

# Appendix

## MATLAB code for P1.8

P1.8: p8\_main.m

```
1 load('nfs/ug/homes-2/c/chenliy5/Desktop/Matrix Algebra/PS1/PS01_dataSet/wordVecV.mat')
2 dsize = size(V,2);
3 tsize = size(V,1);
4
5 %(a)-----
6 fprintf('a)\n');
7 min_distance(V);
8
9 %(b)-----
10 fprintf('b)\n');
11 sum_fterm = sum(V,1);
12 Vd_norm = V;
13
14 for i=1:dsize
15     Vd_norm(:,i) = V(:,i)/sum_fterm(i);
16 end
17 min_distance(Vd_norm);
18
19 %(c)-----
20 fprintf('c)\n');
21 fdot=zeros(1,tsize);
22 for row=1:tsize
23     for col=1:dsize
24         if(V(row,col) > 0)
25             fdot(row)= fdot(row) + 1;
26         end
27     end
28 end
29
30 Wd=V;
31 for row=1:tsize
32     for col=1:dsize
33         Wd(row,col) = (Vd_norm(row,col))*sqrt(log(dsize/fdot(row)));
34     end
35 end
36 min_distance(Wd);
```

P1.8: min\_distance.m

```
1 function [d1,d2,d3,d4] = min_distance(V)
2 angle=360;
3 distance=zeros(10,10);
4 d1=0;
5 d2=0;
6
7 %angle distance
8 for i=1:size(V,2)
9     Vd1 = V(:,i);
10    for j=1:size(V,2)
11        if i~=j
12            Vd2 = V(:,j);
13            angle_temp = abs(acos(dot(Vd1,Vd2)/(norm(Vd1)*norm(Vd2))));
14            if(angle_temp ~= 0 && angle>=angle_temp)
15                angle=angle_temp;
16                d1=i;
17                d2=j;
18            end
19        end
20    end
21 end
22
23 %Euclidean distance
24 for i=1:size(V,2)
25     Vd1 = V(:,i);
26     for j=1:size(V,2)
27         if i~=j
28             Vd2 = V(:,j);
29             distance_temp = norm(Vd1-Vd2);
30             distance(i,j) = distance_temp;
```

```

31         end
32     end
33 end
34
35 distanceCopy = (distance + diag(Inf(size(diag(distance))))); % put Inf on diagonal
36 [~,ndx] = min(distanceCopy); % get the linear index of the minimum value
37 misdistance=min(distanceCopy(:));
38 [d3,d4] = find(distanceCopy==misdistance);
39
40
41 fprintf('%dth and %dth are closest in Euclidean distance\n', d3(1), d3(2));
42 fprintf('The smallest Euclidean distance is %.4f\n', misdistance);
43 fprintf('%dth and %dth are closest in angle distance\n', d1, d2);
44 fprintf('The smallest angle distance is %.4f\n', angle);
45 end

```

## MATLAB code for P1.9

P1.9: p9\_main.m

```

1  close all; clc;
2
3  [x,y]=meshgrid(-2:0.2:3.5, -2:0.2:3.5);
4  f1 = 2*x+3*y+1;
5  f2 = x.^2+y.^2-x.*y-5;
6  f3 = (x-5)*cos(y-5)-(y-5)*sin(x-5);
7
8  figure(1)
9  plotContour(x,y,f1,1);
10 plotContour(x,y,f2,2);
11 plotContour(x,y,f3,3);
12
13 figure(2)
14 plotTangentPlane(x,y,f1,1);
15
16 plotTangentPlane(x,y,f2,2);
17
18 plotTangentPlane(x,y,f3,3);
19
20 function plotContour(x,y,f,subPlotInd)
21 subplot(1,3,subPlotInd);
22
23 contour(x,y,f,21);
24 hold on
25
26 [dx,dy] = gradient(f,.2,.2);
27 x0 = 1;
28 y0 = 0;
29 t = (x == x0) & (y == y0);
30 indt = find(t);
31 %f_grad2 = [dx(indt) dy(indt)];
32 gradX = linspace(1,1.5);
33 tangentX = linspace(0.5,1.5);
34
35 gradY = dy(indt)/dx(indt)*(gradX - 1);
36 tangentY = -dx(indt)/dy(indt)*(tangentX - 1);
37 plot(gradX,gradY);
38 hold on
39 plot(tangentX,tangentY);
40 axis equal
41 xlabel('x');
42 ylabel('y');
43 hold off;
44 end
45
46 function plotTangentPlane(x,y,f,subPlotInd)
47 subplot(1,3,subPlotInd);
48 surf(x,y,f)
49 hold on
50
51 [dx,dy] = gradient(f,.2,.2);
52 x0 = 1;
53 y0 = 0;
54 t = (x == x0) & (y == y0);
55 indt = find(t);

