گزارش پروژه دوم متلب

درس محاسبات عددی

امیرمرتضی رضائی

810101429

بهار 1402

سوال اول:

توضیح کلی مبحث حداقل مربعات:

در نظر بگیرید مجموعه ای از n داده دوتایی به فرم (x_i , y_i) داریم و مدلی که این داده ها را توصیف می کند، یک چند جمله ای درجه m است.

$$y = f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$$

در روش رگرسیون یا حداقل مربعات، ضرایب را به نحوی پیدا می کنیم که مجموع مربع اختلاف مقدار واقعی و چند جمله ای، بر روی تمام داده ها، حداقل شود.

$$S_r = \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_m x_i^m))^2$$

برای پیدا کردن ضرایبی که به ازاء آنها، تابع S_r حداقل شود، باید از این تابع نسبت به ضرایب مشتق گرفت و آن را مساوی صفر قرار داد. پس خواهیم داشت:

$$\frac{\partial S_r}{\partial a_0} = -2 \sum_{i=1}^{n} \left(y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_m x_i^m) \right) = 0$$

$$\frac{\partial S_r}{\partial a_1} = -2 \sum_{i=1}^{n} x_i \left(y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_m x_i^m) \right) = 0$$

$$\frac{\partial S_r}{\partial a_2} = -2 \sum_{i=1}^{n} x_i^2 \left(y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_m x_i^m) \right) = 0$$

$\ldots$

حال با ساده کردن این معادلات، به دستگاه معادلات خطی زیر می رسیم:

$$(n)a_0 \quad + \left(\sum x_i\right) a_1 + \left(\sum x_i^2\right) a_2 + \cdots = \sum y_i$$

$$\left(\sum x_i\right) a_0 + \left(\sum x_i^2\right) a_1 + \left(\sum x_i^3\right) a_2 + \cdots = \sum x_i y_i$$

$$\left(\sum x_i^2\right) a_0 + \left(\sum x_i^3\right) a_1 + \left(\sum x_i^4\right) a_2 + \cdots = \sum x_i^2 y_i$$

$$\cdots$$

در نهایت برای بدست آوردن ضرایب، کافیست این دستگاه معادلات خطی را حل کنیم.

الف)

توضیح مبحث تئوری:

اگر مجموعه ای از n داده دوتایی به فرم (x_i , y_i) داشته باشیم و بخواهیم آنها را با یک چند جمله ای درجه 1 (خطی) توصیف کنیم، خواهیم داشت:

$$f(x) = a_0 + a_1 x$$

با توجه به توضیحات ارائه شده، می توان نتیجه گرفت که در نهایت، دستگاه معادلات بدست آمده به صورت زیر خواهد بود:

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} \sum y_i \\ \sum x_i y_i \end{Bmatrix}$$

برای حل این دستگاه ، می توانیم از روش حذفی گاوس استفاده کنیم. می دانیم در این روش، برای مثال برای حل معادله ای مانند: AX=B ، باید ماتریس ضرایب و پاسخ ها را کنار هم قرار دهیم ([A:B]) و سپس با استفاده از عملیات سطری مقدماتی و با استفاده از عناصر روی قطر اصلی ، در هر ستون عناصر زیر قطر اصلی را به صفر تبدیل کنیم تا ماتریس ضرایب ما به یک ماتریس بالا مثلثی تبدیل شود.

حل سوال:

با توجه به توضیحات، مشخص می شود که ماتریس های A و B و X به صورت زیر خواهند بود:

$$A = \begin{bmatrix} n & \sum p \\ \sum p & \sum p^2 \end{bmatrix} \ , \ B = \begin{Bmatrix} \sum q \\ \sum pq \end{Bmatrix} \ , \ X = \begin{Bmatrix} a \\ b \end{Bmatrix}$$

ابتدا داده های سوال را در ماتریس به نام q_p وارد می کنیم که در آن، ستون اول مقادیر q و ستون دوم مقادیر p می باشند.

```
q_p=[1200,140;970,190;845,220;650,260;550,280;480,310;380,330;80,400;0,450];
```

