به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

درس سیستم‌های هوشمند

تمرین شماره ۱

نام و نام خانوادگی: محمدمهدی رحیمی

شماره دانشجویی: ۸۱۰۱۹۷۵۱۰

آبان ۱۴۰۰

فهرست سوالات

سوال ۱

در این سوال با دو روش مختلف به بهینه سازی می پردازیم. از نرم افزار Matlab نیز در حل آن استفاده کردیم.

الف:

راه حل به شکل زیر می باشد:

ابتدا گرادیان را محاسبه می کنیم و داریم

$$\nabla f = \begin{bmatrix} 6x_1 + 6x_2 + 12 \\ 16x_2 + 6x_1 + 8 \end{bmatrix}$$

نقطه ایستا را پیدا می کنیم:

$$\nabla f = 0 \Rightarrow (x_1, x_2) = \left(-\frac{12}{5}, \frac{2}{5}\right)$$

با توجه به کاهای اصلی می توان متوجه شد که ماتریس مثبت معین می باشد یعنی نقطه پیدا شده مینیمم محلی است

$$H = \begin{bmatrix} 6 & 6 \\ 6 & 16 \end{bmatrix}$$

برای مقادیر ویژه می توان از روش زیر نیز استفاده کرد

$$\det(H - \lambda I) = 0 \qquad \begin{vmatrix} 6-\lambda & 6 \\ 6 & 16-\lambda \end{vmatrix} = \lambda^2 - 22\lambda + 60 = 0 \Rightarrow \begin{cases} \lambda_1 = 11 + \sqrt{61} \\ \lambda_2 = 11 - \sqrt{61} \end{cases}$$

