به نام خدا

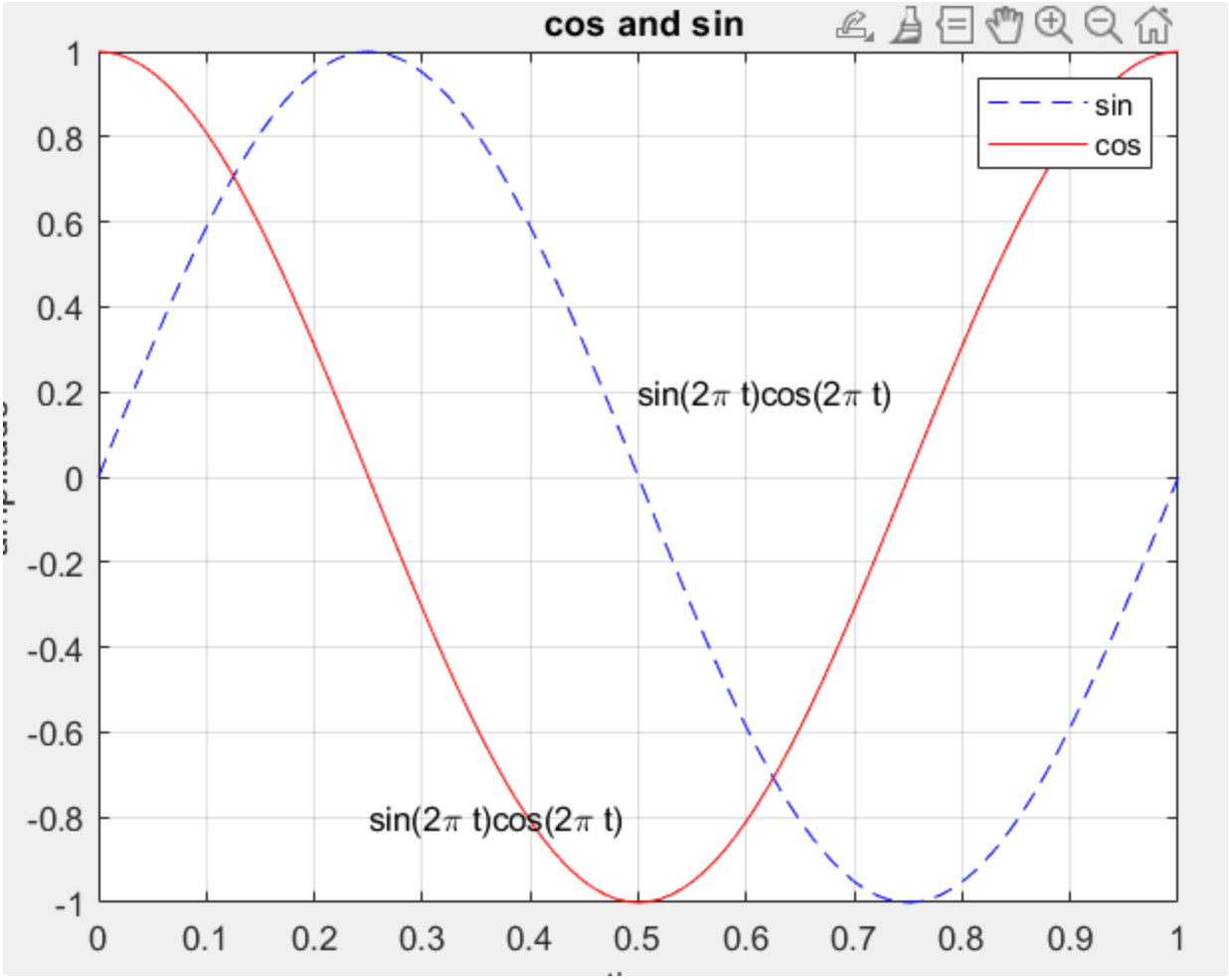
پارسا دقیق

۸۱۰۱۰۱۴۱۹

گزارش کار پروژه اول

بخش اول:

تمرین ۱-۱)



این کد متلب یک نمودار ساده از توابع سینوس و کسینوس ترسیم می‌کند و چندین ویژگی تزئینی به آن اضافه می‌کند.

```
t = 0:0.01:1;
```

این خط یک بردار زمانی از ۰ تا ۱ با گام ۰٫۰۱ ایجاد می‌کند. این بردار به عنوان محور x نمودار استفاده می‌شود.

```
z1 = sin(2*pi*t);
```

```
z2 = cos(2*pi*t);
```

در این دو خط، مقادیر توابع سینوس و کسینوس برای هر مقدار از بردار زمان محاسبه و در متغیرهای z1 و z2 ذخیره می‌شود.

```
figure;
```

این خط یک پنجره جدید برای نمایش نمودار ایجاد می‌کند.

```
plot(t, z1, '--b');
```

این خط نمودار تابع سینوس را رسم می‌کند.

