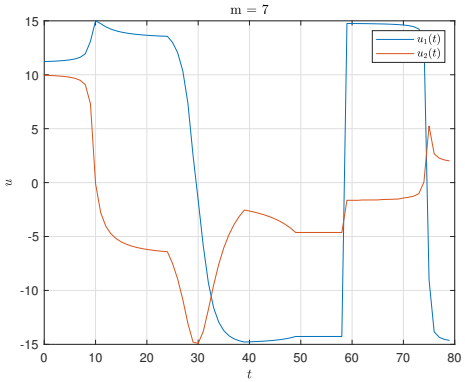


(b) Control signal $u^{(6)}(t) = (u_1^{(6)}(t), u_2^{(6)}(t))$, from $t = 0$ to $t = T - 1$.

Figure 39: Waypoints with the iterative reweighting technique ($m = 6$).

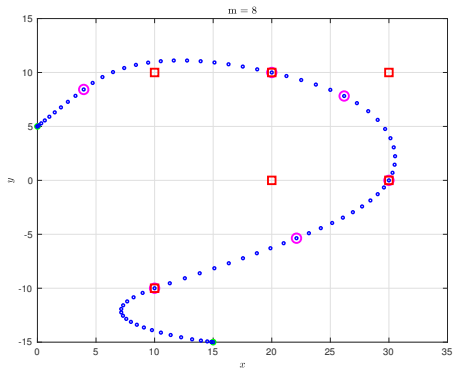


(a) Positions of the robot in $x^{(7)}$ from $t = 0$ to $t = T$.

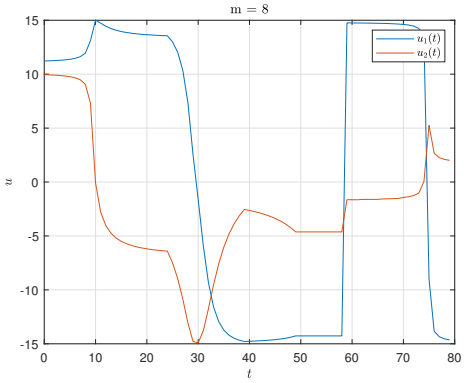


(b) Control signal $u^{(7)}(t) = (u_1^{(7)}(t), u_2^{(7)}(t))$, from $t = 0$ to $t = T - 1$.

Figure 40: Waypoints with the iterative reweighting technique ($m = 7$).

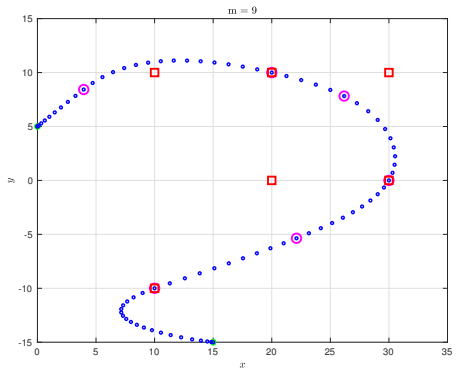


(a) Positions of the robot in $x^{(8)}$ from $t = 0$ to $t = T$.

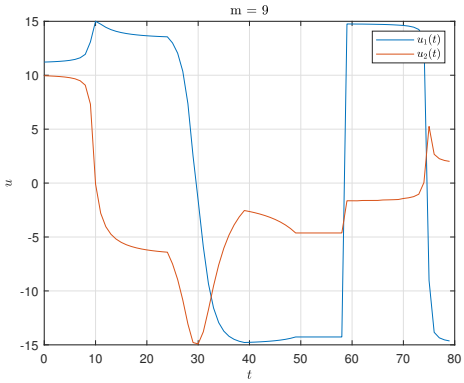


(b) Control signal $u^{(8)}(t) = (u_1^{(8)}(t), u_2^{(8)}(t))$, from $t = 0$ to $t = T - 1$.

Figure 41: Waypoints with the iterative reweighting technique ($m = 8$).

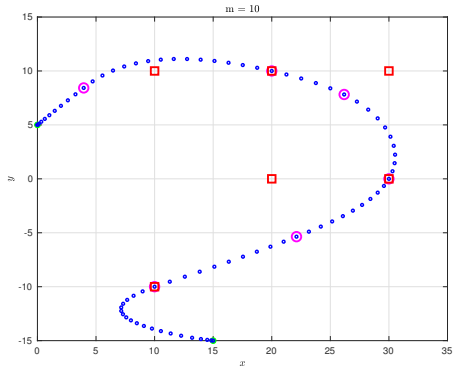


(a) Positions of the robot in $x^{(9)}$ from $t = 0$ to $t = T$.

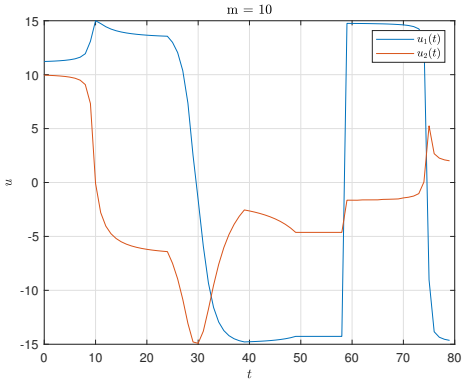


(b) Control signal $u^{(9)}(t) = (u_1^{(9)}(t), u_2^{(9)}(t))$, from $t = 0$ to $t = T - 1$.

Figure 42: Waypoints with the iterative reweighting technique ($m = 9$).



(a) Positions of the robot in $x^{(10)}$ from $t = 0$ to $t = T$.



(b) Control signal $u^{(10)}(t) = (u_1^{(10)}(t), u_2^{(10)}(t))$, from $t = 0$ to $t = T - 1$.

Figure 43: Waypoints with the iterative reweighting technique ($m = 10$).

m	Number of Waypoints Captured
1	1
2	2
3	2
4	3
5	3
6	3
7	3
8	3
8	3
10	3

Table 5: Number of waypoints captured by the robots for all the values of m (umber of times that the reweighting technique is applied).

12 P1 - Task 12

The weight is given by $\frac{1}{\left\|Ex^{(m)}(\tau_k)-w_k\right\|_2+\epsilon}$.

The denominator corresponds to the distance between the position of the robot at times τ_k and the waypoints ($\left\|Ex^{(m)}(\tau_k) - w_k\right\|_2 + \epsilon$), plus a small constant.

That being said, when the distance between the robot and the a waypoint, plus the small ϵ, is less than 1, the weight grows proportionally inverse to the distance.

As a result, whenever the robot passes close to a waypoint this formulation will try to capture the robot position and bring it closer to the waypoint for the instants of time τ_k - considering the minimisation problem, which will now penalise more the distance of the robot to the waypoint whenever it's already close to it (smaller than 1).

Whenever the robot is considerably far from the waypoint the weight will be small and the attraction of the robot position at time τ_k is diminished.

13 P2 - Task 1

$$\underset{(s,r)\in\mathbf{R}^n\times\mathbf{R}}{\text{minimize}} \quad \underbrace{\frac{1}{K}\sum_{k=1}^{K}\left(\log\left(1+\exp(s^T x_k - r)\right) - y_k\left(s^T x_k - r\right)\right)}_{f(s,r)}. \tag{10}$$

In order to prove the convexity of the objective function f, the function can be divided in the sum of two functions, $f_{1k}(s,r)$ and $f_{2k}(s,r)$:

$$f(s,r) = \frac{1}{K}\sum_{k=1}^{K}(f_{1k}(s,r) + f_{2k}(s,r)) \tag{11}$$

where:

$$f_{1k}(s,r) = \log(1+\mathrm{e}^{s^T x_k - r}) \tag{12}$$

and:

$$f_{2k}(s,r) = -y_k(s^T x_k - r) \tag{13}$$

We can define f_{1k} in a composition:

$$f_{1k}(s,r) = h \circ g \tag{14}$$

The function $h(x)$ can be defined as:

$$h(x) = \log(1+\mathrm{e}^x) \tag{15}$$

From the *simple convex functions list*, $h(x) : \mathbf{R} \to \mathbf{R}$, logistic function, is convex and non decreasing.

