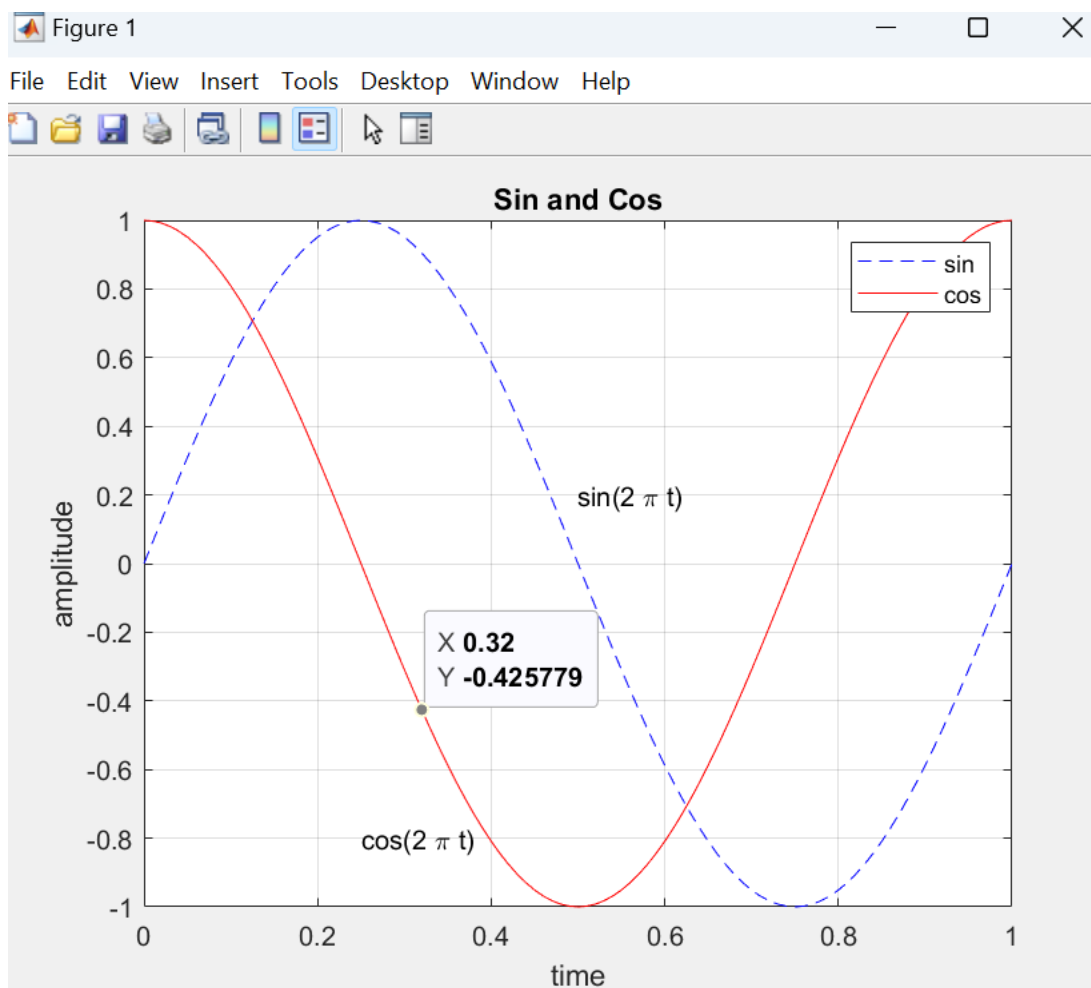
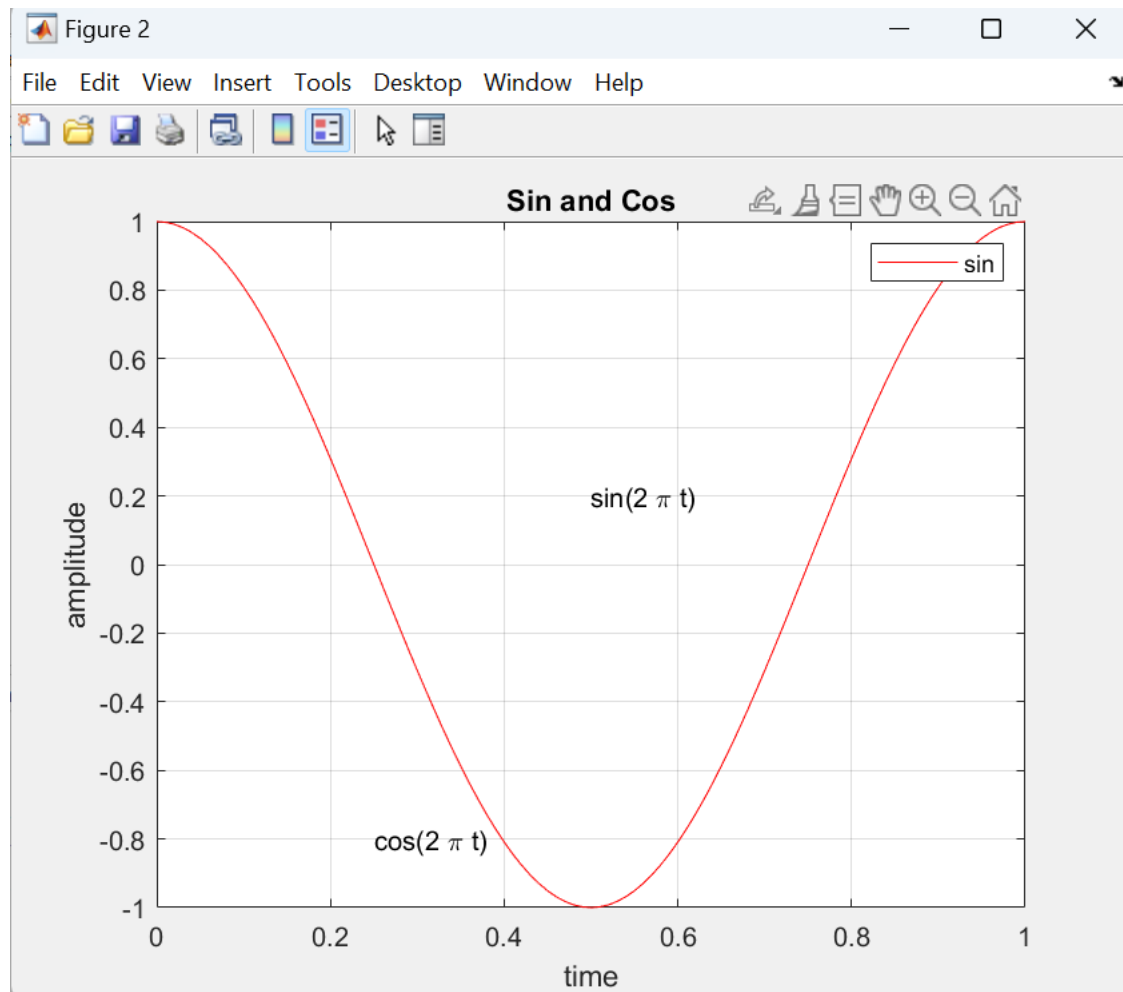