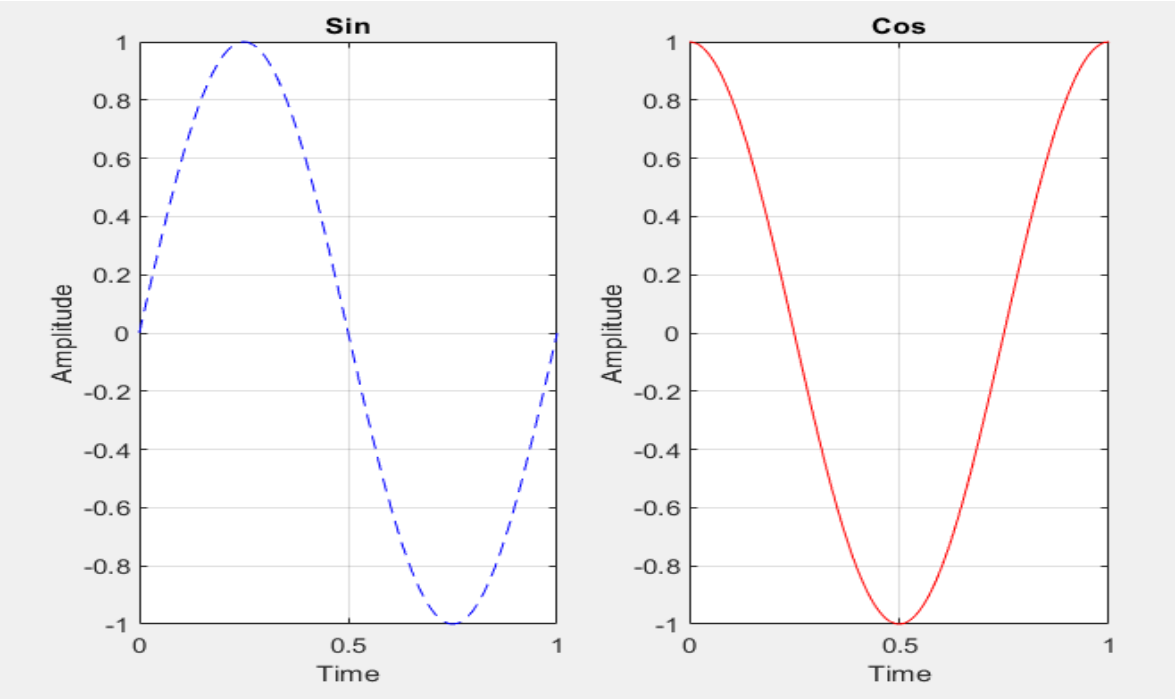
```
hold on;
```

این دستور بسیار مهم است. با استفاده از آن، نمودار فعلی حفظ می‌شود تا بتوان نمودارهای بعدی را بر روی آن اضافه کرد. اگر این دستور را حذف کنیم، هر بار که یک دستور `plot` جدید اجرا می‌شود، نمودار قبلی پاک می‌شود و فقط نمودار آخر نمایش داده می‌شود.

```
plot(t, z2, 'r');
```

این خط نمودار تابع کسینوس را به عنوان یک خط قرمز رنگ بر روی نمودار قبلی اضافه می‌کند.

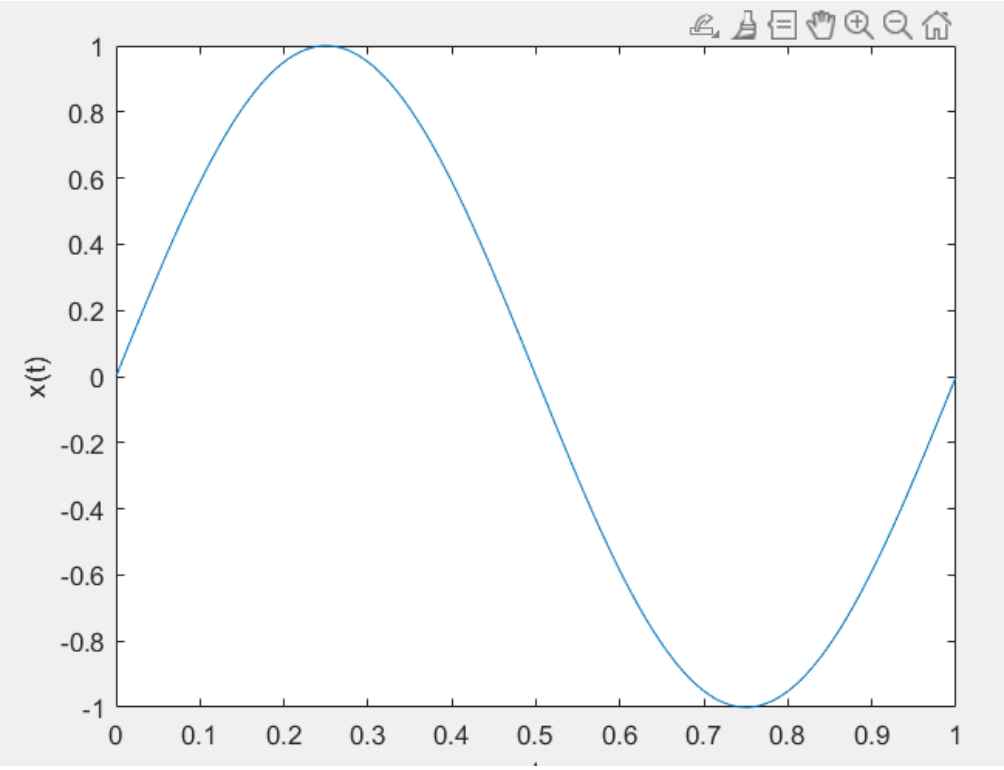
تمرین ۲-۱)



دستور subplot(1,2,1) به این معنی است که یک صفحه یک در دو از رسم باز شود و اول میخواهیم نمودار اول را رسم کنیم.

بخش دوم:

