Note: Due to file upload size limit only including final output screenshot of MATLAB command window for each question. For report with all screenshots please refer to report uploaded on google drive.

Solution to Q1 (LP formulation to Network flow problem):
In the given Network flow problem the goal is to minimize the total cost of flow through the network, subject to the constraints described. This can be formulated as the LP:

$$\begin{aligned} &\textit{minimize} && \sum_{i,j=1}^{n} c_{ij}x_{ij} \\ &\textit{subject to} && b_i + \sum_{j=1}^{n} x_{ji} - \sum_{j=1}^{n} x_{ij} = 0 \quad i = 1,\dots,n \\ &&& l_{ij} \le x_{ij} \le u_{ij} \end{aligned}$$

Here, we vary the upper bound for the data rates on each link (denoted by $u_{i,j}$) as 5Mbps, 10Mbps, 30 Mbps, 50 Mbps and 100Mbps. And observe the trend in the cost function as the maximum permissible data rate per link is increased.

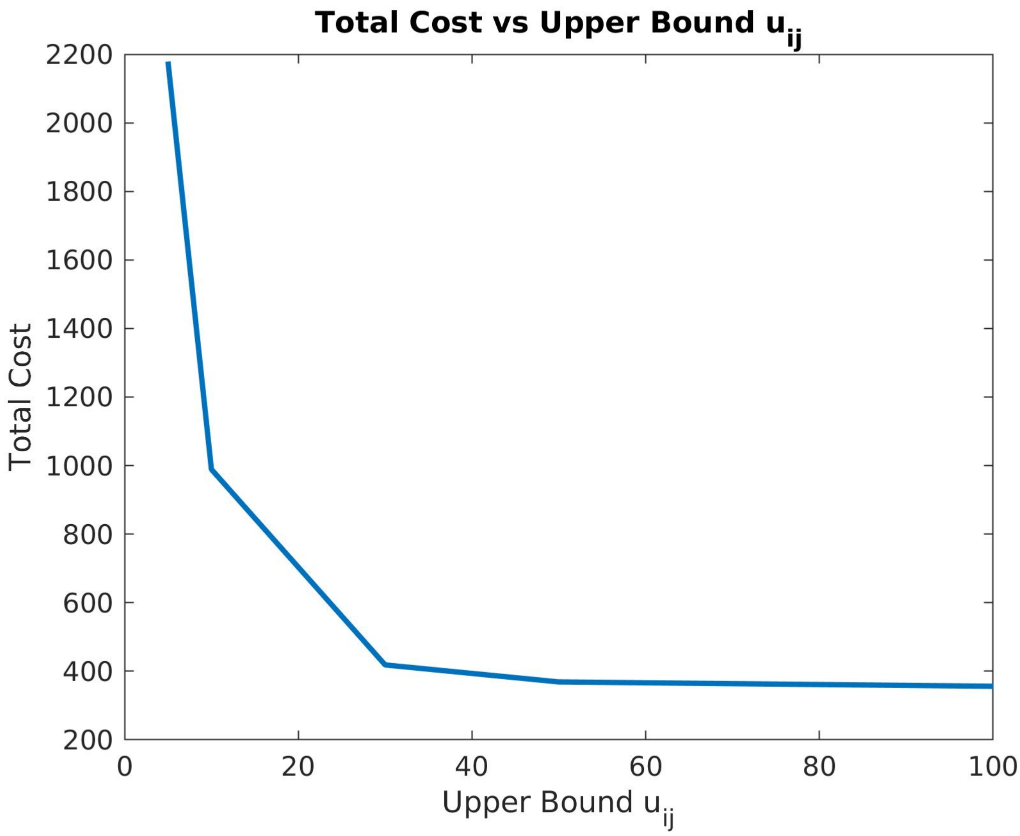


Figure 1: Plot of Total Cost vs Upper Bound u_{ij}

We can observe that as the upper bound for the data rates on each link increases the total cost of flow through the network decreases. This is because the cost of the flow along any link from node i to node j is given by $c_{ij}x_{ij}$, where c_{ij} are given constants and doesn't change, so by increasing upper bound we can allow more data to flow along the link which has low value for c_{ij} and thus we can reduce the total cost of flow through the network.

Optimal Value: Minimum total cost of flow through the network is **+356.21** for u_{ij} = **100 Mbps**.

Code:

```
clc; clear all; close all;
%% Initialization
n = 100;
network = rand(n,2); % Generating a network with 100 nodes
l = 0; % lower bound

cost = zeros(n,n);
for i = 1:n
    for j=1:n
        cost(i,j) = norm(network(i) - network(j))/sqrt(2); % Computing cost matrix
    end
end

b = zeros(n,1); % external supply
b(1:n) = 200;
b(n/2 + 1:end) = -200;

%% Solving using CVX for different values of u_ij

U_ranges = [5,10,30,50,100]; % ranges of u_ij upper bound
cost_function_trend = zeros(length(U_ranges),1);
itr = 1;

for u = U_ranges
    cvx_begin
          variables x(n,n)
          minimize(sum(sum(cost.*x))); % Minimizing total cost across the network

          for i = 1:n
             b(i) +  sum(x(:,i)) - sum(x(i,:)) == 0; % conservation of flow
          end

          for i = 1:n
              for j = 1:n
                  x(i,j) >= l; % lower bound
                  x(i,j) <= u; % upper bound
              end
          end


%        Other method to specify constraints
%            b + sum(x,1)' - sum(x,2) == zeros(n,1); % conservation of flow
%
%            x(:)>=l*ones(n*n,1); % lower bound
%            x(:)<=u*ones(n*n,1); % upper bound

    cvx_end

    cost_function_trend(itr) = cvx_optval;
    itr = itr+1;
end

%% Plotting Total Cost vs Upper Bound u_ij

plot(U_ranges,cost_function_trend,'LineWidth',2);
title('Total Cost vs Upper Bound u_{ij}');
xlabel('Upper Bound u_{ij}');
ylabel('Total Cost');
print('-djpeg','Plot_Q1.jpg', '-r300'); % Saving image
close all;
```