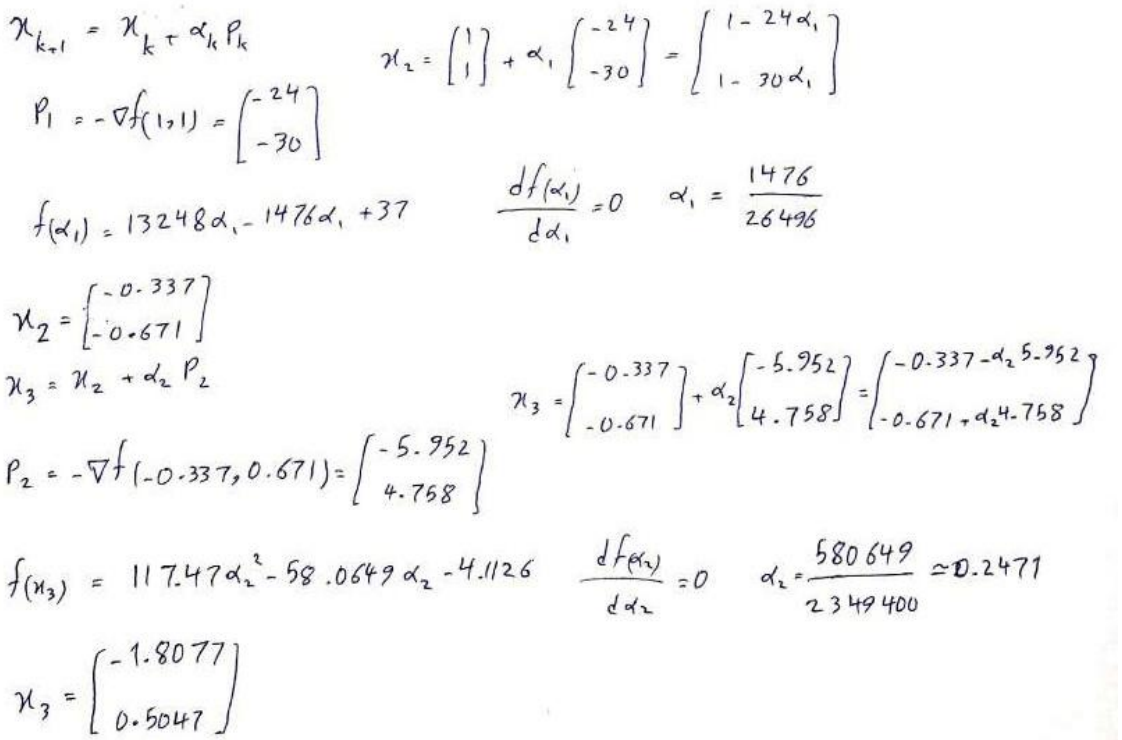
شکل ۱-۱ : راه حل محاسبه نقاط ایستا

همانطور که مشاهده می کنیداز دو روش نشان دادیم که ماتریس مثبت معین بوده. و نقطه ایستا را با به دست آوردن گرادیان و برابر صفر قرار دادن ان می توان به دست آورد که با توجه به مثبت معین بودن تابع مینیمم می باشد.

ب:

بخش تحلیل دستی:

در این روش با استفاده از روش تحلیل و استفاده از گرادیان و نقطه شروع بیان شده میزان بهینه را به دست می آوریم. و مطابق شکل داریم:



شکل ۱-۲: راه حل تحلیل قسمت ب سوال یک

هماطور که مشاهده می شود به نظر می رسد به سمت جواب همگرا می باشد.

شبیه ساز کامپیوتری:

در این بخش قطعه کد زیر را داریم:

```
syms f(x1,x2) g(x1,x2) ff(a) temp(a)
f(x1,x2) = 3*x1^2 + 12*x1 + 8*x2^2 + 8*x2 + 6*x1*x2;          %It is our main function
xx = [1;1];        %It is starting point
n = 30;        %Number of iteration
error = 0.0001;        %threshold for error to stop loop
precision = 6;        % precision of showing number in table which is show in command window
aa = zeros(1,n);          %array of alpha
p = zeros(2,n);        %direction of gradient
er = zeros(1,n);          %array of errors
g(x1,x2) = [diff(f,x1);diff(f,x2)];        %calculate gradient
p = -g(xx(1),xx(2));          %first direction for starting point
for i = 1:n
    ff(a) = f(xx(1,i)+a*p(1,i),xx(2,i)+a*p(2,i));
    temp(a) = diff(ff,a);
    eq = temp(a) == 0;
    aa(i) = solve(eq);
    xx(:,i+1) = xx(:,i)+aa(i)*p(:,i);
    p(:,i+1) = -g(xx(1,i+1),xx(2,i+1));
    er(i) = double(norm(p(:,i))); %this error find norm of gradient
```

```
    %er(i) = norm(xx(:,i+1)-xx(:,i));%this error find differnce between two x
    disp("  i       x1       x2      alpha      error")
    disp(vpa([i xx(1,i) xx(2,i) aa(i) er(i)], precision))
    if er(i) <= error
        disp(['With maximum error ' num2str(er(i)) ' in ' num2str(i) 'th iteration we have: X1 = '
num2str(xx(1,i)) ' X2 = ' num2str(xx(2,i))])
        break
    end
end
```

با اجرای قطعه کد بالا خروجی نمایش داده می شود که در آن در هر مرحله با محاسبه الفای جدید مقدار x1 وx2 جدید را محاسبه می کند. برای خطا دو تابع در نظر گرفته شده که یکی از آن ها اختلاف متغیر به دست آمده و متغیر قسمت قبل را محاسبه می کند و تابع خطای دیگر نرم بردار گرادیان را اندازه می گیرد. کد به شکلی زده شده است که می توان ابتدای کد تعداد دفعاتی که حلقه اجرا شود و یا دقت نمایش اعداد و میزان خطا برای قطع کردن حلقه را تنظیم کرد. و خروجی آن به شکل زیر خواهد بود:

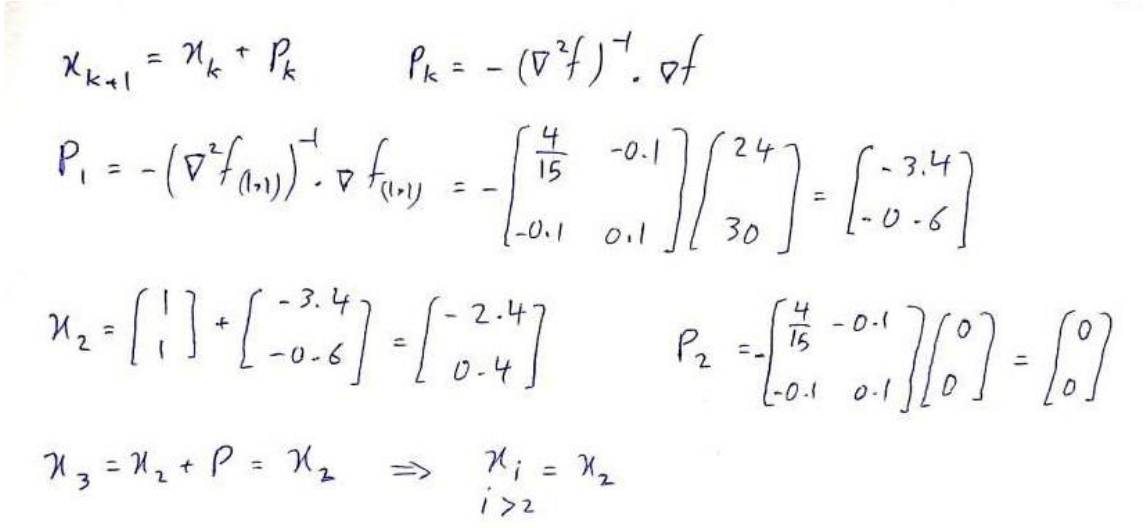
```
   i         x1         x2       alpha       error
[ 15.0, -2.39998, 0.400003, 0.0557065, 0.000189052]

   i         x1         x2       alpha       error
[ 16.0, -2.39999, 0.399995, 0.246988, 0.0000375022]

With maximum error 3.7502e-05 in 16th iteration we have: X1 = -2.4 X1 = 0.39999
```

شکل ۱-۳: خروجی کد

همانطور که مشاهده می شود قسمتی از خروجی کد می باشد و برای هر مرحله نمایش می دهد.

ج:

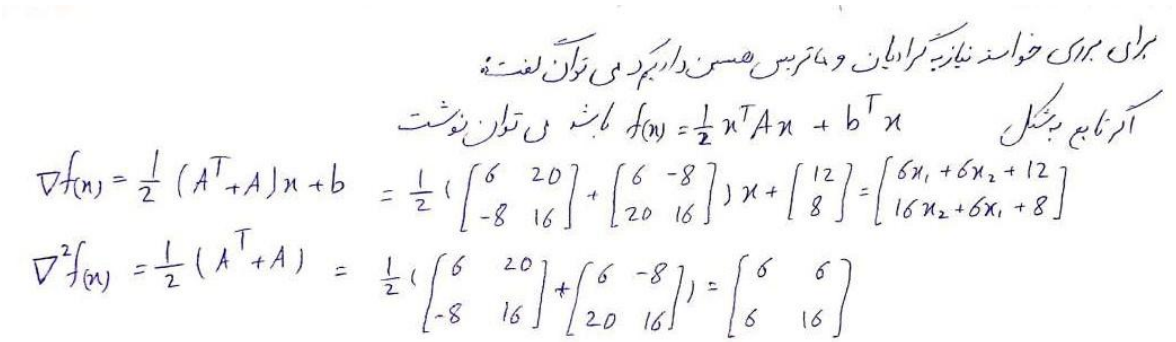
برای این قسمت محاسبات دستی زیر را داریم:



شکل ۱-۴: محاسبات دسته به روش نیوتن

همانطور که مشاهده شد خروجی با یک مرحله تکرار به میزان مطلوب رسید. یعنی این روش نیز کارآمد می باشد.

د:



شکل۱-۵: راه حل قسمت د

همانطور که مشاهده می شود گرادیان و ماتریس هسین محاسبه شده که دقیقا برابر با قسمت الف می باشد و یعنی همان نتایج را خواهند داشت و می توان گفت که تفاوتی ندارند.

سوال ۲

در این بخش از روش های پرادیان نزولی با طول پله ثابت و همچنین از روش تحلیلی و آرمیجو نیز استفاده کردیم . در انتها نیز از روش فرا ابتکاری تبرید شبیه سازی استفاده می کنیم.