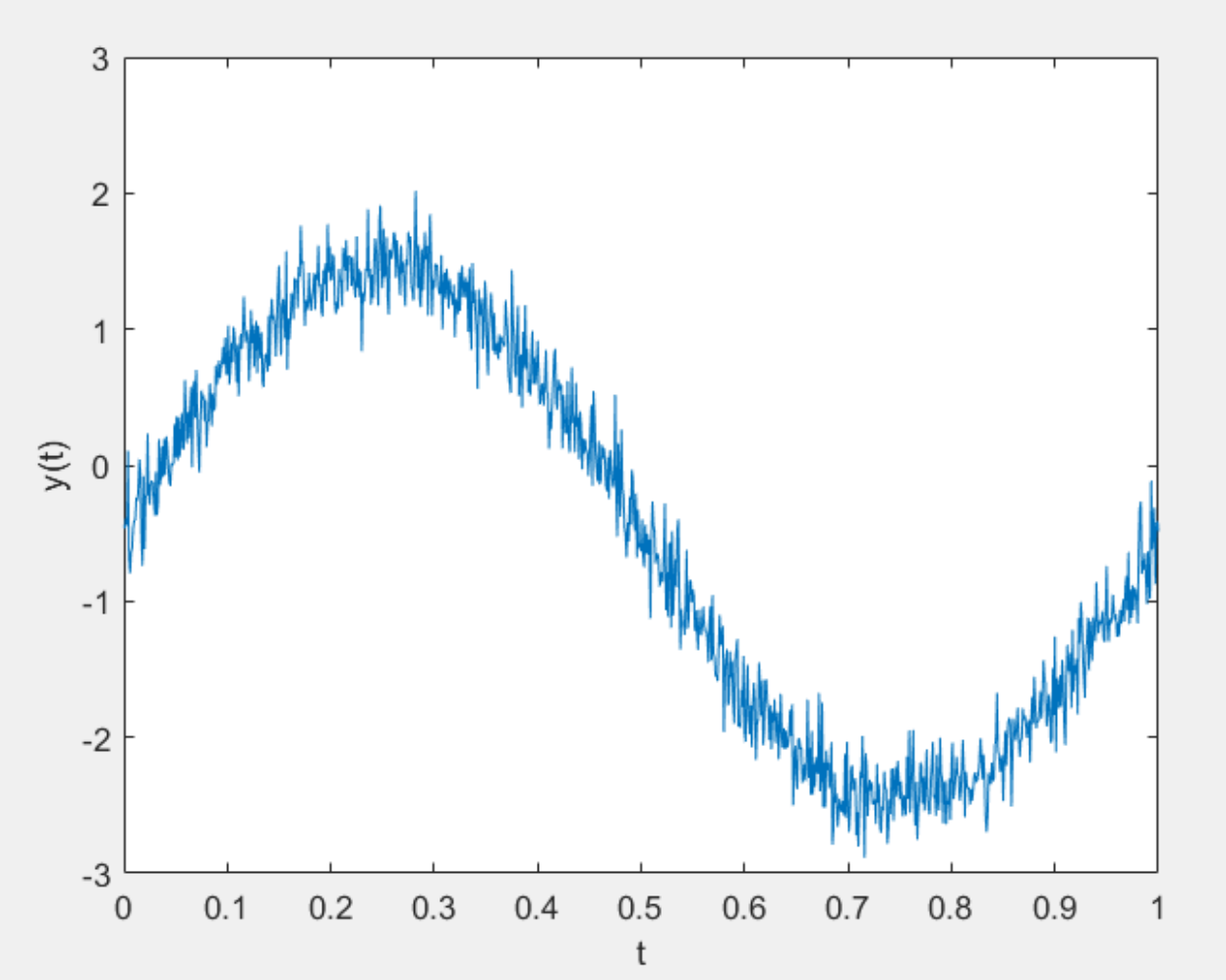
تمرین ۱-۲)



با دستور load متغیرها را بارگذاری می‌کنیم.

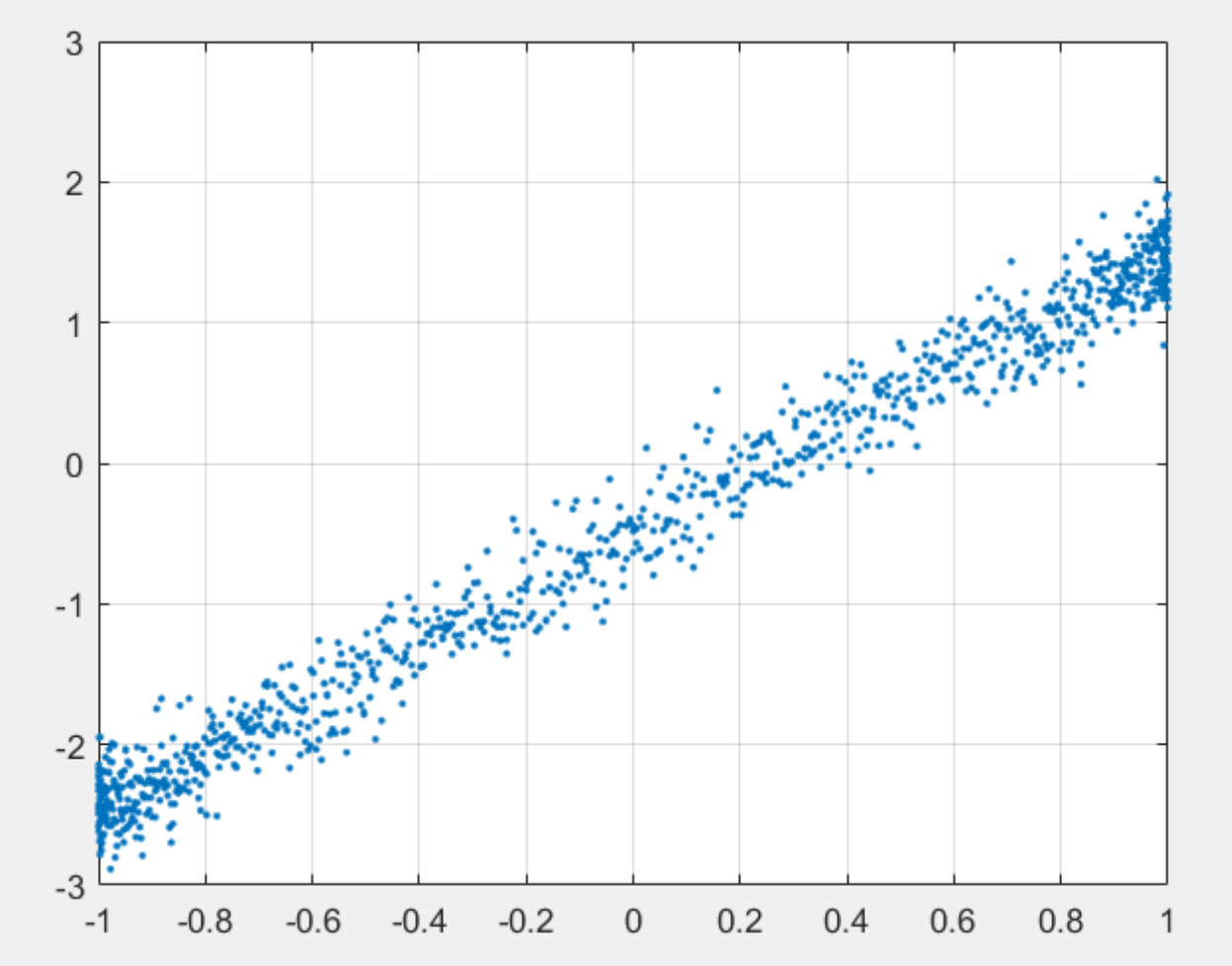
با دستور plot شکل را کشیده و سپس محورها را نام‌گذاری می‌کنیم.