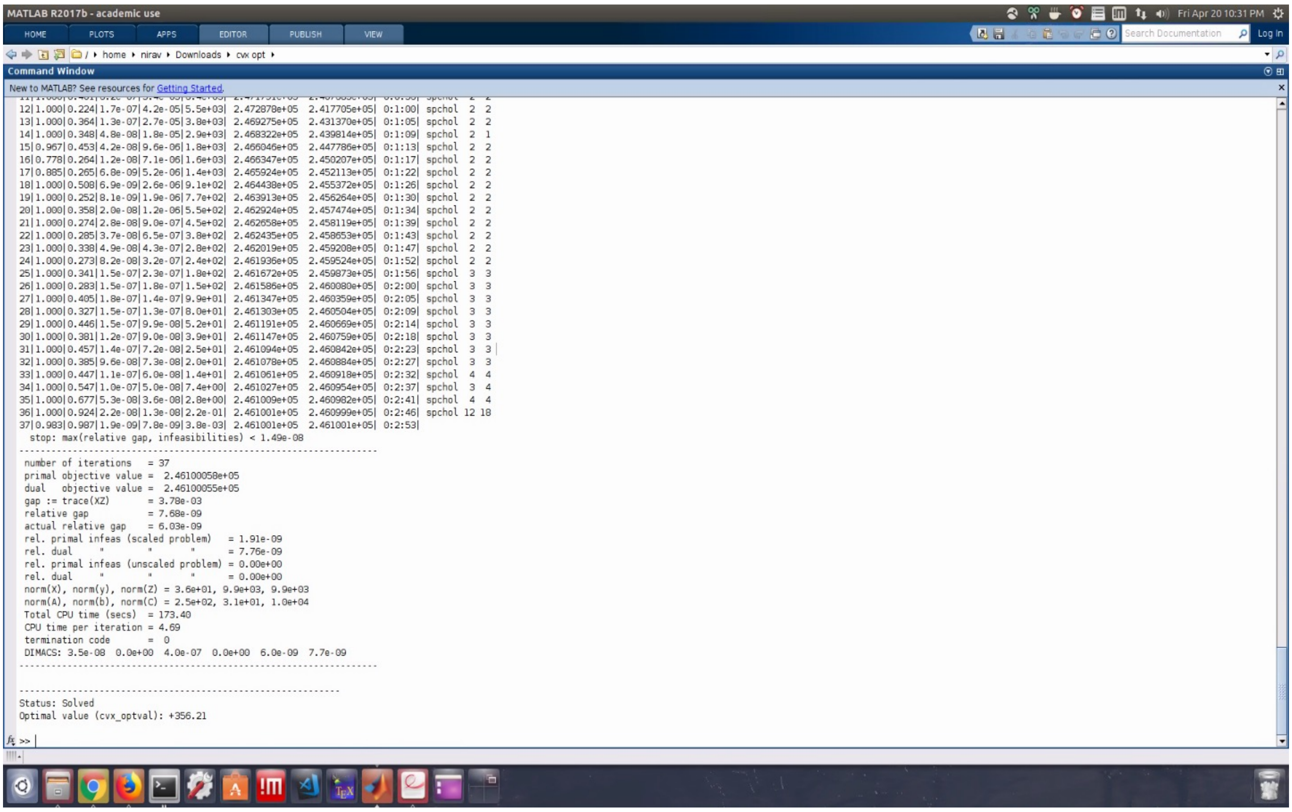


Figure 2: Screenshot of the MATLAB command window for Q1

Solution to Q2 (Eigenvalue optimization via SDP):

SDP Formulation for Part (a): Minimize the maximum eigenvalue $\lambda_1(x)$.

$$\begin{aligned} &\textit{minimize} \;\; t \\ &\textit{subject to} \;\; A(x) \preccurlyeq tI \end{aligned}$$

The variables are $x \in \mathbb{R}^n$ and $t \in \mathbb{R}$.
Optimal Value: $-\infty$ (Unbounded)

SDP Formulation for Part (b): Minimize the spread of the eigenvalues, $\lambda_1(x) - \lambda_m(x)$.

$$\begin{aligned} &\textit{minimize} \;\; t_1 - t_2 \\ &\textit{subject to} \;\; t_2 I \preccurlyeq A(x) \preccurlyeq t_1 I \end{aligned}$$

The variables are $t_1 \in \mathbb{R}$, $t_2 \in \mathbb{R}$ and $x \in \mathbb{R}^2$.
Optimal Value: +1.74635

SDP Formulation for Part (c): Minimize the condition number of $A(x)$, subject to $A(x) > 0$.

$$\begin{aligned} &\textit{minimize} \;\; t \\ &\textit{subject to} \;\; I \preccurlyeq sA_0 + x_1 A_1 + x_2 A_2 \preccurlyeq tI \\ &\qquad\qquad s \geq 0 \end{aligned}$$

The variables are $t \in \mathbb{R}$, $s \in \mathbb{R}$ and $x \in \mathbb{R}^2$.
Optimal Value: +22.8898

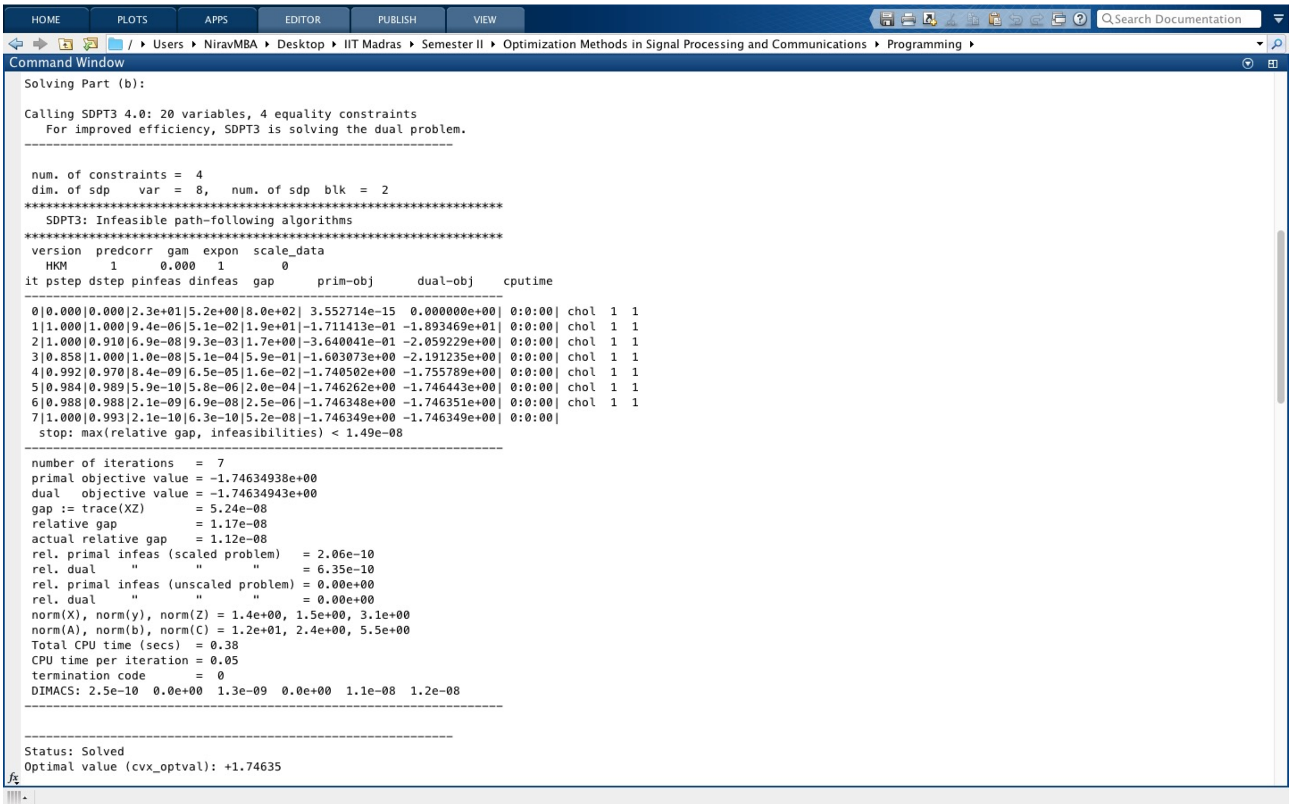
SDP Formulation for Part (d): Minimize the sum of the absolute values of the eigenvalues.

$$\begin{aligned} &\textit{minimize} \;\; \mathbf{tr}A^+ - \mathbf{tr}A^- \\ &\textit{subject to} \;\; A(x) = A^+ - A^- \\ &\qquad A^+ \succcurlyeq 0 \\ &\qquad A^- \succcurlyeq 0 \end{aligned}$$