```

```

56 f_grad = [dx(indt) dy(indt)];
57 f_plane=f(indt)+f_grad(1)*(x-1)+f_grad(2)*y;
58 surf(x,y,f_plane);
59 xlabel('x');
60 ylabel('y');
61 zlabel('z');
62 end

```

## MATLAB code for P1.10

P1.10: p10\_main.m

```

1  close all; clc;
2
3  x=linspace(-2,2);
4  y=linspace(-1,1);
5  origin=[0;0];
6  x1=[3;-1];
7  b1=[-1;1];
8  v1=[1;2];
9
10 % (c) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11 figure(1)
12 % Draw arrow vector x1
13 drawArrow = @(j,k) quiver(j(1),j(1),k(1),k(2),0);
14 drawArrow(origin,x1);
15 text(2.5,-1,'x1');
16 hold on
17 % Draw the subspace V1
18 y=v1(2)/v1(1)*x;
19 l1=plot(x,y,'g—');
20 hold on
21 % Draw the affine set A1
22 A1=v1(2)/v1(1)*(x+1)+1;
23 l2=plot(x,A1,'b—');
24
25 legend([l1,l2], 'Subspace V1', 'Affine set A1');
26 hold on
27
28 % Draw projection on V1
29 proj_l1=1/5.*v1;
30 q=drawArrow(origin,proj_l1);
31 q.LineWidth=1;
32 q.DisplayName='projection';
33 text(0.2,0.2,'Proj_v_1(x1)');
34 % Draw projection on A1
35 proj_A1=b1;
36 q=drawArrow(origin,proj_A1);
37 q.LineWidth=1;
38 q.DisplayName='projection';
39 text(-1,1,'Proj-A_1(x1)');
40
41 xlim([-2 3])
42 ylim([-2 2])
43 xlabel('x');
44 ylabel('y');
45 axis equal
46 hold on
47
48
49 % (d) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50
51 x3=[3;0;-1;2;2];
52 b3=[-1;0;1;-2;1];
53 v1_d=[0;1;2;0;0];
54 v2_d=[1;3;0;1;0];
55 v3_d=[0;1;5;0;1];
56 subspace = [v1_d, v2_d, v3_d];
57
58 ortho_basis = gramshmidt(subspace);
59 v1_ortho = ortho_basis(:, 1);
60 v2_ortho = ortho_basis(:, 2);
61 v3_ortho = ortho_basis(:, 3);
62 disp('orthonormal basis')
63 disp(ortho_basis)

```

```

64
65 proj_x_onV = proj_v(x3, v1_ortho) + proj_v(x3, v2_ortho) + proj_v(x3, v3_ortho);
66 disp('projecting onto orthonormal basis')
67 disp(proj_x_onV)
68
69 proj_x_onA = proj_v(x3 - b3, v1_ortho) + proj_v(x3 - b3, v2_ortho) + proj_v(x3 - b3, v3_ortho)
    + b3;
70 disp('projecting onto orthonomral affine')
71 disp(proj_x_onA)