الف:

برای این سوال قطعه کد زیر را داریم:

```
syms f(x1,x2) g(x1,x2)
f(x1,x2) = x1^2 - 10*x2*cos(0.2*pi*x1) + x2^2 - 15*x1*cos(0.4*pi*x2);   %It is our main function
initials = {[7;0] [9;6]};        %It is starting points
n = 1000;        %Number of iteration
error = 0.01;        %threshold for error to stop loop
precision = 6;         % precision of showing number in table which is show in command window
a = 0.005;          % alpha
r = zeros(2,2);
p = zeros(2,n);         %direction of gradient
er = zeros(1,n);           %array of errors
g(x1,x2) = [diff(f,x1);diff(f,x2)];         %calculate gradient
for j = 1:2
   xx = cell2mat(initials(j));
   p = -g(xx(1),xx(2));    %first direction for starting point
   for i = 1:n
      xx(:,i+1) = xx(:,i) + a*p(:,i);
      p(:,i+1) = -g(xx(1,i+1),xx(2,i+1));
      er(i) = double(norm(-p(:,i+1))); %this error find norm of gradient
      %er(i) = norm(xx(:,i+1)-xx(:,i));%this error find differnce between two x
      disp("  i     x1     x2     error")
      disp(vpa([i xx(1,i) xx(2,i) er(i)], precision))
      if er(i) <= error
         r(:,1) = xx(:,i+1);
         disp(['If initial point is x1 = ' num2str(xx(1,1)) ' and x2 = ' num2str(xx(2,1))])
         disp(['With maximum error ' num2str(er(i)) ' in ' num2str(i) 'th iteration we have: X1 = '
num2str(xx(1,i)) ' X1 = ' num2str(xx(2,i))])
         break
      end
   end
end
disp('optimization done')
for k = 1:2
   xx = cell2mat(initials(k));
  disp(['If initial point is x1 = ' num2str(xx(1,1)) ' and x2 = ' num2str(xx(2,1))])
  disp([' we have: X1 = ' num2str(r(1,k)) ' X2 = ' num2str(r(2,k))])
  disp(['Value of function is ' num2str(double(f(r(1,k),r(2,k)))) ])
end
```

کد در این قسمت به شکل بالا می باشد که می توان تعداد دفعات تکرار حلقه ، اندازه خطا ، مقدار های اولیه ، دقت نمایش اعداد و الفا را تغییر داد. انتهای خروجی به شکل زیر خواهد بود:

برای نقطه شروع اولی داریم:

```
If initial point is x1 = 7 and x2 = 0
With maximum error 0.0099467 in 504th iteration we have: X1 = 7.4944 X1 = -0.00019766
```

شکل۲-۱ : خروجی برای نقطه شروع اول

اگر ورودی ما ۷ و ۰ باشد در تکرار ۵۰۴ با خطای کمتر از ۰,۰۱ به مقدار بهینه می رسد.

برای نقطه شروع دوم داریم:

```
If initial point is x1 = 9 and x2 = 6
With maximum error 0.008965 in 67th iteration we have: X1 = 9.7688 X1 = 4.9995
```

شکل۲-۲: خروجی برای نقطه شروع دوم

و خروجی کلی به شکل زیر است:

```
optimization done
If initial point is x1 = 7 and x2 = 0
 we have: X1 = 7.4944 X2 = -0.00019766
Value of function is -56.25
If initial point is x1 = 9 and x2 = 6
 we have: X1 = 9.7688 X2 = 4.9995
Value of function is -75.5759
```

شکل۲-۳: خروجی کد اجرا شده

با توجه به کد می توان مقادیر اولیه و طول گام را تغییر داد که برای دو مقدار اولیه کد اجرا خواهد شد و در هر بار اجرای حلقه خطا و مقدار متغیر ها ذکر می شوند. با توجه به نکته ای که در دستور کار آمده است متوجه می شویم که ممکن است در کمینه های محلی قرار بگیریم و امکان یافتن کمینه کلی نباشد. همانطور که مشاهده می شود در حالت اول مقدار تابع ۵۶,۲۵- می باشد که این یک کمینه محلی می باشد و کد در همان کمینه محلی می ماند. اما برای نقطه شروع دیگر با توجه به مینیممی که در دستور کار ذکر شده است متوجه می شویم به اندازه بهینه خود رسیده است. می دانیم که نقاط کمینه اگر سطح تابع را رسم کنیم مانند فرورفتگی می باشند و اگر در فرایند بهینه سازی به سمت یک مینیمم محلی همگرا شویم نمی توان از آن خارج شد.