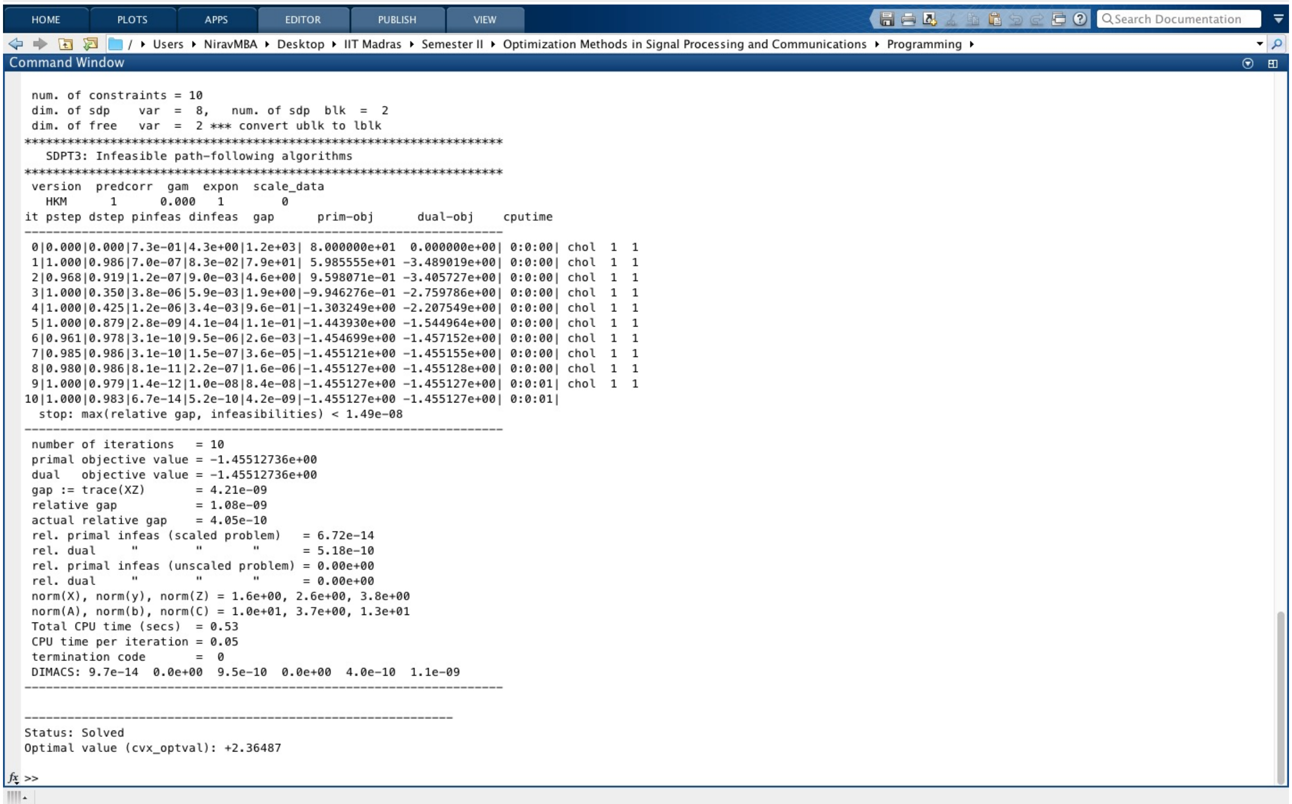
The variables are $A^+, A^- \in S^2$ and $x \in \mathbb{R}^2$.
Optimal Value: +2.36487

Code:

```
clc; clear all;

%% Initializing A's

A0 = [0.16, 0.43, 0.36, 0.37; 0.43, 1.60, 1.07, 0.84; 0.36,  1.07, 0.87, 0.65; 0.37,
    0.84, 0.65, 1.19];
A1 = [1.88, 1.36, 0.57,1.84; 1.36, 1.26, 0.39, 1.23; 0.57, 0.39, 0.82, 1.14; 1.84,
    1.23, 1.14, 2.43];
A2 = [1.38,1.24, 1.10,1.17; 1.24, 1.49, 1.22, 1.20; 1.10, 1.22, 1.38, 1.17; 1.17, 1.20,
     1.17, 1.15];

%% a) Minimize the maximum eigenvalue lambda_1(x)

fprintf('Solving Part (a): \n');

cvx_begin sdp
      variables x(2) t
      minimize(t); % Minimizing objective
      A0 + x(1)*A1 + x(2)*A2 <= t*eye(4); % constraints
cvx_end

%% b) Minimize the spread of the eigenvalues, lambda_1(x) - lambda_m(x)

fprintf('Solving Part (b): \n');

cvx_begin sdp
      variables x(2) t1 t2
      minimize(t1 - t2); % Minimizing objective
      A0 + x(1)*A1 + x(2)*A2 <= t1*eye(4);
      A0 + x(1)*A1 + x(2)*A2 >= t2*eye(4);
cvx_end

%% c)Minimize the condition number of A(x), subject to A(x) > 0.


fprintf('Solving Part (c): \n');

cvx_begin sdp
      variables y(2) t s
      minimize(t); % Minimizing objective
      s*A0 + y(1)*A1 + y(2)*A2 <= t*eye(4);
      s*A0 + y(1)*A1 + y(2)*A2 >= eye(4);
      s>=0;
cvx_end

%% d) Minimize the sum of the absolute values of the eigenvalues.

fprintf('Solving Part (d): \n');

cvx_begin sdp
      variables x(2)
      variable A_plus(4,4) symmetric
      variable A_minus(4,4) symmetric
      minimize(trace(A_plus) + trace(A_minus)); % Minimizing objective
      A0 + x(1)*A1 + x(2)*A2 == A_plus - A_minus;
      A_plus >= 0;
      A_minus >= 0;
cvx_end
```



```
Solving Part (b):

Calling SDPT3 4.0: 20 variables, 4 equality constraints
   For improved efficiency, SDPT3 is solving the dual problem.
------------------------------------------------------------


 num. of constraints =  4
 dim. of sdp    var  =  8,   num. of sdp  blk  =  2
*******************************************************************
   SDPT3: Infeasible path-following algorithms
*******************************************************************
 version  predcorr  gam  expon  scale_data
   HKM      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|2.3e+01|5.2e+00|8.0e+02| 3.552714e-15  0.000000e+00| 0:0:00| chol  1  1
 1|1.000|1.000|9.4e-06|5.1e-02|1.9e+01|-1.711413e-01 -1.893469e+01| 0:0:00| chol  1  1
 2|1.000|0.910|6.9e-08|9.3e-03|1.7e+00|-3.640041e-01 -2.059229e+00| 0:0:00| chol  1  1
 3|0.858|1.000|1.0e-08|5.1e-04|5.9e-01|-1.603073e+00 -2.191235e+00| 0:0:00| chol  1  1
 4|0.992|0.970|8.4e-09|6.5e-05|1.6e-02|-1.740502e+00 -1.755789e+00| 0:0:00| chol  1  1
 5|0.984|0.989|5.9e-10|5.8e-06|2.0e-04|-1.746262e+00 -1.746443e+00| 0:0:00| chol  1  1
 6|0.988|0.988|2.1e-09|6.9e-08|2.5e-06|-1.746348e+00 -1.746351e+00| 0:0:00| chol  1  1
 7|1.000|0.993|2.1e-10|6.3e-10|5.2e-08|-1.746349e+00 -1.746349e+00| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   =  7
 primal objective value = -1.74634938e+00
 dual   objective value = -1.74634943e+00
 gap := trace(XZ)       = 5.24e-08
 relative gap           = 1.17e-08
 actual relative gap    = 1.12e-08
 rel. primal infeas (scaled problem)   = 2.06e-10
 rel. dual     "        "       "      = 6.35e-10
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 1.4e+00, 1.5e+00, 3.1e+00
 norm(A), norm(b), norm(C) = 1.2e+01, 2.4e+00, 5.5e+00
 Total CPU time (secs)  = 0.38
 CPU time per iteration = 0.05
 termination code       =  0
 DIMACS: 2.5e-10  0.0e+00  1.3e-09  0.0e+00  1.1e-08  1.2e-08
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +1.74635
```



```
 num. of constraints = 10
 dim. of sdp    var  =  8,   num. of sdp  blk  =  2
 dim. of free   var  =  2 *** convert ublk to lblk
*******************************************************************
   SDPT3: Infeasible path-following algorithms
*******************************************************************
 version  predcorr  gam  expon  scale_data
   HKM      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|7.3e-01|4.3e+00|1.2e+03| 8.000000e+01  0.000000e+00| 0:0:00| chol  1  1
 1|1.000|0.986|7.0e-07|8.3e-02|7.9e+01| 5.985555e+01 -3.489019e+00| 0:0:00| chol  1  1
 2|0.968|0.919|1.2e-07|9.0e-03|4.6e+00| 9.598071e-01 -3.405727e+00| 0:0:00| chol  1  1
 3|1.000|0.350|3.8e-06|5.9e-03|1.9e+00|-9.946276e-01 -2.759786e+00| 0:0:00| chol  1  1
 4|1.000|0.425|1.2e-06|3.4e-03|9.6e-01|-1.303249e+00 -2.207549e+00| 0:0:00| chol  1  1
 5|1.000|0.879|2.8e-09|4.1e-04|1.1e-01|-1.443930e+00 -1.544964e+00| 0:0:00| chol  1  1
 6|0.961|0.978|3.1e-10|9.5e-06|2.6e-03|-1.454699e+00 -1.457152e+00| 0:0:00| chol  1  1
 7|0.985|0.986|3.1e-10|1.5e-07|3.6e-05|-1.455121e+00 -1.455155e+00| 0:0:00| chol  1  1
 8|0.980|0.986|8.1e-11|2.2e-07|1.6e-06|-1.455127e+00 -1.455128e+00| 0:0:00| chol  1  1
 9|1.000|0.979|1.4e-12|1.0e-08|8.4e-08|-1.455127e+00 -1.455127e+00| 0:0:01| chol  1  1
10|1.000|0.983|6.7e-14|5.2e-10|4.2e-09|-1.455127e+00 -1.455127e+00| 0:0:01|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   = 10
 primal objective value = -1.45512736e+00
 dual   objective value = -1.45512736e+00
 gap := trace(XZ)       = 4.21e-09
 relative gap           = 1.08e-09
 actual relative gap    = 4.05e-10
 rel. primal infeas (scaled problem)   = 6.72e-14
 rel. dual     "        "       "      = 5.18e-10
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 1.6e+00, 2.6e+00, 3.8e+00
 norm(A), norm(b), norm(C) = 1.0e+01, 3.7e+00, 1.3e+01
 Total CPU time (secs)  = 0.53
 CPU time per iteration = 0.05
 termination code       =  0
 DIMACS: 9.7e-14  0.0e+00  9.5e-10  0.0e+00  4.0e-10  1.1e-09
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +2.36487

>>
```