```

## MATLAB code for P1.11

### P1.11: p11\_main.m

```

1 function [ time_pos , sq_wave , B_unnorm ] = generate_data
2 close all; clc;
3 n_comps = 30;
4 period = 10;
5 fundFreq = 1/ period ;
6 time_pos = 0:0.0001:2* period ;
7 harmonics = 2*(1: n_comps ) -1;
8 sq_wave = floor (0.9* sin (2* pi* fundFreq * time_pos )) +.5; %% generate the signal
9 B_unnorm = sin (2* pi* fundFreq *( harmonics'* time_pos )) /2; %% generate the basis
10
11 figure(1)
12 plot(time_pos ,sq_wave);
13 xlabel('time ops');
14 ylabel('sq wave');
15
16 figure(2)
17 basis_1 = B_unnorm(1, :);
18 basis_2 = B_unnorm(2, :);
19 basis_3 = B_unnorm(3, :);
20 basis_4 = B_unnorm(4, :);
21 basis_5 = B_unnorm(5, :);
22 basis_6 = B_unnorm(6, :);
23 basis_30 = B_unnorm(30, :);
24
25 subplot(7, 1, 1)
26 plot(time_pos , basis_1);
27 xlabel('time ops');
28 ylabel('sq wave');
29 title('1st basis vector')
30 axis([0 20 -1 1]);
31 hold on
32
33 subplot(7, 1, 2)
34 plot(time_pos , basis_2);
35 xlabel('time ops');
36 ylabel('sq wave');
37 title('2nd basis vector')
38 axis([0 20 -1 1]);
39
40 subplot(7, 1, 3)
41 plot(time_pos , basis_3);
42 xlabel('time ops');
43 ylabel('sq wave');
44 title('3rd basis vectors')
45 axis([0 20 -1 1]);
46 hold on
47
48 subplot(7, 1, 4)
49 plot(time_pos , basis_4);
50 xlabel('time ops');
51 ylabel('sq wave');
52 title('4th basis vectors')
53 axis([0 20 -1 1]);
54 hold on
55
56 subplot(7, 1, 5)
57 plot(time_pos , basis_5);
58 xlabel('time ops');
59 ylabel('sq wave');
60 title('5th basis vectors')
61 axis([0 20 -1 1]);

```

```

62 hold on
63
64 subplot(7, 1, 6)
65 plot(time_pos, basis_6);
66 xlabel('time ops');
67 ylabel('sq wave');
68 title('6th basis vectors')
69 axis([0 20 -1 1]);
70 hold on
71
72 subplot(7, 1, 7)
73 plot(time_pos, basis_30);
74 xlabel('time ops');
75 ylabel('sq wave');
76 title('30th basis vectors')
77 axis([0 20 -1 1]);
78 hold on
79
80 figure(2)
81 norm_B_unnorm=B_unnorm;
82 for i=1:size(B_unnorm,1)
83 norm_B_unnorm(i,:)=B_unnorm(i,:)/norm(B_unnorm(i,:));
84 end
85
86 coeff = zeros(size(B_unnorm,1),1);
87 for i = 1:size(B_unnorm,1)
88     coeff(i,:) = dot(sq_wave,B_unnorm(i,:)/dot(B_unnorm(i,:),B_unnorm(i,:)));
89 end
90 [~,coeff_i]=sort(coeff);
91
92 approx = 0;
93 for i = 1:6
94     [B_ind, ~] = find(coeff_i==i);
95     approx = approx+ coeff(i, :) *B_unnorm(B_ind, :);
96 end
97 [B_ind, ~] = find(30);
98 approx = approx+ coeff(30, :) *B_unnorm(B_ind, :);
99
100 plot(time_pos, approx);
101 hold on
102 plot(time_pos, sq_wave);
103 xlabel('time op');
104 ylabel('sq wave');
105 title('Approximation');
106 end

```

## MATLAB code for P1.12

P1.12: p12\_main.m

```

1 close all; clc;
2
3 origin=[0 0]';
4 x = [3 2]';
5 v0 = [-2 -4]'; % offset
6 v = [-2 5]'; % slope
7 t=linspace(-3,3);
8
9 A=-v.*t+v0;
10
11
12 figure(1);
13
14 %l2 norm
15 xlabel('x');
16 ylabel('y');
17 plot(3,2,'o');
18 text(3.5,2,'x');
19 hold on
20 plot(A(1,:), A(2,:));
21 axis equal;
22 l2_norm=sqrt(dot(A-x,A-x));
23 [~,ind2]=min(l2_norm);
24 drawcircle(x(1, :), x(2, :), l2_norm(ind2))
25 plot(A(1,ind2),A(2,ind2),'o');