The function $g(s,r)$ can be defined as:

$$g(s,r) = s^T x_k - r \tag{16}$$

$$s^T x_k - r = \begin{bmatrix} x_k & -1 \end{bmatrix} \begin{bmatrix} s \\ r \end{bmatrix} \tag{17}$$

With the equation 17 is clear that the function $g(s,r) : \mathbf{R}^n \to \mathbf{R}$ is affine and convex.

Using the property that states that an affine map ($g(s,r)$) followed by a convex function ($h(x)$) is an operation that preserves convexity, it is proved that $f_{1k}(s,r) : \mathbf{R}^n \to \mathbf{R}$ is convex.

For the function f_{2k}:

$$y_k(s^T x_k - r) = y_k \begin{bmatrix} x_k & -1 \end{bmatrix} \begin{bmatrix} s \\ r \end{bmatrix} \tag{18}$$

Looking at equation 18, where y_k is a constant, it is visible that the function $f_{2k}(s,r)$ is affine, simple convex function, where $f_{2k}(s,r) : \mathbf{R}^n \to \mathbf{R}$

Since the functions $f_{1k}(s,r)$ and $f_{2k}(s,r)$ are convex, knowing that the sum of convex functions is an operation that preserves convexity and $\frac{1}{K}$ is a constant, the result of the sum, $f(s,r) : \mathbf{R}^n \to \mathbf{R}$, is also a convex function.

14 P2 - Task 2

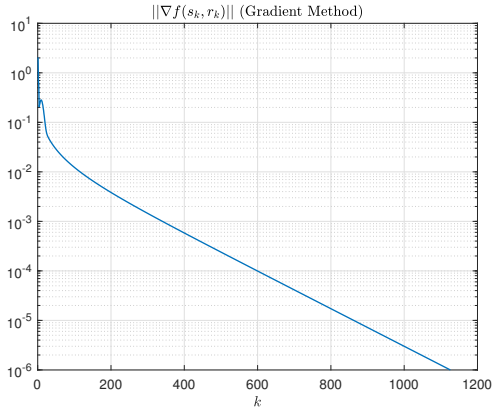
Listing 10: Part 2 - Task 2 Code

```
%% Task 2
clear all; close all; clc;
load data1.mat

K = length(Y);
[l, c] = size(X);

a = 1;
gama = 10^-4;
beta = 0.5;
eps = 10^-6;

k = 1;
criteria = 0;

v_ones = -1*ones(1, length(X));

A = (1/K)*[X; v_ones];

x(:,1) = [-ones(1,l)'; 0];

tic
while criteria == 0
    g(:,k) = sum(A.*((exp(x(:,k)'*[X;v_ones]))./(1 + ...
                                        (exp(x(:,k)'*[X;v_ones])))  - Y), 2);

    gnorm(k) = norm(g(:,k));
    if gnorm(k) < eps
        criteria = 1;
        break;
    end

    d(:,k) = -g(:,k);

    c = 0;
    a = 1;
    while c == 0

        b = x(:,k) + a*d(:,k);
        member1 = (1/K)*sum(log(1+exp(b'*[X;v_ones]))-Y.*(b'*[X;v_ones]));
        z = x(:,k);
        member2 = (1/K)*sum(log(1+exp(z'*[X;v_ones]))-Y.*(z'*[X;v_ones]))...
                                                    + gama*g(:,k)'*a*d(:,k);

        if member1 < member2
            c = 1;
```

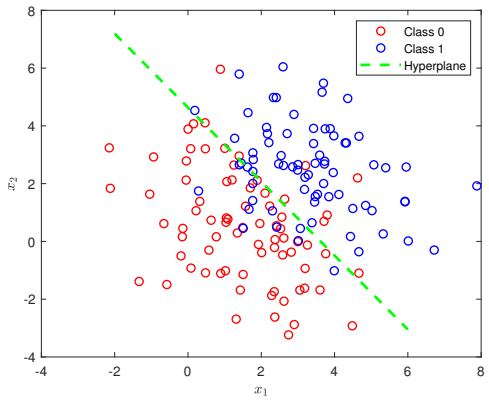
```
47              break;
48          end
49          a = beta*a;
50      end
51
52      x(:,k+1) = x(:,k) + a*d(:,k);
53      k = k + 1;
54  end
55  toc
56  s = [x(1, end); x(2, end)]
57  r = x(3, end)
58
59  figure()
60  semilogy(gnorm, 'linewidth', 1);
61  xlabel('$k$', 'interpreter','latex');
62  title('$|| \nabla f(s_{k},r_{k}) ||$ (Gradient Method)',...
63                                                  'interpreter','latex')
64  grid on;
65
66  x1 = X(1,:)';
67  x2 = X(2,:)';
68  class = Y';
69
70  tx = -2:0.001:6;
71
72  figure()
73  plot(x1(class==0), x2(class==0), 'ro', 'linewidth', 1); hold on;
74  plot(x1(class==1), x2(class==1), 'bo', 'linewidth', 1); hold on;
75  plot(tx, ((-s(1)*tx)+r)/s(2), 'g--', 'linewidth', 2);
76  xlabel('$x_{1}$', 'Interpreter', 'latex');
77  ylabel('$x_{2}$', 'Interpreter', 'latex');
78  legend('Class 0', 'Class 1', 'Hyperplane', 'location', 'northeast');
```

Result `data1.mat`:

- $s = (1.3495, 1.0540)$
- $r = 4.8815$



(a) Norm of the gradient along the iterations of the gradient method.



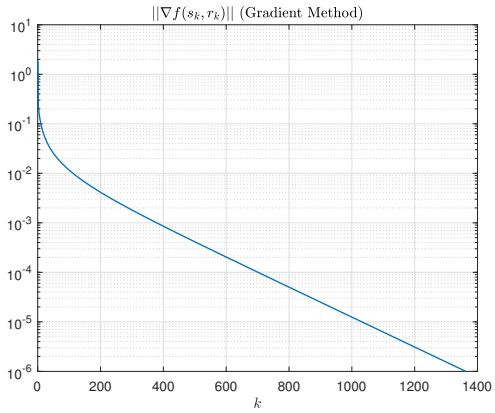
(b) The dataset `data1.mat` with the green line $\{x \in \mathbf{R}^2 : s^T x = r\}$ superimposed, where (s, r) is the solution of problem (1).

Figure 44: Gradient method results (applied to `data1.mat`).

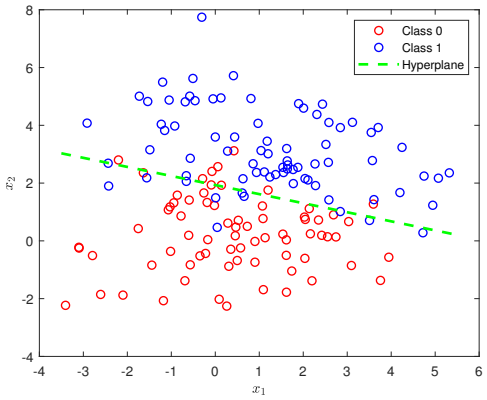
15 P2 - Task 3

Result `data2.mat`:

- $s = (0.7402, 2.3577)$
- $r = 4.5553$



(a) Norm of the gradient along the iterations of the gradient method.



(b) The dataset `data2.mat` with the green line $\{x \in \mathbf{R}^2 : s^T x = r\}$ superimposed, where (s, r) is the solution of problem (1).