Figure 3: Screenshot of the MATLAB command window for Q2

Solution to Q3 (Norm minimization):

Part (a): Obtain the optimality conditions for this problem.
Norm minimization problem

$$minimize \quad \|Ax - b\|_{\frac{3}{2}}$$

can be re-written as

$$minimize \quad (\sum_{i=1}^{m} |a_i^T x - b_i|^{\frac{3}{2}})^{\frac{2}{3}}$$

where a_i's are the rows of A.
Optimal Value: +2.47141

Part (b): Formulate this problem as an SDP.

$$\begin{aligned} &minimize \;\; 1^T t \\ &subject\ to \;\; s_i^{3/2} \leq t_i \;\; i = 1, \ldots, m \\ &\qquad\qquad\quad - s_i \leq a_i^T x - b_i \leq s_i \;\; i = 1, \ldots, m \end{aligned}$$

First m inequalities can be re-written as $s_i^2 \leq \sqrt{s_i} t_i$ which can be written as a matrix inequality using a characterization of PSD matrices. But these matrix inequalities are not linear. Hence, by introducing new variable u for the non-linear terms as: $u_i \leq \sqrt{s_i}$ we can obtain the LMI formulation to formulate above problem as SDP.

$$\begin{aligned} &minimize \;\; 1^T t \\ &subject\ to \;\; - s_i \leq a_i^T x - b_i \leq s_i \quad i = 1, \ldots, m \\ &\qquad\qquad\quad \begin{bmatrix} u_i & s_i \\ s_i & t_i \end{bmatrix} \geq 0 \quad i = 1, \ldots, m \\ &\qquad\qquad\quad u_i \leq \sqrt{s_i} \quad i = 1, \ldots, m \end{aligned}$$

Optimal Value: +3.88523

Code:

```
clc; clear all;
%% Initializing A & B
A = [-0.94, 1.19; -1.67,1.19; 0.13,-0.04; 0.29,0.33; -1.15,0.18];
B = [-0.19;0.72;-0.59;2.18;-0.14];
m = length(A);
%% a) Using norm.

fprintf('Solving Part (a): \n');

cvx_begin
      variables x(2,1)
        minimize(norm(A*x-B,3/2)); % Minimizing objective
cvx_end



%% b) Formulate this problem as an SDP.

fprintf('Solving Part (b): \n');

cvx_begin sdp
      variables x(2) t(m,1) u(m,1) s(m,1)
      minimize(sum(t)); % Minimizing objective

      A*x - B <= s; % LMI Constraints
      A*x - B >= -s;


      for i = 1:m
          [u(i) s(i); s(i) t(i)] >= 0; % PSD Constraints
```

```
            u(i) <= sqrt(s(i));
        end

cvx_end
```