```

```

26 text(A(1,ind2)-2,A(2,ind2),'y-(-2-)');
27 axis equal;
28 axis([-15 15 -15 15])
29 title('l2 norm ball');
30
31 figure(2);
32 %l1 norm
33 subplot(1,2,1);
34 xlabel('x');
35 ylabel('y');
36 plot(3,2,'o');
37 text(3.5,2,'x');
38 hold on
39 plot(A(1,:), A(2,:));
40 axis equal;
41 l1_norm=sum(abs(A-x));
42 [~,ind1]=min(l1_norm);
43 drawdiamond(x(1, :), x(2, :), l1_norm(ind1))
44 plot(A(1,ind1),A(2,ind1),'o');
45 text(A(1,ind2)-2,A(2,ind2),'y-(-2-)');
46 axis equal;
47 axis([-15 15 -15 15])
48 title('l1 norm ball');
49
50 %linf norm
51 subplot(1,2,2);
52 xlabel('x');
53 ylabel('y');
54 plot(3,2,'o');
55 text(3.5,2,'x');
56 hold on
57 plot(A(1,:), A(2,:));
58 axis equal;
59 linf_norm=max(abs(A-x));
60 [~,ind_inf]=min(linf_norm);
61 drawsquare(x(1, :), x(2, :), linf_norm(ind_inf))
62 plot(A(1,ind_inf),A(2,ind_inf),'o');
63 text(A(1,ind2)-2,A(2,ind2),'y-(-2-)');
64 axis equal;
65 axis([-15 15 -15 15])
66 title('linf norm ball');
67
68 %(e)
69 [y1,r]=proj_cvx(x,v0,v,1);
70 [yinf,r]=proj_cvx(x,v0,v,inf);
71 y1
72 yinf
73
74 %
75 function drawcircle(x,y,r)
76 % (x,y) -center
77 %r - radius
78 th = 0:pi/50:2*pi;
79 xunit = r * cos(th) + x;
80 yunit = r * sin(th) + y;
81 plot(xunit, yunit);
82 end
83
84 function drawdiamond(x,y,r)
85 % (x,y) -center
86 %r - from center to vertices
87 ver=[x-r y;x y-r;x+r y;x y+r];
88 pgon=polyshape(ver);
89 plot(pgon);
90 end
91
92 function drawsquare(x,y,r)
93 % (x,y) -center
94 %r - from center to 4 sides
95 ver=[x-r y+r;x-r y-r;x+r y-r;x+r y+r];
96 pgon=polyshape(ver);
97 plot(pgon);
98 end
99
100 function [y, r] = proj_cvx (x, v0, v, nrm)%% x, v0 and v must be column vectors
101 objtv = @(y) norm (x-y, nrm); %% objective is L2 norm

```

```

102 cvx_begin
103 variable y(2) %% 2-d variable we are optimizing over
104 variable t(1) %% real valued parameter that defines
105 minimize ( objtv (y)) %% defining the objective
106 subject to
107 v0 + t*v == y %% the projection y must be in set A
108 cvx_end
109 r = objtv (y); %% minimum value of the objective
110 end

```

## MATLAB code for P1.13

P1.13: p13\_main.m

```

1
2 %Gram-Schmidt
3 syms x;
4 V=[x.^0;x.^1;x.^2;x.^3;x.^4;x.^5];
5 U=sin(x);
6
7 %(a)
8 Q=V;
9 for i=1:6
10     if(i>1)
11         for j=1:i-1
12             innerprod = innerprod_(Q(j),V(i));%int(V(i)*Q(j),-pi,pi);
13             Q(i,1)=Q(i,1)-innerprod*Q(j);
14         end
15     end
16     norm=sqrt(innerprod_(Q(i),Q(i)));
17     Q(i)=Q(i)/norm;
18 end
19
20
21 %(b)
22 A=zeros(6,1);
23 for i=1:6
24     A(i,1)=innerprod_(U,Q(i))/innerprod_(Q(i),Q(i));
25 end
26
27 f=0;
28 for i=1:6
29     f= f+A(i)*Q(i);
30 end
31
32 hold on
33 fplot(f,[-pi,pi])
34
35 hold on
36 fplot(U,[-pi,pi])
37
38 hold on
39 fplot(x-x^3/6+x^5/120, [-pi,pi])
40
41 legend('Orthonormal basis approximation','sinx','Taylor approximation')
42
43 function e = innerprod_(v-,q-)
44 e=int(v-*q,-pi,pi);
45 end

```