Figure 45: Gradient method results (applied to `data2.mat`).

16 P2 - Task 4

Result `data3.mat`:

- $s = (-1.3082, 1.4078)$
- $r = 0.8049$

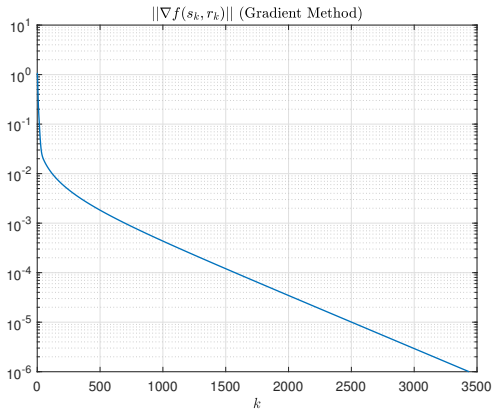


Figure 46: Norm of the gradient along the iterations of the gradient method (applied to `data3.mat`).

Result `data4.mat`:

- $s = (0.1098, -0.6423)$
- $r = 0.1019$

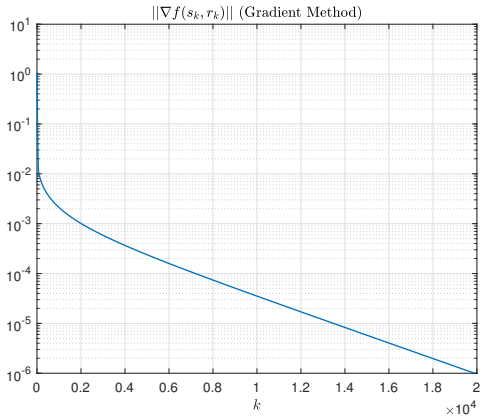


Figure 47: Norm of the gradient along the iterations of the gradient method (applied to `data4.mat`).

17 P2 - Task 5

(a) $x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ and $A = \begin{bmatrix} a_1 & a_2 & \cdots & a_K \end{bmatrix} = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{K1} \\ a_{12} & a_{22} & \cdots & a_{K2} \\ a_{13} & a_{23} & \cdots & a_{K3} \end{bmatrix}$

$$\nabla p(x) = \begin{bmatrix} \frac{\partial p}{\partial x_1} \\ \frac{\partial p}{\partial x_2} \\ \frac{\partial p}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{K} \dot{\phi}\left(a_k^T x\right) \frac{\partial}{\partial x_1}(a_k^T x) \\ \sum_{k=1}^{K} \dot{\phi}\left(a_k^T x\right) \frac{\partial}{\partial x_2}(a_k^T x) \\ \sum_{k=1}^{K} \dot{\phi}\left(a_k^T x\right) \frac{\partial}{\partial x_3}(a_k^T x) \end{bmatrix} =$$

$$= \begin{bmatrix} \dot{\phi}\left(a_1^T x\right) \frac{\partial}{\partial x_1}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) + \cdots + \dot{\phi}\left(a_K^T x\right) \frac{\partial}{\partial x_1}(a_{K1}x_1 + a_{K2}x_2 + a_{K3}x_3) \\ \dot{\phi}\left(a_1^T x\right) \frac{\partial}{\partial x_2}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) + \cdots + \dot{\phi}\left(a_K^T x\right) \frac{\partial}{\partial x_2}(a_{K1}x_1 + a_{K2}x_2 + a_{K3}x_3) \\ \dot{\phi}\left(a_1^T x\right) \frac{\partial}{\partial x_3}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) + \cdots + \dot{\phi}\left(a_K^T x\right) \frac{\partial}{\partial x_3}(a_{K1}x_1 + a_{K2}x_2 + a_{K3}x_3) \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}\dot{\phi}\left(a_1^T x\right) + a_{21}\dot{\phi}\left(a_2^T x\right) + \cdots + a_{K1}\dot{\phi}\left(a_K^T x\right) \\ a_{12}\dot{\phi}\left(a_1^T x\right) + a_{22}\dot{\phi}\left(a_2^T x\right) + \cdots + a_{K2}\dot{\phi}\left(a_K^T x\right) \\ a_{13}\dot{\phi}\left(a_1^T x\right) + a_{23}\dot{\phi}\left(a_2^T x\right) + \cdots + a_{K3}\dot{\phi}\left(a_K^T x\right) \end{bmatrix} = A \begin{bmatrix} \dot{\phi}\left(a_1^T x\right) \\ \dot{\phi}\left(a_2^T x\right) \\ \vdots \\ \dot{\phi}\left(a_K^T x\right) \end{bmatrix}$$

$$= Av \tag{19}$$

(b) By the Schwarz's theorem, H is symmetric, which means that the computation of the upper triangular half of H is enough to compute the whole Hessian.

$$\nabla^2 p(x) = H = \begin{bmatrix} \frac{\partial^2 p}{\partial x_1^2} & \frac{\partial^2 p}{\partial x_1 x_2} & \frac{\partial^2 p}{\partial x_1 x_3} \\ \frac{\partial^2 p}{\partial x_2 x_1} & \frac{\partial^2 p}{\partial x_2^2} & \frac{\partial^2 p}{\partial x_2 x_3} \\ \frac{\partial^2 p}{\partial x_3 x_1} & \frac{\partial^2 p}{\partial x_3 x_2} & \frac{\partial^2 p}{\partial x_3^2} \end{bmatrix} \tag{20}$$

Using previous knowledge and steps taken from question (a):

$$\begin{aligned}\frac{\partial^2 p}{\partial x_1^2} &= a_{11}\ddot{\phi}\left(a_1^T x\right)\frac{\partial}{\partial x_1}(a_1^T x) + a_{21}\ddot{\phi}\left(a_2^T x\right)\frac{\partial}{\partial x_1}(a_2^T x) + \cdots + a_{K1}\ddot{\phi}\left(a_K^T x\right)\frac{\partial}{\partial x_1}(a_K^T x) = \\ &= a_{11}\ddot{\phi}\left(a_1^T x\right)a_{11} + a_{21}\ddot{\phi}\left(a_2^T x\right)a_{21} + \cdots + a_{K1}\ddot{\phi}\left(a_K^T x\right)a_{K1}\end{aligned} \tag{21}$$

Analogously

$$\frac{\partial^2 p}{\partial x_2^2} = a_{12}\ddot{\phi}\left(a_1^T x\right)a_{12} + a_{22}\ddot{\phi}\left(a_2^T x\right)a_{22} + \cdots + a_{K2}\ddot{\phi}\left(a_K^T x\right)a_{K2} \tag{22}$$

$$\frac{\partial^2 p}{\partial x_3^2} = a_{13}\ddot{\phi}\left(a_1^T x\right)a_{13} + a_{23}\ddot{\phi}\left(a_2^T x\right)a_{23} + \cdots + a_{K3}\ddot{\phi}\left(a_K^T x\right)a_{K3} \tag{23}$$

And

$$\frac{\partial^2 p}{\partial x_1 x_2} = a_{11}\ddot{\phi}\left(a_1^T x\right)a_{12} + a_{21}\ddot{\phi}\left(a_2^T x\right)a_{22} + \cdots + a_{K1}\ddot{\phi}\left(a_K^T x\right)a_{K2} \tag{24}$$

$$\frac{\partial^2 p}{\partial x_1 x_3} = a_{11}\ddot{\phi}\left(a_1^T x\right)a_{13} + a_{21}\ddot{\phi}\left(a_2^T x\right)a_{23} + \cdots + a_{K1}\ddot{\phi}\left(a_K^T x\right)a_{K3} \tag{25}$$

$$\frac{\partial^2 p}{\partial x_2 x_3} = a_{12}\ddot{\phi}\left(a_1^T x\right)a_{13} + a_{22}\ddot{\phi}\left(a_2^T x\right)a_{23} + \cdots + a_{K2}\ddot{\phi}\left(a_K^T x\right)a_{K3} \tag{26}$$