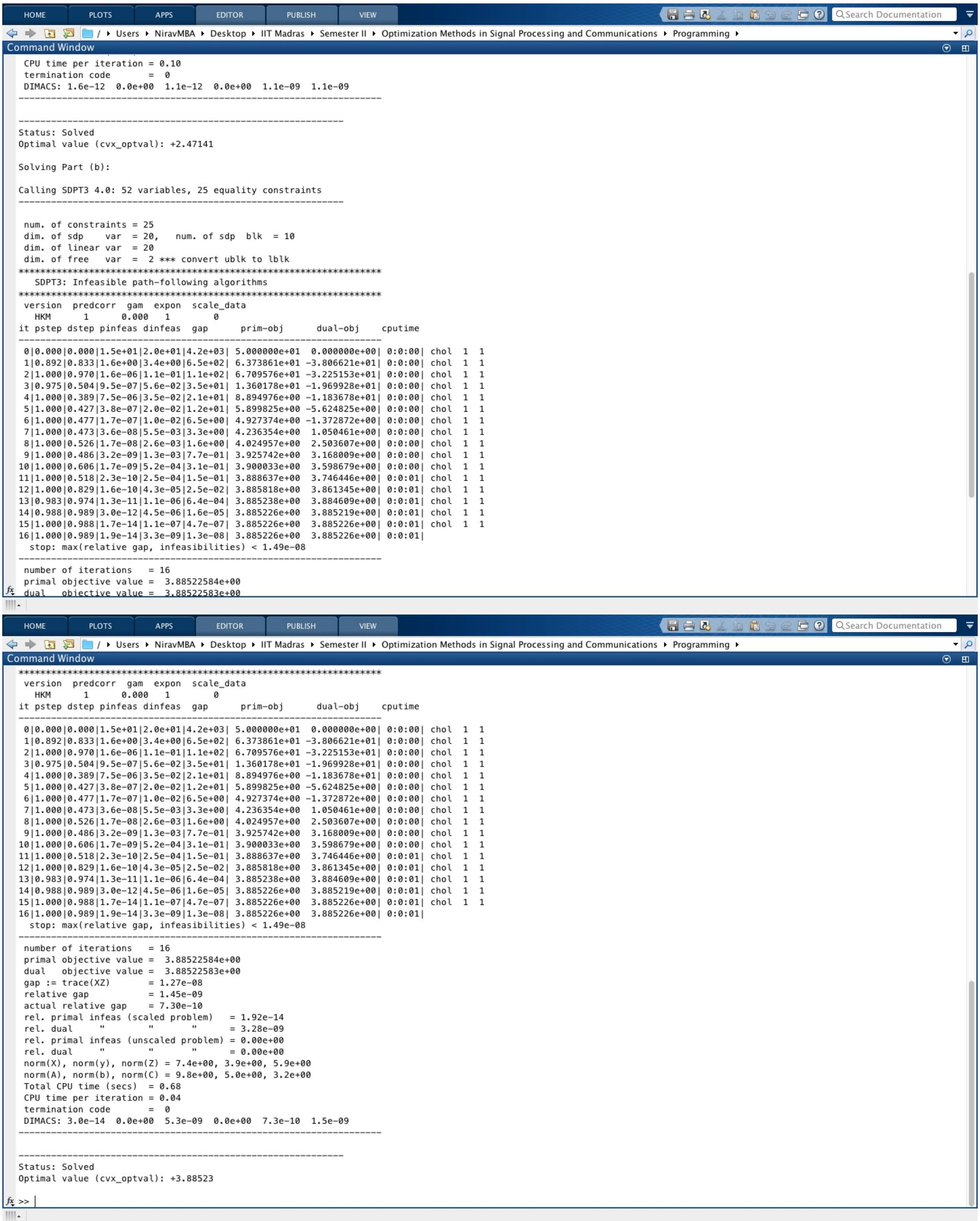


Figure 4: Screenshot of the MATLAB command window for Q3

Solution to Q4 (Optimal vehicle speed scheduling):

Part (a): For a given problem, the fuel consumed over the i^{th} segment is $(d_i/s_i)\Phi(s_i)$, so the total fuel used is $\sum_{i=1}^{n}(d_i/s_i)\Phi(s_i)$. The vehicle arrives at waypoint i at time $\tau_i = \sum_{j=1}^{n}(d_j/s_j)$. Thus our problem is

$$
\begin{array}{ll}
\textit{minimize} & \sum_{i=1}^{n}(d_i/s_i)\Phi(s_i) \\
\textit{subject to} & s_i^{min} \le s_i \le s_i^{max} \quad i = 1,\ldots,n \\
& \tau_i^{min} \le \sum_{j=1}^{n}(d_j/s_j) \le \tau_i^{max} \quad i = 1,\ldots,n
\end{array}
$$

with variables $s_1,\ldots,s_n$.
This is not a convex problem in the current form: the objective function need not be convex in s_i, but the inequalities $\tau_i^{min} \le \sum_{j=1}^{n}(d_j/s_j)$ are not convex.

However, by making a change of variables we can formulate this as a convex problem. We now formulate the problem using the transit times of the segments, t_i, as the optimization variable, where $t_i = d_i/s_i$. (We then have $s_i = d_i/t_i$.) Our problem can be written as

$$
\begin{array}{ll}
\textit{minimize} & \sum_{i=1}^{n} t_i\Phi(d_i/t_i) \\
\textit{subject to} & d_i/s_i^{max} \le t_i \le d_i/s_i^{min} \quad i = 1,\ldots,n \\
& \tau_i^{min} \le \sum_{j=1}^{n} t_j \le \tau_i^{max} \quad i = 1,\ldots,n
\end{array}
$$

with variables $t_1,\ldots,t_n$.
This is now a convex problem. The function $t_i\Phi(d_i/t_i)$, the perspective of Φ, is convex jointly in d_i and t_i; in particular, it is convex in t_i. Therefore the objective function is convex, since it is a positive weighted sum of convex functions. The constraints are all linear in t. Then once we solve this problem using CVX and find t_i^* we recover the optimal speeds using $s_i^* = d_i^*/t_i^*$.

Part (b): Optimal Value: Optimal fuel consumption is +**2617.83**

Code:

```
clc; clear all; close all;
%% Loading data from file
veh_speed_sched_data

%% Sloving using CVX
    cvx_begin
        variable t(n) % Transit times of the segments (doing change of variables)
        minimize(sum(a*d.^2.*inv_pos(t)+b*d+c*t)) % Minimize transformed objective
        t<=d./smin;
        t>=d./smax;
        tau_min<=cumsum(t);
        tau_max>=cumsum(t);
    cvx_end

%% Ploting graph of Optimal Speed vs Segment
    s=d./t; % optimal speed
    stairs(s,'LineWidth',2); % Using stairs to show constant speed over the segments
    title('Optimal Speed vs Segment');
    xlabel('Segment i');
    ylabel('Optimal Speed s_i');
    print('-djpeg','speed.jpg', '-r300'); % Saving image
    close all;
```