حال باید برای تشکیل ماتریس ها، باید مقادیر n و $\sum p$ و $\sum p^2$ و $\sum q$ و $\sum pq$ را محاسبه کنیم. ابتدا تعداد داده ها را توسط تابع size به n نسبت میدهیم. سپس متغیر های sum_p، sum_p2، sum_q و sum_pq را ایجاد کرده و مقادیر اولیه آنها را برابر با صفر قرار می دهیم.

```
n=size(q_p,1);
sum_p=0;
sum_p2=0;
sum_q=0;
sum_pq=0;
```

اکنون در یک حلقه روی تمام سطر های ماتریس q_p حرکت کرده و در هر مرحله مقادیر $\sum p$ و $\sum p^2$ و $\sum q$ و $\sum pq$ را محاسبه کرده و آنها را در sum_p، sum_p2، sum_q و sum_pq ذخیره می کنیم. سپس ماتریس های A و B را به همان ترتیب گفته شده تشکیل می دهیم.

```
for i=(1:n)
    sum_p=sum_p + q_p(i,2);
    sum_p2=sum_p2 + (q_p(i,2)^2);
    sum_q=sum_q + q_p(i,1);
    sum_pq= sum_pq + (q_p(i,1)*q_p(i,2));
end
A=[n,sum_p;sum_p,sum_p2];
B=[sum_q;sum_pq];
```

حال برای حل دستگاه و یافتن درایه های ماتریس X (a و b) به روش حذفی گاوس، تابعی با نام gaussian_elimination ایجاد می کنیم که ماتریس های A و B ورودی های آن و ماتریس X خروجی آن باشد. در این تابع، باید ماتریس A را بالامثلثی گردانیم و سپس توسط روش حذفی گاوس، به یافتن ماتریس X بپردازیم.

ابتدا تعداد متغیر ها را توسط تابع size یافته و در متغیر n ذخیره می کنیم. سپس مقادیر A_new و B_new را برابر با مقادیر A و B قرار می دهیم. حال دو حلقه می زنیم که در اولی متغیر i از سطر اول تا سطر n-1 ام تغییر کرده و در حلقه ی درونی، متغیر j در هر مرحله از سطر i+1 ام تا سطر آخر (n ام) تغییر می کند. حال برای هر سطر عاملی به نام z را برابر با مقدار تقسیم درایه ای که قصد داریم آن را صفر کنیم بر درایه روی قطر اصلی در همان ستون قرار می دهیم. و ضرب z در تمام درایه های سطری که با درایه ی روی قطر اصلی آن می خواهیم درایه ی مورد نظر را صفر کنیم را از سطری که درایه مورد نظر در آن قرار دارد کم می کنیم. بدین ترتیب ماتریس A به ماتریسی بالا مثلثی تبدیل می گردد. همچنین در سطر پایانی نیز همین اعمال را بر روی ماتریس پاسخ های B انجام می دهیم.

```
function X = gaussian_elimination(A,B)
    n=size(A,1);
    A_new=A;
    B_new=B;
    for i=(1:n-1)
        for j=(i+1:n)
            z=A_new(j,i)/A_new(i,i);
            A_new(j,i:n) = A_new(j,i:n)-z*A_new(i,i:n);
            B_new(j) = B_new(j)-z*B_new(i);
        end
    end
```

اکنون باید پاسخ ماتریس مجهولات X را بیابیم. با توجه به اینکه ماتریس A بالا مثلثی شده است، می توان اظهار داشت که حاصل ضرب تنها درایه ی غیر صفر سطر n ام ماتریس A_new در درایه n ام ماتریس X برابر است با درایه ی سطر n ام ماتریس B_new. سپس برای بدست آوردن باقی ضرایب مجهول، در یک حلقه که متغیر i در آن از سطر یکی مانده به آخر یکی یکی به سطر اول باز می گردد، عنصر i ام ماتریس X برابر است با حاصل تفریق عنصر i ام ماتریس B_new با حاصل ضرب درایه ها ی سطر متناظر ماتریس A_new در درایه های بعدی ماتریس X تقسیم بر عنصر روی قطر اصلی سطر i ام ماتریس A_new.

```
    X(n) = B_new(n)/A_new(n,n);
    for i = n-1:-1:1
        X(i) = (B_new(i)-A_new(i,i+1:n)*X(i+1:n))/A_new(i,i);
    end
end
```