The upper triangular half of H is now computed, the other half is symmetric. It is possible to deduce the following pattern:

$$\frac{\partial^2 p}{\partial x_j x_i} = \sum_{k=1}^{K} a_{kj}\ddot{\phi}\left(a_k^T x\right)a_{ki}, \quad i,j \in \{1,2,3\}$$

Which can be re-written in the form of the following products

$$\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{K1} \\ a_{12} & a_{22} & \cdots & a_{K2} \\ a_{13} & a_{23} & \cdots & a_{K3} \end{bmatrix} \begin{bmatrix} \ddot{\phi}\left(a_1^T x\right) & & & \\ & \ddot{\phi}\left(a_2^T x\right) & & \\ & & \ddots & \\ & & & \ddot{\phi}\left(a_K^T x\right) \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \vdots & \vdots & \vdots \\ a_{K1} & a_{K2} & a_{K3} \end{bmatrix} \tag{27}$$

$$\therefore \nabla^2 p(x) = ADA^T \tag{28}$$

18 P2 - Task 6

Listing 11: Part 2 - Task 6 Code

```
%% Task 6
clear all; close all; clc;
load data1.mat

K = length(Y);
[l, c] = size(X);


a = 1;
gama = 10^-4;
beta = 0.5;
eps = 10^-6;

k = 1;
criteria = 0;

v_ones = -1*ones(1, length(X));

A1 = (1/K)*[X; v_ones];

va = [X; v_ones];

x(:,1) = [-ones(1,1)';0];

tic
while criteria == 0
    g(:,k) = sum(A1.*((exp(x(:,k)'*[X;v_ones]))./(1 + ...
                                    (exp(x(:,k)'*[X;v_ones])))  - Y), 2);
    grad = g(:,k);

    gnorm(k) = norm(g(:,k));
    if gnorm(k) < eps
        criteria = 1;
        break;
    end

    H = va*(1/K)*diag(((exp(x(:,k)'*[X;v_ones]))./((1 + ...
                                      (exp(x(:,k)'*[X;v_ones]))).^2)))*va';
    d(:,k) = -(H\grad);

    c = 0;
    a = 1;
    while c == 0

        b = x(:,k) + a*d(:,k);
        member1 = (1/K)*sum(log(1+exp(b'*[X;v_ones]))-Y.*(b'*[X;v_ones]));
```

```
        z = x(:,k);
        member2 = (1/K)*sum(log(1+exp(z'*[X;v_ones]))-Y.*(z'*[X;v_ones]))...
                                                    + gama*g(:,k)'*a*d(:,k);

        if member1 < member2
            c = 1;
            break;
        end
        a = beta*a;
    end

    x(:,k+1) = x(:,k) + a*d(:,k);
    step(k) = a;
    k = k + 1;
end
toc
s = [x(1, end); x(2, end)]
r = x(3, end)

figure()
semilogy(gnorm, 'linewidth', 1);
xlabel('$k$', 'interpreter','latex');
title('$|| \nabla f(s_{k},r_{k}) ||$ (Newton Method)',...
                                                'interpreter','latex');
grid on;

figure()
stem(1:1:k-1, step,'.','MarkerSize', 25, 'linewidth', 1);
xlabel('$k$', 'interpreter','latex');
title('$\alpha_k$ (Newton Method)','interpreter','latex');
```

Result `data1.mat`:

- $s = (1.3496, 1.0540)$ and $r = 4.8817$

Result `data2.mat`:

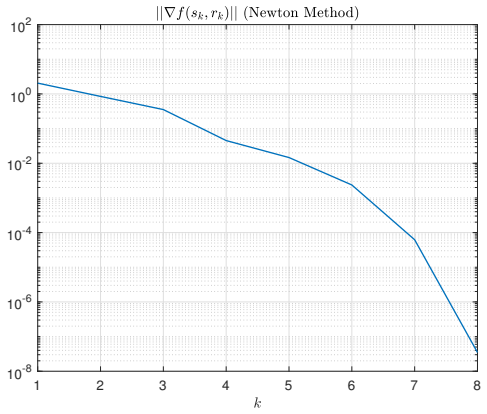
- $s = (0.7402, 2.3577)$ and $r = 4.5554$

Result `data3.mat`:

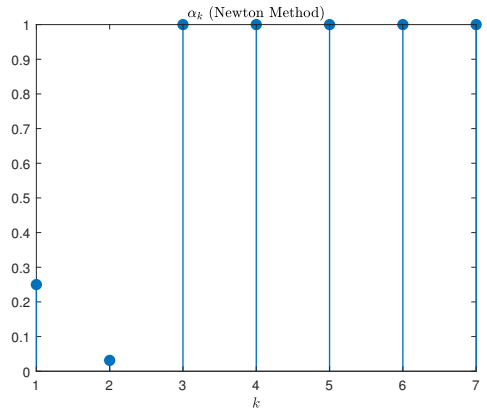
- $s = (-1.3083, 1.4079)$ and $r = 0.8049$

Result `data4.mat`:

- $s = (0.1099, -0.6424)$ and $r = 0.1019$

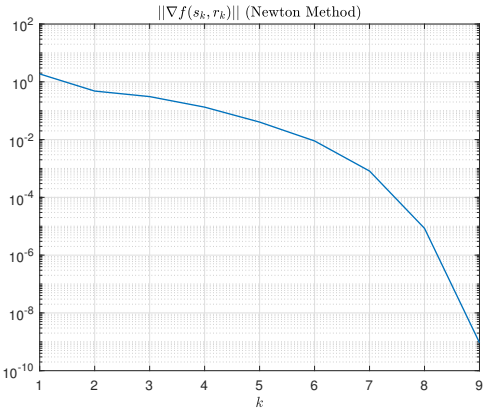


(a) Norm of the gradient along the iterations of the Newton method, when the Newton method is applied to problem (1).

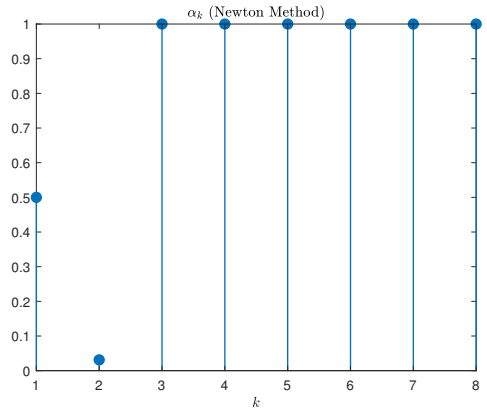


(b) Values of the stepsizes along the iterations of the Newton method, when the Newton method is applied to problem (1).

Figure 48: Newton method results (applied to `data1.mat`).

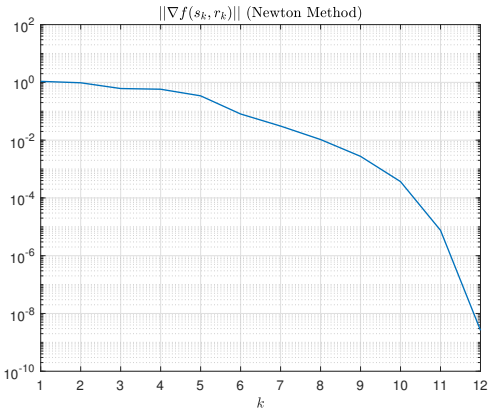


(a) Norm of the gradient along the iterations of the Newton method, when the Newton method is applied to problem (1).