(۱-۱) برنامه به طور کلی نموداری از سینوس و کسینوس در بازه صفر تا دو پی با فاصله زمانی ۰.۲ پی بین هر دو نقطه رو رسم می‌کند.



حال اگر دستور **hold on** را برداریم، نمودار اول یا همان سینوس کشیده نمی‌شود و فقط کسینوس کشیده می‌شود منتها به شکل زیر:



(۲-۱) با استفاده از دستور subplot می‌توان نمودار را به چند قسمت تقسیم کرد و در هر کدام نموداری دلخواه رسم کرد:

```
t=0:0.01:1;
z1 = sin(2*pi*t);
z2 = cos(2*pi*t);

figure;
subplot(1, 2, 1)
plot(t, z1, '--b')
sin_x0 = [0.5];
sin_y0 = [0.2];
sin_s = ['sin(2 \pi t)'];
text(sin_x0, sin_y0, sin_s)
title('Sin')
legend('sin')
xlabel('time')
ylabel('amplitude')

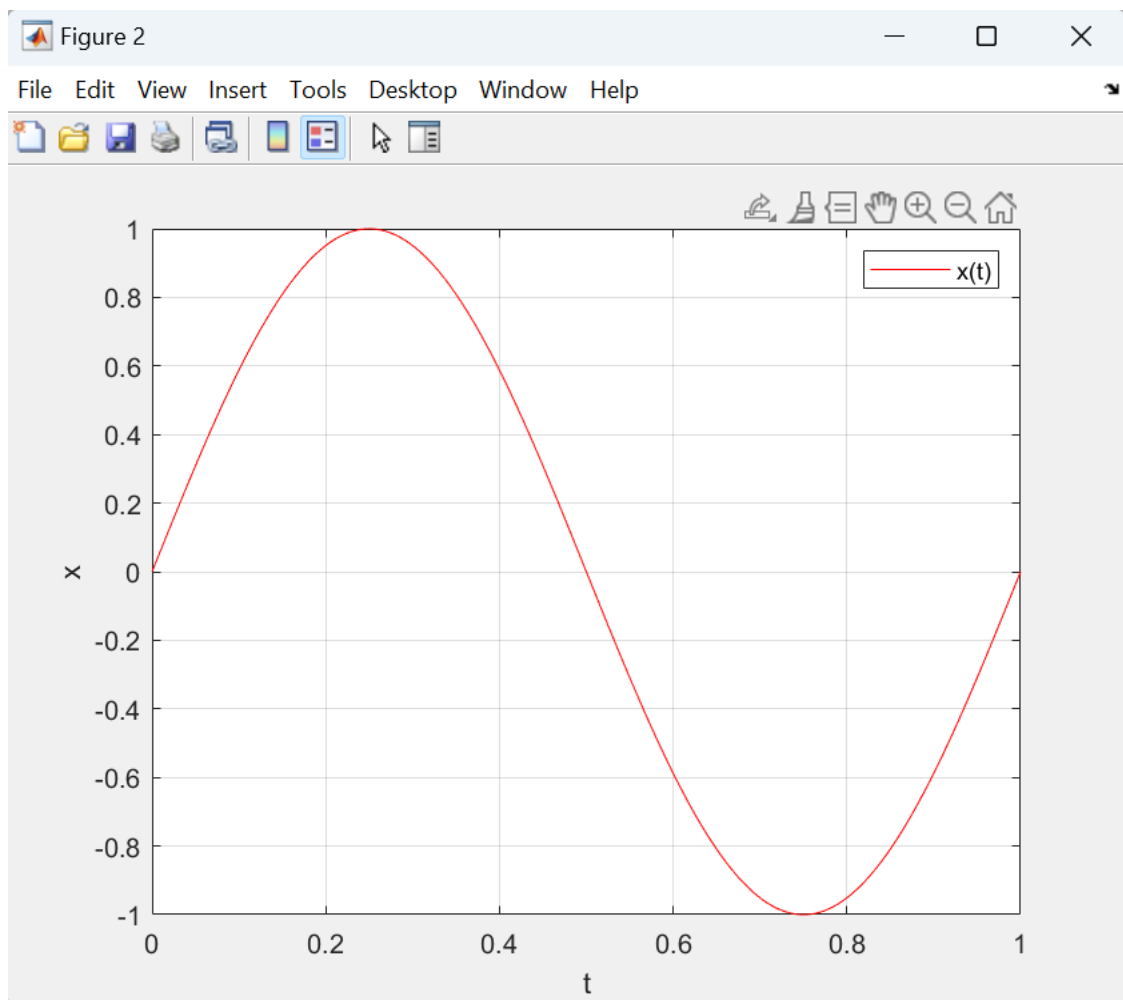
subplot(1, 2, 2)
plot(t, z2, 'r')
cos_x0 = [0.25];
cos_y0 = [-0.8];
cos_s = ['cos(2 \pi t)'];
text(cos_x0, cos_y0, cos_s); %Add Comment

title('Cos');
legend('cos')
xlabel('time')
ylabel('amplitude')
grid on
```