بدین ترتیب، مقادیر درایه های ماتریس X بدست آمده و سپس آنها را در متغیر های a و b ذخیره کرده و در نهایت نیز مقادیر آنها را چاپ می کنیم.

```
X = gaussian_elimination(A,B);
a=X(1);
b=X(2);
fprintf("a= %d , b= %d\n",a,b);
```

```
a= 1.713442e+03 , b= -3.979060e+00
```

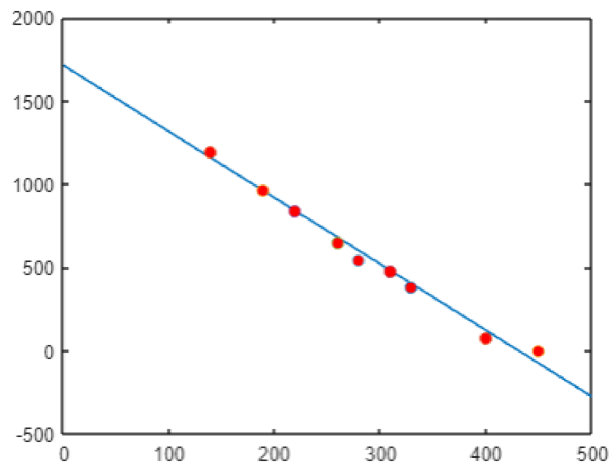
حال در بازه زمانی 0 تا 500، چند جمله ای q را طبق ضرایب بدست آمده و متغیر نمادین p، توسط تابع polyval ایجاد می کنیم. سپس نمودار q بر حسب p را رسم می کنیم.

```
syms p;
p = linspace(0,500);
q=polyval([b,a],p);
plot(p,q);
```

اکنون برای مشخص کردن نقاط تابع اصلی روی نمودار، در یک حلقه که به تعداد داده ها تکرار می شود، در هر مرحله ابتدا نمودار را نگه داشته و سپس نقطه ی متناظر روی آن سطر از ماتریس داده ها را با دایره های قرمز روی نمودار مشخص می کنیم.

```
for j=(1:n)
    hold on;
    plot(q_p(j,2),q_p(j,1), 'o','MarkerFaceColor','r');
end
```

در نهایت نمودار ترسیم شده به صورت زیر می باشد:



حال می خواهیم مجموع مجذورات خطا را محاسبه کرده و آن را در متغیری به نام sum_S ذخیره کنیم. بنابراین ابتدا متغیری با همین نام تعریف کرده و مقدار اولیه آن را برابر با صفر قرار می دهیم. سپس در یک حلقه روی سطر های ماتریس داده ها حرکت کرده و در هر مرحله برای هر سطر مقدار مربع خطا را محاسبه کرده و مقدار sum_S را آپدیت می کنیم. در انتها نیز آن را چاپ می کنیم.

```
sum_S=0;
for k=(1:n)
    sum_S=sum_S+( (q_p(k,1)-(a+b*q_p(k,2)) )^2);
end
fprintf("sum_S= %d\n", sum_S);
```

```
sum_S= 1.348802e+04
```

ب)

توضیح مبحث تئوری:

اگر مجموعه ای از n داده دوتایی به فرم (x_i , y_i) داشته باشیم و بخواهیم آنها را با یک چند جمله ای درجه 2 توصیف کنیم، خواهیم داشت:

$$f(x) = a_0 + a_1x + a_2x^2$$

با توجه به توضیحات ارائه شده، می توان نتیجه گرفت که در نهایت، دستگاه معادلات بدست آمده به صورت زیر خواهد بود:

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 \, y_i \end{Bmatrix}$$

برای حل این دستگاه ، می توانیم از روش حذفی گاوس استفاده کنیم. می دانیم در این روش، برای مثال برای حل معادله ای مانند: AX=B ، باید ماتریس ضرایب و پاسخ ها را کنار هم قرار دهیم ([A:B]) و سپس با استفاده از عملیات سطری مقدماتی و با استفاده از عناصر روی قطر اصلی ، در هر ستون عناصر زیر قطر اصلی را به صفر تبدیل کنیم تا ماتریس ضرایب ما به یک ماتریس بالا مثلثی تبدیل شود.