طول گام به حدی باید باشد تا روش همگرا باشد زیرا اگر طول گام بزرگ باشد واگرا شده و از کمینه محلی خارج می شود و برای نقاط اولیه متفاوت مقدار بیشینه طول گام ممکن می تواند متفاوت باشد که برای تعدادی از حالات باید کمتر از ۰,۱ باشد و مقدار آن را می توان با قانون Lipschitz به دست آورد.

ب:

روش تحلیلی:

در این قسمت با روش تحلیل که کد آن به شکل زیر می باشد مسئله را حل می کنیم:

```
syms f(x1,x2) g(x1,x2) ff(a) temp(a)
f(x1,x2) = x1^2 - 10*x2*cos(0.2*pi*x1) + x2^2 - 15*x1*cos(0.4*pi*x2);           %It is our main
function
initials = {[0;0] [1;1]};        %It is starting points
n = 100;        %Number of iteration
error = 0.01;       %threshold for error to stop loop
percision = 5;         %percision of showing number in table which is show in command window
aa = zeros(2,1);           %array of alpha
p = zeros(2,n);        %direction of gradient
er = zeros(1,n);           %array of errors
g(x1,x2) = [diff(f,x1);diff(f,x2)];        %calculate gradient          %first direction for starting point
 for j = 1:2
    xx = cell2mat(initials(j));
    p = -g(xx(1),xx(2));
    for i = 1:n
        ff(a) = f(xx(1,i)+a*p(1,i),xx(2,i)+a*p(2,i));
        temp(a) = diff(ff,a);
        eq = temp(a) == 0;
        aa(j,i) = vpasolve(eq);
        xx(:,i+1) = xx(:,i)+aa(j,i)*p(:,i);
        p(:,i+1) = -g(xx(1,i+1),xx(2,i+1));
        er(i) = double(norm(p(:,i))); %this error find norm of gradient
        %er(i) = norm(xx(:,i+1)-xx(:,i));%this error find differnce between two x
        disp("  i       x1       x2       alpha      error")
        disp(vpa([i xx(1,i) xx(2,i) aa(j,i) er(i)],percision))
        if er(i) <= error
            r(:,j) = xx(:,i+1);
            disp(['If initial point is x1 = ' num2str(xx(1,1)) ' and x2 = ' num2str(xx(2,1))])
            disp(['With maximum error ' num2str(er(i)) ' in ' num2str(i) 'th iteration we have: X1 = '
num2str(xx(1,i+1)) ' X2 = ' num2str(xx(2,i+1))])
            break
        end
    end
end
disp('optimization done')
for k = 1:2
    xx = cell2mat(initials(k));
   disp(['If initial point is x1 = ' num2str(xx(1,1)) ' and x2 = ' num2str(xx(2,1))])
   disp([' we have: X1 = ' num2str(r(1,k)) ' X2 = ' num2str(r(2,k))])
   disp(['Value of function is ' num2str(double(f(r(1,k),r(2,k)))) ])
end
for k = 1:2
    xx = cell2mat(initials(k));
    disp(['And array of alpha for initial point  x1 = ' num2str(xx(1,1)) ' and x2 = ' num2str(xx(2,1))])
    disp(aa(k,:))
end
```

در این کد نیز مانند قبل می توان دقت نمای اعداد ، میزان خطا ، تعداد حداقل تکرار و نقاط اولیه را مشخص کرد.

خروجی کد برای مقدار اولیه ۰ و ۰ به شکل زیر می باشد:

```
If initial point is x1 = 0 and x2 = 0
With maximum error 0.0031452 in 10th iteration we have: X1 = 0.4014 X2 = 1.1141
```

شکل۲-۴: خروجی کد برای نقطه شروع اول

که مقدار بهینه را در تکرار ۱۰ به ما میدهد.