تمرین ۲-۲)



کد مانند قسمت قبل است فقط داده ما نویز هم دارد.

تمرین ۳-۲)



در این قسمت مشاهده می‌کنیم که بین x و y یک رابطه خطی برقرار است. به عبارتی y=mx+h که h عرض از مبدا و m شیب است.

تمرین ۴-۲)

تابع زیر را می‌نیموم میکنیم با استفاده از قواعد مشتق.

$S = \Sigma(y_i - mx_i - h)^2$

$\partial S/\partial m = 0$

$\partial S/\partial h = 0$

$\partial S/\partial m = -2\Sigma x_i(y_i - mx_i - h) = 0$

$\partial S/\partial h = -2\Sigma(y_i - mx_i - h) = 0$

$\Sigma x_i(y_i - mx_i - h) = 0$

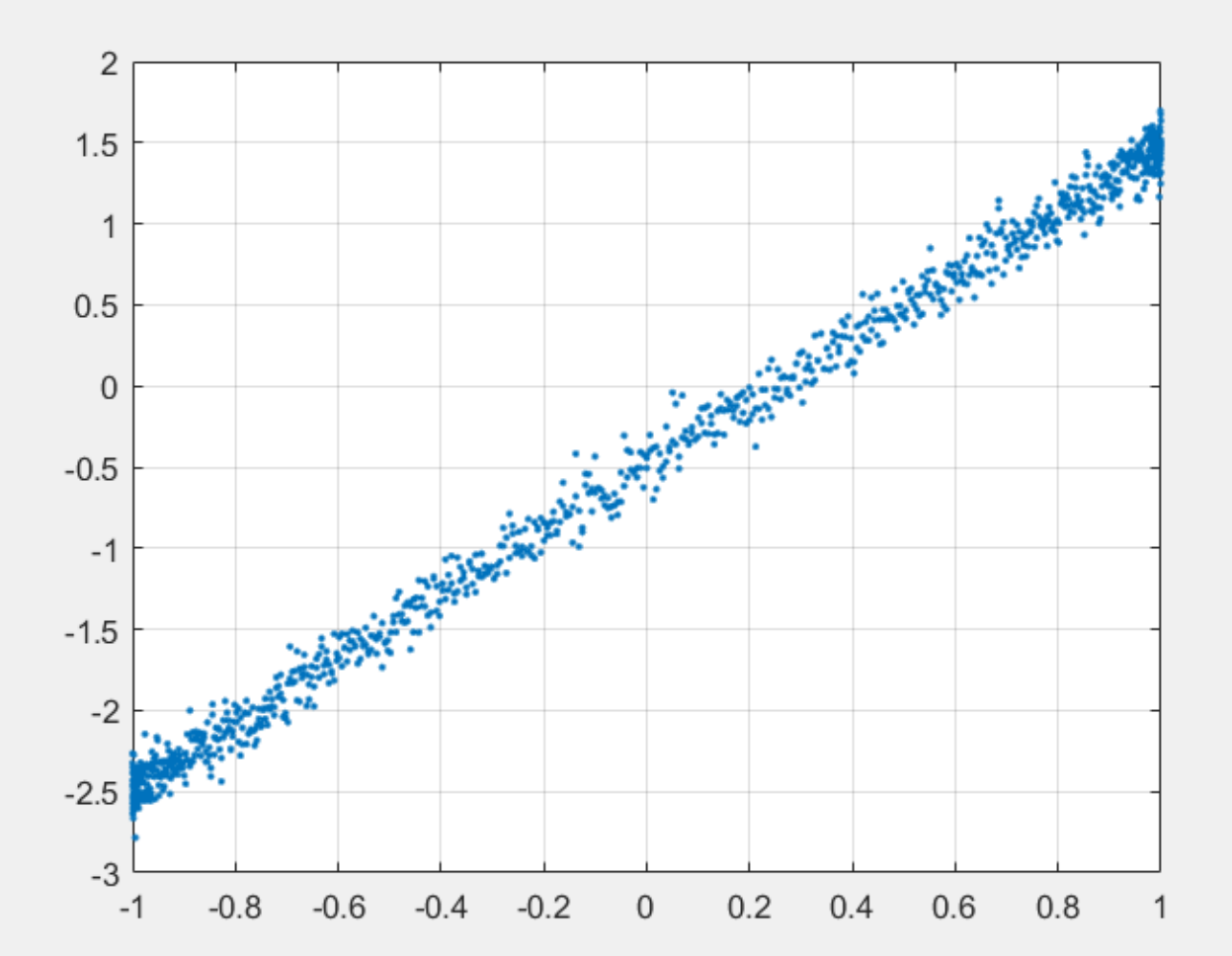
$\Sigma(y_i - mx_i - h) = 0$

$$\mathbf{m = \Sigma[(x_i - \bar{x})(y_i - \bar{y})] / \Sigma(x_i - \bar{x})^2}$$

$$\mathbf{h = \bar{y} - m\bar{x}}$$

با توجه به فرمول بالا شیب و عرض از مبدا را محاسبه کرده ایم.