حل سوال:

با توجه به توضیحات، مشخص می شود که ماتریس های A و B و X به صورت زیر خواهند بود:

$$A = \begin{bmatrix} n & \sum p & \sum p^2 \\ \sum p & \sum p^2 & \sum p^3 \\ \sum p^2 & \sum p^3 & \sum p^4 \end{bmatrix} \;,\; B = \begin{Bmatrix} \sum q \\ \sum pq \\ \sum p^2 q \end{Bmatrix} \;,\; X = \begin{Bmatrix} a \\ b \\ c \end{Bmatrix}$$

ابتدا داده های سوال را در ماتریس به نام q_p وارد می کنیم که در آن، ستون اول مقادیر q و ستون دوم مقادیر p می باشند.

```
q_p=[1200,140;970,190;845,220;650,260;550,280;480,310;380,330;80,400;0,450];
```

حال باید برای تشکیل ماتریس ها، باید مقادیر n و $\sum p$ و $\sum p^2$ و $\sum p^3$ و $\sum p^4$ و $\sum q$ و $\sum pq$ و $\sum p^2 q$ را محاسبه کنیم. ابتدا تعداد داده ها را توسط تابع size به n نسبت می دهیم. سپس متغیر های sum_p، sum_p2، sum_p3، sum_p4، sum_q، sum_pq و sum_p2q را ایجاد کرده و مقادیر اولیه آنها را برابر با صفر قرار می دهیم.

```
n=size(q_p,1);
sum_p=0;
sum_p2=0;
sum_p3=0;
sum_p4=0;
sum_q=0;
sum_pq=0;
sum_p2q=0;
```

اکنون در یک حلقه روی تمام سطر های ماتریس q_p حرکت کرده و در هر مرحله مقادیر $\sum p$ و $\sum p^2$ و $\sum p^3$ و $\sum p^4$ و $\sum q$ و $\sum pq$ و $\sum p^2 q$ را محاسبه کرده و آنها را در sum_p، sum_p2، sum_p3، sum_p4، sum_q، sum_pq و sum_p2q ذخیره می کنیم. سپس ماتریس های A و B را به همان ترتیب گفته شده تشکیل می دهیم.

```
for i=(1:n)
    sum_p=sum_p + q_p(i,2);
    sum_p2=sum_p2 + (q_p(i,2)^2);
    sum_p3=sum_p3 + (q_p(i,2)^3);
    sum_p4=sum_p4 + (q_p(i,2)^4);
    sum_q=sum_q + q_p(i,1);
    sum_pq= sum_pq + (q_p(i,1)*q_p(i,2));
    sum_p2q= sum_p2q + (q_p(i,1)*q_p(i,2)*q_p(i,2));
end
A=[n,sum_p,sum_p2;sum_p,sum_p2,sum_p3;sum_p2,sum_p3,sum_p4];
B=[sum_q;sum_pq;sum_p2q];
```

حال برای حل دستگاه و یافتن درایه های ماتریس X (a و b) به روش حذفی گاوس، تابعی با نام gaussian_elimination ایجاد می کنیم که ماتریس های A و B ورودی های آن و ماتریس X خروجی آن باشد. در این تابع، باید ماتریس A را بالامثلثی گردانیم و سپس توسط روش حذفی گاوس، به یافتن ماتریس X بپردازیم.

ابتدا تعداد متغیر ها را توسط تابع size یافته و در متغیر n ذخیره می کنیم. سپس مقادیر A_new و B_new را برابر با مقادیر A و B قرار می دهیم. حال دو حلقه می زنیم که در اولی متغیر i از سطر اول تا سطر n-1 ام تغییر کرده و در حلقه ی درونی، متغیر j در هر مرحله از سطر i+1 ام تا سطر آخر (n ام) تغییر می کند. حال برای هر سطر عاملی به نام z را برابر با مقدار تقسیم درایه ای که قصد داریم آن را صفر کنیم بر درایه روی قطر اصلی در همان ستون قرار می دهیم. و ضرب z در تمام درایه های سطری که با درایه ی روی قطر اصلی آن می خواهیم درایه ی مورد نظر را صفر کنیم را از سطری که درایه مورد نظر در آن قرار دارد کم می کنیم. بدین ترتیب ماتریس A به ماتریسی بالا مثلثی تبدیل می گردد. همچنین در سطر پایانی نیز همین اعمال را بر روی ماتریس پاسخ های B انجام می دهیم.

```
function X = gaussian_elimination(A,B)
    n=size(A,1);
    A_new=A;
    B_new=B;
    for i=(1:n-1)
        for j=(i+1:n)
            z=A_new(j,i)/A_new(i,i);
            A_new(j,i:n) = A_new(j,i:n)-z*A_new(i,i:n);
            B_new(j) = B_new(j)-z*B_new(i);
        end
    end
```