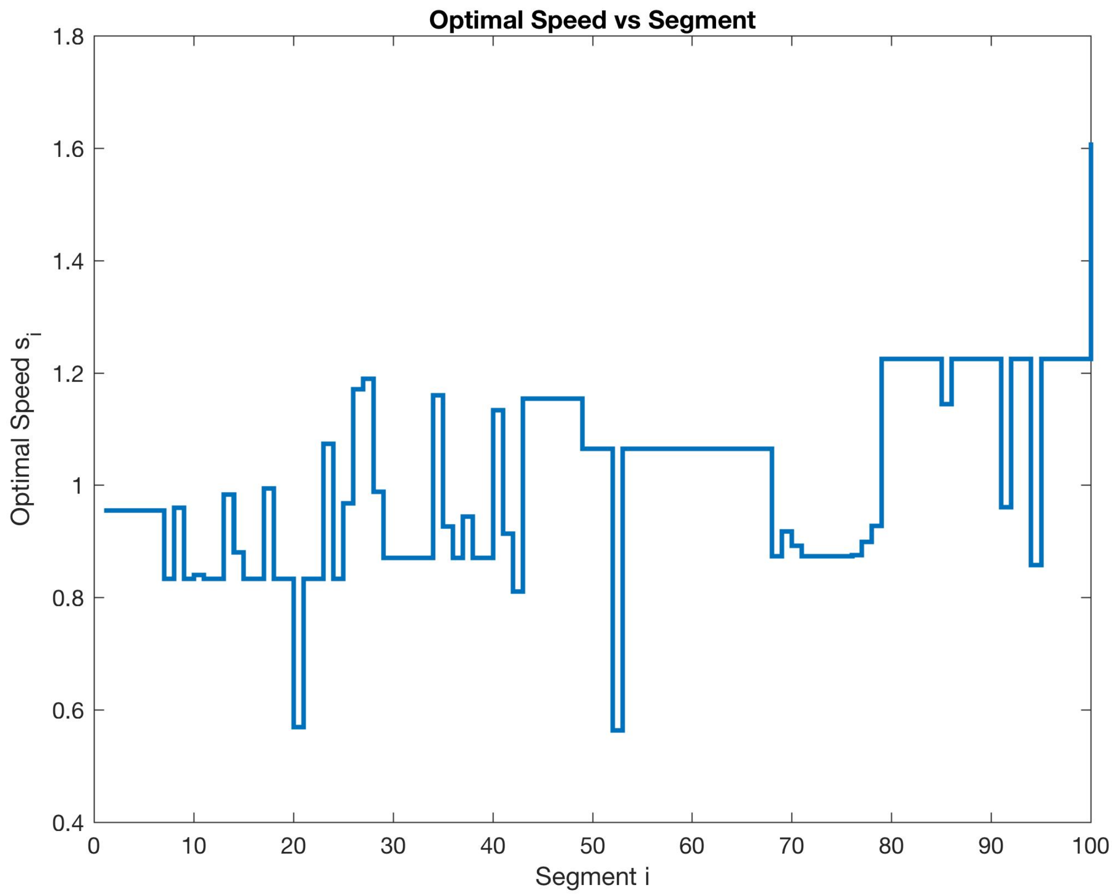
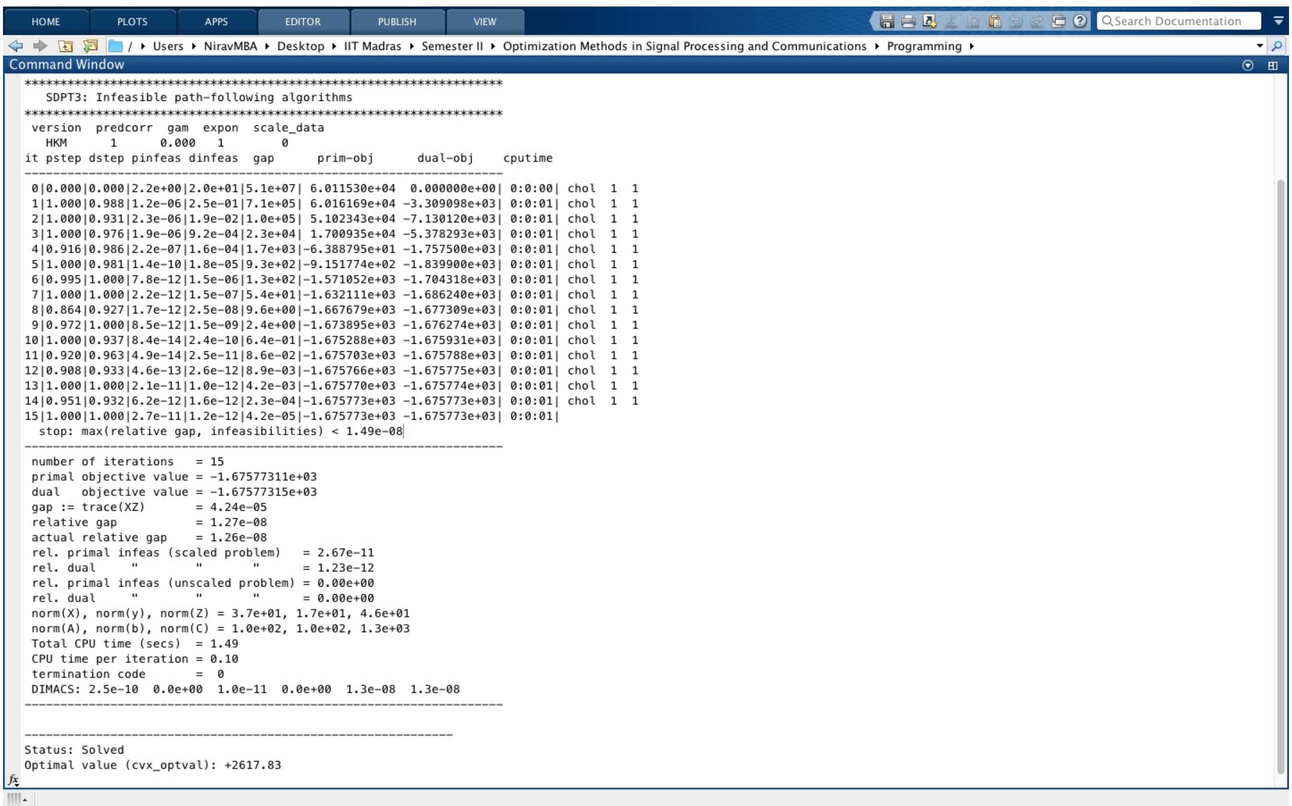


Figure 5: Plot of Optimal Speed versus Segment



```
*********************************************************************
   SDPT3: Infeasible path-following algorithms
*********************************************************************
 version  predcorr  gam  expon  scale_data
   HKM      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
---------------------------------------------------------------------
 0|0.000|0.000|2.2e+00|2.0e+01|5.1e+07| 6.011530e+04  0.000000e+00| 0:0:00| chol  1  1
 1|1.000|0.988|1.2e-06|2.5e-01|7.1e+05| 6.016169e+04 -3.309098e+03| 0:0:01| chol  1  1
 2|1.000|0.931|2.3e-06|1.9e-02|1.0e+05| 5.102343e+04 -7.130120e+03| 0:0:01| chol  1  1
 3|1.000|0.976|1.9e-06|9.2e-04|2.3e+04| 1.700935e+04 -5.378293e+03| 0:0:01| chol  1  1
 4|0.916|0.986|2.2e-07|1.6e-04|1.7e+03|-6.388795e+01 -1.757500e+03| 0:0:01| chol  1  1
 5|1.000|0.981|1.4e-10|1.8e-05|9.3e+02|-9.151774e+02 -1.839900e+03| 0:0:01| chol  1  1
 6|0.995|1.000|7.8e-12|1.5e-06|1.3e+02|-1.571052e+03 -1.704318e+03| 0:0:01| chol  1  1
 7|1.000|1.000|2.2e-12|1.5e-07|5.4e+01|-1.632111e+03 -1.686240e+03| 0:0:01| chol  1  1
 8|0.864|0.927|1.7e-12|2.5e-08|9.6e+00|-1.667679e+03 -1.677309e+03| 0:0:01| chol  1  1
 9|0.972|1.000|8.5e-12|1.5e-09|2.4e+00|-1.673895e+03 -1.676274e+03| 0:0:01| chol  1  1
10|1.000|0.937|8.4e-14|2.4e-10|6.4e-01|-1.675288e+03 -1.675931e+03| 0:0:01| chol  1  1
11|0.920|0.963|4.9e-14|2.5e-11|8.6e-02|-1.675703e+03 -1.675788e+03| 0:0:01| chol  1  1
12|0.908|0.933|4.6e-13|2.6e-12|8.9e-03|-1.675766e+03 -1.675775e+03| 0:0:01| chol  1  1
13|1.000|1.000|2.1e-11|1.0e-12|4.2e-03|-1.675770e+03 -1.675774e+03| 0:0:01| chol  1  1
14|0.951|0.932|6.2e-12|1.6e-12|2.3e-04|-1.675773e+03 -1.675773e+03| 0:0:01| chol  1  1
15|1.000|1.000|2.7e-11|1.2e-12|4.2e-05|-1.675773e+03 -1.675773e+03| 0:0:01|
  stop: max(relative gap, infeasibilities) < 1.49e-08
---------------------------------------------------------------------
 number of iterations   = 15
 primal objective value = -1.67577311e+03
 dual   objective value = -1.67577315e+03
 gap := trace(XZ)       = 4.24e-05
 relative gap           = 1.27e-08
 actual relative gap    = 1.26e-08
 rel. primal infeas (scaled problem)   = 2.67e-11
 rel. dual     "        "       "      = 1.23e-12
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 3.7e+01, 1.7e+01, 4.6e+01
 norm(A), norm(b), norm(C) = 1.0e+02, 1.0e+02, 1.3e+03
 Total CPU time (secs)  = 1.49
 CPU time per iteration = 0.10
 termination code       =  0
 DIMACS: 2.5e-10  0.0e+00  1.0e-11  0.0e+00  1.3e-08  1.3e-08
---------------------------------------------------------------------


------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +2617.83
```

Figure 6: Screenshot of the MATLAB command window for Q4

Solution to Q5 (Support Vector Classifiers):

The goal is to find a function $f(x) = a^T x - b$ that classifies the non-separable points $\{x_1, ..., x_N\}$ and $\{y_1, ..., y_M\}$ by doing a trade-off between the number of misclassifications and the width of the separating slab. a and b can be obtained by solving the following problem:

$$\begin{aligned} &\textit{minimize} \;\; \|a\|_2 + \gamma * (1^T u + 1^T v) \\ &\textit{subject to} \;\; a^T x_i - b \geq 1 - u_i \quad i = 1, \ldots, N \\ &\qquad\qquad a^T y_i - b \leq -(1 - v_i) \quad i = 1, \ldots, M \\ &\qquad\qquad u \succcurlyeq 0 \\ &\qquad\qquad v \succcurlyeq 0 \end{aligned}$$

where γ gives the relative weight of the number of misclassified points compared to the width of the slab.