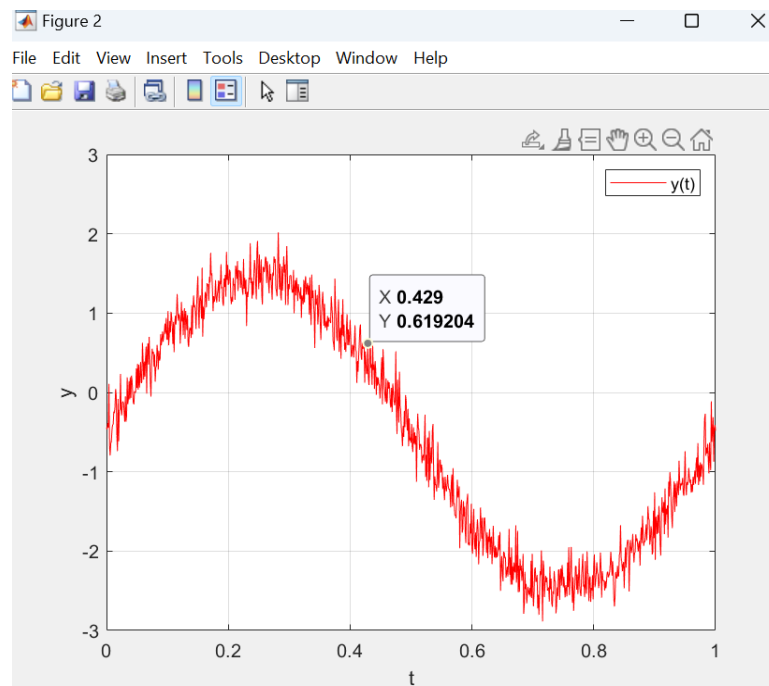
برحسب t به شکل زیر

(۱-۱) ورودی x

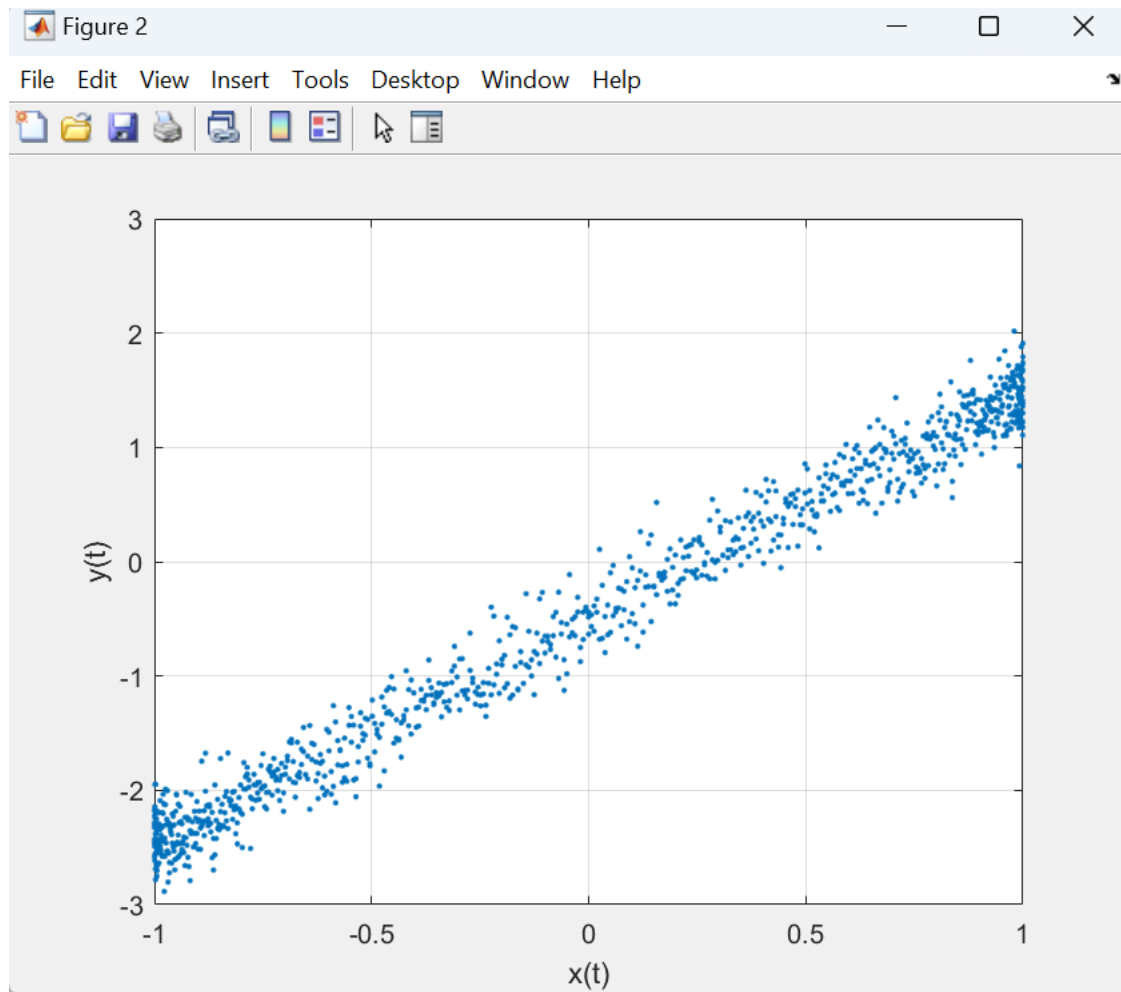
است:



۲-۲) خروجی y بر حسب t به شکل زیر است:



۲-۳) متغیر y بر حسب x به شکل زیر است:



که شیب آن مقدار α و عرض از مبدا آن مقدار β را مشخص می‌کند.

۴-۲) برای مینیمم کردن تابع $f(\alpha, \beta)$ یکبار از نسبت به α مشتق جزئی گرفته و برابر ۰ می‌گذاریم و یکبار دیگر از آن برحسب β مشتق جزئی گرفته و برابر ۰ می‌گذاریم. حال معادلات بدست آمده به این شکل است:

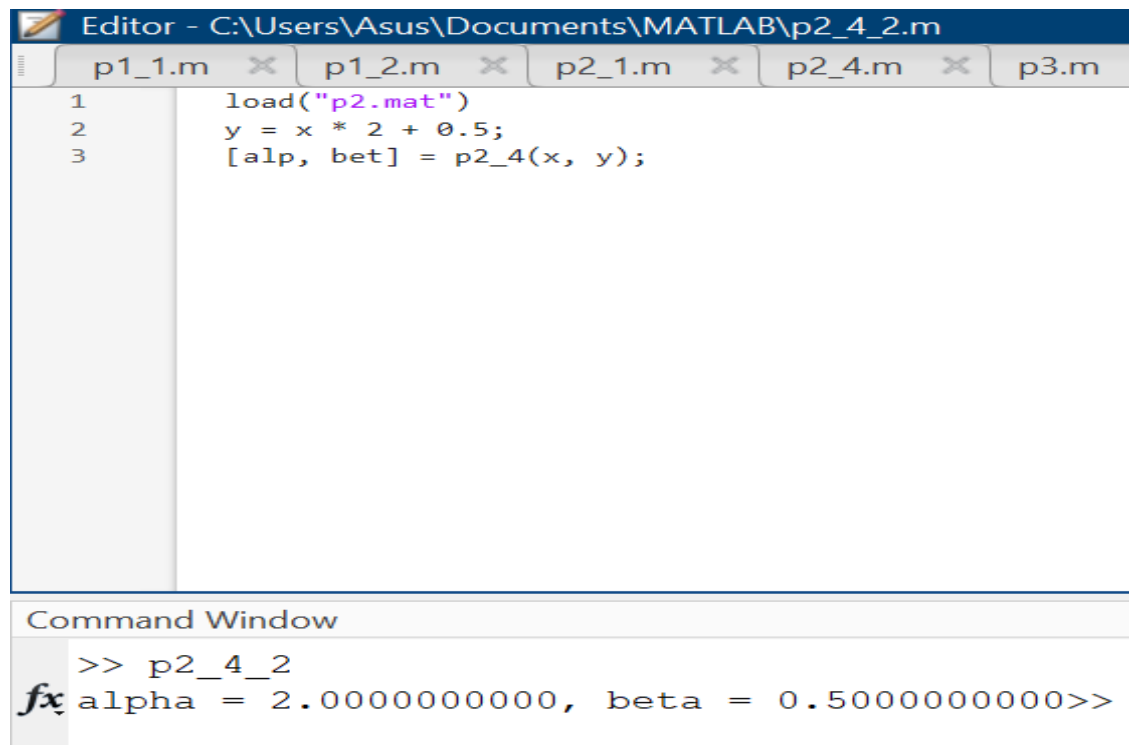
$$\begin{aligned}\sum -x(t)y(t) + \alpha(x(t))^2 + \beta x(t) &= 0 \\ \sum y(t) - \alpha x(t) - \beta &= 0\end{aligned}$$

که با حل معادله به مقادیر زیر می‌رسیم:

```
Command Window
New to MATLAB? See resources for Getting Started.

>> p2_4
fx alpha = 1.9735789512, beta = -0.4983386533>>
```

برای تست تابع ابتدا خودمان y را میسازیم و تست می‌کنیم:



The image shows the MATLAB Editor window with the file `p2_4_2.m` open. The code in the editor is as follows:

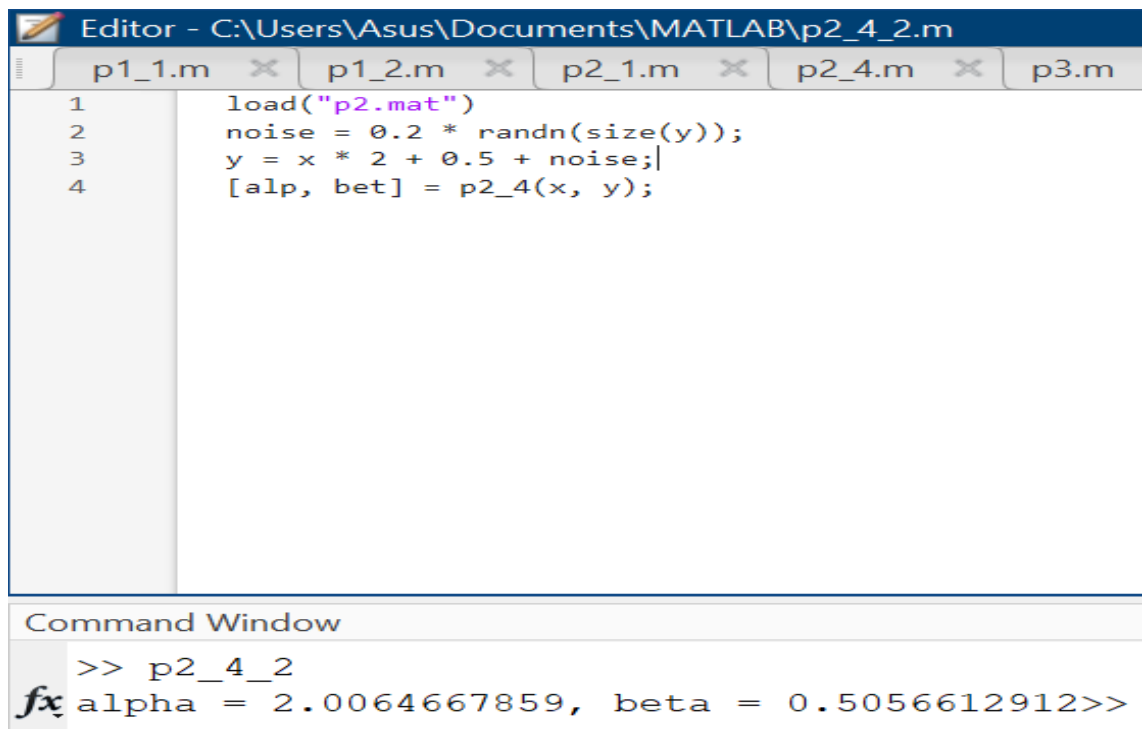
```
1 load("p2.mat")
2 y = x * 2 + 0.5;
3 [alp, bet] = p2_4(x, y);
```

Below the editor is the Command Window, which shows the execution of the script:

```
>> p2_4_2
fx alpha = 2.0000000000, beta = 0.5000000000>>
```

که دقیقا همان چیزیست که مشخص کرده‌ایم.

مثالی دیگر برای اطمینان درستی تابع به این صورت است که y را خودمان به صورت $2 * x + 5$ تولید کردیم، حال به آن نویزی اضافه می‌کنیم و به تابع می‌دهیم که خروجی به صورت زیر شد:



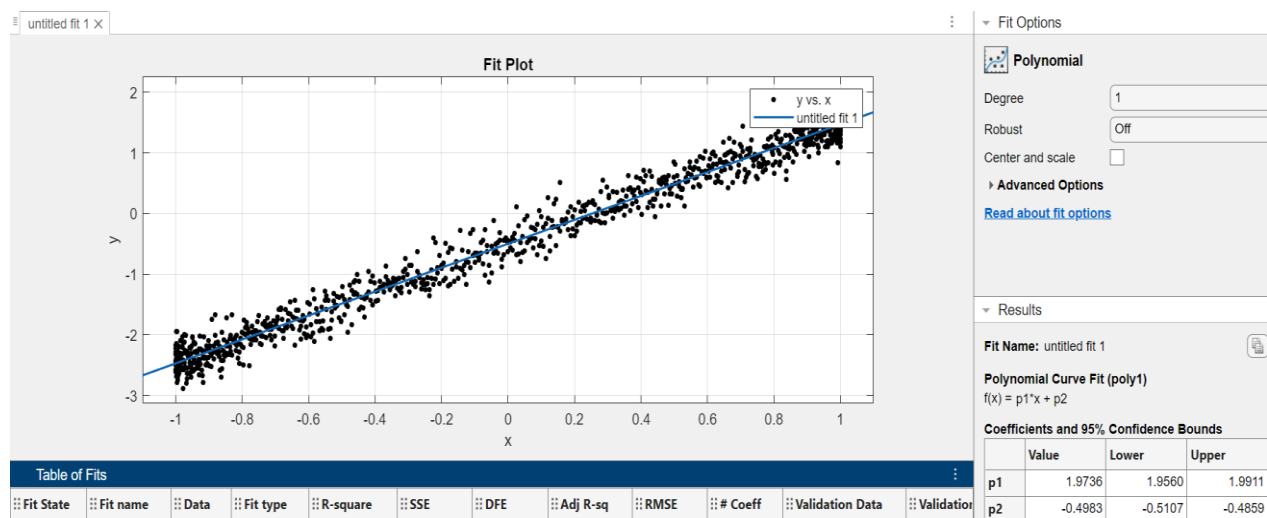
The image shows the MATLAB Editor window with the file `p2_4_2.m` open. The code in the editor is as follows:

```
1 load("p2.mat")
2 noise = 0.2 * randn(size(y));
3 y = x * 2 + 0.5 + noise;
4 [alp, bet] = p2_4(x, y);
```