اکنون باید پاسخ ماتریس مجهولات X را بیابیم. با توجه به اینکه ماتریس A بالا مثلثی شده است، می توان اظهار داشت که حاصل ضرب تنها درایه ی غیر صفر سطر n ام ماتریس A_new در درایه n ام ماتریس X برابر است با درایه ی سطر n ام ماتریس B_new. سپس برای بدست آوردن باقی ضرایب مجهول، در یک حلقه که متغیر i در آن از سطر یکی مانده به آخر یکی یکی به سطر اول باز می گردد، عنصر i ام ماتریس X برابر است با حاصل تفریق عنصر i ام ماتریس B_new با حاصل ضرب درایه ها ی سطر متناظر ماتریس A_new در درایه های بعدی ماتریس X تقسیم بر عنصر روی قطر اصلی سطر i ام ماتریس A_new.

```
    X=zeros(n,1);
    X(n) = B_new(n)/A_new(n,n);
    for i = n-1:-1:1
        X(i) = (B_new(i)-A_new(i,i+1:n)*X(i+1:n))/A_new(i,i);
    end
end
```

بدین ترتیب، مقادیر درایه های ماتریس X بدست آمده و سپس آنها را در متغیر های a و b و c ذخیره کرده و در نهایت نیز مقادیر آنها را چاپ می کنیم.

```
X = gaussian_elimination(A,B);
a=X(1);
b=X(2);
c=X(3);
fprintf("a= %d , b= %d , c= %d\n",a,b,c);
```

```
a= 1.984111e+03 , b= -6.007007e+00 , c= 3.419859e-03
```

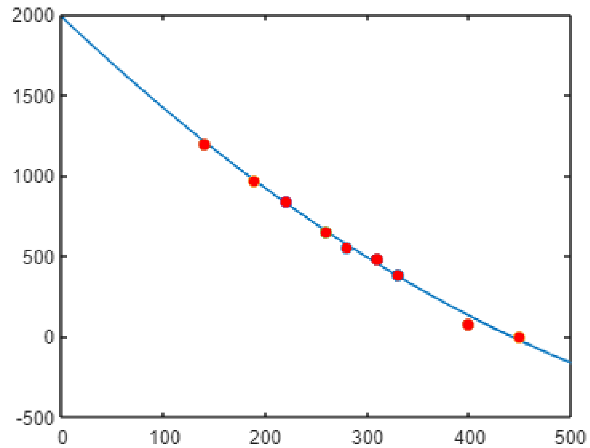
حال در بازه زمانی 0 تا 500، چند جمله ای q را طبق ضرایب بدست آمده و متغیر نمادین p، توسط تابع polyval ایجاد می کنیم. سپس نمودار q بر حسب p را رسم می کنیم.

```
syms p;
p = linspace(0,500);
q=polyval([c,b,a],p);
plot(p,q);
```

اکنون برای مشخص کردن نقاط تابع اصلی روی نمودار، در یک حلقه که به تعداد داده ها تکرار می شود، در هر مرحله ابتدا نمودار را نگه داشته و سپس نقطه ی متناظر روی آن سطر از ماتریس داده ها را با دایره های قرمز روی نمودار مشخص می کنیم.

```
for j=(1:n)
    hold on;
    plot(q_p(j,2),q_p(j,1), 'o','MarkerFaceColor','r');
end
```

در نهایت نمودار ترسیم شده به صورت زیر می باشد:



حال می خواهیم مجموع مجذورات خطا را محاسبه کرده و آن را در متغیری به نام sum_S ذخیره کنیم. بنابراین ابتدا متغیری با همین نام تعریف کرده و مقدار اولیه آن را برابر با صفر قرار می دهیم. سپس در یک حلقه روی سطر های ماتریس داده ها حرکت کرده و در هر مرحله برای هر سطر مقدار مربع خطا را محاسبه کرده و مقدار sum_S را آپدیت می کنیم. در انتها نیز آن را چاپ می کنیم.

```
sum_S=0;
for k=(1:n)
    sum_S=sum_S+( (q_p(k,1)-( a+b*q_p(k,2)+c*(q_p(k,2)^2) ) )^2);
end
fprintf("sum_S= %d\n", sum_S);
```

```
sum_S= 4.778769e+03
```