We now vary value of γ form 0 to 2 with step size of 0.05 to find the optimal trade-off point.

Chosen Optimal trade-off point: $F1(\|a\|_2) : 0.80527$ and $F2(1^T u + 1^T v) : 13.618$ for $\gamma = 0.1$

Optimal Value: +2.1671

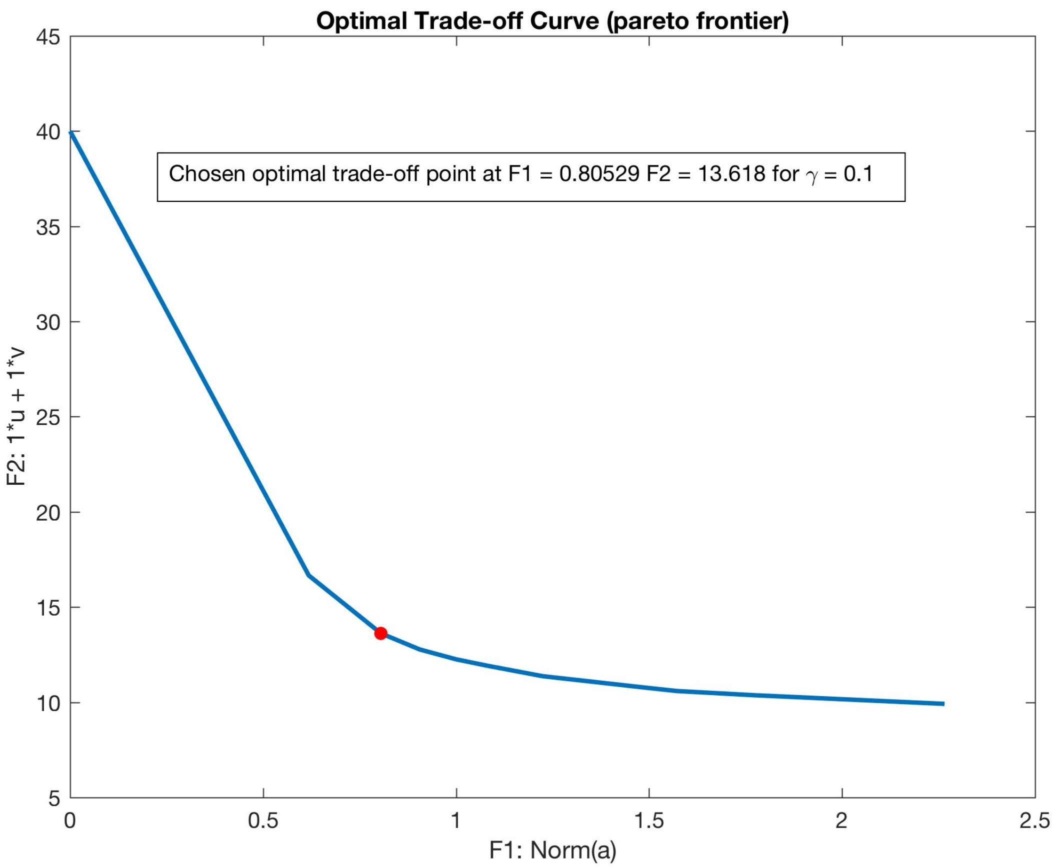


Figure 7: Plot of the Optimal trade-off curve (pareto frontier)

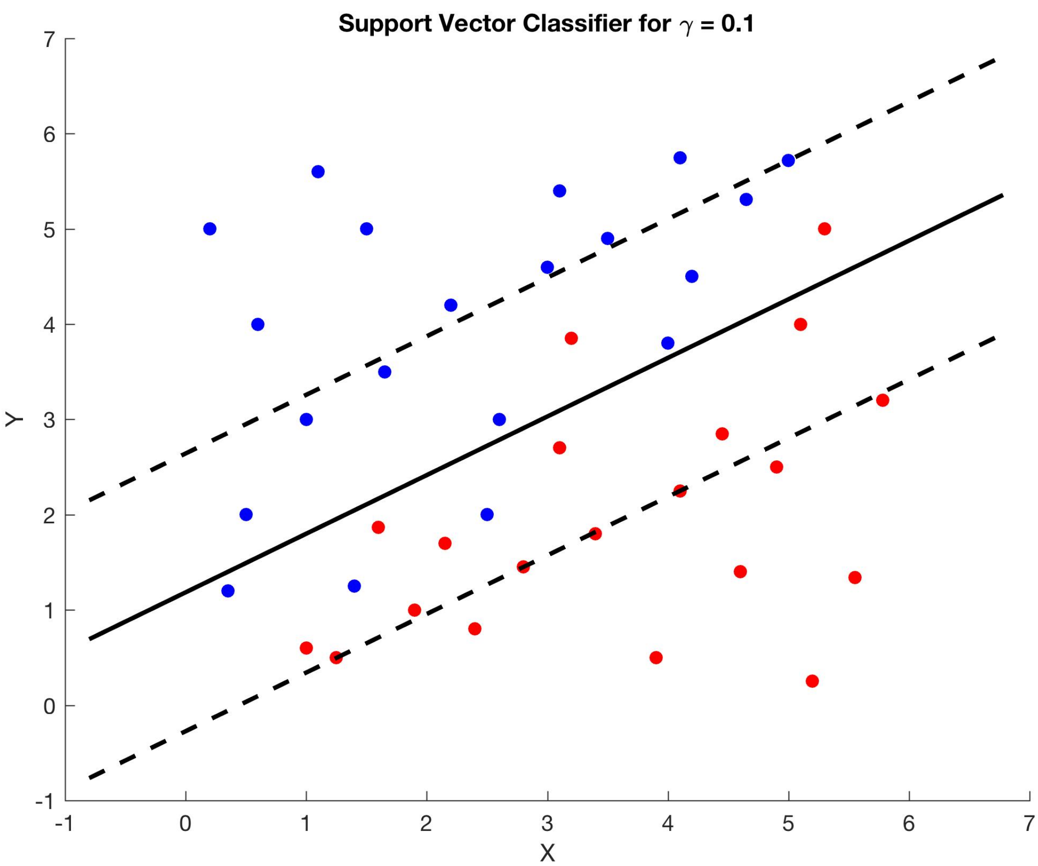


Figure 8: Plot showing linear discriminator for SVC corresponding to chosen optimal point