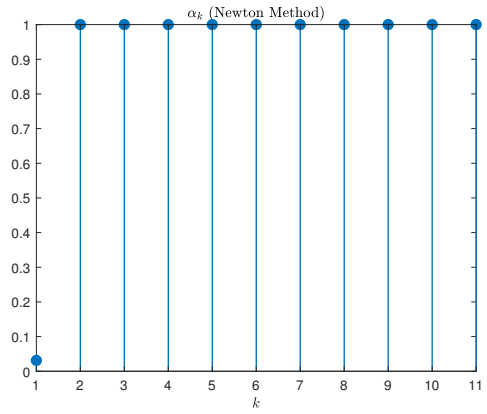


(b) Values of the stepsizes along the iterations of the Newton method, when the Newton method is applied to problem (1).

Figure 49: Newton method results (applied to `data2.mat`).

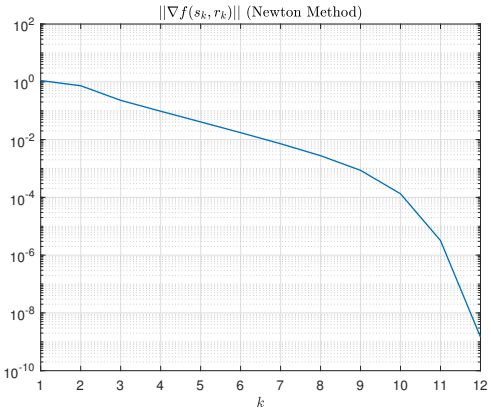


(a) Norm of the gradient along the iterations of the Newton method, when the Newton method is applied to problem (1).

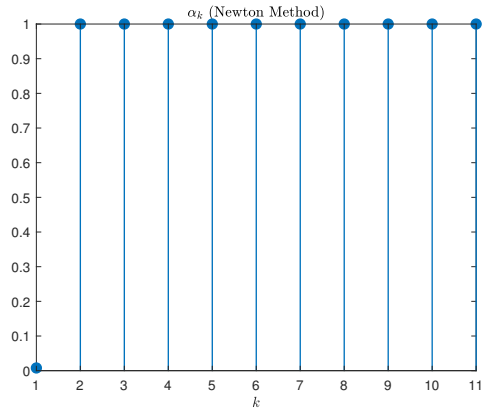


(b) Values of the stepsizes along the iterations of the Newton method, when the Newton method is applied to problem (1).

Figure 50: Newton method results (applied to `data3.mat`).



(a) Norm of the gradient along the iterations of the Newton method, when the Newton method is applied to problem (1).



(b) Values of the stepsizes along the iterations of the Newton method, when the Newton method is applied to problem (1).

Figure 51: Newton method results (applied to `data4.mat`).

19 P2 - Task 7

	Gradient Method			Newton Method		
	Time (ms)	Iterations	Time/Iter. (ms)	Time (ms)	Iterations	Time/Iter (ms)
`data1.mat`	76,1	1 126	0,0676	16,9	8	2,113
`data2.mat`	87,7	1 363	0,0643	16,4	9	1,822
`data3.mat`	776,2	3 437	0,2258	39,9	12	7,492
`data4.mat`	656 847,4	19 893	33,0190	3 673,1	12	306,092
Cost per Iteration	$\mathcal{O}(n)$			$\mathcal{O}(n^3)$		

Table 6: Time performance table of the Gradient Method and Newton Method.

Observing table 6 and the figures of the norm of the gradient, it becomes clear that the Newton method converges in a lot less number of iterations, in fact, generally speaking around 10 iterations.

However the cost per iteration of the Newton method is higher and takes more computational effort, this translates to the higher values of time per iteration relatively to the Gradient method as we can see in the table 6. This corroborates the cost per iteration of $\mathcal{O}(n^3)$ for the Newton method and $\mathcal{O}(n)$ for the Gradient Method.

Let's consider `data3.mat` with $n = 30$ and $K = 500$. This dataset has an increase of dimensions and a noticeable increase of data points. This becomes a little bit more complex for both the Gradient Method and the Newton Method. With the Gradient method, it takes more time because of the increase in the number of dimensions, where $\mathcal{O}(nk)$ and that translates to higher computational effort. With the Newton Method, it takes more time because of the increase of data points, where $\mathcal{O}(n^3)$. In essence, it is known that the Gradient method can handle better large datasets when not associated to a high number of dimensions, and the Newton Method can handle higher dimensions, but the complexity will grow to the power of 3 as more data is considered.

For `data4.mat` what was concluded before becomes even more evident, because now $n = 100$ and $K = 8000$. The number of points associated to an even higher number of dimensions increased dramatically, causing the Gradient method to converge in approximately 11 minutes, whereas the Newton method converged in 4 seconds.

To sum up, because the Newton method requires the computation of Hessians (more matrix operations), that problem becomes more complex with large datasets and the cost per iteration is more expensive, whereas the Gradient method has a lower cost per iteration, but high dimensions with large datasets might cause it to be inefficient.

20 P2 - Task 8

Listing 12: Part 2 - Task 8 Code

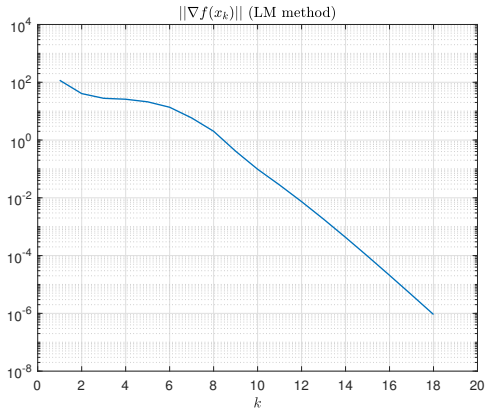
```
%% Task 8 - lmdata1.mat
clear all; close all; clc;
load lmdata1.mat

x = xinit;

eps = 10^-6;
lambda(1) = 1;

k = 1;
criteria = 0;
tic
while criteria == 0
    previous_f = 0;
    next_f = 0;
    gf1 = zeros(length(x),1);
    gf2 = zeros(length(x),1);

    gp1 = zeros(length(x), length(y));
    f1 = 0;
    f2 = 0;
    fp = [];
    for i = 1:length(y)
        ind1 = 2*iA(i,2)-1;
        ind2 = 2*iA(i,2);

        aux1 = [gf1(ind1); gf1(ind2)] + (-2)*(norm(A(:,iA(i,1)) -...
            [x(ind1); x(ind2)])-y(i)) * (A(:,iA(i,1)) - ...
            [x(ind1); x(ind2)])/norm(A(:,iA(i,1)) - [x(ind1); x(ind2)]);

        gf1(ind1) = aux1(1);
        gf1(ind2) = aux1(2);

        gpaux1 = -(A(:,iA(i,1)) - [x(ind1); x(ind2)])/norm(A(:,iA(i,1))...
                                                    - [x(ind1); x(ind2)]);

        gp1(ind1,i) = gpaux1(1);
        gp1(ind2,i) = gpaux1(2);

        f1 = f1 + (norm(A(:,iA(i,1)) - [x(ind1); x(ind2)])-y(i))^2;
        fp = [fp; (norm(A(:,iA(i,1)) - [x(ind1); x(ind2)])-y(i))];
    end

    gp2 = zeros(length(x), length(z));
    for i = 1:length(z)
        ind1 = 2*iS(i,2)-1;
```

```
47          ind2 = 2*iS(i,2);
48
49          idx1 = 2*iS(i,1)-1;
50          idx2 = 2*iS(i,1);
51
52          aux2 = [gf2(ind1); gf2(ind2); gf2(idx1); gf2(idx2)] +...
53              (-2)*(norm([x(idx1); x(idx2)] - [x(ind1); x(ind2)])-z(i)) * ...
54              ([x(idx1); x(idx2); x(ind1); x(ind2)] - ...
55              [x(ind1); x(ind2);x(idx1); x(idx2)])/norm([x(idx1); x(idx2)]...
56              - [x(ind1); x(ind2)]);
57
58          gf2(ind1) = aux2(1);
59          gf2(ind2) = aux2(2);
60          gf2(idx1) = aux2(3);
61          gf2(idx2) = aux2(4);
62
63          gpaux2 = -([x(idx1); x(idx2); x(ind1); x(ind2)] - [x(ind1);...
64              x(ind2);x(idx1); x(idx2)])/norm([x(idx1); x(idx2)] - ...
65                                                  [x(ind1); x(ind2)]);
66
67          gp2(ind1,i) = gpaux2(1);
68          gp2(ind2,i) = gpaux2(2);
69          gp2(idx1,i) = gpaux2(3);
70          gp2(idx2,i) = gpaux2(4);
71
72          f2 = f2 + (norm([x(idx1); x(idx2)] - [x(ind1); x(ind2)])-z(i))^2;
73          fp = [fp; (norm([x(idx1); x(idx2)] - [x(ind1); x(ind2)])-z(i))];
74      end
75
76      previous_f = f1 + f2;
77
78      gnorm(k) = norm(gf1+gf2);
79      if gnorm(k) < eps
80          criteria = 1;
81          break;
82      end
83
84      gp = [gp1 gp2];
85
86      Al = [gp'; sqrt(lambda(k))*eye(16)];
87      b = [gp'*x - fp; sqrt(lambda(k))*x];
88
89      new_x = Al\b;
90
91      nf1 = 0;
92      nf2 = 0;
93      for i = 1:length(y)
94          ind1 = 2*iA(i,2)-1;
95          ind2 = 2*iA(i,2);
96
97          nf1 = nf1 + (norm(A(:,iA(i,1)) - [new_x(ind1); new_x(ind2)])...
```