برای مقدار اولیه ۱و ۱ داریم:

```
If initial point is x1 = 1 and x2 = 1
With maximum error 0.0061831 in 26th iteration we have: X1 = 7.4979 X2 = -0.00023932
```

شکل۲-۵ : خروجی کد برای نقطه شروع دوم

در این حالت نیز در تکرار ۲۶ به مقدار بهینه می رسیم.

جدول هر مرحله از عملیات نیز قابل مشاهده می باشد که در هر تکرار مقادیر چه تغییری کرده اند و خروجی نهایی به شکل زیر می باشد:

```
optimization done
If initial point is x1 = 0 and x2 = 0
 we have: X1 = 0.4014 X2 = 1.1141
Value of function is -10.4095
If initial point is x1 = 1 and x2 = 1
 we have: X1 = 7.4979 X2 = -0.00023932
Value of function is -56.25
```

شکل ۲-۶: خروجی نهایی کد

همانطور که مشاهده می کنید هرکدام از نقاط به یک مقدار کمینه محلی همگرا شدند.

و در انتها ارایه طول گام به دست آمده نمایش داده می شود:

```
And array of alpha for initial point  x1 = 1 and x2 = 1
  Columns 1 through 11

    0.0680   -0.0855    0.0413   -0.0699    0.0417   -0.0707    0.0416   -0.0706    0.0416   -0.0706
```

شکل ۲-۷ : آرایه طول گام برای مقدار اول

```
And array of alpha for initial point  x1 = 1 and x2 = 1
  Columns 1 through 11

    0.0515    0.5651    0.0065    0.2195    0.0061    0.2198    0.0059    0.2202    0.0058    0.2206    0.0057

  Columns 12 through 22

    0.2208    0.0057    0.2209    0.0057    0.2210    0.0057    0.2211    0.0057    0.2211    0.0057    0.2211

  Columns 23 through 26

    0.0056    0.2211    0.0056    0.2211
```

شکل۲-۸ : آرایه طول گام برای مقدار دوم

روش آرمیجو:

برای این قسمت قطعه کد زیر را داریم:

```
syms f(x1,x2) g(x1,x2) ff(a) temp(a)
f(x1,x2) = x1^2 - 10*x2*cos(0.2*pi*x1) + x2^2 - 15*x1*cos(0.4*pi*x2);           %It is our main function
initials = {[10;5] [0;0]};        %It is starting points
percision = 2;        %percision of showing number in table which is show in command window
max_iteration = 200;
alpha = zeros(2,1);          %array of alpha
alpha(1,1) = 2;
alpha(2,1) = 2;
beta = 0.05;
c1 = 0.5;
p = zeros(2,max_iteration);        %direction of gradient
g(x1,x2) = [diff(f,x1);diff(f,x2)];        %calculate gradient          %first direction for starting point
 for j = 1:2
    xx = cell2mat(initials(j));
    p = -g(xx(1),xx(2));
    i = 1;
    while f(xx(1,i)+alpha(j,i)*p(1,i),xx(2,i)+alpha(j,i)*p(2,i)) > f(xx(1,i),xx(2,i)) + c1 * alpha(j,i) * g(xx(1,i),xx(2,i))' * p(:,i) || i > max_iteration
        i = i+1;
        alpha(j,i) = beta * alpha(j,i-1);
        xx(:,i) = xx(:,i-1) +alpha(j,i)* p(:,i-1);
        p(:,i) = -g(xx(1,i),xx(2,i));
        disp("  i    x1    x2   alpha   f(x)")
        disp(vpa([i xx(1,i) xx(2,i) alpha(j,i) f(xx(1,i),xx(2,i))],percision))
    end
    r(:,j) = xx(:,i);
 end
disp('optimization done')
for k = 1:2
    xx = cell2mat(initials(k));
   disp(['If initial point is x1 = ' num2str(xx(1,1)) ' and x2 = ' num2str(xx(2,1))])
   disp([' we have: X1 = ' num2str(r(1,k)) ' X2 = ' num2str(r(2,k))])
   disp(['Value of function is ' num2str(double(f(r(1,k),r(2,k)))) ])
end
for k = 1:2
```

```
    xx = cell2mat(initials(k));
    disp(['And array of alpha for initial point  x1 = ' num2str(xx(1,1)) ' and x2 = ' num2str(xx(2,1))])
    disp(alpha(k,:))
end
```