Code:

```
clc; clear all; close all;

%% Loading data from xlsx file

data_C1 = xlsread('Q5Data_Classification', 'Data1'); % Class 1 data
data_C2 = xlsread('Q5Data_Classification', 'Data2'); % Class 2 data

n = 2; % Dimension
N = length(data_C1); % Class 1: Number of data points
M = length(data_C2); % Class 2: Number of data points

%% Setting range of gamma's from 0 to 2 with step size of 0.05
g = [0:0.05:2];

F1 = zeros(length(g),1); % for storing norm(a) for all values of gamma
F2 = zeros(length(g),1); % for storing 1'*u + 1'*v for all values of gamma
optimal_values = zeros(length(g),1); % for storing optimal values for all values of
    gamma

%% Solution via CVX
for l = 1:length(g) % Running for all values of gamma
    cvx_begin
        variables a(n) b(1) u(N) v(M)
        minimize (norm(a) + g(l)*(ones(1,N)*u + ones(1,M)*v)) % Minimizing |a||_2 +
    gamma*(1'*u + 1'*v)
        data_C1*a - b >= 1 - u;
        data_C2*a - b <= -(1 - v);
        u >= 0;
        v >= 0;
    cvx_end
    F1(l) = norm(a); % Storing norm(a) for current gamma
    F2(l) = (ones(1,N)*u + ones(1,M)*v); % storing 1'*u + 1'*v for current gamma
    optimal_values(l) = cvx_optval; % storing optimal value for current gamma
end
```

```
[sorted,ind] = sort(optimal_values);

%% Plotting optimal trade-off curve (pareto frontier)
figure
plot(F1,F2,'LineWidth',2); % Plotting norm(a) vs. 1'*u + 1'*v
title('Optimal Trade-off Curve (pareto frontier)');
xlabel('F1: Norm(a)');
ylabel('F2: 1*u + 1*v');
hold on;
scatter(F1(3),F2(3),'r','filled'); % highlighting chosen optimal trade-off point
str = ['Chosen optimal trade-off point at F1 = ',num2str(F1(3)),' F2 = ',num2str(F2(3))
    ,' for \gamma = 0.1'];
dim = [0.2 0.4 0.4 0.4];
annotation('textbox',dim,'String',str,'FitBoxToText','on'); % Placing text box on plot
print('-djpeg','trade-off.jpg', '-r300'); % Saving image
close all;

%% Choosing g = 0.1 as optimal trade-off by observing from the above plot and finding
    optimal variables
g = 0.1;
cvx_begin
    variables a(n) b(1) u(N) v(M)
    minimize (norm(a) + g*(ones(1,N)*u + ones(1,M)*v))
    data_C1*a - b >= 1 - u;
    data_C2*a - b <= -(1 - v);
    u >= 0;
    v >= 0;
cvx_end

%% Displaying results
t_min = min([data_C1(:,1);data_C2(:,1)]); % Finding min of x-axis
t_max = max([data_C1(:,1);data_C2(:,1)]); % Finding max of x-axis
tt = linspace(t_min-1,t_max+1,100);
p = -a(1)*tt/a(2) + b/a(2); % Finding linear discriminator
p1 = -a(1)*tt/a(2) + (b+1)/a(2); % Finding linear slab for class 1
p2 = -a(1)*tt/a(2) + (b-1)/a(2); % Finding linear slab for class 2

figure
scatter(data_C1(:,1),data_C1(:,2), 'b', 'filled'); % Plotting class 1 data
hold on;
scatter(data_C2(:,1), data_C2(:,2), 'r','filled'); % Plotting class 2 data

plot(tt,p, '-k', tt,p1, '--k', tt,p2, '--k','LineWidth',2); % Plotting linear
    discriminator
title('Support Vector Classifier for \gamma = 0.1');
xlabel('X');
ylabel('Y');
print('-djpeg','SVM.jpg', '-r300'); % Saving image
close all;
```

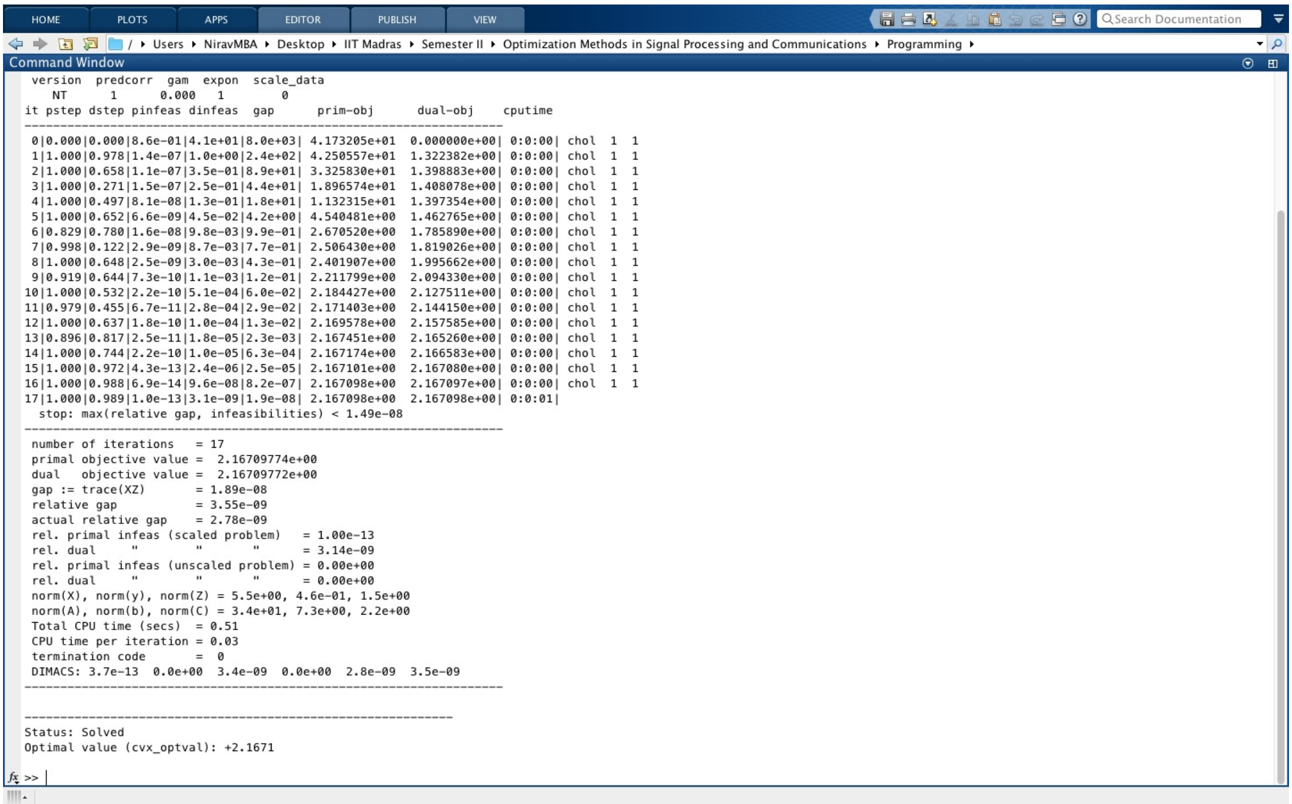


Figure 9: Screenshot of the MATLAB command window for Q5

References:

1. Boyd, Stephen, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

2. Boyd, Stephen, and Lieven Vandenberghe. Convex optimization. Solutions Manual.

3. Grant, Michael, Stephen Boyd, and Yinyu Ye. "The CVX users' guide." Stanford University, 2011 [2011-06-28]. (2009).

4. Examples from CVX Research

Extra

Solution to Q0:

```
cvx_begin
        variables x y u v z

% a)
          x+2*y==0;
          x-y==0;

% b)
        square_pos( square( x + y ) ) <= x - y
% OR
        variable t
        square( x+y ) <= t;
        square( t ) <= x - y
% OR
        ( x + y )^4 <= x - y;

% c)

        inv_pos(x) + inv_pos(y)  <= 1;

% d)
        norm( [ u ; v ] ) <= 3*x + y;
        max( x , 1 ) <= u;
        max( y , 2 ) <= v;

% e)
        x >= inv_pos(y);
        x >= 0;
        y >= 0;

% OR
        geomean([x,y])>=1
% OR
        [ x 1; 1 y ] == semidefinite(2)

% f)

        quad_over_lin(x + y , sqrt(y)) <= x - y + 5;

% g)

        pow_pos(x,3) + pow_pos(y,3) <= 1;
        x>=0;
        y>=0;

% h)
        x+z <= 1+geo_mean([x-quad_over_lin(z,y),y]);
        x>=0;
        y>=0;

cvx_end
```