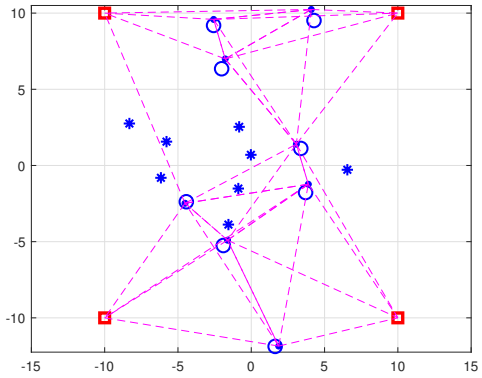
```
                                                                    -y(i))^2;
    end
    for i = 1:length(z)
        ind1 = 2*iS(i,2)-1;
        ind2 = 2*iS(i,2);

        idx1 = 2*iS(i,1)-1;
        idx2 = 2*iS(i,1);

        nf2 = nf2 + (norm([new_x(idx1); new_x(idx2)] - ...
                                        [new_x(ind1); new_x(ind2)])-z(i))^2;
    end

    next_f = nf1 + nf2;

    if next_f < previous_f
        lambda(k+1) = 0.7*lambda(k);
        x = new_x;
    else
        lambda(k+1) = 2*lambda(k);
    end

    k = k + 1;
end
toc

x1 = x(1:2:end)';
x2 = x(2:2:end)';
x = [x1;x2];

xi1 = xinit(1:2:end)';
xi2 = xinit(2:2:end)';
xi = [xi1;xi2];

figure()
semilogy(gnorm, 'linewidth', 1);
xlabel('$k$', 'interpreter','latex');
xlim([0 20]);
title('$|| \nabla f(x_{k}) ||$ (LM method)','interpreter','latex')
grid on;

a = A(:,iA(:,1))';
sa = S(:,iA(:,2))';

s1 = S(:,iS(:,1))';
s2 = S(:,iS(:,2))';

figure()
plot(S(1,:), S(2,:), 'b.', 'MarkerSize', 15); hold on;
plot(A(1,:), A(2,:), 'rs', 'LineWidth', 2.5, 'MarkerSize', 10); hold on;
plot(x(1,:), x(2,:), 'bo', 'MarkerSize', 10, 'LineWidth', 1.5); hold on;
```

```
149 plot(xi(1,:), xi(2,:), 'b*', 'LineWidth', 1.5, 'MarkerSize', 8); hold on;
150 plot([sa(:,1) a(:,1)]',[sa(:,2) a(:,2)]', 'm-'); hold on;
151 plot([s1(:,1) s2(:,1)]',[s1(:,2) s2(:,2)]', 'm-');
152 axis([-15 15 -12.25 10.5]);
153 grid on;
```

Result `lmdata1.mat`:



(a) Norm of the gradient along the iterations of the LM method, when the LM method is applied to problem (2).



(b) Network localisation setup for task 8, described in the file `lmdata1.mat`: anchors are the red squares (matrix `A`); sensors are the blue dots (matrix `S`); the initialisation for the LM method are the blue stars (vector `xinit`); from this initialisation, the LM method converged to the blue circles.

Figure 52: LM method results (applied to `lmdata1.mat`).

21 P2 - Task 9

Listing 13: Part 2 - Task 9 Code

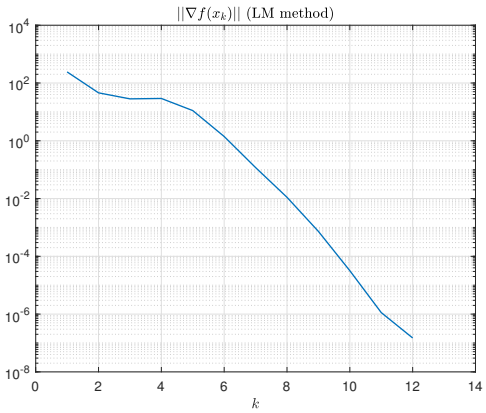
```
%% Task 8 - lmdata2.mat
clear all; close all; clc;
load lmdata2.mat


for j = 1:25
    x = randi([-15 15], 16, 1);
    xinit = x;

    eps = 10^-6;
    lambda(1) = 1;

    k = 1;
    criteria = 0;
    gnorm = [];
    tic
    while criteria == 0
        previous_f = 0;
        next_f = 0;
        gf1 = zeros(length(x),1);
        gf2 = zeros(length(x),1);

        gp1 = zeros(length(x), length(y));
        f1 = 0;
        f2 = 0;
        fp = [];
        for i = 1:length(y)
            ind1 = 2*iA(i,2)-1;
            ind2 = 2*iA(i,2);

            aux1 = [gf1(ind1); gf1(ind2)] + (-2)*(norm(A(:,iA(i,1)) - ...
                [x(ind1); x(ind2)])-y(i)) * ...
                (A(:,iA(i,1)) - [x(ind1); x(ind2)])/norm(A(:,iA(i,1)) -...
                                                    [x(ind1); x(ind2)]);

                                                gf1(ind1) = aux1(1);
            gf1(ind2) = aux1(2);

            gpaux1 = -(A(:,iA(i,1)) - [x(ind1); x(ind2)])/...
                                norm(A(:,iA(i,1)) - [x(ind1); x(ind2)]);

            gp1(ind1,i) = gpaux1(1);
            gp1(ind2,i) = gpaux1(2);

            f1 = f1 + (norm(A(:,iA(i,1)) - [x(ind1); x(ind2)])-y(i))^2;
            fp = [fp; (norm(A(:,iA(i,1)) - [x(ind1); x(ind2)])-y(i))];
```

```
47          end
48
49          gp2 = zeros(length(x), length(z));
50          for i = 1:length(z)
51              ind1 = 2*iS(i,2)-1;
52              ind2 = 2*iS(i,2);
53
54              idx1 = 2*iS(i,1)-1;
55              idx2 = 2*iS(i,1);
56
57              aux2 = [gf2(ind1); gf2(ind2); gf2(idx1); gf2(idx2)] +...
58                  (-2)*(norm([x(idx1); x(idx2)] - [x(ind1); x(ind2)])-z(i))...
59                  *([x(idx1); x(idx2); x(ind1); x(ind2)] - ...
60                  [x(ind1); x(ind2);x(idx1); x(idx2)])/...
61                  norm([x(idx1); x(idx2)] - [x(ind1); x(ind2)]);
62
63              gf2(ind1) = aux2(1);
64              gf2(ind2) = aux2(2);
65              gf2(idx1) = aux2(3);
66              gf2(idx2) = aux2(4);
67
68              gpaux2 = -([x(idx1); x(idx2); x(ind1); x(ind2)] - ...
69                  [x(ind1); x(ind2);x(idx1); x(idx2)])/...
70                  norm([x(idx1); x(idx2)] - [x(ind1); x(ind2)]);
71
72              gp2(ind1,i) = gpaux2(1);
73              gp2(ind2,i) = gpaux2(2);
74              gp2(idx1,i) = gpaux2(3);
75              gp2(idx2,i) = gpaux2(4);
76
77              f2 = f2 + (norm([x(idx1); x(idx2)] - [x(ind1); x(ind2)])-z(i))^2;
78              fp = [fp; (norm([x(idx1); x(idx2)] - [x(ind1); x(ind2)])-z(i))];
79          end
80
81          previous_f = f1 + f2;
82
83          gnorm(k) = norm(gf1+gf2);
84          if gnorm(k) < eps
85              criteria = 1;
86              break;
87          end
88
89          gp = [gp1 gp2];
90
91          Al = [gp'; sqrt(lambda(k))*eye(16)];
92          b = [gp'*x - fp; sqrt(lambda(k))*x];
93
94          new_x = Al\b;
95
96          nf1 = 0;
97          nf2 = 0;
```