با اجرای کد هر مرحله از محاسبات نمایش داده می شود. می توان مقادیر ثابت را عوض کرد تا روند محاسبات تغییر کند برای مثال بتا قابل تغییر می باشد . خروجی کد به شکل زیر می باشد:

```
optimization done
If initial point is x1 = 10 and x2 = 5
 we have: X1 = 9.5285 X2 = 4.9976
Value of function is -74.9576
If initial point is x1 = 0 and x2 = 0
 we have: X1 = 1.4828 X2 = 0.88494
Value of function is -12.1462
And array of alpha for initial point  x1 = 10 and x2 = 5
    2.0000    0.1000    0.0050

And array of alpha for initial point  x1 = 0 and x2 = 0
    2.0000    0.1000    0.0050
```

شکل ۲-۹: خروجی تابع

همانطور که مشاهده می کنید مقدار تابع و متغیر ها در حالت بهینه نمایش داده می شود و آرایه طول گام ها نیز نشان داده می شود.

ج:

در این بخش از روش تبرید شبیه سازی استفاده می کنیم و پارامتر های انتخابی به صورت تجربی می باشند.

و کد زیر را داریم:

```
syms f(x1,x2)
f(x1,x2) = x1^2 - 10*x2*cos(0.2*pi*x1) + x2^2 - 15*x1*cos(0.4*pi*x2);   %It is our main function
x = [0,0];
x0 = x;
ff = double(f(x(1),x(2)));
f0 = ff;
n = 100;
j = 0;
while j < n
  T = (j)/n*100;
  for i = 0:1000
     N = normrnd(x,T);
     xx = x + N;
     ftemp = double(f(xx(1),xx(2)));
```

```
      diff = ftemp - ff;
      if exp(-diff/T) > rand(1)
         x = xx;
         ff = ftemp;
      end
      if ftemp < f0
         xfinal = xx;
         f0 = ftemp;
      end
   end
    j = j + 1;
end
disp('optimization done')
disp(['If initial point is x1 = ' num2str(x(1)) ' and x2 = ' num2str(x(2))])
disp([' we have: X1 = ' num2str(xfinal(1)) ' X2 = ' num2str(xfinal(2))])
disp(['Value of function is ' num2str(f0) ])
```

و خروجی این کد به شکل زیر می باشد:

```
optimization done
If initial point is x1 = 0 and x2 = 0
 we have: X1 = 9.7173 X2 = 4.9263
Value of function is -74.9277
```

شکل ۲-۱۰: خروجی کد

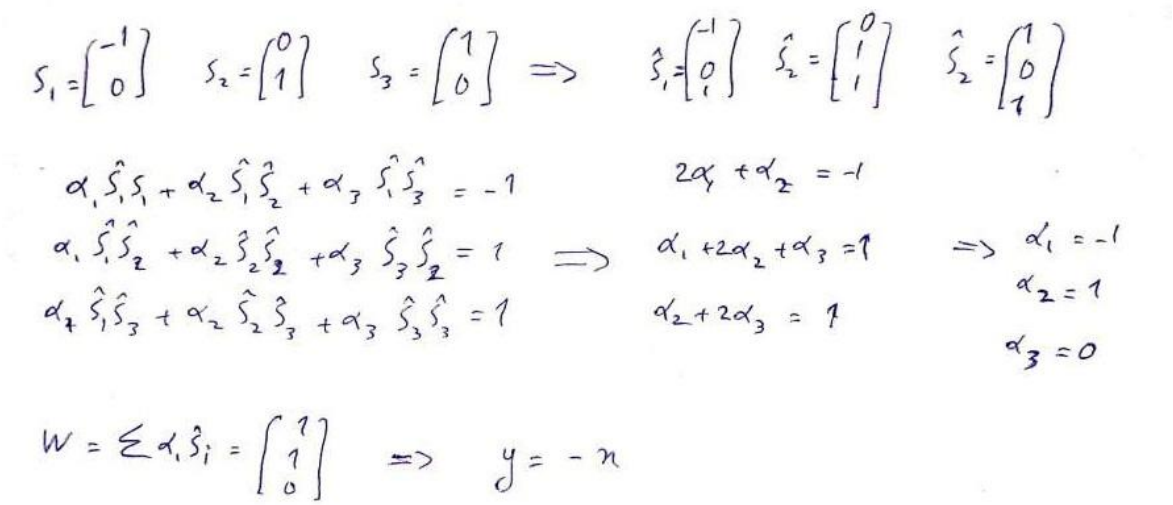
همانطور که مشاهده می کنید مقدار بهینه را خیلی خوب به دست اورده و مانند روش های قبلی در کمینه های محلی به دام نیوفتاده است.