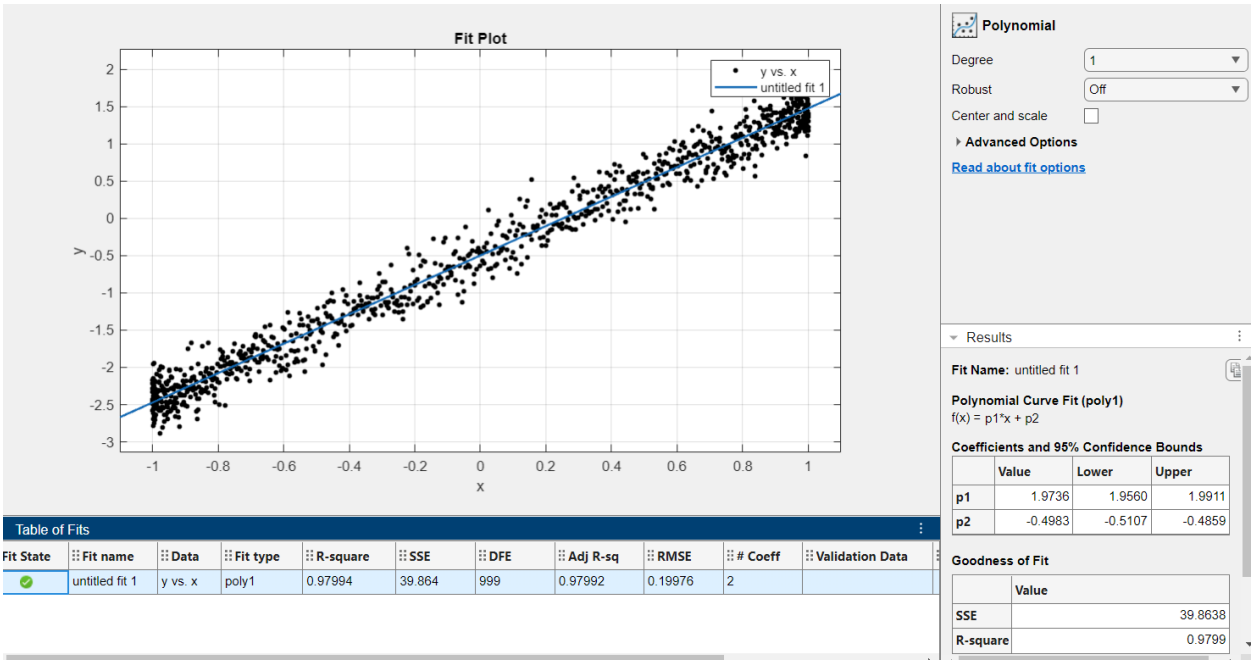
در شکل زیر هم کلیت داده را کشیده ایم که بسیار شبیه داده واقعی است. باید دقت کنیم که داده نویزی ما از نوع نویز سفید از نوع گوسی است.



Name	Value
h	1.9736
i	1001
m	-0.4983
new_y	*1x1001 double*
noise	*1x1001 double*
t	*1x1001 double*
x	*1x1001 double*
y	*1x1001 double*

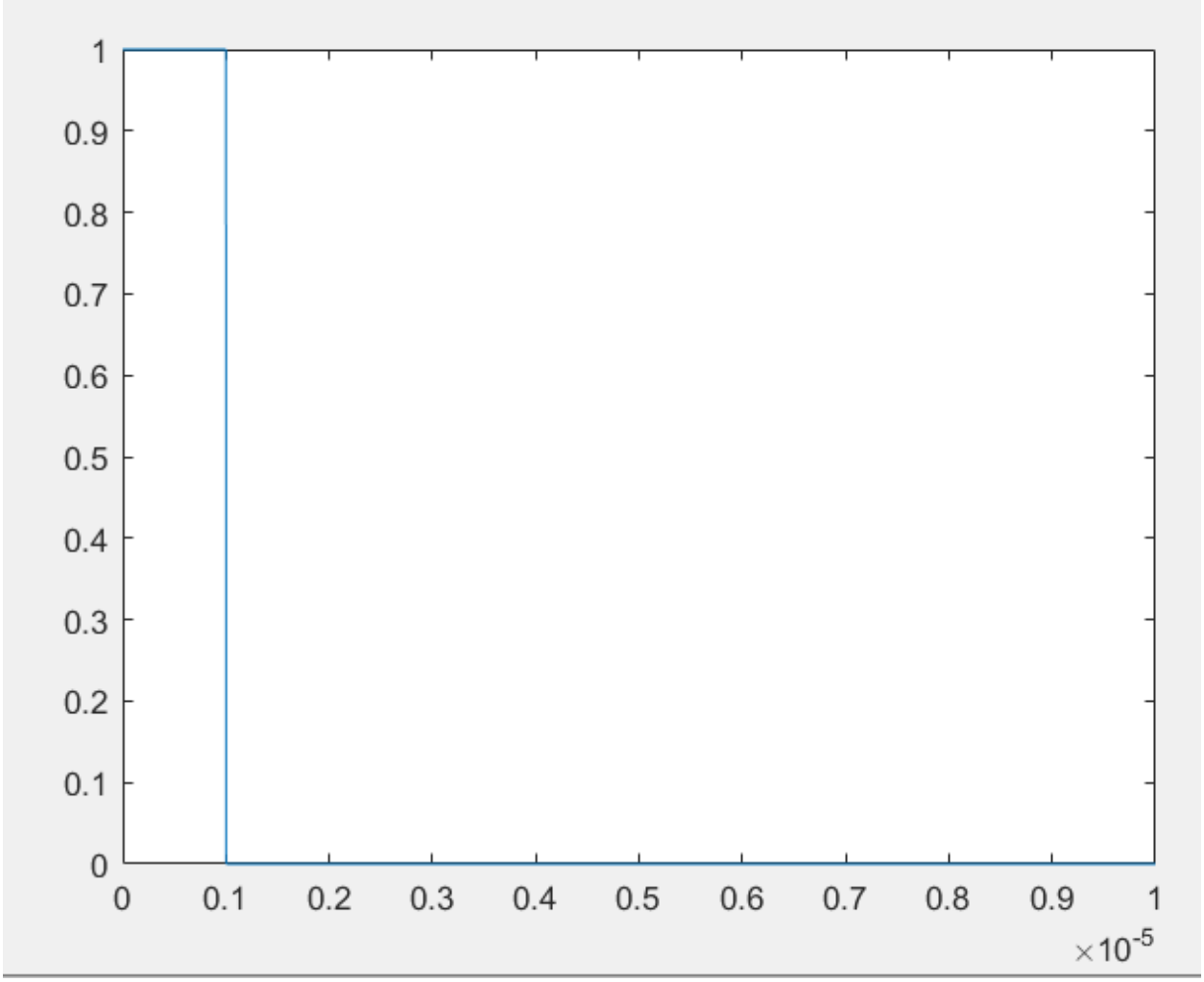
تمرین ۵-۲)