ج)

در ابتدا همانند قسمت های قبلی، ماتریس مقادیر ورودی را تعریف کرده و سپس دقیقا مشابه آنچه در دو قسمت پیشین انجام دادیم، متغیر هایی را برای بدست آوردن درایه های ماتریس ها تعریف کرده و سپس در یک حلقه مقدار آنها را می یابیم.

```
q_p=[1200,140;970,190;845,220;650,260;550,280;480,310;380,330;80,400;0,450];
n=size(q_p,1);
sum_p=0;
sum_p2=0;
sum_p3=0;
sum_p4=0;
sum_q=0;
sum_pq=0;
sum_p2q=0;
for i=(1:n)
    sum_p=sum_p + q_p(i,2);
    sum_p2=sum_p2 + (q_p(i,2)^2);
    sum_p3=sum_p3 + (q_p(i,2)^3);
    sum_p4=sum_p4 + (q_p(i,2)^4);
    sum_q=sum_q + q_p(i,1);
    sum_pq= sum_pq + (q_p(i,1)*q_p(i,2));
    sum_p2q= sum_p2q + (q_p(i,1)*q_p(i,2)*q_p(i,2));
end
```

اکنون ماتریس های A_1 و B_1 و X_1 را برای چندجمله ای درجه 1 اختصاص داده و همانند قسمت الف، مقادیر دو ماتریس اول را وارد کرده و سپس از روی آنها و توسط تابع gaussian_elimination که شرح آن به تفضیل در قسمت های قبل آورده شده، مقادیر ماتریس X_1 و از روی آن مقادیر a_1 و b_1 را که همان ضرایب چندجمله ای درجه 1 هستند، بدست می آوریم. اکنون همین اعمال را برای چند جمله ای درجه 2 نیز روی ماتریس های A_2 و B_2 و X_2 انجام داده و مقادیر ضرایب چندجمله ای درجه 2 که a_1 و b_1 و c_1 هستند، بدست می آوریم.

```
A_1=[n,sum_p;sum_p,sum_p2];
B_1=[sum_q;sum_pq];
X_1 = gaussian_elimination(A_1,B_1);
a_1=X_1(1);
b_1=X_1(2);
A_2=[n,sum_p,sum_p2;sum_p,sum_p2,sum_p3;sum_p2,sum_p3,sum_p4];
B_2=[sum_q;sum_pq;sum_p2q];
X_2 = gaussian_elimination(A_2,B_2);
a_2=X_2(1);
b_2=X_2(2);
c_2=X_2(3);
```

اکنون می خواهیم مقادیری از جدول را بدست بیاوریم که به ازاء آنها ، خطای چندجمله ای ها کمتر از 10 کیلوگرم باشد.

ابتدا از چندجمله ای درجه 1 آغاز می کنیم. در یک حلقه که از خانه اول تا خانه پایانی ماتریس ورودی را می پیماید، با یک شرط بررسی می کنیم که اگر قدر مطلق اختلاف حاصل چندجمله ای با مقدار واقعی کمتر از 10باشد، آن سطر از جدول را چاپ کند.

```
fprintf("****DARAJEH 1****\n");
for k=(1:n)
    if abs((a_1+b_1*q_p(k,2))-q_p(k,1))<10
        fprintf("q=%d  , p=%d\n",q_p(k,1), q_p(k,2));
    end
end
```

در انتها برای چندجمله ای درجه 2 نیز همین اقدامات را انجام می دهیم.

```
fprintf("****DARAJEH 2****\n");
for k=(1:n)
    if abs((a_2+b_2*q_p(k,2)+c_2*(q_p(k,2)^2))-q_p(k,1))<10
        fprintf("q=%d  , p=%d\n",q_p(k,1), q_p(k,2));
    end
end
```

نتایج بدست آمده به شرح زیر می باشد:

```
****DARAJEH 1****

q=845   , p=220
q=480   , p=310


****DARAJEH 2****

q=970   , p=190
q=650   , p=260
q=380   , p=330
```

سوال دوم:

توضیح مبحث تئوری:

برای مشتق گیری عددی از یک تابع بدون داشتن ضابطه ی آن که تنها برخی نقاط و مقادیر آنها در تابع را در اختیار داریم، می توان با استفاده از بسط تیلور روابطی را بدست آورد.