سوال ۳

در قسمت اول این سوال از روابط ماشین بردار پشتیبان استفاده میکنیم.

الف:

برای این بخش به صورت دستی محاسبات زیر را انجام داده و داده ها را به دو قسمت تقسیم می کنیم:



شکل۳-۱: محاسبات انجام شده برای جدا سازی داده ها

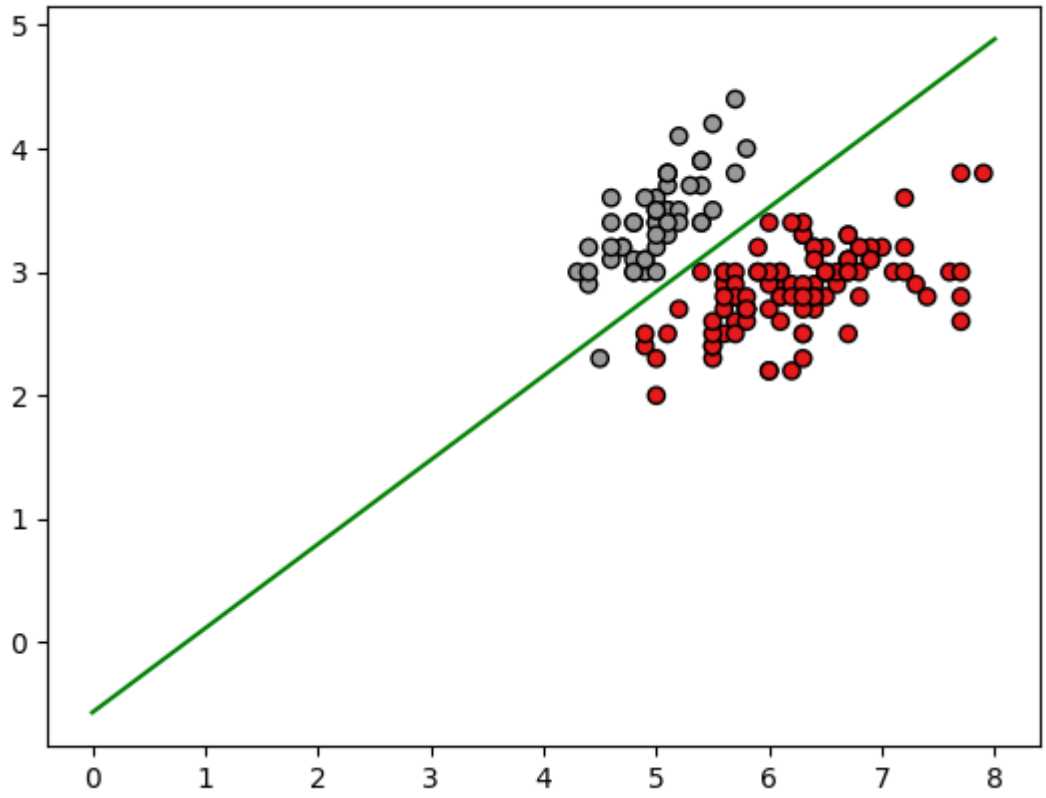
ب:

برای روش گرادیان نزول تصادفی خروجی زیر را داریم:

```
w is
[[-7.49616957]
 [10.99664954]]


b is
[6.23134623]
```

شکل ۳-۲ : خروجی برای حالت گرادیان نزول تصادفی

نمودار آن نیز به شکل زیر می باشد:



شکل ۳-۳ : شکل خروجی برای حالت گرادیان نزول تصادفی