با توجه به شکل می‌بینیم که curve fitter به ما m=1.9763 و h=-0.4983 را محاسبه کرده که بسیار به اعداد ما نزدیک است.



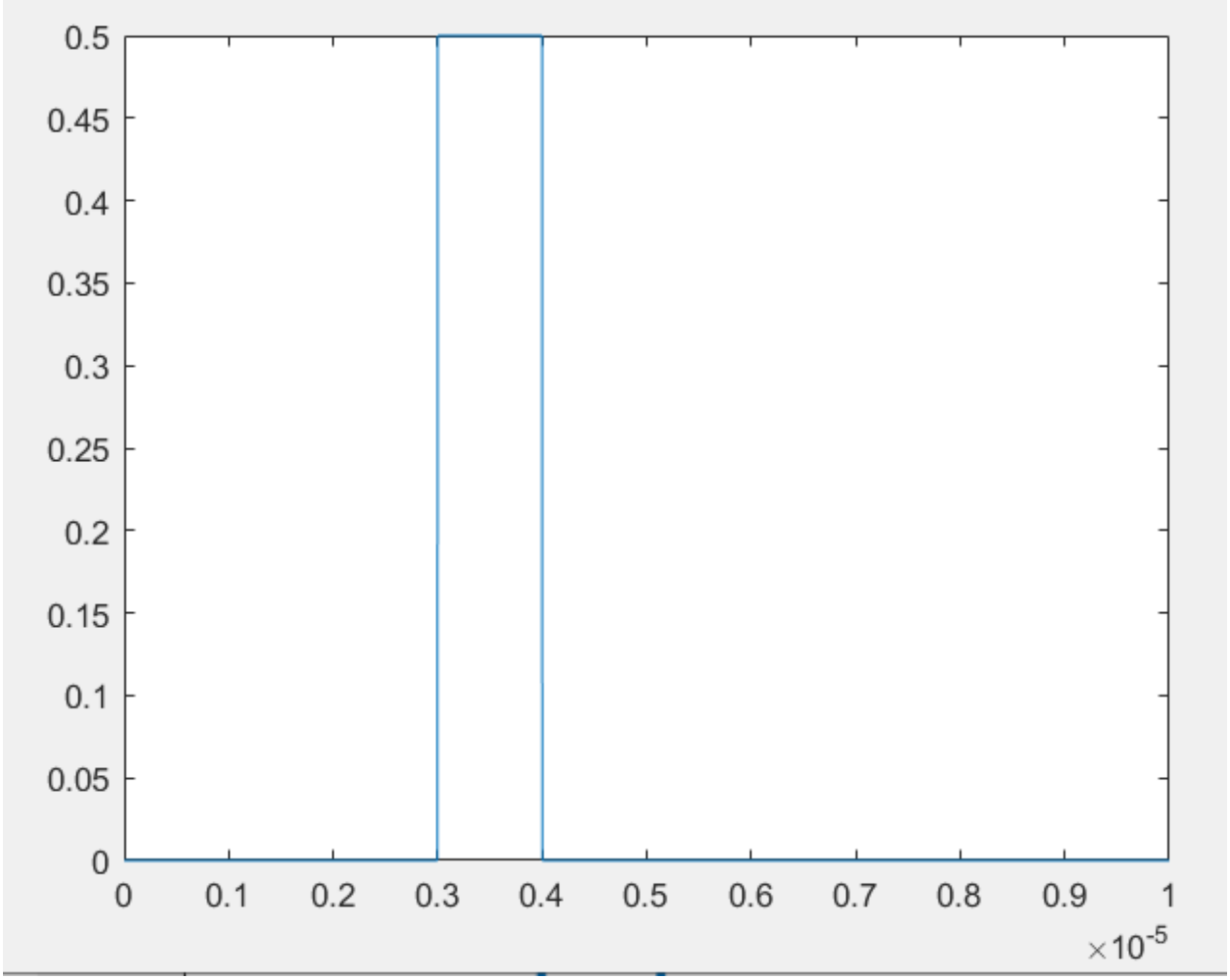
بخش سوم:

تمرین ۱-۳)



ما 1e4 تا نقطه داریم. و 0.1 داده های ما مقدار ۱ می‌گیرند.

تمرین ۲-۳)



مقدار از td تا td+tau برابر ۱ و بقیه جاها صفر است.