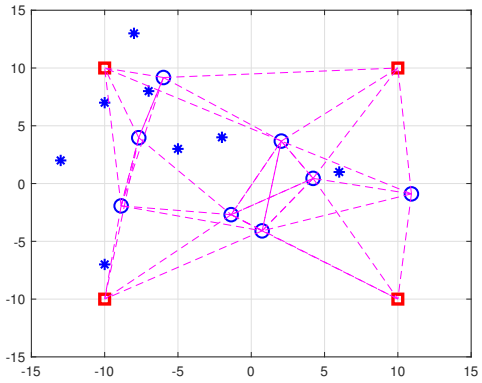
```
98          for i = 1:length(y)
99              ind1 = 2*iA(i,2)-1;
100             ind2 = 2*iA(i,2);
101
102             nf1 = nf1 + (norm(A(:,iA(i,1)) - [new_x(ind1);...
103                                                 new_x(ind2)])-y(i))^2;
104         end
105         for i = 1:length(z)
106             ind1 = 2*iS(i,2)-1;
107             ind2 = 2*iS(i,2);
108
109             idx1 = 2*iS(i,1)-1;
110             idx2 = 2*iS(i,1);
111
112             nf2 = nf2 + (norm([new_x(idx1); new_x(idx2)] - ...
113                                     [new_x(ind1); new_x(ind2)])-z(i))^2;
114         end
115
116         next_f = nf1 + nf2;
117
118         if next_f < previous_f
119             lambda(k+1) = 0.7*lambda(k);
120             x = new_x;
121         else
122             lambda(k+1) = 2*lambda(k);
123         end
124
125         k = k + 1;
126     end
127     eval_f(j) = previous_f;
128     num_iter(j) = k;
129     eval_x(:,j) = x;
130     eval_xinit(:,j) = xinit;
131     eval_gnorm(j).norm = gnorm;
132 end
133 toc
134
135 [min_cost, idx] = min(eval_f);
136 [min_iter, idxi] = min(num_iter);
137
138 if eval_f(idxi) == min_cost
139     idx = idxi;
140 end
141
142 x = eval_x(:,idx);
143 xinit = eval_xinit(:,idx);
144
145 x1 = x(1:2:end)';
146 x2 = x(2:2:end)';
147 x = [x1;x2];
148
```

```
149 xi1 = xinit(1:2:end)';
150 xi2 = xinit(2:2:end)';
151 xi = [xi1;xi2];
152
153 figure()
154 semilogy(eval_gnorm(idx).norm, 'linewidth', 1);
155 xlabel('$k$', 'interpreter','latex');
156 xlim([0 num_iter(idx)+2]);
157 title('$|| \nabla f(x_{k}) ||$ (LM method)','interpreter','latex')
158 grid on;
159
160 a = A(:,iA(:,1))';
161 sa = x(:,iA(:,2))';
162
163 s1 = x(:,iS(:,1))';
164 s2 = x(:,iS(:,2))';
165
166 figure()
167 plot(A(1,:), A(2,:), 'rs', 'LineWidth', 2.5, 'MarkerSize', 10); hold on;
168 plot(x(1,:), x(2,:), 'bo', 'MarkerSize', 10, 'LineWidth', 1.5); hold on;
169 plot(xi(1,:), xi(2,:), 'b*', 'LineWidth', 1.5, 'MarkerSize', 8); hold on;
170 plot([sa(:,1) a(:,1)]',[sa(:,2) a(:,2)]', 'm--'); hold on;
171 plot([s1(:,1) s2(:,1)]',[s1(:,2) s2(:,2)]', 'm--');
172 axis([-15 15 -15 15]);
173 grid on;
```

Result `lmdata2.mat`:
Smallest value of the cost function obtained: 4.4945.



(a) Norm of the gradient along the iterations of the LM method, when the LM method is applied to problem (2).



(b) Network localisation setup for task 9, described in the file `lmdata2.mat`: anchors are the red squares (matrix `A`); the best random initialisation for the LM method are the blue stars (random vector `xinit`); from this initialisation, the LM method converged to the blue circles.

Figure 53: LM method results (applied to `lmdata2.mat`).

22 P3 - Task 1

This task requests to solve problem 29 using the KKT conditions , that is, to find a closed-form solution for $y^\star$.

$$\begin{aligned} \underset{y \in \mathbf{R}^n}{\text{minimize}} \quad & \|p - y\|_2 \\ \text{subject to} \quad & y \in D(c, r). \end{aligned} \tag{29}$$

The cost function in the problem 29 is non-differentiable, it is not possible to use it in *KKT system*. In order to make it differentiable and use in *KKT system*, the cost function is squared and multiplied by $\frac{1}{2}$ to simplify calculus later. Since $\frac{1}{2}$ is a constant it won't affect the minimisation problem :

$$\begin{aligned} \underset{y \in \mathbf{R}^n}{\text{minimize}} \quad & \tfrac{1}{2} \|p - y^\star\|_2^2 \\ \text{subject to} \quad & \|y^\star - c\|_2^2 \leq r^2 \end{aligned} \tag{30}$$

$$\begin{aligned} \underset{y \in \mathbf{R}^n}{\text{minimize}} \quad & \tfrac{1}{2} \|p - y^\star\|_2^2 \\ \text{subject to} \quad & \|y^\star - c\|_2^2 - r^2 \leq 0 \end{aligned} \tag{31}$$

From the problem 31 it is possible to define $F(y^\star)$ and $g(y^\star)$:

$$F(y^\star) = \frac{1}{2} \|p - y^\star\|_2^2 \tag{32}$$

$$g(y^\star) = \|y^\star - c\|_2^2 - r^2 \tag{33}$$

KKT conditions

The system of equations 34 presents the **KKT conditions** used to find a closed-form solution for $y^\star$.