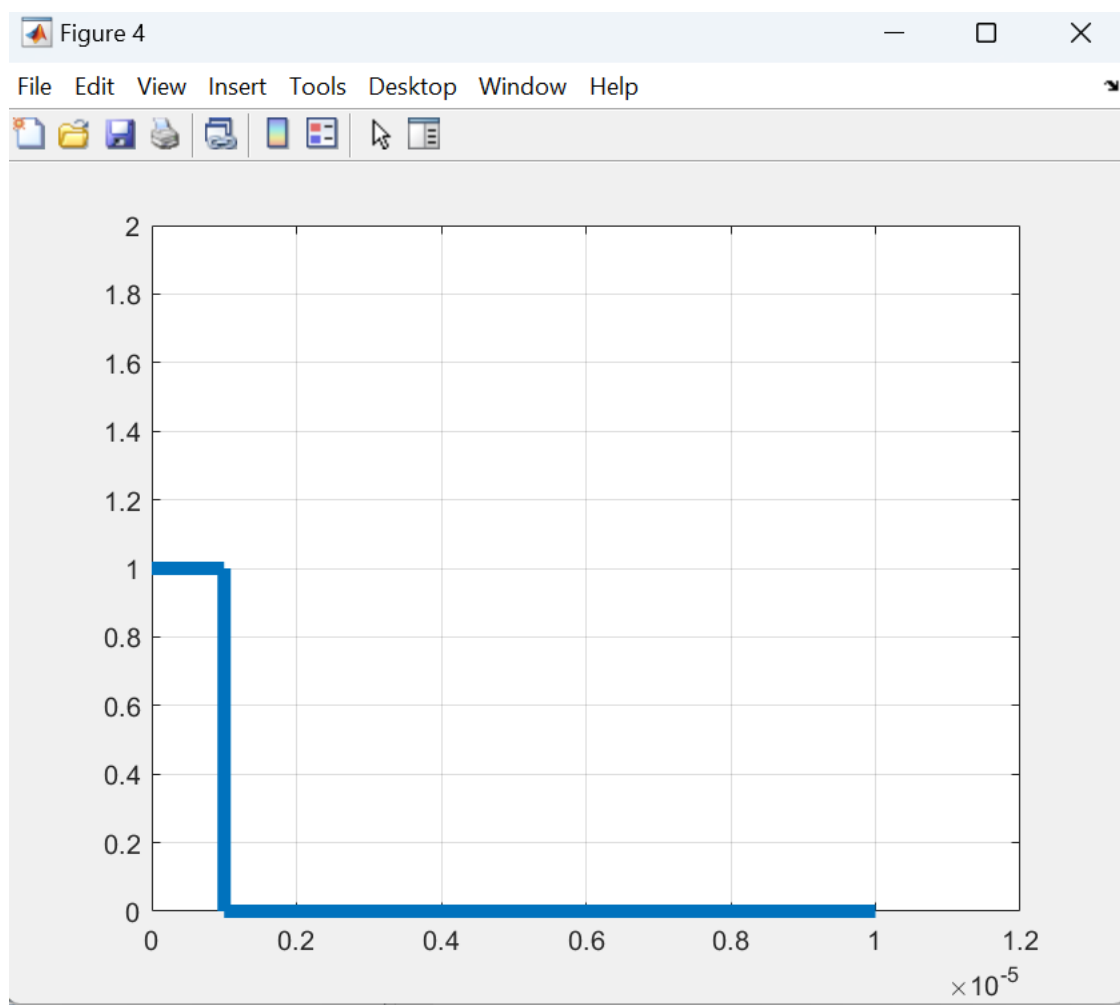
Below the editor is the Command Window, which shows the execution of the script:

```
>> p2_4_2
fx alpha = 2.0064667859, beta = 0.5056612912>>
```