تمرین ۳-۳)

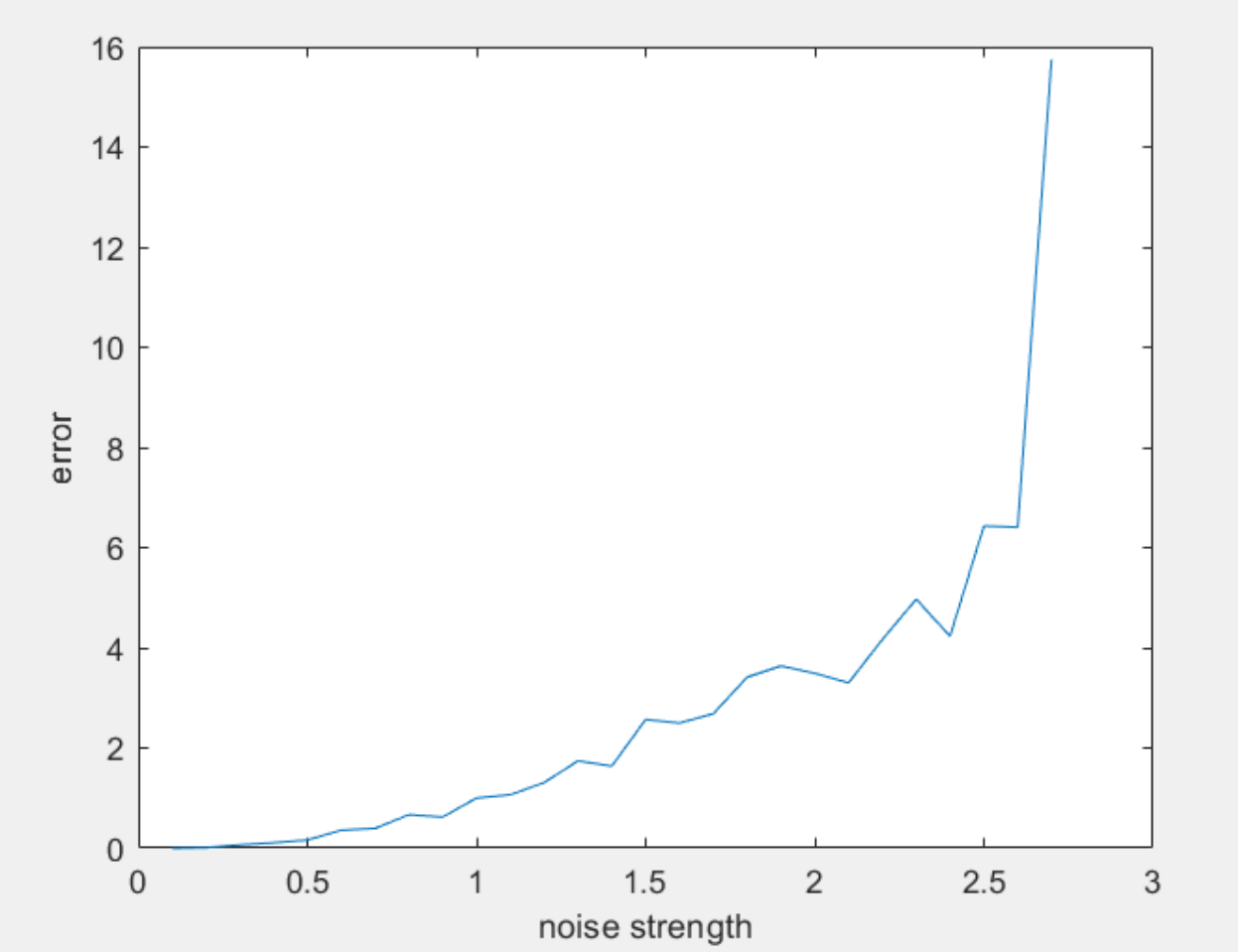
نحوه کار به ترتیب امده است:

- **ایجاد سیگنال‌ها :**دو سیگنال، یکی برای فرستاده شده و دیگری برای دریافتی، ساخته می‌شوند .
- **همبستگی :**این دو سیگنال با هم مقایسه می‌شوند تا بیشترین شباهت بین آن‌ها پیدا شود. جایی که بیشترین شباهت وجود دارد، نشان‌دهنده زمانی است که سیگنال برگشته است .
- **محاسبه تأخیر :**با دانستن زمان رفت و برگشت سیگنال، می‌توان فاصله تا هدف را محاسبه کرد.

pulse_length	1000
R	450

با توجه به محاسبات مقدار R محاسبه شد.

تمرین ٤-٣)



به طور خلاصه و تیتروار:

- **ایجاد سیگنال‌ها :**یک سیگنال پالس و یک سیگنال پالس با تأخیر ایجاد می‌شود .
- **اضافه کردن نویز :**نویز به سیگنال دریافتی اضافه می‌شود .
- **همبستگی :** همبستگی بین سیگنال فرستاده شده و دریافتی محاسبه می‌شود تا تأخیر زمانی پیدا شود .
- **محاسبه فاصله :**تأخیر زمانی به فاصله تبدیل می‌شود .
- **محاسبه خطا :**خطا بین فاصله محاسبه شده و فاصله واقعی محاسبه می‌شود .
- **تکرار و رسم نمودار :**این مراحل برای سطوح مختلف نویز تکرار می‌شوند و نمودار خطا به ازای قدرت نویز رسم می‌شود.

بخش چهارم:

تمرین ۱-۴)