برای مثال اگر بخواهیم مشتق تابع را در نقطه ی x_i بدست آوریم و مقادیر تابع در نقاط پیشین و پسین یعنی x_{i-1} و x_{i+1} را داشته باشیم می توانیم از رابطه ی زیر استفاده کنیم که دقت آن از مرتبه h^2 می باشد.(h، فاصله دو نقطه ی متوالی می باشد) :

$$\begin{cases} f_{i+1} = f_i + hf_i' + \dfrac{h^2}{2}f_i'' + O(h^3) \\ f_{i-1} = f_i - hf_i' + \dfrac{h^2}{2}f_i'' + O(h^3) \end{cases} \Rightarrow \quad f_i' = \frac{f_{i+1} - f_{i-1}}{2h} + O(h^2) \quad (1)$$

همچنین اگر مقادیر تابع در نقاط پسین موجود نبوده و تنها مقادیر آنرا در نقاط پیشین در دست داشته باشیم، می توانیم از روابط زیر استفاده کنیم. باز مشاهده می شود که دقت آن از مرتبه h^2 خواهد بود :

$$\begin{cases} f_{i-1} = f_i - hf_i' + \dfrac{h^2}{2}f_i'' + O(h^3) \\ f_{i-2} = f_i - 2hf_i' + \dfrac{(2h)^2}{2}f_i'' + O(h^3) \end{cases} \Rightarrow \quad f_i' = \frac{f_{i-2} - 4f_{i-1} + 3f_i}{2h} + O(h^2) \quad (2)$$

حل سوال:

می دانیم سرعت برابر است با مشتق مکان. بنابراین برای بدست آوردن سرعت ربات در لحظات مختلف، کافیست تا مشتق تابع مکان آن را در لحظات دلخواه بدست آوریم. از آنجا که برای زمان های 0.1 تا 0.9 ثانیه، مقادیر توابع مکان در هر دو نقطه پسین و پیشین موجود است، از رابطه ی (1) و برای زمان 1 ثانیه، چون مقادیر توابع در لحظات بعدی مشخص نیست، از رابطه ی (2) استفاده می کنیم.

همچنین چون مکان ربات در هر لحظه ، در دو راستای x و y موجود است، باید سرعت ربات را در هر لحظه در هریک از این جهات بدست آوریم.

بنابراین ابتدا تابعی به نام centered_derivative ایجاد می کنیم که با گرفتن مقادیر تابع در نقطه‌ی بعدی (f_after) و نقطه‌ی قبلی (f_before) و طول هر گام (h) ، مقدار مشتق تابع در نقطه‌ی مرکزی را طبق رابطه‌ی (1) بدست آورده و مقدار بدست آمده را در متغیری به نام v_centered ذخیره نماید.

```
function v_centered= centered_derivative(f_after,f_before,h)
    v_centered=(f_after-f_before)/(2*h);
end
```

حال تابعی دیگر به نام backward_derivative ایجاد می کنیم که با گرفتن مقادیر تابع در نقطه‌ی‌فعلی (f) و نقطه‌ی قبلی (f_before1) و نقطه‌ی پیش از آن (f_before2) و طول هر گام (h) ، مقدار مشتق تابع در نقطه فعلی را طبق رابطه‌ی (2) بدست آورده و مقدار بدست آمده را در متغیری به نام v_backward ذخیره نماید.

```
function v_backward= backward_derivative(f,f_before1,f_before2,h)
    v_backward=(3*f - 4*f_before1 + f_before2)/(2*h);
end
```

اکنون برای حل سوال، ابتدا مقادیر داده های سوال را در سه ماتریس t و x و y وارد کرده و با استفاده از تابع horzcat، آنها را در کنار هم قرار داده و در ماتریسی به نام data ذخیره می کنیم.

```
t=[0;0.1;0.2;0.3;0.4;0.5;0.6;0.7;0.8;0.9;1];
x=[0.500;0.564;0.678;0.814;0.943;1.038;1.069;1.009;0.830;0.503;0.000];
y=[1.500;1.536;1.574;1.598;1.591;1.538;1.421;1.225;0.934;0.531;0.000];
data=horzcat(t,x,y);
```