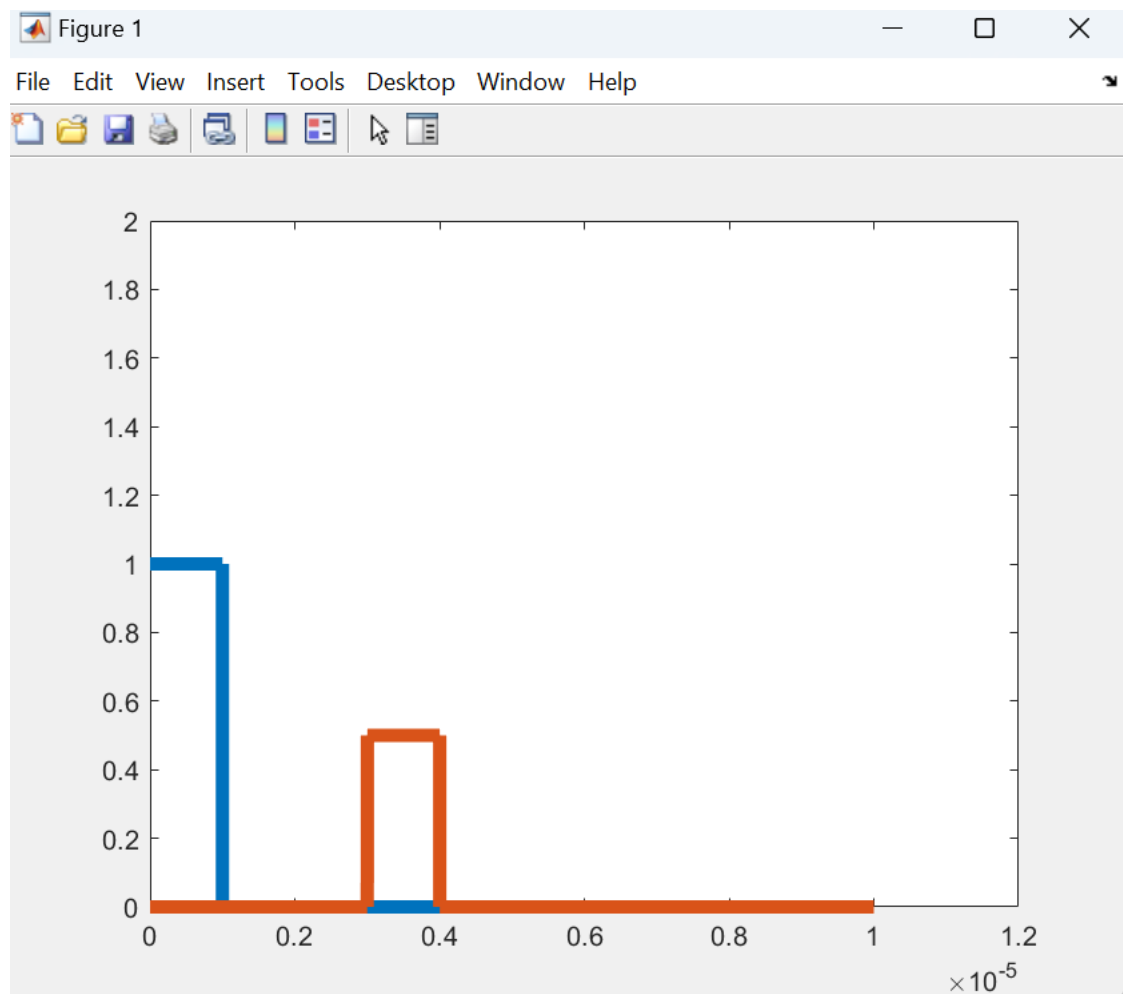
۵-۲) نتایج بدست آمده به صورت زیر است که تقریباً برابر همان مقداری است که در بخش قبل بدست آوردیم:



۱-۳) سیگنال ارسالی به شکل زیر است:



۲-۳) سیگنال ارسالی به شکل زیر است:



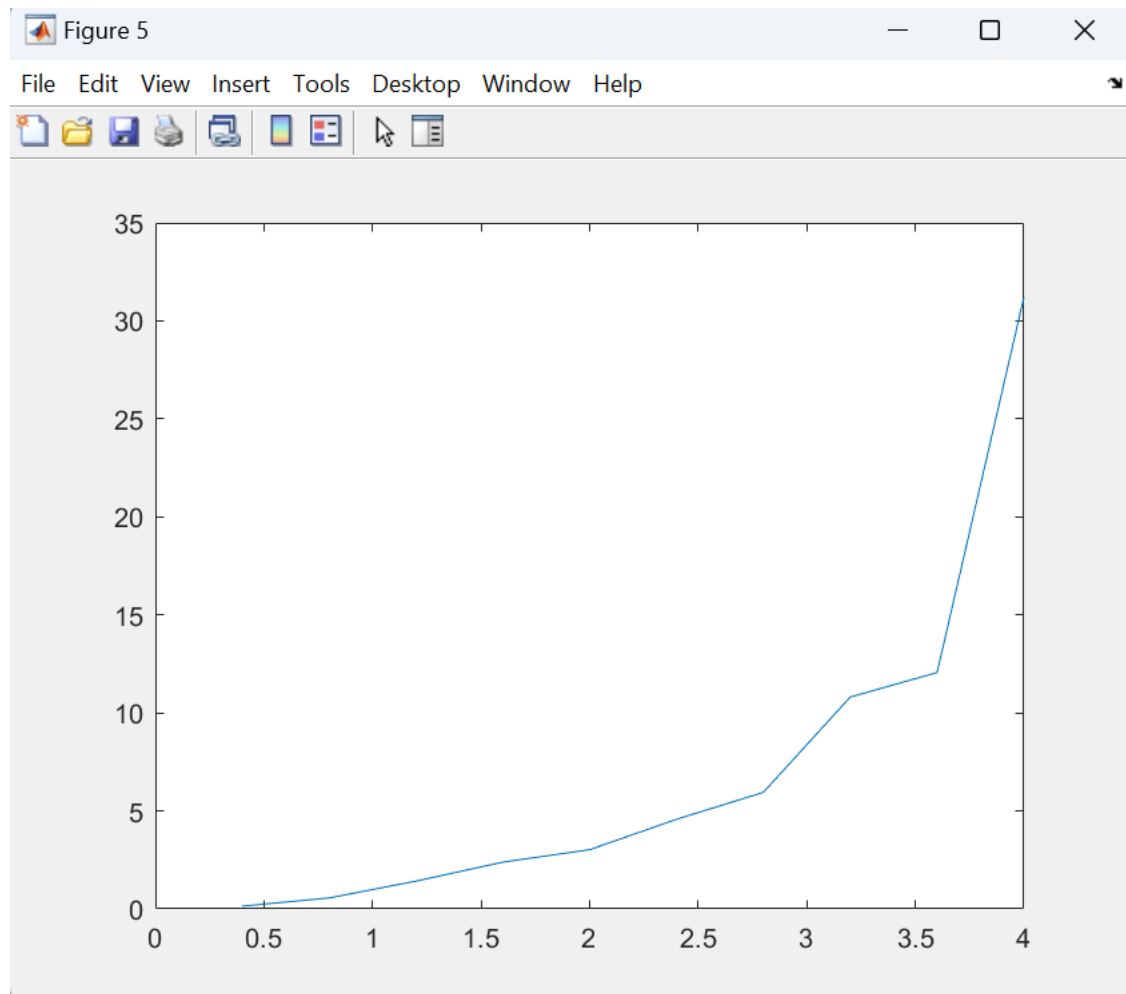
۳-۳) با استفاده ضرب داخلی و نقطه ماکسیمم آن، td بدست می‌آید و به کمک رابطه R به شکل زیر بدست می‌آید:

```

Command Window
New to MATLAB? See resources for Getting Started.

my_R =
    449.9885
  