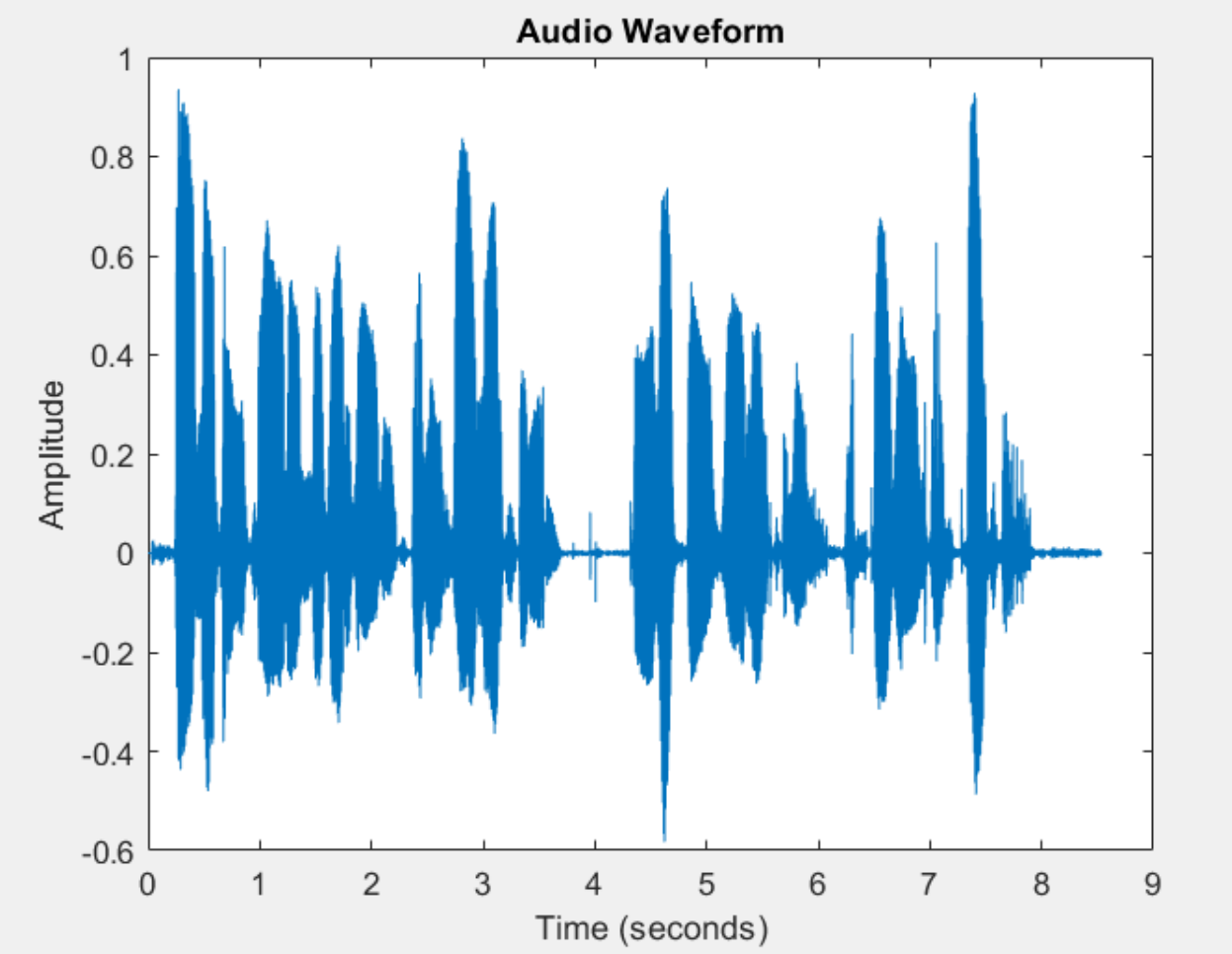
با توجه به کد زیر ابتدا با audioread شعر را در x ذخیره می‌کنیم. همچنین fs همان فرکانس نمونه‌برداری است. با soundهم میتوانیم چک کنیم و فایل را پلی کنیم.

```
[x, fs] = audioread('poem.wav');
sound(x,fs)
```

Workspace

Name	Value
fs	48000
x	*409600x1 double*

تمرین ۲-۴)



- دستور `audioread` فایل صوتی WAV را می‌خواند و داده‌های صوتی را در متغیر `x` و فرکانس نمونه‌برداری را در متغیر `fs` ذخیره می‌کند.
- با استفاده از فرکانس نمونه‌برداری، یک بردار زمانی `t` ایجاد می‌شود. هر عنصر در بردار `t` به یک نمونه در داده‌های صوتی `x` مربوط می‌شود.
- دستور `plot(t, x)` داده‌های صوتی (`x`) را بر حسب بردار زمان (`t`) رسم می‌کند.
- برچسب‌های محور `x` (زمان بر حسب ثانیه) و محور `y` (دامنه) اضافه می‌شوند.
- یک عنوان برای نمودار در نظر گرفته می‌شود.

تمرین ۳-۴)

توضیح کد:

1. **تعریف پارامترها:**
   - فرکانس نمونه‌برداری سیگنال صوتی را به ۴۸۰۰۰ هرتز تنظیم می‌کند.
   - طول سیگنال ورودی را محاسبه می‌کند.
2. **کنترل سرعت:**
   - **سرعت دو برابر :**
     - یک آرایه جدید با نصف طول آرایه ورودی ایجاد می‌کند.
     - هر عنصر دوم آرایه ورودی را در آرایه جدید کپی می‌کند.
     - در واقع، هر نمونه دوم سیگنال اصلی حذف می‌شود تا سرعت پخش دو برابر شود.
   - **سرعت نصف :**
     - یک آرایه جدید با دو برابر طول آرایه ورودی ایجاد می‌کند.
     - هر عنصر آرایه ورودی را در ردیف‌های زوج آرایه جدید کپی می‌کند.
     - ردیف‌های فرد آرایه جدید را با میانگین گیری از دو ردیف مجاور در آرایه ورودی پر می‌کند. این کار باعث می‌شود سیگنال کشیده شود و سرعت پخش نصف شود.
3. **پخش صدا:**
   - پس از اعمال تغییرات بر روی سیگنال، از تابع `sound` برای پخش سیگنال اصلاح شده استفاده می‌شود.

تمرین ۴-۴)

```
function p4_4(x, speed)
    if speed <= 0
        error('Speed must be greater than 0.');
    end
    fs =  48000;
    new_fs = speed * fs;
    x_resampled = resample(x, fs, new_fs);
    sound(x_resampled, new_fs);
end
```

با استفاده از تابع resample میتوانیم هر سرعتی پلی کنیم. مانند تست زیر که 0.8 و 1.8 سرعت مورد نظر است.

```
[x, fs] = audioread('poem.wav');
p4_4(x, 1.8);
p4_4(x, 0.8);
```