$$\begin{cases} \nabla F(y^\star) + \mu \nabla g(y^\star) = 0 \qquad (a) \quad (\textit{stationarity}) \\ g(y^\star) \leq 0 \qquad (b) \quad \text{(primal feasibility)} \\ \mu \geq 0 \qquad (c) \quad \text{(dual feasibility)} \\ \mu g(y^\star) = 0 \qquad (d) \quad \text{(complementary slackness)} \end{cases} \tag{34}$$

Replace the gradient of $F(y^\star)$, the gradient of $g(y^\star)$ and the function $g(y^\star)$:

$$\begin{cases} -(p - y^\star) + 2\mu(y^\star - c) = 0 \\ \|y^\star - c\|_2 - r \leq 0 \\ \mu \geq 0 \\ \mu(\|y^\star - c\|_2^2 - r^2) = 0 \end{cases} \tag{35}$$

$$\begin{cases} -p + y^\star + 2\mu y^\star - 2\mu c = 0 \\ \|y^\star - c\|_2 - r \leq 0 \\ \mu \geq 0 \\ \mu \|y^\star - c\|_2^2 - \mu r^2 = 0 \end{cases} \tag{36}$$

Look at the first equation of system 36 and try to extract $y^\star$:

$$\begin{cases} y^\star = \frac{p+2\mu c}{1+2\mu} \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ \_\_\_\_\_ \end{cases} \tag{37}$$

Replace $y^\star$ in the fourth equation of system 36 and solve in order of μ:

$$\begin{cases} y^\star = \frac{p+2\mu c}{1+2\mu} \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ \mu \left\| \frac{p+2\mu c}{1+2\mu} - c \right\|_2^2 - \mu r^2 = 0 \end{cases} \tag{38}$$

$$\begin{cases} y^\star = \frac{p+2\mu c}{1+2\mu} \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ \mu \left\| \frac{p-c}{1+2\mu} \right\|_2^2 = \mu r^2 \end{cases} \tag{39}$$

$$\mu = 0 \vee \left\| \frac{p-c}{1+2\mu} \right\|_2^2 = r^2 \tag{40}$$

$$\mu = 0 \vee \frac{\|p-c\|_2}{|1+2\mu|} = r \tag{41}$$

$$\mu = 0 \vee \mu = -\frac{1}{2} \pm \frac{\|p-c\|_2}{2r} \tag{42}$$

Recall the expression of $y^\star$:

$$y^\star = \frac{p+2\mu c}{1+2\mu} \tag{43}$$

The solution $\mu = 0$ is considered if we assume that the point can be inside the disk, because if you replace $\mu = 0$ in the equation of $y^\star$ you obtain the point p. That means that the distance of the point p to the disk is the point itself. That said, the value of μ used to calculate the closed form expression is given by the following equation:

$$\mu = -\frac{1}{2} \pm \frac{\|p - c\|_2}{2r} \tag{44}$$

Plug μ in the expression of $y^\star$:

$$y^\star = \frac{1}{1 + (-1 \pm \frac{\|p-c\|_2}{r})}(p + (-1 \pm \frac{\|p - c\|_2}{r})c) \tag{45}$$

$$y^\star = c \pm \frac{(p - c)}{\|p - c\|_2} r \tag{46}$$

Considering $\frac{(p-c)}{\|p-c\|_2}$ a unit norm vector collinear with the line formed by p and c and oriented to p, choosing the positive sign or the negative is a matter of geometrically keeping the orientation of the unit norm vector towards p or choosing the opposite orientation.

Since the distance is being minimized the orientation of the unit norm vector should be towards p. This unit norm vector is applied on c and multiplied by a factor r. This factor r should be equal to the radius of the disk, in order to get the geometrically farthest points in relation to the centre of the disk, also the closest points of the disk to the point p.

As a result, the solution uses the positive sign

$$y^\star = c + \frac{(p - c)}{\|p - c\|_2} r \tag{47}$$

23 P3 - Task 2

Analysing the function

$$\underset{x,u}{\text{minimize}} \sum_{k=1}^{K} \|Ex(\tau_k) - w_k\|_2^2 + \lambda \sum_{t=1}^{T-1} \|u(t) - u(t-1)\|_2^2 =$$

$$= \underset{x,u}{\text{minimize}} \sum_{k=1}^{K} (Ex(\tau_k) - w_k)^T (Ex(\tau_k) - w_k) + \lambda \sum_{t=1}^{T-1} (u(t) - u(t-1))^T (u(t) - u(t-1))$$

$$= \underset{X}{\text{minimize}} \quad \frac{1}{2}(X - P)^T A_m (X - P)$$

Where

$$A_m = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \lambda & & \\ & & & & \ddots & \\ & & & & & \lambda \end{bmatrix}, \quad X = \begin{bmatrix} Ex(\tau_1) \\ \vdots \\ Ex(\tau_K) \\ u(1) \\ \vdots \\ u(T-1) \end{bmatrix} \quad \text{and} \quad P = \begin{bmatrix} w_1 \\ \vdots \\ w_K \\ u(0) \\ \vdots \\ u(T-2) \end{bmatrix}$$

The diagonal of ones is of size K and the diagonal of λ is of size $T-2$.

Analysing the constraints

$$x(t+1) = Ax(t) + bu(t) \quad t = 1, \ldots, T-1 \tag{48}$$

$$x(1) = \begin{bmatrix} P_1 \\ V_1 \end{bmatrix} \tag{49}$$

$$x(T) = \begin{bmatrix} P_2 \\ V_2 \end{bmatrix} \tag{50}$$

$$A^T \begin{bmatrix} P_1 \\ V_1 \end{bmatrix} + A^{T-1} B u(1) + \cdots + B u(T-1) = \begin{bmatrix} P_2 \\ V_2 \end{bmatrix} \tag{51}$$

$$A^{T-1} B u(1) + \cdots + B u(T-1) = \begin{bmatrix} P_2 \\ V_2 \end{bmatrix} - A^T \begin{bmatrix} P_1 \\ V_1 \end{bmatrix} \tag{52}$$

$$\begin{bmatrix} A^{T-1}B & \ldots & B \end{bmatrix} \begin{bmatrix} u(1) \\ \vdots \\ u(T-1) \end{bmatrix} = \begin{bmatrix} P_2 \\ V_2 \end{bmatrix} - A^T \begin{bmatrix} P_1 \\ V_1 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix}0_1 & \dots & 0_K & A^{T-1}B & \dots & B\end{bmatrix}}_{C}\underbrace{\begin{bmatrix}Ex(\tau_1)\\ \vdots \\ Ex(\tau_K)\\ u(1)\\ \vdots \\ u(T-1)\end{bmatrix}}_{X} = \underbrace{\begin{bmatrix}P_2\\ V_2\end{bmatrix} - A^T\begin{bmatrix}P_1\\ V_1\end{bmatrix}}_{d}$$

Applying the KKT conditions

$$\begin{cases}\nabla F(X) + \gamma \nabla h(X) = 0\\ h(X) = 0\end{cases} = \begin{cases}A_m(X-P) + \gamma C^T = 0\\ CX = d\end{cases} =$$

$$= \begin{cases}X = P - A_m^{-1}C^T\gamma\\ CX = d\end{cases} = \begin{cases}X = P - A_m^{-1}C^T\gamma\\ C(P - A_m^{-1}C^T\gamma) = d\end{cases} = \begin{cases}X = P - A_m^{-1}C^T\gamma\\ CP - d = \gamma(CA_m^{-1}C^T)\end{cases} =$$

$$= \begin{cases}X = P - A_m^{-1}C^T\gamma\\ \gamma = (CA_m^{-1}C^T)^{-1}(CP-d)\end{cases} =$$

$$= X = P - A_m^{-1}C^T(CA_m^{-1}C^T)^{-1}(CP-d) \tag{53}$$