```

۳-۴) ضریب نویز را از ۰.۴ تا ۴ به گام‌های ۰.۴ اضافه شده و نتایج زیر بدست آمده است که نشان می‌دهد تا تقریباً ضریب ۳ تخمین خوبی از R بدست می‌آید:



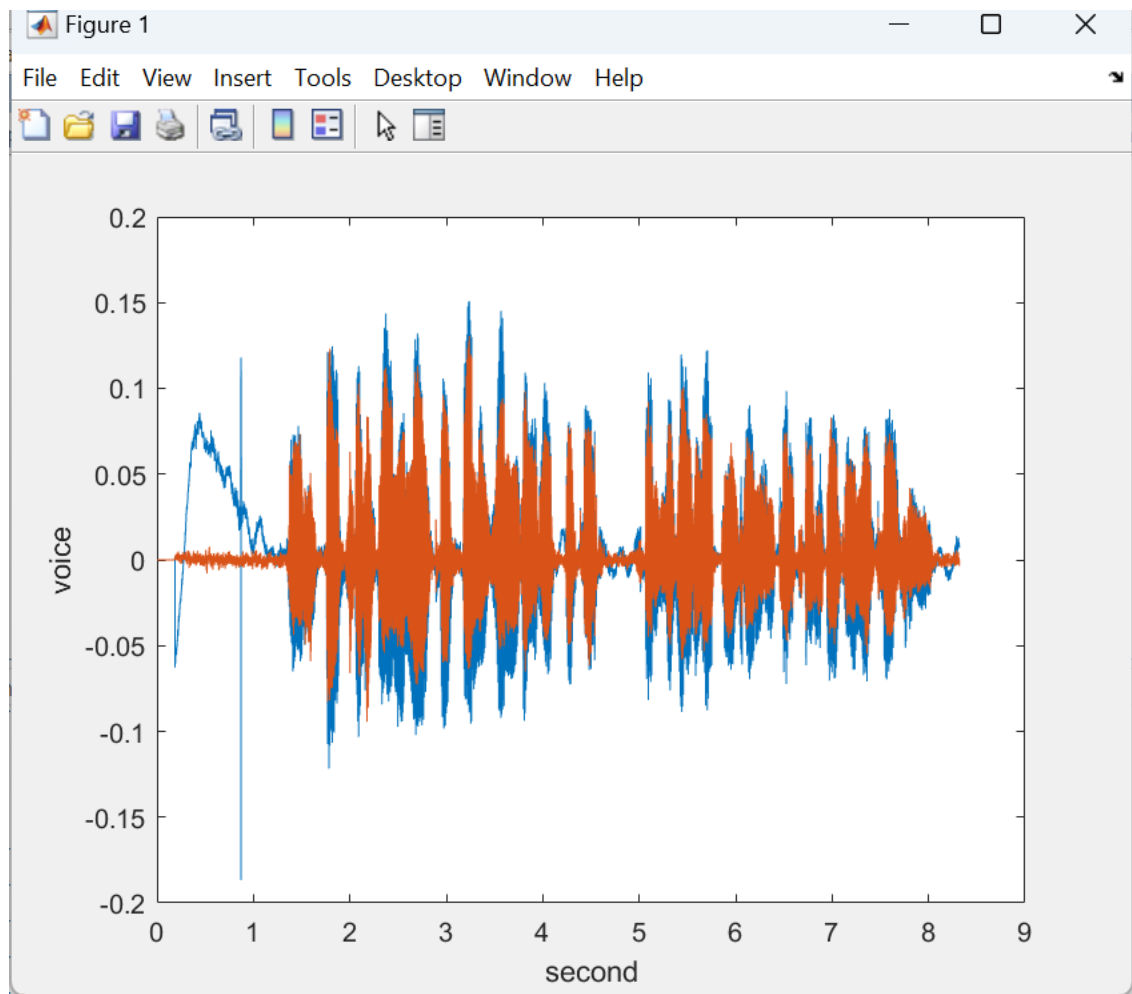
۴-۱) فرکانس نمونه برداری فایل صوتی به صورت زیر است:

$f_s =$

48000

>>

۴-۲) نمودار سیگنال فایل صوتی مطابق جدول زیر است:



۳-۴) در این بخش برای دو برابر کردن، داده‌های زوج را دور ریخته و برای ۰.۵ برابر کردن، سیگنال را یکی در میان با مقادیر سیگنال \times جایگزاری کرده و نقاط خالی را میانگین دو خانه مجاور قرار می‌دهیم.

۴-۴) در این بخش برای سرعت‌های کمتر از ۱، به ازای هر سیگنال آن را $1/\text{speed}$ برابر می‌کند و برای بیشتر از ۱، به ازای هر $10 * \text{speed}$ سیگنال، آن را ۱۰ سیگنال می‌کند.