سپس طول هر گام را با محاسبه اختلاف دو نقطه‌ی متوالی دلخواه، بدست آورده و در متغیر h ذخیره می کنیم.

```
h=data(2,1)-data(1,1);
```

حال از آنجا که سرعت ربات در 10 زمان مختلف خواسته شده، ماتریسی به نام result با 10 سطر و 3 ستون و مقدار اولیه 0 ایجاد میکنیم. ستون اول برای ذخیره زمان، ستون دوم برای ذخیره سرعت متناظر در راستای x ، و ستون سوم برای ذخیره سرعت متناظر در راستای y می باشد.

```
result=zeros(10, 3);
```

همانطور که پیشتر اشاره شد، برای بدست آوردن سرعت ربات در زمان های 0.1 تا 0.9 ثانیه، باید از رابطه‌ی اول و در نتیجه تابع centered_derivative و برای زمان 1 ثانیه از رابطه ی دوم و در نتیجه، تابع backward_derivative استفاده کنیم. بنابراین ابتدا در یک حلقه از سطر دوم (زمان 0.1) تا سطر دهم (زمان0.9) ماتریس data حرکت کرده و در هر مرحله ابتدا زمان هر سطر را در درایه ی اول هر سطر ماتریس result ذخیره می کنیم. سپس در ستون دوم، مقدار سرعت در راستای x را توسط تابع centered_derivative و با ورودی دادن مقادیر سطر های قبلی و بعدی و طول هر گام، محاسبه کرده و ذخیره می کنیم. سپس همین کار را در ستون سوم و برای بدست آوردن سرعت در آن لحظه در راستای y انجام می دهیم.

توجه شود از آنجا که به سرعت در لحظه ی 0 نیازی نداریم، مقدار حلقه از 2 شروع می شود. ولی چون میخواهیم ماتریس result را از همان سطر اول تکمیل کنیم، اندیس درایه های ماتریس result برابر با i-1 میباشد.

```
for i=(2:10)
    result(i-1,1)=data(i,1);
    result(i-1,2)=centered_derivative(data(i+1,2),data(i-1,2),h);
    result(i-1,3)=centered_derivative(data(i+1,3),data(i-1,3),h);
end
```

حال برای سطر آخر و زمان 1 ثانیه، ابتدا ستون اول سطر دهم ماتریس result را برابر با مقدار ستون اول سطر یازدهم ماتریس data قرار می دهیم (که برابر با 1 می باشد!). سپس برای ستون دوم و سرعت در راستای x، از تابع bacward_derivative استفاده کرده و مقادیر ستون دوم سطر های 11 و 10 و 9 و مقدار h را به عنوان ورودی به آن می دهیم. در نهایت نیز برای سرعت در راستای y، همین کار را برای ستون سوم آنها انجام می دهیم.

```
result(10,1)=data(11,1);
result(10,2)=backward_derivative(data(11,2),data(10,2),data(9,2),h);
result(10,3)=backward_derivative(data(11,3),data(10,3),data(9,3),h);
```

در پایان برای نمایش مقادیر بدست آمده از یک حلقه استفاده می کنیم که از سطر اول تا سطر دهم ماتریس result را پیموده و مقادیر هر ستون در هر سطر را به صورت زیر چاپ می کند.

```
for j=(1:10)
    fprintf("t=%d , v_x=%d , v_y=%d\n",result(j,1),result(j,2),result(j,3));
end
```

مقادیر بدست آمده بدین صورت می باشد:

```
t=1.000000e-01 , v_x=8.900000e-01 , v_y=3.700000e-01
t=2.000000e-01 , v_x=1.250000e+00 , v_y=3.100000e-01
t=3.000000e-01 , v_x=1.325000e+00 , v_y=8.500000e-02
t=4.000000e-01 , v_x=1.120000e+00 , v_y=-3.000000e-01
t=5.000000e-01 , v_x=6.300000e-01 , v_y=-8.500000e-01
t=6.000000e-01 , v_x=-1.450000e-01 , v_y=-1.565000e+00
t=7.000000e-01 , v_x=-1.195000e+00 , v_y=-2.435000e+00
t=8.000000e-01 , v_x=-2.530000e+00 , v_y=-3.470000e+00
t=9.000000e-01 , v_x=-4.150000e+00 , v_y=-4.670000e+00
t=1 , v_x=-5.910000e+00 , v_y=-5.950000e+00
```